เอกสารประกอบการสอน

รหัสวิชา 04-000-104

ชื่อวิชา การโปรแกรมคอมพิวเตอร์

(Computer Programming)

พงศกร เจริญเนตรกุล

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย

ปีการศึกษา 2566

คำนำ

เอกสารประกอบการสอน รหัสวิชา 04-000-104 รายวิชาการโปรแกรมคอมพิวเตอร์ (Computer Programming) มีวัตถุประสงค์เพื่อใช้ในการเรียนการสอนของนักศึกษาระดับปริญญาตรี หลักสูตรวิศวกรรมศาสตรบัณฑิต ในสาขาวิชาวิศวกรรมต่างๆ ของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย ซึ่งอยู่ในกลุ่มวิชาพื้นฐานทางวิศวกรรม ดำเนินการสอนโดยสาขาวิศวกรรมคอมพิวเตอร์ โดยได้จัดแบ่งสาระสำคัญของเอกสารประกอบการสอนประกอบด้วย แผนบริหารการสอนประจำรายวิชา แผนการสอน หัวข้อบรรยายเนื้อหารายวิชา แบบฝึกหัดท้ายบท ใบฝึกปฏิบัติการ และบรรณานุกรม เอกสารประกอบการสอนนี้มีเนื้อหาสาระเกี่ยวกับ ระบบคอมพิวเตอร์ วิธีการออกแบบและพัฒนาโปรแกรม พื้นฐานการเขียนโปรแกรม เงื่อนไขและการตัดสินใจ โครงสร้างการวนรอบ ฟังก์ชัน อาร์เรย์ การประยุกต์ใช้โปรแกรมสำเร็จรูปทางวิศวกรรม Scilab

ผู้จัดทำได้รับคำแนะนำและการสนับสนุนอย่างดีจากคณบดีคณะวิศวกรรมศาสตร์ รศ.สุรสิทธิ์ ระวังวงศ์ พร้อมทั้ง ผศ.นราธร สังข์ประเสริฐ อ.ณัฐพล หนูฤทธิ ดร.เกสรา เพชรกระจ่าง ผศ.ปิยะพร มูลทองชุน อ.นงนาถ ระวังวงศ์ และคณาจารย์ผู้ร่วมงานในและนอกมหาวิทยาลัยและโดยเฉพาะอย่างยิ่งเจ้าของหนังสือ ตำรา หรือเอกสารใด ๆ ที่ใช้ในการอ้างอิงในเอกสารประกอบการสอนนี้สำเร็จลุล่วงไปด้วยดี สุดท้ายนี้ผู้จัดทำหวังเป็นอย่างยิ่งว่าเอกสารประกอบการสอนเล่มนี้ จะก่อให้เกิดประโยชน์ในการเรียนการสอนได้อย่างมีประสิทธิภาพ เป็นประโยชน์ต่อนักศึกษา แต่หากมีข้อบกพร่องประการใดยินดีรับข้อเสนอแนะเพื่อการแก้ไข ปรับปรุงต่อไป จักขอบพระคุณเป็นอย่างยิ่ง

พงศกร เจริญเนตรกุล

สารบัญ

		หน้า
คำนำ		(1)
สารบัญ		(2)
แผนบริหารการสอนประจำรายวิชา		(12)
แผนการสอนเรื่อง ระบบคอมพิวเตอร์		1
บทที่ 1	หัวข้อบรรยายเรื่อง ระบบคอมพิวเตอร์	4
	1.1 นิยามของคอมพิวเตอร์	4
	1.2 วิวัฒนาการของคอมพิวเตอร์	6
	1.3 ประเภทของคอมพิวเตอร์	10
	1.4 องค์ประกอบหลักของคอมพิวเตอร์	15
	แบบฝึกหัดท้ายบท	37
แผนการสอนเรื่อง วิธีการออกแบบและพัฒนาโปรแกรม		39
บทที่ 2	หัวข้อบรรยายเรื่อง วิธีการออกแบบและพัฒนาโปรแกรม	42
	2.1 ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์	42
	2.2 การออกแบบอัลกอริทึมด้วยผังงาน (Flowchart)	59
	2.3 การออกแบบอัลกอริทึมด้วยรหัสเทียม (Pseudocode)	71
	แบบฝึกหัดท้ายบท	87
	ใบฝึกปฏิบัติการ	88

สารบัญ (ต่อ)

		หน้า
แผนการสอนเรื่อง พื้นฐานการเขียนโปรแกรม		101
บทที่ 3	หัวข้อบรรยายเรื่อง พื้นฐานการเขียนโปรแกรม	104
	3.1 แนะนำภาษาซี (Introduction to C)	105
	3.2 การติดตั้งโปรแกรมสำหรับพัฒนาโปรแกรมภาษาซี (C/C++ IDE)	111
	3.3 พื้นฐานและโครงสร้างภาษา C	144
	3.4 ตัวแปรและตัวดำเนินการ	154
	3.5 การแสดงผลข้อความทางหน้าจอและรับค่าจากคีย์บอร์ด	183
	3.6 ข้อผิดพลาดที่เกิดขึ้นบ่อย (Common Errors)	197
	แบบฝึกหัดท้ายบท	207
	ใบฝึกปฏิบัติการ	210
แผนการสอนเรื่อง เงื่อนไขและการตัดสินใจ		217
บทที่ 4	หัวข้อบรรยายเรื่อง เงื่อนไขและการตัดสินใจ	220
	4.1 แนะนำโปรแกรมแบบโครงสร้างควบคุม	221
	4.2 เงื่อนไขและเครื่องหมายที่ใช้ในการเปรียบเทียบ	225
	4.3 ตัวดำเนินการทางตรรกะ (Logical Operators)	227
	4.4 โครงสร้างการเลือกแบบถ้าแล้ว (if-else)	231
	4.5 โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)	244
	แบบฝึกหัดท้ายบท	253
	ใบฝึกปฏิบัติการ	254

สารบัญ (ต่อ)

		หน้า
แผนการสอนเรื่อง โครงสร้างการวนรอบ		281
บทที่ 5	หัวข้อบรรยายเรื่อง โครงสร้างการวนรอบ	284
	5.1 การวนรอบ	285
	5.2 คำสั่งวนรอบ for	288
	5.3 คำสั่งวนรอบ while	293
	5.4 คำสั่งวนรอบ do ... while	297
	5.5 การวนรอบแบบซ้อนหลายชั้น (Nested Loop)	299
	แบบฝึกหัดท้ายบท	304
	ใบฝึกปฏิบัติการ	306
แผนการสอนเรื่อง ฟังก์ชัน		329
บทที่ 6	หัวข้อบรรยายเรื่อง ฟังก์ชัน	332
	6.1 ความหมายและประโยชน์ของฟังก์ชัน	333
	6.2 ฟังก์ชันในภาษา C	339
	6.3 การสร้างฟังก์ชันโดยผู้เขียนโปรแกรม	344
	6.4 การเรียกใช้งานฟังก์ชัน	353
	6.5 ฟังก์ชันแบบเรียกตัวเอง	362
	6.6 ขอบเขตของตัวแปร	366
	แบบฝึกหัดท้ายบท	373
	ใบฝึกปฏิบัติการ	375
แผนการสอนเรื่อง อาร์เรย์		385
บทที่ 7	หัวข้อบรรยายเรื่อง อาร์เรย์	388
	7.1 นิยามของอาร์เรย์	389

สารบัญ (ต่อ)

	หน้า
7.2 ตัวแปรอาร์เรย์แบบ 1 มิติ	391
7.3 การเข้าถึงค่าในอาร์เรย์	394
7.4 การกำหนดค่าเริ่มต้นให้กับอาร์เรย์	399
7.5 การส่งผ่านอาร์เรย์ไปยังฟังก์ชัน	405
7.6 ตัวแปรอาร์เรย์แบบหลายมิติ	415
7.7 สายอักขระ (String)	427
7.8 ข้อผิดพลาดที่พบบ่อยในการใช้งานอาร์เรย์	435
แบบฝึกหัดท้ายบท	439
ใบฝึกปฏิบัติการ	441
แผนการสอนเรื่อง การประยุกต์ใช้โปรแกรมสำเร็จรูปทางวิศวกรรม Scilab	451
บทที่ 8 หัวข้อบรรยายเรื่อง การประยุกต์ใช้โปรแกรมสำเร็จรูปทางวิศวกรรม Scilab	454
8.1 ความรู้พื้นฐานเกี่ยวกับโปรแกรม Scilab	455
8.2 ความสามารถของ Scilab	456
8.3 การสร้างและใช้งานตัวแปร	460
8.4 โครงสร้างเงื่อนไขและการวนรอบของ Scilab	464
8.5 การเขียนฟังก์ชันใน Scilab	467
8.6 การวาดกราฟใน Scilab	470
8.7 การประยุกต์ใช้ Scilab ในงานวิศวกรรม	473

สารบัญ (ต่อ)

	หน้า
แบบฝึกหัดท้ายบท	477
ใบฝึกปฏิบัติการ	478
บรรณานุกรรม	483

แผนบริหารการสอนประจำรายวิชา

รหัสวิชา 04-000-104

ชื่อวิชา การโปรแกรมคอมพิวเตอร์

Computer Programming

จำนวนหน่วยกิต 3(2-3-5)

เวลาเรียน 75 ชั่วโมง/ภาคเรียน

จุดมุ่งหมายรายวิชา

1. มีความเข้าใจแนวคิดและองค์ประกอบของระบบคอมพิวเตอร์ การอันตรกิริยาระหว่างฮาร์ดแวร์และซอฟต์แวร์ และการประมวลผลข้อมูลทางอิเล็กทรอนิกส์
2. มีความเข้าใจวิธีการออกแบบและพัฒนาโปรแกรม การเขียนโปรแกรมด้วยภาษาระดับสูง
3. เป็นนักเทคโนโลยีสารสนเทศที่มีคุณภาพตรงกับความต้องการของตลาดแรงงาน

คำอธิบายรายวิชา

แนวคิดและองค์ประกอบของระบบคอมพิวเตอร์ การอันตรกิริยาระหว่างฮาร์ดแวร์และซอฟต์แวร์ การประมวลผลข้อมูลทางอิเล็กทรอนิกส์ วิธีการออกแบบและพัฒนาโปรแกรม การเขียนโปรแกรมด้วยภาษาระดับสูง การประยุกต์ใช้โปรแกรมด้วยภาษาระดับสูง การประยุกต์ใช้โปรแกรมสำเร็จรูปทางวิศวกรรม และการฝึกปฏิบัติ

Concept and components of computer systems; the interaction between hardware and software; electronic data processing; how to design and program development; high-level programming language; application of high-level language programming; the application of software engineering; and practice

รายละเอียดคำอธิบายรายวิชา	เนื้อหา
แนวคิดและองค์ประกอบของระบบคอมพิวเตอร์ การอันตรกิริยาระหว่างฮาร์ดแวร์และซอฟต์แวร์ การประมวลผลข้อมูลทางอิเล็กทรอนิกส์	บทที่ 1,3
วิธีการออกแบบและพัฒนาโปรแกรม	บทที่ 2
การเขียนโปรแกรมด้วยภาษาระดับสูง การประยุกต์ใช้โปรแกรมด้วยภาษาระดับสูง	บทที่ 3-7
การประยุกต์ใช้โปรแกรมสำเร็จรูปทางวิศวกรรม และการฝึกปฏิบัติ	บทที่ 8

จำนวนชั่วโมงที่ใช้ต่อภาคการศึกษา

บรรยาย	สอนเสริม	การฝึกปฏิบัติ/งานภาคสนาม/การฝึกงาน	การศึกษาด้วยตนเอง
30 ชั่วโมง	ไม่มี	45 ชั่วโมง	75 ชั่วโมง

เอกสารอ่านประกอบ

รศ.ดร.จันทนา จันทราพรชัย. (2555) สถาปัตยกรรมคอมพิวเตอร์และการออกแบบ (COMPUTER ARCHITECTURE AND DESING), สำนักพิมพ์ท้อป, ISBN: 978-616-282-009-0

รศ.ธีรวัฒน์ ประกอบผล. (2551) สถาปัตยกรรมคอมพิวเตอร์, สมาคมส่งเสริมเทคโนโลยี, ISBN: 978-974-443-369-5

ผศ.นราธร สังข์ประเสริฐ. (2563) การเขียนโปรแกรมด้วยภาษาซี ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-3945-2

โอภาส เอี่ยมสิริวงศ์. (2559) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-2740-4

รศ.ธีรวัฒน์ ประกอบผล. (2562) คู่มือการเขียนโปรแกรมภาษา C ฉบับสมบูรณ์, ซิมพลิฟาย, ISBN: 978-616-26-24896

นราธร สังข์ประเสริฐ, เอกสารประกอบการบรรยายวิชา 04-511-101 การโปรแกรมคอมพิวเตอร์, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย , 2558

Sites.google.com, "รหัสเทียม Psuedo Code - Programming", 2016. [Online]. Available: https://sites.google.com/site/programmingm42/-rhas-theiym-psuedo-code. [Accessed: 17- Jan- 2016].

Sites.google.com, "การแก้ปัญหา - Programming", 2016. [Online]. Available: https://sites.google.com/site/programmingm42/kar-kae-payha. [Accessed: 17- Jan- 2016].

คณาจารย์ผู้สอนวิชา Introduction to Computer Programming, เอกสารประกอบการบรรยายวิชา 241-101 Introduction to Computer Programming, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยสงขลานครินทร์, 2551

เอกสารอ่านประกอบ (ต่อ)

อ.สุธน แซ่ว่อง, เอกสารประกอบการบรรยายอบรมภาษาซีเบื้องต้น, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยสงขลานครินทร์, 30 มีนาคม 2552

Y. D. Liang, Supplement II.C:Dev-C++ Tutorial for Introduction to C++ Programming, http://www.bloodshed.net/dev/devcpp.html

ดร. แสงสุรีย์ วสุพงศ์อัยยะ, เอกสารประกอบการบรรยายอบรมภาษาซีเบื้องต้น, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยสงขลานครินทร์, 30 มีนาคม 2552

แหล่งเรียนเพิ่มเติม

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase1] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=aDBmXMF9jOU

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase2] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=wslM0Aqepxs

Mooc การเขียนโปรแกรมคอมพิวเตอร์พื้นฐาน ด้วยภาษาซี 1 | Fundamentals of the C Programming Language https://lms.thaimooc.org/courses/course-v1:KMUTT+KMUTT013+2019/about

Mooc ทักษะการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี | Computer programming skills with C language https://lms.thaimooc.org/courses/course-v1:SU+SU011+2021/about

Coursera: Programming in C, ผู้สอน: Charles Russell Severance https://www.coursera.org/learn/programming-c

Coursera: ความเชี่ยวชาญพิเศษ การเขียนโปรแกรม C เบื้องต้น, ผู้สอน: Andrew D. Hilton et al. https://www.coursera.org/specializations/c-programming

Playlist Youtube บันทึกการสอนในห้องเรียน, วิชาการโปรแกรมคอมพิวเตอร์, ดร.พงศกร เจริญเนตรกุล 2566 https://www.youtube.com/watch?v=ewEfmiXaRi0&list=PL322g2itJn4mAE1utyQzF-C0SP0BRkej3&pp=gAQBiAQB

แผนบริหารการสอนประจำวิชา

สัปดาห์ที่	หัวข้อ/รายละเอียด	จำนวน (ชั่วโมง)	กิจกรรมการเรียนการสอนและสื่อที่ใช้
1	**ทฤษฎี** บทที่ 1 ระบบคอมพิวเตอร์ 1.1 นิยามของคอมพิวเตอร์ 1.2 วิวัฒนาการของคอมพิวเตอร์ 1.3 ประเภทของคอมพิวเตอร์ 1.4 องค์ประกอบหลักของคอมพิวเตอร์ บทที่ 2 วิธีการออกแบบและพัฒนาโปรแกรม 2.1 ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์ 2.2.1 การวางแผน (Planning) 2.2.2 การวิเคราะห์ปัญหา (Analysis) 2.2.3 การออกแบบ (Design) 2.2.4 การพัฒนาโปรแกรม (Implementation) 2.2.5 การดูแลรักษาโปรแกรม (Maintenance) 2.2 การออกแบบอัลกอริทึมด้วยผังงาน (Flowchart)	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง 2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต 3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน 4. ทำใบฝึกปฏิบัติการในโปรแกรมพัฒนาภาษาซี 5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)

สัปดาห์ที่	หัวข้อ/รายละเอียด	จำนวน (ชั่วโมง)	กิจกรรมการเรียนการสอนและสื่อที่ใช้
	ปฏิบัติ -ออกแบบโปรแกรมด้วยผังงานระบบเบื้องต้น		
2	**ทฤษฎี** บทที่ 2 วิธีการออกแบบและพัฒนาโปรแกรม (ต่อ) 2.3 การออกแบบอัลกอริทึมด้วยรหัสเทียม (Pseudocode) **ปฏิบัติ** -ออกแบบโปรแกรมด้วยผังงานโปรแกรม -ออกแบบโปรแกรมด้วยรหัสเทียม (Pseudocode)	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง 2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต 3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน 4. ทำใบฝึกปฏิบัติการในโปรแกรมพัฒนาภาษาซี 5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)
3	**ทฤษฎี** บทที่ 3 พื้นฐานการเขียนโปรแกรม 3.1 แนะนำภาษาซี (Introduction to C) 3.2 การติดตั้งโปรแกรมสำหรับพัฒนาโปรแกรมภาษาซี (C/C++ IDE) 3.3 พื้นฐานและโครงสร้างของภาษา C **ปฏิบัติ**	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง 2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต 3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน 4. ทำใบฝึกปฏิบัติการในโปรแกรมพัฒนาภาษาซี

สัปดาห์ที่	หัวข้อ/รายละเอียด	จำนวน (ชั่วโมง)	กิจกรรมการเรียนการสอนและสื่อที่ใช้
	-ติดตั้งโปรแกรมสำหรับพัฒนาโปรแกรมภาษาซี (C/C++ IDE) -เริ่มต้นเขียนโปรแกรมภาษาซี		5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)
4	**ทฤษฎี** บทที่ 3 พื้นฐานการเขียนโปรแกรม (ต่อ) 3.4 ตัวแปรและตัวดำเนินการ 3.5 การแสดงผลข้อความทางหน้าจอและรับค่าจากคีย์บอร์ด 3.6 ข้อผิดพลาดที่เกิดขึ้นบ่อย (Common Errors) **ปฏิบัติ** -เขียนโปรแกรมภาษาซีเพื่อแสดงค่าด้วยฟังก์ชัน printf -เขียนโปรแกรมภาษาซีเพื่อรับค่าด้วยฟังก์ชัน scanf	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง 2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต 3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน 4. ทำใบฝึกปฏิบัติการในโปรแกรมพัฒนาภาษาซี 5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)
5	**ทฤษฎี** บทที่ 4 เงื่อนไขและการตัดสินใจ 4.1 แนะนำโปรแกรมแบบโครงสร้างควบคุม 4.2 เงื่อนไขและเครื่องหมายที่ใช้ในการเปรียบเทียบ	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง 2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต 3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน

สัปดาห์ที่	หัวข้อ/รายละเอียด	จำนวน (ชั่วโมง)	กิจกรรมการเรียนการสอนและสื่อที่ใช้
	4.3 ตัวดำเนินการทางตรรกะ (Logical Operators) **ปฏิบัติ** -เขียนโปรแกรมภาษาซีเพื่อออกแบบเงื่อนไข โดยใช้เครื่องหมายเปรียบเทียบและตัวดำเนินการทางตรรกะ		4. ทำใบฝึกปฏิบัติการในโปรแกรมพัฒนาภาษาซี 5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)
6	**ทฤษฎี** บทที่ 4 เงื่อนไขและการตัดสินใจ (ต่อ) 4.4 โครงสร้างการเลือกแบบถ้าแล้ว (if-else) 4.5 โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case) **ปฏิบัติ** -เขียนโปรแกรมโดยใช้โครงสร้าง if-else -เขียนโปรแกรมโดยใช้โครงสร้าง switch-case	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง 2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต 3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน 4. ทำใบฝึกปฏิบัติการในโปรแกรมพัฒนาภาษาซี 5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)
7	**ทฤษฎี** บทที่ 5 โครงสร้างการวนรอบ 5.1 การวนรอบ 5.2 คำสั่งวนรอบ for 5.3 คำสั่งวนรอบ while	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง 2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต

สัปดาห์ที่	หัวข้อ/รายละเอียด	จำนวน (ชั่วโมง)	กิจกรรมการเรียนการสอนและสื่อที่ใช้
	ปฏิบัติ -เขียนโปรแกรมโดยใช้โครงสร้าง for -เขียนโปรแกรมโดยใช้โครงสร้าง while		3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน 4. ทำใบฝึกปฏิบัติการในโปรแกรมพัฒนาภาษาซี 5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)
8	**ทฤษฎี** บทที่ 5 โครงสร้างการวนรอบ (ต่อ) 5.4 คำสั่งวนรอบ do ... while 5.5 การวนรอบแบบซ้อนหลายชั้น (Nested Loop) **ปฏิบัติ** -เขียนโปรแกรมโดยใช้โครงสร้าง do ... while -เขียนโปรแกรมโดยใช้การวนรอบแบบซ้อนหลายชั้น (Nested Loop)	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง 2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต 3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน 4. ทำใบฝึกปฏิบัติการในโปรแกรมพัฒนาภาษาซี 5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)
9	สอบกลางภาคเรียน	5	แบบทดสอบปรนัยและอัตนัย
10	**ทฤษฎี** บทที่ 6 ฟังก์ชัน	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง

สัปดาห์ที่	หัวข้อ/รายละเอียด	จำนวน (ชั่วโมง)	กิจกรรมการเรียนการสอนและสื่อที่ใช้
	6.1 ความหมายและประโยชน์ของฟังก์ชัน 6.2 ฟังก์ชันในภาษา C **ปฏิบัติ** -ทดลองออกแบบฟังก์ชันโดยแยก input, algorithm และ output		2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต 3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน 4. ทำใบฝึกปฏิบัติการในโปรแกรมพัฒนาภาษาซี 5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)
11	**ทฤษฎี** บทที่ 6 ฟังก์ชัน (ต่อ) 6.3 การสร้างฟังก์ชันโดยผู้เขียนโปรแกรม 6.4 การเรียกใช้งานฟังก์ชัน **ปฏิบัติ** -เขียนต้นแบบฟังก์ชันและนิยามฟังก์ชัน -เขียนโปรแกรมโดยเรียกใช้งานฟังก์ชัน	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง 2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต 3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน 4. ทำใบฝึกปฏิบัติการในโปรแกรมพัฒนาภาษาซี 5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)
12	บทที่ 6 ฟังก์ชัน (ต่อ) 6.5 ฟังก์ชันแบบเรียกตัวเอง	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง

สัปดาห์ที่	หัวข้อ/รายละเอียด	จำนวน (ชั่วโมง)	กิจกรรมการเรียนการสอนและสื่อที่ใช้
	6.6 ขอบเขตของตัวแปร		2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต 3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน 4. ทำใบฝึกปฏิบัติการในโปรแกรมพัฒนาภาษาซี 5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)
13	**ทฤษฎี** บทที่ 7 อาร์เรย์ 7.1 นิยามของอาร์เรย์ 7.2 ตัวแปรอาร์เรย์แบบ 1 มิติ 7.3 การเข้าถึงค่าในอาร์เรย์ **ปฏิบัติ** -เขียนโปรแกรมเพื่อประกาศตัวแปรอาร์เรย์ -เขียนโปรแกรมเพื่อแสดงผลและรับค่าอาร์เรย์	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง 2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต 3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน 4. ทำใบฝึกปฏิบัติการในโปรแกรมพัฒนาภาษาซี 5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)
14	**ทฤษฎี** บทที่ 7 อาร์เรย์(ต่อ)	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง

สัปดาห์ที่	หัวข้อ/รายละเอียด	จำนวน (ชั่วโมง)	กิจกรรมการเรียนการสอนและสื่อที่ใช้
	7.4 การกำหนดค่าเริ่มต้นให้กับอาร์เรย์ 7.5 การส่งผ่านอาร์เรย์ไปยังฟังก์ชัน 7.6 ตัวแปรอาร์เรย์แบบหลายมิติ **ปฏิบัติ** -เขียนโปรแกรมเพื่อแสดงผล-รับค่าอาร์เรย์ และดึงค่าในอาร์เรย์มาคำนวณ -เขียนโปรแกรมโดยใช้งานตัวแปรอาร์เรย์สองมิติ		2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต 3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน 4. ทำใบฝึกปฏิบัติการในโปรแกรมพัฒนาภาษาซี 5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)
15	**ทฤษฎี** บทที่ 7 อาร์เรย์(ต่อ) 7.7 สายอักขระ (String) 7.8 ข้อผิดพลาดที่พบบ่อยในการใช้งานอาร์เรย์ **ปฏิบัติ** -เขียนโปรแกรมโดยใช้งานตัวแปรสายอักขระ	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง 2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต 3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน 4. ทำใบฝึกปฏิบัติการในโปรแกรมพัฒนาภาษาซี 5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)
16	บทที่ 8 การประยุกต์ใช้โปรแกรมสำเร็จรูปทางวิศวกรรม Scilab	5	1. การบรรยายในชั้นเรียนด้วยภาพเลื่อนหรือภาพนิ่ง

สัปดาห์ที่	หัวข้อ/รายละเอียด	จำนวน (ชั่วโมง)	กิจกรรมการเรียนการสอนและสื่อที่ใช้
	ทฤษฎี 8.1 ความรู้พื้นฐานเกี่ยวกับโปรแกรม Scilab 8.2 ความสามารถของ Scilab 8.3 การสร้างและใช้งานตัวแปร 8.4 โครงสร้างเงื่อนไขและการวนรอบของ Scilab 8.5 การเขียนฟังก์ชันใน Scilab 8.6 การวาดกราฟใน Scilab 8.7 การประยุกต์ใช้ Scilab ในงานวิศวกรรม **ปฏิบัติ** -เขียนโปรแกรม Scilab เพื่อแก้ไขปัญหาทางวิศวกรรมและแสดงผลลัพธ์โดยใช้กราฟ		2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียนโดยใช้อินเทอร์เน็ต 3. จับกลุ่มระดมสมองแก้ไขโจทย์ในชั้นเรียน 4. ทำใบฝึกปฏิบัติการในโปรแกรมScilab 5. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)
17	สอบปลายภาคเรียน	5	แบบทดสอบปรนัยและอัตนัย
	รวมชั่วโมงสอน	**75**	

การประเมินผลการเรียนรู้

ผลการเรียนรู้	วิธีการประเมิน	สัปดาห์ที่ประเมิน	สัดส่วนของการประเมินผล
1. คุณธรรม จริยธรรม	1.1 ประเมินจากการเข้าชั้นเรียน การส่งแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการตามกำหนดเวลาที่มอบหมาย 1.2 ประเมินจากพฤติกรรมของนักศึกษาในการปฏิบัติตามกฎระเบียบและข้อบังคับต่าง ๆ ทั้งในข้อบังคับในชั้นเรียน ข้อบังคับของมหาวิทยาลัย 1.3 ประเมินจากการคัดลอกแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการ 1.4 ประเมินจากการมีส่วนร่วมในห้องเรียน และการปิดคอมพิวเตอร์ เก็บขยะ เก็บเก้าอี้หลังจากใช้งานเสร็จ	ตลอดภาคการศึกษา	10%
2. ความรู้	2.1 แบบฝึกหัดท้ายบท 2.2 การสอบกลางภาคเรียนและปลายภาคเรียน	2-8, 11-16 9, 17	20% 40%
3. ทักษะทางปัญญา	3.1 ประเมินจากใบฝึกปฏิบัติการ	ตลอดภาคการศึกษา	20%

ผลการเรียนรู้	วิธีการประเมิน	สัปดาห์ที่ประเมิน	สัดส่วนของการประเมินผล
4. ทักษะความสัมพันธ์ระหว่างบุคคลและความรับผิดชอบ	4.1 พิจารณาจากการทำใบฝึกปฏิบัติการ 4.3 ประเมินจากพฤติกรรมในการทำงานกลุ่ม การระดมความคิดเห็น การตอบคำถามในชั้นเรียน	ตลอดภาคการศึกษา	5%
5. ทักษะวิเคราะห์เชิงตัวเลข การสื่อสารและการใช้เทคโนโลยีสารสนเทศ	5.1 การทดสอบหรือการประเมินจากงานที่มอบหมาย 5.2 สังเกตการใช้คอมพิวเตอร์ในการเขียนโปรแกรม การทำใบฝึกปฏิบัติการ และการค้นหาข้อมูลเพิ่มเติมจากอินเทอร์เน็ต	ตลอดภาคการศึกษา	5%
	รวม		100%

แผนการสอนเรื่อง ระบบคอมพิวเตอร์

สัปดาห์ที่ 1

จำนวนชั่วโมงที่ใช้สอน 3 ชั่วโมง

วัตถุประสงค์ เพื่อให้นักศึกษาสามารถ

1. บอกนิยามของคอมพิวเตอร์และสามารถ อธิบาย การทำงานเบื้องต้นของคอมพิวเตอร์ได้อย่างชัดเจน
2. บอกวิวัฒนาการของคอมพิวเตอร์ในแต่ละยุค และสามารถอธิบายความแตกต่างระหว่างเทคโนโลยีในแต่ละยุค เช่น ยุคหลอดสุญญากาศ ทรานซิสเตอร์ วงจรรวม และไมโครโปรเซสเซอร์
3. บอกประเภทของคอมพิวเตอร์ต่าง ๆ เช่น ซูเปอร์คอมพิวเตอร์ เมนเฟรมคอมพิวเตอร์ และไมโครคอมพิวเตอร์ และสามารถ อธิบาย ลักษณะและการใช้งานของคอมพิวเตอร์แต่ละประเภทได้
4. บอกองค์ประกอบหลักของคอมพิวเตอร์ เช่น CPU, RAM, และหน่วยเก็บข้อมูล และสามารถอธิบายการทำงานของแต่ละองค์ประกอบได้
5. เข้าใจความแตกต่างระหว่าง RAM และ ROM และสามารถอธิบายบทบาทของหน่วยความจำในระบบคอมพิวเตอร์ได้
6. บอกแนวคิดของการเข้ารหัสตัวอักษร (Character Encoding) เช่น ASCII และ UTF-8 และสามารถ อธิบาย การทำงานของการเข้ารหัสตัวอักษรในการเก็บและแสดงข้อมูลได้

หัวข้อบรรยาย

บทที่ 1 ระบบคอมพิวเตอร์

1.1 นิยามของคอมพิวเตอร์

1.2 วิวัฒนาการของคอมพิวเตอร์

1.3 ประเภทของคอมพิวเตอร์

1.4 องค์ประกอบหลักของคอมพิวเตอร์

กิจกรรมการเรียนการสอน

1. การบรรยายในชั้นเรียน
2. การหาข้อมูลเพิ่มเติมนอกเหนือจากบทเรียน โดยใช้อินเทอร์เน็ต
3. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)

สื่อการสอน

1. เอกสารประกอบการสอนแบบภาพเลื่อนหรือภาพนิ่ง
2. การเข้าถึงข้อมูลจากอินเทอร์เน็ต

การประเมินผลการเรียนรู้

ผลการเรียนรู้	วิธีการประเมิน	สัปดาห์ที่ประเมิน	สัดส่วนของการประเมินผล
1. คุณธรรมจริยธรรม	1.1 ประเมินจากการเข้าชั้นเรียน การส่งแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการตามกำหนดเวลาที่มอบหมาย 1.2 ประเมินจากพฤติกรรมของนักศึกษาในการปฏิบัติตามกฎระเบียบและข้อบังคับต่าง ๆ ทั้งในข้อบังคับในชั้นเรียน ข้อบังคับของมหาวิทยาลัย 1.3 ประเมินจากการคัดลอกแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการ 1.4 ประเมินจากการมีส่วนร่วมในห้องเรียนและการปิดคอมพิวเตอร์ เก็บขยะ เก็บเก้าอี้หลังจากใช้งานเสร็จ	ตลอดภาคการศึกษา	10%
2. ความรู้	2.1 แบบฝึกหัดท้ายบท 2.2 การสอบกลางภาคเรียนและปลายภาคเรียน	2-8, 11-16 9, 17	20% 40%
3. ทักษะทางปัญญา	3.1 ประเมินจากใบฝึกปฏิบัติการ	ตลอดภาคการศึกษา	20%
4. ทักษะความสัมพันธ์ระหว่างบุคคลและความรับผิดชอบ	4.1 พิจารณาจากการทำใบฝึกปฏิบัติการ 4.3 ประเมินจากพฤติกรรมในการทำงานกลุ่ม การระดมความคิดเห็น การตอบคำถามในชั้นเรียน	ตลอดภาคการศึกษา	5%
5. ทักษะวิเคราะห์เชิงตัวเลข การสื่อสารและการใช้เทคโนโลยีสารสนเทศ	5.1 การทดสอบหรือการประเมินจากงานที่มอบหมาย 5.2 สังเกตการใช้คอมพิวเตอร์ในการเขียนโปรแกรม การทำใบฝึกปฏิบัติการ และการค้นหาข้อมูลเพิ่มเติมจากอินเทอร์เน็ต	ตลอดภาคการศึกษา	5%
	รวม		100%

อ่าน Slide เวอร์ชันล่าสุดได้ที่ Google Slide

https://s.coecore.com/scp1

การโปรแกรมคอมพิวเตอร์

บทที่ 1 ระบบคอมพิวเตอร์

(Computer System)

ผู้สอน ดร.พงศกร เจริญเนตรกุล

Outline

- ❑ นิยามและคอมพิวเตอร์รูปแบบต่างๆ
- ❑ วิวัฒนาการของคอมพิวเตอร์ในยุคต่างๆ
- ❑ ประเภทของคอมพิวเตอร์
- ❑ องค์ประกอบหลักของคอมพิวเตอร์ Hardware / Software

นิยามและคอมพิวเตอร์รูปแบบต่างๆ

นิยามของ “คอมพิวเตอร์”

“**An electronic machine** that is used for storing, organizing, and finding words, numbers, and pictures, for doing calculations, and for controlling other machines.”

อ้างอิงนิยาม “Computer” จาก Cambridge Dictionary

เครื่องมือที่ใช้ในการจัดเก็บ ประมวลผล คำนวณ และค้นหาข้อมูลต่าง ๆ (อักษร ตัวเลข ภาพ ฯลฯ) โดยมีชุดคำสั่งเพื่อใช้ในการประมวลผล และยังใช้เพื่อควบคุมเครื่องจักรอื่น ๆ อีกด้วย



อ้างอิงรูปภาพ: pixabay.com/th/photos/แอปเปิล-คอมพวเตอร-โตะ-1868496/

วิวัฒนาการของคอมพิวเตอร์ในยุคต่างๆ

ยุคที่ 1: ยุคหลอดสุญญากาศ (ค.ศ.1951-1958)

เทคโนโลยี: ใช้หลอดสุญญากาศในการสร้างวงจรอิเล็กทรอนิกส์

ตัวอย่างคอมพิวเตอร์: ENIAC, UNIVAC I

ลักษณะเด่น: ขนาดใหญ่ ใช้พลังงานมาก และมีความร้อนสูง

การใช้งาน: การคำนวณทางวิทยาศาสตร์และการทหาร





 อ้างอิงรูปภาพ: https://simple.wikipedia.org/wiki/Vacuum_tube และ https://th.wikipedia.org/wiki/อีนิแอก

ยุคที่ 2: ยุคทรานซิสเตอร์ (ค.ศ.1959-1963)

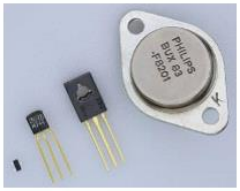
เทคโนโลยี: ใช้ทรานซิสเตอร์แทนหลอดสุญญากาศ

ตัวอย่างคอมพิวเตอร์: IBM 1620, UNIVAC III

ลักษณะเด่น: ขนาดเล็กลง ใช้พลังงานน้อยลง และมีความน่าเชื่อถือมากขึ้น

การใช้งาน: การประมวลผลข้อมูลในธุรกิจและองค์กร





 อ้างอิงรูปภาพ: https://en.wikipedia.org/wiki/Transistor_computer และ https://en.wikipedia.org/wiki/Transistor

ยุคที่ 3: ยุควงจรรวม (ค.ศ.1964-1970)

เทคโนโลยี: ใช้วงจรรวม (Integrated Circuits) ซึ่งประกอบด้วยทรานซิสเตอร์หลายตัวในแผงเดียว

ตัวอย่างคอมพิวเตอร์: IBM System/360, PDP-8

ลักษณะเด่น: ความเร็วสูงขึ้น ราคาถูกลง และขนาดเล็กลง

การใช้งาน: การประมวลผลหลายงานพร้อมกัน (multi-tasking) และการพัฒนาซอฟต์แวร์ระบบ



 อ้างอิงรูปภาพ: https://en.wikipedia.org/wiki/IBM_System/360

ยุคที่ 4: ยุคไมโครโปรเซสเซอร์ หรือ VLSI (ค.ศ.1971-1980)

เทคโนโลยี: ใช้ไมโครโปรเซสเซอร์ที่รวมเอาวงจรทั้งหมดไว้ในชิปเดียว

ตัวอย่างคอมพิวเตอร์: Apple II, IBM PC

ลักษณะเด่น: ขนาดเล็กลงอย่างมาก ความเร็วสูงขึ้นและราคาถูกลง

การใช้งาน: เกิดไมโครโพรเซสเซอร์, คอมพิวเตอร์ส่วนบุคคล, อินเทอร์เฟสแบบกราฟิก



 อ้างอิงรูปภาพ: https://en.wikipedia.org/wiki/Apple_II

ยุคที่ 5: ยุค Ultra Large-Scale Integraed (ULSI) (ค.ศ.1981-ปัจจุบัน)

เทคโนโลยี: ไมโครโปรเซสเซอร์มีขนาดทรานซิสเตอร์มากกว่า 1 ล้านตัว

ตัวอย่างคอมพิวเตอร์: Apple M2, Intel® Core™ Ultra 7

ลักษณะเด่น: คอมพิวเตอร์ประมวลผลได้เร็วมากกว่าเดิม, ประสิทธิภาพสูง, ขนาดเล็กลง, กินพลังงานน้อยลง

การใช้งาน: คอมพิวเตอร์ส่วนบุคคล



 อ้างอิงรูปภาพ: https://www.beartai.com/tech/it-news/1069833

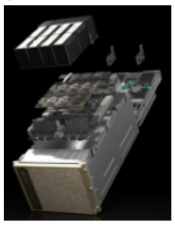
ยุคที่ 6: ยุคปัญญาประดิษฐ์และเทคโนโลยีอนาคต (ปัจจุบัน-อนาคต)

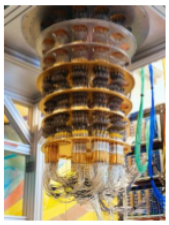
เทคโนโลยี: ใช้ปัญญาประดิษฐ์ (AI), การเรียนรู้ของเครื่อง (Machine Learning), ควอนตัมคอมพิวติ้ง (Quantum Computing)

ตัวอย่างคอมพิวเตอร์: Nvidia DGX H200, Google's Quantum Computer

ลักษณะเด่น: ความสามารถในการประมวลผลที่ซับซ้อน การเรียนรู้จากข้อมูล และการคำนวณที่รวดเร็วมากขึ้น

การใช้งาน: การวิเคราะห์ข้อมูลขนาดใหญ่ การแพทย์ การเงิน และอุตสาหกรรมต่างๆ





อ้างอิงรูปภาพ: https://quantumai.google/quantumcomputer และ https://www.nvidia.com/en-us/data-center/dgx-h200/

"คอมพิวเตอร์" ในรูปแบบต่างๆ



Laptop / Notebook สำหรับพกพา



PC / Server



Raspberry PI 5 คอมพิวเตอร์ขนาดเล็ก



PC Gamming สำหรับเล่นเกมส์



Raspberry PI Zero คอมพิวเตอร์ขนาดจิ๋ว

อ้างอิงรูปภาพ:
https://pixabay.com/th/photos/macbook-%E0%B9%81%E0%B8%A5%E0%B8%9B%E0%B8%97%E0%B8%AD%E0%B8%9E-google-%E0%B9%81%E0%B8%AA%E0%B8%94%E0%B8%87-459196/
https://www.raspberrypi.com/products/raspberry-pi-5/
https://www.dell.com/en-th/shop/cty/sc/desktops
https://www.cnx-software.com/2021/10/29/raspberry-pi-zero-2-w-mini-review-benchmarks-and-thermal-performance/

ประเภทของคอมพิวเตอร์

ประเภทของคอมพิวเตอร์

1.ซูเปอร์คอมพิวเตอร์ (Supercomputer)

มีความสามารถสูงสุดในบรรดาคอมพิวเตอร์ทั้งหมด ใช้สำหรับการคำนวณที่ซับซ้อน เช่น การวิจัยทางวิทยาศาสตร์ การคำนวณทางสภาพอากาศ และการจำลองสถานการณ์ต่าง ๆ

สามารถประมวลผลข้อมูลในระดับเพตะฟล็อปส์ (petaflops) หรือสูงกว่าเมนเฟรมคอมพิวเตอร์ (Mainframe Computer)



อ้างอิงรูปภาพ: https://pixabay.com/th/photos/

อ้างอิง OPENAI. (2024). ChatGPT (July ChatGPT 4O) [Large Language Model]. https://chatgpt.com

ประเภทของคอมพิวเตอร์

2.เมนเฟรมคอมพิวเตอร์ (Mainframe Computer)

ใช้ในองค์กรขนาดใหญ่เพื่อจัดการกับข้อมูลและการทำงานที่ต้องการความเสถียรสูง เช่น ธนาคาร ประกันภัย และองค์กรรัฐ

สามารถรองรับผู้ใช้งานหลายพันคนพร้อมกันได้ และมีความน่าเชื่อถือสูง



อ้างอิงรูปภาพ: https://pixabay.com/th/photos/

อ้างอิง OPENAI. (2024). ChatGPT (July ChatGPT 4O) [Large Language Model]. https://chatgpt.com

ประเภทของคอมพิวเตอร์

3.มินิคอมพิวเตอร์ (Minicomputer)

ขนาดเล็กกว่าเมนเฟรม แต่ใหญ่กว่าคอมพิวเตอร์ส่วนบุคคล ใช้ในองค์กรขนาดกลางที่ต้องการการประมวลผลแบบกลาง ๆ

ปัจจุบันคำว่า "มินิคอมพิวเตอร์" อาจไม่ค่อยใช้ เนื่องจากเทคโนโลยีได้พัฒนาไปมาก



อ้างอิงรูปภาพ: https://pixabay.com/th/photos/

อ้างอิง OPENAI. (2024). ChatGPT (July ChatGPT 4O) [Large Language Model]. https://chatgpt.com

ประเภทของคอมพิวเตอร์

4.ไมโครคอมพิวเตอร์ (Microcomputer)

หรือที่รู้จักกันในชื่อ "คอมพิวเตอร์ส่วนบุคคล" (Personal Computer หรือ PC) ใช้สำหรับการทำงานทั่วไปที่บ้านหรือในสำนักงาน

มีขนาดเล็กและราคาถูก ใช้งานง่ายและสามารถประยุกต์ใช้งานได้หลากหลาย



อ้างอิงรูปภาพ: https://pixabay.com/th/photos/

อ้างอิง OPENAI. (2024). ChatGPT (July ChatGPT 4O) [Large Language Model]. https://chatgpt.com

ประเภทของคอมพิวเตอร์

5.เวิร์คสเตชัน (Workstation)

คอมพิวเตอร์ประสิทธิภาพสูงสำหรับงานที่ต้องการการคำนวณและกราฟิกที่ซับซ้อน เช่น งานวิศวกรรม การออกแบบกราฟิก และการทำงานในห้องปฏิบัติการ

มีความเสถียรและประสิทธิภาพสูงกว่า PC ทั่วไป



อ้างอิงรูปภาพ: https://www.dell.com/en-us/shop/desktop-computers/precision-5860-tower/spd/precision-5860-workstation

อ้างอิง OPENAI. (2024). ChatGPT (July ChatGPT 4O) [Large Language Model]. https://chatgpt.com

ประเภทของคอมพิวเตอร์

6.แล็ปท็อป (Laptop)

คอมพิวเตอร์พกพาที่มีขนาดเล็กและเบา สามารถทำงานได้เหมือนกับ PC แต่สะดวกต่อการพกพา

มีแบตเตอรี่ในตัว ใช้งานได้ทั้งในและนอกสถานที่



อ้างอิงรูปภาพ: https://pixabay.com/th/photos/

อ้างอิง OPENAI. (2024). ChatGPT (July ChatGPT 4O) [Large Language Model]. https://chatgpt.com

ประเภทของคอมพิวเตอร์

7.แท็บเล็ต (Tablet)

คอมพิวเตอร์แบบพกพาที่ใช้หน้าจอสัมผัส ใช้งานง่ายและสะดวกสำหรับการท่องอินเทอร์เน็ต ดู

วิดีโอ และอ่านหนังสือ

มีขนาดเล็กกว่าแล็ปท็อปและไม่มีคีย์บอร์ดแบบดั้งเดิม



อ้างอิงรูปภาพ: https://pixabay.com/th/photos/

อ้างอิง OPENAI. (2024). ChatGPT (July ChatGPT 4O) [Large Language Model]. https://chatgpt.com

ประเภทของคอมพิวเตอร์

8.สมาร์ทโฟน (Smartphone)

คอมพิวเตอร์ขนาดเล็กที่รวมความสามารถในการโทรศัพท์และการประมวลผลข้อมูล ใช้ระบบปฏิบัติการที่คล้ายกับคอมพิวเตอร์เช่น iOS และ Android

สามารถทำงานต่าง ๆ ได้หลากหลาย เช่น ส่งอีเมล เล่นเกม และใช้งานแอปพลิเคชันต่าง ๆ

ประเภทของคอมพิวเตอร์

9.เซิร์ฟเวอร์ (Server)

คอมพิวเตอร์ที่ให้บริการต่าง ๆ แก่อุปกรณ์อื่น ๆ ในเครือข่าย เช่น การเก็บข้อมูล การประมวลผลข้อมูล และการให้บริการเว็บ

มีประสิทธิภาพและความเสถียรสูง เพื่อรองรับการทำงานของผู้ใช้หลายคนพร้อมกัน



อ้างอิงภาพ: https://www.dell.com/en-th/shop/cty/sc/desktops

ประเภทของคอมพิวเตอร์

10.ไมโครคอนโทรลเลอร์ (Microcontroller)

คอมพิวเตอร์ขนาดเล็กที่มีการประมวลผลเฉพาะด้าน ใช้ในอุปกรณ์อิเล็กทรอนิกส์ต่าง ๆ เช่น เครื่องใช้ไฟฟ้า รถยนต์ และระบบควบคุมอัตโนมัติ

มีความสามารถจำกัด แต่มีการใช้งานที่เฉพาะเจาะจง

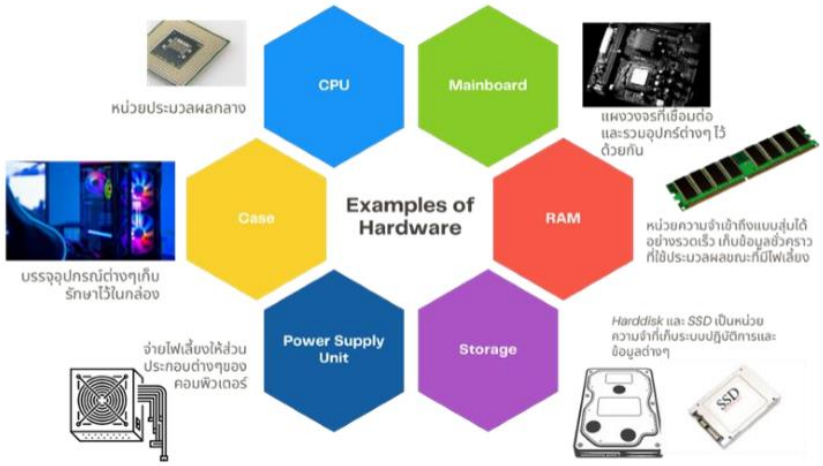




อ้างอิงภาพ: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ https://th.cytron.io/p-nodemcu-esp32

 อ้างอิง OPENAI. (2024). ChatGPT (July ChatGPT 4O) [Large Language Model]. https://chatgpt.com

องค์ประกอบหลักของคอมพิวเตอร์



องค์ประกอบหลักของคอมพิวเตอร์

Hardware



Software



Hardware

ฮาร์ดแวร์ (Hardware)

นิยามของ Hardware ของ Computer = Physical (จับต้องได้)

Hardware ของคอมพิวเตอร์หมายถึงส่วนประกอบทางกายภาพที่ประกอบขึ้นเป็นระบบคอมพิวเตอร์ ซึ่งรวมถึงอุปกรณ์และส่วนประกอบต่างๆ ที่ทำหน้าที่ในการประมวลผลข้อมูล เก็บข้อมูล และเชื่อมต่อกับอุปกรณ์ภายนอก ประกอบด้วย: CPU, RAM, STORAGE, MAINBOARD, GPU, I/O Device, etc.

ฮาร์ดแวร์ (Hardware)

หน่วยประมวลผลกลาง (CPU): ทำหน้าที่เป็นสมองของคอมพิวเตอร์ในการประมวลผลคำสั่งและข้อมูล





ข้างอิงรูปภาพ:
https://www.nextstepreborn.co.th/product/%E0%B8%A1%E0%B8%B7%E0%B8%AD%E0%B8%AA%E0%B8%AD%E0%B8%87-cpu-intel-core-i5-6500-3-2-ghz/
https://www.trustedreviews.com/explainer/what-is-a-cpu-2950255

หน่วยประมวลผลกลาง CPU : Central Processing Unit

หน่วยประมวลผลกลาง ที่ทำหน้าที่ประมวลผลคำสั่งและข้อมูล
ทำหน้าที่คล้ายกับสมองมนุษย์ (+RAM)

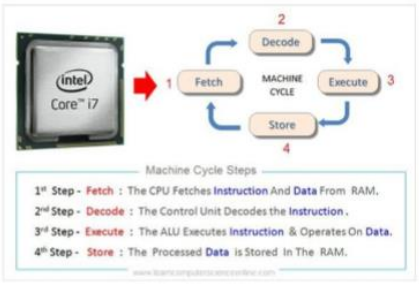
ประกอบด้วยหน่วยหลัก ได้แก่

1) หน่วยคำนวณทางคณิตศาสตร์และตรรกะ : ALU (Arithmetic & Logical Unit)
2) หน่วยควบคุม : CU
3) ตัวแปรเก็บข้อมูลของ CPU : Registers (หน่วยความจำที่ติดต่อกับ CPU ได้เร็วที่สุด ส่วนใหญ่อยู่ใน CPU)
4) หน่วยเชื่อมต่อกับ I/O : I/O Interface
5) หน่วยเก็บข้อมูลชั่วคราวที่ใช้บ่อย : Cache

หน่วยประมวลผลกลาง CPU : Central Processing Unit

ขั้นตอนในการทำงานของ CPU

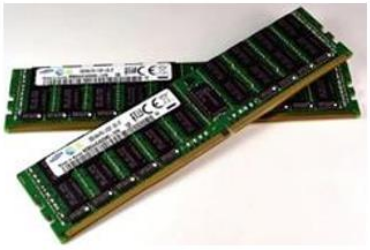
Fetch Ins / Decode Ins / Execute Ins / Write Data (Ins=Instruction=ชุดคำสั่ง)



อ้างอิงรูปภาพ:
https://www.learncomputerscienceonline.com

ฮาร์ดแวร์ (Hardware)

หน่วยความจำ (RAM): ใช้สำหรับเก็บข้อมูลชั่วคราวที่ใช้งานในขณะนั้นเพื่อให้ CPU เข้าถึงได้รวดเร็ว





อ้างอิงรูปภาพ:
https://www.mindphp.com/คู่มือ/73-คืออะไร/3696-ram-แรม-คืออะไร.html#gogle_vignette
https://www.nextstepreborn.co.th/product/ram-kingston-fury-beast-16gb-8gbx2-ddr4-3200mhz-rgb/

ฮาร์ดแวร์ (Hardware)

หน่วยเก็บข้อมูล (Storage): อุปกรณ์ที่ใช้เก็บข้อมูลระยะยาว เช่น ฮาร์ดดิสก์ (HDD) และโซลิดสเตทไดรฟ์ (SSD)





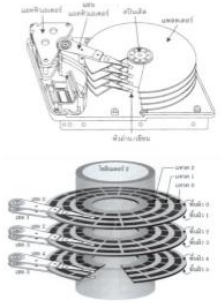


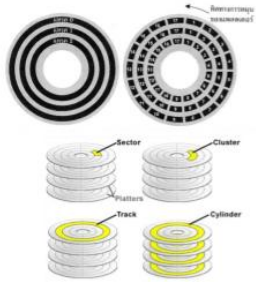
อ้างอิงรูปภาพ: https://www.westerndigital.com/th-th/solutions/color-drives

ฮาร์ดดิส (Harddisk)

ฮาร์ดดิส (Harddisk) ใช้หลักการเหนี่ยวนำของสนามแม่เหล็กโดยแม่เหล็กเพื่อใช้ในการบันทึกข้อมูล ซึ่งมีสองขั้วคือ ขั้วเหนือและขั้วใต้ ข้อมูลที่ใช้เก็บบนแผ่นดิสก์จะอยู่ในลักษณะ " 0 " กับ " 1 " เหมือนกับข้อมูลดิจิตอลอื่น ๆ





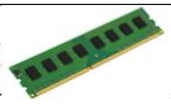


ส่วนเก็บข้อมูล (เรียงจากส่วนหลัก->ย่อย)

- Platter
- Cylinder
- Cluster
- Track
- Sector

อ้างอิงรูปภาพ: https://www.westerndigital.com/th-th/solutions/color-drives

RAM vs ROM

RAM 	ROM 
Random Access Memory	Read Only Memory
หน่วยความจำที่เข้าถึง(อ่าน/เขียน/ลบ)ได้แบบสุ่ม	หน่วยความจำที่อ่านได้อย่างเดียว
Volatile – ต้องมีไฟเลี้ยงถึงข้อมูลยังอยู่	Non-volatile - ไม่มีไฟเลี้ยงก็ข้อมูลไม่หาย
ใช้เก็บข้อมูลในการประมวลผล / ข้อมูลของ Apps ที่กำลังรันในระบบ	ใช้เก็บข้อมูล OS โปรแกรม ข้อมูลของผู้ใช้ ภาพ วิดิโอ เกมส์ ฯลฯ ที่ปิดเครื่องเปิดใหม่แล้วจะไม่หายไป

** Rom ที่อยู่ใน Spec มือถือนั่นผิดนะ ที่จริงมันคือ Flash mem สามารถอ่าน/เขียน/ลบ/เขียนทับใหม่ได้

** เพิ่มเติม Basic input/output system (BIOS) = หน่วยความจำที่เก็บข้อมูลพื้นฐานของระบบ

Unit of Memory

บิต (Bit) เป็นหน่วยข้อมูลเล็กที่สุดของคอมพิวเตอร์

8 บิต (Bit) = 1 ไบต์ (Byte)

ตัวอย่างเลขฐานสองในคอมพิวเตอร์⟶ 1011 0101 (bin) / B5 (hex)

Unit of Memory

1 Byte (ไบต์)	= 1 ตัวอักษร	1 Bytes	2^{0}
1 KB (กิโลไบต์)	= 1024 ตัวอักษร	1,024 Bytes	2^{10}
1 MB (เมกกะไบต์)	= 1024 KB	1,048,576 Bytes	2^{20}
1 GB (กิกะไบต์)	= 1024 MB	1,073,741,824 Bytes	2^{30}
1 TB (เทราไบต์)	= 1024 GB	1,099,511,627,776 Bytes	2^{40}

Character Encoding

รหัสข้อมูล (Data representation) ⟶ เป็นรหัสที่ใช้แทนตัวเลข ตัวอักษร และสัญลักษณ์ต่าง ๆ โดยใช้เลขฐานสองมาเรียงต่อกันเป็นกลุ่ม

ตัวอย่าง Encoding Character

1) BCD (Binary Code Decimal)

2) EBCDIC (Extended Binary Coded Decimal Interchange Code)

3) **ASCII** (American National Standard Institute) ใช้ 1 byte แทน 1 ตัวอักษร ซึ่ง ASCII code สามารถมีตัวอักษรได้ ทั้งหมด 256 ตัว

Character Encoding

Standard ASCII Chart / ASCII Table - Hex to Decimal Code Conversion

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

อ้างอิงรูปภาพ:https://www.commfront.com/pages/ascii-chart

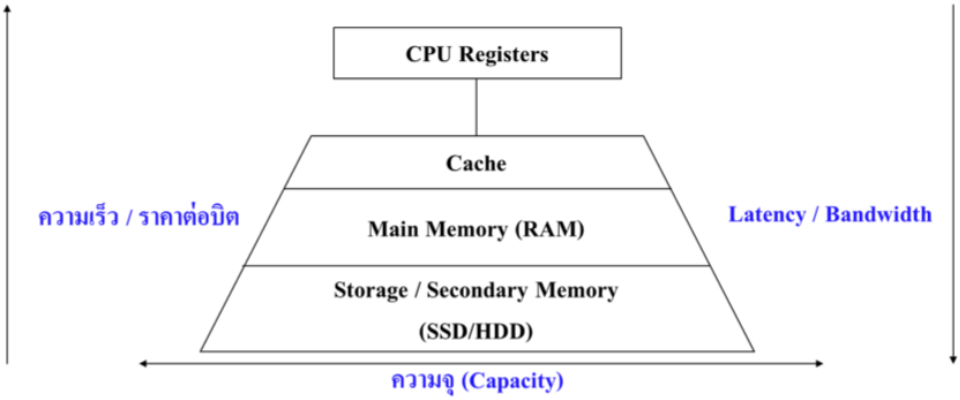
Character Encoding

4) UTF-8 (Multi language) หรือ Unicode 1 ตัวอักษรใช้เนื้อที่ในการเก็บข้อมูลหลายไบต์ จึงทำให้สามารถสร้างตัวแทนอักษรต่างๆได้หลากหลาย จึงรองรับตัวอักษรหลายภาษา

Wrong Encoding



Hardware : Memory Hierarchy



Computer Programming 41

Hardware : Memory Hierarchy

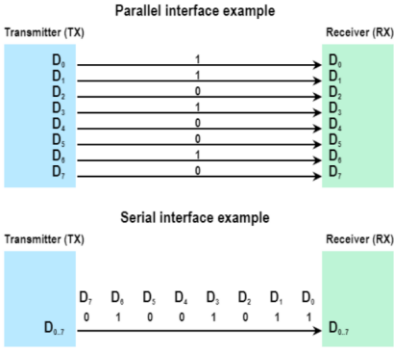
(Updated 2019)	Access Type	Capacity	Latency	Bandwidth
CPU Register	Random	64 - 2048 Bytes	Few clocks	System Clock Rate
Cache Mem	Random	8KB - 10MB	Few clocks	~4,000GB/s or more
Main Mem (RAM)	Random	128MB - 128GB	< 6 ns	~60GB/s or more
Harddisk	Direct	500GB -10TB	~2-8 ms	50MB/s – 300MB/s
SSD	Random	64GB – 5TB	10-30 µs	500MB/s – 3,500MB/s

Computer Programming 42

รูปแบบการส่งข้อมูล

พอร์ตอนุกรม(Serial) ส่งข้อมูลเรียงกันไปทีละบิตบนสายหนึ่งเส้น

พอร์ตขนาน (Parallel) ส่งข้อมูลในแต่ละบิตออกไปพร้อม ๆ กันบนสายหลาย ๆ เส้น



อ้างอิงรูปภาพ: https://en.wikipedia.org/wiki/Serial_communication

ฮาร์ดแวร์ (Hardware)

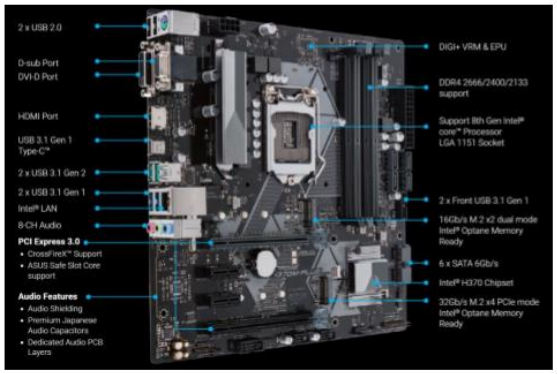
เมนบอร์ด (Motherboard): แผงวงจรหลักที่เชื่อมต่อส่วนประกอบต่าง ๆ ของคอมพิวเตอร์



อ้างอิงรูปภาพ:
https://www.asus.com/supportonly/prime%20x470-pro/helpdesk_cpu/

Back Panel Ports of Mainboard

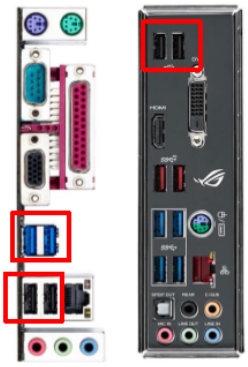




*Plug and Play เป็นมาตราฐานได้ผู้ผลิตอุปกรณ์คอมพิวเตอร์ตกลงกัน เพื่อใช้ในการเชื่อมต่อระหว่างอุปกรณ์ฮาร์ดแวร์และซอฟแวร์ เพื่อให้อุปกรณ์สามารถที่จะจัดการติดตั้งตนเองเข้ากับระบบได้ เมื่อเชื่อมต่อเข้ากับระบบ

อ้างอิงรูปภาพ: https://www.asus.com/th/motherboards-components/motherboards/prime/prime-h370m-plus/

Back Panel Ports of Mainboard



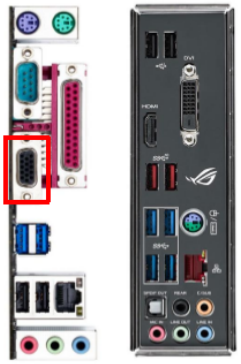
USB (Universal Serial Bus) **เป็นมาตรฐานการเชื่อมต่อที่ใช้สำหรับเชื่อมต่ออุปกรณ์อิเล็กทรอนิกส์ต่าง ๆ เช่น คอมพิวเตอร์ สมาร์ทโฟน แป้นพิมพ์ เมาส์ เครื่องพิมพ์ และอุปกรณ์จัดเก็บข้อมูล** โดยมีการพัฒนาเป็นหลายรุ่นเพื่อเพิ่มความเร็วในการถ่ายโอนข้อมูลและการชาร์จพลังงาน รุ่นที่สำคัญ ได้แก่:

1. **USB 1.0/1.1**: มีความเร็วการถ่ายโอนข้อมูลสูงสุด 12 Mbps
2. **USB 2.0**: เพิ่มความเร็วสูงสุดเป็น 480 Mbps
3. **USB 3.0**: เพิ่มความเร็วเป็น 5 Gbps (หรือเรียกว่า SuperSpeed)
4. **USB 3.1/3.2**: ความเร็วสูงสุด 10-20 Gbps
5. **USB4**: ความเร็วสูงสุด 40 Gbps และสนับสนุนการเชื่อมต่อ Thunderbolt 3

นอกจากการถ่ายโอนข้อมูลแล้ว USB ยังใช้ในการชาร์จพลังงาน โดยรุ่นใหม่ ๆ มีการรองรับการชาร์จเร็ว (fast charging) ด้วย

อ้างอิงรูปภาพ: https://www.asus.com/th/motherboards-components/motherboards/prime/prime-h370m-plus/

Back Panel Ports of Mainboard



VGA (Video Graphics Array) หรือที่เรียกว่า **D-Sub** เป็น**พอร์ตเชื่อมต่อแบบอนาล็อกที่ใช้สำหรับการส่งสัญญาณภาพระหว่างคอมพิวเตอร์และจอภาพหรือโปรเจ็กเตอร์** พอร์ตนี้มีการใช้งานอย่างแพร่หลายในอดีตตั้งแต่ช่วงปี 1980s แต่ในปัจจุบันถูกแทนที่ด้วยพอร์ตแบบดิจิทัล เช่น HDMI และ DisplayPort เนื่องจากสามารถส่งสัญญาณภาพที่มีความละเอียดและคุณภาพสูงกว่า

พอร์ต VGA มีลักษณะเป็นพอร์ต 15 ขา โดยส่งสัญญาณภาพในรูปแบบอนาล็อก และรองรับความละเอียดสูงสุดได้ถึง 2048x1536 พิกเซล อย่างไรก็ตาม ความชัดเจนและความคมชัดของภาพจะลดลงเมื่อใช้สายที่ยาวขึ้น

ข้างอิงรูปภาพ: https://www.asus.com/th/motherboards-components/motherboards/prime/prime-h370m-plus/

Back Panel Ports of Mainboard



พอร์ต PS/2 เป็นพอร์ตเชื่อมต่อแบบอนุกรมที่ใช้สำหรับการเชื่อมต่อแป้นพิมพ์ (keyboard) และเมาส์ (mouse) กับคอมพิวเตอร์ ถูกพัฒนาโดย IBM ในช่วงปลายปี 1980s และใช้เป็นมาตรฐานในคอมพิวเตอร์ส่วนใหญ่จนกระทั่ง USB เข้ามาแทนที่

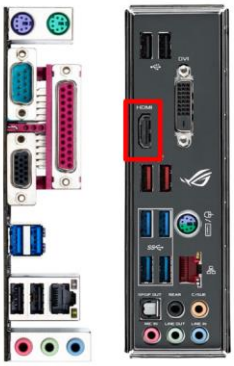
ลักษณะของพอร์ต PS/2 คือเป็นพอร์ตวงกลมที่มี 6 ขา และมีสีที่แยกกันเพื่อระบุการใช้งาน:

- สีม่วงสำหรับแป้นพิมพ์
- สีเขียวสำหรับเมาส์

ข้อดีของพอร์ต PS/2 คือสามารถสื่อสารได้โดยตรงกับระบบ BIOS ของคอมพิวเตอร์ ทำให้สามารถใช้งานได้แม้จะไม่มีไดรเวอร์ แต่ข้อเสียคือไม่รองรับการเสียบใช้งานขณะเปิดเครื่อง (hot-swapping) และปัจจุบัน PS/2 ถูกแทนที่ด้วย USB ในการเชื่อมต่อแป้นพิมพ์และเมาส์ส่วนใหญ่แล้ว

ข้างอิงรูปภาพ: https://www.asus.com/th/motherboards-components/motherboards/prime/prime-h370m-plus/

Back Panel Ports of Mainboard



HDMI (High-Definition Multimedia Interface) เป็น**มาตรฐานการเชื่อมต่อสำหรับการส่งสัญญาณภาพและเสียงในรูปแบบดิจิทัล** ใช้เชื่อมต่ออุปกรณ์ต่าง ๆ เช่น ทีวี จอคอมพิวเตอร์ เครื่องเล่นเกม คอมพิวเตอร์ เครื่องเล่น Blu-ray และโปรเจ็กเตอร์ โดยพอร์ต HDMI สามารถส่งสัญญาณได้ทั้งภาพและเสียงพร้อมกันในสายเดียว ซึ่งเป็นข้อได้เปรียบเมื่อเทียบกับพอร์ตแบบอนาล็อก เช่น VGA หรือ DVI ที่อาจต้องใช้สายแยกกันสำหรับภาพและเสียง

HDMI รองรับการส่งภาพที่ความละเอียดสูง (เช่น 1080p, 4K, 8K) และรองรับเสียงหลายช่องทาง (เช่น Dolby Digital, DTS) รวมถึงการส่งสัญญาณแบบ HDR (High Dynamic Range) เพื่อเพิ่มคุณภาพของภาพและความคมชัด

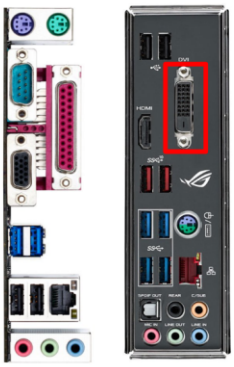
เวอร์ชันของ HDMI ที่พบบ่อย:

- **HDMI 1.4**: รองรับภาพ 4K ที่ 30 Hz
- **HDMI 2.0**: รองรับภาพ 4K ที่ 60 Hz
- **HDMI 2.1**: รองรับภาพ 8K ที่ 60 Hz และ 4K ที่ 120 Hz

HDMI ยังรองรับเทคโนโลยีอย่าง ARC (Audio Return Channel) สำหรับส่งสัญญาณเสียงกลับไปที่เครื่องเสียง และ eARC สำหรับการส่งเสียงคุณภาพสูงขึ้น

อ้างอิงรูปภาพ: https://www.asus.com/th/motherboards-components/motherboards/prime/prime-h370m-plus/

Back Panel Ports of Mainboard



DVI (Digital Visual Interface) เป็น**พอร์ตเชื่อมต่อสำหรับส่งสัญญาณภาพจากคอมพิวเตอร์ไปยังจอภาพ**, โปรเจ็กเตอร์ หรืออุปกรณ์แสดงผลอื่น ๆ โดยออกแบบมาเพื่อรองรับทั้งสัญญาณภาพแบบดิจิทัลและอนาล็อก ซึ่งทำให้มีความยืดหยุ่นในการใช้งานเมื่อเทียบกับพอร์ตแบบ VGA

DVI มีหลายประเภทที่รองรับการเชื่อมต่อสัญญาณแตกต่างกัน:

- **DVI-D**: ส่งสัญญาณดิจิทัลเท่านั้น
- **DVI-A**: ส่งสัญญาณอนาล็อกเท่านั้น
- **DVI-I**: รองรับทั้งสัญญาณดิจิทัลและอนาล็อก

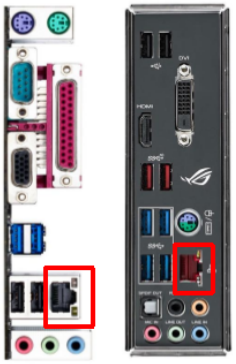
ความสามารถในการส่งสัญญาณภาพของ DVI:

- **Single Link**: รองรับความละเอียดสูงสุด 1920x1200 พิกเซล
- **Dual Link**: รองรับความละเอียดสูงสุด 2560x1600 พิกเซล

DVI ไม่รองรับการส่งสัญญาณเสียง ซึ่งแตกต่างจาก HDMI ที่สามารถส่งได้ทั้งภาพและเสียงในสายเดียว ปัจจุบัน DVI ยังคงพบใช้งานในบางอุปกรณ์ แต่ถูกแทนที่ด้วย HDMI และ DisplayPort ในหลาย ๆ กรณี

อ้างอิงรูปภาพ: https://www.asus.com/th/motherboards-components/motherboards/prime/prime-h370m-plus/

Back Panel Ports of Mainboard



LAN Port หรือ RJ-45 เป็น**พอร์ตที่ใช้ในการเชื่อมต่อเครือข่ายคอมพิวเตอร์**ผ่านสาย Ethernet โดยทำหน้าที่เชื่อมต่ออุปกรณ์ต่าง ๆ เช่น คอมพิวเตอร์ เราเตอร์ สวิตช์ หรือโมเด็ม เพื่อให้สามารถสื่อสารข้อมูลผ่านเครือข่ายแบบมีสาย (Wired Network) ได้

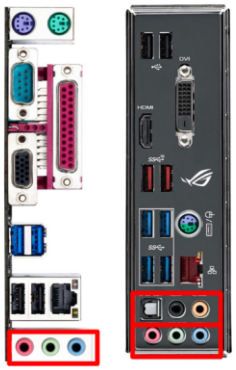
ลักษณะของ LAN Port เป็นพอร์ตสี่เหลี่ยมที่ใช้สายที่มีหัวต่อแบบ RJ-45 ซึ่งมี 8 ขาสำหรับส่งข้อมูล โดยพอร์ตนี้รองรับการเชื่อมต่อเครือข่ายด้วยความเร็วที่หลากหลาย เช่น:

- **10 Mbps** (Ethernet)
- **100 Mbps** (Fast Ethernet)
- **1 Gbps** (Gigabit Ethernet)
- **10 Gbps** (10-Gigabit Ethernet)

LAN Port เป็นวิธีการเชื่อมต่อที่มั่นคงและเร็วกว่าเครือข่ายไร้สาย (Wi-Fi) ในหลาย ๆ กรณี และมักใช้ในสภาพแวดล้อมที่ต้องการการเชื่อมต่อที่มีเสถียรภาพ เช่นในสำนักงาน เซิร์ฟเวอร์ หรือการใช้งานในบ้านที่ต้องการการถ่ายโอนข้อมูลปริมาณมาก

อ้างอิงรูปภาพ: https://www.asus.com/th/motherboards-components/motherboards/prime/prime-h370m-plus/

Back Panel Ports of Mainboard



Optical SPDIF (Sony/Philips Digital Interface Format) และ **Audio Port** เป็นพอร์ตที่ใช้ในการส่งสัญญาณเสียงระหว่างอุปกรณ์ต่าง ๆ แต่ทำงานในลักษณะที่แตกต่างกันตามเทคโนโลยีและการใช้งาน:

Optical SPDIF

- **SPDIF** เป็นมาตรฐานสำหรับการส่งสัญญาณเสียงในรูปแบบดิจิทัล โดยสามารถส่งสัญญาณเสียงคุณภาพสูง เช่น PCM, Dolby Digital หรือ DTS
- **Optical SPDIF** ใช้สายไฟเบอร์ออปติก (optical fiber) ในการส่งข้อมูล ซึ่งช่วยลดการรบกวนสัญญาณ (interference) ได้ดี
- นิยมใช้ในอุปกรณ์เครื่องเสียงโฮมเธียเตอร์ โทรทัศน์ หรือเครื่องเล่น Blu-ray เพื่อเชื่อมต่อกับลำโพงหรือระบบเสียงที่รองรับสัญญาณดิจิทัล
- ข้อดีคือการส่งสัญญาณเสียงดิจิทัลคุณภาพสูงและความสามารถในการส่งเสียงแบบหลายช่อง (multichannel audio)

Audio Port

- **Analog Audio Ports** เป็นพอร์ตสำหรับส่งสัญญาณเสียงแบบอนาล็อกที่ใช้ทั่วไปในอุปกรณ์คอมพิวเตอร์, โทรศัพท์มือถือ, ลำโพง, และหูฟัง
- ประเภทที่พบได้บ่อย:
  - **3.5mm Jack**: พอร์ตเล็กที่ใช้เชื่อมต่อหูฟังหรือไมโครโฟน มีสีเขียวสำหรับหูฟังและสีชมพูสำหรับไมโครโฟนในบางกรณี
  - **6.35mm Jack**: ใช้ในอุปกรณ์เครื่องเสียงระดับมืออาชีพ เช่น เครื่องขยายเสียงและเครื่องดนตรี
  - **RCA (Red and White)**: พอร์ตคู่ที่ใช้ในเครื่องเสียงสเตอริโอสำหรับส่งสัญญาณเสียงซ้ายและขวาแบบอนาล็อก

อ้างอิงรูปภาพ: https://www.asus.com/th/motherboards-components/motherboards/prime/prime-h370m-plus/

ฮาร์ดแวร์ (Hardware)

การ์ดกราฟิก (GPU): อุปกรณ์ที่ช่วยในการประมวลผลภาพและกราฟิก รวมไปถึงแบบจำลอง AI



อ้างอิงรูปภาพ: https://th.aorus.com/blog-detail.php?i=880

ฮาร์ดแวร์ (Hardware)

อุปกรณ์อินพุตและเอาต์พุต (I/O Devices): เช่น คีย์บอร์ด เมาส์ จอภาพ และเครื่องพิมพ์ ที่ใช้ในการป้อนและแสดงผลข้อมูล





อ้างอิงรูปภาพ:
https://www.bnn.in.th/th/p/gaming-gear/pc-gaming-accessories/gaming-keyboard/keychron-gaming-keyboard-k8-pro-frame-a-hot-swappable-red-switch-489524880689?_x9xl94
https://www.logitech.com/th-th/products/mice/m331-silent-plus-mouse.910-004914.html

ฮาร์ดแวร์ (Hardware)

Power Supply Unit (PSU): หน่วยจ่ายไฟ ทำหน้าที่จ่ายไฟฟ้าให้กับทุกส่วนของคอมพิวเตอร์

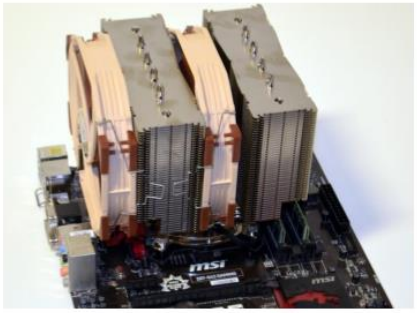


อ้างอิงรูปภาพ:
https://www.autonetpc.com/product/corsair-cv-series-cv650-650-watt-80-plus-bronze-certified-psu/

ฮาร์ดแวร์ (Hardware)

Cooling System: ระบบระบายความร้อน ใช้ในการระบายความร้อนจาก CPU และอุปกรณ์อื่น ๆ เพื่อป้องกันความร้อนสะสม





อ้างอิงรูปภาพ:
https://www.tweaktown.com/reviews/10141/corsair-icue-h150i-rgb-pro-xt-cpu-liquid-cooler/index.html
https://play3r.net/reviews/cooling/noctua-nh-d15-review/

Software

ซอฟต์แวร์ (Software)

ซอฟต์แวร์ (Software) ของคอมพิวเตอร์

หมายถึงชุดคำสั่งหรือโปรแกรมที่สั่งงานให้ฮาร์ดแวร์ทำงานตามที่ต้องการ ซอฟต์แวร์สามารถแบ่งออกเป็นหลายประเภทตามหน้าที่และการใช้งาน ได้แก่ ซอฟต์แวร์ระบบ (System Software) และซอฟต์แวร์ประยุกต์ (Application Software)



ซอฟต์แวร์ (Software)

1.ระบบปฏิบัติการ (OS : Operating system)

ระบบปฏิบัติการ (Operating System หรือ OS) เป็นซอฟต์แวร์พื้นฐานที่ทำหน้าที่เป็นตัวกลางระหว่างฮาร์ดแวร์ของคอมพิวเตอร์กับซอฟต์แวร์หรือโปรแกรมต่าง ๆ ระบบปฏิบัติการช่วยให้ผู้ใช้สามารถสั่งงานและควบคุมการทำงานของคอมพิวเตอร์ได้อย่างมีประสิทธิภาพ

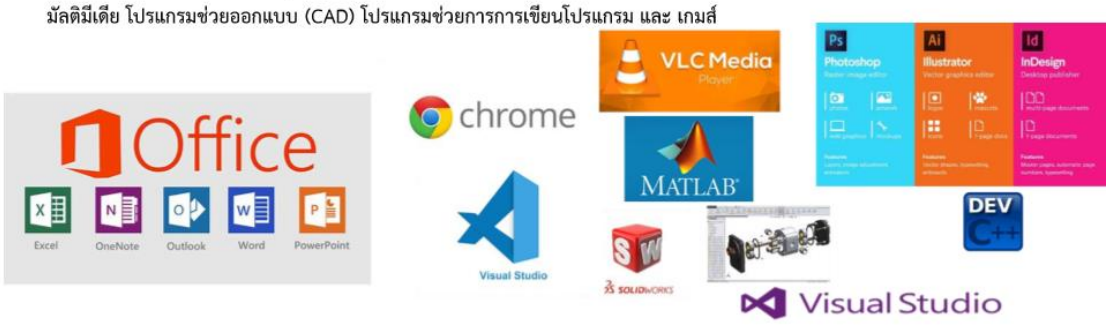
ระบบปฏิบัติการมีหน้าที่หลัก ๆ ได้แก่ การจัดการฮาร์ดแวร์ การจัดการไฟล์ การจัดการกระบวนการ (Process) การจัดการหน่วยความจำ การจัดการความปลอดภัย การจัดการเครือข่าย รวมไปถึงการมีโปรแกรมพื้นฐานให้ใช้งาน และโปรแกรมการตั้งค่าต่างๆ



ซอฟต์แวร์ (Software)

2.โปรแกรมประยุกต์ (Applications)

เป็นซอฟต์แวร์ที่พัฒนาเพื่อช่วยผู้ใช้ในการทำงานเฉพาะด้าน เช่น โปรแกรมสำนักงาน (Office Software) โปรแกรมกราฟิกและมัลติมีเดีย โปรแกรมช่วยออกแบบ (CAD) โปรแกรมช่วยการการเขียนโปรแกรม และ เกมส์



สรุป

Definition of a Computer: A computer is an electronic machine used for storing, organizing, and calculating data. It controls other machines with a set of instructions.

Evolution of Computers:

- **1st Generation (1951-1958)**: Vacuum tubes (ENIAC, UNIVAC I)
- **2nd Generation (1959-1963)**: Transistors (IBM 1620)
- **3rd Generation (1964-1970)**: Integrated Circuits (IBM System/360)
- **4th Generation (1971-1980)**: Microprocessors (Apple II, IBM PC)
- **5th Generation (1981-present)**: ULSI (Apple M2, Intel® Core™ Ultra 7)
- **6th Generation (future)**: AI and Quantum Computing (Nvidia DGX H200)

สรุป

Types of Computers:

Supercomputer
Mainframe Computer
Minicomputer
Microcomputer (PC)
Workstation
Laptop
Tablet
Smartphone
Server
Microcontroller

สรุป

Main Components of Computers:

- Hardware: CPU, RAM, storage devices, motherboards, GPU, I/O devices, power supply, cooling systems.
- Software: System software (Operating Systems) and Application software (Office tools, CAD, etc.).

Memory:

- Types: RAM (volatile) vs. ROM (non-volatile).
- Units of Memory: Bit, Byte, KB, MB, GB, TB.
- Character Encoding: ASCII, UTF-8.

Motherboard Components:

- Ports: USB, VGA, HDMI, LAN, Audio, DVI, Optical SPDIF.

Software:

- Operating Systems: Manages hardware, files, processes, memory, and security.
- Applications: Specific software for various user needs.

แหล่งศึกษาเพิ่มเติม

รศ.ดร.จันทนา จันทราพรชัย. (2555) สถาปัตยกรรมคอมพิวเตอร์และการออกแบบ (COMPUTER ARCHITECTURE AND DESING), สำนักพิมพ์ท้อป, ISBN: 978-616-282-009-0

รศ.ธีรวัฒน์ ประกอบผล. (2551) สถาปัตยกรรมคอมพิวเตอร์, สมาคมส่งเสริมเทคโนโลยี, ISBN: 978-974-443-369-5

ผศ.นราธร สังข์ประเสริฐ. (2563) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-3945-2

โอภาส เอี่ยมสิริวงศ์. (2559) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-2740-4

รศ.ธีรวัฒน์ ประกอบผล. (2562) คู่มือการเขียนโปรแกรมภาษา C ฉบับสมบูรณ์, ซิมพลิฟาย, ISBN: 978-616-26-24896

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase1] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=aDBmXMF9jOU

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase2] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=wslM0Aqepxs

Mooc การเขียนโปรแกรมคอมพิวเตอร์พื้นฐาน ด้วยภาษาซี 1 | Fundamentals of the C Programming Language https://lms.thaimooc.org/courses/course-v1:KMUTT+KMUTT013+2019/about

Mooc ทักษะการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี | Computer programming skills with C language https://lms.thaimooc.org/courses/course-v1:SU+SU011+2021/about

Coursera: Programming in C ผู้สอน: Charles Russell Severance https://www.coursera.org/learn/programming-c

Coursera: ความเชี่ยวชาญพิเศษ การเขียนโปรแกรม C เบื้องต้น ผู้สอน: Andrew D. Hilton et al. https://www.coursera.org/specializations/c-programming

แบบฝึกหัดท้ายบท

1.คอมพิวเตอร์คืออะไร

2.คอมพิวเตอร์ในยุคที่ 1 ใช้เทคโนโลยีอะไร

3.จงยกตัวอย่างคอมพิวเตอร์ในยุคหลอดสุญญากาศ (ยุคที่ 1)

4.คอมพิวเตอร์ในยุคที่ 2 ใช้เทคโนโลยีอะไรแทนหลอดสุญญากาศ

5.ลักษณะเด่นของคอมพิวเตอร์ในยุคที่ 2 คืออะไร

6.คอมพิวเตอร์ในยุคที่ 3 ใช้เทคโนโลยีวงจรแบบใด

7.จงอธิบายลักษณะเด่นของคอมพิวเตอร์ในยุคที่ 4

8.ยุคที่ 5 ของคอมพิวเตอร์มีชื่อเรียกว่าอะไร

9.คอมพิวเตอร์ในยุคที่ 6 ใช้เทคโนโลยีหลักอะไรบ้าง

10.หน่วยประมวลผลกลาง (CPU) ทำหน้าที่อะไร

11.องค์ประกอบใดของ CPU ที่ทำหน้าที่คำนวณทางคณิตศาสตร์และตรรกะ

12.RAM และ ROM แตกต่างกันอย่างไร

13.หน่วยของหน่วยความจำเล็กที่สุดคืออะไร

14.หน่วยความจำ 1 กิโลไบต์ (KB) เท่ากับกี่ไบต์ (Bytes)

15.หน่วยความจำแบบใดที่สามารถประมวลผลข้อมูลได้เร็วที่สุดในระบบคอมพิวเตอร์

16.พอร์ตอนุกรม (Serial Port) ทำงานอย่างไร

17.USB 3.0 มีความเร็วในการถ่ายโอนข้อมูลสูงสุดเท่าไร

18.อุปกรณ์อินพุต (Input Device) คืออะไร จงยกตัวอย่าง

19.พอร์ต HDMI ทำหน้าที่อะไร

20.จงอธิบายบทบาทของซอฟต์แวร์ระบบ (System Software)

แผนการสอนเรื่อง วิธีการออกแบบและพัฒนาโปรแกรม

สัปดาห์ที่ 2-3

จำนวนชั่วโมงที่ใช้สอน 7 ชั่วโมง

วัตถุประสงค์ เพื่อให้นักศึกษาสามารถ

1. ระบุขั้นตอนต่าง ๆ ของกระบวนการพัฒนาโปรแกรมได้ (เช่น การวางแผน การออกแบบ การทดสอบ และการแก้ไขข้อผิดพลาด)
2. อธิบายการทำงานของผังงานและรหัสเทียมในการออกแบบอัลกอริทึมได้
3. เขียนอัลกอริทึมเบื้องต้นโดยใช้รหัสเทียมหรือผังงานเพื่อแก้ปัญหาอย่างง่ายได้
4. ออกแบบ โปรแกรมใหม่จากอัลกอริทึมที่สร้างขึ้น โดยใช้ผังงานและรหัสเทียมที่เหมาะสม

หัวข้อบรรยาย

บทที่ 2 วิธีการออกแบบและพัฒนาโปรแกรม

2.1 ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

การวางแผน (Planning)

การวิเคราะห์ปัญหา (Analysis)

การออกแบบ (Design)

การพัฒนาโปรแกรม (Implementation)

การดูแลรักษาโปรแกรม (Maintenance)

2.2 การออกแบบอัลกอริทึมด้วยผังงาน (Flowchart)

2.3 การออกแบบอัลกอริทึมด้วยรหัสเทียม (Pseudocode)

กิจกรรมการเรียนการสอน

1. การบรรยายในชั้นเรียน
2. การให้ผู้เรียนทดลองเขียนรหัสต้นฉบับและทดลองรันโปรแกรมระหว่างบรรยาย
3. การให้ผู้เรียนทำใบฝึกปฏิบัติการ
4. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)

สื่อการสอน

1. เอกสารประกอบการสอนแบบภาพเลื่อนหรือภาพนิ่ง
2. ใบฝึกปฏิบัติการ
3. โปรแกรมพัฒนาโปรแกรมภาษาซี เช่น DevC++, VS Code เป็นต้น
4. เว็บไซต์ช่วยออกแบบผังงาน https://app.diagrams.net/
5. การเข้าถึงข้อมูลจากอินเทอร์เน็ต

การประเมินผลการเรียนรู้

ผลการเรียนรู้	วิธีการประเมิน	สัปดาห์ที่ประเมิน	สัดส่วนของการประเมินผล
1. คุณธรรม จริยธรรม	1.1 ประเมินจากการเข้าชั้นเรียน การส่งแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการตามกำหนดเวลาที่มอบหมาย 1.2 ประเมินจากพฤติกรรมของนักศึกษาในการปฏิบัติตามกฎระเบียบและข้อบังคับต่าง ๆ ทั้งในข้อบังคับในชั้นเรียน ข้อบังคับของมหาวิทยาลัย 1.3 ประเมินจากการคัดลอกแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการ 1.4 ประเมินจากการมีส่วนร่วมในห้องเรียนและการปิดคอมพิวเตอร์ เก็บขยะ เก็บเก้าอี้หลังจากใช้งานเสร็จ	ตลอดภาคการศึกษา	10%
2. ความรู้	2.1 แบบฝึกหัดท้ายบท 2.2 การสอบกลางภาคเรียนและปลายภาคเรียน	2-8, 11-16 9, 17	20% 40%
3. ทักษะทางปัญญา	3.1 ประเมินจากใบฝึกปฏิบัติการ	ตลอดภาคการศึกษา	20%
4. ทักษะความสัมพันธ์ระหว่างบุคคลและความรับผิดชอบ	4.1 พิจารณาจากการทำใบฝึกปฏิบัติการ 4.3 ประเมินจากพฤติกรรมในการทำงานกลุ่ม การระดมความคิดเห็น การตอบคำถามในชั้นเรียน	ตลอดภาคการศึกษา	5%
5. ทักษะวิเคราะห์เชิงตัวเลข การสื่อสารและการใช้เทคโนโลยีสารสนเทศ	5.1 การทดสอบหรือการประเมินจากงานที่มอบหมาย 5.2 สังเกตการใช้คอมพิวเตอร์ในการเขียนโปรแกรม การทำใบฝึกปฏิบัติการ และการค้นหาข้อมูลเพิ่มเติมจากอินเทอร์เน็ต	ตลอดภาคการศึกษา	5%
	รวม		100%

อ่าน Slide เวอร์ชันล่าสุดได้ที่ Google Slide

https://s.coecore.com/scp2

การโปรแกรมคอมพิวเตอร์

บทที่ 2 วิธีการออกแบบและพัฒนาโปรแกรม

(Algorithm Design and Program Development)

ผู้สอน ดร.พงศกร เจริญเนตรกุล

Outline

- ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์
- การออกแบบขั้นตอนวิธีด้วย**ผังงาน** (Flowchart)
- การออกแบบขั้นตอนวิธีด้วย**รหัสเทียม** (Pseudocode)

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

- การวางแผน (Planning)
- การวิเคราะห์ปัญหา (Analysis)
- การออกแบบ (Design)
- การพัฒนาโปรแกรม (Implementation)
  - การเขียนโปรแกรม (Coding)
  - การทดสอบโปรแกรม (Testing)
  - การแก้ไขข้อผิดพลาดของโปรแกรม (Debugging)
- การดูแลรักษาโปรแกรม (Maintenance)

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์
การวางแผน (Planning)

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

- การวางแผน (Planning)
- การวิเคราะห์ปัญหา (Analysis)
- การออกแบบ (Design)
- การพัฒนาโปรแกรม (Implementation)
  - การเขียนโปรแกรม (Coding)
  - การทดสอบโปรแกรม (Testing)
  - การแก้ไขข้อผิดพลาดของโปรแกรม (Debugging)
- การดูแลรักษาโปรแกรม (Maintenance)

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

การวางแผน (Planning)

- ปัญหาคืออะไร ? (ความต้องการของผู้ใช้งาน) - indentify the problems
- สิ่งจำเป็นเพื่อใช้แก้ปัญหา ? (วิธีแก้ไขปัญหา) - seek the proper solution
- ลำดับขั้นตอนในการแก้ปัญหา ? - algorithm



ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

การวางแผน (Planning)

ตัวอย่างในชีวิตประจำวัน เช่น **การวางแผนทำอาหารเย็น**

ปัญหาคืออะไร ? (Identify the problems)

ความต้องการของผู้ใช้งาน:

- ต้องการทำอาหารเย็นสำหรับครอบครัว 4 คน (Output)
- มีเวลาในการทำอาหารจำกัด (เช่น 1 ชั่วโมง) (with condition)
- ต้องการทำอาหารที่มีประโยชน์และเหมาะสมกับคนในครอบครัว (with condition)

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

การวางแผน (Planning)

ตัวอย่างในชีวิตประจำวัน เช่น **การวางแผนทำอาหารเย็น**

สิ่งจำเป็นเพื่อใช้แก้ปัญหา ? (Seek the proper solution)

วิธีแก้ไขปัญหา:

- หาสูตรอาหารที่ทำได้ภายใน 1 ชั่วโมง
- ตรวจสอบวัตถุดิบที่มีอยู่ในครัวและวัตถุดิบที่ต้องซื้อเพิ่มเติม
- จัดเตรียมอุปกรณ์ทำอาหาร เช่น กระทะ หม้อ เตา
- ตรวจสอบเวลาที่ใช้ในแต่ละขั้นตอนของการทำอาหารเพื่อให้ทันเวลา

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

การวางแผน (Planning)

ตัวอย่างในชีวิตประจำวัน เช่น **การวางแผนทำอาหารเย็น**

ลำดับขั้นตอนในการแก้ปัญหา ? (Algorithm)

1. **ระบุข้อจำกัดและความต้องการ**
   - เวลาในการทำอาหาร: 1 ชั่วโมง
   - จำนวนคน: 4 คน
   - ประเภทอาหาร: ต้องมีประโยชน์และเหมาะสมกับทุกคนในครอบครัว
2. **เลือกเมนูอาหาร**
   - หาสูตรอาหารที่เหมาะกับข้อจำกัดด้านเวลาและจำนวนคน เช่น ผัดผักกับข้าวสวย
3. **ตรวจสอบวัตถุดิบและอุปกรณ์**
   - ตรวจสอบว่ามีวัตถุดิบตามที่สูตรต้องการหรือไม่
   - ถ้าวัตถุดิบขาดหาย ต้องเตรียมลิสต์เพื่อไปซื้อเพิ่มเติม

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

การวางแผน (Planning)

ตัวอย่างในชีวิตประจำวัน เช่น **การวางแผนทำอาหารเย็น**

ลำดับขั้นตอนในการแก้ปัญหา ? (Algorithm)

4. วางแผนการทำอาหาร

- เริ่มด้วยการล้างและเตรียมวัตถุดิบ
- ตั้งหม้อหุงข้าวก่อนเพื่อให้ข้าวเสร็จพร้อมอาหาร
- ระหว่างหุงข้าว ให้เตรียมวัตถุดิบอื่นๆ เช่น หั่นผัก และจัดเตรียมเนื้อสัตว์

5. ลงมือทำตามลำดับขั้นตอนของสูตร

- ผัดผักและเนื้อสัตว์
- จัดจานและเสิร์ฟอาหาร

6. ตรวจสอบเวลาและปรับตามสถานการณ์

- ตรวจสอบเวลาที่ใช้ในแต่ละขั้นตอน เพื่อให้แน่ใจว่าอาหารเสร็จตรงเวลา

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

การวิเคราะห์ปัญหา (Analysis)

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

- การวางแผน (Planning)
- **การวิเคราะห์ปัญหา (Analysis)**
- การออกแบบ (Design)
- การพัฒนาโปรแกรม (Implementation)
  - การเขียนโปรแกรม (Coding)
  - การทดสอบโปรแกรม (Testing)
  - การแก้ไขข้อผิดพลาดของโปรแกรม (Debugging)
- การดูแลรักษาโปรแกรม (Maintenance)

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

การวิเคราะห์ปัญหา

สามารถใช้ **ระบบ (system)** ในการ identify ทั้ง **Input** , **Output** และ **Process** ได้

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

การซักผ้า

- Input: ผ้าที่สกปรก, ผงซักฟอก, น้ำ
- Output: ผ้าที่สะอาด

Process (กระบวนการแก้ปัญหา)

1. ใส่ผ้าในเครื่องซักผ้า
2. เติมผงซักฟอกและน้ำ
3. ตั้งโปรแกรมซัก
4. เครื่องซักผ้าทำงานตามเวลา
5. ปั่นผ้าแห้ง
6. เอาผ้าออกจากเครื่อง





Computer Programming 15

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

การชงกาแฟ

- Input: ผงกาแฟ, น้ำร้อน, แก้ว
- Output: กาแฟที่พร้อมดื่ม

Process (กระบวนการแก้ปัญหา)

1. ต้มน้ำให้เดือด
2. ใส่ผงกาแฟลงในแก้ว
3. เทน้ำร้อนใส่แก้ว
4. คนผสมให้เข้ากัน
5. เสิร์ฟกาแฟ



Computer Programming 16

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

การออกแบบ (Design)

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

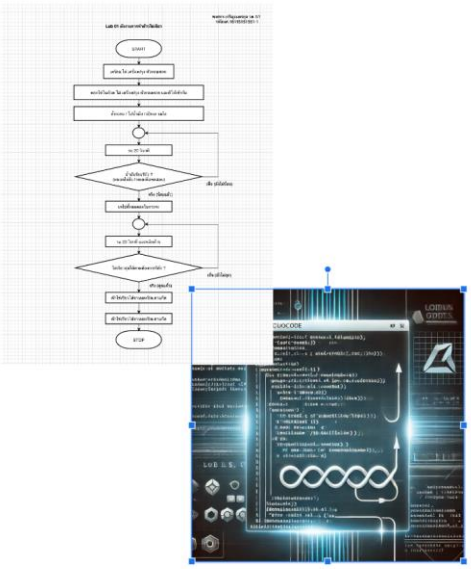
- การวางแผน (Planning)
- การวิเคราะห์ปัญหา (Analysis)
- **การออกแบบ (Design)**
- การพัฒนาโปรแกรม (Implementation)
  - การเขียนโปรแกรม (Coding)
  - การทดสอบโปรแกรม (Testing)
  - การแก้ไขข้อผิดพลาดของโปรแกรม (Debugging)
- การดูแลรักษาโปรแกรม (Maintenance)

การออกแบบ (Design)

อัลกอริทึม (Algorithm) = การอธิบายขั้นตอนการทำงานที่ชัดเจน เพื่อแก้ปัญหาหรือทำงานอย่างใดอย่างหนึ่ง โดยการเขียนสามารถใช้ 2 รูปแบบของผังงานต่อไปนี้ในการออกแบบได้

1.ผังลำดับงาน (Flowchart) คือ การใช้รูปภาพหรือแผนภาพแสดงขั้นตอนการแก้ปัญหาให้เห็นภาพได้ง่ายขึ้น

2.รหัสเทียม (Pseudocode) คือ การเขียนขั้นตอนการแก้ปัญหาโดยใช้ข้อความภาษาไทยหรือภาษาอังกฤษ ซึ่งใช้คำที่คล้ายกับภาษาโปรแกรมเพื่อให้เข้าใจโครงสร้างของโปรแกรมได้ง่าย



Computer Programming 19

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

การพัฒนาโปรแกรม (Implementation)

Computer Programming 20

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

- การวางแผน (Planning)
- การวิเคราะห์ปัญหา (Analysis)
- การออกแบบ (Design)
- **การพัฒนาโปรแกรม (Implementation)**
  - การเขียนโปรแกรม (Coding)
  - การทดสอบโปรแกรม (Testing)
  - การแก้ไขข้อผิดพลาดของโปรแกรม (Debugging)
- การดูแลรักษาโปรแกรม (Maintenance)

การพัฒนาโปรแกรม (Implementation)

การเขียนโปรแกรม (Coding)

การเขียนโปรแกรม (Coding) เป็นกระบวนการที่นำอัลกอริทึม (Algorithm) หรือขั้นตอนการแก้ปัญหาที่วางแผนไว้ มาสร้างเป็นโปรแกรมจริง โดยใช้ภาษาคอมพิวเตอร์ เช่น Python, C++, หรือ Java เพื่อสั่งให้คอมพิวเตอร์ทำงานตามที่กำหนด





ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

- การวางแผน (Planning)
- การวิเคราะห์ปัญหา (Analysis)
- การออกแบบ (Design)
- **การพัฒนาโปรแกรม (Implementation)**
  - การเขียนโปรแกรม (Coding)
  - การทดสอบโปรแกรม (Testing)
  - การแก้ไขข้อผิดพลาดของโปรแกรม (Debugging)
- การดูแลรักษาโปรแกรม (Maintenance)

การพัฒนาโปรแกรม (Implementation)

การทดสอบโปรแกรม (Tesing)

การทดสอบโปรแกรม (Testing) เป็นกระบวนการตรวจสอบว่าโปรแกรมที่เขียนขึ้นทำงานถูกต้องตามที่ออกแบบไว้หรือไม่ โดยการรันโปรแกรมและตรวจสอบผลลัพธ์ เพื่อหาข้อผิดพลาด (Bug) และปรับแก้ไขให้โปรแกรมทำงานได้อย่างมีประสิทธิภาพและไม่เกิดปัญหาในสถานการณ์ต่าง ๆ



การพัฒนาโปรแกรม (Implementation)

การทดสอบโปรแกรม (Tesing)

ข้อผิดพลาด (Bug) ของโปรแกรม

- ไวยากรณ์ภาษา (Syntax Error)
- ข้อผิดพลาดระหว่างรันโปรแกรม (Runtime Error)
- ข้อผิดพลาดที่เกิดจากตีความหมายปัญหาผิดไป (Logical Error)

การพัฒนาโปรแกรม (Implementation)

ข้อผิดพลาด (Bug) ของโปรแกรม

ข้อผิดพลาดของไวยากรณ์ภาษา (Syntax Error) เกิดขึ้นเมื่อเขียนโค้ดไม่ถูกต้องตามกฎของภาษาคอมพิวเตอร์ เช่น การพิมพ์คำสั่งผิด หรือลืมเครื่องหมายที่จำเป็น เช่นวงเล็บหรือจุดทศนิยม เมื่อเกิดข้อผิดพลาดนี้ โปรแกรมจะไม่สามารถทำงานได้เลยจนกว่าจะถูกแก้ไข

ตัวอย่าง:

c Copy code

```c
#include <stdio.h>

int main() {
    printf("Hello, World!"  // ลืมปิดวงเล็บท้ายคำสั่ง
    return 0;
}
```

การพัฒนาโปรแกรม (Implementation)

ข้อผิดพลาด (Bug) ของโปรแกรม

ข้อผิดพลาดระหว่างรันโปรแกรม (Runtime Error)

Runtime Error เกิดขึ้นเมื่อโปรแกรมรันแล้ว แต่เผิญต้นไปเจอกับข้อผิดพลาด เช่น การหารด้วยศูนย์ หน่วยความจำเต็ม

ตัวอย่าง:

```c
#include <stdio.h>

int main() {
    int x = 10;
    int y = 0;
    int result = x / y;  // เกิดข้อผิดพลาดเนื่องจากหารด้วยศูนย์
    printf("Result: %d\n", result);
    return 0;
}
```


การพัฒนาโปรแกรม (Implementation)

ข้อผิดพลาด (Bug) ของโปรแกรม

ข้อผิดพลาดที่เกิดจากตีความหมายปัญหาผิดไป (Logical Error)

Logical Error คือข้อผิดพลาดที่เกิดจากการเขียนโค้ดตามตรรกะที่ผิด แม้โปรแกรมจะทำงานได้แต่ผลลัพธ์ที่ได้ไม่ถูกต้อง

ตัวอย่าง:

```c
#include <stdio.h>

int main() {
    int x = 5;
    int y = 10;
    int result = x - y;  // ตั้งใจจะบวกแต่เขียนเป็นลบ
    printf("Result: %d\n", result);
    return 0;
}
```

การพัฒนาโปรแกรม (Implementation)

ข้อผิดพลาด (Bug) ของโปรแกรม

สรุป

Syntax Error เขียนโค้ดผิดกฎภาษา เช่น ลืมเครื่องหมายปิดวงเล็บ

Runtime Error ข้อผิดพลาดที่เกิดขณะรันโปรแกรม เช่น การหารด้วยศูนย์

Logical Error ข้อผิดพลาดที่เกิดจากตรรกะผิดพลาด เช่น ใช้เครื่องหมายผิด

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

- การวางแผน (Planning)
- การวิเคราะห์ปัญหา (Analysis)
- การออกแบบ (Design)
- **การพัฒนาโปรแกรม (Implementation)**
  - การเขียนโปรแกรม (Coding)
  - การทดสอบโปรแกรม (Testing)
  - การแก้ไขข้อผิดพลาดของโปรแกรม (Debugging)
- การดูแลรักษาโปรแกรม (Maintenance)

การพัฒนาโปรแกรม (Implementation)

การแก้ไขข้อผิดพลาดของโปรแกรม (Debugging)

เป็น กระบวนการค้นหาและแก้ไขข้อผิดพลาดในโปรแกรม เพื่อให้โปรแกรมทำงานได้อย่างถูกต้องและมีประสิทธิภาพมากขึ้น โดยเริ่มจากการตรวจสอบว่าข้อผิดพลาดเกิดขึ้นที่ไหนและเกิดจากอะไร จากนั้นทำการแก้ไขข้อผิดพลาดเหล่านั้น

ตัวอย่างขั้นตอนการ Debugging:

1. **ระบุปัญหา:** ตรวจสอบว่าโปรแกรมเกิดข้อผิดพลาดที่ส่วนไหน
2. **วิเคราะห์ปัญหา:** ตรวจสอบสาเหตุที่ทำให้เกิดข้อผิดพลาด เช่น โค้ดที่เขียนผิดหรือการใช้งานค่าผิดพลาด
3. **แก้ไขปัญหา:** ทำการแก้ไขโค้ดที่มีปัญหา
4. **ทดสอบโปรแกรม:** รันโปรแกรมใหม่เพื่อตรวจสอบว่าข้อผิดพลาดถูกแก้ไขหรือไม่
5. **ตรวจสอบผลลัพธ์:** ตรวจดูว่าผลลัพธ์ที่ได้ถูกต้องตามที่คาดหวังหรือไม่

Computer Programming 31

การดูแลรักษาโปรแกรม (Maintenance)

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์

- การวางแผน (Planning)
- การวิเคราะห์ปัญหา (Analysis)
- การออกแบบ (Design)
- **การพัฒนาโปรแกรม (Implementation)**
  - การเขียนโปรแกรม (Coding)
  - การทดสอบโปรแกรม (Testing)
  - การแก้ไขข้อผิดพลาดของโปรแกรม (Debugging)
- การดูแลรักษาโปรแกรม (Maintenance)

การดูแลรักษาโปรแกรม (Maintenance)

การดูแลรักษาโปรแกรม (Maintenance) เป็นกระบวนการปรับปรุงและอัปเดตโปรแกรมหลังจากที่โปรแกรมถูกพัฒนาและนำไปใช้งานแล้ว เพื่อให้โปรแกรมทำงานได้อย่างมีประสิทธิภาพและตอบสนองต่อการเปลี่ยนแปลงในอนาคต เช่น การแก้ไขบั๊ก การเพิ่มฟีเจอร์ใหม่ หรือการปรับให้เข้ากับเทคโนโลยีที่เปลี่ยนแปลงไป

ตัวอย่างการดูแลรักษาโปรแกรม

การแก้ไขข้อผิดพลาด (Corrective Maintenance) การปรับปรุงโปรแกรมเพื่อแก้ไขบั๊กหรือข้อผิดพลาดที่พบหลังการใช้งาน

การปรับปรุงเพื่อประสิทธิภาพ (Perfective Maintenance) การเพิ่มฟีเจอร์ใหม่หรือปรับปรุงฟังก์ชันที่มีอยู่ให้ใช้งานได้ดีขึ้น

การปรับปรุงเพื่อความเข้ากันได้ (Adaptive Maintenance) การปรับโปรแกรมให้เข้ากับฮาร์ดแวร์หรือซอฟต์แวร์รุ่นใหม่ เช่น การอัปเดตโปรแกรมให้เข้ากับระบบปฏิบัติการใหม่

การป้องกันปัญหาในอนาคต (Preventive Maintenance) การปรับปรุงโค้ดเพื่อป้องกันปัญหาที่อาจเกิดขึ้นในอนาคต เช่น การทำให้โค้ดง่ายต่อการปรับแก้ไข

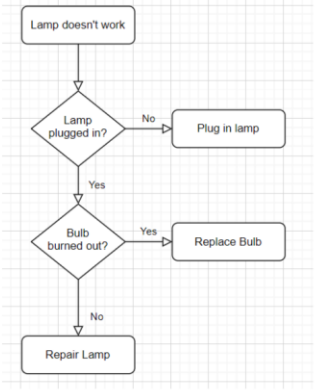




ผังงาน (Flowchart)

ผังงาน (Flowchart)

ผังงาน = การใช้รูปภาพหรือแผนภาพซึ่งเป็นสัญลักษณ์ทรงเรขาคณิตต่างๆ ที่แสดงขั้นตอนการแก้ปัญหาตามลำดับให้เห็นภาพขั้นตอนการทำงานได้ง่ายขึ้น



ผังงาน (Flowchart)

ประเภทของผังงาน มี 2 ประเภท ดังนี้

1.ผังงานระบบ (System Flowchart)

เป็นแผนภาพที่ใช้แสดงภาพรวมของขั้นตอนการทำงานภายในระบบต่าง ๆ โดยเน้นการอธิบายขั้นตอนใหญ่ ๆ และการเชื่อมโยงระหว่างแต่ละส่วนของระบบงาน ตั้งแต่การรับข้อมูลเข้า (Input) ไปจนถึงการประมวลผล (Process) และการแสดงผลลัพธ์ (Output) โดยจะอธิบายขั้นตอนในเชิงกว้าง ซึ่งช่วยให้ผู้ใช้เห็นภาพรวมของการทำงานของระบบทั้งหมด รวมถึงการเชื่อมโยงระหว่างอุปกรณ์หรือระบบย่อย ๆ ต่าง ๆ

ตัวอย่างผังงานระบบ ในระบบการสั่งซื้อสินค้าออนไลน์ ผังงานระบบจะแสดงขั้นตอน เช่น การกรอกข้อมูลสินค้า, การประมวลผลคำสั่งซื้อ, การชำระเงิน, และการจัดส่งสินค้า โดยแสดงการเชื่อมโยงระหว่างส่วนต่าง ๆ ของระบบ เช่น การเชื่อมต่อกับฐานข้อมูล การประมวลผลคำสั่งซื้อ และการแจ้งเตือนผู้ใช้

ผังงาน (Flowchart)

2.ผังงานโปรแกรม (Program Flowchart)

ผังงานโปรแกรม (Program Flowchart) เป็นแผนภาพที่แสดงรายละเอียดขั้นตอนของการทำงานภายในโปรแกรม โดยจะลงลึกในระดับของคำสั่งที่โปรแกรมเมอร์ใช้เพื่อแก้ปัญหา โดยการแสดงแต่ละขั้นตอนของการทำงานจะละเอียดมากขึ้น เช่น การคำนวณค่า การตรวจสอบเงื่อนไข การวนซ้ำ (loop) และการแสดงผล ข้อมูลในผังงานโปรแกรมช่วยให้ผู้เขียนโปรแกรมและผู้ตรวจสอบสามารถเข้าใจลำดับคำสั่งและโครงสร้างการทำงานของโปรแกรมได้ง่ายขึ้น

ตัวอย่างผังงานโปรแกรม ในการเขียนโปรแกรมเพื่อคำนวณผลรวมของตัวเลข 1 ถึง 100 ผังงานโปรแกรมจะระบุขั้นตอนที่ละเอียด เช่น เริ่มจากการกำหนดตัวแปรเริ่มต้น, การตรวจสอบเงื่อนไขว่าตัวเลขถึง 100 แล้วหรือยัง, การบวกค่าลงในตัวแปรผลรวม, และการแสดงผลลัพธ์เมื่อการคำนวณเสร็จสิ้น

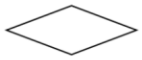
ผังงาน (Flowchart)

สัญลักษณ์มาตรฐานผังงานโปรแกรม

สัญลักษณ์ของผังงาน	ความหมาย
	**จุดเริ่มต้น /จุดสิ้นสุดของผังงาน** ต้องใส่ ณ จุดแรก และสุดท้ายของผังงานเสมอ Start Stop
	**กระบวนการ / การดำเนินการ / การประมวลผล / การคำนวณ / กำหนดค่า** อย่างใดอย่างหนึ่งตามที่กล่าวมา เช่น ประกาศตัวแปร คำนวณสูตรทางคณิตศาสตร์ เรียกใช้คำสั่ง PI = 3.1416 AreaCircle = PI x R^2
	**แสดงข้อความและค่าของตัวแปรผ่านทางจอภาพ** Hello JohnWick PI

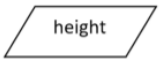
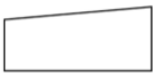
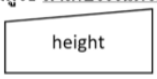
ผังงาน (Flowchart)

สัญลักษณ์มาตรฐานผังงานโปรแกรม

สัญลักษณ์ของผังงาน	ความหมาย
	**ใช้ในการตัดสินใจ** หรือ หลายทางเลือก เช่น ถ้า อุณหภูมิมากกว่า 27 ให้เปิดแอร์ เปิดแอร์ จริง Temp > 27 เท็จ เปิดพัดลม
	**จุดเชื่อมต่อภายในหน้าเดียวกัน** ใช้ในการเชื่อมรวมเส้นทางที่มาจากหลายทางให้ไปทางเดียวกัน เข้ามาได้หลายทางแต่ต้องออกแค่หนึ่งทางเสมอ 

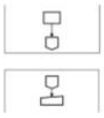
ผังงาน (Flowchart)

สัญลักษณ์มาตรฐานผังงานโปรแกรม

สัญลักษณ์ของผังงาน	ความหมาย
	นำเข้าข้อมูล (มีอยู่แล้ว เช่น ข้อมูลจากเครือข่าย จากไฟล์ จากฐานข้อมูล) หรือนำเข้าข้อมูลที่ไม่ระบุวิธีการหรือแหล่งข้อมูล มาเก็บไว้ในตัวแปรของโปรแกรม 
	การรับข้อมูลเข้าทางแป้นพิมพ์ จากการกด keyboard ของผู้ใช้ มาเก็บไว้ในตัวแปรของโปรแกรม เช่นรับค่าส่วนสูงมาเก็บไว้ในตัวแปร height 

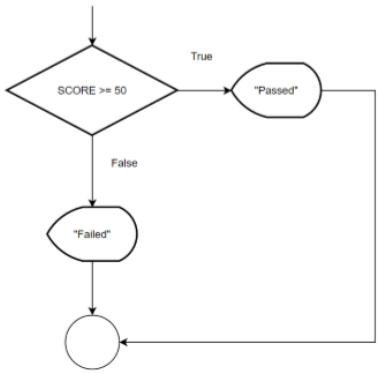
ผังงาน (Flowchart)

สัญลักษณ์มาตรฐานผังงานโปรแกรม

สัญลักษณ์ของผังงาน	ความหมาย
	แสดงทิศทางการทำงานของผังงาน (บน ล่าง ซ้าย ขวา เท่านั้น ไม่มีเส้นเฉียง)
	**จุดเชื่อมต่อระหว่างหน้า** หากเขียนผังงานไม่พอในหนึ่งหน้าให้เขียนอักษรใดสักตัวหนึ่งล้อมด้วยสัญลักษณ์นี้ไว้ในหน้าเดิมและในกระดาษแผ่นใหม่ให้เขียนอักษรเดิมในหน้าที่แล้วล้อมด้วยสัญลักษณ์นี้และดำเนินการต่อผังงานจากจุดนี้ได้เลย 

ผังงาน (Flowchart)

สัญลักษณ์การตัดสินใจ (Decision)



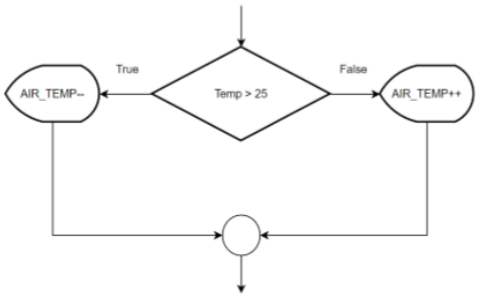
Flowchart นี้แสดงกระบวนการตัดสินใจว่าผู้ใช้งานสอบผ่านหรือไม่

เปรียบเทียบค่า ตรวจสอบว่า "SCORE" มากกว่าหรือเท่ากับ 50 หรือไม่

- ถ้า **True** (จริง) จะแสดงคำว่า **"Passed"** (ผ่าน)
- ถ้า **False** (ไม่จริง) จะแสดงคำว่า **"Failed"** (ไม่ผ่าน)

ผังงาน (Flowchart)

สัญลักษณ์การตัดสินใจ (Decision)



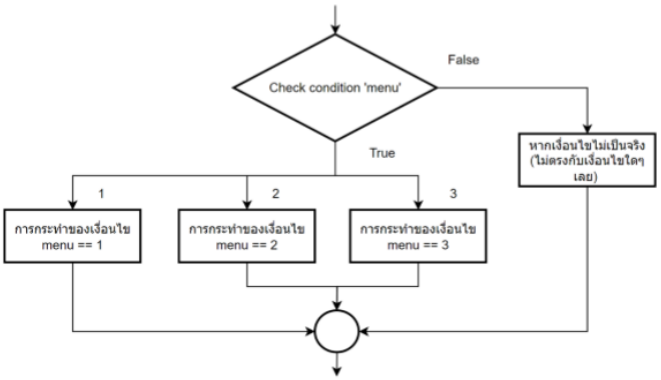
Flowchart นี้แสดงกระบวนการปรับอุณหภูมิของเครื่องปรับอากาศตามเงื่อนไขดังนี้:

เปรียบเทียบค่าอุณหภูมิ (Temp > 25)

- ถ้า **True** (อุณหภูมิมากกว่า 25): ลดค่าอุณหภูมิลง (AIR_TEMP--)
- ถ้า **False** (อุณหภูมิน้อยกว่าหรือเท่ากับ 25): เพิ่มค่าอุณหภูมิขึ้น (AIR_TEMP++)

ผังงาน (Flowchart)

สัญลักษณ์การตัดสินใจ (Decision) แบบหลายทางเลือก



เปรียบเทียบหลายเงื่อนไขแบบ switch-case

ถ้า menu == 1 จริง(True)
ให้ดำเนินการบางอย่างของทางเลือก == 1

ถ้า menu == 2 จริง(True)
ให้ดำเนินการบางอย่างของทางเลือก == 2

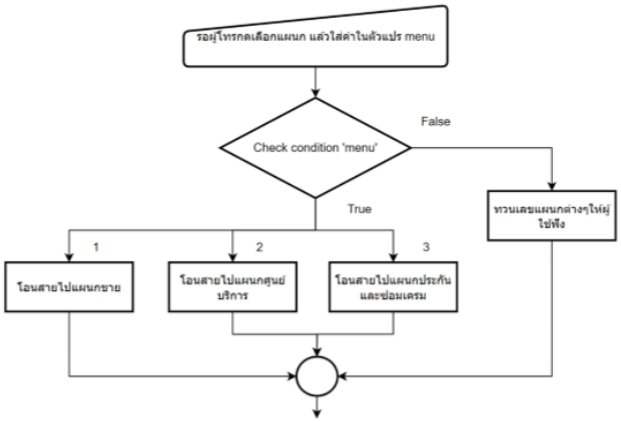
ถ้า menu == 3 จริง(True)
ให้ดำเนินการบางอย่างของทางเลือก == 3

ถ้า menu == 4 จริง(True)
ให้ดำเนินการบางอย่างของทางเลือก == 4

ถ้าไม่จริง (False) ซึ่งไม่ตรงกับเงื่อนไขใดๆเลย สามารถให้กระทำบางอย่าง เช่น บอกว่าตัวเลือก/เงื่อนไขที่เลือกมาไม่มีอยู่

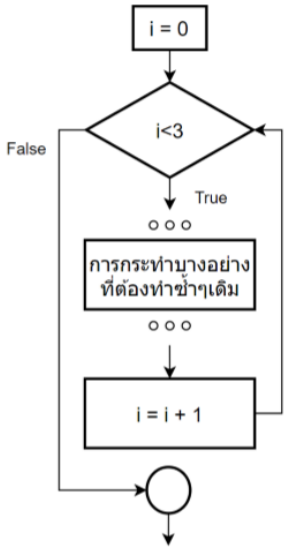
ผังงาน (Flowchart)

ตัวอย่าง สัญลักษณ์การตัดสินใจ (Decision) แบบหลายทางเลือก กรณีระบบโทรศัพท์สำนักงาน



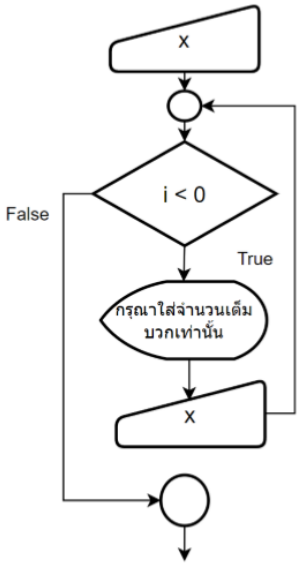
ผังงาน (Flowchart)

การใช้สัญลักษณ์การตัดสินใจ (Decision) เพื่อวนซ้ำการทำงาน (Loop) แบบ For



ผังงาน (Flowchart)

การใช้สัญลักษณ์การตัดสินใจ (Decision) เพื่อวนซ้ำการทำงาน (Loop) แบบ while



หลักการเขียนผังงาน

- เขียนผังงานจากบนลงล่างหรือจากซ้ายไปขวา
- ใช้สัญลักษณ์ที่เป็นมาตรฐานเท่านั้น
- อธิบายสั้น กระชับ และเข้าใจง่าย ถ้าอธิบายไม่หมด ใช้เส้นประโยงไปยังพื้นที่ว่างข้างขวาเพื่ออธิบายเพิ่มเติม
- ทุกผังงานต้องมีลูกศรเพื่อแสดงทิศทางของกระบวนการ
- หากเป็นเงื่อนไข ต้องมีการกำหนด True False ด้วย
- หากต้องเชื่อมเส้นที่ไกล ควรใช้จุดเชื่อมต่อแทน และหลีกเลี่ยงเส้นไขว้กัน
- ก่อนนำผังงานไปเขียนเป็นโปรแกรม ควรตรวจสอบความถูกต้องของการทำงานเสมอ

ประโยชน์ของผังงาน

1. ช่วยลำดับขั้นตอนการทำงานของโปรแกรมอย่างเป็นระบบ
2. สามารถแปลงเป็นโปรแกรมได้ง่ายและไม่สับสน
3. ช่วยให้ตรวจสอบและแก้ไขโปรแกรมได้สะดวกเมื่อเกิดข้อผิดพลาด
4. ช่วยในการดัดแปลงและพัฒนาโปรแกรมได้อย่างรวดเร็ว
5. ช่วยให้การสื่อสารและการศึกษาร่วมกันง่ายขึ้นด้วยมาตรฐานเดียวกัน

Question?

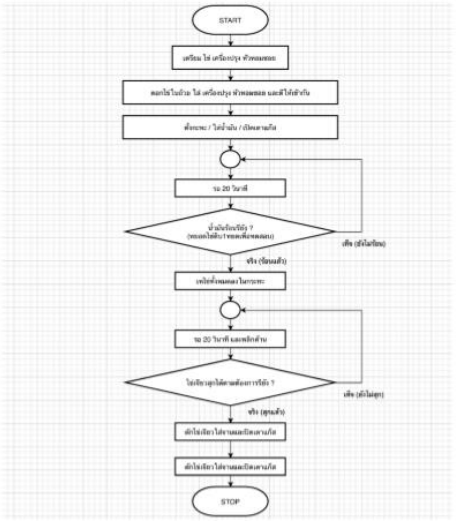
ควรเขียนผังงานก่อนหรือหลัง**เขียนโปรแกรม เพราะเหตุใด**

การนำอัลกอริทึ่มไปวางแผนสิ่งต่างๆในชีวิตประจำวัน

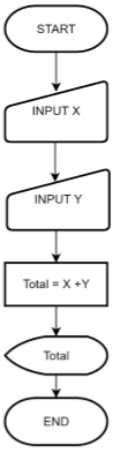
- การดำเนินชีวิตที่มีประสิทธิภาพ ควรมีการวางแผนอย่างมีระเบียบและมีแบบแผน เพื่อให้สามารถจัดการกับกิจกรรมต่างๆ ได้อย่างมีประสิทธิภาพ และมีความสามารถในการแก้ปัญหาที่เกิดขึ้นได้ดี สิ่งสำคัญคือต้องรู้จักตั้งเป้าหมายในชีวิตอย่างชัดเจน ไม่ว่าจะเป็นด้านการงาน ความสัมพันธ์ หรือสุขภาพ และวางแผนวิธีการที่จะทำให้ไปถึงเป้าหมายเหล่านั้น
- เมื่อตั้งเป้าหมายแล้ว การมีวิธีจัดการชีวิตที่ดีจะช่วยให้เราสามารถรับมือกับความท้าทายหรืออุปสรรคได้ ไม่ว่าจะเป็นการจัดการเวลา การทำงานอย่างเป็นระบบ หรือการเรียนรู้ที่จะปรับตัวและแก้ไขปัญหาอย่างเหมาะสม
- **สรุปก็คือ การดำเนินชีวิตอย่างมีแบบแผน คือการรู้จักตั้งเป้าหมาย วางแผน และมีความยืดหยุ่นในการจัดการและแก้ปัญหาในระหว่างทาง เพื่อไปสู่เป้าหมายของชีวิตที่ตั้งไว้**

ตัวอย่างผังงานในชีวิตประจำวัน

การทำข้าวไข่เจียว



ตัวอย่างผังงานโปรแกรมบวกเลขสองจำนวน




```c
#include <stdio.h>

int main() {
    // ประกาศตัวแปร
    int x, y, total;

    // รับค่าจากผู้ใช้
    printf("INPUT X: ");
    scanf("%d", &x);

    printf("INPUT Y: ");
    scanf("%d", &y);

    // บวกค่า X และ Y
    total = x + y;

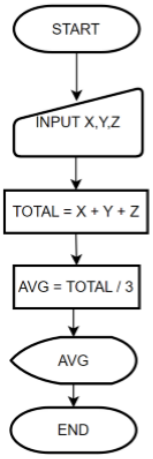
    // แสดงผลลัพธ์
    printf("Total = %d\n", total);

    return 0;
```

```
!gcc -Wall lab.c -o run
!./run
```

```
INPUT X: 4
INPUT Y: 3
Total = 7
```

ตัวอย่างผังงานโปรแกรมหาค่าเฉลี่ยจากเลข 3 ค่า




```c
#include <stdio.h>

int main() {
    // ประกาศตัวแปร
    float x, y, z, total, avg;

    // รับค่าจากผู้ใช้
    printf("Enter value for X,Y,Z: ");
    scanf("%f,%f,%f", &x,&y,&z);

    // บวกค่า X, Y และ Z
    total = x + y + z;

    // หาค่าเฉลี่ย
    avg = total / 3;

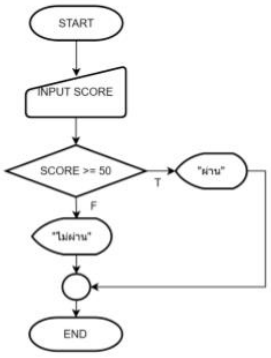
    // แสดงผลลัพธ์ค่าเฉลี่ย
    printf("Average = %.2f\n", avg);

    return 0;
}
```

```
!gcc -Wall lab.c -o run
!./run
```

```
Enter value for X,Y,Z: 1,2,3
Average = 2.00
```


ตัวอย่างผังงานโปรแกรมตัดสินคะแนนผ่านไม่ผ่าน




```c
#include <stdio.h>

int main() {
    // ประกาศตัวแปร
    int score;

    // รับค่าคะแนนจากผู้ใช้
    printf("Enter your score: ");
    scanf("%d", &score);

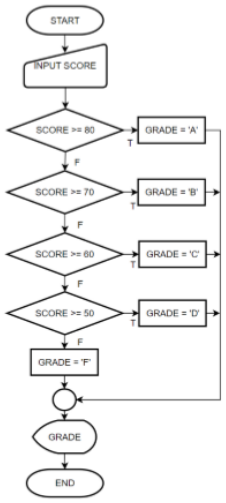
    // ตรวจสอบว่าคะแนนมากกว่าหรือเท่ากับ 50 หรือไม่
    if (score >= 50) {
        printf("ผ่าน\n");  // ถ้าคะแนนมากกว่าหรือเท่ากับ 50 ให้แสดงผล "ผ่าน"
    } else {
        printf("ไม่ผ่าน\n");  // ถ้าคะแนนน้อยกว่า 50 ให้แสดงผล "ไม่ผ่าน"
    }

    return 0;
}
```

```
!gcc -Wall lab.c -o run
!./run
```

```
Enter your score: 50
ผ่าน
```

ตัวอย่างผังงานโปรแกรมตัดเกรด A B C D F จากคะแนน




```c
#include <stdio.h>

int main() {
    // ประกาศตัวแปร
    int score;
    char grade;

    // รับค่าคะแนนจากผู้ใช้
    printf("Enter your score: ");
    scanf("%d", &score);

    // ตรวจสอบเกรดตามคะแนน
    if (score >= 80) {
        grade = 'A';
    } else if (score >= 70) {
        grade = 'B';
    } else if (score >= 60) {
        grade = 'C';
    } else if (score >= 50) {
        grade = 'D';
    } else {
        grade = 'F';
    }

    // แสดงผลเกรด
    printf("Your grade is: %c\n", grade);

    return 0;
}
```

```
Overwriting lab.c
```

```
!gcc -Wall lab.c -o run
!./run
```

```
Enter your score: 70
Your grade is: B
```


รหัสเทียม (Pseudocode)

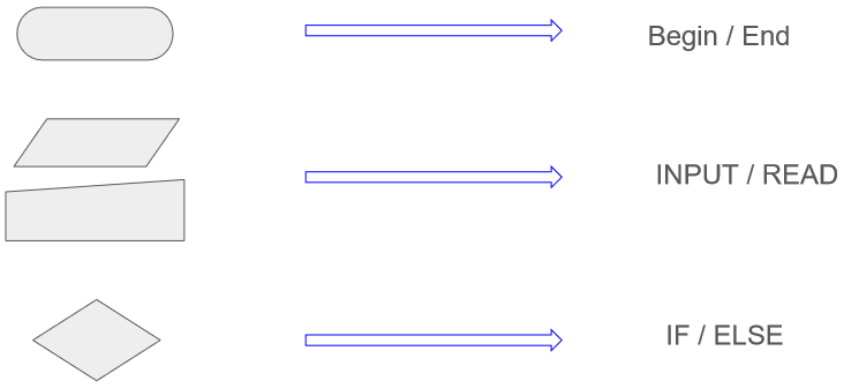
รหัสเทียม (Pseudocode)

- การออกแบบอัลกอริทึม(ขั้นตอนวิธี)ในรูปแบบประโยคภาษาอังกฤษตามมาตรฐาน ที่มีรูปแบบที่ตายตัว สั้นๆ เข้าใจได้ง่าย ซึ่งจะมีลักษณะคล้ายๆ ภาษาที่ใช้สำหรับเขียนโปรแกรม

```
Algorithm averaging 3 numbers
        Variables : number1, number2, number3, total, average
        Begin
                INPUT number1, number2, number3
                total = number1 + number2 + number3
                average = total / 3
                PRINT average
        End averaging 3 numbers
```

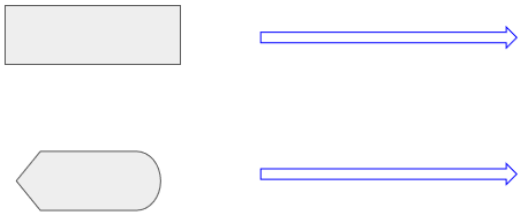
Computer Programming 59

รหัสเทียม (Pseudocode)



Computer Programming 60

รหัสเทียม (Pseudocode)



การกำหนดค่า (Expression)
เช่น X = 1

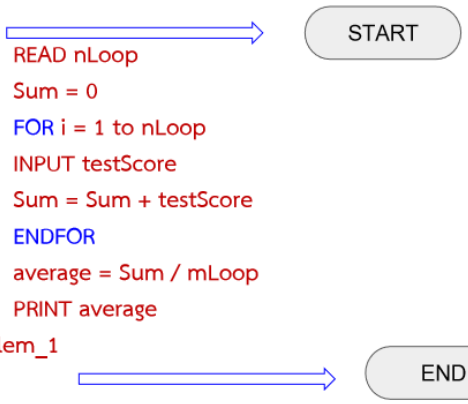
PRINT

รหัสเทียม (Pseudocode)

1. ใช้เพื่ออธิบายขั้นตอนการทำงานของโปรแกรมอย่างง่าย ๆ
2. ไม่ต้องเขียนตามกฎไวยากรณ์ของภาษาโปรแกรม
3. เข้าใจง่าย ใช้ภาษาธรรมดาผสมกับการเขียนเชิงโปรแกรม
4. ช่วยวางแผนก่อนการเขียนโค้ดจริง
5. เน้นที่ลำดับการทำงาน ไม่ใช่รายละเอียดการทำงานของโปรแกรม
6. เหมาะสำหรับการอธิบายความคิดเชิงตรรกะหรือขั้นตอนในการแก้ปัญหา
7. ไม่สามารถนำรหัสเทียมไปรันเพื่อให้ทำงานในภาษาโปรแกรมใดๆได้

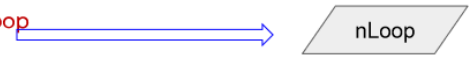
ตัวอย่างรหัสเทียม (Pseudocode)

```
Algorithm Problem_1
        Variables : nLoop, Sum, testScore, average
        Begin
                READ nLoop
                Sum = 0
                FOR i = 1 to nLoop
                INPUT testScore
                Sum = Sum + testScore
                ENDFOR
                average = Sum / mLoop
                PRINT average
        End Problem_1
```



ตัวอย่างรหัสเทียม (Pseudocode)

```
Algorithm Problem_1
        Variables : nLoop, Sum, testScore, average
        Begin
                READ nLoop
                Sum = 0
                FOR i = 1 to nLoop
                INPUT testScore
                Sum = Sum + testScore
                ENDFOR
                average = Sum / nLoop
                PRINT average
        End Problem_1
```



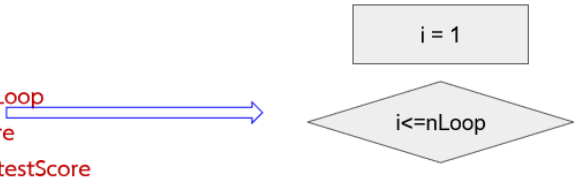
ตัวอย่างรหัสเทียม (Pseudocode)

```
Algorithm Problem_1
        Variables : nLoop, Sum, testScore, average
        Begin
                READ nLoop
                Sum = 0
                FOR i = 1 to nLoop
                INPUT testScore
                Sum = Sum + testScore
                ENDFOR
                average = Sum / nLoop
                PRINT average
        End Problem_1
```



ตัวอย่างรหัสเทียม (Pseudocode)

```
Algorithm Problem_1
        Variables : nLoop, Sum, testScore, average
        Begin
                READ nLoop
                Sum = 0
                FOR i = 1 to nLoop
                INPUT testScore
                Sum = Sum + testScore
                ENDFOR
                average = Sum / nLoop
                PRINT average
        End Problem_1
```



ตัวอย่างรหัสเทียม (Pseudocode)

```
Algorithm Problem_1
        Variables : nLoop, Sum, testScore, average
        Begin
                READ nLoop
                Sum = 0
                FOR i = 1 to nLoop
                INPUT testScore
                Sum = Sum + testScore
                ENDFOR
                average = Sum / nLoop
                PRINT average
        End Problem_1
```



ตัวอย่างรหัสเทียม (Pseudocode)

```
Algorithm Problem_1
        Variables : nLoop, Sum, testScore, average
        Begin
                READ nLoop
                Sum = 0
                FOR i = 1 to nLoop
                INPUT testScore
                Sum = Sum + testScore
                ENDFOR
                average = Sum / nLoop
                PRINT average
        End Problem_1
```



ตัวอย่างรหัสเทียม (Pseudocode)

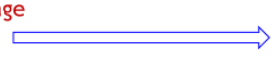
```
Algorithm Problem_1
        Variables : nLoop, Sum, testScore, average
        Begin
                READ nLoop
                Sum = 0
                FOR i = 1 to nLoop
                INPUT testScore
                Sum = Sum + testScore
                ENDFOR
                average = Sum / nLoop
                PRINT average
        End Problem_1
```



average = Sum / mLoop

ตัวอย่างรหัสเทียม (Pseudocode)

```
Algorithm Problem_1
        Variables : nLoop, Sum, testScore, average
        Begin
                READ nLoop
                Sum = 0
                FOR i = 1 to nLoop
                INPUT testScore
                Sum = Sum + testScore
                ENDFOR
                average = Sum / nLoop
                PRINT average
        End Problem_1
```

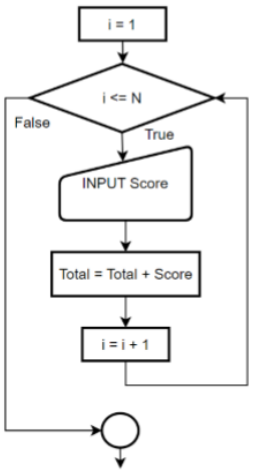


average

การแปลง Pseudocode เป็น Flowchart

ตัวอย่าง โปรแกรมบวกเลข N จำนวน

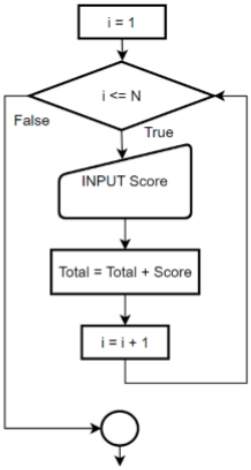
```
FOR i = 1 to N
        INPUT
Score
        Total =
Total + Score
ENDFOR
```



การแปลง Pseudocode เป็น Flowchart

ตัวอย่าง โปรแกรมบวกเลข N จำนวน

i	IF CONDITION (N=5)	Input Score	Total (New) = Total (Old) + Score
i=1	1 <= 5 True	1	Total (New) = 0 + 1 = 1
i=2	2 <= 5 True	2	Total (New) = 1 + 2 = 3
i=3	3 <= 5 True	4	Total (New) = 3 + 4 = 7
i=4	4 <= 5 True	6	Total (New) = 7 + 6 = 13
i=5	5 <= 5 True	8	Total (New) = 13 + 8 = 21
i=6	6 <= 5 False		



รูปแบบ Pseudocode

1) การกำหนดค่า และการคำนวณ

variable_name = expression

variable_name เป็นชื่อตัวแปร
expression เป็นค่าข้อมูลหรือนิพจน์

เช่น score = 50

score = 50

area = 3.14159265359 * R * R

area = 3.14159265359 * R *R

รูปแบบ Pseudocode

2) การอ่าน / รับข้อมูล

การอ่านข้อมูลจะมีคำสั่ง ได้แก่ READ, INPUT หรือ GET

READ variable_name

INPUT variable_name

GET variable_name

READ มักใช้เมื่อเราต้องการอ่านข้อมูลจากไฟล์หรือแหล่งข้อมูลภายนอก
INPUT ใช้เพื่อรับข้อมูลจากผู้ใช้ผ่านทางอินพุต
GET ใช้ในลักษณะเดียวกับ INPUT แต่เป็นอีกวิธีการเขียนเพื่อลดความซ้ำซ้อนของคำสั่ง

ตัวอย่างรหัสเทียม (Pseudocode)

```
Algorithm Problem_1
        Variables : nLoop, Sum, testScore, average
        Begin
                READ nLoop
                Sum = 0
                FOR i = 1 to nLoop
                INPUT testScore
                Sum = Sum + testScore
                ENDFOR
                average = Sum / nLoop
                PRINT average
        End Problem_1
```





รูปแบบ Pseudocode

3) การแสดงผลข้อมูล

การแสดงผลข้อมูลจะมีคำสั่ง ได้แก่ PRINT, PROMT หรือ WRITE

PRINT variable_name

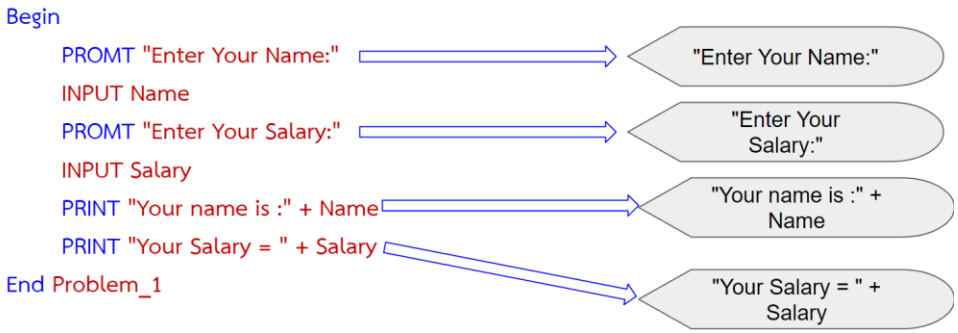
PROMT variable_name

WRITE variable_name

PRINT , PROMT มักใช้เมื่อเราต้องการแสดงค่าข้อมูลจากตัวแปรหรือข้อความ ออกทางหน้าจอ

WRITE จะใช้เพื่อบันทึกข้อมูลไว้ในไฟล์เพื่อเก็บลงฮาร์ดิสหรือสตอเรจอื่นๆ

ตัวอย่างรหัสเทียม (Pseudocode)



รูปแบบ Pseudocode

4) การกำหนดเงื่อนไข

การกำหนดเงื่อนไขหรือทางเลือกจะมีคำสั่ง IF .... ELSE IF .... ELSE ... ENDIF

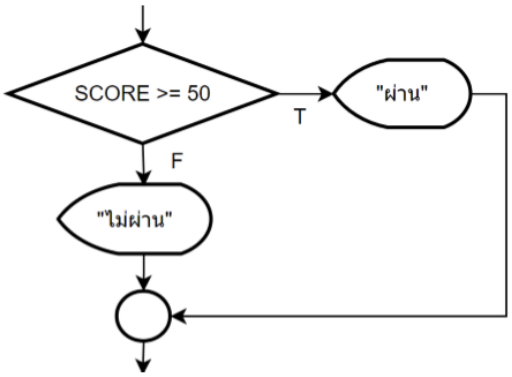
```
IF < Condition 1 > THEN          // ถ้า <เงื่อนไขที่ 1>
    Activity 1                   // ทำ Activity 1 ถ้าตรงตามเงื่อนไขที่ 1
ELSE IF < Condition 2 > THEN     // ถ้าไม่ตรงตาม <เงื่อนไขที่ 1> เช็คตามเงื่อนไขที่ 2
    Activity 2                   // ทำ Activity 2 ถ้าตรงตามเงื่อนไขที่ 2
ELSE                             // มิเช่นนั้น (ไม่ตรงตามเงื่อนไขใด)
    Activity 3                   // ถ้าไม่ตรงตาม เงื่อนไขใดๆ ให้ทำ Activity 3
ENDIF                            // จบเงื่อนไข
```

รูปแบบ Pseudocode

4) การกำหนดเงื่อนไข

ตัวอย่างการกำหนดเงื่อนไข

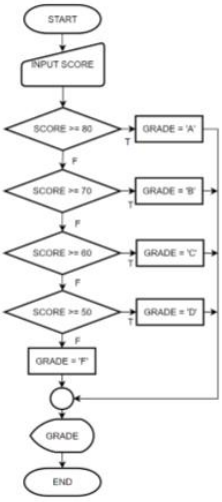
```
IF SCORE >= 50 THEN
        PRINT "ผ่าน"
ELSE
        PRINT "ไม่ผ่าน"
ENDIF
```



รูปแบบ Pseudocode

ตัวอย่างการกำหนดเงื่อนไข

```
IF SCORE >= 80 THEN
        GRADE = 'A'
ELSE IF SCORE >= 70 THEN
        GRADE = 'B'
ELSE IF SCORE >= 60 THEN
        GRADE = 'C'
ELSE IF SCORE >= 50 THEN
        GRADE = 'D'
ELSE
GRADE = 'F'
ENDIF
```

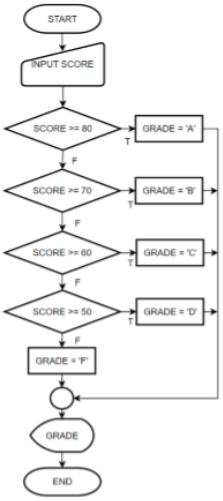


รูปแบบ Pseudocode

5) การกำหนดเงื่อนไขแบบ SWITCH CASE

ตัวอย่างการกำหนดเงื่อนไขแบบ SWITCH CASE

```
CASE SCORE OF
>= 80 : GRADE = 'A'
>= 70 : GRADE = 'B'
>= 60 : GRADE = 'C'
>= 50 : GRADE = 'D'
< 50 : GRADE = 'F'
ENDCASE
```



รูปแบบ Pseudocode

6) การวนรอบทำซ้ำแบบ FOR (ต้องมีตัว index นับรอบ)

```
FOR i=1 to N
    activity1
    activity2
    activity3
ENDFOR
```

รูปแบบ Pseudocode

6) การวนรอบทำซ้ำแบบ FOR (ต้องมีตัว index นับรอบ)

ตัวอย่าง

```
READ nLoop
Sum = 0
FOR i = 1 to nLoop
    INPUT testScore
    Sum = Sum + testScore
ENDFOR
```


รูปแบบ Pseudocode

7) การวนรอบทำซ้ำแบบ WHILE
(วนทำซ้ำจนกว่า Condition ของ while จะเป็นเท็จ)

```
WHILE <Condition>
    activity1
    activity2
    activity3
ENDWHILE
```

รูปแบบ Pseudocode

7) การวนรอบทำซ้ำแบบ WHILE

(วนทำซ้ำจนกว่า Condition ของ while จะเป็นเท็จ)

ตัวอย่าง

```
INPUT SALARY
WHILE SALARY < 0
    PRINT "SALARY MUST >=0, PLEASE INPUT AGAIN"
    INPUT SALARY
ENDWHILE
```


รูปแบบ Pseudocode

8) การวนรอบทำซ้ำแบบ DO ... UNTIL (WHILE)

(ทำก่อน 1 รอบ แล้ววนทำซ้ำจนกว่า Condition ของ until จะเป็นเท็จ)

```
DO
    activity1
    activity2
    activity3
UNTIL <Condition>
```

รูปแบบ Pseudocode

8) การวนรอบทำซ้ำแบบ DO ... UNTIL (WHILE)

(ทำก่อน 1 รอบ แล้ววนทำซ้ำจนกว่า Condition ของ until จะเป็นเท็จ)

ตัวอย่าง

```
DO
    PRINT "INPUT YOUR SALARY(>=0):"
    INPUT SALARY
UNTIL SALARY < 0
```


เอกสารอ้างอิง

นราธร สังข์ประเสริฐ, เอกสารประกอบการบรรยายวิชา 04-511-101 การโปรแกรมคอมพิวเตอร์, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย , 2558

Sites.google.com, "รหัสเทียม Psuedo Code - Programming", 2016. [Online]. Available: https://sites.google.com/site/programmingm42/-rhas-theiym-psuedo-code. [Accessed: 17- Jan- 2016].

Sites.google.com, "การแก้ปัญหา - Programming", 2016. [Online]. Available: https://sites.google.com/site/programmingm42/kar-kae-payha. [Accessed: 17- Jan- 2016].

แหล่งศึกษาเพิ่มเติม

ผศ.นราธร สังข์ประเสริฐ. (2563) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-3945-2

โอภาส เอี่ยมสิริวงศ์. (2559) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-2740-4

รศ.ธีรวัฒน์ ประกอบผล. (2562) คู่มือการเขียนโปรแกรมภาษา C ฉบับสมบูรณ์, ซิมพลิฟาย, ISBN: 978-616-26-24896

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase1] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=aDBmXMF9jOU

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase2] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=wslM0Aqepxs

Mooc การเขียนโปรแกรมคอมพิวเตอร์พื้นฐาน ด้วยภาษาซี 1 | Fundamentals of the C Programming Language https://lms.thaimooc.org/courses/course-v1:KMUTT+KMUTT013+2019/about

Mooc ทักษะการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี | Computer programming skills with C language https://lms.thaimooc.org/courses/course-v1:SU+SU011+2021/about

Coursera: Programming in C, ผู้สอน: Charles Russell Severance https://www.coursera.org/learn/programming-c

Coursera: ความเชี่ยวชาญพิเศษ การเขียนโปรแกรม C เบื้องต้น, ผู้สอน: Andrew D. Hilton et al. https://www.coursera.org/specializations/c-programming

Playlist Youtube บันทึกการสอนในห้องเรียน, วิชาการโปรแกรมคอมพิวเตอร์, ดร.พงศกร เจริญเนตรกุล 2566 https://www.youtube.com/watch?v=ewEfmiXaRi0&list=PL322g2itJn4mAE1utyOzF-C0SP0BRkej3&pp=gAQBiAQB

แบบฝึกหัดท้ายบท

1. การพัฒนาโปรแกรมมีกี่ขั้นตอน และขั้นตอนใดบ้างที่เป็นขั้นตอนสำคัญ
2. ผังงาน (Flowchart) คืออะไร และมีประโยชน์อย่างไร
3. รหัสเทียม (Pseudocode) ต่างจากผังงานอย่างไร
4. การวางแผน (Planning) ในการเขียนโปรแกรมเกี่ยวข้องกับการทำอะไรบ้าง
5. ขั้นตอนการ Debugging คืออะไร
6. ข้อผิดพลาดประเภท Syntax Error เกิดจากสาเหตุใด
7. ข้อผิดพลาดประเภท Runtime Error เกิดขึ้นเมื่อใด
8. Logical Error คืออะไร
9. อัลกอริทึม (Algorithm) คืออะไร
10. การวิเคราะห์ปัญหา (Analysis) ในการพัฒนาโปรแกรมควรคำนึงถึงอะไรบ้าง
11. ผังงานระบบ (System Flowchart) ต่างจากผังงานโปรแกรม (Program Flowchart) อย่างไร
12. ขั้นตอนใดในกระบวนการพัฒนาโปรแกรมที่เกี่ยวข้องกับการทดสอบว่าโปรแกรมทำงานถูกต้องหรือไม่
13. การแสดงผลในรหัสเทียมสามารถใช้คำสั่งใดได้บ้าง
14. การใช้เงื่อนไข IF/ELSE มีความสำคัญอย่างไรในโปรแกรม
15. การวนลูป (Loop) ชนิดใดทำงานซ้ำจนกว่าจะเจอเงื่อนไขที่เป็นเท็จ (False)
16. การเขียนโปรแกรมที่มีการวนลูปแบบ FOR ต้องใช้ตัวแปรอะไรเพื่อควบคุมจำนวนครั้งในการวนลูป
17. ในการออกแบบอัลกอริทึม คุณสามารถใช้เครื่องมือใดในการออกแบบได้บ้าง
18. การบวกเลขสองจำนวนสามารถใช้โครงสร้างใดในรหัสเทียม
19. ข้อดีของการเขียนรหัสเทียมคืออะไร
20. เหตุใดการเขียนผังงานจึงควรตรวจสอบก่อนนำไปเขียนเป็นโปรแกรม

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

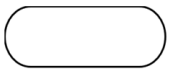
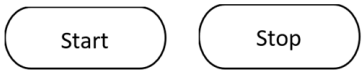
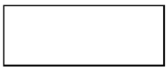
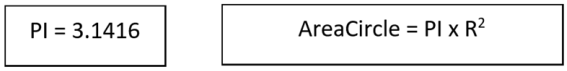
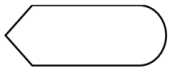
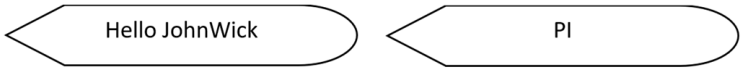
ใบฝึกปฏิบัติการที่ 2.1

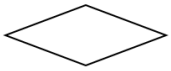
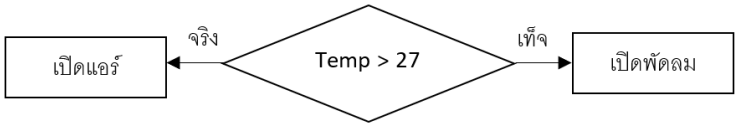
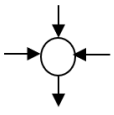
เรื่อง DailyLife Flow Chart

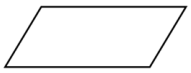
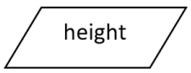
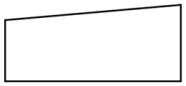
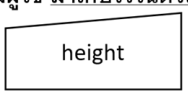
วัตถุประสงค์

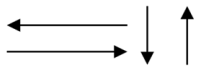
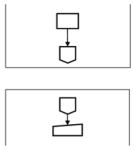
1. สามารถออกแบบขั้นตอนวิธีการจากกรณีศึกษาการทำกิจกรรมต่างๆในชีวิตประจำวันได้
2. สามารถเขียนผังงานได้อย่างถูกต้อง

ทบทวนความรู้

สัญลักษณ์ของผังงาน	ความหมาย
	**จุดเริ่มต้น /จุดสิ้นสุดของผังงาน** ต้องใส่ ณ จุดแรก และสุดท้ายของผังงานเสมอ 
	**กระบวนการ / การดำเนินการ / การประมวลผล / การคำนวณ / กำหนดค่า** อย่างใดอย่างหนึ่งตามที่กล่าวมา เช่น ประกาศตัวแปร คำนวณสูตรทางคณิตศาสตร์ เรียกใช้คำสั่ง 
	**แสดงข้อความและค่าของตัวแปรผ่านทางจอภาพ** 

สัญลักษณ์ของผังงาน	ความหมาย
	**ใช้ในการตัดสินใจ** หรือ หลายทางเลือก เช่น ถ้า อุณหภูมิมากกว่า 27 ให้เปิดแอร์ 
	**จุดเชื่อมต่อภายในหน้าเดียวกัน** ใช้ในการเชื่อมรวมเส้นทางที่มาจากหลายทางให้ไปทางเดียวกัน เข้ามาได้หลายทางแต่ต้องออกแค่หนึ่งทางเสมอ 

สัญลักษณ์ของผังงาน	ความหมาย
	**นำเข้าข้อมูล (มีอยู่แล้ว เช่น ข้อมูลจากเครือข่าย จากไฟล์ จากฐานข้อมูล) หรือนำเข้าข้อมูลที่ไม่ระบุวิธีการหรือแหล่งข้อมูล <u>มาเก็บไว้ในตัวแปร</u>ของโปรแกรม** 
	**การรับข้อมูลเข้าทางแป้นพิมพ์ จากการกด keyboard ของผู้ใช้ <u>มาเก็บไว้ในตัวแปร</u>ของโปรแกรม เช่นรับค่าส่วนสูงมาเก็บไว้ในตัวแปร height** 

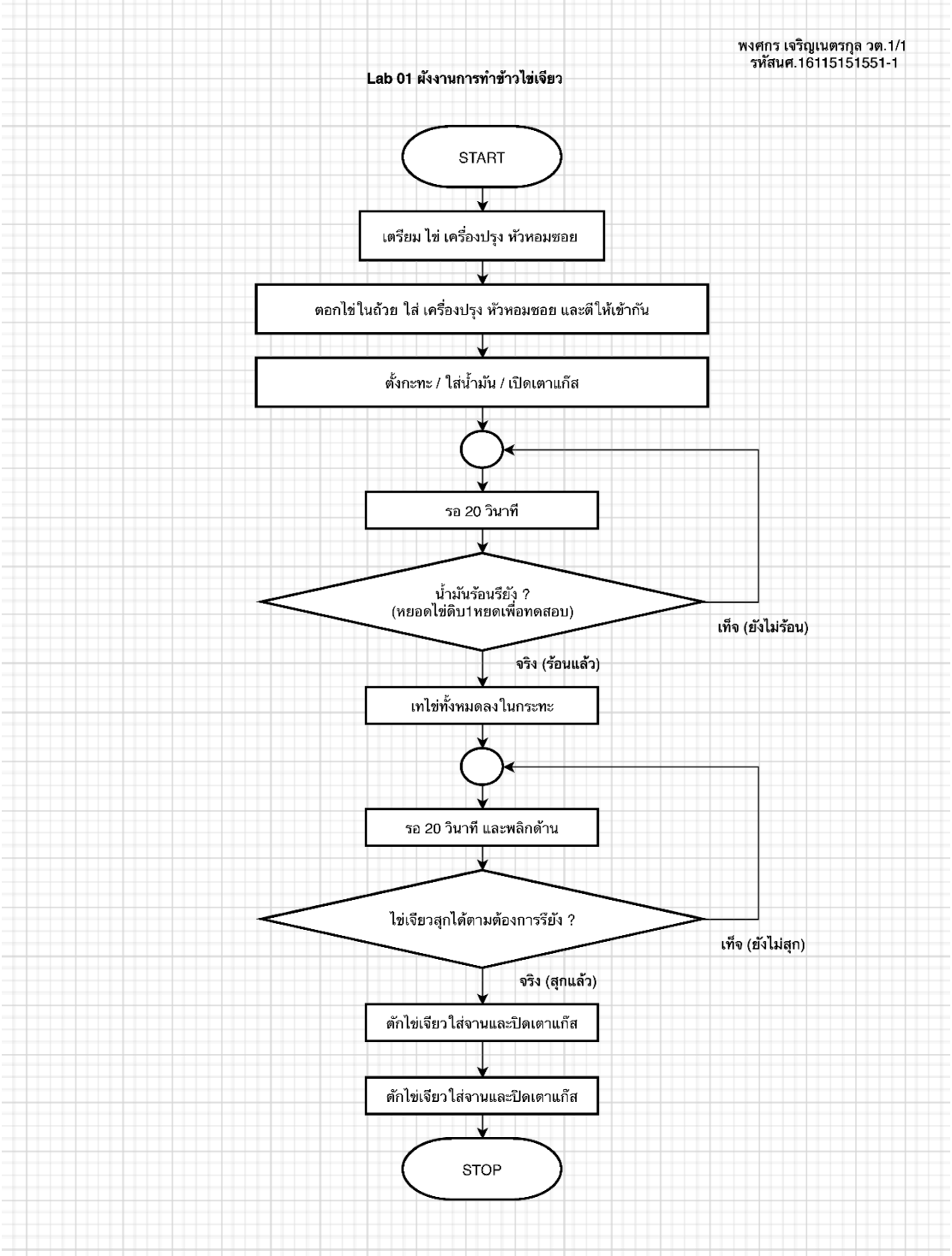
สัญลักษณ์ของผังงาน	ความหมาย
	**แสดงทิศทางการทำงานของผังงาน (บน ล่าง ซ้าย ขวา เท่านั้น ไม่มีเส้นเฉียง)**
	**จุดเชื่อมต่อระหว่างหน้า** หากเขียนผังงานไม่พอในหนึ่งหน้าให้เขียนอักษรใดสักตัวหนึ่งล้อมด้วยสัญลักษณ์นี้ไว้ในหน้าเดิมและในกระดาษแผ่นใหม่ให้เขียนอักษรเดิมในหน้าที่แล้วล้อมด้วยสัญลักษณ์นี้และดำเนินการต่อผังงานจากจุดนี้ได้เลย 

คำสั่ง

จงเขียน Flowchart ในชีวิตประจำวัน เช่น การเดินทางมาเรียนในมหาวิทยาลัย การไปซื้อของที่เซเว่น การลงทะเบียนเรียน การอาบน้ำ การแปรงฟัน การซักผ้า การตื่นนอน การเข้าห้องน้ำ การเดินทางไปเที่ยว การทำอาหาร การล้างจาน การถูพื้น การสมัคร... การเล่นเกมส์...

- โดยมีจุดเริ่มต้น-จุดจบ ทิศทาง กระบวนการ เงื่อนไข จุดเชื่อมต่อ (ต้องมีทุกองค์ประกอบ)
- โดยใช้เวบไซต์ https://app.diagrams.net/ ในการเขียนแล้ว export ออกมาเป็นไฟล์ PDF
- บอกชื่อแผนผังที่เขียนและชื่อ-นามสกุล รหัสนศ.ด้วย

ตัวอย่าง


พงศกร เจริญเนตรกุล วต.1/1
รหัสนศ.16115151551-1
Lab 01 ผังงานการทำข้าวไข่เจียว
START
เตรียม ไข่ เครื่องปรุง หัวหอมซอย
ตอกไข่ในถ้วย ใส่ เครื่องปรุง หัวหอมซอย และตีให้เข้ากัน
ตั้งกะทะ / ใส่น้ำมัน / เปิดเตาแก๊ส
รอ 20 วินาที
น้ำมันร้อนรึยัง ?
(หยอดไข่ดิบ1หยดเพื่อทดสอบ)
เท็จ (ยังไม่ร้อน)
จริง (ร้อนแล้ว)
เทไข่ทั้งหมดลงในกระทะ
รอ 20 วินาที และพลิกด้าน
ไข่เจียวสุกได้ตามต้องการรึยัง ?
เท็จ (ยังไม่สุก)
จริง (สุกแล้ว)
ตักไข่เจียวใส่จานและปิดเตาแก๊ส
ตักไข่เจียวใส่จานและปิดเตาแก๊ส
STOP

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

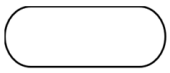
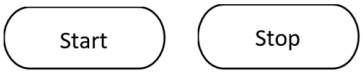
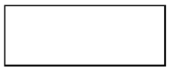
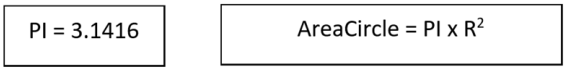
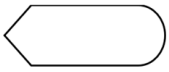
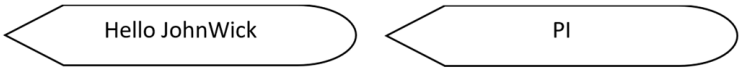
ใบฝึกปฏิบัติการที่ 2.2

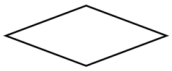
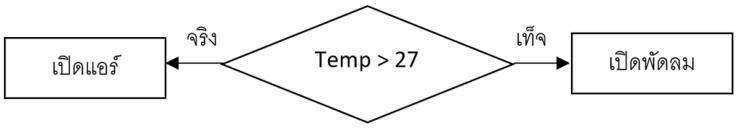
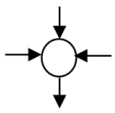
เรื่อง Programming Flowchart

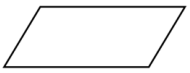
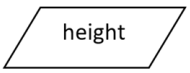
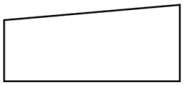
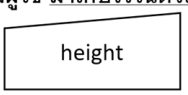
วัตถุประสงค์

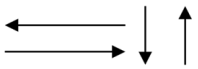
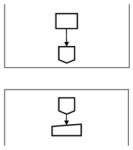
1. สามารถออกแบบขั้นตอนวิธีการในการเขียนโปรแกรมอย่างง่ายได้
2. สามารถเขียนผังงานได้อย่างถูกต้อง

ทบทวนความรู้

สัญลักษณ์ของผังงาน	ความหมาย
	**จุดเริ่มต้น /จุดสิ้นสุดของผังงาน** ต้องใส่ ณ จุดแรก และสุดท้ายของผังงานเสมอ 
	**กระบวนการ / การดำเนินการ / การประมวลผล / การคำนวณ / กำหนดค่า** อย่างใดอย่างหนึ่งตามที่กล่าวมา เช่น ประกาศตัวแปร คำนวณสูตรทางคณิตศาสตร์ เรียกใช้คำสั่ง 
	**แสดงข้อความและค่าของตัวแปรผ่านทางจอภาพ** 

สัญลักษณ์ของผังงาน	ความหมาย
	**ใช้ในการตัดสินใจ** หรือ หลายทางเลือก เช่น ถ้า อุณหภูมิมากกว่า 27 ให้เปิดแอร์ 
	**จุดเชื่อมต่อภายในหน้าเดียวกัน** ใช้ในการเชื่อมรวมเส้นทางที่มาจากหลายทางให้ไปทางเดียวกัน เข้ามาได้หลายออกหนึ่งเสมอ 

สัญลักษณ์ของผังงาน	ความหมาย
	**นำเข้าข้อมูล (มีอยู่แล้ว เช่น ข้อมูลจากเครือข่าย จากไฟล์ จากฐานข้อมูล) หรือนำเข้าข้อมูลที่ไม่ระบุวิธีการหรือแหล่งข้อมูล มาเก็บไว้ในตัวแปรของโปรแกรม** 
	**การรับข้อมูลเข้าทางแป้นพิมพ์ จากการกด keyboard ของผู้ใช้ มาเก็บไว้ในตัวแปรของโปรแกรม เช่นรับค่าส่วนสูงมาเก็บไว้ในตัวแปร height** 

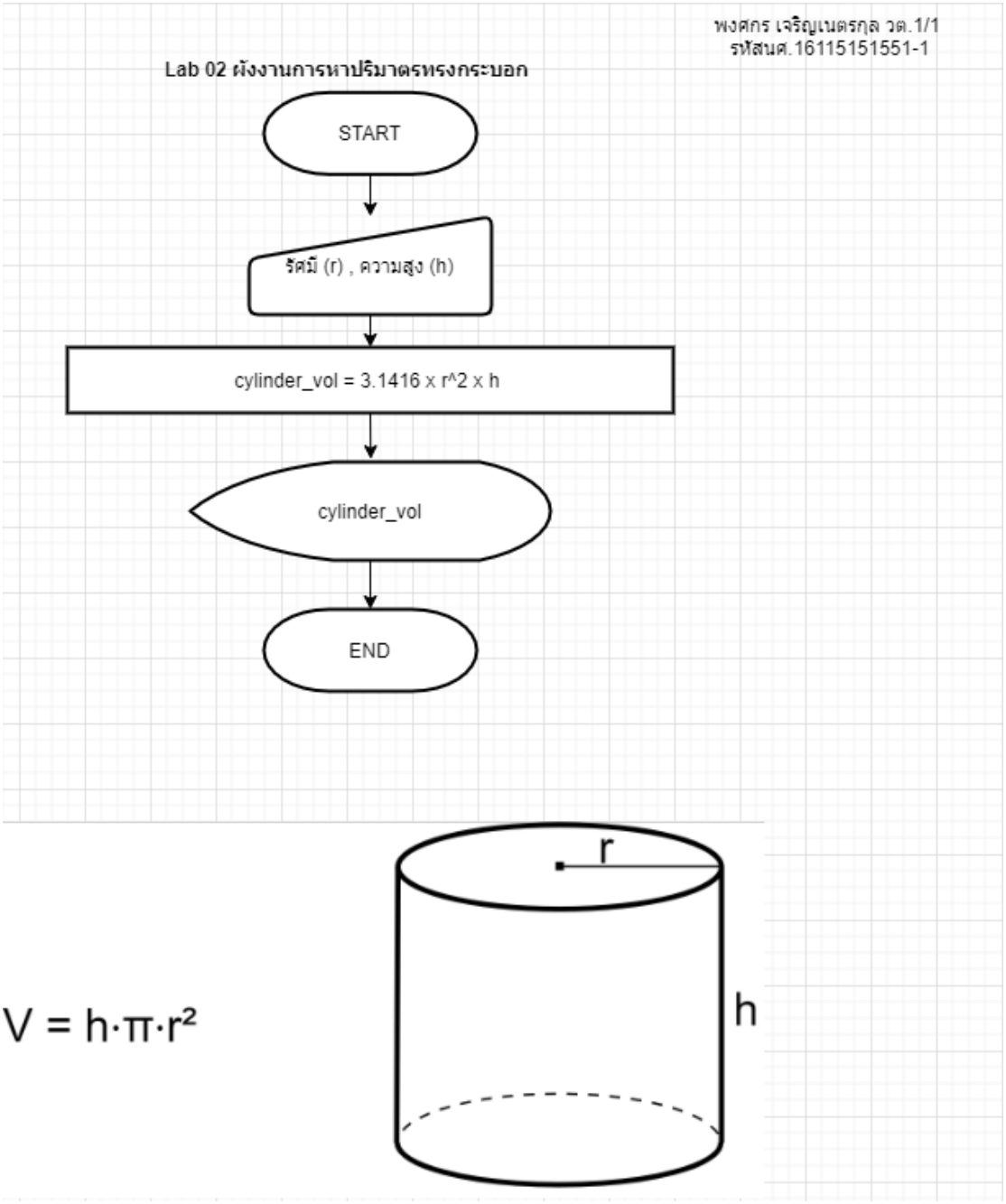
สัญลักษณ์ของผังงาน	ความหมาย
	**แสดงทิศทางการทำงานของผังงาน (บน ล่าง ซ้าย ขวา เท่านั้น ไม่มีเส้นเฉียง)**
	**จุดเชื่อมต่อระหว่างหน้า** หากเขียนผังงานไม่พอในหนึ่งหน้าให้เขียนอักษรใดสักตัวหนึ่งล้อมด้วยสัญลักษณ์นี้ไว้ในหน้าเดิมและในกระดาษแผ่นใหม่ให้เขียนอักษรเดิมในหน้าที่แล้วล้อมด้วยสัญลักษณ์นี้และดำเนินการต่อผังงานจากจุดนี้ได้เลย 

คำสั่ง

จงเขียน Flowchart โปรแกรม (คิดโจทย์เอง) เช่น โปรแกรมคำนวณค่าพื้นที่ สามเหลี่ยม และวงกลม

- โดยมีจุดเริ่มต้น-จบ รับข้อมูลเข้า ทิศทาง กระบวนการ แสดงผลทางหน้าจอโดยมีจุดเริ่มต้น-จุดจบ ทิศทาง กระบวนการ เงื่อนไข จุดเชื่อมต่อ (ต้องมีทุกองค์ประกอบ)
- โดยใช้เวบไซต์ https://app.diagrams.net/ ในการเขียนแล้ว export ออกมาเป็นไฟล์ PDF
- บอกชื่อแผนผังที่เขียนและชื่อ-นามสกุล รหัสนศ.ด้วย

ตัวอย่าง


พงศกร เจริญเนตรกุล วต.1/1
รหัสนศ.16115151551-1
Lab 02 ผังงานการหาปริมาตรทรงกระบอก
START
รัศมี (r) , ความสูง (h)
cylinder_vol = 3.1416 x r^2 x h
cylinder_vol
END
V = h·π·r²
r
h

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

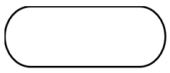
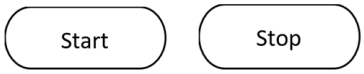
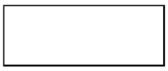
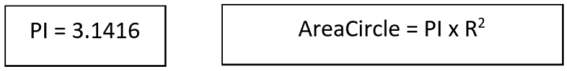
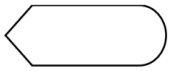
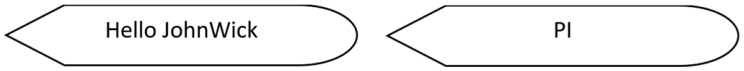
ใบฝึกปฏิบัติการที่ 2.3

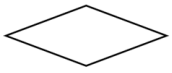
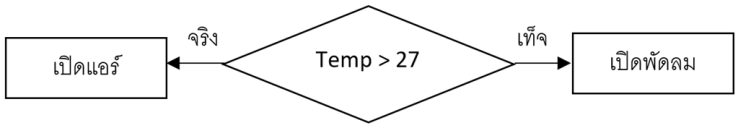
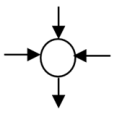
เรื่อง Flowchart Enter to Violent Content Web

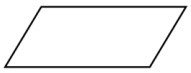
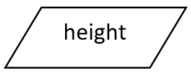
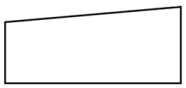
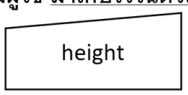
วัตถุประสงค์

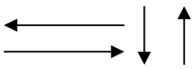
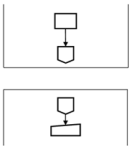
1. สามารถออกแบบขั้นตอนวิธีการในการเขียนโปรแกรมตามคำสั่งได้
2. สามารถเขียนผังงานโดยใช้สัญลักษณ์ได้อย่างถูกต้อง

ทบทวนความรู้

สัญลักษณ์ของผังงาน	ความหมาย
	**จุดเริ่มต้น /จุดสิ้นสุดของผังงาน** ต้องใส่ ณ จุดแรก และสุดท้ายของผังงานเสมอ 
	**กระบวนการ / การดำเนินการ / การประมวลผล / การคำนวณ / กำหนดค่า** อย่างใดอย่างหนึ่งตามที่กล่าวมา เช่น ประกาศตัวแปร คำนวณสูตรทางคณิตศาสตร์ เรียกใช้คำสั่ง 
	**แสดงข้อความและค่าของตัวแปรผ่านทางจอภาพ** 

สัญลักษณ์ของผังงาน	ความหมาย
	**ใช้ในการตัดสินใจ** หรือ หลายทางเลือก เช่น ถ้า อุณหภูมิมากกว่า 27 ให้เปิดแอร์ 
	**จุดเชื่อมต่อภายในหน้าเดียวกัน** ใช้ในการเชื่อมรวมเส้นทางที่มาจากหลายทางให้ไปทางเดียวกัน เข้ามาได้หลายออกหนึ่งเสมอ 

สัญลักษณ์ของผังงาน	ความหมาย
	**นำเข้าข้อมูล (มีอยู่แล้ว เช่น ข้อมูลจากเครือข่าย จากไฟล์ จากฐานข้อมูล) หรือนำเข้าข้อมูลที่ไม่ระบุวิธีการหรือแหล่งข้อมูล มาเก็บไว้ในตัวแปรของโปรแกรม** 
	**การรับข้อมูลเข้าทางแป้นพิมพ์ จากการกด keyboard ของผู้ใช้ มาเก็บไว้ในตัวแปรของโปรแกรม เช่นรับค่าส่วนสูงมาเก็บไว้ในตัวแปร height** 

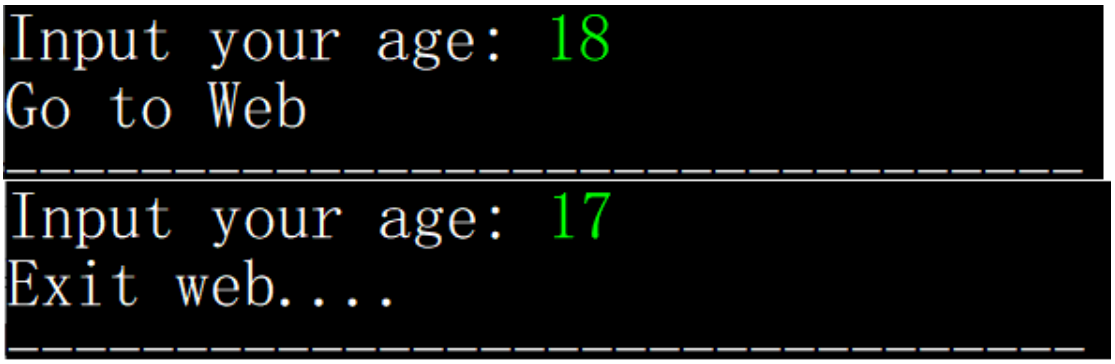
สัญลักษณ์ของผังงาน	ความหมาย
	**แสดงทิศทางการทำงานของผังงาน (บน ล่าง ซ้าย ขวา เท่านั้น ไม่มีเส้นเฉียง)**
	**จุดเชื่อมต่อระหว่างหน้า** หากเขียนผังงานไม่พอในหนึ่งหน้าให้เขียนอักษรใดสักตัวหนึ่งล้อมด้วยสัญลักษณ์นี้ไว้ในหน้าเดิมและในกระดาษแผ่นใหม่ให้เขียนอักษรเดิมในหน้าที่แล้วล้อมด้วยสัญลักษณ์นี้และดำเนินการต่อผังงานจากจุดนี้ได้เลย 

คำสั่ง

จงเขียน Flowchart โปรแกรมตรวจสอบอายุของผู้เข้าเวบไซต์ที่มีความรุนแรงของเนื้อหา

- ให้รับค่า อายุ จากแป้นพิมพ์ มาเก็บไว้ในตัวแปร age
- แสดงค่าในตัวแปร age ออกทางหน้าจอ
- ถัดจากนั้นให้ตรวจสอบเงื่อนไขอายุ
- ถ้า age มากกว่าหรือเท่ากับ 18 ให้ พิมพ์ออกทางหน้าจอว่า Go to Web
- อื่นๆ (ELSE) ให้ พิมพ์ออกทางหน้าจอว่า Exit Web
- เขียนผังงานโดยใช้เวบไซต์ https://app.diagrams.net/ ในการเขียนแล้ว export ออกมาเป็นไฟล์ PDF

ตัวอย่างผลลัพธ์การทำงานของโปรแกรม หากนำ Flowchart ไปเขียนเป็นโปรแกรม



ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

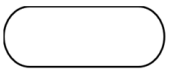
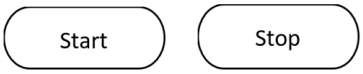
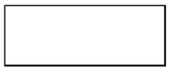
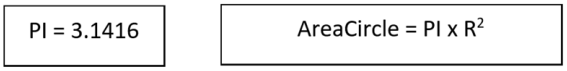
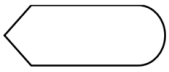
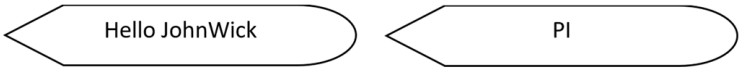
ใบฝึกปฏิบัติการที่ 2.4

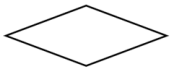
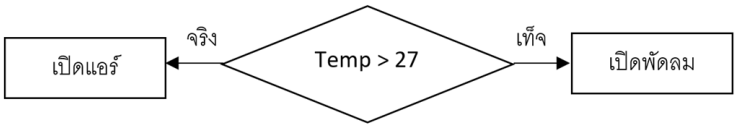
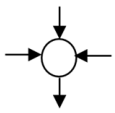
เรื่อง Flowchart Find EVEN Number 1-200

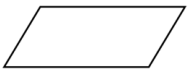
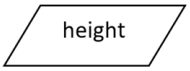
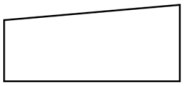
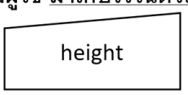
วัตถุประสงค์

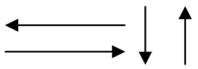
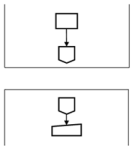
1. สามารถเข้าใจกระบวนการทำงานของโปรแกรมจากโจทย์จริงและออกแบบเป็นขั้นตอนวิธีได้
2. สามารถเขียนผังงานโดยใช้สัญลักษณ์ได้อย่างถูกต้อง

ทบทวนความรู้

สัญลักษณ์ของผังงาน	ความหมาย
	**จุดเริ่มต้น /จุดสิ้นสุดของผังงาน** ต้องใส่ ณ จุดแรก และสุดท้ายของผังงานเสมอ 
	**กระบวนการ / การดำเนินการ / การประมวลผล / การคำนวณ / กำหนดค่า** อย่างใดอย่างหนึ่งตามที่กล่าวมา เช่น ประกาศตัวแปร คำนวณสูตรทางคณิตศาสตร์ เรียกใช้คำสั่ง 
	**แสดงข้อความและค่าของตัวแปรผ่านทางจอภาพ** 

สัญลักษณ์ของผังงาน	ความหมาย
	**ใช้ในการตัดสินใจ** หรือ หลายทางเลือก เช่น ถ้า อุณหภูมิมากกว่า 27 ให้เปิดแอร์ 
	**จุดเชื่อมต่อภายในหน้าเดียวกัน** ใช้ในการเชื่อมรวมเส้นทางที่มาจากหลายทางให้ไปทางเดียวกัน เข้ามาได้หลายออกหนึ่งเสมอ 

สัญลักษณ์ของผังงาน	ความหมาย
	**นำเข้าข้อมูล (มีอยู่แล้ว เช่น ข้อมูลจากเครือข่าย จากไฟล์ จากฐานข้อมูล) หรือนำเข้าข้อมูลที่ไม่ระบุวิธีการหรือแหล่งข้อมูล มาเก็บไว้ในตัวแปรของโปรแกรม** 
	**การรับข้อมูลเข้าทางแป้นพิมพ์ จากการกด keyboard ของผู้ใช้ มาเก็บไว้ในตัวแปรของโปรแกรม เช่นรับค่าส่วนสูงมาเก็บไว้ในตัวแปร height** 

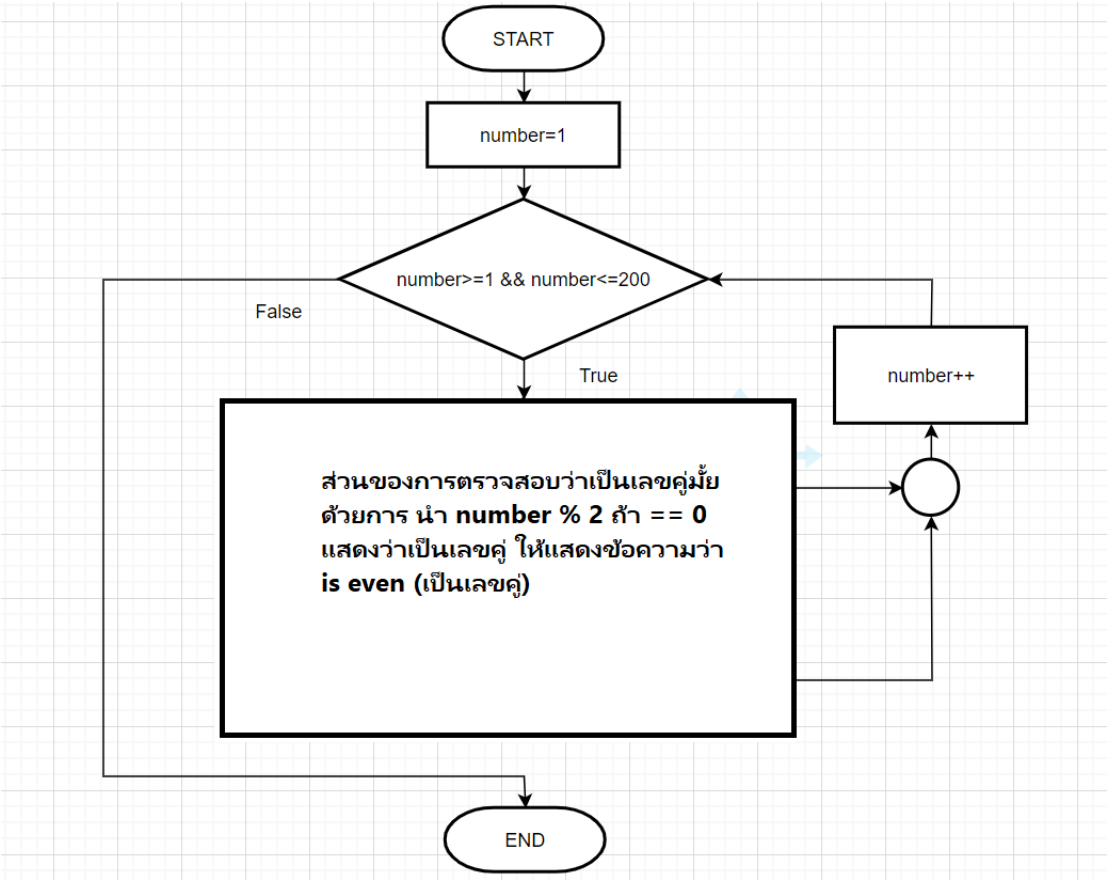
สัญลักษณ์ของผังงาน	ความหมาย
	**แสดงทิศทางการทำงานของผังงาน (บน ล่าง ซ้าย ขวา เท่านั้น ไม่มีเส้นเฉียง)**
	**จุดเชื่อมต่อระหว่างหน้า** หากเขียนผังงานไม่พอในหนึ่งหน้าให้เขียนอักษรใดสักตัวหนึ่งล้อมด้วยสัญลักษณ์นี้ไว้ในหน้าเดิมและในกระดาษแผ่นใหม่ให้เขียนอักษรเดิมในหน้าที่แล้วล้อมด้วยสัญลักษณ์นี้และดำเนินการต่อผังงานจากจุดนี้ได้เลย 

คำสั่ง

จงเขียน Flowchart โปรแกรมวนค่า num ตั้งแต่ 1-200 และ ตรวจสอบว่าตัวแปร num เป็นเลขคู่หรือเปล่า ถ้าเป็นเลขคู่ให้แสดงค่า num ออกมา ตามด้วยข้อความ is even (เป็นเลขคู่)

- เขียนผังงานโดยใช้เวบไซต์ https://app.diagrams.net/ ในการเขียนแล้ว export ออกมาเป็นไฟล์ PDF

ตัวอย่าง


START
number=1
number>=1 && number<=200
False
True
number++
ส่วนของการตรวจสอบว่าเป็นเลขคู่มั้ย
ด้วยการ นำ number % 2 ถ้า == 0
แสดงว่าเป็นเลขคู่ ให้แสดงข้อความว่า
is even (เป็นเลขคู่)
END

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 2.5

เรื่อง Total Cholesterol (TC)

วัตถุประสงค์

1. สามารถนำโจทย์จริงๆมาออกแบบเป็นขั้นตอนวิธีการในการเขียนโปรแกรมได้อย่างถูกต้อง
2. สามารถเขียนผังงานโดยใช้สัญลักษณ์ได้อย่างถูกต้อง

คำสั่ง

จงเขียน Flowchart คำนวณค่า Total Cholesterol (TC) โดยรับค่า LDL HDL และ TG จากคีย์บอร์ด มา คำนวณค่า TC จากนั้นนำมาตรวจสอบว่าค่า TC อยู่ในระดับความเสี่ยงใดตามภาพด้านล่าง



- เขียนผังงานโดยใช้เวบไซต์ https://app.diagrams.net/ ในการเขียนแล้ว export ออกมาเป็นไฟล์ PDF

แผนการสอนเรื่อง พื้นฐานการเขียนโปรแกรม

สัปดาห์ที่ 3-4

จำนวนชั่วโมงที่ใช้สอน 10 ชั่วโมง

วัตถุประสงค์ เพื่อให้นักศึกษาสามารถ

1. ระบุโครงสร้างและองค์ประกอบพื้นฐานของภาษา C เช่น ตัวแปร ตัวดำเนินการ และโครงสร้างของโปรแกรม
2. อธิบายวิธีการทำงานของฟังก์ชัน printf() และ scanf() ในการแสดงผลและรับข้อมูลจากผู้ใช้
3. ประกาศตัวแปรให้เหมาะสมกับชนิดของข้อมูลที่ต้องการเก็บค่าได้
4. เลือกใช้ตัวดำเนินการทางคณิตศาสตร์และตรรกศาสตร์เพื่อแก้โจทย์ปัญหาได้
5. เขียนโปรแกรมภาษา C พื้นฐานที่ใช้ฟังก์ชันprintf() และ scanf() เพื่อแสดงผลและรับค่าข้อความในโปรแกรมได้
6. วิเคราะห์และระบุข้อผิดพลาดที่พบบ่อยในโปรแกรมภาษา C และแก้ไขโปรแกรมได้

หัวข้อบรรยาย

บทที่ 3 พื้นฐานการเขียนโปรแกรม

3.1 แนะนำภาษาซี (Introduction to C)

3.2 การติดตั้งโปรแกรมสำหรับพัฒนาโปรแกรมภาษาซี (C/C++ IDE)

3.3 พื้นฐานและโครงสร้างของภาษา C

3.4 ตัวแปรและตัวดำเนินการ

3.5 การแสดงผลข้อความทางหน้าจอและรับค่าจากคีย์บอร์ด

3.6 ข้อผิดพลาดที่เกิดขึ้นบ่อย (Common Errors)

กิจกรรมการเรียนการสอน

1. การบรรยายในชั้นเรียน
2. การให้ผู้เรียนทดลองเขียนรหัสต้นฉบับและทดลองรันโปรแกรมระหว่างบรรยาย
3. การให้ผู้เรียนทำใบฝึกปฏิบัติการ
4. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)

สื่อการสอน

1. เอกสารประกอบการสอนแบบภาพเลื่อนหรือภาพนิ่ง
2. ใบฝึกปฏิบัติการ
3. โปรแกรมพัฒนาโปรแกรมภาษาซี เช่น DevC++, VS Code เป็นต้น
4. การเข้าถึงข้อมูลจากอินเทอร์เน็ต

การประเมินผลการเรียนรู้

ผลการเรียนรู้	วิธีการประเมิน	สัปดาห์ที่ประเมิน	สัดส่วนของการประเมินผล
1. คุณธรรมจริยธรรม	1.1 ประเมินจากการเข้าชั้นเรียน การส่งแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการตามกำหนดเวลาที่มอบหมาย 1.2 ประเมินจากพฤติกรรมของนักศึกษาในการปฏิบัติตามกฎระเบียบและข้อบังคับต่างๆ ทั้งในข้อบังคับในชั้นเรียน ข้อบังคับของมหาวิทยาลัย 1.3 ประเมินจากการคัดลอกแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการ 1.4 ประเมินจากการมีส่วนร่วมในห้องเรียนและการปิดคอมพิวเตอร์ เก็บขยะ เก็บเก้าอี้หลังจากใช้งานเสร็จ	ตลอดภาคการศึกษา	10%
2. ความรู้	2.1 แบบฝึกหัดท้ายบท 2.2 การสอบกลางภาคเรียนและปลายภาคเรียน	2-8, 11-16 9, 17	20% 40%
3. ทักษะทางปัญญา	3.1 ประเมินจากใบฝึกปฏิบัติการ	ตลอดภาคการศึกษา	20%
4. ทักษะความสัมพันธ์ระหว่างบุคคลและความรับผิดชอบ	4.1 พิจารณาจากการทำใบฝึกปฏิบัติการ 4.3 ประเมินจากพฤติกรรมในการทำงานกลุ่ม การระดมความคิดเห็น การตอบคำถามในชั้นเรียน	ตลอดภาคการศึกษา	5%
5. ทักษะวิเคราะห์เชิงตัวเลข การสื่อสารและการใช้เทคโนโลยีสารสนเทศ	5.1 การทดสอบหรือการประเมินจากงานที่มอบหมาย 5.2 สังเกตการใช้คอมพิวเตอร์ในการเขียนโปรแกรม การทำใบฝึกปฏิบัติการ และการค้นหาข้อมูลเพิ่มเติมจากอินเทอร์เน็ต	ตลอดภาคการศึกษา	5%
	รวม		100%

อ่าน Slide เวอร์ชันล่าสุดได้ที่ Google Slide

https://s.coecore.com/scp3

การโปรแกรมคอมพิวเตอร์

บทที่ 3 พื้นฐานการเขียนโปรแกรม

ผู้สอน ดร.พงศกร เจริญเนตรกุล

Review

- ❑ การติดตั้งโปรแกรมสำหรับพัฒนาโปรแกรมภาษาซี (C/C++ IDE)
- ❑ พื้นฐานและโครงสร้างภาษาซี
- ❑ ตัวแปรและตัวดำเนินการ
- ❑ การแสดงผลข้อความ และการรับค่าจากคีย์บอร์ด
- ❑ ข้อผิดพลาดที่เกิดขึ้นบ่อย

แนะนำภาษาซี

แนะนำภาษาซี

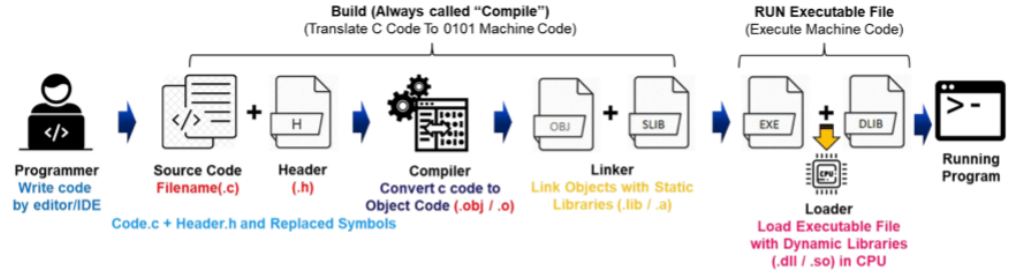


Dennis Ritchie

- ออกแบบและพัฒนาโดย Dennis Ritchie และ Brian Kernighan เมื่อปี ค.ศ. 1972
- มีมาตรฐานรองรับ จาก ANSI มีการพัฒนาต่อยอดเป็น C++ เป็นภาษาที่ได้รับความนิยมอยู่ในอันดับ2-3 (ข้อมูลปี2018-2019) และใช้จนถึงปัจจุบัน
- **ทำไมต้อง C ?** ⟶ ง่ายและทำให้เข้าใจการทำงานระดับต่ำของคอมพิวเตอร์ได้ดี, เป็นที่นิยมในการเขียนลงในฮาร์ดแวร์ของอุปกรณ์ต่าง ๆ, มีขนาดเล็กแต่ทำงานได้รวดเร็ว, เป็นต้นแบบภาษาอื่น ๆ เช่น C#, Java

แนะนำภาษาซี

องค์ประกอบภาษาซี

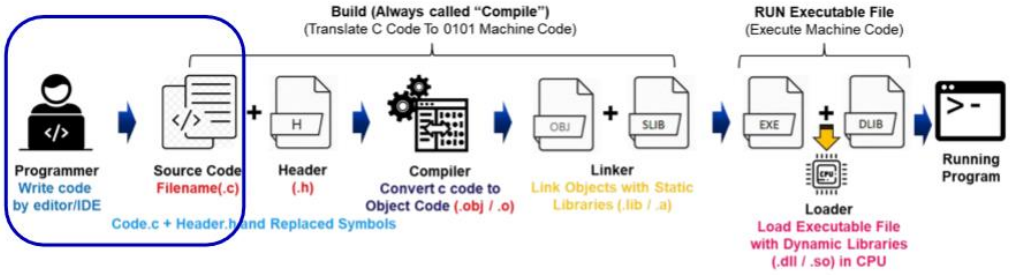


แนะนำภาษาซี

1 Programmer (นักพัฒนา)

นักพัฒนาเขียนโค้ดโดยใช้โปรแกรมแก้ไขหรือ IDE (Integrated Development Environment)

โค้ดที่เขียนมักอยู่ในไฟล์ที่มีนามสกุล .c (สำหรับภาษา C) หรือ .cpp (สำหรับภาษา C++)



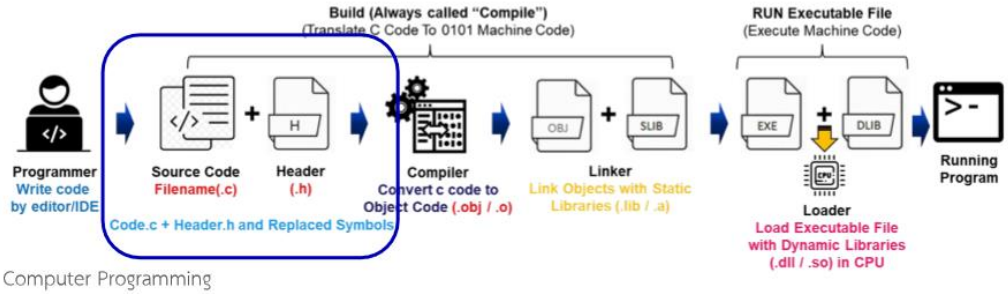
แนะนำภาษาซี

2 Source Code (โค้ดต้นฉบับ)

โค้ดที่นักพัฒนาเขียนจะถูกจัดเก็บเป็นไฟล์ที่มีนามสกุล .c (หรือ .cpp)

นอกจากนี้ยังมีการใช้ไฟล์เฮดเดอร์ (header files) ที่มีนามสกุล .h ซึ่งจะบรรจุคำประกาศฟังก์ชันและตัวแปรที่จะใช้ในโปรแกรม

ในขั้นนี้ ไฟล์ที่สำคัญคือ code.c และ header.h ซึ่งทำงานร่วมกันในโปรแกรม



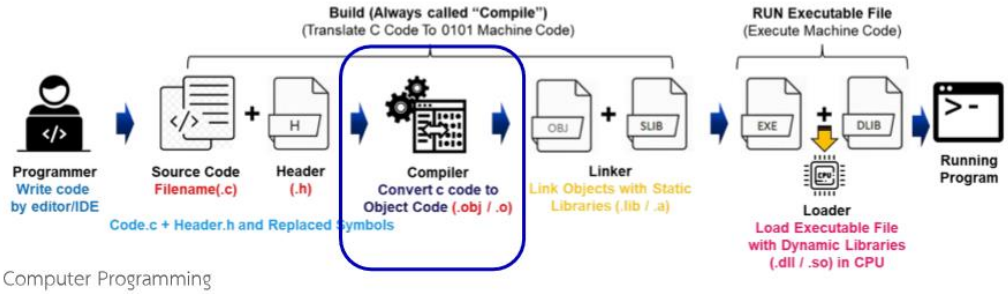
Computer Programming 7

แนะนำภาษาซี

3 Compiler (คอมไพเลอร์)

คอมไพเลอร์จะแปลงโค้ดจาก C หรือ C++ ให้กลายเป็น object code (.obj หรือ .o)

ในขั้นตอนนี้ โค้ดต้นฉบับจะถูกแปลงเป็นรหัสที่สามารถอ่านได้โดยเครื่อง (machine code)



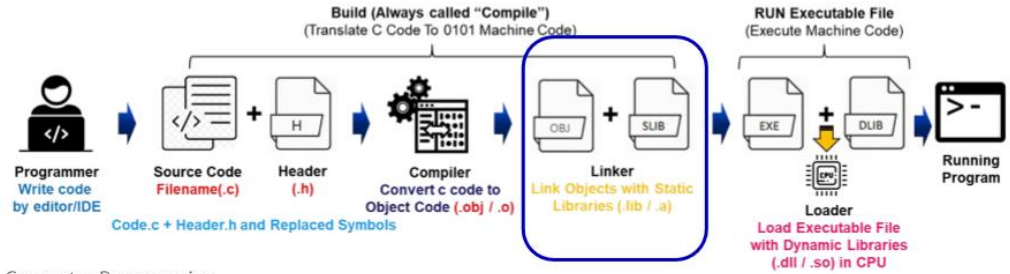
Computer Programming 8

แนะนำภาษาซี

4 Linker (ลิงเกอร์)

ลิงเกอร์จะเชื่อมโยง object code กับไลบรารีที่จำเป็น (Run-Time Library) (ทั้งไลบรารีแบบสแตติก .lib หรือ .a)

กระบวนการนี้จะสร้างไฟล์เอ็กซีคิวเทเบิล (ไฟล์ที่สามารถรันได้) เช่นไฟล์ .exe สำหรับ Windows

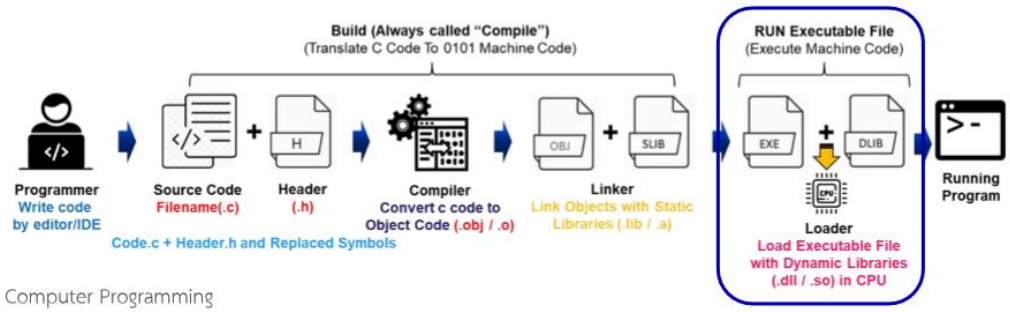


แนะนำภาษาซี

5 Loader (โหลดเดอร์)

ไฟล์ที่รันได้จะถูกโหลดเข้าไปใน CPU โดย loader

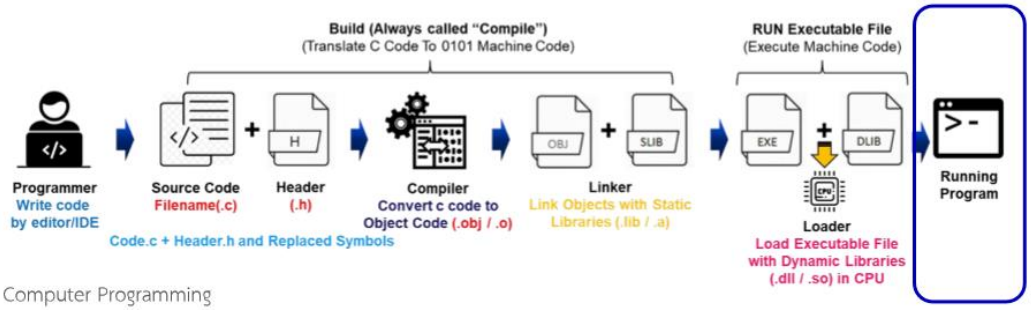
Loader จะนำไฟล์เอ็กซีคิวเทเบิลไปรวมกับไลบรารีแบบไดนามิก (เช่น .dll บน Windows หรือ .so บน Linux)



แนะนำภาษาซี

6 Running Program (การทำงานของโปรแกรม)ไฟล์ที่รันได้จะถูกโหลดเข้าไปใน CPU โดย loader

เมื่อโปรแกรมถูกโหลดและไลบรารีทั้งหมดถูกรวมกันอย่างถูกต้อง โปรแกรมจะเริ่มทำงาน



Computer Programming 11

ก่อนลงมือเขียนโปรแกรมหรือซอฟต์แวร์

Source code หรือ ซอฟต์แวร์คอมพิวเตอร์ ต่างๆ ถือเป็น ทรัพย์สินทางปัญญา – Intellectual Property ที่เป็นลิขสิทธิ์ (Copyright) ตามพ.ร.บ. ลิขสิทธิ์ พ.ศ. 2537

โดยซอฟต์แวร์หรือโปรแกรมคอมพิวเตอร์คือสิ่งที่ได้รับความคุ้มครองภายใต้พระราชบัญญัติลิขสิทธิ์ พ.ศ.2537 ("พ.ร.บ. ลิขสิทธิ์") โดยจัดเป็นงานอันมีลิขสิทธิ์ประเภทงานวรรณกรรม (Literary work)

Computer Programming 12

ก่อนลงมือเขียนโปรแกรมหรือซอฟต์แวร์

“พ.ร.บ. ลิขสิทธิ์ กำหนดว่า โปรแกรมคอมพิวเตอร์ หมายถึง คำสั่ง ชุดคำสั่ง หรือสิ่งอื่นใดที่นำไปใช้กับคอมพิวเตอร์ เพื่อให้เครื่องคอมพิวเตอร์ทำงานหรือให้ได้ผลอย่างหนึ่งอย่างใด ในกรณีที่บุคคลใด ทำซ้ำ ดัดแปลง เผยแพร่ต่อสาธารณชน หรือให้เช่าต้นฉบับหรือสำเนาของโปรแกรมคอมพิวเตอร์ บุคคลนั้นจะมีความผิดฐานละเมิดลิขสิทธิ์ซอฟต์แวร์ มีโทษปรับตั้งแต่ 20,000 บาทถึง 200,000 บาท

และหากเป็นการกระทำเพื่อการค้า จะต้องโทษจำคุกตั้งแต่ 6 เดือน ถึง 4ปี หรือปรับตั้งแต่ 100,000 ถึง800,000 บาท หรือทั้งจำทั้งปรับ”

อ้างอิง https://www.dol.go.th/it/Pages/ความรู้เกี่ยวกับซอฟต์แวร์/การละเมิดลิขสิทธิ์ซอฟต์แวร์-และกฎหมายที่เกี่ยวข้อง.aspx

ก่อนลงมือเขียนโปรแกรมหรือซอฟต์แวร์

เพราะฉะนั้น ผู้ที่เป็นคนสร้าง ใช้งาน ควรเพิ่งรู้สิทธิ์ของตน ที่ควรกระทำได้ และไม่ควรกระทำ ในลิขสิทธิ์ของโปรแกรมคอมพิวเตอร์ และSoftware

การติดตั้งโปรแกรมสำหรับพัฒนาโปรแกรมภาษาซี (C/C++ IDE)

IDE #1 DEV C++ (Bloodshed Software)

IDE #1 DEV C++ (Bloodshed Software)

Download ที่ https://sourceforge.net/projects/orwelldevcpp/files/latest/download



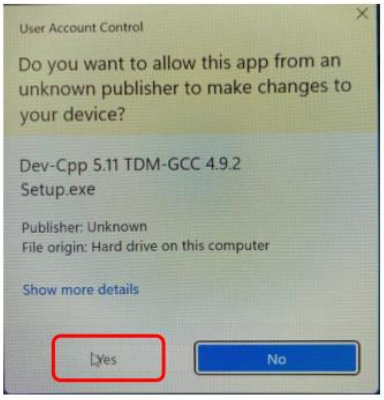
IDE #1 DEV C++ (Bloodshed Software)

เปิดตัวติดตั้ง



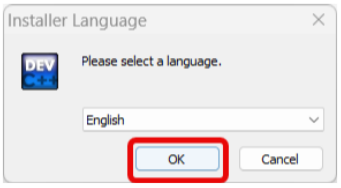
IDE #1 DEV C++ (Bloodshed Software)

กด Yes เพื่อ allow การเปลี่ยนแปลงในคอมพิวเตอร์



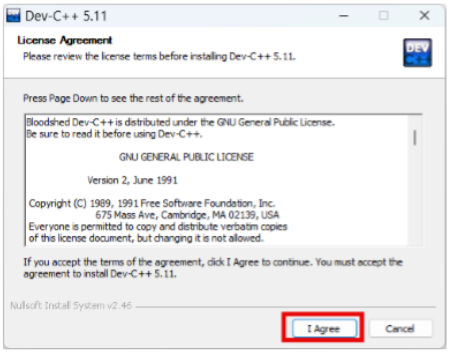
IDE #1 DEV C++ (Bloodshed Software)

เลือกภาษาแล้วกด OK



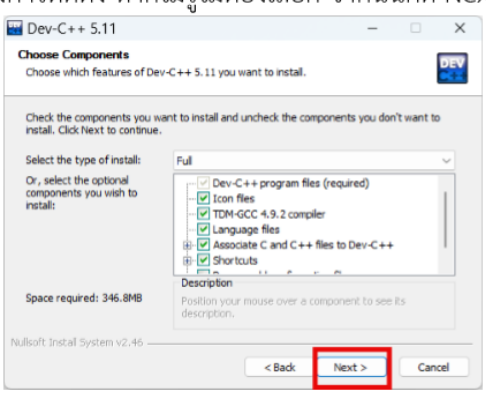
IDE #1 DEV C++ (Bloodshed Software)

อ่าน License Agreement ให้จบ และ กด I Agree เพื่อยอมรับ terms of agreement



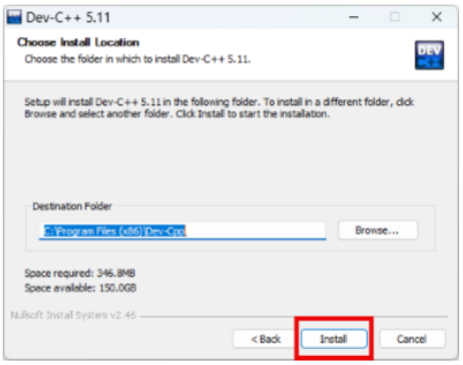
IDE #1 DEV C++ (Bloodshed Software)

เลือก Feature ที่ต้องการติดตั้ง หากไม่รู้ไม่ต้องเลือก จากนั้นกด Next >



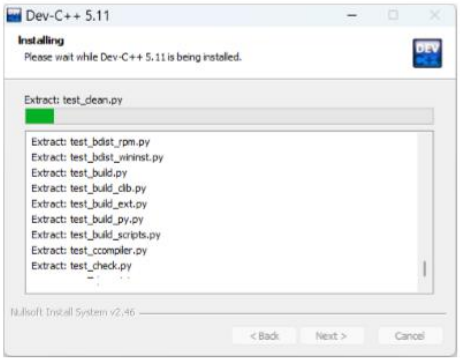
IDE #1 DEV C++ (Bloodshed Software)

เลือกที่ติดตั้งโปรแกรม หลังจากนั้นกด Install



IDE #1 DEV C++ (Bloodshed Software)

รอกระบวนการติดตั้งจนเสร็จ



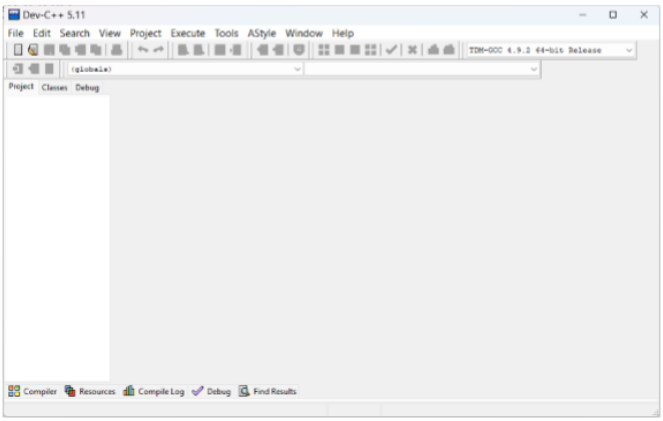
IDE #1 DEV C++ (Bloodshed Software)

กด Finish



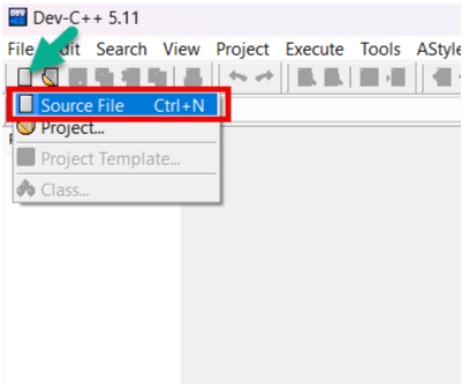
IDE #1 DEV C++ (Bloodshed Software)

เปิดโปรแกรม



IDE #1 DEV C++ (Bloodshed Software)

เปิด Source File เพื่อเขียนโปรแกรม



IDE #1 DEV C++ (Bloodshed Software)

ติดตั้งภาษาไทย เพื่อสามารถ comment (เขียนคำอธิบายโปรแกรม) เป็นภาษาไทยได้



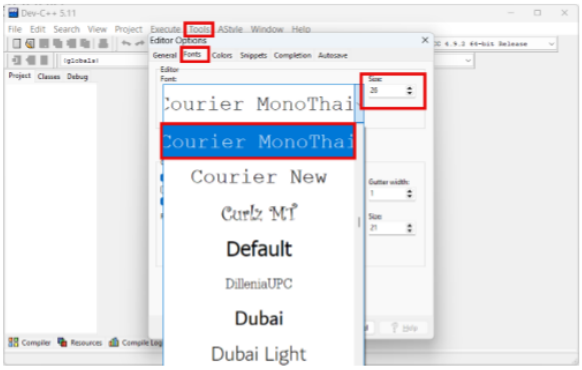
IDE #1 DEV C++ (Bloodshed Software)

ติดตั้ง font ภาษาไทย เพื่อสามารถ comment (เขียนคำอธิบายโปรแกรม) เป็นภาษาไทยได้



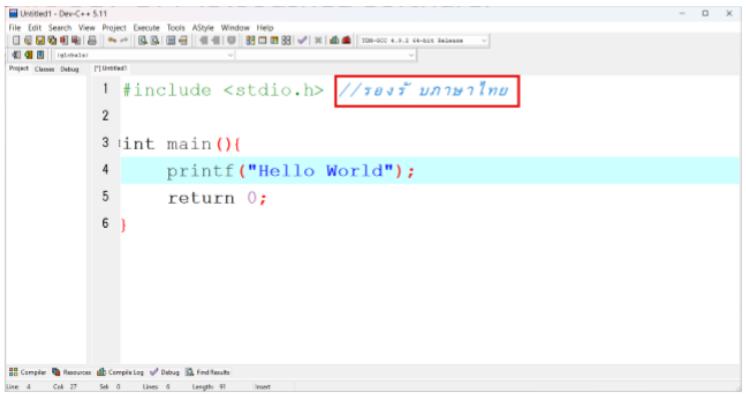
IDE #1 DEV C++ (Bloodshed Software)

ตั้งค่า font ภาษาไทย เพื่อสามารถ comment (เขียนคำอธิบายโปรแกรม) เป็นภาษาไทยได้



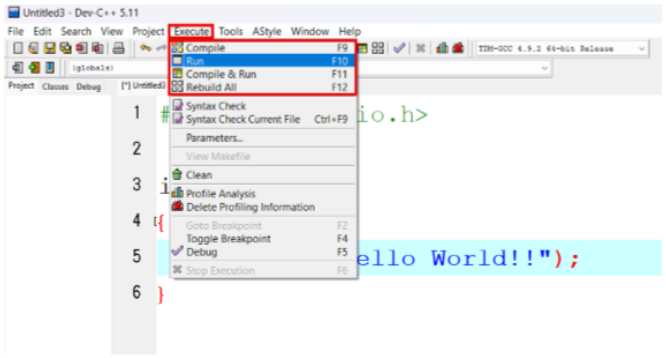
IDE #1 DEV C++ (Bloodshed Software)

ทดสอบเขียนโปรแกรม และเขียน Comment ภาษาไทย



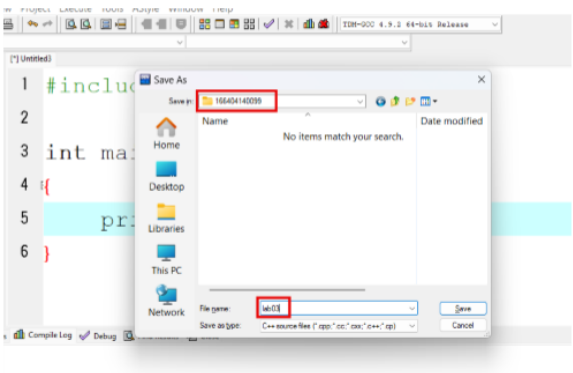
IDE #1 DEV C++ (Bloodshed Software)

สามารถกดปุ่ม F11 เพื่อ compile และ run โปรแกรมได้



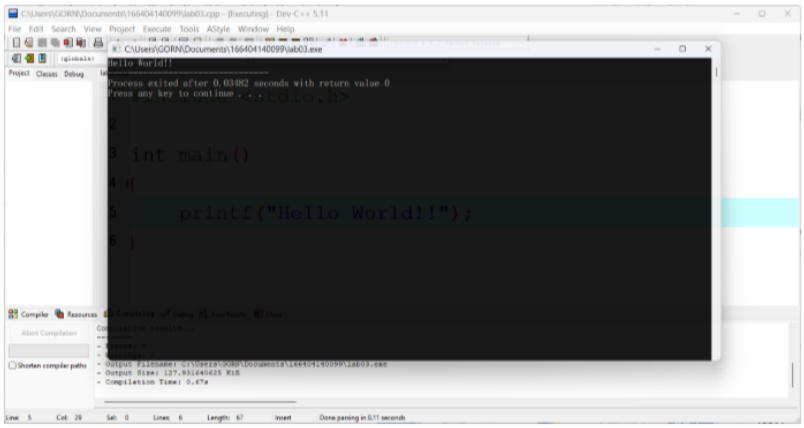
IDE #1 DEV C++ (Bloodshed Software)

หากยังไม่ได้ save source code โปรแกรมจะถามให้บันทึกก่อน ให้บันทึกใน path ที่ไม่มีชื่อภาษาไทย



IDE #1 DEV C++ (Bloodshed Software)

หากไม่มี syntex error จะขึ้นหน้า cli และแสดงผลลัพธ์การรันโปรแกรม



IDE #1 DEV C++ (Bloodshed Software)

หากมี syntex error ให้ไปตรวจสอบ bug (syntex) ในช่อง error list

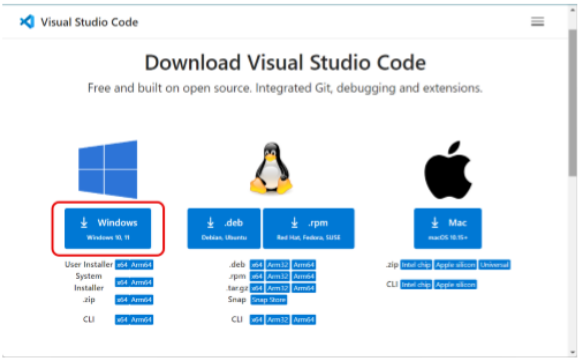


IDE #2 VS CODE & MinGW

IDE #2 VS CODE & MinGW

Download VS Code สำหรับคอมพิวเตอร์ปกติ ตัวอย่างนี้จะติดตั้ง x64 เวอร์ชั่น User Install

https://code.visualstudio.com/download



IDE #2 VS CODE & MinGW

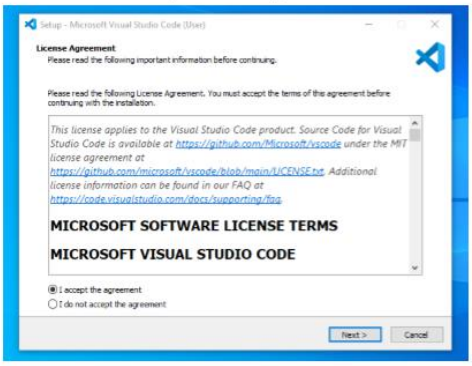
When the download finishes, then the Visual Studio Code Icon appears in the downloads folder. Click on the Installer icon to start the installation process of the Visual Studio Code.



REF: https://www.geeksforgeeks.org/how-to-install-visual-studio-code-on-windows/

IDE #2 VS CODE & MinGW

After the Installer opens, it will ask you to accept the terms and conditions of the Visual Studio Code. Click on I accept the agreement and then click the Next button.

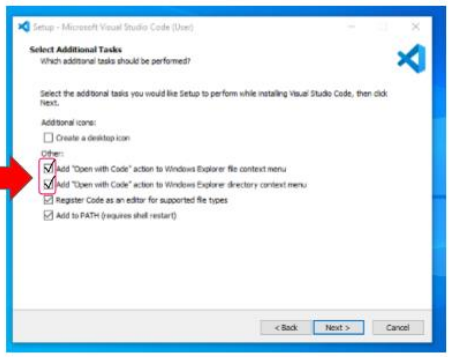


REF: https://www.geeksforgeeks.org/how-to-install-visual-studio-code-on-windows/

IDE #2 VS CODE & MinGW

Choose the location data for running the Visual Studio Code. It will then ask you to browse the location. Then click on the Next button.

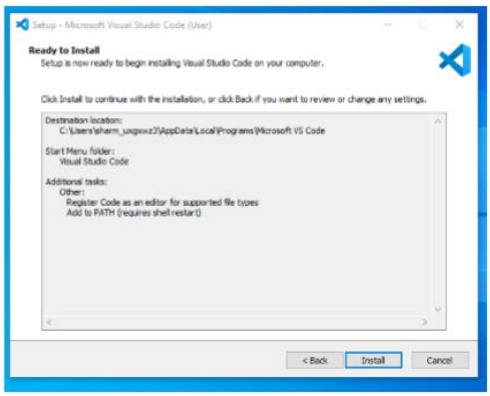
ติ๊กตรงนี้ด้วย



REF: https://www.geeksforgeeks.org/how-to-install-visual-studio-code-on-windows/

IDE #2 VS CODE & MinGW

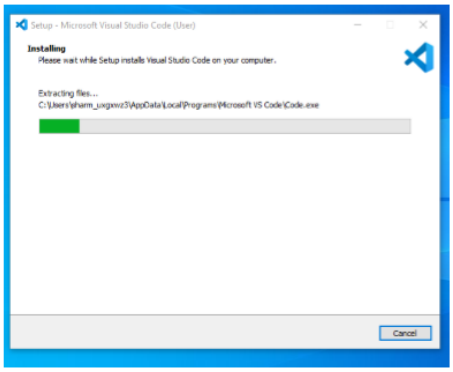
Then it will ask to begin the installation setup. Click on the Install button.



REF: https://www.geeksforgeeks.org/how-to-install-visual-studio-code-on-windows/

IDE #2 VS CODE & MinGW

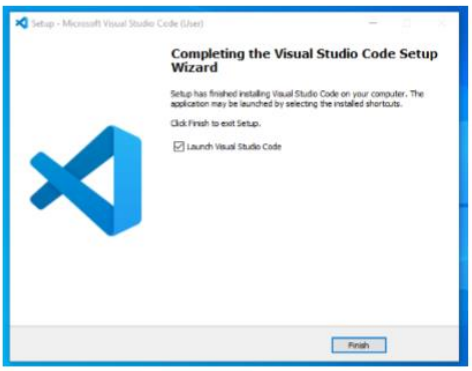
After clicking on Install, it will take about 1 minute to install the Visual Studio Code on your device.



REF: https://www.geeksforgeeks.org/how-to-install-visual-studio-code-on-windows/

IDE #2 VS CODE & MinGW

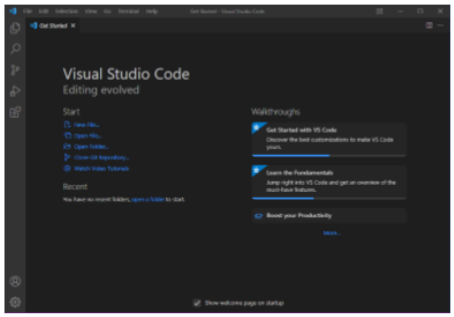
After the Installation setup for Visual Studio Code is finished, it will show a window like this below. Tick the “Launch Visual Studio Code” checkbox and then click Next.



REF: https://www.geeksforgeeks.org/how-to-install-visual-studio-code-on-windows/

IDE #2 VS CODE & MinGW

After the previous step, the Visual Studio Code window opens successfully. Now you can create a new file in the Visual Studio Code window and choose a language of yours to begin your programming journey!

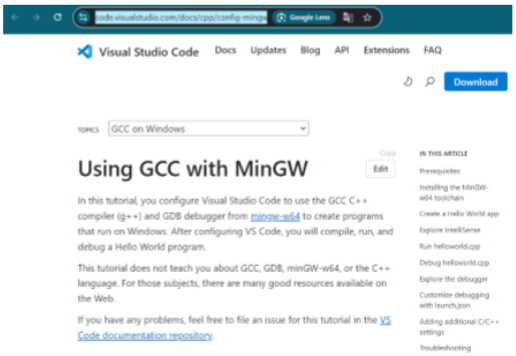


REF: https://www.geeksforgeeks.org/how-to-install-visual-studio-code-on-windows/

IDE #2 VS CODE & MinGW

ติดตั้ง MinGW เพื่อใช้ชุดคอมไพลเลอร์ของ GCC และ G++

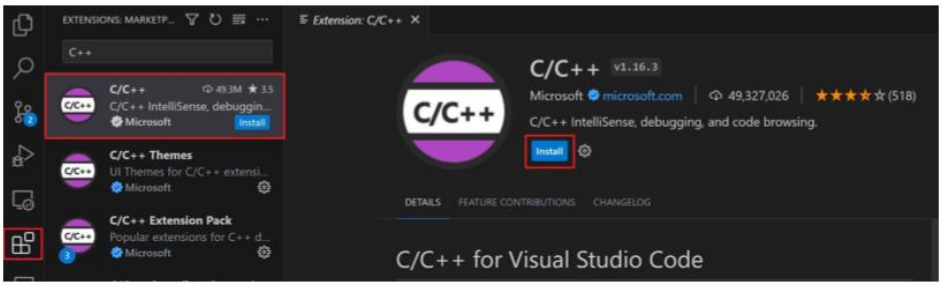
คู่มือการติดตั้ง MinGW > https://code.visualstudio.com/docs/cpp/config-mingw



IDE #2 VS CODE & MinGW

ติดตั้ง Extension **C/C++ extension for VS Code**

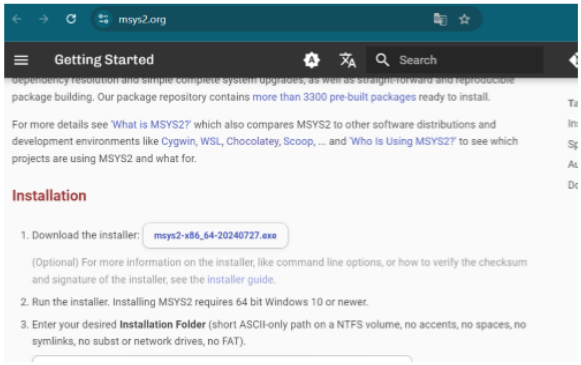
Install the C/C++ extension for VS Code. You can install the C/C++ extension by searching for 'C++' in the Extensions view (`Ctrl+Shift+X`).



IDE #2 VS CODE & MinGW

ติดตั้ง Installing the MinGW-w64 toolchain โดยดาวน์โหลดจาก https://www.msys2.org/

หรือ https://github.com/msys2/msys2-installer/releases/download/2024-01-13/msys2-x86_64-20240113.exe



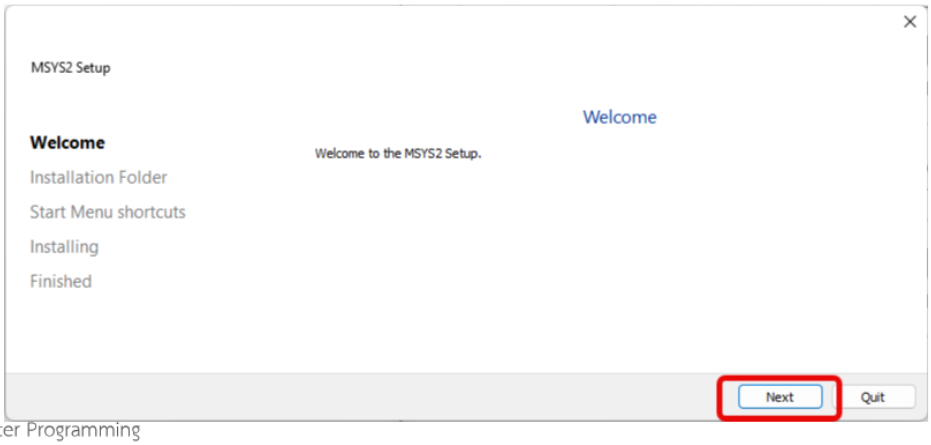
IDE #2 VS CODE & MinGW

เปิดตัวติดตั้ง msys2 ดังภาพ



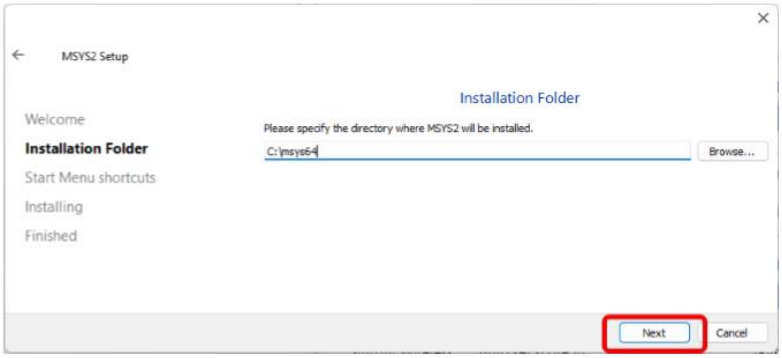
IDE #2 VS CODE & MinGW

กด Next



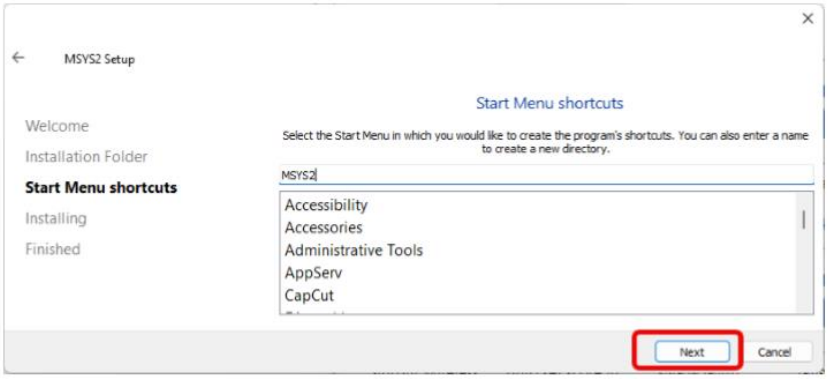
IDE #2 VS CODE & MinGW

ติดตั้งที่ C:\msys64 จากนั้นกด Next



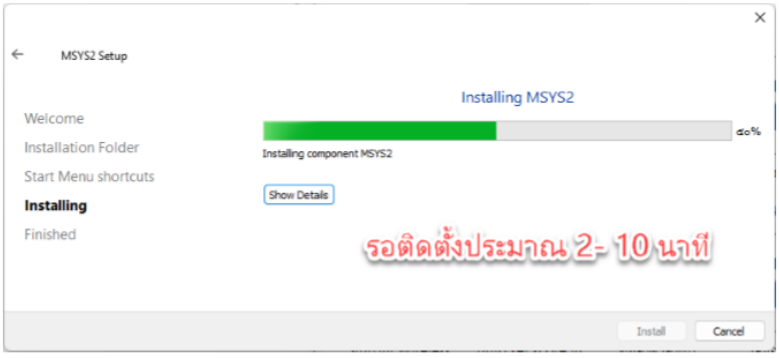
IDE #2 VS CODE & MinGW

กด Next



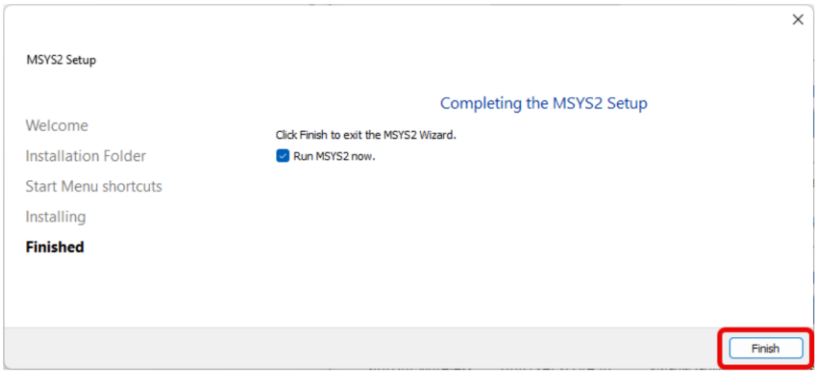
IDE #2 VS CODE & MinGW

รอกระบวนการติดตั้ง msys2



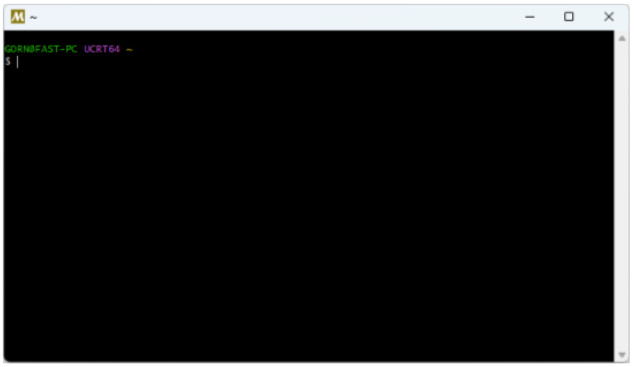
IDE #2 VS CODE & MinGW

หลังจากติดตั้งเสร็จ ให้กด Finish โดยต้องติ๊ก Run MSYS2 now. ด้วย



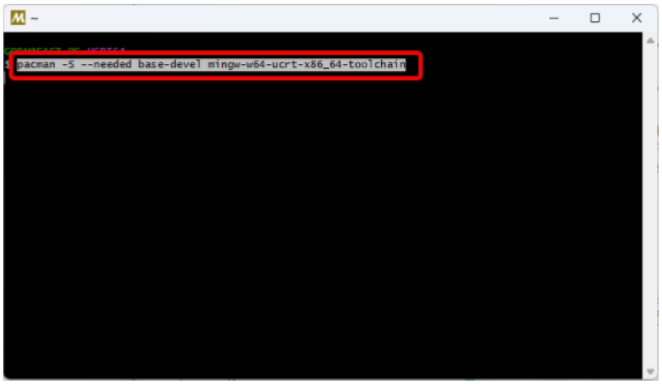
IDE #2 VS CODE & MinGW

หาหน้าต่าง MSYS2



IDE #2 VS CODE & MinGW

ใช้ pacman ติดตั้ง mingw-w64 โดยใช้คำสั่ง pacman -S --needed base-devel mingw-w64-ucrt-x86_64-toolchain



IDE #2 VS CODE & MinGW

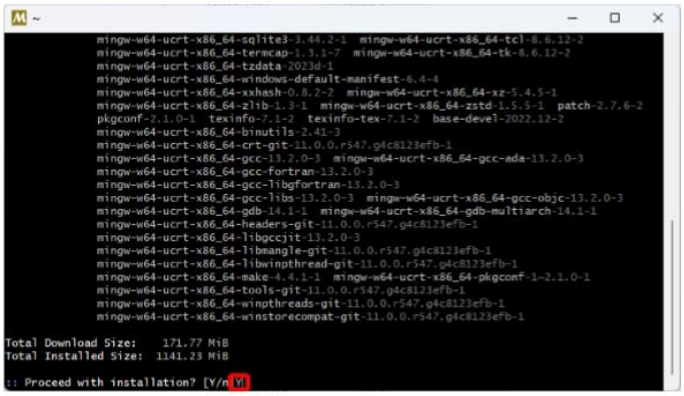
กด Enter เพื่อติดตั้งทั้งหมด

```
$ pacman -S --needed base-devel mingw-w64-ucrt-x86_64-toolchain
warning: base-devel-2022.12-2 is up to date -- skipping
:: There are 19 members in group mingw-w64-ucrt-x86_64-toolchain:
:: Repository ucrt64
   1) mingw-w64-ucrt-x86_64-binutils  2) mingw-w64-ucrt-x86_64-crt-git
   3) mingw-w64-ucrt-x86_64-gcc  4) mingw-w64-ucrt-x86_64-gcc-ada
   5) mingw-w64-ucrt-x86_64-gcc-fortran  6) mingw-w64-ucrt-x86_64-gcc-libgfortran
   7) mingw-w64-ucrt-x86_64-gcc-libs  8) mingw-w64-ucrt-x86_64-gcc-objc
   9) mingw-w64-ucrt-x86_64-gdb  10) mingw-w64-ucrt-x86_64-gdb-multiarch
   11) mingw-w64-ucrt-x86_64-headers-git  12) mingw-w64-ucrt-x86_64-libgccjit
   13) mingw-w64-ucrt-x86_64-libmangle-git  14) mingw-w64-ucrt-x86_64-libwinpthread-git
   15) mingw-w64-ucrt-x86_64-make  16) mingw-w64-ucrt-x86_64-pkgconf
   17) mingw-w64-ucrt-x86_64-tools-git  18) mingw-w64-ucrt-x86_64-winpthreads-git
   19) mingw-w64-ucrt-x86_64-winstorecompat-git

Enter a selection (default=all):
```

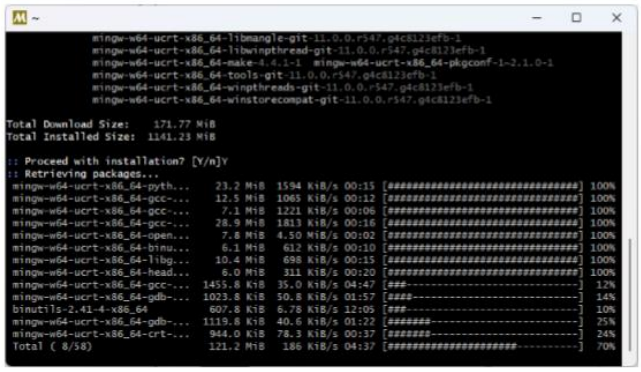
IDE #2 VS CODE & MinGW

เพื่อพิมพ์ Y และกด Enter เพื่อเริ่มติดตั้ง



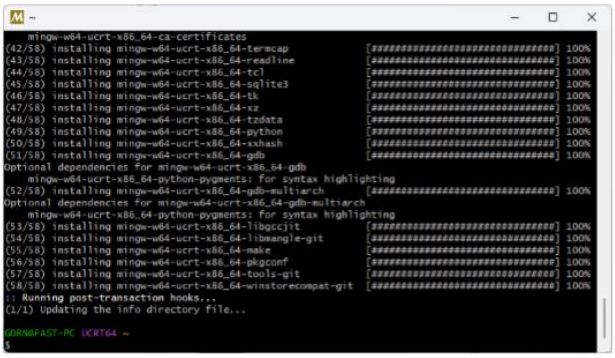
IDE #2 VS CODE & MinGW

รอจน Package ดาวน์โหลดและติดตั้งครบทั้งหมด



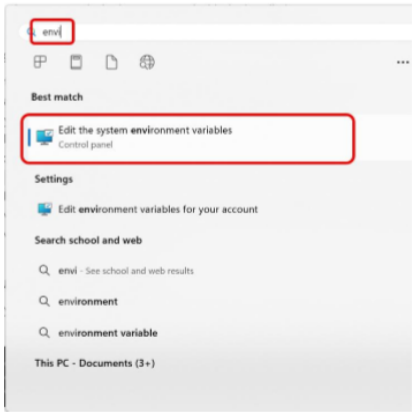
IDE #2 VS CODE & MinGW

เมื่อติดตั้งทั้งหมดแล้วจะขึ้นดังหน้าต่างนี้ สามารถกดปิดหน้าต่าง MSYS2 นี้ได้เลย



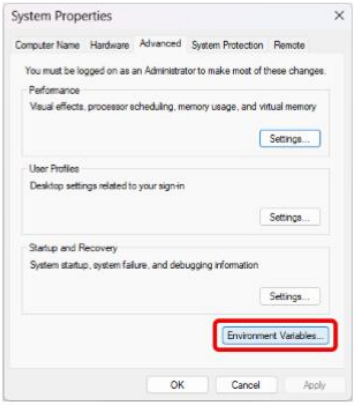
IDE #2 VS CODE & MinGW

กด Start แล้ว ค้นหาว่า envi จากนั้นเลือก Edit the system environment variables



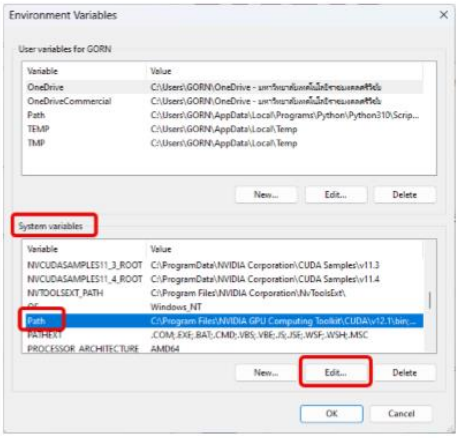
IDE #2 VS CODE & MinGW

คลิก Environment Variables...



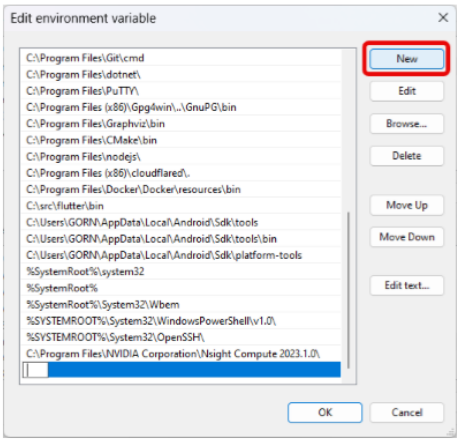
IDE #2 VS CODE & MinGW

หา System variables ชื่อ Path และคลิก Edit...



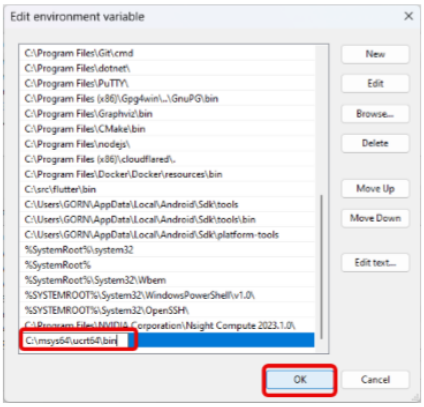
IDE #2 VS CODE & MinGW

คลิก New



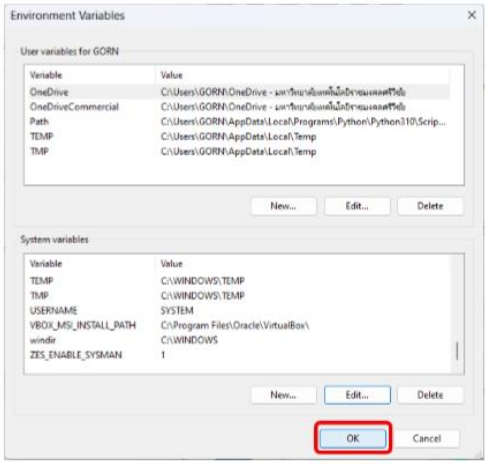
IDE #2 VS CODE & MinGW

นำ C:\msys64\ucrt64\bin ไปวาง และกด OK



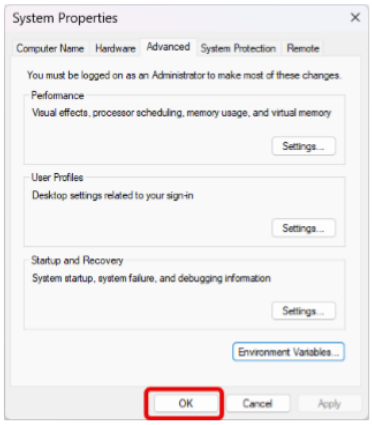
IDE #2 VS CODE & MinGW

กด OK เพื่อปิดหน้าต่าง



IDE #2 VS CODE & MinGW

กด OK เพื่อปิดหน้าต่าง



IDE #2 VS CODE & MinGW

เปิดโปรแกรม Power Shell

และพิมพ์คำสั่ง

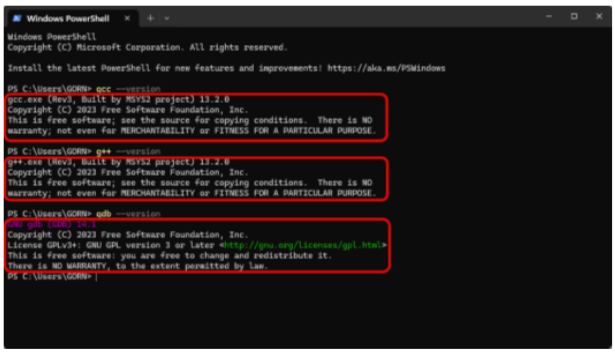
gcc --version

g++ --version

gdb --version

หากขึ้นดังรูปแสดงว่า

สามารถใช้งาน compiler และ debugger ได้



IDE #2 VS CODE & MinGW

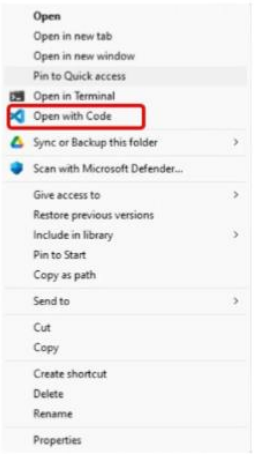
สร้างโฟลเดอร์สำหรับเก็บ Source code ซึ่ง **Path ของโฟลเดอร์ไม่ควรมีภาษาไทย** เพราะจะทำให้ compiler ไม่สามารถเข้าถึงไฟล์ได้

จากนั้นคลิกขวา เลือก Show more options

และเลือก Open with Code

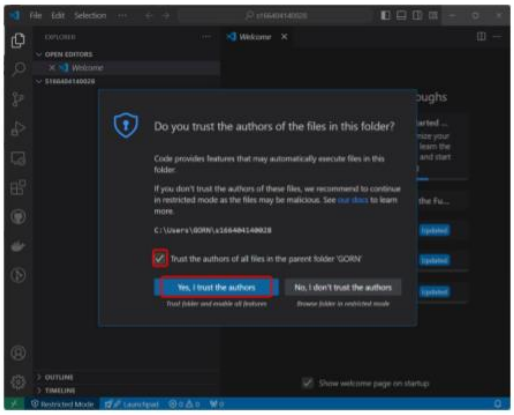


s166404
140028



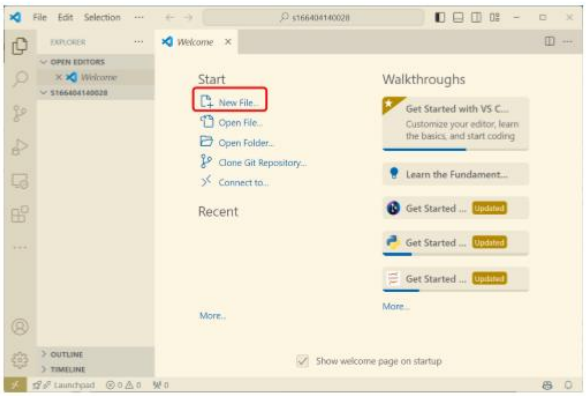
IDE #2 VS CODE & MinGW

กด Yes, I trust the authors เพื่อให้สามารถ execute ไฟล์ที่เราเขียนโปรแกรมได้



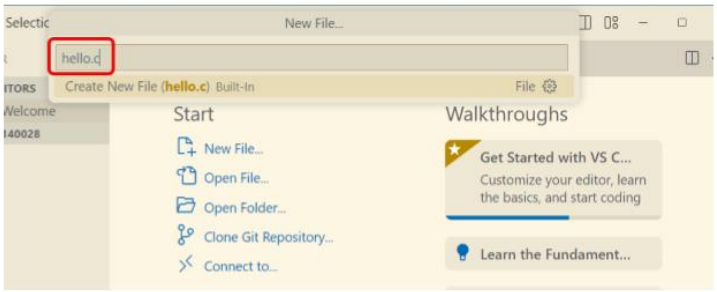
IDE #2 VS CODE & MinGW

คลิก New Files



IDE #2 VS CODE & MinGW

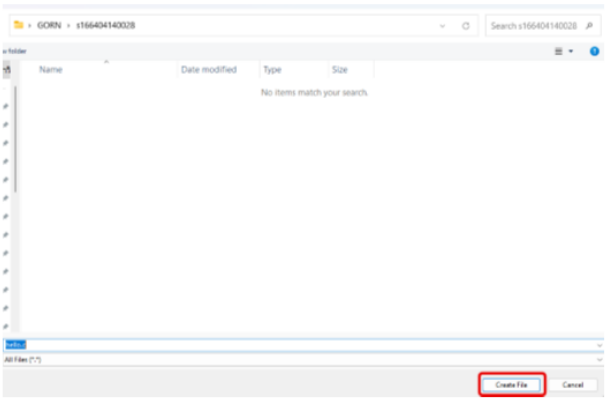
ตั้งชื่อไฟล์ hello.c แล้วกด Enter



Computer Programming 71

IDE #2 VS CODE & MinGW

คลิก Create File



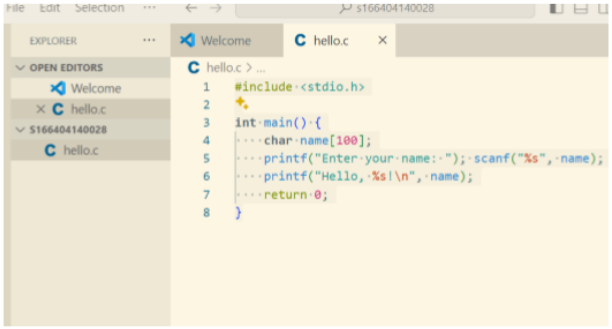
Computer Programming 72

IDE #2 VS CODE & MinGW

พิมพ์ Code ดังต่อไปนี้

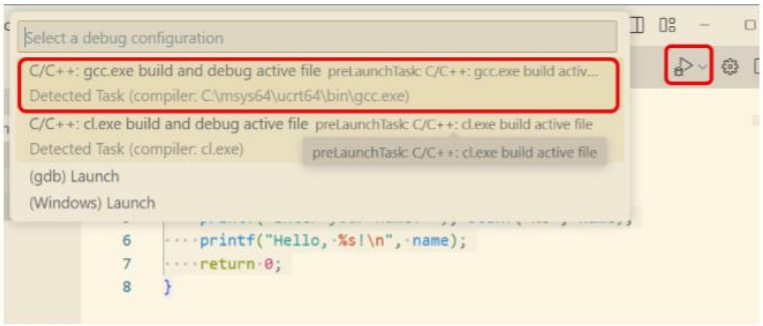
```c
#include <stdio.h>

int main() {
    char name[100];
    printf("Enter your name: ");
    scanf("%s", name);
    printf("Hello, %s!\n", name);
    return 0;
}
```



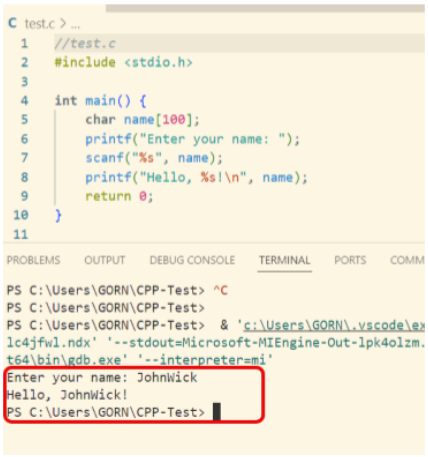
IDE #2 VS CODE & MinGW

กดปุ่ม debug และเลือก C/C++: gcc.exe .....



IDE #2 VS CODE & MinGW

คลิกที่แถบ TERMINAL จากนั้น ให้ลองพิมพ์ชื่อนศ.ลงไปแล้วกดปุ่ม Enter จะได้ผลลัพธ์ดังภาพ



IDE #3 ONLINE GDB

IDE #3 ONLINE GDB

สามารถเข้าเว็บไซต์ https://www.onlinegdb.com/ เพื่อเขียนโปรแกรมได้เลย

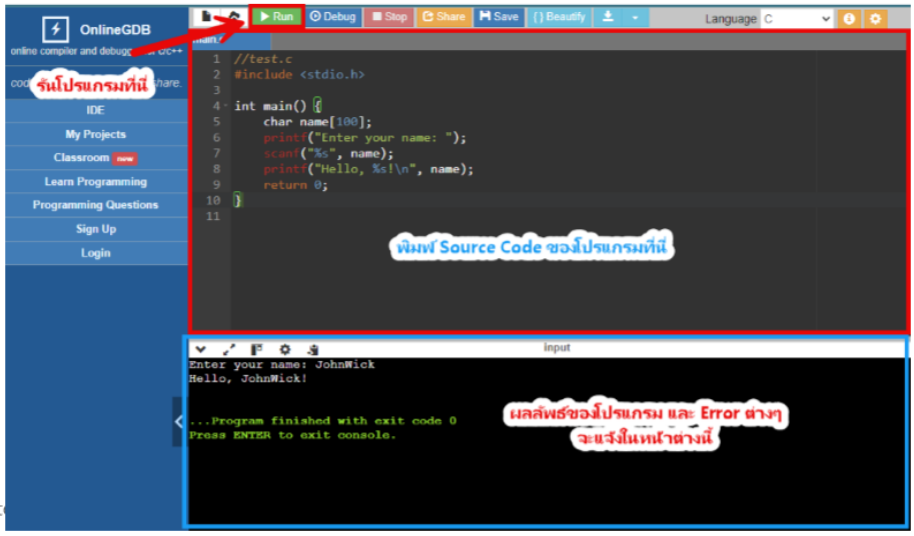


IDE #3 ONLINE GDB

เปลี่ยนภาษาที่มุมขวาบน



IDE #3 ONLINE GDB



พื้นฐานและโครงสร้างภาษาซี

เริ่มต้นกับ “Hello World”

```
#include <stdio.h>

int main()

{

        // Hello World Code

        printf("Hello World");

}
```


เริ่มต้นกับ “Hello World”

```
#include <stdio.h>

int main()

{

        // Hello World Code

        printf("Hello World");

}
```



เป็นคำสั่งในภาษา C ที่ใช้สำหรับนำเข้าหรือเรียกใช้ไลบรารีมาตรฐาน (Standard Input/Output Library) ของภาษา C ที่ชื่อว่า stdio.h (ย่อมาจาก "Standard Input Output header") ไลบรารีนี้มีฟังก์ชันที่เกี่ยวข้องกับการรับข้อมูลเข้าและแสดงผลข้อมูลออก เช่น ฟังก์ชัน printf() สำหรับการพิมพ์ข้อมูลออกทางหน้าจอ และฟังก์ชัน scanf() สำหรับการรับข้อมูลเข้าจากผู้ใช้

เริ่มต้นกับ “Hello World”

```
#include <stdio.h>
int main()
{
        // Hello World Code
        printf("Hello World");
}
```



int main() เป็น**ฟังก์ชันหลัก (main function)** ในภาษา C และ C++ ที่เป็นจุดเริ่มต้นการทำงานของโปรแกรมทุกโปรแกรม เมื่อโปรแกรมถูกเรียกใช้งาน การทำงานทั้งหมดของโปรแกรมจะเริ่มต้นที่ฟังก์ชันนี้

ฟังก์ชัน main จะต้องมีอยู่ในทุกโปรแกรม C หรือ C++ และรูปแบบการเขียนมาตรฐานที่นิยมใช้คือ int main() ซึ่งหมายความว่าฟังก์ชันนี้จะคืนค่าเป็นจำนวนเต็ม (ประเภท int) ให้กับระบบปฏิบัติการเมื่อโปรแกรมสิ้นสุดการทำงาน ค่าที่คืนกลับมานั้นมักจะเป็น 0 หากโปรแกรมทำงานสำเร็จโดยไม่มีข้อผิดพลาด

เริ่มต้นกับ “Hello World”

```
#include <stdio.h>
int main()
{
        // Hello World Code
        printf("Hello World");
}
```





{ ..... } เป็นการระบุขอบเขตของ ฟังก์ชัน main โดย { ใช้บอกจุดเริ่มต้น และ } ใช้บอกจุดสิ้นสุด

เริ่มต้นกับ “Hello World”

```
#include <stdio.h>

int main()

{

    // Hello World Code

    printf("Hello World");

}
```

Comment หรือ คำอธิบายโปรแกรม สามารถเขียนข้อความใดๆก็ที่โปรแกรมเมอร์ต้องการ

Computer Programming 85

เริ่มต้นกับ “Hello World”

```
#include <stdio.h>

int main()

{

    // Hello World Code

    printf("Hello World");

}
```

เป็นฟังก์ชันในภาษา C ที่ใช้สำหรับพิมพ์ข้อความ "Hello World" ออกทางหน้าจอ console หรือ cli

Computer Programming 86

สรุปโครงสร้างหลักของภาษาซี

ส่วนของ Preprocessor เช่น #include #define #pragma

```
int main()
{
        ส่วนของ Code ในฟังก์ชัน main
}
```


คำอธิบายโปรแกรม (Comment) // /* */

Comment เป็นข้อมูลที่ช่วยอธิบายการทำงานของโปรแกรม, ฟังก์ชัน หรือคำสั่ง เพื่อให้เข้าใจโค้ดได้ง่ายขึ้น

- สามารถเขียนด้วยภาษาใดก็ได้ เช่น ภาษาไทย
- ไม่มีผลต่อการทำงานของโปรแกรม
- เมื่อทดลองแก้ไขโปรแกรม เราสามารถใช้ comment เพื่อปิดบางส่วนของโค้ดไว้แทนการลบ เผื่อเรียกใช้งานอีกครั้งในภายหลัง

รูปแบบการเขียน comment มี 2 แบบ:

1. // ใช้สำหรับเขียนคำอธิบายตั้งแต่จุดนี้จนจบบรรทัด
2. /* ... */ ใช้สำหรับคำอธิบายหลายบรรทัด ข้อความจะเริ่มจาก /* และจบที่ */

คำอธิบายโปรแกรม (Comment) // /* */

แบบ // (คำอธิบายบรรทัดเดียว)

```c
// ฟังก์ชันหลักของโปรแกรม
int main() {
    printf("Hello, World!\n"); // แสดงข้อความ Hello, World!
    return 0;
}
```


คำอธิบายโปรแกรม (Comment) // /* */

แบบ /* ..... */ (คำอธิบายหลายบรรทัด)

```c
/*
   ฟังก์ชันหลักของโปรแกรม
   แสดงข้อความ "Hello, World!"
*/
int main() {
    printf("Hello, World!\n");
    return 0;
}
```

Preprocessor

Preprocessor คือส่วนของโปรแกรมที่เริ่มต้นด้วยเครื่องหมาย # เช่น #define

- คำสั่ง #define symbol another_symbol ใช้แทนที่ทุกคำที่กำหนด (symbol) ด้วยคำใหม่ (another_symbol) ในซอร์สโค้ด
- มักใช้สำหรับกำหนดค่าคงที่ เพื่อให้การแก้ไขทำได้ง่ายขึ้น

ตัวอย่าง: หากมีเลข 50 ใช้ในหลายจุดของโปรแกรมเพื่อเป็นเกณฑ์ตัดเกรด การใช้ #define จะช่วยให้เปลี่ยนค่าได้ง่ายขึ้น

```
#define FAIL_SCORE 50
```

ค่าคงที่ FAIL_SCORE จะถูกใช้แทนที่ด้วยเลข 50 ในโปรแกรม

Preprocessor

#include <stdio.h> เป็นคำสั่งที่บอกให้โปรแกรมนำ header file มาตรฐาน stdio.h เข้ามา เนื่องจากโปรแกรมต้องใช้คำสั่งสำหรับการรับส่งข้อมูลกับ input output

ตัวอย่างเช่น stdio.h มีข้อมูลของฟังก์ชัน printf() ที่ใช้ในโปรแกรม

stdio.h

- เป็น header file มาตรฐานในภาษา C/C++ สำหรับการจัดการอินพุตและเอาต์พุต
- รวมฟังก์ชันสำหรับการแสดงผลและรับข้อมูล เช่น printf(), scanf()
- ใช้ในการจัดการไฟล์ เช่น fopen(), fclose(), fread(), fwrite()
- ช่วยในการจัดการการรับส่งข้อมูลจาก/ไปยังคอนโซลหรือไฟล์
- จำเป็นต้องใช้ #include <stdio.h> เพื่อเรียกใช้งานฟังก์ชันเหล่านี้ในโปรแกรม

int main()

- เป็นฟังก์ชันหลักที่ทุกโปรแกรมในภาษา C/C++ จะเริ่มทำงานจากจุดนี้
- คืนค่าประเภท int (จำนวนเต็ม) ให้ระบบปฏิบัติการเมื่อโปรแกรมทำงานเสร็จสิ้น
- โดยปกติจะคืนค่า 0 (return 0;) เพื่อบอกว่าการทำงานสำเร็จ (หรือคืนค่าตัวเลขอื่นในกรณีมีข้อผิดพลาด)
- สามารถรับอาร์กิวเมนต์จากบรรทัดคำสั่งในรูปแบบ int main(int argc, char *argv[])
- เป็นจุดเริ่มต้นและจุดสิ้นสุดของการทำงานของโปรแกรม

int main()

```c
// ฟังก์ชันหลักของโปรแกรม
int main() {
    printf("Hello, World!\n"); // แสดงข้อความ Hello, World!
    return 0;
}
```

วงเล็บปีกกาเปิด { ระบุจุดเริ่มต้นของฟังก์ชันหรือขอบเขตของโค้ด

วงเล็บปีกกาปิด } ระบุจุดสิ้นสุดของฟังก์ชันหรือขอบเขตของโค้ด

เครื่องหมาย ; ใช้เพื่อบอกว่าคำสั่งจบลง หากไม่ใส่จะเกิด syntax error

การย่อหน้า ช่วยให้โค้ดอ่านง่ายขึ้นและทำให้การหาขอบเขตของคำสั่งสะดวก

printf (“Hello World”) ;

printf เป็นฟังก์ชันในภาษา C ใช้สำหรับแสดงผลข้อความไปยังหน้าจอ

ข้อความ "Hello World" คือข้อความที่ต้องการแสดง

เครื่องหมายวงเล็บ () ใช้ในการส่งค่า (ข้อความ) ไปยังฟังก์ชัน printf

เครื่องหมาย ; ใช้บอกว่าคำสั่งนี้สิ้นสุดแล้ว

เมื่อรันโปรแกรม จะแสดงข้อความ "Hello World!!!" บนหน้าจอ

ตัวอย่างโปรแกรม

```
# include <stdio.h>
int main ()
{
   printf("My name is John Wick \n");
   return 0;
}
```



Computer Programming 97

จงเขียนโปรแกรมแสดงชื่อ นามสกุล พร้อมกับรหัสนักศึกษา

```
# include <stdio.h>
int main ()
{
   ..........
   return 0;
}
```

Computer Programming 98

จงเขียนโปรแกรมแสดงชื่อ นามสกุล พร้อมกับรหัสนักศึกษา

```
# include <stdio.h>

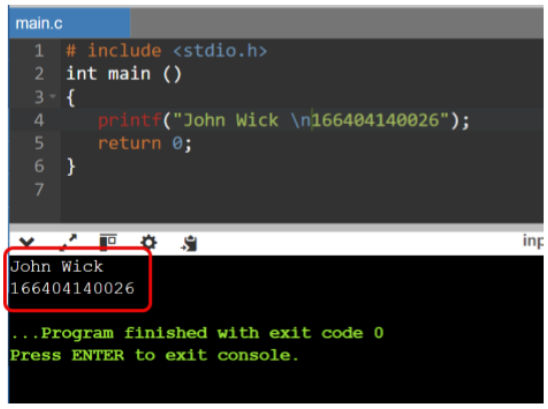
int main ()

{

   printf("John Wick \n 166404140026");

   return 0;

}
```



ตัวแปร (variable) และตัวดำเนินการ (operator)

ตัวแปร (variable)

ตัวแปร = กล่องเก็บข้อมูลในโปรแกรมที่เราสามารถใส่และดึงข้อมูลออกมาใช้งานได้

เช่น ถ้าเรามีตัวแปรชื่อว่า "อายุ" เราก็สามารถใส่ตัวเลขอายุลงไปในตัวแปรนี้และเรียกใช้อีกครั้งเมื่อเราต้องการ

ตัวแปรสำคัญเพราะมันช่วยให้โปรแกรมเก็บและจัดการข้อมูลได้ง่าย ทำให้การเขียนโปรแกรมมีความยืดหยุ่นและเข้าใจง่ายขึ้น

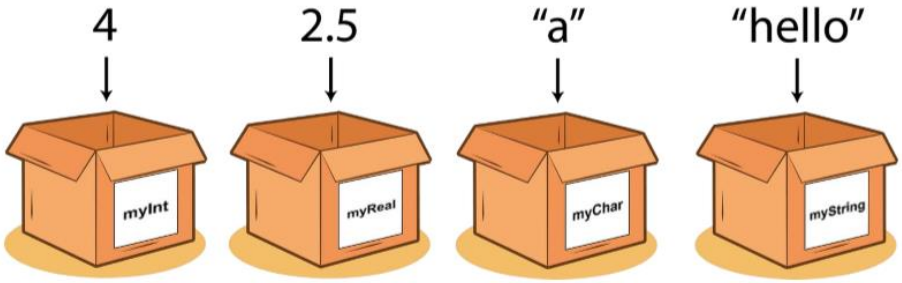




อ้างอิงรูปภาพ: Generate AI Image from DALL-E

ตัวแปร (variable)

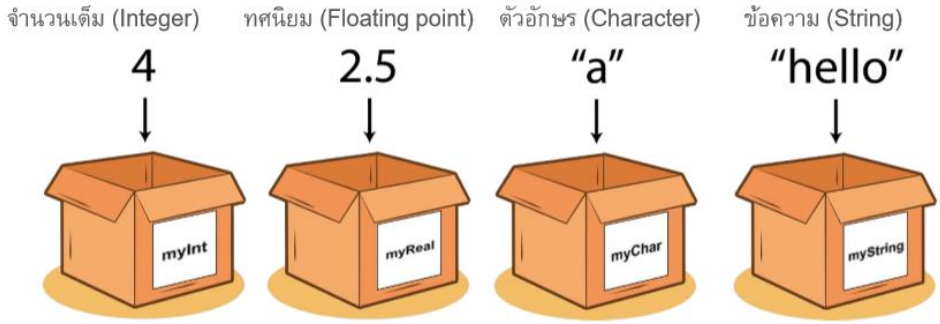
ตัวแปร สามารถตั้งชื่อได้ เป็นภาษาอังกฤษ **โดยจะใช้ชื่อในการเรียกใช้งาน**



IMG REF : https://stevenpcurtis.medium.com/what-is-a-variable-3447ac1331b9

ตัวแปร (variable)

ตัวแปร **มีหลายชนิด ต้องใช้ให้เหมาะสมกับข้อมูลที่จะเก็บ**



Computer Programming IMG REF : https://stevenpcurtis.medium.com/what-is-a-variable-3447ac1331b9 103

ตัวแปร (variable)

ตัวแปร **มีชื่อ ต้องตั้งชื่อตัวแปรให้สื่อกับข้อมูลที่เก็บ**

นศ. พอเดาได้ไหม ตัวแปรไหนเก็บค่าอะไร

listName

x2

xyz

x1

birthday

firstname

aaa

age

var1

lastname

Computer Programming 104

ตัวแปร (variable)

Snake Case และ **Camel Case** เป็นรูปแบบการตั้งชื่อตัวแปรในโปรแกรมที่นิยมใช้กัน โดยแต่ละรูปแบบมีวิธีเขียนต่างกัน ดังนี้

- **Snake Case** จะใช้เครื่องหมายขีดล่าง (underscore) _ เชื่อมคำที่เป็นส่วนประกอบของชื่อ เช่น total_price หรือ user_name ทำให้แต่ละคำแยกจากกันอย่างชัดเจน
- **Camel Case** จะเขียนคำแรกด้วยตัวอักษรเล็ก และคำถัดไปจะขึ้นต้นด้วยตัวอักษรใหญ่เพื่อแยกคำ เช่น totalPrice หรือ userName

ทั้งสองรูปแบบมีเป้าหมายเพื่อทำให้ชื่อของตัวแปรอ่านง่ายและชัดเจนว่าเป็นการรวมหลายคำเข้าด้วยกัน

ตัวแปร (variable)

ภาษา C จะให้ความสำคัญกับชนิดข้อมูลอย่างเคร่งครัด

- **int** เก็บตัวเลขจำนวนเต็ม
- **float** เก็บตัวเลขทศนิยม
- **double** เก็บตัวเลขทศนิยมที่มีความแม่นยำสูงกว่า
- **char** เก็บตัวอักษร

ข้อมูลบางชนิดสามารถใช้ตัวปรับอย่าง **signed (ติดลบได้)**, **unsigned (ไม่เอาที่ติดลบ)**, **short (เก็บแค่น้อยๆ)**, และ **long (เก็บได้กว้างขึ้น)** เพื่อปรับขนาดหรือความสามารถในการเก็บข้อมูลได้

ตัวแปร (variable)

ภาษา C จะให้ความสำคัญกับชนิดข้อมูลอย่างเคร่งครัด

- int เก็บตัวเลขจำนวนเต็ม
- float เก็บตัวเลขทศนิยม
- double เก็บตัวเลขทศนิยมที่มีความแม่นยำสูงกว่า
- char เก็บตัวอักษร

ข้อมูลบางชนิดสามารถใช้ตัวปรับอย่าง **signed (ติดลบได้)**, **unsigned (ไม่เอาที่ติดลบ)**, **short (เก็บแค่น้อยๆ)**, และ **long (เก็บได้กว้างขึ้น)** เพื่อปรับขนาดหรือความสามารถในการเก็บข้อมูลได้

ตัวแปร (variable)

ชนิด (Type)	ขนาดในการจัดเก็บ (Storage Size)	ช่วงที่สามารถเก็บค่าได้ (Value Range)
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767
		or -2,147,483,648 to 2,147,483,647

ตัวแปร (variable)

ชนิด (Type)	ขนาดในการจัดเก็บ (Storage Size)	ช่วงที่สามารถเก็บค่าได้ (Value Range)
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

ตัวแปร (variable)

ชนิด (Type)	ขนาดในการจัดเก็บ (Storage Size)	ช่วงที่สามารถเก็บค่าได้ (Value Range)	ความแม่นยำ (Precision)
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

ตัวแปร (variable)

ทำไมต้องมีตัวแปรในโปรแกรม

- จัดเก็บข้อมูล > เก็บค่าหรือข้อมูลต่างๆ ที่เราต้องการใช้ในโปรแกรม เช่น ตัวเลข ข้อความ หรือผลลัพธ์จากการคำนวณ
- นำข้อมูลมาใช้ซ้ำได้ > เมื่อเก็บข้อมูลไว้ในตัวแปร เราสามารถเรียกใช้งานข้อมูลนั้นได้หลายครั้งโดยไม่ต้องเขียนซ้ำ
- ทำให้โปรแกรมยืดหยุ่น > ตัวแปรช่วยให้โปรแกรมทำงานกับข้อมูลที่เปลี่ยนแปลงได้ เช่น การคำนวณค่าตัวเลขต่างๆ ที่อาจเปลี่ยนไปตามการทำงาน
- ทำให้โค้ดอ่านง่ายและแก้ไขได้ง่าย > การใช้ชื่อตัวแปรช่วยให้โค้ดมีความชัดเจน เข้าใจง่าย และแก้ไขได้สะดวก ไม่ต้องยุ่งกับการจัดการหน่วยความจำโดยตรง

ตัวแปร (variable)

ค่าคงที่ในโปรแกรม (Constant)

จำนวนเต็ม (ฐานสิบ Decimal): เช่น 10, 20, -5

เลขฐาน 16 (Hexadecimal): เช่น 0x32, 0x5FB (ขึ้นต้นด้วย 0x)

เลขฐาน 8 (Octal): เช่น 013, 041, 07 (ขึ้นต้นด้วยศูนย์)

ทศนิยม: เช่น 7.2, 5.6, 0.002, 2e-3, -3.14159e2, -314.159

ตัวอักษร: เช่น 'c', '1', '5', ' ' (เว้นวรรค), '$'

ตัวอักษรพิเศษ: เริ่มต้นด้วย \ เช่น '\t' (แท็บ)

อักขระพิเศษ: ใช้ \ เพื่อยกเลิกความพิเศษ เช่น '\\' คือ \

ข้อความ String: เช่น "Hello", "" (ว่างเปล่า), " " (เว้นวรรคหนึ่งตัว)

ตัวแปร (variable)

ข้อมูลในโปรแกรม ที่ต้องดึงมาเก็บในตัวแปรมาจากไหนบ้าง

- โปรแกรมส่วนใหญ่เกี่ยวข้องกับการประมวลผลข้อมูล
- ข้อมูลภายในโปรแกรมเอง
  - ค่าคงที่
  - ตัวแปร
- ข้อมูลจากภายนอก
  - การกด keyboard
  - ไฟล์
  - ระบบเครือข่าย
  - อื่นๆ

รหัสมาตรฐานในการจัดเก็บตัวอักษร (Character Encoding)

ASCII (American Standard Code for Information Interchange) เป็นมาตรฐานการเข้ารหัสอักขระเพื่อใช้ในคอมพิวเตอร์

- ใช้ตัวเลข 7 บิต (0-127) แทนตัวอักษร สัญลักษณ์ และตัวเลข
- ตัวอักษร: เช่น 'A' จะเก็บเป็นเลข 65, 'a' จะเก็บเป็นเลข 97
- ตัวเลข: เช่น '0' จะเก็บเป็นเลข 48, '9' จะเก็บเป็นเลข 57
- สัญลักษณ์พิเศษ: เช่น '!' จะเก็บเป็นเลข 33, '@' จะเก็บเป็นเลข 64
- ใช้สำหรับการสื่อสารข้อมูลระหว่างอุปกรณ์คอมพิวเตอร์
- ASCIIแบบปกติ ไม่รองรับภาษาไทย
- ASCII ขยายไปยัง Extended ASCII เพื่อเพิ่มอักขระพิเศษ

Dec	Char/Description	Dec	Char	Dec	Char	Dec	Char
0	null	33	!	65	A	97	a
1	start of heading	34	"	66	B	98	b
2	start of text	35	#	67	C	99	c
3	end of text	36	$	68	D	100	d
4	end of trans. block	37	%	69	E	101	e
5	enquiry	38	&	70	F	102	f
6	acknowledge	39	'	71	G	103	g
7	bell	40	(	72	H	104	h
8	backspace	41	)	73	I	105	i
9	horizontal tab	42	*	74	J	106	j
10	new line	43	+	75	K	107	k
11	vertical tab	44	,	76	L	108	l
12	new page	45	-	77	M	109	m
13	carriage return	46	.	78	N	110	n
14	shift out	47	/	79	O	111	o
15	shift in	48	0	80	P	112	p
16	data link escape	49	1	81	Q	113	q
17	device control 1	50	2	82	R	114	r
18	device control 2	51	3	83	S	115	s
19	device control 3	52	4	84	T	116	t
20	device control 4	53	5	85	U	117	u
21	neg acknowledge	54	6	86	V	118	v
22	synchronous idle	55	7	87	W	119	w
23	end of trans. block	56	8	88	X	120	x
24	cancel	57	9	89	Y	121	y
25	end of medium	58	:	90	Z	122	z
26	substitute	59	;	91	[	123	{
27	escape	60	<	92	\	124	|
28	file separator	61	=	93	]	125	}
29	group separator	62	>	94	^	126	~
30	record separator	63	?	95	_	127	DEL
31	unit separator	64	@	96	`		

รหัสมาตรฐานในการจัดเก็บตัวอักษร (Character Encoding)



Unicode

- Unicode เป็นมาตรฐานการเข้ารหัสอักขระที่ครอบคลุมอักขระจากทุกภาษาทั่วโลก รวมถึงสัญลักษณ์พิเศษต่างๆ
- มีขนาดบิตที่ยืดหยุ่น รองรับอักขระมากกว่า 1 ล้านตัว
- UTF-8 เป็นรูปแบบการเข้ารหัสที่นิยมมากที่สุดสำหรับ Unicode ซึ่งใช้ 1 ถึง 4 ไบต์ในการเก็บอักขระแต่ละตัว
- UTF-8 เข้ากันได้กับ ASCII ซึ่งใช้เพียง 1 ไบต์ (8 บิต) สำหรับอักขระที่อยู่ในช่วง ASCII (0-127)
- อักขระที่อยู่นอกช่วง ASCII จะใช้หลายไบต์เพื่อเก็บข้อมูล
- UTF-16 และ UTF-32 เป็นอีกสองรูปแบบการเข้ารหัส Unicode โดย UTF-16 ใช้ 2 หรือ 4 ไบต์ และ UTF-32 ใช้ 4 ไบต์ต่ออักขระ
- Unicode ทำให้การแลกเปลี่ยนข้อมูลระหว่างระบบที่ใช้ภาษาต่างๆ เป็นไปได้อย่างราบรื่น

ตัวแปร (variable)

- ตัวแปรจะใช้หน่วยความจำของคอมพิวเตอร์ในการจดจำข้อมูล
- การเข้าถึงหน่วยความจำคอมพิวเตอร์มีความยากที่จะอ้างอิงตำแหน่งในหน่วยความจำ เราจึงใช้ชื่อของตัวแปรในการอ้างอิง
- ตัวแปร (ภาษาซี) ไม่สามารถใช้งานได้ทันที ต้องถูกประกาศก่อนการใช้งาน
- การประกาศตัวแปร ประกอบด้วย ชนิดข้อมูล ชื่อตัวแปร (อาจจะมีการกำหนดค่าเริ่มต้นด้วย) และตามด้วย ; (semi-colon) เช่น int x; หรือ int x = 2;
- ชื่อตัวแปร เป็นการประกอบกันระหว่าง ตัวอักษร ตัวเลข เครื่องหมาย _ (underscore) ทั้งนี้ห้ามขึ้นต้นด้วยตัวเลข และไม่เป็นคำสงวน

ตัวอย่างการประกาศตัวแปร (variable)

ประกาศตัวแปร cf มีชนิดเป็นตัวอักษร 1 ตัว

```
char cf;
```

ประกาศตัวแปร number มีชนิดเป็นเลขจำนวนเต็มพร้อมทั้งกำหนดให้มีค่าเท่ากับ 8

```
int number=8;
```

ประกาศตัวแปร price มีชนิดเป็นเลขทศนิยม

```
float price;
```


ตัวอย่างการประกาศตัวแปร (variable)

ประกาศตัวแปรชื่อ join_char ชนิดข้อมูลเป็นตัวอักษร 1 ตัว และกำหนดให้มีค่าเป็นตัวอักษร '/'

```
char join_char = '/';
```

ประกาศตัวแปร score และ Score ชนิดเป็นเลขจำนวนเต็ม โดย score มีค่าเท่ากับ 4 และ Score มีค่าเท่ากับ 1127:

```
int score = 4, Score = 1127;
```

ประกาศตัวแปรชื่อ value ชนิดเป็นเลขทศนิยม และกำหนดค่าให้เป็น 22.7

```
float value = 22.7;
```

ข้อกำหนดการตั้งชื่อตัวแปรในภาษาซี

อักษรตัวแรกจะต้องเป็นตัวอักษรภาษาอังกฤษหรือเครื่องหมาย _ (underline character)

เช่น name _name

สามารถตั้งชื่อตัวแปรโดยใช้ตัวเลขร่วมกับตัวอักษรภาษาอังกฤษ ได้แต่ห้ามใช้ตัวเลขเป็นตัวอักษรตัวแรก

เช่น name1 name2

Computer Programming 119

ข้อกำหนดการตั้งชื่อตัวแปรในภาษาซี

ตัวอักษรภาษาอังกฤษตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ถือเป็นคนละตัวกัน

เช่น name กับ NAME ถือเป็นคนละตัวแปร

Computer Programming 120

ตัวอย่างการตั้งชื่อตัวแปรในภาษาซี

ชื่อตัวแปร	อธิบาย	สามารถใช้ได้หรือไม่
age	ชื่อทั่วไป ใช้ตัวอักษรเล็กทั้งหมด	✅ ใช้ได้
Age	ตัวอักษรแรกเป็นตัวใหญ่	✅ ใช้ได้
age_2	มีตัวเลขและขีดล่างในชื่อ	✅ ใช้ได้
_age	เริ่มต้นด้วยขีดล่าง	✅ ใช้ได้
2age	เริ่มต้นด้วยตัวเลข	❌ ใช้ไม่ได้
total-price	มีเครื่องหมายขีดกลาง (ไม่อนุญาตในภาษา C)	❌ ใช้ไม่ได้
float	ชื่อชนิดข้อมูลที่สงวนไว้ (keyword)	❌ ใช้ไม่ได้
studentScore	การตั้งชื่อแบบ Camel Case	✅ ใช้ได้
student_score	การตั้งชื่อแบบ Snake Case	✅ ใช้ได้
int	ชื่อชนิดข้อมูลที่สงวนไว้ (keyword)	❌ ใช้ไม่ได้
sum123	มีตัวอักษรและตัวเลขผสมกัน	✅ ใช้ได้
void	ชื่อที่สงวนไว้สำหรับคำสั่งของภาษา C (keyword)	❌ ใช้ไม่ได้

การกำหนดค่าด้วย =

เครื่องหมาย = จะเป็นการกำหนดค่าตัวแปรด้านซ้ายให้มีค่าเท่ากับด้านขวา

เช่น value = 10; name ="John Wick"; pi = 3.14159;

หากเป็นนิพจน์ หรือ สมการ ต้องหาค่าคงที่มาก่อนจึงจะทำการกำหนดค่าตัวแปรที่อยู่ด้านซ้ายได้

เช่น y = ax + b; cArea = PI * R * R;

ไม่สามารถนำค่าคงที่มากำหนดด้วยตัวแปรได้ เพราะไม่เป็นความจริง

เช่น 100 = a + 3; 1 = 2; 50 = a + 24 * 2;

#define

#define ในภาษา C เป็นคำสั่งที่ใช้สำหรับสร้าง แมโคร (macro) ซึ่ง**เป็นการกำหนดค่าคงที่** หรือแทนที่ข้อความบางส่วนในโปรแกรม โดยจะทำการแทนค่าก่อนที่โปรแกรมจะถูกคอมไพล์ ตัวอย่างเช่น การกำหนดค่าของ pi หรือ อื่นๆที่คงที่ไม่เปลี่ยนแปลง

#define pi 3.14159

ในตัวอย่างนี้
#define pi 3.14159 เป็นการสั่งให้คอมไพเลอร์แทนที่ทุกครั้งที่เห็นคำว่า pi ในโปรแกรมด้วยค่าคงที่ 3.14159

โดยคำสั่งนี้ไม่ใช้หน่วยความจำเพิ่มเติม เนื่องจากเป็นการแทนที่ข้อความในช่วงคอมไพล์ ไม่ใช่ตัวแปร

```c
#include <stdio.h>

#define pi 3.14

int main() {
    float radius = 5;
    float area = pi * radius * radius;  // pi ถูกแทนด้วย 3.14
    printf("The area of the circle is: %.2f\n", area);
    return 0;
}
```


ตัวอย่างการใช้งานตัวแปร

```c
#include <stdio.h>   // เรียกใช้ไลบรารีมาตรฐานสำหรับฟังก์ชัน I/O เช่น printf
#define VALUE 10     // กำหนดค่าคงที่ VALUE เป็น 10

int main() {
    int a = VALUE;     // ประกาศตัวแปร a และกำหนดค่าเป็น 10 (ตามที่ define ไว้)
    float b;           // ประกาศตัวแปร b เป็นชนิด float เพื่อเก็บค่าทศนิยม
    b = a * VALUE;     // กำหนดให้ b มีค่าเป็น a คูณด้วย VALUE ซึ่งเท่ากับ 10 * 10

    printf("%.2f\n", b);    // แสดงผลลัพธ์ของ b เป็นเลขทศนิยม 2 ตำแหน่ง

    return 0;          // คืนค่า 0 เพื่อบอกว่าการทำงานของโปรแกรมสิ้นสุดอย่างถูกต้อง
}
```

ตัวดำเนินการในภาษาซี

ตัวดำเนินการในภาษาซี แบ่งออกเป็น 5 ประเภท ได้แก่

1 ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators) ใช้สำหรับการคำนวณทางคณิตศาสตร์

+ (บวก) - (ลบ) * (คูณ) / (หาร) % (หารเอาเศษ) ++ (เพิ่มค่าขึ้น1) -- (ลดค่าลง1)

2 ตัวดำเนินการเปรียบเทียบ (Relational Operators) ใช้เปรียบเทียบค่าระหว่างตัวแปรหรือค่าคงที่

== (เท่ากับ) != (ไม่เท่ากับ) > (มากกว่า) < (น้อยกว่า)

>= (มากกว่าหรือเท่ากับ) <= (น้อยกว่าหรือเท่ากับ)

ตัวดำเนินการในภาษาซี

3 ตัวดำเนินการทางตรรกศาสตร์ (Logical Operators) ใช้ในการตรวจสอบเงื่อนไขหลายเงื่อนไข

&& (และ) || (หรือ) ! (ไม่)

4 ตัวดำเนินการทางบิต (Bitwise Operators) ใช้สำหรับการดำเนินการกับข้อมูลในระดับบิต

& (AND) | (OR) ^ (XOR) ~ (NOT) << (เลื่อนบิตซ้าย) >> (เลื่อนบิตขวา)

5 ตัวดำเนินประกอบ (Compound Operators) ใช้สำหรับกำหนดค่าหรือตัวแปร

+=, -=, *=, /=, %= (กำหนดค่าและดำเนินการทางคณิตศาสตร์พร้อมกัน)

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators) ในภาษาซีใช้สำหรับการคำนวณพื้นฐาน มีทั้งหมด 5 ตัวหลักๆ ดังนี้

1. **+ (บวก)** ใช้สำหรับการบวกค่าของตัวเลขสองตัว
   - ตัวอย่าง a + b
2. **- (ลบ)** ใช้สำหรับการลบค่าของตัวเลขสองตัว
   - ตัวอย่าง a - b
3. *** (คูณ)** ใช้สำหรับการคูณค่าของตัวเลขสองตัว
   - ตัวอย่าง a * b
4. **/ (หาร)** ใช้สำหรับการหารตัวเลขสองตัว โดยให้ผลลัพธ์เป็นการหารปกติ
   - ตัวอย่าง a / b (ถ้า a และ b เป็นจำนวนเต็ม ผลลัพธ์จะเป็นจำนวนเต็ม)
5. **% (หารเอาเศษ)** ใช้สำหรับการหารตัวเลขสองตัวและคืนค่าเฉพาะเศษของการหาร (เฉพาะการหารจำนวนเต็ม)
   - ตัวอย่าง a % b

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

การใช้ตัวดำเนินการทางคณิตศาสตร์ในภาษาซี

ผลลัพธ์ของการคำนวณจะขึ้นอยู่กับชนิดข้อมูลที่ใหญ่ที่สุดของตัวแปรที่ใช้

ตัวอย่าง

int x = 6; float y = 2.0;

x * 5 ผลลัพธ์เป็น int เพราะใช้ตัวแปร int

x / 5 ผลลัพธ์เป็น int ทศนิยมจะหายไป

x * y ผลลัพธ์เป็น float เพราะมีตัวแปร float มาร่วมคำนวณ

ตัวอย่างการดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

จากคำสั่งต่อไปนี้

```
int x,y,z;
x = 2;
y = 3;
z = x + y;
```

Z = 5

จงหาค่าของตัวแปร z ?

ตัวอย่างการดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

```
int x, y;
x = 3;
x = 7;
y = x * 2;
```

จงหาค่าของตัวแปร y

y = 14

```
int x;
x = 4;
x = x + 5;
```

จงหาค่าของตัวแปร x

x = 9

การหารเอาเศษ (Modulus)

วิธีหารเอาเศษ (MOD)

ตัวอย่าง 10 % 3

1 นำตัวเลขจำนวนเต็มสองตัวมาหารกัน เช่น 10 / 3 โดยใช้เครื่องหมาย % เพื่อหาค่าเศษ

2 ผลลัพธ์ที่ได้คือเศษจากการหาร (remainder) เช่น 10 % 3 จะได้ 1 (เพราะ 3 ลงตัว 3 ครั้ง เหลือเศษ 1)

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

เครื่องหมาย	ความหมาย
+	บวก
-	ลบ
*	คูณ
/	หาร
%	หารเอาเศษ (Modulo)
++	เพิ่มค่าครั้งละ 1
--	ลดค่าครั้งละ 1

ลำดับความสำคัญของตัวดำเนินการหลัก

ก่อน

ตัวดำเนินการ	ลำดับความสำคัญ	ทิศทางการทำงาน
()	1	ซ้ายไปขวา (Left to Right)
* / %	2	ซ้ายไปขวา (Left to Right)
+ -	3	ซ้ายไปขวา (Left to Right)

หลัง

การกำหนดค่าและลำดับของการดำเนินการ

ลำดับของตัวดำเนินการ จะเกิดขึ้นตามระดับของตัวดำเนินการ

การคูณ และหาร จะกระทำก่อนบวกและลบ ถ้าระดับเท่ากัน จึงคำนวนจากซ้ายไปขวา

จงลำดับความสำคัญของตัวดำเนินการดังนี้

```
result = total - num1 * num2 / divisor + addValue;
```

การกำหนดค่าและลำดับของการดำเนินการ

result = total - num1 * num2 / divisor + addValue;

ลำดับความสำคัญของตัวดำเนินการ

1. **คูณ** * จะถูกดำเนินการก่อน เนื่องจากมีลำดับความสำคัญสูงกว่าการบวกและลบ
   - num1 * num2
2. **หาร** / จะถูกดำเนินการหลังจากการคูณ เพราะทำงานจากซ้ายไปขวา
   - (num1 * num2) / divisor
3. **ลบ - และ บวก +** หลังจากที่คูณและหารเสร็จแล้ว จะทำการลบและบวก เพราะจะทำจากซ้ายไปขวา
   - total - (ผลลัพธ์จากการคูณและหาร) + addValue

การกำหนดค่าและลำดับของการดำเนินการ

ตัวอย่างการใช้วงเล็บเพื่อเปลี่ยนลำดับ: หากต้องการให้การบวกหรือลบมีลำดับความสำคัญก่อนคูณและหาร เราสามารถใช้วงเล็บได้

result = (total - num1) * (num2 / divisor) + addValue;

ในตัวอย่างนี้

- วงเล็บจะบังคับให้ total - num1 ทำก่อน
- และ num2 / divisor ทำก่อนการคูณ

เพื่อป้องกันความสับสน ควรใช้ ( ) เพื่อระบุว่าตัวดำเนินการใดต้องทำก่อน

ฝึกหาค่านิพจน์

นิพจน์	คำตอบ	คำอธิบาย
10 / 5	2	หารจำนวนเต็มปกติ: 10 หาร 5 ได้ 2
3 / 9	0	หารจำนวนเต็ม: 3 หาร 9 ได้ 0 เพราะในภาษา C การหารจำนวนเต็มจะทิ้งค่าทศนิยม
7 / 2	3	หารจำนวนเต็ม: 7 หาร 2 ได้ 3 (ทิ้งค่าทศนิยมที่เหลืออยู่)
7.0 / 2	3.5	หารทศนิยม: 7.0 เป็น float การหารนี้ให้ผลลัพธ์เป็นทศนิยม 3.5
21 % 4	1	Modulus (หารเอาเศษ): 21 หาร 4 เหลือเศษ 1
10 % 7	3	Modulus (หารเอาเศษ): 10 หาร 7 เหลือเศษ 3
10 % 2	0	Modulus (หารเอาเศษ): 10 หาร 2 เหลือเศษ 0 (ลงตัวพอดี)
3 % 7	3	Modulus (หารเอาเศษ): 3 หาร 7 เหลือเศษ 3 (เพราะ 3 น้อยกว่า 7 ไม่สามารถหารได้ จึงเหลือเศษเท่ากับ 3)

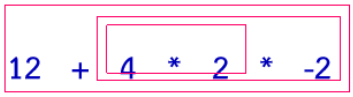
ฝึกหาค่านิพจน์

```
12 + 4 * 2 * -2
```

```
3 * 5 % 7 + -7
```

```
( 2 + 7 / 2 ) * ( 8 - 3 - 7 % 5 )
```

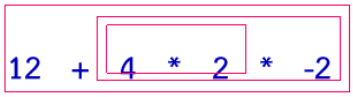
ฝึกหาค่านิพจน์

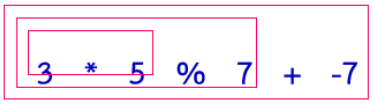


3 * 5 % 7 + -7

(2 + 7 / 2) * (8 - 3 - 7 % 5)

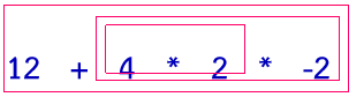
ฝึกหาค่านิพจน์

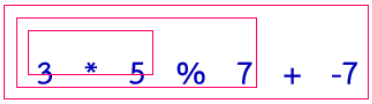


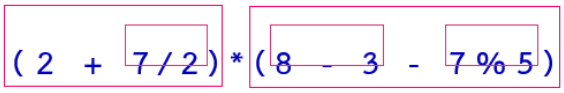


(2 + 7 / 2) * (8 - 3 - 7 % 5)

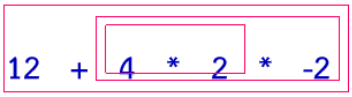
ฝึกหาค่านิพจน์



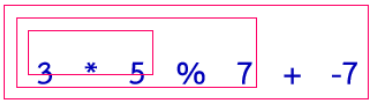




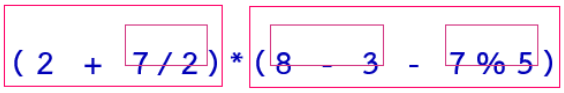
ฝึกหาค่านิพจน์



-4



-6



15

ฝึกหาค่านิพจน์

```
int a = 5;
int b = 2;
int c = 4;
a = a / b;
b = c;
b = c % 6;
c = c / 2;
```

หลังรันทุกบรรทัดเสร็จ

a = ?

b = ?

c = ?

Computer Programming 143

ฝึกหาค่านิพจน์

```
int a = 5;
int b = 2;
int c = 4;
a = a / b;
b = c;
b = c % 6;
c = c / 2;
```

main.c

```c
#include <stdio.h>

int main()
{
    int a = 5;
    int b = 2;
    int c = 4;
    a = a / b;
    b = c;
    b = c % 6;
    c = c / 2;
    printf("a={%d} b={%d} c={%d}",a,b,c);
    return 0;
}
```

```
a={2} b={4} c={2}

...Program finished with exit code 0
Press ENTER to exit console.
```

Computer Programming 144

ข้อควรระวัง เครื่องหมาย = และ == ไม่เหมือนกัน

ในภาษาซี (C) เครื่องหมาย = และ == มีความหมายและการใช้งานที่แตกต่างกันอย่างชัดเจน

1. **เครื่องหมาย = (Assignment Operator)**
   - ใช้สำหรับการกำหนดค่าให้กับตัวแปร (assignment)
   - ความหมายคือ "นำค่าทางขวาของเครื่องหมาย = ไปใส่ในตัวแปรทางซ้าย"
   - เช่น `int i;  i=0;`
2. **เครื่องหมาย == (Equality Operator)**
   - ใช้สำหรับการเปรียบเทียบค่า (comparison)
   - ใช้เพื่อตรวจสอบว่าค่าของตัวแปรทางซ้ายและขวามีค่าเท่ากันหรือไม่
   - ผลลัพธ์จะเป็นค่าจริง (True) หรือเท็จ (False) โดยปกติจะใช้ในเงื่อนไข if หรือ while
   - เช่น
   - `a = 5, b=6;`
   - `if(a==b) { printf("a and b are equal!"); }`

สรุป = ใช้สำหรับการกำหนดค่าให้กับตัวแปร ส่วน == ใช้สำหรับการเปรียบเทียบค่าของตัวแปรสองตัวว่ามีค่าเท่ากันหรือไม่

ตัวดำเนินการเพิ่มค่าและตัวดำเนินการลดค่า (Increment & Decrement)

++	คือ	บวกเพิ่ม1	**เช่น** a++ หมายถึง เพิ่มค่า a ขึ้นทีละ 1 (a = a + 1)
--	คือ	ลบออก1	**เช่น** b-- หมายถึง ลดค่า b ลงทีละ 1 (b = b - 1)

วางหน้าตัวแปร (Prefix: ++a หรือ --a) จะทำการเพิ่มหรือลดค่าของตัวแปรก่อน แล้วจึงส่งคืนค่าของตัวแปรที่ถูกเปลี่ยนแปลงแล้ว

เช่น `int a = 5; int b = ++a;` // a จะถูกเพิ่มค่าขึ้นก่อน ทำให้ a = 6 และ b = 6

วางหลังตัวแปร (Postfix: a++ หรือ a--) ส่งคืนค่าของตัวแปรก่อน แล้วจึงทำการเพิ่มหรือลดค่าของตัวแปรนั้น

เช่น `int a = 5; int b = a++;` // b จะได้ค่าของ a ก่อนทำให้ b = 5 จากนั้น a จึงถูกเพิ่มขึ้นเป็น a = 6

ตัวอย่างการใช้ Prefix และ Postfix ในโปรแกรม

```
#include <stdio.h>
int main() {
    int a = 5, b = 5;
    int prefix, postfix;
    prefix = ++a;    // เพิ่มค่า a ก่อน ทำให้ a = 6 และ prefix = 6
    postfix = b++;   // ส่งค่าของ b ก่อน ทำให้ postfix = 5 และ b = 6
    printf("Prefix: a = %d, prefix = %d\n", a, prefix);       // Output: a = 6, prefix = 6
    printf("Postfix: b = %d, postfix = %d\n", b, postfix);    // Output: b = 6, postfix = 5
}
```


การแปลงชนิดข้อมูล (Type Conversion)

1.การแปลงชนิดข้อมูลโดยอัตโนมัติ (Implicit Type Conversion)

1.1 การแปลงผ่านเครื่องหมาย =

1.2 การแปลงเมื่อเกิดการดำเนินการ

2. การแปลงผ่านการบังคับ (Explicit Type Conversion / Type Casting)

การแปลงชนิดข้อมูล (Type Conversion)

1.1 การแปลงผ่านเครื่องหมาย =

ตัวอย่างที่พบบ่อยได้แก่ การแปลงชนิดข้อมูลจากชนิดที่มีขนาดเล็กกว่าไปยังชนิดที่มีขนาดใหญ่กว่า เช่น จาก int เป็น float หรือ double ซึ่งจะไม่ทำให้เกิดการสูญเสียข้อมูล แต่หากแปลงจากชนิดข้อมูลขนาดใหญ่ไปชนิดข้อมูลขนาดเล็ก เช่น จาก double เป็น int อาจทำให้เกิดการสูญเสียข้อมูลได้

ตัวอย่าง การแปลงจาก int เป็น float

```
int a = 5;
float b;

b = a;  // a (int) จะถูกแปลงเป็น float โดยอัตโนมัติ
printf("b = %f\n", b);  // Output: b = 5.000000
```

ตัวอย่าง การแปลงจาก float เป็น int

```
float x = 10.75;
int y;

y = x;  // x (float) จะถูกแปลงเป็น int โดยตัดส่วนทศนิยมออก
printf("y = %d\n", y);  // Output: y = 10
```


การแปลงชนิดข้อมูล (Type Conversion)

1.2 การแปลงเมื่อเกิดการดำเนินการ

ตัวอย่างของการแปลงชนิดข้อมูลในขณะเกิดการดำเนินการได้แก่ การบวก การลบ การคูณ การหาร หรือการเปรียบเทียบระหว่างตัวแปรที่มีชนิดต่างกัน เช่น การดำเนินการระหว่างตัวแปรชนิด int กับ float หรือ char กับ double เป็นต้น

ตัวอย่าง

```
#include <stdio.h>

int main() {
    int a = 5;
    float b = 7.8;
    float result;

    result = a + b;  // a (int) จะถูกแปลงเป็น float ก่อน แล้วค่อยบวกกับ b
    printf("Result of a + b = %f\n", result);  // Output: Result of a + b = 12.800000
}
```

การแปลงชนิดข้อมูล (Type Conversion)

2. การแปลงผ่านการบังคับ (Explicit Type Conversion / **Type Casting**)

การแปลงข้อมูลแบบนี้จะต้องใส่ชนิดข้อมูลไว้ในวงเล็บหน้าค่าที่ต้องการแปลง เช่น (float), (int)

ตัวอย่าง

```
int a = 5, b = 2;
float result;

result = (float)a / b;  // แปลง a จาก int เป็น float ทำให้เกิดการหารแบบทศนิยม
printf("Result = %f\n", result);  // Output: Result = 2.500000
```


ตัวดำเนินประกอบ (Compound Operators)

จะใช้ในการคำนวณและการกำหนดค่าในคำสั่งเดียวกัน **ทำให้การเขียนโปรแกรมสั้นลงและอ่านง่ายขึ้น** โดยตัวดำเนินประกอบจะรวมการดำเนินการทางคณิตศาสตร์ เช่น บวก ลบ คูณ หาร หรือการดำเนินการทางบิต เข้ากับการกำหนดค่า (=) ในคำสั่งเดียว โดยตัวดำเนินประกอบเหล่านี้จะทำให้การกำหนดค่าและการคำนวณกระชับมากขึ้น แทนที่จะเขียนเป็นคำสั่งแยก

ตัวอย่าง

x = x + 1;	สามารถแทนได้ด้วย	x += 1;
a = a - b;	สามารถแทนได้ด้วย	a -= b;
x = x * 2;	สามารถแทนได้ด้วย	x *= 2;
y = y / 5;	สามารถแทนได้ด้วย	y /= 5;

ตัวดำเนินประกอบ (Compound Operators)

จงหาค่า ของ a และ b หลังรัน Code ต่อไปนี้

```
int a = 2, b = 4;
a += 2;
a *= b+1;
```

a=? b=?

ขอบเขตของตัวแปร

ขอบเขตของตัวแปร

ตัวแปรสามารถถูกประกาศได้ทั้งภายนอก (Global Variable) และภายในฟังก์ชัน (Local Variable)

ตัวแปรจะถูกสร้างขึ้นเมื่อมีการประกาศตัวแปร

- หากเป็นตัวแปร Local Variable จะถูกทำลายเมื่อฟังก์ชันจบการทำงาน
- หากเป็นตัวแปร Global Variable จะติดอยู่ตลอดที่โปรแกรมยังทำงาน (ยกเว้นมีการสั่งทำลาย)

*** ไม่ควรประกาศตัวแปรเป็นแบบ Global ทั้งหมด เพราะจะเกิดการผิดพลาดในการอัพเดทค่าได้ และกินหน่วยความจำค่อนข้างมาก

ขอบเขตของตัวแปร

```
// ตัวแปรภายใน (Local Variable) ที่ไม่เกี่ยวข้องกัน
#include <stdio.h>
void my_func();
int main()
{
    double x = 1.1;
    my_func();
    printf("In main x = %f\n",x);
    return 0;
}
void my_func()
{
    double x;
    x = 2.5;
    printf("In my_func x = %f\n",x);
}
```

ขอบเขตของตัวแปร

```
// การใช้ตัวแปรภายนอก
#include <iostream.h>
double x;
void my_func();
int main()
{
  x = 1.1;
  my_func();
  printf("In main x = %f\n",x);
  return 0;
}
void my_func()
{
  x = 2.5;
  printf("In my_func x = %f\n",x);
}
```


ขอบเขตของตัวแปร

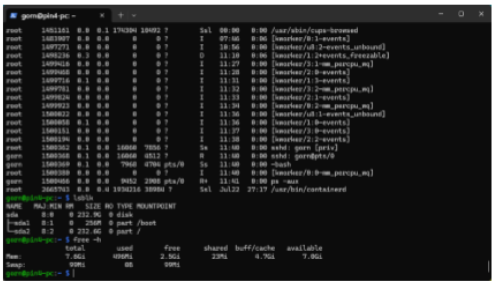
```
// หากมีตัวแปรภายนอกชื่อเดียวกับภายใน จะเรียกตัวแปรภายในก่อน
#include <stdio.h>
double x;
void my_func();
int main()
{
        double x = 1.1;
        my_func();
        printf("In main x = %f\n",x);
        return 0;
}
void my_func()
{
        x = 2.5;
        printf("In my_func x = %f\n",x);
}
```

การแสดงผลและรับค่า (ทางหน้าจอ Command line)

การแสดงผลข้อความ (ทางหน้าจอ Command line)

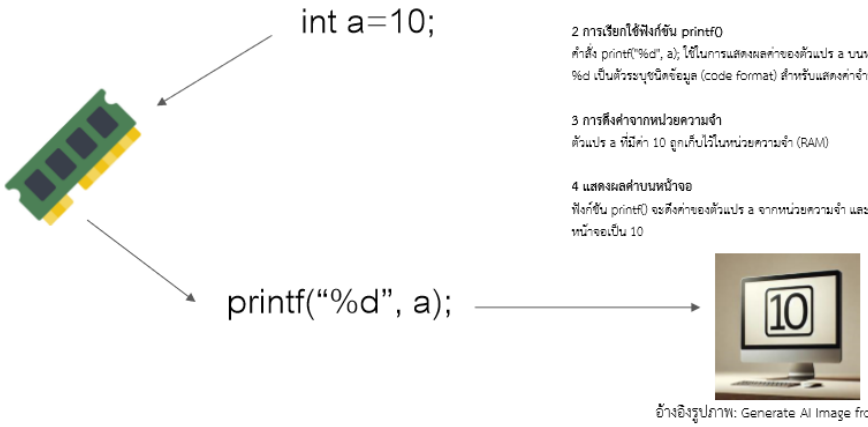
การแสดงผลทางหน้าจอในภาษาซี (C) มักจะใช้ฟังก์ชันมาตรฐานจากไลบรารี stdio.h

ฟังก์ชันหลักที่ใช้คือ printf() ฟังก์ชันนี้สามารถแสดงผลข้อมูลประเภทต่างๆ เช่น ข้อความ ตัวเลข ทศนิยม หรืออักขระ โดยมีรูปแบบการใช้งานที่หลากหลาย



การแสดงผลข้อความ (ทางหน้าจอ Command line)

หลักการทำงานของ printf



1 การกำหนดค่าให้ตัวแปร
คำสั่ง int a = 10; จะเก็บค่าตัวเลข 10 ลงในตัวแปร a ซึ่งเป็นตัวแปรชนิด int (จำนวนเต็ม)

2 การเรียกใช้ฟังก์ชัน printf()
คำสั่ง printf("%d", a); ใช้ในการแสดงผลค่าของตัวแปร a บนหน้าจอ
%d เป็นตัวระบุชนิดข้อมูล (code format) สำหรับแสดงค่าจำนวนเต็ม

3 การดึงค่าจากหน่วยความจำ
ตัวแปร a ที่มีค่า 10 ถูกเก็บไว้ในหน่วยความจำ (RAM)

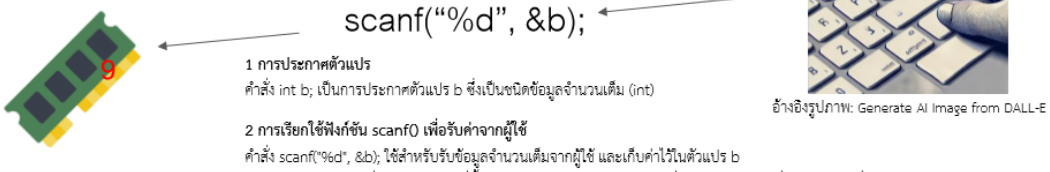
4 แสดงผลค่าบนหน้าจอ
ฟังก์ชัน printf() จะดึงค่าของตัวแปร a จากหน่วยความจำ และแสดงผลค่าดังกล่าวบนหน้าจอเป็น 10

อ้างอิงรูปภาพ: Generate AI Image from DALL-E

Beautiful Icon by Flaticon License (smashicons https://smashicons.com/)

การรับค่าด้วยscanf (ทางหน้าจอ Command line)

หลักการทำงานของ scanf



อ้างอิงรูปภาพ: Generate AI Image from DALL-E

1 การประกาศตัวแปร
คำสั่ง int b; เป็นการประกาศตัวแปร b ซึ่งเป็นชนิดข้อมูลจำนวนเต็ม (int)

2 การเรียกใช้ฟังก์ชัน scanf() เพื่อรับค่าจากผู้ใช้
คำสั่ง scanf("%d", &b); ใช้สำหรับรับข้อมูลจำนวนเต็มจากผู้ใช้ และเก็บค่าไว้ในตัวแปร b
%d คือรูปแบบข้อมูลที่ต้องการรับ (ในที่นี้คือจำนวนเต็ม int) **&b** คือที่อยู่ของตัวแปร b ที่จะเก็บข้อมูลที่ได้รับเข้ามา

3 การรับค่าจากแป้นพิมพ์
เมื่อผู้ใช้ป้อนค่า เช่น 9 ผ่านแป้นพิมพ์ ค่านั้นจะถูกส่งไปยังฟังก์ชัน scanf()

4 การเก็บค่าลงหน่วยความจำ
ค่าที่ผู้ใช้ป้อน (9) จะถูกเก็บลงในตัวแปร b ที่อยู่ในหน่วยความจำ (RAM)

5 ผลลัพธ์การทำงาน
หลังจาก scanf() ทำงานเสร็จ ตัวแปร b จะมีค่าเป็น 9 ซึ่งสามารถนำไปใช้ในโปรแกรมต่อไปได้

Beautiful Icon by Flaticon License (smashicons https://smashicons.com/)

printf()

printf = การแสดงผลข้อความ (ทางหน้าจอ Command line)

ฟังก์ชัน printf() ใช้สำหรับแสดงผลข้อมูลบนหน้าจอ โดยมีรูปแบบดังนี้

printf("**ข้อความที่ต้องการแสดง %ตัวระบุข้อมูล**", **ตัวแปรที่ต้องการแสดง**);

ส่วน **%ตัวระบุข้อมูล** คือ code format หรือ format specifier ที่ใช้บอกชนิดของข้อมูลที่จะถูกแสดง เช่น %d, %f, %s เป็นต้น

Code Format หรือ Format Specifier

%d : ใช้แสดงผลจำนวนเต็ม (integer) เช่น 5, -3

%f : ใช้แสดงผลตัวเลขทศนิยม (float) เช่น 3.14, 5.0

%lf : ใช้แสดงผลตัวเลขทศนิยมแบบ double

%c : ใช้แสดงผลตัวอักษร (character) เช่น 'A', 'b'

%s : ใช้แสดงผลข้อความ (string) เช่น "Hello", "World"

%u : ใช้แสดงผลจำนวนเต็มบวก (unsigned integer)

%x หรือ %X : ใช้แสดงผลเลขฐานสิบหก (hexadecimal)

%o : ใช้แสดงผลเลขฐานแปด (octal)

%p : ใช้แสดงที่อยู่หน่วยความจำ (pointer address)

%% : ใช้แสดงเครื่องหมาย % บนหน้าจอ

รหัสควบคุม (Escape Sequence)

\n	ใช้สำหรับขึ้นบรรทัดใหม่ (new line)
\t	ใช้สำหรับเว้นวรรคแบบแท็บ (tab space)
\'	Single Quote '
\"	Double Quote "
\a	เสียง beep

ตัวอย่าง printf

การใช้ \n เป็น escape sequence ที่บอกให้โปรแกรม tab ไป1ช่วง เช่น

printf("Hello\tMy name is John Wick!");



ตัวอย่าง printf

การใช้ \t เป็น escape sequence ที่บอกให้โปรแกรมขึ้นบรรทัดใหม่ เช่น

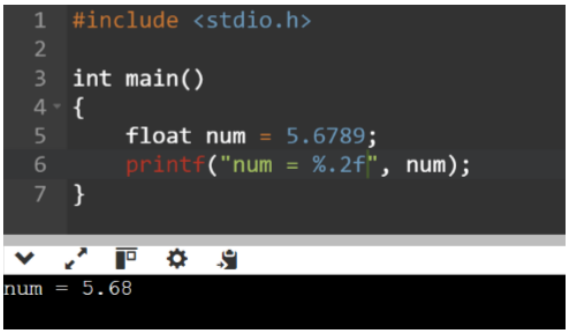
printf("Hello\nMy name is John Wick!");



ตัวอย่าง printf

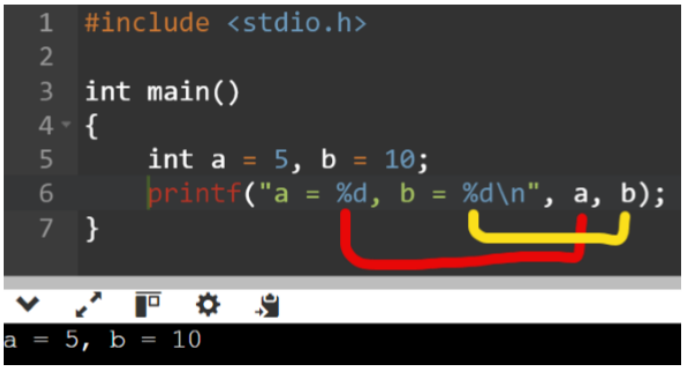
สามารถกำหนดจำนวนหลักของตัวเลขทศนิยมได้

เช่น %.2f จะกำหนดให้แสดงผลตัวเลขทศนิยม 2 ตำแหน่ง



ตัวอย่าง printf

แสดงผลตัวแปรหลายตัวได้ในคำสั่งเดียวโดยการใช้ตัวระบุข้อมูลหลาย ๆ ตัว



ตัวอย่าง printf

printf from int

int count=17848;

printf("%d",count);

```c
#include <stdio.h>

int main()
{
    int count=17848;
    printf("%d",count);
```

```
17848
```


ตัวอย่าง printf

printf from float

float pi=3.141592;

printf("%.4f",pi);

```c
#include <stdio.h>

int main()
{
    float pi=3.141592;
    printf("%.4f",pi);
}
```

```
3.1416
```

ตัวอย่าง printf

printf from char

```
char grade='A';
printf("%c",grade);
```

```
#include <stdio.h>

int main()
{
    char grade='A';
    printf("%c",grade);
}
```

```
A
```


ตัวอย่าง printf

printf from char [] (string)

```
char name[100] = "JohnWick";
printf("my name is %s",name);
```

```
#include <stdio.h>

int main()
{
    char name[100] = "JohnWick";
    printf("my name is %s",name);
}
```

```
input
my name is JohnWick
```

scanf()

scanf = รับค่าจากคีย์บอร์ด (ทางหน้าจอ Command line)

ฟังก์ชัน scanf() ใช้สำหรับรับค่าการพิมพ์จากคีย์บอร์ด โดยมีรูปแบบดังนี้

scanf("%ตัวระบุข้อมูล", ที่อยู่ของตัวแปร(Address)ที่ต้องการแสดง);

scanf("%ตัวระบุข้อมูล", &ชื่อตัวแปร); // การใส่ & หน้าชื่อตัวแปรคือการบอกว่าให้ส่งค่า address ของตัวแปรมา

ส่วน %ตัวระบุข้อมูล คือ code format หรือ format specifier ที่ใช้บอกชนิดของข้อมูลที่จะถูกแสดง เช่น %d, %f, %s เป็นต้น

เช่น

```
int count;
scanf("%d",&count);
```

ตัวอย่าง scanf

```c
#include <stdio.h>

int main() {

    int num;          // ตัวแปรประเภท int

    // รับค่าตัวแปรประเภท int

    printf("Enter an integer number: ");

    scanf("%d", &num);  // ใช้ %d สำหรับ int

    printf("You entered: %d\n", num);

}
```



ตัวอย่าง scanf

```c
#include <stdio.h>

int main() {

    float decimal;    // ตัวแปรประเภท float

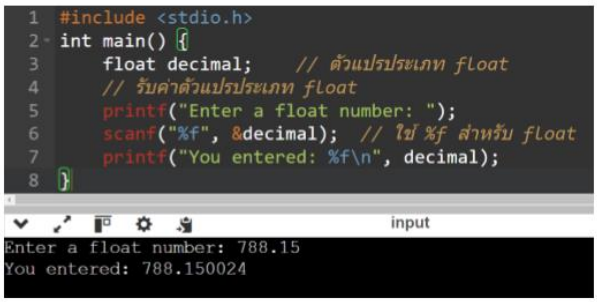
    // รับค่าตัวแปรประเภท float

    printf("Enter a float number: ");

    scanf("%f", &decimal);  // ใช้ %f สำหรับ float   << ห้ามใส่ %.2f ในส่วนของ %ตัวระบุข้อมูล

    printf("You entered: %f\n", decimal);

}
```



ตัวอย่าง scanf

```c
#include <stdio.h>

int main() {

    char character;   // ตัวแปรประเภท char (ตัวอักษร)

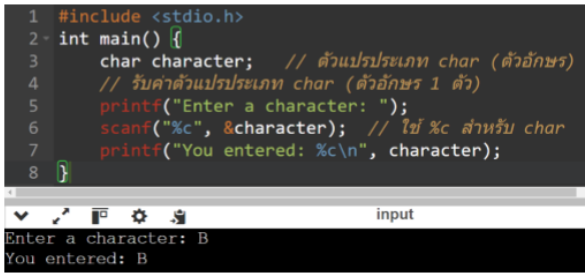
    // รับค่าตัวแปรประเภท char (ตัวอักษร 1 ตัว)

    printf("Enter a character: ");

    scanf("%c", &character);  // ใช้ %c สำหรับ char

    printf("You entered: %c\n", character);

}
```



ตัวอย่าง scanf

```c
#include <stdio.h>

int main() {

    char str[100];    // ตัวแปรประเภท char[] (สตริงหรือข้อความ)

    // รับค่าตัวแปรประเภท char[] (ข้อความหรือสตริง)

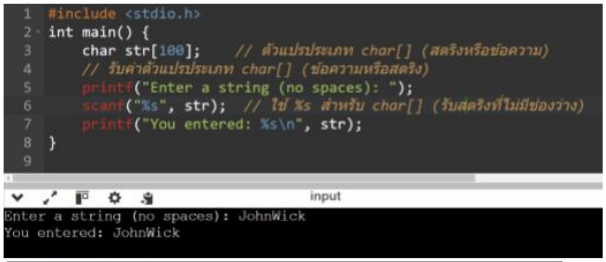
    printf("Enter a string (no spaces): ");

    scanf("%s", str);  // ใช้ %s สำหรับ char[] (รับสตริงที่ไม่มีช่องว่าง)

    printf("You entered: %s\n", str);

}
```

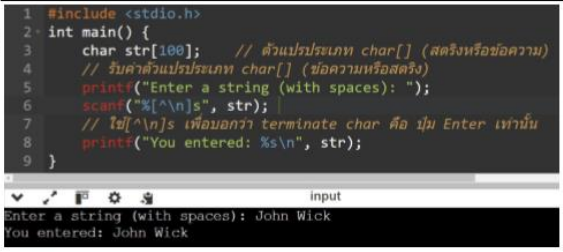
ตัวอย่าง scanf



```
#include <stdio.h>
int main() {
    char str[100];    // ตัวแปรประเภท char[] (สตริงหรือข้อความ)
    // รับค่าตัวแปรประเภท char[] (ข้อความหรือสตริง)
    printf("Enter a string (no spaces): ");
    scanf("%s", str);  // ใช้ %s สำหรับ char[] (รับสตริงที่ไม่มีช่องว่าง)
    printf("You entered: %s\n", str);
}
```

```
input
Enter a string (no spaces): JohnWick
You entered: JohnWick
```

%[^\n]s มีเว้นวรรคได้



```
#include <stdio.h>
int main() {
    char str[100];    // ตัวแปรประเภท char[] (สตริงหรือข้อความ)
    // รับค่าตัวแปรประเภท char[] (ข้อความหรือสตริง)
    printf("Enter a string (with spaces): ");
    scanf("%[^\n]s", str);
    // ใช้[^\n]s เพื่อบอกว่า terminate char คือ ปุ่ม Enter เท่านั้น
    printf("You entered: %s\n", str);
}
```

```
input
Enter a string (with spaces): John Wick
You entered: John Wick
```


ตัวอย่าง printf และ scanf

```
#include<stdio.h>

int main(){

    char name[100];

    int salary;

    printf("Please enter your name: ");

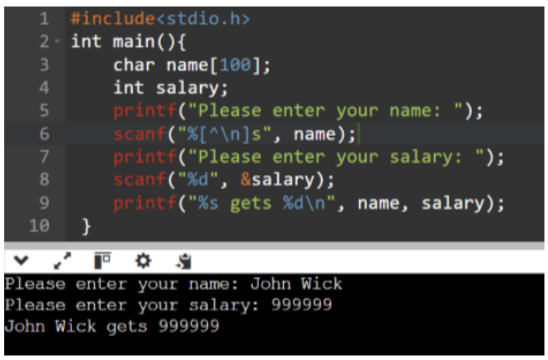
    scanf("%s", name);

    printf("Please enter your salary: ");

    scanf("%d", &salary);

    printf("%s gets %d\n", name, salary);

}
```



```
#include<stdio.h>
int main(){
    char name[100];
    int salary;
    printf("Please enter your name: ");
    scanf("%[^\n]s", name);
    printf("Please enter your salary: ");
    scanf("%d", &salary);
    printf("%s gets %d\n", name, salary);
}
```

```
Please enter your name: John Wick
Please enter your salary: 999999
John Wick gets 999999
```

ตัวอย่าง printf และ scanf

```
#include<stdio.h>
int main()
{
    int a = 14, b = 3;
    int i = 10, j = 2, k = 5, x = 4, y = 5;
    /* Difference between / and %% */
    printf("Difference between / and %\n");
    printf("a = %3d , b = %3d \n",a,b);
    printf("a/b = %3d \n",a/b);
    printf("a%%b = %3d \n\n",a%b);
}
```

```
Difference between / and %
a =  14 , b =   3
a/b =   4
a%b =   2
```


ตัวอย่าง printf และ scanf

```
#include<stdio.h>
int main()
{
    int a = 14, b = 3;
    int i = 10, j = 2, k = 5, x = 4, y = 5;
    /*Using casting*/
    printf("Using casting\n");
    printf("(float) a/b = %f\n\n",(float)a/b);
}
```

```
Using casting
(float) a/b = 4.666667
```

ตัวอย่าง printf และ scanf

```
#include<stdio.h>
int main()
{
    int a = 14, b = 3;
    int i = 10, j = 2, k = 5, x = 4, y = 5;
    /* Precedence of arithmetic operators */
    printf("Precedence of arithmetic operators\n");
    printf("i = 10, j = 2, k = 5, x = 4, y = 5\n");
    printf("i+j*k = %d\n", i+j*k);
    printf("(i+j)*k  = %d\n", (i+j)*k);
    printf("i+(j*k)  = %d\n", i+(j*k));
    printf("x*y+i/10*k-8 = %d\n", x*y+i/10*k-8);
}
```

```
Precedence of arithmetic operators
i = 10, j = 2, k = 5, x = 4, y = 5
i+j*k = 20
(i+j)*k  = 60
i+(j*k)  = 20
x*y+i/10*k-8 = 17
```


ตัวอย่าง printf และ scanf

```
#include <stdio.h>
int main ()
{
    int i,j;
    printf("Enter the value of i: ");
    scanf("%d",&i);
    printf("\nEnter the value of j: ");
    scanf("%d",&j);
    /* Logical Operator */
    printf("\nUsing Logical Operator\n");
    printf(" i&&j is %d\n",i&&j);
    printf(" i||j is %d\n",i||j);
    printf(" !i   is %d\n",!i);
    printf(" !j   is %d\n",!j);
}
```

```
Enter the value of i: 1

Enter the value of j: 0

Using Logical Operator
 i&&j is 0
 i||j is 1
 !i   is 0
 !j   is 1
```

ข้อผิดพลาดที่เกิดขึ้นบ่อย

ข้อผิดพลาดที่เกิดขึ้นบ่อย

ลืมปิดปีกกาของ main

```
#include<stdio.h>

int main(){
        return 0;
```

ข้อความแสดงข้อผิดพลาด

```
In function 'main':
    syntax error at end of input
```

การตีความ

รูปแบบมีข้อผิดพลาดตอนท้ายของโปรแกรม

ข้อผิดพลาดที่เกิดขึ้นบ่อย

พิมพ์คำว่า #include ผิด

```
#inlude<stdio.h>

int main(){

}
```

ข้อความแสดงข้อผิดพลาด

Line 1:

invalid preprocessing directive #inlude

การตีความ

ส่วนของ preprocessor ไม่ถูกต้อง ในส่วนของ #inlude

ข้อผิดพลาดที่เกิดขึ้นบ่อย

พิมพ์ stdio.h ผิด

```
#include<stio.h>

int main(){

}
```

ข้อความแสดงข้อผิดพลาด

Line 1:

stio.h: No such file or directory.

การตีความ

ไม่มีไฟล์ที่กำหนด ให้ตรวจสอบชื่อไฟล์ให้ถูกต้อง

ข้อผิดพลาดที่เกิดขึ้นบ่อย

พิมพ์ชื่อฟังก์ชัน main ผิด

```
#include<stdio.h>
int man(){
}
```

ไม่มีข้อผิดพลาดในการคอมไพล์

ผิดพลาดในขณะลิงค์

```
[Linker error] undefined reference to 'WinMain@16'
ld returned 1 exit status
```

ไม่สามารถหา main ฟังก์ชันได้

ข้อผิดพลาดที่เกิดขึ้นบ่อย

ลืมใส่ ;

```
#include<stdio.h>
int main(){
  int x
}
```

ข้อความแสดงข้อผิดพลาด

```
In function main:
    Line 4: syntax error before '}' token
```

การตีความ

ไปยังบรรทัดที่ 4 และมองหาคำสั่งในบรรทัดก่อนหน้าว่าตกหล่น ; หรือไม่

ข้อผิดพลาดที่เกิดขึ้นบ่อย

ลืมใส่ ;

```
#include<stdio.h>
int main(){
   int x
   x = 2;
}
```

ข้อความแสดงข้อผิดพลาด

In function main:

Line 4: syntax error before "x"

การตีความ

–ไปยังบรรทัดที่ 4 และมองหาคำสั่งในบรรทัดก่อนหน้าว่าตกหล่น ; หรือไม่

ข้อผิดพลาดที่เกิดขึ้นบ่อย

ลืมใส่ ; หลายๆที่

```
#include<stdio.h>
int main(){
   int x
   x = 2
}
```

ข้อความแสดงข้อผิดพลาด

```
In function main:
   Line 4: syntax error before "x"
```

Compiler จะบอกแค่อันเดียว ให้แก้ทีละจุด

ข้อผิดพลาดที่เกิดขึ้นบ่อย

ลืมประกาศตัวแปร

```
#include<stdio.h>
int main(){
    x = 2;
}
```

ข้อความแสดงข้อผิดพลาด

```
In function main:
    Line 3: 'x' undeclared (first use in this function)
```

การตีความ สังเกตจากหมายเลขบรรทัดจากนั้น ดูว่าตัวแปรใด ที่อยู่ภายใน '... ' ที่ถูกระบุว่า ไม่ได้ประกาศ (undeclared) ก่อนนำมาใช้งาน

ข้อผิดพลาดที่เกิดขึ้นบ่อย

ลืมปิดวงเล็บเวลาใช้ printf หรือ scanf

```
#include<stdio.h>
int main(){
  printf("Hi\n";
}
```

ข้อความแสดงข้อผิดพลาด

```
In function main:
    Line 3: syntax error before ';' token
```

การตีความ

มีข้อผิดพลาดเกิดขึ้นที่บรรทัดที่ 3 และมีข้อผิดพลาดก่อนsyntax error befor)จะส่วนของโปรแกรม(token) ;

ข้อผิดพลาดที่เกิดขึ้นบ่อย

ลืมเปิดวงเล็บเวลาใช้ printf หรือ scanf

```
#include<stdio.h>
int main(){
  printf"Hi\n");
}
```

•ข้อความแสดงข้อผิดพลาด

In function main:

Line 3: syntax error before string constant

•การตีความ

บรรทัดที่ 3 มีรูปแบบผิดพลาดก่อนจะเริ่มชุดข้อความ "..."

ข้อผิดพลาดที่เกิดขึ้นบ่อย

ลืมปิดเครื่องหมายคำพูด

```
#include<stdio.h>
int main(){
  printf("Hi\n);
}
```

ข้อความแสดงข้อผิดพลาด

In function main:

Line 3: missing terminating " character

Line 4: syntax error before '}' token

การตีความ

บรรทัดที่ 3 ขาดเครื่องหมายปิดข้อความ (") ทำให้เกิดข้อผิดพลาดอื่น ๆ ตามมาให้ทำการแก้ข้อผิดพลาดที่อยู่ตอนต้นสุดก่อน

ข้อผิดพลาดที่เกิดขึ้นบ่อย

ลืมเปิดเครื่องหมายคำพูด

```
#include<stdio.h>
int main(){
  printf(Hi\n");
}
```

ข้อความแสดงข้อผิดพลาด

```
In function main:
-Line 3: stray '\' in program
-Line 3: 'Hi' undeclared (first use in this function)
-Line 3: syntax error before 'n'
-Line 3: missing terminating " character
```

การตีความ

บรรทัดที่ 3 เนื่องจากตกเครื่องหมายคำพูดของข้อความ ทำให้คอมไพเลอร์ตีความ คำต่าง ๆ ในข้อความ และแสดงข้อผิดพลาดมากมาย เพราะคำเหล่านั้นไม่ตรงหลักการ

ข้อผิดพลาดที่เกิดขึ้นบ่อย

ลืมเปิดวงเล็บเครื่องหมายคำพูด

```
#include<stdio.h>
int main(){
  printfHi\n");
}
```

ข้อความแสดงข้อผิดพลาด

```
In function main:
Line 3: stray '\' in program
Line 3: `printfHi' undeclared (first use in this function)
Line 3: syntax error before "n"
Line 3: missing terminating " character
```

การตีความ

เช่นเดียวกับการลืมเครื่องหมายคำพูด แต่ยิ่งผิดพลาดมากกว่าเพราะทำให้คำสั่ง printf ผิดพลาดไปด้วย นั้นคือเป็นคำสั่งที่ไม่รู้จัก

ข้อผิดพลาดที่เกิดขึ้นบ่อย

เอาตัวแปรไปใส่ค่าคงที่

```
#include<stdio.h>
int main(){
  int x;
  2 = x;
}
```

ข้อความแสดงข้อผิดพลาด

In function main:

-Line 4: invalid lvalue in assignment

การตีความ

บรรทัดที่ 4 การให้ค่าไม่ถูกต้องเนื่องจาก ค่าทางซ้าย(lvalue-left value)ควรเป็นตัวแปร

ข้อผิดพลาดอันตรายที่ Compiler ไม่สามารถเตือนได้

```
#include <stdio.h>
int main() {
    int x;
    scanf("%d");  //very dangerous!!
    scanf("%d", x);          //forgot &
    printf("%d");
    printf("Hi", x);
}
```

เอกสารอ้างอิง

- คณาจารย์ผู้สอนวิชา Introduction to Computer Programming, เอกสารประกอบการบรรยายวิชา 241-101 Introduction to Computer Programming, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยสงขลานครินทร์, 2551
- อ.สุธน แซ่ว่อง, เอกสารประกอบการบรรยายอบรมภาษาซีเบื้องต้น, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยสงขลานครินทร์, 30 มีนาคม 2552
- Y. D. Liang, Supplement II.C:Dev-C++ Tutorial for Introduction to C++ Programming, http://www.bloodshed.net/dev/devcpp.html
- ดร. แสงสุรีย์ วสุพงศ์อัยยะ, เอกสารประกอบการบรรยายอบรมภาษาซีเบื้องต้น, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยสงขลานครินทร์, 30 มีนาคม 2552

แหล่งศึกษาเพิ่มเติม

ผศ.นราธร สังข์ประเสริฐ. (2563) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-3945-2

โอภาส เอี่ยมสิริวงศ์. (2559) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-2740-4

รศ.ธีรวัฒน์ ประกอบผล. (2562) คู่มือการเขียนโปรแกรมภาษา C ฉบับสมบูรณ์, ซิมพลิฟาย, ISBN: 978-616-26-24896

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase1] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=aDBmXMF9jOU

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase2] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=wslM0Aqepxs

Mooc การเขียนโปรแกรมคอมพิวเตอร์พื้นฐาน ด้วยภาษาซี 1 | Fundamentals of the C Programming Language https://lms.thaimooc.org/courses/course-v1:KMUTT+KMUTT013+2019/about

Mooc ทักษะการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี | Computer programming skills with C language https://lms.thaimooc.org/courses/course-v1:SU+SU011+2021/about

Coursera: Programming in C, ผู้สอน: Charles Russell Severance https://www.coursera.org/learn/programming-c

Coursera: ความเชี่ยวชาญพิเศษ การเขียนโปรแกรม C เบื้องต้น, ผู้สอน: Andrew D. Hilton et al. https://www.coursera.org/specializations/c-programming

Playlist Youtube บันทึกการสอนในห้องเรียน, วิชาการโปรแกรมคอมพิวเตอร์, ดร.พงศกร เจริญเนตรกุล 2566 https://www.youtube.com/watch?v=ewEfmiXaRi0&list=PL322g2itJn4mAE1utyOzF-C0SP0BRkej3&pp=gAQBiAQB

แบบฝึกหัดท้ายบท

1. ภาษา C ถูกพัฒนาโดยใคร
2. คำสั่งใดในภาษา C ใช้สำหรับแสดงผลข้อความ
3. คำสั่งใดใช้ในการนำเข้าไลบรารี stdio.h
4. ไฟล์ซอร์สโค้ดในภาษา C มีนามสกุลอะไร
5. คอมไพเลอร์ทำหน้าที่อะไรในกระบวนการเขียนโปรแกรม
6. อธิบายความแตกต่างระหว่างตัวดำเนินการ = และ == ในภาษา C
7. อะไรคือหน้าที่ของฟังก์ชัน main() ในภาษา C
8. อธิบายว่า printf("%d", a); ใช้แสดงผลข้อมูลอะไร
9. ทำไมภาษา C ถึงถูกใช้ในการเขียนโปรแกรมควบคุมฮาร์ดแวร์
10. ตัวแปรประเภท int ในภาษา C ใช้สำหรับเก็บข้อมูลชนิดใด
11. เขียนโปรแกรมที่แสดงผลข้อความ "Hello, World!"
12. เขียนโปรแกรมเพื่อรับค่าจากผู้ใช้เป็นจำนวนเต็มและแสดงผลกลับไปยังหน้าจอ
13. ในโปรแกรมต่อไปนี้ จะเกิดอะไรขึ้นถ้าคอมไพล์โดยไม่มีการใช้ #include <stdio.h>

```
int main() {
    printf("Hello");
    return 0;
}
```

14. กำหนดให้ตัวแปร x = 5 และ y = 10 จงเขียนโปรแกรมที่สลับค่าของตัวแปรทั้งสองโดยไม่ใช้ตัวแปรเสริม
15. เขียนโปรแกรมรับค่าอุณหภูมิเป็นองศาเซลเซียสจากผู้ใช้ แล้วแปลงเป็นองศาฟาเรนไฮต์
16. จงวิเคราะห์ข้อผิดพลาดจากโค้ดต่อไปนี้

```
#include<stdio.h>
int main() {
    int a = 10;
    printf("The value of a is: %f", a);
    return 0;
```

```
}
```

17. กำหนดให้ a = 5, b = 10 และ c = a + b * 2 จงหาผลลัพธ์ของตัวแปร c

18. จงหาผลลัพธ์ของโค้ดต่อไปนี้

```
#include<stdio.h>
int main() {
    int a = 5, b = 10;
    if(a == b) {
        printf("Equal");
    } else {
        printf("Not equal");
    }
    return 0;
}
```

19. ทำไมการใช้ scanf("%s", name); จึงอาจเกิดปัญหาถ้าไม่กำหนดขนาดของตัวแปร name ให้เหมาะสม

20. จงเปรียบเทียบคำสั่ง #define pi 3.14159 เมื่อเทียบกับการใช้ const float pi = 3.14159;

21. จงหาผลลัพธ์ของโค้ดต่อไปนี้

```
#include<stdio.h>
int main() {
    int x = 5;
    if(x = 0) {
        printf("X is zero");
    } else {
        printf("X is not zero");
    }
    return 0;
}
```

22. ทำไมการใช้ตัวแปร global ในโปรแกรมขนาดใหญ่จึงไม่เป็นแนวทางที่ดี

23. การใช้วงเล็บในนิพจน์คณิตศาสตร์มีความสำคัญอย่างไร

24. การเลือกใช้ฟังก์ชัน scanf() สำหรับการรับข้อมูล string มีข้อดีและข้อเสียอย่างไร

25. จงประเมินความปลอดภัยในการใช้คำสั่ง scanf("%d", &var); โดยไม่ตรวจสอบข้อผิดพลาดใดๆในการกรอกข้อมูลของผู้ใช้

26. จงเขียนโปรแกรมที่รับชื่อและอายุจากผู้ใช้แล้วแสดงผลข้อความ เช่น "Hello [name], you are [age] years old"

27. จงสร้างโปรแกรมที่รับค่าความกว้างและความยาวของสี่เหลี่ยมผืนผ้า แล้วคำนวณพื้นที่

28. จงเขียนโปรแกรมที่แปลงค่าระหว่างหน่วยเซนติเมตรและเมตร

29. จงสร้างโปรแกรมที่คำนวณผลรวมของตัวเลขตั้งแต่ 1 ถึง N ที่ผู้ใช้กรอก

30. จงสร้างโปรแกรมที่ตรวจสอบว่าค่าที่ผู้ใช้กรอกเป็นเลขคู่หรือเลขคี่

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน.....

ใบฝึกปฏิบัติการที่ 3.1

เรื่อง printf my information

วัตถุประสงค์

1. สามารถเขียนโครงสร้างของโปรแกรม คำสั่งแสดงข้อความทางหน้าจอได้
2. สามารถนำโปรแกรมไปรันเพื่อแสดงผลในหน้า Console ได้

ทบทวนความรู้

printf = การแสดงผลข้อความ (ทางหน้าจอ Command line)

ฟังก์ชัน printf() ใช้สำหรับแสดงผลข้อมูลบนหน้าจอ โดยมีรูปแบบดังนี้

printf("**ข้อความที่ต้องการแสดง %ตัวระบุข้อมูล**", **ตัวแปรที่ต้องการแสดง**);

ส่วน **%ตัวระบุข้อมูล** คือ code format หรือ format specifier ที่ใช้บอกชนิดของข้อมูลที่จะถูกแสดง เช่น %d, %f, %s เป็นต้น

Computer Programming 164

ตัวอย่าง printf

printf from char [] (string)

```
char name[100] = "JohnWick";

printf("my name is %s",name);
```

```
1  #include <stdio.h>
2
3  int main()
4  {
5      char name[100] = "JohnWick";
6      printf("my name is %s",name);
7  }
```

input

```
my name is JohnWick
```

Computer Programming 174

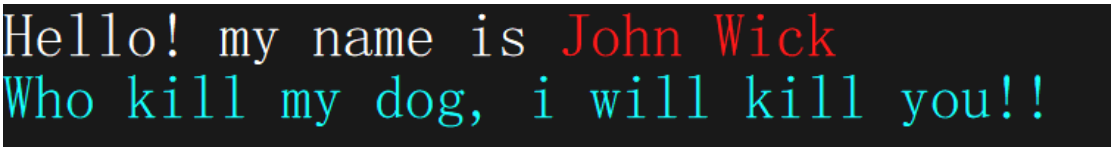
คำสั่ง

จงเขียนโปรแกรมแสดง ชื่อจริง ชื่อเล่น รหัสนศ. และข้อมูลอื่นๆ ของตัวเอง

ตัวอย่างการเลือกสีข้อความใน console

```
HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE); // must include windows.h
SetConsoleTextAttribute(hConsole, 15); printf("Hello! my name is ");
SetConsoleTextAttribute(hConsole, 12); printf("%s","John Wick");
SetConsoleTextAttribute(hConsole, 11); printf("\nWho kill my dog, i will kill you!!");
```

ตัวอย่างการทำงานของ code ด้านบน



ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน.....

ใบฝึกปฏิบัติการที่ 3.2

เรื่อง printf from variables

วัตถุประสงค์

1. สามารถประกาศตัวแปร และดึงค่าจากตัวแปรมาแสดงผลด้วย printf ได้
2. สามารถนำโปรแกรมไปรันเพื่อแสดงผลในหน้า Console ได้

ทบทวนความรู้

ตัวแปร (variable)

- ตัวแปรจะใช้หน่วยความจำของคอมพิวเตอร์ในการจดจำข้อมูล
- การเข้าถึงหน่วยความจำคอมพิวเตอร์มีความยากที่จะอ้างอิงตำแหน่งในหน่วยความจำ เราจึงใช้ชื่อของตัวแปรในการอ้างอิง
- ตัวแปร (ภาษาซี) ไม่สามารถใช้งานได้ทันที ต้องถูกประกาศก่อนการใช้งาน
- การประกาศตัวแปร ประกอบด้วย ชนิดข้อมูล ชื่อตัวแปร (อาจจะมีการกำหนดค่าเริ่มต้นด้วย) และตามด้วย ; (semi-colon) เช่น int x; หรือ int x = 2;
- ชื่อตัวแปร เป็นการประกอบกันระหว่าง ตัวอักษร ตัวเลข เครื่องหมาย _ (underscore) ทั้งนี้ห้ามขึ้นต้นด้วยตัวเลข และไม่เป็นคำสงวน

Computer Programming 116

ตัวอย่างการประกาศตัวแปร (variable)

ประกาศตัวแปร cf มีชนิดเป็นตัวอักษร 1 ตัว

```
char cf;
```

ประกาศตัวแปร number มีชนิดเป็นเลขจำนวนเต็มพร้อมทั้งกำหนดให้มีค่าเท่ากับ 8

```
int number=8;
```

ประกาศตัวแปร price มีชนิดเป็นเลขทศนิยม

```
float price;
```

Computer Programming 117

ตัวแปร (variable)

ภาษา C จะให้ความสำคัญกับชนิดข้อมูลอย่างเคร่งครัด

- int เก็บตัวเลขจำนวนเต็ม
- float เก็บตัวเลขทศนิยม
- double เก็บตัวเลขทศนิยมที่มีความแม่นยำสูงกว่า
- char เก็บตัวอักษร

ข้อมูลบางชนิดสามารถใช้ตัวปรับอย่าง **signed (ติดลบได้)**, **unsigned (ไม่เอาที่ติดลบ)**, **short (เก็บแค่น้อยๆ)**, และ **long (เก็บได้กว้างขึ้น)** เพื่อปรับขนาดหรือความสามารถในการเก็บข้อมูลได้

Computer Programming 106

Code Format หรือ Format Specifier

%d : ใช้แสดงผลจำนวนเต็ม (integer) เช่น 5, -3

%f : ใช้แสดงผลตัวเลขทศนิยม (float) เช่น 3.14, 5.0

%lf : ใช้แสดงผลตัวเลขทศนิยมแบบ double

%c : ใช้แสดงผลตัวอักษร (character) เช่น 'A', 'b'

%s : ใช้แสดงผลข้อความ (string) เช่น "Hello", "World"

%u : ใช้แสดงผลจำนวนเต็มบวก (unsigned integer)

%x หรือ %X : ใช้แสดงผลเลขฐานสิบหก (hexadecimal)

%o : ใช้แสดงผลเลขฐานแปด (octal)

%p : ใช้แสดงที่อยู่หน่วยความจำ (pointer address)

%% : ใช้แสดงเครื่องหมาย % บนหน้าจอ

Computer Programming 165

รหัสควบคุม (Escape Sequence)

\n	ใช้สำหรับขึ้นบรรทัดใหม่ (new line)
\t	ใช้สำหรับเว้นวรรคแบบแท็บ (tab space)
\'	Single Quote '
\"	Double Quote "
\a	เสียง beep

Computer Programming 166

คำสั่ง

ให้นศ. ประกาศตัวแปร กำหนดค่าของตัวแปรและแสดงค่าของตัวแปรผ่านคำสั่ง printf ดังตัวแปรต่อไปนี้

(1) ตัวแปรจำนวนเต็ม ชื่อ num กำหนดค่าเท่ากับ 40

(2) ตัวแปรทศนิยมเลขทศนิยมไม่เกิน 6 หลัก ชื่อ abc กำหนดค่าเท่ากับ 3.14159 (แสดงผลทศนิยมแค่4ตำแหน่ง)

(3) ตัวแปรตัวอักษรชื่อ c กำหนดค่าเท่ากับตัวอักษรเลข 4

(4) ตัวแปรทศนิยมที่มีเลขทศนิยมเกิน 6 หลัก ชื่อ average ให้มีค่า 3.91818383918389 (ต้องแสดงทศนิยมให้ครบทุกหลัก)

(5) ตัวแปรชุดตัวอักษร (ข้อความ) ชื่อ name ให้มีค่าชื่อภาษาอังกฤษของนศ.

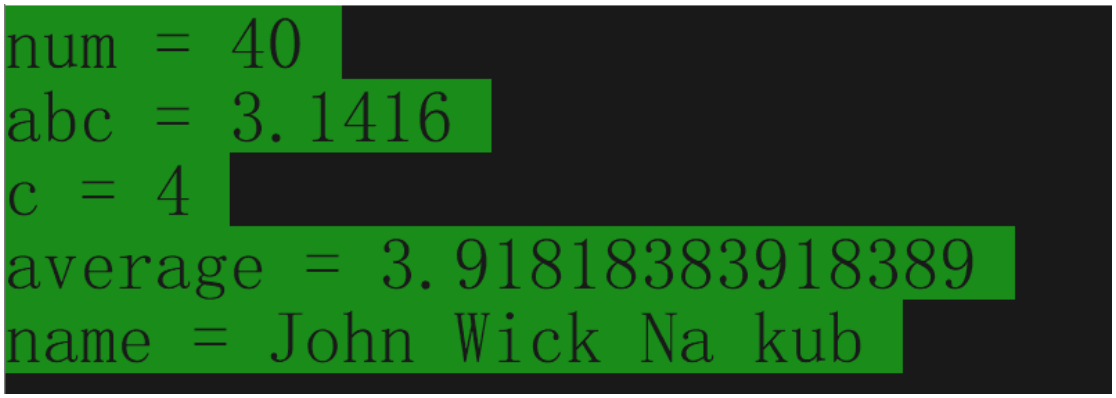
******* หลังจากประกาศตัวแปรแล้ว ให้ดึงชื่อตัวแปร = ค่าของตัวแปร มาแสดงผลด้วยคำสั่ง printf *******

BONUS : หลังจากแสดงค่าในตัวแปร ให้scanf รับค่าค่าจากผู้ใช้ และพิมพ์ออกทางหน้าจออีกครั้ง -> ให้ *

ตัวอย่างข้อ (1)

```
int num;
num=40;
printf("num = %d \n",num);
```

ตัวอย่างผลลัพธ์



ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 3.3

เรื่อง printf from variables

วัตถุประสงค์

1. สามารถประกาศตัวแปร และดึงค่าจากตัวแปรมาแสดงผลด้วย printf ได้
2. สามารถนำโปรแกรมไปรันเพื่อแสดงผลในหน้า Console ได้

ทบทวนความรู้

scanf = รับค่าจากคีย์บอร์ด (ทางหน้าจอ Command line)

ฟังก์ชัน scanf() ใช้สำหรับรับค่าการพิมพ์จากคีย์บอร์ด โดยมีรูปแบบดังนี้

scanf("%ตัวระบุข้อมูล", ที่อยู่ของตัวแปร(Address)ที่ต้องการแสดง);

scanf("%ตัวระบุข้อมูล", &ชื่อตัวแปร); // การใส่ & หน้าชื่อตัวแปรคือการบอกว่าให้ส่งค่า address ของตัวแปรมา

ส่วน %ตัวระบุข้อมูล คือ code format หรือ format specifier ที่ใช้บอกชนิดของข้อมูลที่จะถูกแสดง เช่น %d, %f, %s เป็นต้น

เช่น int count;

scanf("%d",&count);

ตัวอย่าง scanf

```c
#include <stdio.h>
int main() {
    char str[100];    // ตัวแปรประเภท char[] (สตริงหรือข้อความ)
    // รับค่าตัวแปรประเภท char[] (ข้อความหรือสตริง)
    printf("Enter a string (no spaces): ");
    scanf("%s", str);  // ใช้ %s สำหรับ char[] (รับสตริงที่ไม่มีช่องว่าง)
    printf("You entered: %s\n", str);
}
```

```
input
Enter a string (no spaces): JohnWick
You entered: JohnWick
```

%[^\n]s มีเว้นวรรคได้

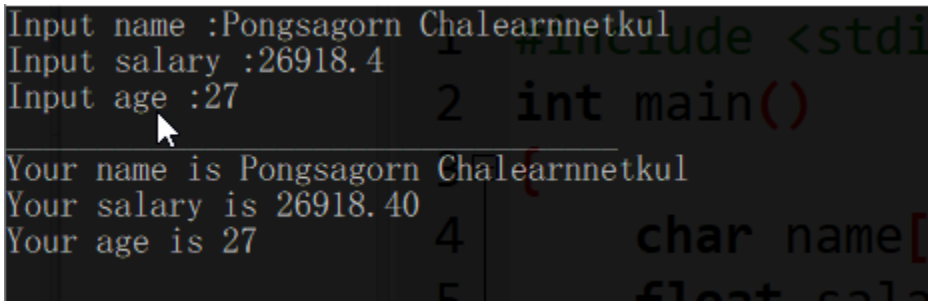
```c
#include <stdio.h>
int main() {
    char str[100];    // ตัวแปรประเภท char[] (สตริงหรือข้อความ)
    // รับค่าตัวแปรประเภท char[] (ข้อความหรือสตริง)
    printf("Enter a string (with spaces): ");
    scanf("%[^\n]s", str);
    // ใช้[^\n]s เพื่อบอกว่า terminate char คือ ปุ่ม Enter เท่านั้น
    printf("You entered: %s\n", str);
}
```

```
input
Enter a string (with spaces): John Wick
You entered: John Wick
```

คำสั่ง

ให้นศ.เขียนโปรแกรมรับค่า ชื่อ เงินเดือน อายุ ด้วย คำสั่ง scanf และแสดงผลด้วย printf ดังตัวอย่าง

ตัวอย่างผลลัพธ์



แผนการสอนเรื่อง เงื่อนไขและการตัดสินใจ

สัปดาห์ที่ 5-6

จำนวนชั่วโมงที่ใช้สอน 10 ชั่วโมง

วัตถุประสงค์ เพื่อให้นักศึกษาสามารถ

1. บอกโครงสร้างการเลือก (Selection Structure) และโครงสร้างการวนซ้ำ (Loop Structure) ได้
2. เลือกใช้เงื่อนไขและตัวดำเนินการทางตรรกะต่าง ๆ ในภาษา C ได้
3. อธิบายการทำงานของโครงสร้าง if-else และ switch-case ได้
4. บรรยายความแตกต่างระหว่างการใช้ if-else กับ switch-case ในโปรแกรมได้
5. ใช้โครงสร้าง if-else ในการตัดสินใจตามเงื่อนไขที่กำหนดได้
6. ประยุกต์ใช้ตัวดำเนินการทางตรรกะ เช่น AND, OR และ NOT ในการเขียนโปรแกรมได้
7. วิเคราะห์การทำงานของโค้ดที่มีโครงสร้างการเลือก if-else และ switch-case ได้
8. ประเมินว่าเมื่อใดควรใช้ if-else แทน switch-case เพื่อให้เหมาะสมกับเงื่อนไขต่าง ๆ ในโปรแกรมได้
9. ตรวจสอบความถูกต้องของเงื่อนไขในโครงสร้างการเลือก if-else หรือ switch-case ได้
10. เขียนโปรแกรมที่ใช้โครงสร้างการเลือกได้อย่างเหมาะสม
11. พัฒนาโปรแกรมที่ใช้เงื่อนไขและตัวดำเนินการทางตรรกะในการแก้ไขปัญหาที่กำหนดได้

หัวข้อบรรยาย

บทที่ 4 เงื่อนไขและการตัดสินใจ

4.1 แนะนำโปรแกรมแบบโครงสร้างควบคุม

4.2 เงื่อนไขและเครื่องหมายที่ใช้ในการเปรียบเทียบ

4.3 ตัวดำเนินการทางตรรกะ (Logical Operators)

4.4 โครงสร้างการเลือกแบบถ้าแล้ว (if-else)

4.5 โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

กิจกรรมการเรียนการสอน

1. การบรรยายในชั้นเรียน
2. การให้ผู้เรียนทดลองเขียนรหัสต้นฉบับและทดลองรันโปรแกรมระหว่างบรรยาย
3. การให้ผู้เรียนทำใบฝึกปฏิบัติการ
4. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)

สื่อการสอน

1. เอกสารประกอบการสอนแบบภาพเลื่อนหรือภาพนิ่ง
2. ใบฝึกปฏิบัติการ
3. โปรแกรมพัฒนาโปรแกรมภาษาซี เช่น DevC++, VS Code เป็นต้น
4. การเข้าถึงข้อมูลจากอินเทอร์เน็ต

การประเมินผลการเรียนรู้

ผลการเรียนรู้	วิธีการประเมิน	สัปดาห์ที่ประเมิน	สัดส่วนของการประเมินผล
1. คุณธรรมจริยธรรม	1.1 ประเมินจากการเข้าชั้นเรียน การส่งแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการตามกำหนดเวลาที่มอบหมาย 1.2 ประเมินจากพฤติกรรมของนักศึกษาในการปฏิบัติตามกฎระเบียบและข้อบังคับต่างๆ ทั้งในข้อบังคับในชั้นเรียน ข้อบังคับของมหาวิทยาลัย 1.3 ประเมินจากการคัดลอกแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการ 1.4 ประเมินจากการมีส่วนร่วมในห้องเรียนและการปิดคอมพิวเตอร์ เก็บขยะ เก็บเก้าอี้หลังจากใช้งานเสร็จ	ตลอดภาคการศึกษา	10%
2. ความรู้	2.1 แบบฝึกหัดท้ายบท 2.2 การสอบกลางภาคเรียนและปลายภาคเรียน	2-8, 11-16 9, 17	20% 40%
3. ทักษะทางปัญญา	3.1 ประเมินจากใบฝึกปฏิบัติการ	ตลอดภาคการศึกษา	20%
4. ทักษะความสัมพันธ์ระหว่างบุคคลและความรับผิดชอบ	4.1 พิจารณาจากการทำใบฝึกปฏิบัติการ 4.3 ประเมินจากพฤติกรรมในการทำงานกลุ่ม การระดมความคิดเห็น การตอบคำถามในชั้นเรียน	ตลอดภาคการศึกษา	5%
5. ทักษะวิเคราะห์เชิงตัวเลข การสื่อสารและการใช้เทคโนโลยีสารสนเทศ	5.1 การทดสอบหรือการประเมินจากงานที่มอบหมาย 5.2 สังเกตการใช้คอมพิวเตอร์ในการเขียนโปรแกรม การทำใบฝึกปฏิบัติการ และการค้นหาข้อมูลเพิ่มเติมจากอินเทอร์เน็ต	ตลอดภาคการศึกษา	5%
	รวม		100%

อ่าน Slide เวอร์ชันล่าสุดได้ที่ Google Slide

https://s.coecore.com/scp4

การโปรแกรมคอมพิวเตอร์

บทที่ 4 เงื่อนไขและการตัดสินใจ

(Condition and Decision)

ผู้สอน ดร.พงศกร เจริญเนตรกุล

Outline

- โปรแกรมแบบโครงสร้างควบคุม
- เงื่อนไขและเครื่องหมายที่ใช้ในการเปรียบเทียบ
- ตัวดำเนินการทางตรรกะ (Logical Operators)
- โครงสร้างการเลือกแบบถ้าแล้ว (if-else)
- โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

โครงสร้างควบคุม (Control Structure)

โครงสร้างควบคุม (Control Structure)

โครงสร้างควบคุม (Control Structure) เป็นกลไกที่ใช้ในการควบคุมการทำงานของโปรแกรมให้เป็นไปตามเงื่อนไขที่กำหนด โดยสามารถแบ่งออกเป็น 3 ประเภทหลัก ๆ ได้แก่

1 โครงสร้างการเลือก (Selection Structure) เช่น if()-else , switch-case() ใช้ในการตัดสินใจเลือกทำสิ่งใดสิ่งหนึ่งตามเงื่อนไขที่กำหนดไว้

2 โครงสร้างการวนซ้ำ (Loop Structure) เช่น for(), while(), do...while() ใช้ในการทำงานซ้ำ ๆ จนกว่าจะเป็นไปตามเงื่อนไขที่กำหนด

3 โครงสร้างการกระโดด (Jump Structure) เช่น break, continue, goto ใช้ในการเปลี่ยนการทำงานของโปรแกรมไปยังตำแหน่งอื่น

1 โครงสร้างการเลือก (Selection Structure)

if-else จะใช้เลือกทำงานตามเงื่อนไขที่กำหนด ถ้าเงื่อนไขเป็นจริง (true) ก็จะทำคำสั่งใน if แต่ถ้าเป็นเท็จ (false) ก็จะทำคำสั่งใน else

```
if (condition) {
    // ทำงานเมื่อเงื่อนไขเป็นจริง
} else {
    // ทำงานเมื่อเงื่อนไขเป็นเท็จ
}
```


1 โครงสร้างการเลือก (Selection Structure)

switch-case ใช้เลือกการทำงานตามค่าเฉพาะ เช่น เมื่อค่าตัวแปรตรงกับกรณีใดกรณีหนึ่ง

```
switch (variable) {
    case 1:
        // ทำงานเมื่อค่าเป็น 1
        break;
    case 2:
        // ทำงานเมื่อค่าเป็น 2
        break;
    default:
        // ทำงานเมื่อไม่มีกรณีใดตรง
}
```

2 โครงสร้างการวนซ้ำ (Loop Structure)

for ใช้ในกรณีที่รู้จำนวนครั้งในการทำซ้ำชัดเจน

```
for (initialization; condition; update) {
    // ทำงานในแต่ละรอบของการวนซ้ำ
}
```


2 โครงสร้างการวนซ้ำ (Loop Structure)

while ใช้เมื่อยังไม่รู้จำนวนครั้งในการทำซ้ำแน่นอน แต่จะทำงานจนกว่าเงื่อนไขจะเป็นเท็จ

```
while (condition) {
    // ทำงานตราบใดที่เงื่อนไขเป็นจริง
}
```

2 โครงสร้างการวนซ้ำ (Loop Structure)

do-while คล้าย while แต่จะทำงานอย่างน้อย 1 ครั้งก่อน แล้วค่อยตรวจสอบเงื่อนไข

```
do {
    // ทำงานอย่างน้อย 1 ครั้ง
} while (condition);
```


3 โครงสร้างการกระโดด (Jump Structure)

break; ใช้เพื่อออกจากลูปหรือ switch-case

continue; ใช้เพื่อข้ามคำสั่งที่เหลือในลูป แล้ววนซ้ำรอบถัดไป

goto label_xx; ใช้กระโดดไปยังตำแหน่งที่กำหนด (ไม่นิยมใช้เพราะอาจทำให้โค้ดอ่านยาก)

เงื่อนไขและเครื่องหมายที่ใช้ในการเปรียบเทียบ

เงื่อนไขและเครื่องหมายที่ใช้ในการเปรียบเทียบ

เงื่อนไข (Condition) ที่ใช้เป็นตัวควบคุมการทำงานของโครงสร้างควบคุมแบบการเลือกและวนซ้ำ จะใช้เงื่อนไข หรือ condition เป็นนิพจน์ที่มีค่าทางตรรกะ(Logic) หรือเรียกว่าค่า boolean คือ ค่า จริง (True) หรือ เท็จ (False) ซึ่งในภาษาซี จะใช้ค่า 1 หรือ 0

ในภาษาซี ค่าที่ไม่ใช่ศูนย์จะถือว่ามีค่าความจริงเป็น จริง (True)

ตัวอย่างเงื่อนไข เช่น

- `x` ค่าตัวแปรเดี่ยว ถ้า x เป็นศูนย์ แสดงว่า เป็นเท็จ
- `x>y` เปรียบเทียบค่าของ x กับ y
- `x>0 && x <10` ค่าของ x ต้องมากกว่าศูนย์ และ น้อยกว่าสิบ
- `x != 10` ค่าของ x ต้องไม่ใช่สิบ

เงื่อนไขและเครื่องหมายที่ใช้ในการเปรียบเทียบ ผลลัพธ์การเปรียบเทียบจะเป็นจริง หรือ เท็จ เท่านั้น

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
==	เท่ากันหรือไม่	x == 7
>	มากกว่าหรือไม่	x > y
<	น้อยกว่าหรือไม่	x < y
>=	มากกว่าหรือเท่ากันหรือไม่	x >= y
<=	น้อยกว่าหรือเท่ากันหรือไม่	x <= y
!=	ไม่เท่ากันหรือไม่	x != y

ระวังสับสน! = เป็นการกำหนดค่า แต่ == เป็นการเปรียบเทียบว่าเท่ากันหรือไม่

เงื่อนไขและเครื่องหมายที่ใช้ในการเปรียบเทียบ

ถ้า a = 6 , b = 7

นิพจน์	ผลลัพธ์	การแปลความหมาย
a == b	0	เท็จ
a > b	0	เท็จ
a < b	1	จริง
a >= b	0	เท็จ
a <= b	1	จริง
a != b	1	จริง

ตัวดำเนินการทางตรรกะ (Logical Operators)

ตัวดำเนินการทางตรรกะ (Logical Operators)

ตัวดำเนินการทางตรรกะ (Logical Operators) ในภาษา C ใช้สำหรับการดำเนินการกับค่าความจริง (True/False) ซึ่งได้จากการเปรียบเทียบ โดยตัวดำเนินการเหล่านี้จะส่งผลเป็นค่าความจริงเช่นกัน ได้แก่

!	นิเสธ (NOT)	unary operator
&&	และ (AND)	binary operator
||	หรือ (OR)	binary operator

เช่น if (a > 0 && b < 10)

{ // เงื่อนไขจะเป็นจริงหากทั้ง a มากกว่า 0 และ b น้อยกว่า 10 }

! นิเสธ (NOT)

ตัวดำเนินการนี้ จะต้องการตัวถูกดำเนินการเพียงตัวเดียว (Unary Operator)

ผลลัพธ์ที่ได้มาจะเป็นค่าตรงกันข้ามกับ ค่าทางตรรกะเดิม

นิพจน์ a	!a
TRUE	FALSE
FALSE	TRUE

และ && (AND)

a && b จะมีค่าเป็นจริง เพียงกรณีเดียวคือ ค่าทั้งสองของนิพจน์เป็นจริง

a	b	a && b
0	0	0
0	1	0
1	0	0
1	1	1

หรือ || (OR)

a || b จะมีค่าเป็นเท็จ เพียงกรณีเดียวคือ ค่าทั้งสองของนิพจน์ เป็นเท็จ

a	b	a || b
0	0	0
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

ลำดับความสำคัญของตัวดำเนินการ

หากในนิพจน์มีตัวดำเนินการ(operator)มากกว่าหนึ่ง และไม่มีวงเล็บ ความสำคัญของ operator เป็นไปดังนี้

operator	association	priority
! – (ติดลบ) ++ --	right to left	Highest
* / %	left to right	
+ -	left to right	
< <= > >=	left to right	
== !=	left to right	
&&	left to right	
||	left to right	Lowest

ตัวอย่างลำดับความสำคัญของตัวดำเนินการ

กำหนดให้ int i=5, j=7, k=12; float x=22.5;

นิพจน์ expression	Equivalent expression	value
i+2==k-1	(i+2)==(k-1)	0
3*i-j<22	((3*i) - j) < 22	1
i+2*j>k	(i+(2*j)) > k	1
k+3<=-j+3*i	(k+3) <= ((-j) + (3*i))	0
'a'+1 == 'b'	('a'+1) == 'b'	1
25>=x+4.0	25 >= (x+4.0)	0
i+j<=k==12-k	((i+j) <= k)== (12-k)	0
i>6 || j-6 && k	(i>6) || ((j-6) && k)	1

โครงสร้างการเลือกแบบถ้าแล้ว (if-else)

โครงสร้างการเลือกแบบถ้าแล้ว (if-else)

การทำงานของ **if else จะเป็นการตรวจสอบเงื่อนไขบางอย่างก่อนที่จะทำสิ่งใดสิ่งหนึ่ง** ถ้าสิ่งนั้น "เป็นจริง" ก็ให้ทำตามคำสั่งที่กำหนดไว้ แต่ถ้า "ไม่เป็นจริง" ก็ให้ไปทำอย่างอื่นแทน

ยกตัวอย่างง่าย ๆ เหมือนกับการตัดสินใจในชีวิตประจำวัน

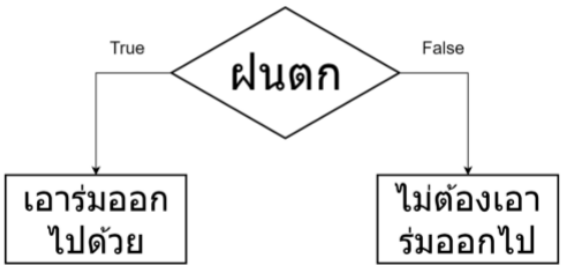
ถ้า (if) วันนี้ฝนตก เราก็จะเอาร่มออกไปด้วย

แต่ถ้า (else) วันนี้ไม่ฝนตก เราก็จะไม่ต้องเอาร่มไป

โครงสร้างการเลือกแบบถ้าแล้ว (if-else)

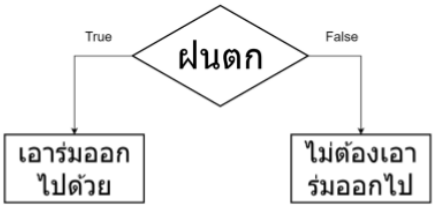
ถ้า (if) วันนี้ฝนตก เราก็จะเอาร่มออกไปด้วย

มิฉะนั้น (else) (วันนี้ไม่ฝนตก) เราก็จะไม่ต้องเอาร่มไป



โครงสร้างการเลือกแบบถ้าแล้ว (if-else)

```
if(ฝนตก)
{
    เอาร่มออกไปด้วย();
}
else
{
    ไม่ต้องเอาร่มออกไป();
}
```



Computer Programming 25

โครงสร้างการเลือกแบบถ้า if

เงื่อนไขต้องเป็นค่าจริงหรือเท็จ

```
if (เงื่อนไขที่ต้องการตรวจสอบ)
```

ตรงนี้ห้ามใส่ ;

```
คำสั่ง; // หากใน if มีมากกว่า 1 คำสั่งให้ใช้ {}
```

ถ้าเงื่อนไขเป็นจริง คำสั่งนี้จะถูกทำงาน

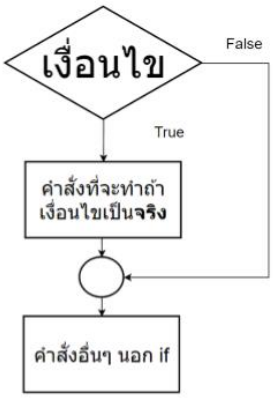
ถ้าเงื่อนไขเป็นเท็จ โปรแกรมจะไม่ทำคำสั่งนี้

Computer Programming 26

โครงสร้างการเลือกแบบถ้า if

if (เงื่อนไขที่ต้องการตรวจสอบ)

คำสั่งที่จะทำหากเงื่อนไขเป็นจริง;



โครงสร้างการเลือกแบบถ้า (if)

ตัวอย่าง if

```
abs_x=x;
if (x < 0)
    abs_x = -x;
```

โปรแกรมนี้ทำงาน ดังนี้

เริ่มจากกำหนดให้ abs_x = x ก่อน

จากนั้นโปรแกรมจะเช็คว่า x น้อยกว่า 0 หรือไม่

ถ้า x น้อยกว่า 0 (จริง) โปรแกรมจะเปลี่ยน abs_x ให้เป็น -x

ถ้า x ไม่ได้ น้อยกว่า 0 (เท็จ) abs_x จะยังคงเท่าเดิม คือ x

ไม่ว่าเงื่อนไขจะเป็นจริงหรือเท็จ โปรแกรมจะพิมพ์ค่า abs_x ออกมา

สรุป

โปรแกรมนี้ใช้ในการหาค่า "ค่าสัมบูรณ์" ของ x โดยตรวจสอบว่า x ติดลบหรือไม่ ถ้าติดลบจะทำให้เป็นบวก แล้วแสดงผล

โครงสร้างการเลือกแบบถ้า (if)

ตัวอย่าง if

```
#include<stdio.h>
int main()
{
  int age;
  printf(”Enter your age: ”);
  scanf(“%d”,&age);
  if(age < 0 || age > 150)
        printf(“Your age is not valid!\n");
  printf(”Goodbye\n”);
}
```

โปรแกรมนี้มีหน้าที่ตรวจสอบว่าอายุที่ผู้ใช้ป้อนเข้ามานั้นเป็นค่าที่ถูกต้องหรือไม่ โดยมีขั้นตอนการทำงานดังนี้

1 โปรแกรมเริ่มต้นด้วยการประกาศตัวแปร age (ซึ่งเป็นตัวแปรประเภท int สำหรับเก็บอายุ)

2 จากนั้นโปรแกรมจะแสดงข้อความว่า "Enter your age: " เพื่อให้ผู้ใช้ป้อนอายุลงไป

3 โปรแกรมจะรับค่าที่ผู้ใช้ป้อนด้วยฟังก์ชัน scanf() แล้วเก็บค่าไว้ในตัวแปร age

4 หลังจากรับค่าแล้ว โปรแกรมจะตรวจสอบว่าอายุที่ป้อนเข้ามานั้นถูกต้องหรือไม่ โดยใช้เงื่อนไข if (age < 0 || age > 150):

ถ้าอายุน้อยกว่า 0 หรือมากกว่า 150 (อายุที่ผิดปกติ) โปรแกรมจะแสดงข้อความว่า "Your age is not valid!"

ถ้าอายุนั้นอยู่ในช่วงปกติ (0-150 ปี) จะไม่แสดงข้อความนี้

5 ไม่ว่าจะเป็นกรณีไหนก็ตาม โปรแกรมจะพิมพ์ "Goodbye" และสิ้นสุดการทำงาน

โครงสร้างการเลือกแบบถ้า (if)

ตัวอย่าง if

```
#include<stdio.h>
int main()
{
  int age;
  printf(”Enter your age: ”);
  scanf(“%d”,&age);
  if(age < 0 || age > 150)
        printf(“Your age is not valid!\n");
  printf(”Goodbye\n”);
}
```

กรณีป้อนอายุถูกต้อง (เช่นอายุ 25)

Enter your age: 25
Goodbye

กรณีป้อนอายุผิดปกติ (เช่นอายุ -5)

Enter your age: -5
Your age is not valid!
Goodbye

กรณีป้อนอายุผิดปกติ (เช่นอายุ 200)

Enter your age: 200
Your age is not valid!
Goodbye

กลุ่มคำสั่ง (Block Statements)

คำสั่งที่อยู่ภายใต้คำสั่ง if-else สามารถมีได้มากกว่า 1 คำสั่ง เรียกว่า กลุ่มคำสั่ง หรือ Block Statements

กลุ่มคำสั่งจะอยู่ภายในเครื่องหมาย {.....}

กลุ่มคำสั่ง (Block Statements)

```
#include<stdio.h>

int main ()
{
  int points= 44;
  if (points>=50)
  {
       printf("Pass exam......\n");
       printf("Congratulations!\n");
  }
  else
  {
       printf("Fail......\n");
       printf("Attempt again\n");
  }
  printf("Bye bye....See you again next semester\n");
}
```

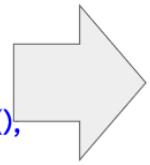
กลุ่มคำสั่ง (Block Statements)

กลุ่มคำสั่ง (Block Statements)

โครงสร้างการเลือกแบบถ้า (if)

การรวมกันของเงื่อนไข โดยใช้ and &&

```
if(ว่าง)
{
        if(ฝนไม่ตก)
        {
                ไปแตะบอล();
    }
}
```

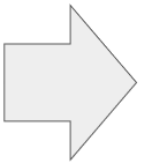


```
if(ว่าง && ฝนไม่ตก)
{
                ไปแตะ
บอล();
}
```


โครงสร้างการเลือกแบบถ้า (if)

การรวมกันของเงื่อนไข โดยใช้ and &&

```
#include<stdio.h>
int main ()
{
        int a =2,b=7;
        if(a>0)
        {
        if (b>0)
            printf("OK.");
        }
}
```



```
#include<stdio.h>
int main ()
{

  int a =2,b=7;
          if((a>0)&&
(b>0))

printf("OK.");
}
```

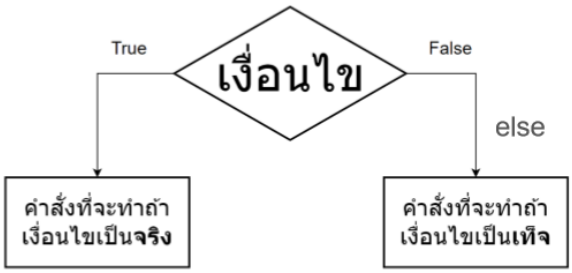
โครงสร้างการเลือกแบบถ้า (if)

จงเขียนโปรแกรมตรวจสอบคะแนนของวิชาที่ไม่นับหน่วยกิต โดยหากได้คะแนนมากกว่าหรือเท่ากับ 50 ให้แสดงคำว่า Pass

```
#include<stdio.h>
int main()
{
   int score;
   // รับค่าคะแนนจากผู้ใช้
   printf("Enter your score: ");
   scanf("%d", &score);
   // ตรวจสอบว่าคะแนนมากกว่าหรือเท่ากับ 50 หรือไม่
   if (score >= 50) {
      printf("Pass: You have successfully passed the course.\n");
   }
}
```


if + else

else สามารถถูกเพิ่มในคำสั่งเงื่อนไขของ if ได้ โปรแกรมจะทำงานที่คำสั่ง else เมื่อเงื่อนไขถูกตรวจสอบว่าเป็นเท็จ



โครงสร้างการเลือกแบบถ้าแล้ว if-else

```
if (เงื่อนไข) {
    // ถ้าเงื่อนไขเป็นจริง ก็ทำคำสั่งในนี้
}
else {
    // ถ้าเงื่อนไขไม่เป็นจริง ก็ทำคำสั่งในนี้แทน
}
```


โครงสร้างการเลือกแบบถ้าแล้ว if-else

ตัวอย่าง

```
int age ;
printf("Enter your age :");
scanf("%d",&age);
if (age >= 18)
{
    printf("You are adult! \n");
}
else {
    printf("You are young! \n");
}
```

```
Enter your age :17
You are young!
```

```
Enter your age :25
You are adult!
```

โครงสร้างการเลือกแบบถ้าแล้ว if-else

ตัวอย่าง เปรียบเทียบจำนวนเต็มสองจำนวนที่ผู้ใช้ป้อนเข้ามา แล้วแสดงผลว่าจำนวนใดมากกว่าหรือน้อยกว่า

```
#include<stdio.h>
int main ()
{
   int a,b;
   printf("Enter integer 1: ") ;
   scanf("%d",&a);
   printf("Enter integer 2: ") ;
   scanf("%d",&b);
   if(a>b)
         printf("%d is greater than %d\n",a,b);
   else
         printf("%d is less than %d\n",a,b);
}
```

อธิบายโปรแกรม

1 โปรแกรมเริ่มต้นด้วยการประกาศตัวแปรสองตัวคือ a และ b

2 โปรแกรมจะแสดงข้อความ "Enter integer 1:" เพื่อให้ผู้ใช้ป้อนค่าจำนวนเต็มแรก จากนั้นรับค่าเก็บไว้ในตัวแปร a ด้วยฟังก์ชัน scanf()

3 โปรแกรมจะแสดงข้อความ "Enter integer 2:" เพื่อให้ผู้ใช้ป้อนค่าจำนวนเต็มที่สอง แล้วรับค่าเก็บไว้ในตัวแปร b

4 โปรแกรมจะตรวจสอบเงื่อนไขโดยใช้ if
4.1 ถ้า a > b โปรแกรมจะแสดงผลว่า a is greater than b (ค่าของ a มากกว่า b)
4.2 ถ้าเงื่อนไข a > b ไม่เป็นจริง (หมายถึง a <= b) โปรแกรมจะแสดงผลว่า a is less than b (ค่าของ a น้อยกว่าหรือเท่ากับ b)

โครงสร้างการเลือกแบบถ้าแล้ว if-else

ตัวอย่าง เปรียบเทียบจำนวนเต็มสองจำนวนที่ผู้ใช้ป้อนเข้ามา แล้วแสดงผลว่าจำนวนใดมากกว่าหรือน้อยกว่า

```
#include<stdio.h>
int main ()
{
   int a,b;
   printf("Enter integer 1: ") ;
   scanf("%d",&a);
   printf("Enter integer 2: ") ;
   scanf("%d",&b);
   if(a>b)
         printf("%d is greater than %d\n",a,b);
   else
         printf("%d is less than %d\n",a,b);
}
```

กรณีที่ป้อน a มากกว่า b

```
Enter integer 1: 10
Enter integer 2: 5
10 is greater than 5
```

กรณีที่ป้อน a น้อยกว่า b

```
Enter integer 1: 4
Enter integer 2: 7
4 is less than 7
```

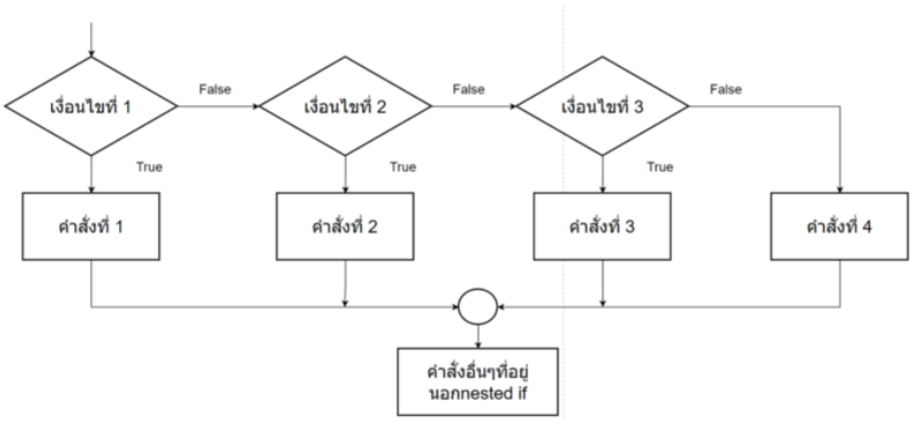
โครงสร้างการเลือกแบบถ้าแล้ว (if-else)

จงเขียนโปรแกรมตรวจสอบคะแนนของวิชาที่ไม่นับหน่วยกิต หากได้คะแนนมากกว่าหรือเท่ากับ 50 ให้แสดงคำว่า Pass นอกเหนือจากนั้น ให้แสดงคำว่า Fail

```
#include<stdio.h>
int main()
{
   int score;
   // รับค่าคะแนนจากผู้ใช้
   printf("Enter your score: ");
   scanf("%d", &score);
   // ตรวจสอบว่าคะแนนมากกว่าหรือเท่ากับ 50 หรือไม่
   if (score >= 50) {
      printf("Pass: You have successfully passed the course.\n");
   } else {
      printf("Fail: You did not pass the course.\n");
   }
}
```


โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

if
else if
else if
else



โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

```
if (เงื่อนไขที่ 1) {
   // ถ้าเงื่อนไขที่ 1 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else if (เงื่อนไขที่ 2) {
   // ถ้าเงื่อนไขที่ 1 ไม่เป็นจริง แต่เงื่อนไขที่ 2 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else if (เงื่อนไขที่ 3) {
   // ถ้าเงื่อนไขที่ 1 และ 2 ไม่เป็นจริง แต่เงื่อนไขที่ 3 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else {
   // ถ้าไม่มีเงื่อนไขใดเป็นจริง จะทำคำสั่งในบล็อกนี้
}
```

- คำสั่งเงื่อนไข if-else-if-else (หรือที่เรียกว่า nested if)
- เป็นโครงสร้างที่ใช้ในการตรวจสอบหลายเงื่อนไขทีละลำดับ
- โดยการทำงานของโปรแกรมจะเริ่มจากตรวจสอบเงื่อนไขแรกก่อน หากเป็นจริง จะทำคำสั่งที่เกี่ยวข้อง และข้ามเงื่อนไขอื่น ๆ ที่เหลือ
- แต่ถ้าไม่เป็นจริงก็จะไปตรวจสอบเงื่อนไขต่อไปจนกว่าจะพบเงื่อนไขที่เป็นจริง หรือเข้าสู่ else หากไม่มีเงื่อนไขใดเป็นจริงเลย

โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

```
#include<stdio.h>
int main ()
{   int a,b;
    printf("Enter integer 1: ") ;
    scanf("%d",&a);
    printf("Enter integer 2: ") ;
    scanf("%d",&b);
    if(a>b)
          printf("%d is greater than %d\n",a,b);
    else if (a==b)
          printf("%d is equal to %d\n",a,b);
    else
          printf("%d is less than %d\n",a,b);
}
```

กรณีที่ป้อน a มากกว่า b

```
Enter integer 1: 10
Enter integer 2: 5
10 is greater than 5
```

กรณีที่ป้อน a น้อยกว่า b

```
Enter integer 1: 4
Enter integer 2: 7
4 is less than 7
```

กรณีที่ป้อน a เท่ากับ b

```
Enter integer 1: 4
Enter integer 2: 7
22 is equal to 22
```

โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

ให้นศ.เขียนโปรแกรมคำนวณการคิดเกรด โดยให้รับคะแนนจากผู้ใช้ หากคะแนนมีค่า

มากกว่าหรือเท่ากับ 80 ให้แสดงเกรด ‘A’

มากกว่าหรือเท่ากับ 70 ให้แสดงเกรด ‘B’

มากกว่าหรือเท่ากับ 60 ให้แสดงเกรด ‘C’

มากกว่าหรือเท่ากับ 50 ให้แสดงเกรด ‘D’

น้อยกว่า 50 ให้แสดงข้อความว่า ‘F’

และไม่ว่าจะได้เกรดอะไรก็ตาม ให้แสดงข้อความ Bye bye....See you again next semester ทุกกรณี

โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

ให้นศ.เขียนโปรแกรมคำนวณการคิดเกรด โดยให้รับคะแนนจากผู้ใช้ หากคะแนนมีค่า

มากกว่าหรือเท่ากับ 80 ให้แสดงเกรด ‘A’

มากกว่าหรือเท่ากับ 70 ให้แสดงเกรด ‘B’

มากกว่าหรือเท่ากับ 60 ให้แสดงเกรด ‘C’

มากกว่าหรือเท่ากับ 50 ให้แสดงเกรด ‘D’

น้อยกว่า 50 ให้แสดงเกรด ‘F’

และไม่ว่าจะได้เกรดอะไรก็ตาม ให้แสดงข้อความ Bye bye....See you again next semester ทุกกรณี



```
int score;
printf("Enter your score: ");
scanf("%d", &score);
// ตรวจสอบเกรดตามคะแนน
if (score >= 80) {
   printf("Grade: A\n");
} else if (score >= 70) {
   printf("Grade: B\n");
} else if (score >= 60) {
   printf("Grade: C\n");
} else if (score >= 50) {
   printf("Grade: D\n");
} else {
   printf("Grade: F\n");
}
// แสดงข้อความนี้ทุกกรณี
printf("Bye bye....See you again next semester\n");
```

สรุป if-else

```
if (เงื่อนไข)
  คำสั่ง; // T
```

```
if (เงื่อนไข)
  คำสั่งที่ 1; // T
else
  คำสั่งที่ 2; // F
```

```
if (เงื่อนไขที่ 1)
  คำสั่งที่ 1; หรือ กลุ่มคำสั่งที่ 1
else if (เงื่อนไขที่ 2)
  คำสั่งที่ 2;
else if (เงื่อนไขที่ 3)
  คำสั่งที่ 3;
else คำสั่งที่ 4;
** หากทำตามเงื่อนไขใดแล้ว จะหลุดออก
จาก nested if เลย
```


โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

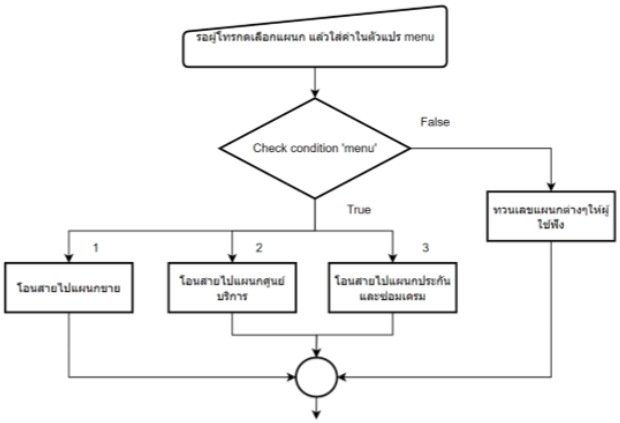
โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

- ใช้ควบคุมให้โปรแกรมเลือกดำเนินการไปในเส้นทางใดเส้นทางหนึ่ง(case) จากทางเลือกหลายๆทาง โดยใช้ค่าที่ต้องการตรวจสอบว่าตรงกับค่าใด
- นิพจน์ที่อยู่ในคำสั่ง switch จะถูกตรวจสอบว่าตรงกับ case ใด แล้วโปรแกรมก็จะทำงานตามคำสั่งที่อยู่ใน case นั้น และคำสั่งอื่นๆที่ตามมาจนจบโครงสร้าง switch-case หรือเจอคำสั่ง break ก็จะออกจากโครงสร้าง switch-case
- โดยค่าที่ใช้ตรวจสอบ จะต้องเป็นจำนวนเต็มหรือตัวอักษรเท่านั้น (int, char)

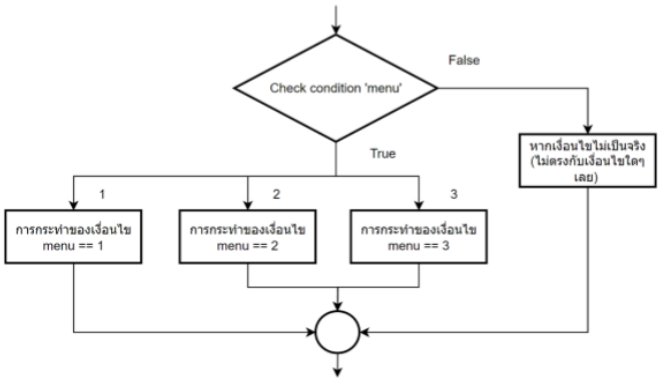
```
switch (ตัวแปรที่ต้องการตรวจสอบค่า)
{
        case X : do_x(); break;
        case Y : do_y(); break;
        case Z : do_z(); break;
        default : do_default();
}
```


โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

ตัวอย่าง switch-case กรณีระบบโทรศัพท์สำนักงาน



โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)



เปรียบเทียบหลายเงื่อนไขแบบ switch-case

ถ้า menu == 1 จริง(True)
ให้ดำเนินการบางอย่างของทางเลือก == 1

ถ้า menu == 2 จริง(True)
ให้ดำเนินการบางอย่างของทางเลือก == 2

ถ้า menu == 3 จริง(True)
ให้ดำเนินการบางอย่างของทางเลือก == 3

ถ้า menu == 4 จริง(True)
ให้ดำเนินการบางอย่างของทางเลือก == 4

ถ้าไม่จริง (False) ซึ่งไม่ตรงกับเงื่อนไขใดๆเลย สามารถให้กระทำบางอย่าง เช่น บอกว่าตัวเลือก/เงื่อนไขที่เลือกมาไม่มีอยู่

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

```
switch ( นิพจน์-ตัวแปรที่ต้องการตรวจสอบ int หรือ char เท่านั้น )
{
  case ค่าที่ 1 :   คำสั่งที่ 1;
  case ค่าที่ 2 :   คำสั่งที่ 2;
  case ค่าที่ 3 :   คำสั่งที่ 3;
  case  ... ...
}
```

นิพจน์ที่อยู่ในคำสั่ง switch จะถูกตรวจสอบตามลำดับว่าตรงกับ case ใด โปรแกรมก็จะทำงานตามคำสั่งที่อยู่ใน case นั้น รวมถึงคำสั่งอื่นๆที่ตามมา ใน case ที่เหลือจนจบโครงสร้าง switch-case หรือเจอคำสั่ง break ก็จะออกจากโครงสร้าง switch-case ได้

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

```
#include<stdio.h>
int main()
{
        int c;
        printf("Enter integer 1 or 2 or 3:");
        scanf("%d",&c);
        switch(c)
        {
        case 1: printf("ONE\n");

        case 2: printf("TWO\n");

        case 3: printf("THREE\n");
        }
}
```

Output

```
Enter integer 1 or 2 or 3: 3
THREE

Enter integer 1 or 2 or 3: 2
TWO
THREE      // สิ่งเกินมา เนื่องจากถ้าตรงcaseใด จะทำจนสุดทาง

Enter integer 1 or 2 or 3: 1
ONE
TWO        // สิ่งเกินมา เนื่องจากถ้าตรงcaseใด จะทำจนสุดทาง
THREE      // สิ่งเกินมา เนื่องจากถ้าตรงcaseใด จะทำจนสุดทาง
```

พระเอกที่จะมาช่วยหยุดสิ่งที่ทำงานเกินมา คือ ..... คำสั่ง break;

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

- คำสั่ง break จะใช้สำหรับการควบคุมการกระทำ โดยบังคับการกระทำในโครงสร้างควบคุมต่างๆ
- บ่อยครั้งที่จะใช้คำสั่ง break เป็นคำสั่งสุดท้ายในแต่ละ case
- หากไม่มีคำสั่ง break ในชุดคำสั่งของ case ใด โปรแกรมจะทำงานต่อไป ในคำสั่งของ case ถัดๆไป ด้วยจนจบ

```
switch (นิพจน์ที่ต้องการตรวจสอบ)
{
  case ค่าที่ 1 :   คำสั่งที่ 1;  break;
  case ค่าที่ 2 :   คำสั่งที่ 2;  break;
  case ค่าที่ 3 :   คำสั่งที่ 3;  break;
  case  ...  ...  break;
}
```

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

```
#include<stdio.h>
int main()
{
        int c;
        printf("Enter integer 1 or 2 or 3:");
        scanf("%d",&c);
        switch(c)
        {
        case 1: printf("ONE\n"); break;

        case 2: printf("TWO\n"); break;

        case 3: printf("THREE\n"); break;
        }
}
```

Output

Enter integer 1 or 2 or 3: 3 THREE
Enter integer 1 or 2 or 3: 2 TWO
Enter integer 1 or 2 or 3: 1 ONE

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

- **default** เป็นอีกกรณีหนึ่งในคำสั่ง switch-case มักวางไว้เป็นกรณีสุดท้าย **โดย default จะทำงานคล้ายๆ else**
- ในกรณีที่ตรวจสอบแล้วพบว่า นิพจน์ มีค่าไม่ตรงกับ case ใดๆเลยข้างต้น โปรแกรมจะเข้าไปทำงานในส่วนของ default
- **ไม่จำเป็นต้องใส่ break หลังชุดคำสั่งของ default**

```
switch (นิพจน์ที่ต้องการตรวจสอบ)
{
 case ค่าที่ 1 :   คำสั่งที่ 1;  break;
 case ค่าที่ 2 :   คำสั่งที่ 2;  break;
 case ค่าที่ 3 :   คำสั่งที่ 3;  break;
 case  ...  ... break;
 default : คำสั่งของ default;
}
```

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

```
#include<stdio.h>
int main()
{
        int c;
        printf("Enter integer 1 or 2 or 3:");
        scanf("%d",&c);
        switch(c)
        {
        case 1: printf("ONE\n"); break;
        case 2: printf("TWO\n"); break;
        case 3: printf("THREE\n"); break;
        default: printf("Out of range");
        }
}
```

Output

Enter integer 1 or 2 or 3: 3 THREE
Enter integer 1 or 2 or 3: 2 TWO
Enter integer 1 or 2 or 3: 1 ONE
Enter integer 1 or 2 or 3: 5 Out of range

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

```
#include<stdio.h>
int main()
{   char grade;
    printf("Rating the service: ");
    scanf("%c",&grade);
    switch(grade)
    {
        case 'G': case 'g':
        case '3': printf("Good\n"); break;
        case 'K': case 'k':
        case '2': printf("Okay\n"); break;
        case 'B': case 'b':
        case '1': printf("Bad\n"); break;
        default: printf("Please choose correctly option!\n");
    }
}
```

กด g / G / 3 ก็แสดง Good เหมือนกัน

กด k / K / 2 ก็แสดง Okay เหมือนกัน

กด b / B / 1 ก็แสดง Bad เหมือนกัน

switch-case vs if-else

การเลือกใช้ switch-case หรือ if-else ขึ้นอยู่กับกรณีและความเหมาะสมของเงื่อนไขที่คุณต้องการตรวจสอบ

1. switch-case:

ควรใช้เมื่อ

- มีการเปรียบเทียบค่าแบบตรงไปตรงมา (เฉพาะค่าคงที่หรือตัวเลข) เช่น การตรวจสอบค่าเท่ากันของตัวแปรกับค่าที่ระบุไว้
- มีเงื่อนไขหลายกรณีที่ต้องตรวจสอบค่าเดียวกัน โดยการเปรียบเทียบค่าเดียวในแต่ละกรณี (เช่น การตรวจสอบค่าอายุ หรือวันในสัปดาห์)
- ทำให้โค้ดดูเรียบร้อยและอ่านง่ายในกรณีที่มีหลายทางเลือกที่เป็นค่าแน่นอน

ข้อดี

- เหมาะกับการตรวจสอบหลายเงื่อนไขที่เกี่ยวกับค่าคงที่ เช่น เลข, ตัวอักษร หรือค่าคงที่อื่น ๆ
- การประมวลผลที่เร็วกว่า เพราะในบางกรณีอาจทำงานได้เร็วกว่า if-else เพราะโครงสร้างของ switch สามารถทำการค้นหาแบบตาราง (table lookup) ได้ ซึ่งเป็นข้อดีเมื่อมีกรณีที่เยอะมาก

ข้อจำกัด

- ไม่รองรับการตรวจสอบเงื่อนไขที่ซับซ้อนหรือเงื่อนไขช่วง (เช่น a > b หรือ a <= 10)
- ใช้ได้กับค่าคงที่หรือค่าคงที่ประเภทจำนวนเต็ม (int) หรืออักขระ (char) เท่านั้น ไม่รองรับเงื่อนไขแบบอื่น เช่น ค่า boolean หรือการเปรียบเทียบที่มีหลายเงื่อนไข

switch-case vs if-else

2. if-else

ควรใช้เมื่อ

- ต้องการตรวจสอบเงื่อนไขที่ซับซ้อน เช่น การตรวจสอบหลายเงื่อนไขพร้อมกัน หรือการเปรียบเทียบแบบช่วง เช่น a > b หรือ a <= 50
- ต้องการเงื่อนไขที่มีการเปรียบเทียบซับซ้อน เช่น การเช็คหลายเงื่อนไขที่ไม่สามารถใช้กับ switch-case ได้
- มีการเปรียบเทียบค่าที่ไม่ใช่ค่าคงที่ เช่น ตัวแปรที่อาจเปลี่ยนแปลงได้ หรือใช้ตรรกะซับซ้อนที่รวมกันหลายเงื่อนไข

ข้อดี:

- ยืดหยุ่น เพราะใช้ได้กับเงื่อนไขทุกประเภท เช่น การเปรียบเทียบค่าที่เปลี่ยนแปลงได้ การเปรียบเทียบช่วง หรือการตรวจสอบค่าแบบซับซ้อน
- รองรับเงื่อนไขแบบใดก็ได้ ไม่ว่าจะเป็นการเปรียบเทียบตัวเลข, ค่า boolean, หรือการเชื่อมโยงเงื่อนไขหลายแบบ

ข้อจำกัด:

- โครงสร้างอาจดูซับซ้อนหรืออ่านยากขึ้น ถ้ามีเงื่อนไขเยอะมาก
- ถ้าใช้ในการตรวจสอบค่าเดิมซ้ำหลาย ๆ ครั้ง (เช่นตรวจสอบค่าตัวเลข) อาจทำงานช้ากว่า switch-case ในบางกรณี

switch-case vs if-else

- switch-case ใช้ในกรณีที่มีทางเลือกหลายทาง โดยขึ้นอยู่กับค่าของตัวแปร(หรือนิพจน์)หนึ่ง **ที่มีค่าเป็น int หรือ char เท่านั้น**
- if-else ใช้ตรวจสอบเงื่อนไข ได้หลากหลายกว่า ตามต้องการ (เช่น เปรียบเทียบค่าน้อยกว่า มากกว่า หรืออยู่ในช่วงใดช่วงหนึ่ง รวมถึงการใช้ logical operator สร้างเงื่อนไขได้ซับซ้อนขึ้น)แต่ถ้ามีหลายทางเลือก ต้องใช้ if ซ้อนกันหลายๆชั้น หรือสร้างเงื่อนไขที่ซับซ้อนขึ้น
- บางกรณีสามารถแปลงคำสั่ง switch-case เป็น if-else ได้ เช่น

```
switch(a)
{
  case 1: คำสั่ง_1; break;
  case 2: คำสั่ง_2; break;
  case 3: คำสั่ง_3; break;
  default: คำสั่ง_4;
}
```



```
if(a==1)
  คำสั่ง_1;
else if (a==2)
  คำสั่ง_2;
else if (a==3)
  คำสั่ง_3;
else
  คำสั่ง_4;
```

Computer Programming

switch-case vs if-else

จงเขียนโปรแกรม รับตัวอักษรหนึ่งตัว แล้วตรวจสอบว่าเป็น สระ(vowel) หรือ พยัญชนะ (consonant) สมมติว่าผู้ใช้ใส่เฉพาะตัวอักษร a-z เท่านั้น

โดยให้เติมส่วนของโปรแกรมนี้ให้สมบูรณ์

```
char ch;
printf("Enter a character(a-z): ");
scanf("%c",&ch);
เติมเต็มโปรแกรมในส่วนนี้  จะเลือกใช้โครงสร้าง if-else หรือ switch-case ก็ได้
คำแนะนำ ให้ตรวจสอบค่า ch ว่าเป็นตัวอักษร สระ a e i o u หรือไม่ ถ้าใช่ก็พิมพ์ vowel ถ้าไม่ใช่ ก็ให้พิมพ์ว่าว่าเป็น consonant
printf("End of the program.\n");
```

Computer Programming

switch-case vs if-else

จงเขียนโปรแกรม รับตัวอักษรหนึ่งตัว แล้วตรวจสอบว่าเป็น สระ(vowel) หรือ พยัญชนะ (consonant) สมมติว่าผู้ใช้ใส่เฉพาะตัวอักษร a-z เท่านั้น

```
char ch;
printf("Enter a character(a-z): ");
scanf("%c",&ch);
if (ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u')
    printf("Vowel\n");
else
    printf("Consonant\n");
printf("End of the program.\n");
```

```
switch (ch)
{
  case 'a':  printf("Vowel\n");  break;
       case 'e':  printf("Vowel\n");  break;
       case 'i':  printf("Vowel\n");  break;
       case 'o':  printf("Vowel\n");  break;
       case 'u':  printf("Vowel\n");  break;
  default :   printf("Consonant\n");
}
printf("End of the program.\n");
```

Computer Programming

เอกสารอ้างอิง

คณาจารย์ผู้สอนวิชา Introduction to Computer Programming, เอกสารประกอบการบรรยายวิชา 241-101 Introduction to Computer Programming, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยสงขลานครินทร์, 2551

นราธร สังข์ประเสริฐ, เอกสารประกอบการบรรยายวิชา 04-511-101 การโปรแกรมคอมพิวเตอร์, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย , 2558

แหล่งศึกษาเพิ่มเติม

ผศ.นราธร สังข์ประเสริฐ. (2563) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-3945-2

โอภาส เอี่ยมสิริวงศ์. (2559) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-2740-4

รศ.ธีรวัฒน์ ประกอบผล. (2562) คู่มือการเขียนโปรแกรมภาษา C ฉบับสมบูรณ์, ซิมพลิฟาย, ISBN: 978-616-26-24896

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase1] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=aDBmXMF9jOU

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase2] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=wslM0Aqepxs

Mooc การเขียนโปรแกรมคอมพิวเตอร์พื้นฐาน ด้วยภาษาซี 1 | Fundamentals of the C Programming Language https://lms.thaimooc.org/courses/course-v1:KMUTT+KMUTT013+2019/about

Mooc ทักษะการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี | Computer programming skills with C language https://lms.thaimooc.org/courses/course-v1:SU+SU011+2021/about

Coursera: Programming in C, ผู้สอน: Charles Russell Severance https://www.coursera.org/learn/programming-c

Coursera: ความเชี่ยวชาญพิเศษ การเขียนโปรแกรม C เบื้องต้น, ผู้สอน: Andrew D. Hilton et al. https://www.coursera.org/specializations/c-programming

Playlist Youtube บันทึกการสอนในห้องเรียน, วิชาการโปรแกรมคอมพิวเตอร์, ดร.พงศกร เจริญเนตรกุล 2566 https://www.youtube.com/watch?v=ewEfmiXaRi0&list=PL322g2itJn4mAE1utyOzF-C0SP0BRkej3&pp=gAQBiAQB

แบบฝึกหัดท้ายบท

1. โครงสร้างควบคุมมีอะไรบ้าง
2. โครงสร้างการเลือกแบบ if-else ใช้สำหรับอะไร
3. โครงสร้างการเลือกแบบ switch-case แตกต่างจาก if-else อย่างไร
4. ตัวดำเนินการทางตรรกะ (Logical Operators) มีกี่ประเภท อะไรบ้าง
5. เครื่องหมาย ‘==’ ในภาษา C หมายถึงอะไร
6. ลำดับความสำคัญของตัวดำเนินการ (Operator Priority) คืออะไร
7. switch-case จะทำงานอย่างไรเมื่อไม่มีคำสั่ง break
8. ตัวอย่างการใช้ switch-case มีอะไรบ้าง
9. ตัวอย่างเงื่อนไขที่ใช้ใน if-else มีอะไรบ้าง
10. อะไรคือเงื่อนไขที่จะทำให้ตัวดำเนินการ && (AND) เป็นจริง
11. อะไรคือเงื่อนไขที่จะทำให้ตัวดำเนินการ || (OR) เป็นจริง
12. การเขียนโปรแกรม if-else สามารถซ้อนเงื่อนไขได้หรือไม่ ถ้าได้ ให้ยกตัวอย่าง
13. ตัวดำเนินการใดใช้ในการปฏิเสธ (Negation)
14. default ใน switch-case ใช้สำหรับอะไร

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

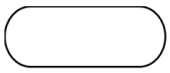
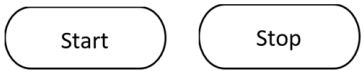
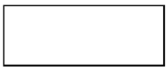
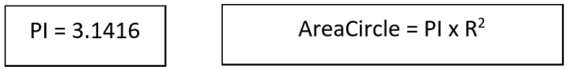
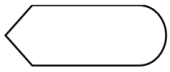
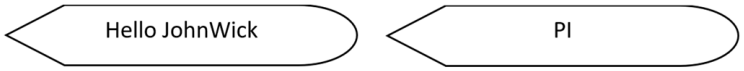
ใบฝึกปฏิบัติการที่ 4.1

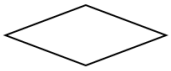
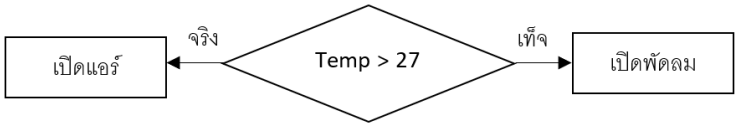
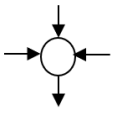
เรื่อง Convert Flowchart to C Code

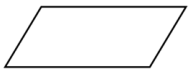
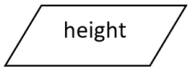
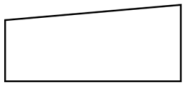
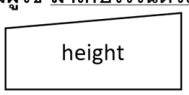
วัตถุประสงค์

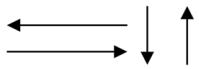
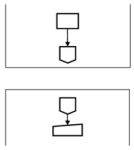
1. สามารถแปลงผังงาน (Flowchart) เป็นโปรแกรมได้
2. สามารถเลือกใช้โครงสร้างการติดสินใจและเงื่อนไขได้อย่างถูกต้อง

ทบทวนความรู้

สัญลักษณ์ของผังงาน	ความหมาย
	**จุดเริ่มต้น /จุดสิ้นสุดของผังงาน** ต้องใส่ ณ จุดแรก และสุดท้ายของผังงานเสมอ 
	**กระบวนการ / การดำเนินการ / การประมวลผล / การคำนวณ / กำหนดค่า** อย่างใดอย่างหนึ่งตามที่กล่าวมา เช่น ประกาศตัวแปร คำนวณสูตรทางคณิตศาสตร์ เรียกใช้คำสั่ง 
	**แสดงข้อความและค่าของตัวแปรผ่านทางจอภาพ** 

สัญลักษณ์ของผังงาน	ความหมาย
	**ใช้ในการตัดสินใจ** หรือ หลายทางเลือก เช่น ถ้า อุณหภูมิมากกว่า 27 ให้เปิดแอร์ 
	**จุดเชื่อมต่อภายในหน้าเดียวกัน** ใช้ในการเชื่อมรวมเส้นทางที่มาจากหลายทางให้ไปทางเดียวกัน เข้ามาได้หลายทางแต่ต้องออกแค่หนึ่งทางเสมอ 

สัญลักษณ์ของผังงาน	ความหมาย
	**นำเข้าข้อมูล (มีอยู่แล้ว เช่น ข้อมูลจากเครือข่าย จากไฟล์ จากฐานข้อมูล) หรือนำเข้าข้อมูลที่ไม่ระบุวิธีการหรือแหล่งข้อมูล มาเก็บไว้ในตัวแปรของโปรแกรม** 
	**การรับข้อมูลเข้าทางแป้นพิมพ์ จากการกด keyboard ของผู้ใช้ มาเก็บไว้ในตัวแปรของโปรแกรม เช่นรับค่าส่วนสูงมาเก็บไว้ในตัวแปร height** 

สัญลักษณ์ของผังงาน	ความหมาย
	**แสดงทิศทางการทำงานของผังงาน (บน ล่าง ซ้าย ขวา เท่านั้น ไม่มีเส้นเฉียง)**
	**จุดเชื่อมต่อระหว่างหน้า** หากเขียนผังงานไม่พอในหนึ่งหน้าให้เขียนอักษรใดสักตัวหนึ่งล้อมด้วยสัญลักษณ์นี้ไว้ในหน้าเดิมและในกระดาษแผ่นใหม่ให้เขียนอักษรเดิมในหน้าที่แล้วล้อมด้วยสัญลักษณ์นี้และดำเนินการต่อผังงานจากจุดนี้ได้เลย 

โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

```
if (เงื่อนไขที่ 1) {
   // ถ้าเงื่อนไขที่ 1 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else if (เงื่อนไขที่ 2) {
   // ถ้าเงื่อนไขที่ 1 ไม่เป็นจริง แต่เงื่อนไขที่ 2 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else if (เงื่อนไขที่ 3) {
   // ถ้าเงื่อนไขที่ 1 และ 2 ไม่เป็นจริง แต่เงื่อนไขที่ 3 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else {
   // ถ้าไม่มีเงื่อนไขใดเป็นจริง จะทำคำสั่งในบล็อกนี้
}
```

- คำสั่งเงื่อนไข if-else-if-else (หรือที่เรียกว่า nested if)
- เป็นโครงสร้างที่ใช้ในการตรวจสอบหลายเงื่อนไขทีละลำดับ
- โดยการทำงานของโปรแกรมจะเริ่มจากตรวจสอบเงื่อนไขแรกก่อน หากเป็นจริง จะทำคำสั่งที่เกี่ยวข้อง และข้ามเงื่อนไขอื่น ๆ ที่เหลือ
- แต่ถ้าไม่เป็นจริงก็จะไปตรวจสอบเงื่อนไขต่อไปจนกว่าจะพบเงื่อนไขที่เป็นจริง หรือเข้าสู่ else หากไม่มีเงื่อนไขใดเป็นจริงเลย

โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

```
#include<stdio.h>
int main ()
{  int a,b;
   printf("Enter integer 1: ") ;
   scanf("%d",&a);
   printf("Enter integer 2: ") ;
   scanf("%d",&b);
   if(a>b)
         printf("%d is greater than %d\n",a,b);
   else if (a==b)
         printf("%d is equal to %d\n",a,b);
   else
         printf("%d is less than %d\n",a,b);
}
```

กรณีที่ป้อน a มากกว่า b

Enter integer 1: 10
Enter integer 2: 5
10 is greater than 5

กรณีที่ป้อน a น้อยกว่า b

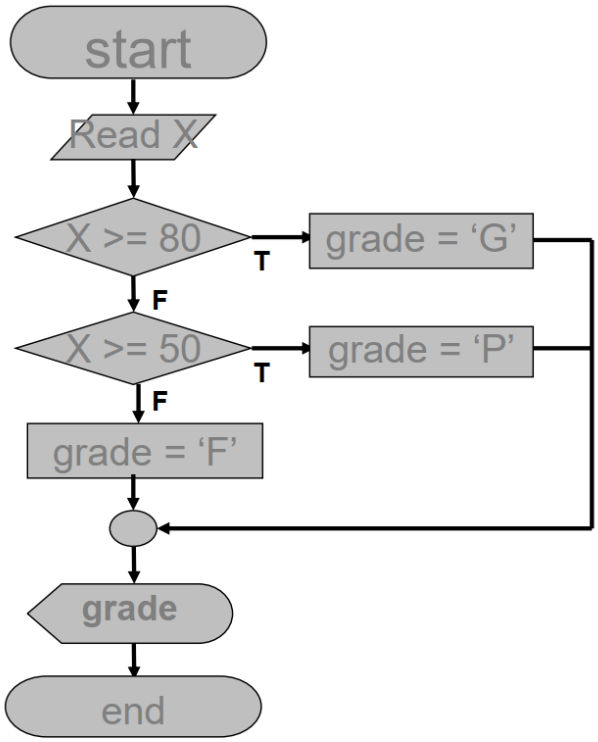
Enter integer 1: 4
Enter integer 2: 7
4 is less than 7

กรณีที่ป้อน a เท่ากับ b

Enter integer 1: 4
Enter integer 2: 7
22 is equal to 22

คำสั่ง

จงแปลง FlowChart ต่อไปนี้ เป็น Code ภาษา C และทดลองรันโปรแกรม



ตัวอย่างการรันโปรแกรม

```
Enter your score: 50
You got grade = P
Enter your score: 80
You got grade = G
Enter your score: 47
You got grade = F
```

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 4.2

เรื่อง ตรวจสอบอายุคนเข้าเว็บไซต์

วัตถุประสงค์

1. สามารถรับค่าจากคีย์บอร์ดมาเก็บไว้ในตัวแปรได้
2. สามารถเลือกใช้โครงสร้างการติดสินใจและเงื่อนไขได้อย่างถูกต้อง

ทบทวนความรู้

โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

```
if (เงื่อนไขที่ 1) {
   // ถ้าเงื่อนไขที่ 1 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else if (เงื่อนไขที่ 2) {
   // ถ้าเงื่อนไขที่ 1 ไม่เป็นจริง แต่เงื่อนไขที่ 2 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else if (เงื่อนไขที่ 3) {
   // ถ้าเงื่อนไขที่ 1 และ 2 ไม่เป็นจริง แต่เงื่อนไขที่ 3 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else {
   // ถ้าไม่มีเงื่อนไขใดเป็นจริง จะทำคำสั่งในบล็อกนี้
}
```

- คำสั่งเงื่อนไข if-else-if-else (หรือที่เรียกว่า nested if)
- เป็นโครงสร้างที่ใช้ในการตรวจสอบหลายเงื่อนไขทีละลำดับ
- โดยการทำงานของโปรแกรมจะเริ่มจากตรวจสอบเงื่อนไขแรกก่อน หากเป็นจริง จะทำคำสั่งที่เกี่ยวข้อง และข้ามเงื่อนไขอื่น ๆ ที่เหลือ
- แต่ถ้าไม่เป็นจริงก็จะไปตรวจสอบเงื่อนไขต่อไปจนกว่าจะพบเงื่อนไขที่เป็นจริง หรือเข้าสู่ else หากไม่มีเงื่อนไขใดเป็นจริงเลย

Computer Programming 43

โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

```
#include<stdio.h>
int main ()
{  int a,b;
   printf("Enter integer 1: ") ;
   scanf("%d",&a);
   printf("Enter integer 2: ") ;
   scanf("%d",&b);
   if(a>b)
         printf("%d is greater than %d\n",a,b);
   else if (a==b)
         printf("%d is equal to %d\n",a,b);
   else
         printf("%d is less than %d\n",a,b);
}
```

กรณีที่ป้อน a มากกว่า b

```
Enter integer 1: 10
Enter integer 2: 5
10 is greater than 5
```

กรณีที่ป้อน a น้อยกว่า b

```
Enter integer 1: 4
Enter integer 2: 7
4 is less than 7
```

กรณีที่ป้อน a เท่ากับ b

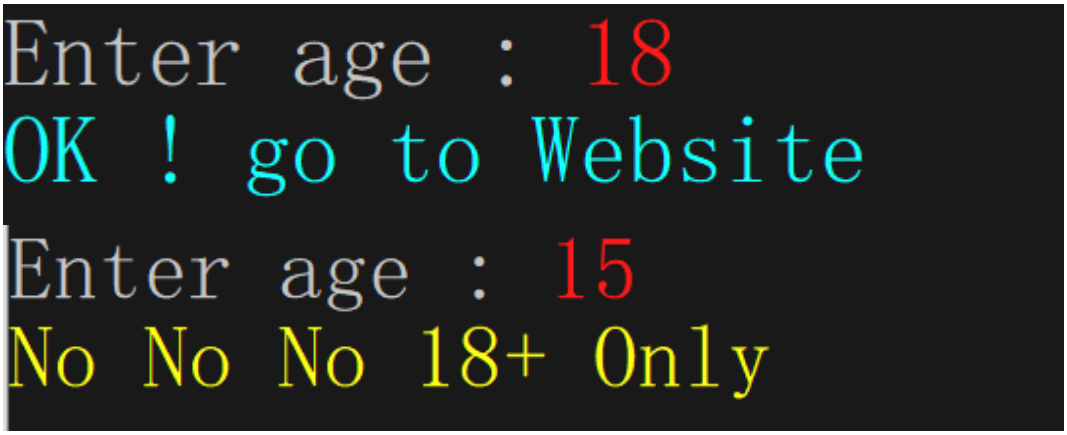
```
Enter integer 1: 4
Enter integer 2: 7
22 is equal to 22
```

Computer Programming 44

คำสั่ง

ให้นศ. เขียนโปรแกรมตรวจสอบอายุคนเข้าเวบไซต์ โดยรับค่าอายุจากผู้ใช้ทางคีย์บอร์ด จากนั้นตรวจสอบเงื่อนไข หากผู้ใช้อายุตั้งแต่ 18 ปี ขึ้นไป ให้แสดงคำว่า OK ! go to Website นอกเหนือจากเงื่อนไขดังกล่าวให้แสดงข้อความว่า No No No 18+ Only

ตัวอย่างการรันโปรแกรม



ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 4.3

เรื่อง IF ELSE คะแนนสอบ

วัตถุประสงค์

1. สามารถรับค่าจากคีย์บอร์ดมาเก็บไว้ในตัวแปรได้
2. สามารถเลือกใช้โครงสร้างการติดสินใจและเงื่อนไขได้อย่างถูกต้อง
3. สามารถใช้โอเปอร์เรเตอร์ทางคณิตศาสตร์ได้

ทบทวนความรู้

โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

```
if (เงื่อนไขที่ 1) {
   // ถ้าเงื่อนไขที่ 1 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else if (เงื่อนไขที่ 2) {
   // ถ้าเงื่อนไขที่ 1 ไม่เป็นจริง แต่เงื่อนไขที่ 2 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else if (เงื่อนไขที่ 3) {
   // ถ้าเงื่อนไขที่ 1 และ 2 ไม่เป็นจริง แต่เงื่อนไขที่ 3 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else {
   // ถ้าไม่มีเงื่อนไขใดเป็นจริง จะทำคำสั่งในบล็อกนี้
}
```

- คำสั่งเงื่อนไข if-else-if-else (หรือที่เรียกว่า nested if)
- เป็นโครงสร้างที่ใช้ในการตรวจสอบหลายเงื่อนไขทีละลำดับ
- โดยการทำงานของโปรแกรมจะเริ่มจากตรวจสอบเงื่อนไขแรกก่อน หากเป็นจริง จะทำคำสั่งที่เกี่ยวข้อง และข้ามเงื่อนไขอื่น ๆ ที่เหลือ
- แต่ถ้าไม่เป็นจริงก็จะไปตรวจสอบเงื่อนไขต่อไปจนกว่าจะพบเงื่อนไขที่เป็นจริง หรือเข้าสู่ else หากไม่มีเงื่อนไขใดเป็นจริงเลย

Computer Programming 43

โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

```
#include<stdio.h>
int main ()
{  int a,b;
   printf("Enter integer 1: ") ;
   scanf("%d",&a);
   printf("Enter integer 2: ") ;
   scanf("%d",&b);
   if(a>b)
         printf("%d is greater than %d\n",a,b);
   else if (a==b)
         printf("%d is equal to %d\n",a,b);
   else
         printf("%d is less than %d\n",a,b);
}
```

กรณีที่ป้อน a มากกว่า b

Enter integer 1: 10
Enter integer 2: 5
10 is greater than 5

กรณีที่ป้อน a น้อยกว่า b

Enter integer 1: 4
Enter integer 2: 7
4 is less than 7

กรณีที่ป้อน a เท่ากับ b

Enter integer 1: 4
Enter integer 2: 7
22 is equal to 22

Computer Programming 44

คำสั่ง

ให้นศ. เขียนโปรแกรมรวมคะแนนสอบของข้อสอบ โดยข้อสอบมี 2 ตอน จากนั้นรวมคะแนนที่ผู้ใช้กรอกมาทั้ง 2 ตอนและแสดงผลคะแนนรวมออกทางหน้าจอ และนำคะแนนรวมมาพิจารณาว่าผ่านหรือไม่ผ่าน

-หากคะแนนรวมมากกว่าหรือเท่ากับ 20 ให้แสดงว่า Passed

-อื่นๆ ให้แสดง Failed

ตัวอย่างการรันโปรแกรม

```
Enter your score sec1: 9
Enter your score sec2: 10
Total Score = 19.0 Failed!!!
```

```
Enter your score sec1: 10
Enter your score sec2: 10
Total Score = 20.0 Passed!!!
```

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 4.4

เรื่อง BMI อ้วนรึเปล่า?

วัตถุประสงค์

1. สามารถรับค่าจากคีย์บอร์ดมาเก็บไว้ในตัวแปรได้
2. สามารถเลือกใช้โครงสร้างการตัดสินใจและเงื่อนไขได้อย่างถูกต้อง
3. สามารถใช้โอเปอร์เรเตอร์ทางคณิตศาสตร์เพื่อการคำนวณสูตรที่ซับซ้อนได้

ทบทวนความรู้

โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

```
if (เงื่อนไขที่ 1) {
   // ถ้าเงื่อนไขที่ 1 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else if (เงื่อนไขที่ 2) {
   // ถ้าเงื่อนไขที่ 1 ไม่เป็นจริง แต่เงื่อนไขที่ 2 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else if (เงื่อนไขที่ 3) {
   // ถ้าเงื่อนไขที่ 1 และ 2 ไม่เป็นจริง แต่เงื่อนไขที่ 3 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else {
   // ถ้าไม่มีเงื่อนไขใดเป็นจริง จะทำคำสั่งในบล็อกนี้
}
```

- คำสั่งเงื่อนไข if-else-if-else (หรือที่เรียกว่า nested if)
- เป็นโครงสร้างที่ใช้ในการตรวจสอบหลายเงื่อนไขทีละลำดับ
- โดยการทำงานของโปรแกรมจะเริ่มจากตรวจสอบเงื่อนไขแรกก่อน หากเป็นจริง จะทำคำสั่งที่เกี่ยวข้อง และข้ามเงื่อนไขอื่น ๆ ที่เหลือ
- แต่ถ้าไม่เป็นจริงก็จะไปตรวจสอบเงื่อนไขต่อไปจนกว่าจะพบเงื่อนไขที่เป็นจริง หรือเข้าสู่ else หากไม่มีเงื่อนไขใดเป็นจริงเลย

Computer Programming 43

โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

```
#include<stdio.h>
int main ()
{  int a,b;
   printf("Enter integer 1: ") ;
   scanf("%d",&a);
   printf("Enter integer 2: ") ;
   scanf("%d",&b);
   if(a>b)
         printf("%d is greater than %d\n",a,b);
   else if (a==b)
         printf("%d is equal to %d\n",a,b);
   else
         printf("%d is less than %d\n",a,b);
}
```

กรณีที่ป้อน a มากกว่า b

Enter integer 1: 10
Enter integer 2: 5
10 is greater than 5

กรณีที่ป้อน a น้อยกว่า b

Enter integer 1: 4
Enter integer 2: 7
4 is less than 7

กรณีที่ป้อน a เท่ากับ b

Enter integer 1: 4
Enter integer 2: 7
22 is equal to 22

Computer Programming 44

คำสั่ง

ให้นศ. เขียนโปรแกรมคำนวณค่า BMI จากสูตรในภาพด้านล่าง และใช้ if-else เพื่อให้แสดงข้อความดังเงื่อนไขดังต่อไปนี้

- ถ้าค่า BMI น้อยกว่า 18.5 ให้แสดงข้อความว่า Too thin
- ถ้าค่า BMI มากกว่าหรือเท่ากับ 18.5 และ น้อยกว่าหรือเท่ากับ 22.9 ให้แสดงข้อความว่า Good
- ถ้าค่า BMI มากกว่า 22.9 และ น้อยกว่าหรือเท่ากับ 24.9 ให้แสดงข้อความว่า Little Fat
- ถ้าค่า BMI มากกว่า 24.9 และ น้อยกว่าหรือเท่ากับ 29.9 ให้แสดงข้อความว่า Big Fat
- ถ้าค่า BMI มากกว่า 29.9 ให้แสดงข้อความว่า Dangerous Fat



ให้ทดสอบความถูกต้องโดยใช้ Test case ต่อไปนี้

w: 50

h: 170

Your BMI: 17.3
Too thin

w: 60
h: 170
Your BMI: 20.8
Good

w: 70
h: 170
Your BMI: 24.2
Little Fat

w: 80
h: 170
Your BMI: 27.7
Big Fat

w: 90
h: 170
Your BMI: 31.1
Dangerous Fat

ตัวอย่างการรันโปรแกรม



ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 4.5

เรื่อง โปรแกรมหน้าร้าน

วัตถุประสงค์

1. สามารถรับค่าจากคีย์บอร์ดมาเก็บไว้ในตัวแปรได้
2. สามารถเลือกใช้โครงสร้างการตัดสินใจและเงื่อนไขได้อย่างถูกต้อง
3. สามารถใช้โอเปอร์เรเตอร์ทางคณิตศาสตร์เพื่อการคำนวณสูตรที่ซับซ้อนได้

ทบทวนความรู้

โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

```
if (เงื่อนไขที่ 1) {
   // ถ้าเงื่อนไขที่ 1 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else if (เงื่อนไขที่ 2) {
   // ถ้าเงื่อนไขที่ 1 ไม่เป็นจริง แต่เงื่อนไขที่ 2 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else if (เงื่อนไขที่ 3) {
   // ถ้าเงื่อนไขที่ 1 และ 2 ไม่เป็นจริง แต่เงื่อนไขที่ 3 เป็นจริง จะทำคำสั่งในบล็อกนี้
}
else {
   // ถ้าไม่มีเงื่อนไขใดเป็นจริง จะทำคำสั่งในบล็อกนี้
}
```

- คำสั่งเงื่อนไข if-else-if-else (หรือที่เรียกว่า nested if)
- เป็นโครงสร้างที่ใช้ในการตรวจสอบหลายเงื่อนไขทีละลำดับ
- โดยการทำงานของโปรแกรมจะเริ่มจากตรวจสอบเงื่อนไขแรกก่อน หากเป็นจริง จะทำคำสั่งที่เกี่ยวข้อง และข้ามเงื่อนไขอื่น ๆ ที่เหลือ
- แต่ถ้าไม่เป็นจริงก็จะไปตรวจสอบเงื่อนไขต่อไปจนกว่าจะพบเงื่อนไขที่เป็นจริง หรือเข้าสู่ else หากไม่มีเงื่อนไขใดเป็นจริงเลย

Computer Programming 43

โครงสร้างการเลือกแบบถ้าแล้ว if - else if - else (Nested if)

```
#include<stdio.h>
int main ()
{  int a,b;
   printf("Enter integer 1: ") ;
   scanf("%d",&a);
   printf("Enter integer 2: ") ;
   scanf("%d",&b);
   if(a>b)
         printf("%d is greater than %d\n",a,b);
   else if (a==b)
         printf("%d is equal to %d\n",a,b);
   else
         printf("%d is less than %d\n",a,b);
}
```

กรณีที่ป้อน a มากกว่า b

Enter integer 1: 10
Enter integer 2: 5
10 is greater than 5

กรณีที่ป้อน a น้อยกว่า b

Enter integer 1: 4
Enter integer 2: 7
4 is less than 7

กรณีที่ป้อน a เท่ากับ b

Enter integer 1: 4
Enter integer 2: 7
22 is equal to 22

Computer Programming 44

คำสั่ง

จงเขียนโปรแกรมขายสินค้าหน้าร้านโดยมีข้อกำหนดต่างๆดังต่อไปนี้

- รับจำนวนของสินค้าต่างๆไปเก็บไว้ในตัวแปรโดยใช้ scanf เดียว การใส่ค่าจำนวนให้ใส่ตามลำดับ ขั้นแต่ละสินค้าด้วยการเว้นวรรค
- แสดงผลรายการสั่งซื้อ x ราคาต่อหน่วย = ราคารวมในแต่ละรายการ
- คิดราคาสินค้ารวม แสดงส่วนลด แสดงราคาสุทธิ

ในร้านค้าของเล่นแห่งหนึ่งขายสินค้า 4 อย่าง

1.เรือดำน้ำ (Submarine) ราคา 4.5 พันล้าน (billion)

2.รถถังยิงมิสซาย (Missile tank) ราคา 1.6 พันล้าน (billion)

3.รถถังพันท์เซอร์ (Panzer tank) ราคา 1.1 พันล้าน (billion)

4.เครื่องบินขับไล่ MiG (MiG AirCraft) ราคา 6.3 พันล้าน (billion)

โดยมีการจัดโปรโปรโมชันส่งเสริมการขายดังนี้

หากซื้อยอดตั้งแต่ 100-200 พันล้าน (billion) จะลด 7.5 %

หากซื้อยอดมากกว่า 200 ถึง 400 พันล้าน (billion) จะลด 10 %

หากซื้อยอดมากกว่า 400 พันล้าน (billion) ขึ้นไป จะลด 15 %

เท่านั้นยังไม่พอ หากซื้อสินค้าในหมวดรถถัง ทุกๆ 50 พันล้าน(ราคาก่อนลด) จะแถมนาฬิกา Richard Mille 2 เรือน

*** หากลูกค้าได้รับของแถมให้แสดงด้วย

ตัวอย่างการรับค่า

```
int nSubmarine,nMissile,nPanzer,nMiG;
printf("Welcome to Millitary Toy\n");
printf("Submarine \t\t 4.5 billion \
        \nMissile tank \t\t 1.6 billion \
        \nPanzer tank \t\t 1.1 billion \
        \nMiG AirCraft \t\t 6.3 billion");
printf("\n Enter #Submarine #MissileTank #Panzer #Mig:");
scanf("%d %d %d %d",&nSubmarine,&nMissile,&nPanzer,&nMiG);
```

ตัวอย่างการรันโปรแกรม

```
!./lab
```

```
==========  ยินดีต้อนรับเข้าสู่ ร้าน Millitary Toy  ==========

รายการสินค้าทั้งหมด
Submarine           4.5 billion
Missile tank        1.6 billion
Panzer tank         1.1 billion
MiG AirCraft        6.3 billion
กรุณากรอกจำนวนที่ต้องการสั่งซื้อ Submarine เว้นวรรค MissileTank เว้นวรรค PanzerTank เว้นวรรค MigAirCraft ตามลำดับ:
40 40 40 10
_____________________________________________________________________
|      รายการ          |      จำนวน      | ราคา      | รวม        |
---------------------------------------------------------------------
|      Submarine       |       40  | 4.5 B     |  180.0B   |
|      Missile Tank    |       40  | 1.6 B     |  64.0B    |
|      Panzer Tank     |       40  | 1.1 B     |  44.0B    |
|      MiG AirCraft    |       10  | 6.3 B     |  63.0B    |
_____________________________________________________________________
ยอดรวม     : 351.00 billion (ยอดมากกว่า200Bได้ส่วนลด 10%)
ส่วนลด     : 35.10 billion
ยอดสุทธิ   : 315.90 billion
ยินดีด้วยคุณได้ของแถมนาฬิกา Richard Mille จำนวน 4 เรือน จากยอดซื้อรถถัง จากยอดซื้อรถถัง 108.0 B
```

```
!./lab
```

```
==========  ยินดีต้อนรับเข้าสู่ ร้าน Millitary Toy  ==========

รายการสินค้าทั้งหมด
Submarine           4.5 billion
Missile tank        1.6 billion
Panzer tank         1.1 billion
MiG AirCraft        6.3 billion
กรุณากรอกจำนวนที่ต้องการสั่งซื้อ Submarine เว้นวรรค MissileTank เว้นวรรค PanzerTank เว้นวรรค MigAirCraft ตามลำดับ:
100 100 100 100
_____________________________________________________________________
|      รายการ          |      จำนวน      | ราคา      | รวม        |
---------------------------------------------------------------------
|      Submarine       |       100 | 4.5 B     |  450.0B   |
|      Missile Tank    |       100 | 1.6 B     |  160.0B   |
|      Panzer Tank     |       100 | 1.1 B     |  110.0B   |
|      MiG AirCraft    |       100 | 6.3 B     |  630.0B   |
_____________________________________________________________________
ยอดรวม     : 1350.00 billion (ยอดมากกว่า400Bได้ส่วนลด 15%)
ส่วนลด     : 202.50 billion
ยอดสุทธิ   : 1147.50 billion
ยินดีด้วยคุณได้ของแถมนาฬิกา Richard Mille จำนวน 10 เรือน จากยอดซื้อรถถัง จากยอดซื้อรถถัง 270.0 B
```

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 4.6

เรื่อง switch-case สั่งอาหาร ด้วยตัวแปร int

วัตถุประสงค์

1. สามารถรับค่าจากคีย์บอร์ดมาเก็บไว้ในตัวแปรได้
2. สามารถเลือกใช้โครงสร้างการติดสินใจแบบ switch-case และเงื่อนไขได้อย่างถูกต้อง

ทบทวนความรู้

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

```
#include<stdio.h>
int main()
{
        int c;
        printf("Enter integer 1 or 2 or 3:");
        scanf("%d",&c);
        switch(c)
        {
        case 1: printf("ONE\n"); break;

        case 2: printf("TWO\n"); break;

        case 3: printf("THREE\n"); break;
        }
}
```

Output

```
Enter integer 1 or 2 or 3: 3
THREE
Enter integer 1 or 2 or 3: 2
TWO
Enter integer 1 or 2 or 3: 1
ONE
```

Computer Programming 55

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

- **default** เป็นอีกกรณีหนึ่งในคำสั่ง switch-case มักวางไว้เป็นกรณีสุดท้าย **โดย default จะทำงานคล้ายๆ else**
- ในกรณีที่ตรวจสอบแล้วพบว่า นิพจน์ มีค่าไม่ตรงกับ case ใดๆเลยข้างต้น โปรแกรมจะเข้าไปทำงานในส่วนของ default
- **ไม่จำเป็นต้องใส่ break หลังชุดคำสั่งของ default**

```
switch (นิพจน์ที่ต้องการตรวจสอบ)
{
 case ค่าที่ 1 :   คำสั่งที่ 1;  break;
 case ค่าที่ 2 :   คำสั่งที่ 2;  break;
 case ค่าที่ 3 :   คำสั่งที่ 3;  break;
 case  ...  ... break;
 default : คำสั่งของ default;
}
```

Computer Programming 56

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

```
#include<stdio.h>
int main()
{
        int c;
        printf("Enter integer 1 or 2 or 3:");
        scanf("%d",&c);
        switch(c)
        {
        case 1: printf("ONE\n"); break;
        case 2: printf("TWO\n"); break;
        case 3: printf("THREE\n"); break;
        default: printf("Out of range");
        }
}
```

Output

Enter integer 1 or 2 or 3: 3 THREE
Enter integer 1 or 2 or 3: 2 TWO
Enter integer 1 or 2 or 3: 1 ONE
Enter integer 1 or 2 or 3: 5 Out of range

Computer Programming 57

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

```
#include<stdio.h>
int main()
{  char grade;
   printf("Rating the service: ");
   scanf("%c",&grade);
   switch(grade)
   {
      case 'G': case 'g':
      case '3': printf("Good\n"); break;
      case 'K': case 'k':
      case '2': printf("Okay\n"); break;
      case 'B': case 'b':
      case '1': printf("Bad\n"); break;
      default: printf("Please choose correctly option!\n");
   }
}
```

กด g / G / 3 ก็แสดง Good เหมือนกัน

กด k / K / 2 ก็แสดง Okay เหมือนกัน

กด b / B / 1 ก็แสดง Bad เหมือนกัน

Computer Programming 58

จงเขียนโปรแกรม แสดงเมนูอาหาร และรับออเดอร์ตามการเลือกของลูกค้า โดยใช้ตัวแปรที่เป็น int ในการตรวจสอบค่าใน switch

โดยโปรแกรมแสดงข้อความเมนูอาหารดังนี้

Menu:

1) Papaya Salad

2) Grilled Chicken

3) Sticky Rice

จากนั้นรับค่าเมนูจากผู้ใช้

Enter Menu Number: -- รอรับค่าจากผู้ใช้ เก็บในตัวแปรชนิด int-- -

> scanf("%d",&menu);

ให้ใช้โครงสร้าง switch-case แสดงข้อความดังเงื่อนไขต่อไปนี้

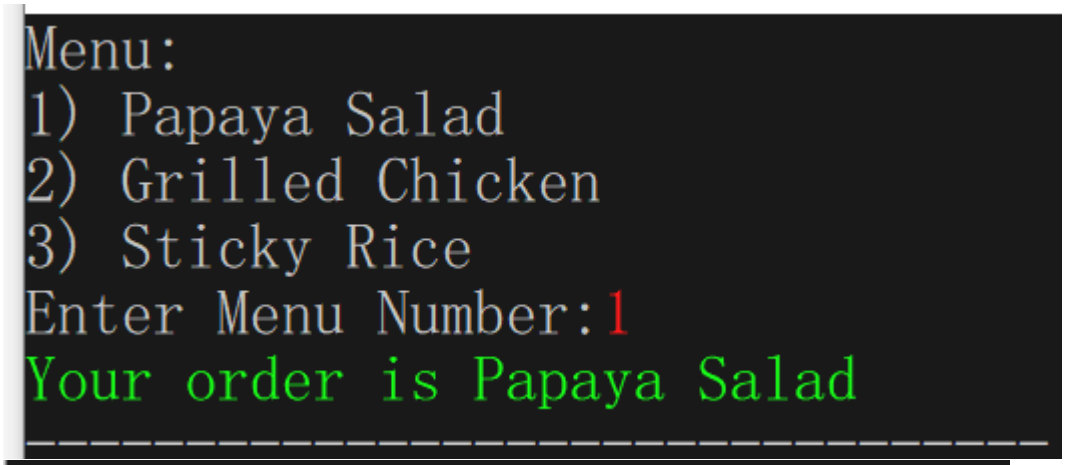
ถ้าผู้ใช้โปรแกรม ใส่เลข 1 ให้แสดงข้อความว่า Your order is Papaya Salad

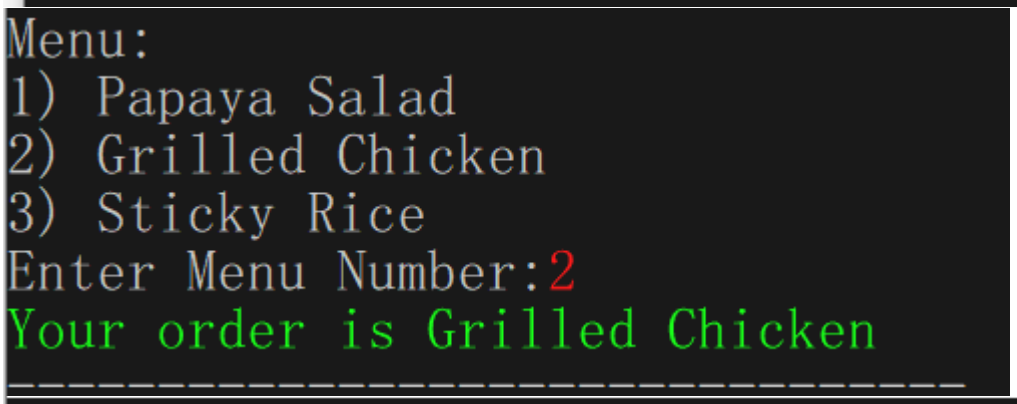
ถ้าผู้ใช้โปรแกรม ใส่เลข 2 ให้แสดงข้อความว่า Your order is Grilled Chicken

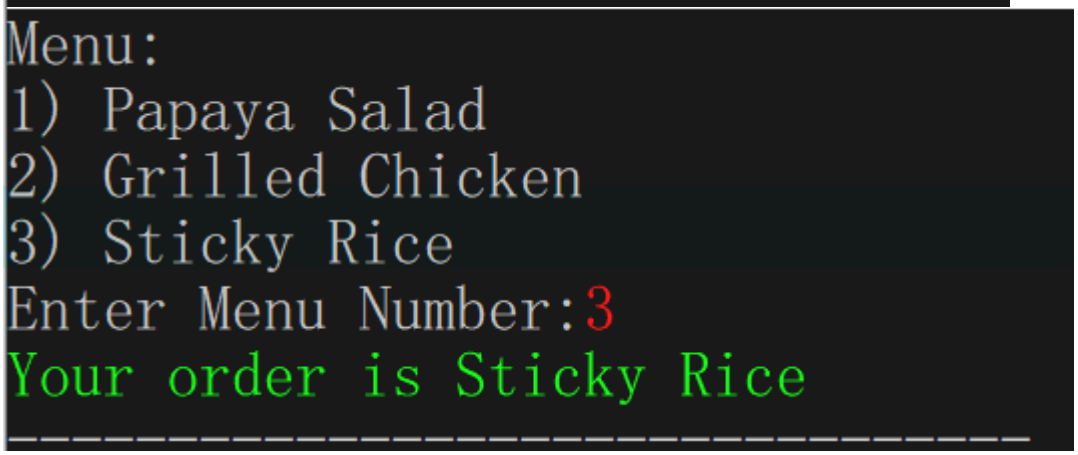
ถ้าผู้ใช้โปรแกรม ใส่เลข 3 ให้แสดงข้อความว่า Your order is Sticky Rice

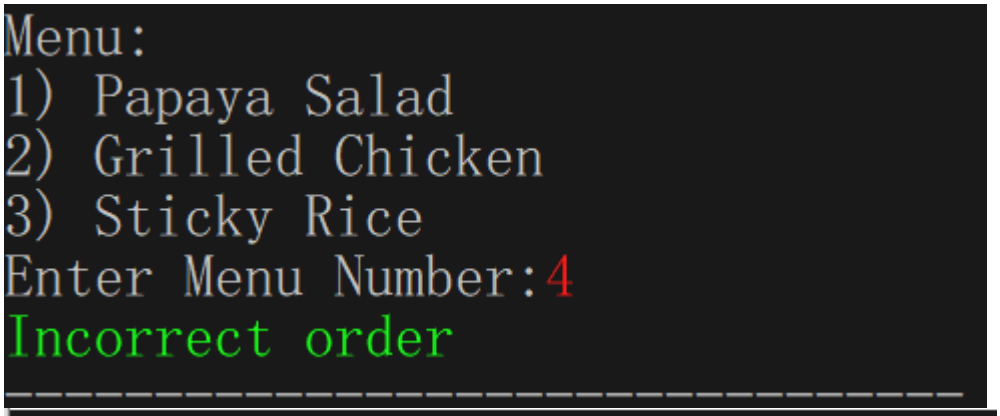
ถ้าผู้ใช้ใส่ค่าอื่นๆ ให้แสดงข้อความว่า Incorrect order

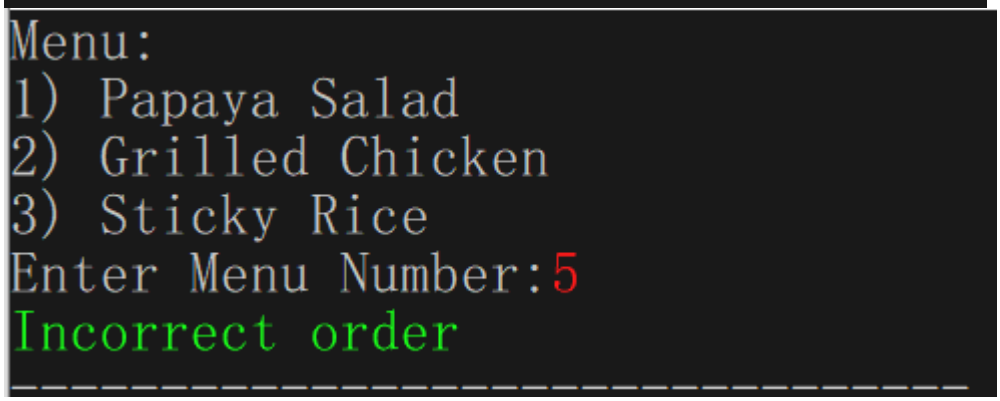
ตัวอย่างผลลัพธ์การทำงานของโปรแกรมนี้








Menu:
1) Papaya Salad
2) Grilled Chicken
3) Sticky Rice
Enter Menu Number:4
Incorrect order


Menu:
1) Papaya Salad
2) Grilled Chicken
3) Sticky Rice
Enter Menu Number:5
Incorrect order

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 4.7

เรื่อง switch-case สั่งอาหาร ด้วยตัวแปร char

วัตถุประสงค์

1. สามารถรับค่าจากคีย์บอร์ดมาเก็บไว้ในตัวแปรได้
2. สามารถเลือกใช้โครงสร้างการติดสินใจแบบ switch-case และเงื่อนไขได้อย่างถูกต้อง

ทบทวนความรู้

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

```
#include<stdio.h>
int main()
{
        int c;
        printf("Enter integer 1 or 2 or 3:");
        scanf("%d",&c);
        switch(c)
        {
        case 1: printf("ONE\n"); break;

        case 2: printf("TWO\n"); break;

        case 3: printf("THREE\n"); break;
        }
}
```

Output

```
Enter integer 1 or 2 or 3: 3
THREE

Enter integer 1 or 2 or 3: 2
TWO

Enter integer 1 or 2 or 3: 1
ONE
```

Computer Programming 55

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

- **default** เป็นอีกกรณีหนึ่งในคำสั่ง switch-case มักวางไว้เป็นกรณีสุดท้าย **โดย default จะทำงานคล้ายๆ else**
- ในกรณีที่ตรวจสอบแล้วพบว่า นิพจน์ มีค่าไม่ตรงกับ case ใดๆเลยข้างต้น โปรแกรมจะเข้าไปทำงานในส่วนของ default
- **ไม่จำเป็นต้องใส่ break หลังชุดคำสั่งของ default**

```
switch (นิพจน์ที่ต้องการตรวจสอบ)
{
 case ค่าที่ 1 :   คำสั่งที่ 1;  break;
 case ค่าที่ 2 :   คำสั่งที่ 2;  break;
 case ค่าที่ 3 :   คำสั่งที่ 3;  break;
 case  ...  ... break;
 default : คำสั่งของ default;
}
```

Computer Programming 56

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

```
#include<stdio.h>
int main()
{
        int c;
        printf("Enter integer 1 or 2 or 3:");
        scanf("%d",&c);
        switch(c)
        {
        case 1: printf("ONE\n"); break;
        case 2: printf("TWO\n"); break;
        case 3: printf("THREE\n"); break;
        default: printf("Out of range");
        }
}
```

Output

Enter integer 1 or 2 or 3: 3 THREE
Enter integer 1 or 2 or 3: 2 TWO
Enter integer 1 or 2 or 3: 1 ONE
Enter integer 1 or 2 or 3: 5 Out of range

Computer Programming 57

โครงสร้างการเลือกแบบทางเลือกหลายทาง (switch-case)

```
#include<stdio.h>
int main()
{  char grade;
   printf("Rating the service: ");
   scanf("%c",&grade);
   switch(grade)
   {
      case 'G': case 'g':
      case '3': printf("Good\n"); break;
      case 'K': case 'k':
      case '2': printf("Okay\n"); break;
      case 'B': case 'b':
      case '1': printf("Bad\n"); break;
      default: printf("Please choose correctly option!\n");
   }
}
```

กด g / G / 3 ก็แสดง Good เหมือนกัน

กด k / K / 2 ก็แสดง Okay เหมือนกัน

กด b / B / 1 ก็แสดง Bad เหมือนกัน

Computer Programming 58

จงเขียนโปรแกรม แสดงเมนูอาหาร และรับออเดอร์ตามการเลือกของลูกค้า

โดยใช้ตัวแปรที่เป็น char ในการตรวจสอบค่าใน switch

ใช้ตัวแปร char เท่านั้น

โดยโปรแกรมแสดงข้อความเมนูอาหารดังนี้

Menu:

1) Papaya Salad

2) Grilled Chicken

3) Sticky Rice

จากนั้นรับค่าเมนูจากผู้ใช้

Enter Menu Number: -- รอรับค่าจากผู้ใช้ เก็บในตัวแปรชนิด char-- -
> scanf("%c",&menu);

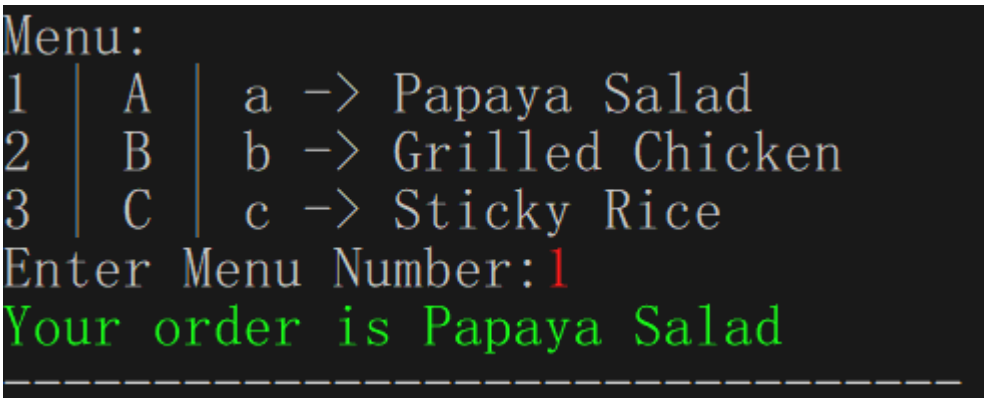
ให้ใช้โครงสร้าง switch-case แสดงข้อความดังเงื่อนไขต่อไปนี้

ถ้าผู้ใช้โปรแกรม **ใส่อักษร a หรือ A หรือ 1** ให้แสดงข้อความว่า **Your order is Papaya Salad**
ถ้าผู้ใช้โปรแกรม **ใส่อักษร b หรือ B หรือ 2** ให้แสดงข้อความว่า **Your order is Grilled Chicken**
ถ้าผู้ใช้โปรแกรม **ใส่อักษร c หรือ C หรือ 3** ให้แสดงข้อความว่า **Your order is Sticky Rice**

ถ้าผู้ใช้ใส่ค่าอื่นๆ ให้แสดงข้อความว่า Incorrect order

ให้ส่งภาพ Output ของโปรแกรม 5 cases (5ภาพ)

ตัวอย่างผลลัพธ์ของโปรแกรม



```
Menu:
1 | A | a -> Papaya Salad
2 | B | b -> Grilled Chicken
3 | C | c -> Sticky Rice
Enter Menu Number:A
Your order is Papaya Salad
------------------------------
```

```
Menu:
1 | A | a -> Papaya Salad
2 | B | b -> Grilled Chicken
3 | C | c -> Sticky Rice
Enter Menu Number:a
Your order is Papaya Salad
------------------------------
```

==========

```
Menu:
1 | A | a -> Papaya Salad
2 | B | b -> Grilled Chicken
3 | C | c -> Sticky Rice
Enter Menu Number:2
Your order is Grilled Chicken
------------------------------
```

```
Menu:
1 | A | a -> Papaya Salad
2 | B | b -> Grilled Chicken
3 | C | c -> Sticky Rice
Enter Menu Number:B
Your order is Grilled Chicken
------------------------------
```

```
Menu:
1 | A | a -> Papaya Salad
2 | B | b -> Grilled Chicken
3 | C | c -> Sticky Rice
Enter Menu Number:b
Your order is Grilled Chicken
--------------------------------
```

==========

```
Menu:
1 | A | a -> Papaya Salad
2 | B | b -> Grilled Chicken
3 | C | c -> Sticky Rice
Enter Menu Number:3
Your order is Sticky Rice
--------------------------------
```

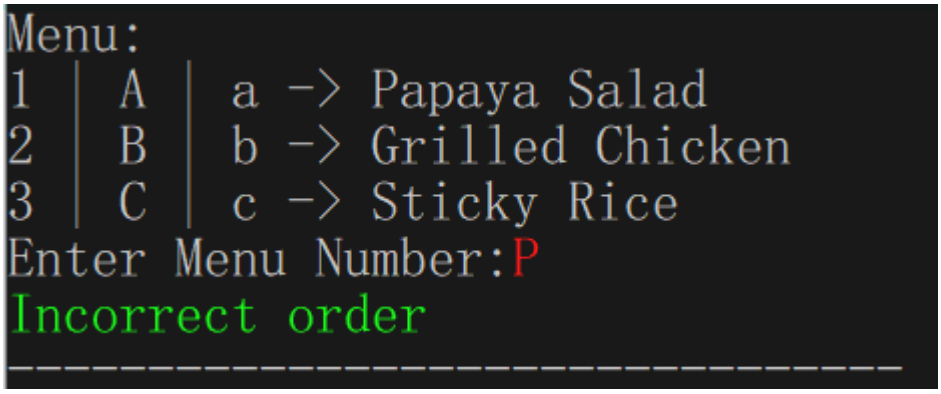
```
Menu:
1 | A | a -> Papaya Salad
2 | B | b -> Grilled Chicken
3 | C | c -> Sticky Rice
Enter Menu Number:C
Your order is Sticky Rice
--------------------------------
```

```
Menu:
1 | A | a -> Papaya Salad
2 | B | b -> Grilled Chicken
3 | C | c -> Sticky Rice
Enter Menu Number:c
Your order is Sticky Rice
--------------------------------
```

==========

```
Menu:
1 | A | a -> Papaya Salad
2 | B | b -> Grilled Chicken
3 | C | c -> Sticky Rice
Enter Menu Number:5
Incorrect order
--------------------------------
```

```
Menu:
1 | A | a -> Papaya Salad
2 | B | b -> Grilled Chicken
3 | C | c -> Sticky Rice
Enter Menu Number:i
Incorrect order
--------------------------------
```


Menu:
1 | A | a -> Papaya Salad
2 | B | b -> Grilled Chicken
3 | C | c -> Sticky Rice
Enter Menu Number:P
Incorrect order

==========
====

แผนการสอนเรื่อง โครงสร้างการวนรอบ

สัปดาห์ที่ 7-8

จำนวนชั่วโมงที่ใช้สอน 10 ชั่วโมง

วัตถุประสงค์ เพื่อให้นักศึกษาสามารถ

1. จำแนกคำสั่งวนรอบประเภทต่าง ๆ ในภาษา C เช่น for loop, while loop, และ do-while loop ได้อย่างถูกต้อง
2. อธิบายหลักการทำงานของแต่ละคำสั่งวนรอบและความแตกต่างระหว่าง for loop, while loop, และ do-while loop
3. เขียนโปรแกรมที่ใช้โครงสร้างการวนรอบเพื่อแก้ปัญหาที่กำหนด เช่น การแสดงผลลัพธ์ตาราง การคูณ หรือการนับจำนวนรอบ
4. วิเคราะห์โครงสร้างการวนรอบในโปรแกรมที่กำหนด และระบุจุดที่ทำให้เกิด infinite loop หรือเงื่อนไขที่ทำให้การวนรอบไม่จบสิ้น
5. ประเมินความเหมาะสมของการเลือกใช้ for loop, while loop, หรือ do-while loop ในการแก้ปัญหาโปรแกรมที่ซับซ้อน
6. ออกแบบและพัฒนาโปรแกรมที่ใช้การวนรอบแบบซ้อนหลายชั้น (Nested Loop) เพื่อจัดการกับข้อมูลในอาร์เรย์หลายมิติหรือการแสดงผลลัพธ์เชิงซ้อน

หัวข้อบรรยาย

บทที่ 5 โครงสร้างการวนรอบ

5.1 การวนรอบ

5.2 คำสั่งวนรอบ for

5.3 คำสั่งวนรอบ while

5.4 คำสั่งวนรอบ do ... while

5.5 การวนรอบแบบซ้อนหลายชั้น (Nested Loop)

กิจกรรมการเรียนการสอน

1. การบรรยายในชั้นเรียน
2. การให้ผู้เรียนทดลองเขียนรหัสต้นฉบับและทดลองรันโปรแกรมระหว่างบรรยาย
3. การให้ผู้เรียนทำใบฝึกปฏิบัติการ
4. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)

สื่อการสอน

1. เอกสารประกอบการสอนแบบภาพเลื่อนหรือภาพนิ่ง
2. ใบฝึกปฏิบัติการ
3. โปรแกรมพัฒนาโปรแกรมภาษาซี เช่น DevC++, VS Code เป็นต้น
4. การเข้าถึงข้อมูลจากอินเทอร์เน็ต

การประเมินผลการเรียนรู้

ผลการเรียนรู้	วิธีการประเมิน	สัปดาห์ที่ประเมิน	สัดส่วนของการประเมินผล
1. คุณธรรมจริยธรรม	1.1 ประเมินจากการเข้าชั้นเรียน การส่งแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการตามกำหนดเวลาที่มอบหมาย 1.2 ประเมินจากพฤติกรรมของนักศึกษาในการปฏิบัติตามกฎระเบียบและข้อบังคับต่างๆ ทั้งในข้อบังคับในชั้นเรียน ข้อบังคับของมหาวิทยาลัย 1.3 ประเมินจากการคัดลอกแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการ 1.4 ประเมินจากการมีส่วนร่วมในห้องเรียนและการปิดคอมพิวเตอร์ เก็บขยะ เก็บเก้าอี้หลังจากใช้งานเสร็จ	ตลอดภาคการศึกษา	10%
2. ความรู้	2.1 แบบฝึกหัดท้ายบท 2.2 การสอบกลางภาคเรียนและปลายภาคเรียน	2-8, 11-16 9, 17	20% 40%
3. ทักษะทางปัญญา	3.1 ประเมินจากใบฝึกปฏิบัติการ	ตลอดภาคการศึกษา	20%
4. ทักษะความสัมพันธ์ระหว่างบุคคลและความรับผิดชอบ	4.1 พิจารณาจากการทำใบฝึกปฏิบัติการ 4.3 ประเมินจากพฤติกรรมในการทำงานกลุ่ม การระดมความคิดเห็น การตอบคำถามในชั้นเรียน	ตลอดภาคการศึกษา	5%
5. ทักษะวิเคราะห์เชิงตัวเลข การสื่อสารและการใช้เทคโนโลยีสารสนเทศ	5.1 การทดสอบหรือการประเมินจากงานที่มอบหมาย 5.2 สังเกตการใช้คอมพิวเตอร์ในการเขียนโปรแกรม การทำใบฝึกปฏิบัติการ และการค้นหาข้อมูลเพิ่มเติมจากอินเทอร์เน็ต	ตลอดภาคการศึกษา	5%
	รวม		100%

อ่าน Slide เวอร์ชันล่าสุดได้ที่ Google Slide

https://s.coecore.com/scp5

การโปรแกรมคอมพิวเตอร์

บทที่ 5 โครงสร้างการวนรอบ (Loop Structure)

ผู้สอน ดร.พงศกร เจริญเนตรกุล

Outline

- คำสั่งวนรอบ for
- คำสั่งวนรอบ while
- คำสั่งวนรอบ do ... while
- การวนรอบแบบซ้อนหลายชั้น (Nested Loop)

การวนรอบ (Loop)

ทำไมต้องวนรอบซ้ำๆ

งานบางอย่าง อาศัยการทำงานซ้ำๆกัน อาจเหมือนกันทุกครั้ง หรือในการทำแต่ละครั้ง มีค่าบางอย่างเปลี่ยนไปอย่างมีรูปแบบ

ตัวอย่าง ต้องการนับเลข 1-200

```
printf("1 \n");
printf("2 \n");
printf("3 \n");
 …
 …
printf("4 \n");
printf("199 \n");
printf("200 \n");
```


การวนรอบ (Loop)

- ในการเขียนโปรแกรมคือกระบวนการที่คำสั่งหรือกลุ่มคำสั่งถูกเรียกใช้ซ้ำหลายครั้งตามเงื่อนไขที่กำหนด
- การวนซ้ำจะทำงานจนกว่าเงื่อนไขนั้นจะเป็นเท็จ
- การวนรอบมีหลายประเภท เช่น for loop, while loop, และ do-while loop ซึ่งมีวิธีการใช้งานแตกต่างกันตามลักษณะของงาน

การวนรอบ (Loop)

ตัวอย่างการใช้วนรอบในชีวิตประจำวัน

1. **การเดินขึ้นบันได**

   เราเดินขึ้นบันไดทีละขั้นโดยทำซ้ำการยกเท้าซ้ายและขวาจนกว่าจะถึงขั้นที่ต้องการ โดยการวนรอบอาจจะมีเงื่อนไขว่า "ถ้ายังไม่ถึงขั้นที่กำหนด ให้ทำซ้ำการเดินขึ้นบันได"

1. **การเตือนความจำ**

   สมมติว่าคุณตั้งนาฬิกาปลุกให้ดังทุก ๆ 30 นาทีเพื่อเตือนคุณให้พักสายตา นาฬิกาปลุกจะทำงานเป็นรอบทุก 30 นาทีจนกว่าจะถึงเวลาหยุดการทำงาน ซึ่งเป็นตัวอย่างของการวนรอบที่ตั้งค่าให้ทำงานซ้ำในช่วงเวลาที่กำหนด

1. **การเติมน้ำมันรถ**

   หากต้องการเติมน้ำมันในถังให้เต็ม คุณจะค่อยๆ เติมน้ำมันทีละลิตรจนกระทั่งถังน้ำมันเต็ม เป็นการวนรอบซ้ำ ๆ ของการเติมน้ำมันตามเงื่อนไขว่า "ถ้าน้ำมันยังไม่เต็ม ให้เติมต่อ"

การวนรอบ (Loop)

ตัวอย่าง ต้องการแสดง ตารางการคูณ

เช่น

```
i=1; printf("2 x %d = %d \n",i,2*i);
i++; printf("2 x %d = %d \n",i,2*i);
i++; printf("2 x %d = %d \n",i,2*i);
i++; printf("2 x %d = %d \n",i,2*i);
    ...
```

Output

2 x 1 = 2

2 x 2 = 4

2 x 3 = 6

2 x 4 = 8

.....

2 x 11 = 22

2 x 12 = 24

การวนรอบ (Loop)

การวนรอบ (loop structure) ในภาษา C มี 3 รูปแบบหลัก ดังนี้

For Loop: ใช้เมื่อรู้จำนวนครั้งที่ต้องการให้ทำซ้ำ	**While Loop**: ใช้เมื่อไม่ทราบจำนวนครั้งที่แน่นอน แต่ทำซ้ำจนกว่าเงื่อนไขจะเป็นเท็จ	**Do-While Loop**: ทำงานอย่างน้อยหนึ่งครั้งก่อนตรวจสอบเงื่อนไข

for (initialization; condition; increment) {
  // คำสั่งที่จะทำซ้ำ
}
	while (condition) {
	  // คำสั่งที่จะทำซ้ำ
	}
		do {
		  // คำสั่งที่จะทำซ้ำ
		} while (condition);

ตัวอย่าง
for (int i = 0; i < 5; i++) {
  printf("รอบที่ %d\n", i);
}
	ตัวอย่าง
	int count = 0;
	while (count < 5) {
	  printf("รอบที่ %d\n", count);
	  count++;
	}
		ตัวอย่าง
		int count = 0;
		do {
		  printf("รอบที่ %d\n", count);
		  count++;
		} while (count < 5);

คำสั่งวนรอบ for

คำสั่งวนรอบ for

for (การกำหนดค่าเริ่มต้นตัวแปร ; เงื่อนไข ; ปรับค่าตัวแปร)

คำสั่ง; หรือ กลุ่มคำสั่ง

จากตัวอย่างก่อนหน้า สามารถเขียนโปรแกรมให้อยู่ในรูปของ for loop ได้ดังนี้

นับ 1 - 200

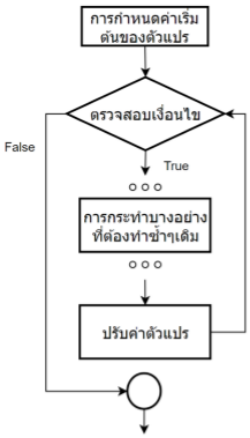
```
int i;
  for (i=0; i<200; i=i+1)  printf("Hello");
```

พิมพ์ตารางการคูณ แม่สอง

```
int i;
for (i=1; i<=12; i++ )
   {   printf("2 x %d = %d \n",i, 2*i);  }
```

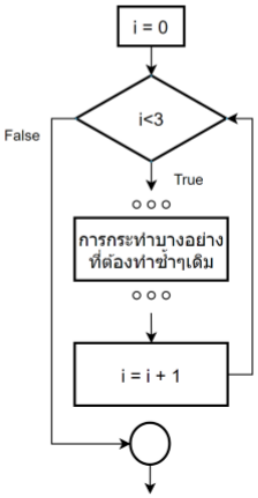

คำสั่งวนรอบ for

Flow chart ของการวนรอบ for



คำสั่งวนรอบ for

```
for( i=0 ; i<3 ; i=i+1)
{
    // ใส่การกระทำบางอย่างที่ต้องทำซ้ำเดิมๆ
}
```



คำสั่งวนรอบ for

- หากเงื่อนไขเป็นจริง loop ก็จะยังคงทำงานต่อไป จนกระทั่งเงื่อนไขเป็นเท็จ
- **การปรับค่าตัวแปร (ลด/เพิ่ม) อาจจะส่งผลให้มีเกิดทำงานไม่รู้จบของ loop ได้ (Infinite Loop)**

ตัวอย่าง Infinity Loop

```
#include<stdio.h>
int main()
{
   int i;
   for (i=0;i<5;i--) // วนรอบไม่รู้จบ แบบ infinity
     printf("* \n");
}
```

คำสั่งวนรอบ for

- มักใช้ ในกรณี รู้จำนวนรอบการทำซ้ำ ที่แน่นอน หรือ มีตัวนับจำนวนรอบ (counter) กำกับ
- ตัว counter ต้องถูกกำหนดค่าเริ่มต้น, และมีการปรับเปลี่ยนค่า
- ควรระวังการกำหนดเงื่อนไขการทำซ้ำ ที่จะต้องกลายเป็นเท็จได้ เพื่อไม่ทำให้เกิดลูปอนันต์ (infinite loop)
- การกำหนด จำนวนรอบ อย่างง่าย

ถ้าต้องการทำ n รอบ

```
int i,n;
for (i=0; i<n; i++)
  { printf("%d \n",i); }
```

```
int i,n;
for (i=1; i<=n; i++)
  { printf("%d \n",i); }
```


คำสั่งวนรอบ for

การนับขึ้น (count up) และนับลง (count down)

```
// count up
int i,n;
for (i=0; i<=n; i++)
  { printf("%d \n",i); }
```

```
// count down
int i,n;
for (i=n; i>=0; i--)
  { printf("%d \n",i); }
```

คำสั่งวนรอบ for

ตัวอย่าง วนรอบเพื่อพิมพ์ค่าตัวเลข

```
#include <stdio.h>

int main() {
    // พิมพ์ตัวเลข 0 ถึง 9
    for (int i = 0; i < 10; i++) {
        printf("%d\n", i);
    }
}
```


คำสั่งวนรอบ for

ตัวอย่าง วนรอบเพื่อนับผลรวมของตัวเลข

```
#include <stdio.h>

int main() {
    int sum = 0;
    // วนรอบเพื่อหาผลรวมของตัวเลข 1 ถึง 10
    for (int i = 1; i <= 10; i++) {
        sum += i; // บวกค่า i เข้าไปใน sum
    }
    printf("ผลรวมคือ: %d\n", sum);
}
```

คำสั่งวนรอบ for

ตัวอย่าง วนรอบเพื่อพิมพ์ค่าในตัวแปรชุด(อาร์เรย์)

```
#include <stdio.h>

int main() {
    int numbers[] = {1, 2, 3, 4, 5};
    int length = sizeof(numbers) / sizeof(numbers[0]);

    // วนรอบเพื่อพิมพ์ค่าของอาร์เรย์
    for (int i = 0; i < length; i++) {
        printf("ตัวเลข: %d\n", numbers[i]);
    }
}
```


คำสั่งวนรอบ while

คำสั่งวนรอบ while

while = ในขณะที่เงื่อนไขเป็นจริงให้ทำคำสั่ง หรือ ทำคำสั่งนั้นจนกว่าเงื่อนไขจะกลายเป็นเท็จ

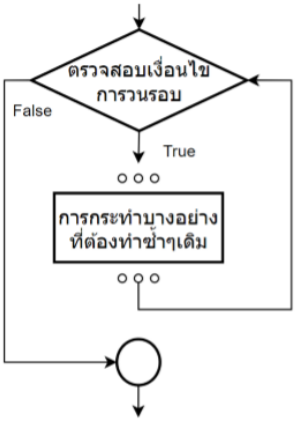
โครงสร้าง while

```
while (เงื่อนไข)
    คำสั่ง; หรือ กลุ่มคำสั่ง{}
```

หากเงื่อนไขเป็นจริง โปรแกรมจะทำงานตามคำสั่งซ้ำ จนกระทั่งเงื่อนไขเป็นเท็จ

คำสั่งวนรอบ while

Flow chart ของการวนรอบ while



คำสั่งวนรอบ while

for VS while

// FOR LOOP

```
#include<stdio.h>
int main()
{
        int i;
        for (i=0;i<5;i++)
        printf("* \n");
}
```

// WHILE LOOP

```
#include<stdio.h>
int main()
{
  int i=0;
  while (i<5)
    {
            printf("* \n");
            i++;
    }
}
```


คำสั่งวนรอบ while

เปรียบเทียบโครงสร้าง FOR และ WHILE

For Loop: ใช้เมื่อรู้จำนวนครั้งที่ต้องการให้ทำซ้ำ	While Loop: ใช้เมื่อไม่ทราบจำนวนครั้งที่แน่นอน แต่ทำซ้ำจนกว่าเงื่อนไขจะเป็นเท็จ

for (initialization; condition; increment) {
  // คำสั่งที่จะทำซ้ำ
}
	while (condition) {
	  // คำสั่งที่จะทำซ้ำ
	}

ตัวอย่าง
for (int i = 0; i < 5; i++) {
  printf("รอบที่ %d\n", i);
}
	ตัวอย่าง
	int count = 0;
	while (count < 5) {
	  printf("รอบที่ %d\n", count);
	  count++;
	}

ตัวอย่างวนรอบ while

วนรอบตรวจสอบเงื่อนไขเพื่อรับค่าข้อมูลที่ต้องการ

```
int num = -1;
// วนรอบจนกว่าผู้ใช้จะป้อนเลขที่เป็นจำนวนบวก
while (num < 0) {
    printf("กรุณากรอกจำนวนบวก: ");
    scanf("%d", &num);
}
printf("คุณกรอกเลขเรียบร้อยแล้ว!\n");
```


ตัวอย่างวนรอบ while

วนรอบตรวจสอบเงื่อนไขเพื่อรับค่าข้อมูลที่ต้องการ

```
int num = 0;
// วนรอบจนกว่าผู้ใช้จะป้อนเลขคู่
while (num % 2 != 0) {
    printf("กรุณากรอกเลขคู่: ");
    scanf("%d", &num);
}
printf("คุณกรอกเลขคู่เรียบร้อยแล้ว!\n");
```

ตัวอย่างวนรอบ while

วนรอบตรวจสอบเงื่อนไขเพื่อรับค่าข้อมูลที่ต้องการ

```
int menu = 0;
// วนรอบจนกว่าผู้ใช้จะเมนูที่เป็นเลข 1-4
while (menu < 1 || menu >4) {
    printf("กรุณาเลือกเมนู(1-4): ");
    scanf("%d", &num);
}
```


คำสั่งวนรอบ do ... while

คำสั่งวนรอบ do ... while

- โครงสร้างแบบการทำซ้ำ while และ for จะต้องมีการตรวจสอบค่าของเงื่อนไขก่อนว่าเป็นจริงหรือเท็จ ก่อนที่จะทำคำสั่ง (กลุ่มคำสั่ง) ภายในรอบ
- **แต่โครงสร้าง do-while จะทำคำสั่งภายในรอบก่อน 1 ครั้งเสมอ แล้วจึงตรวจสอบเงื่อนไขว่าจะวนรอบต่อหรือเปล่า**

คำสั่งวนรอบ do ... while

```
// โครงสร้าง do ... while();
do
{
  กลุ่มคำสั่ง;
}
while ( เงื่อนไข ) ;
```

*** ต้องมี ; เพื่อจบคำสั่ง do-while

คำสั่งวนรอบ do ... while

```
#include<stdio.h>
int main()
{
        int i=6;
        do
        {
            printf("* \n");
            i++;
        }
        while (i<5);
}
```

ถ้าต้องการให้ทำ 6 รอบ ต้องแก้ไข อย่างไร

```
#include<stdio.h>
int main()
{
        int i=6;
        do
        {
            printf("* \n");
            i++;
        }
        while (i<12);
}
```

```
#include<stdio.h>
int main()
{
        int i=-1;
        do
        {
            printf("* \n");
            i++;
        }
        while (i<5);
}
```


คำสั่งวนรอบ do ... while

โปรแกรมรอรับข้อมูลจนกว่าจะได้ค่าในขอบเขตโดยใช้ do-while loop

```
char grade ; Int ok;
do{
        ok=1;
        printf("Enter a grade:");
        scanf("%c",&grade);
        If(grade<'A'|| grade>'E')
       ok=0;
        If (lok)
        { printf("Grade must be A-E\n");
            printf("Please enter again.");
        }
} while(lok);
printf("%c Is OK\n",grade);
```

โปรแกรมนี้ทำหน้าที่รับค่าตัวอักษรที่เป็นเกรดจากผู้ใช้ โดยตรวจสอบว่าเกรดที่รับมาอยู่ในช่วง **'A' ถึง 'E'** หรือไม่ ถ้าไม่ใช่จะให้ผู้ใช้กรอกใหม่จนกว่าจะได้เกรดที่ถูกต้อง

คำอธิบาย

1. โปรแกรมเริ่มต้นด้วยการตั้งค่า ok = 1 เพื่อระบุว่าเงื่อนไขการกรอกถูกต้อง
2. ใช้ลูป do-while ในการวนซ้ำเพื่อให้ผู้ใช้กรอกค่าเกรด โดยการรับค่าเกรดเก็บไว้ในตัวแปร grade
3. ตรวจสอบว่าเกรดอยู่ในช่วง **'A' ถึง 'E'** หรือไม่ ถ้าไม่ใช่ (grade < 'A' || grade > 'E') จะตั้งค่า ok = 0 (ผิดเงื่อนไข)
4. ถ้าเกรดไม่ถูกต้อง (!ok หรือ ok = 0) จะแสดงข้อความให้กรอกใหม่ และลูปจะวนซ้ำจนกว่าจะได้เกรดที่ถูกต้อง
5. เมื่อได้เกรดที่ถูกต้องแล้ว ลูปจะจบ และแสดงผลว่าเกรดนั้น "OK"

การวนรอบแบบซ้อนหลายชั้น (Nested Loop)

การวนรอบแบบซ้อนหลายชั้น (Nested Loop)

การวนรอบแบบซ้อนหลายชั้น (Nested Loop) คือการใช้วนรอบหนึ่งซ้อนอยู่ภายในอีกวนรอบหนึ่ง โดยเมื่อวนรอบภายนอกทำงานแต่ละครั้ง วนรอบภายในจะทำงานซ้ำจนครบทุกครั้งที่กำหนดไว้ การใช้ลูปซ้อนมักพบในงานที่มีการทำซ้ำหลายมิติ เช่น การจัดการข้อมูลในอาร์เรย์ 2 มิติ การแสดงตารางข้อมูล หรือการทำงานที่มีขั้นตอนหลายระดับ

```
for (initialization1; condition1; increment1) {
    // ลูปภายนอก
    for (initialization2; condition2; increment2) {
        // ลูปภายใน
        // คำสั่งที่จะทำซ้ำ
    }
}
```

การวนรอบแบบซ้อนหลายชั้น (Nested Loop)

ตัวอย่าง วนรอบเพื่อแสดงตารางหมากรุก (ลูปซ้อนกัน)

```
#include <stdio.h>

int main() {
    // วนรอบแสดงตารางหมากรุกขนาด 8x8
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            if ((i + j) % 2 == 0)
                printf("■");
            else
                printf(" ");
        }
        printf("\n");
    }
}
```


การวนรอบแบบซ้อนหลายชั้น (Nested Loop)

ตัวอย่าง การแสดงค่าในอาร์เรย์ 2 มิติ

```
#include <stdio.h>

int main() {
    int matrix[3][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };

    // วนรอบเพื่อพิมพ์ค่าทั้งหมดในอาร์เรย์ 2 มิติ
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}
```

การวนรอบแบบซ้อนหลายชั้น (Nested Loop)

จงหาผลลัพธ์ของโปรแกรมด้านล่าง

```
#include<stdio.h>
#define num 5
int main(){
 int i,j;
 for (i=0;i<num;i++){
   for(j=0; j<=i; j++){
              printf("*");
   }
               printf("\n");
 }
}
```

Output

```
*
**
***
****
*****
```


การวนรอบแบบซ้อนหลายชั้น (Nested Loop)

จงหาผลลัพธ์ของโปรแกรมด้านล่าง

```
#include<stdio.h>
#define num 5
int main(){
   int i,j;
   for (i=num;i>0;i--){
              for(j=0;j<i;j++){
              printf("*");
              }
              printf("\n");
   }
}
```

Output

```
*****
****
***
**
*
```

เอกสารอ้างอิง

คณาจารย์ผู้สอนวิชา Introduction to Computer Programming, เอกสารประกอบการบรรยายวิชา 241-101 Introduction to Computer Programming, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยสงขลานครินทร์, 2551

นราธร สังข์ประเสริฐ, เอกสารประกอบการบรรยายวิชา 04-511-101 การโปรแกรมคอมพิวเตอร์, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย , 2558

แหล่งศึกษาเพิ่มเติม

ผศ.นราธร สังข์ประเสริฐ. (2563) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-3945-2

โอภาส เอี่ยมสิริวงศ์. (2559) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-2740-4

รศ.ธีรวัฒน์ ประกอบผล. (2562) คู่มือการเขียนโปรแกรมภาษา C ฉบับสมบูรณ์, ซิมพลิฟาย, ISBN: 978-616-26-24896

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase1] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=aDBmXMF9jOU

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase2] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=wslM0Aqepxs

Mooc การเขียนโปรแกรมคอมพิวเตอร์พื้นฐาน ด้วยภาษาซี 1 | Fundamentals of the C Programming Language https://lms.thaimooc.org/courses/course-v1:KMUTT+KMUTT013+2019/about

Mooc ทักษะการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี | Computer programming skills with C language https://lms.thaimooc.org/courses/course-v1:SU+SU011+2021/about

Coursera: Programming in C, ผู้สอน: Charles Russell Severance https://www.coursera.org/learn/programming-c

Coursera: ความเชี่ยวชาญพิเศษ การเขียนโปรแกรม C เบื้องต้น, ผู้สอน: Andrew D. Hilton et al. https://www.coursera.org/specializations/c-programming

Playlist Youtube บันทึกการสอนในห้องเรียน, วิชาการโปรแกรมคอมพิวเตอร์, ดร.พงศกร เจริญเนตรกุล 2566 https://www.youtube.com/watch?v=ewEfmiXaRi0&list=PL322g2itJn4mAE1utyOzF-C0SP0BRkej3&pp=gAQBiAQB

แบบฝึกหัดท้ายบท

1. คำสั่งวนรอบ (Loop) คืออะไร
2. Loop ประเภทใดบ้างที่มีในภาษา C
3. Loop ประเภทใดที่เหมาะกับการใช้งานเมื่อรู้จำนวนรอบที่แน่นอน
4. while loop เหมาะกับสถานการณ์ใด
5. โครงสร้างของ for loop ในภาษา C คืออะไร
6. โครงสร้างของ while loop ในภาษา C คืออะไร
7. โครงสร้างของ do-while loop ในภาษา C คืออะไร
8. do-while loop ต่างจาก while loop อย่างไร
9. คำสั่งต่อไปนี้เป็น loop ประเภทใด
    a. `for (int i = 0; i < 10; i++) {`
    b. `printf("%d\n", i);`
    c. `}`
10. เมื่อใดที่เราควรใช้ do-while loop
11. for loop นี้จะทำงานกี่รอบ
    a. `for (int i = 0; i < 5; i++) {`
    b. `printf("%d\n", i);`
    c. `}`
12. Infinite loop คืออะไร
13. จาก code ต่อไปนี้ จะแก้การเกิด infinite loop ได้อย่างไรบ้าง
    a. `for (int i = 0; i > -1; i++) {`
    b. `printf("%d\n", i);`
    c. `}`
14. ความแตกต่างระหว่างการนับขึ้น (count up) และการนับลง (count down) ใน for loop คืออะไร
15. คำสั่งใดจะใช้ใน for loop หากต้องการพิมพ์ตัวเลขจาก 10 ถึง 1
16. Loop แบบใดเหมาะกับการรับค่าจากผู้ใช้จนกว่าจะได้รับข้อมูลที่ถูกต้อง
17. การวนรอบแบบซ้อนหลายชั้น (Nested Loop) คืออะไร
18. Nested loop ใช้ในกรณีใดบ้าง
19. คำสั่งนี้จะพิมพ์อะไรออกมา
    a. `for (int i = 0; i < 3; i++) {`
    b. `for (int j = 0; j < 2; j++) {`
        i. `printf("i=%d, j=%d\n", i, j);`

c. }
d. }

20. โปรแกรมนี้มี output เป็นอะไร

a. int num = 5;
b. for (int i = num; i > 0; i--) {
c. for (int j = 0; j < i; j++) {
    i. printf("*");
d. }
e. printf("\n");
f. }

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน.....

ใบฝึกปฏิบัติการที่ 5.1

เรื่อง ใช้ For Loop แสดงสูตรคูณแม่ 3

วัตถุประสงค์

1. สามารถใช้โครงสร้าง for ในการวนรอบอย่างถูกต้อง
2. สามารถแสดงผลสูตรคูณด้วยตัวแปร index ของ for loopได้

ทบทวนความรู้

คำสั่งวนรอบ for

```
for (การกำหนดค่าเริ่มต้นตัวแปร ; เงื่อนไข ; ปรับค่าตัวแปร)
        คำสั่ง; หรือ กลุ่มคำสั่ง
```

จากตัวอย่างก่อนหน้า สามารถเขียนโปรแกรมให้อยู่ในรูปของ for loop ได้ดังนี้

นับ 1 - 200

```
int i;
  for (i=0; i<200; i=i+1)  printf("Hello");
```

พิมพ์ตารางการคูณ แม่สอง

```
 int i;
 for (i=1; i<=12; i++ )
   {   printf("2 x %d = %d \n",i, 2*i);  }
```

Computer Programming 9

คำสั่งวนรอบ for

ตัวอย่าง วนรอบเพื่อพิมพ์ค่าตัวเลข

```
#include <stdio.h>

int main() {
    // พิมพ์ตัวเลข 0 ถึง 9
    for (int i = 0; i < 10; i++) {
        printf("%d\n", i);
    }
}
```

Computer Programming 15

คำสั่ง

จงเขียนโปรแกรมโดยใช้ For Loop แสดงสูตรคูณแม่ 3

3 x 1 = 3

3 x 2 = 6

3 x 3 = 9

3 x 4 = 12

3 x 5 = 15

3 x 6 = 18

3 x 7 = 21

3 x 8 = 24

3 x 9 = 27

3 x 10 = 30

3 x 11 = 33

3 x 12 = 36

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน.....

ใบฝึกปฏิบัติการที่ 5.2

เรื่อง ใช้ For Loop แสดงสูตรคูณแม่ 3 ถอยหลัง

วัตถุประสงค์

1. สามารถใช้โครงสร้าง for ในการวนรอบแบบนับลงอย่างถูกต้อง
2. สามารถแสดงผลสูตรคูณด้วยตัวแปร index ของ for loopได้

ทบทวนความรู้

คำสั่งวนรอบ for

ตัวอย่าง วนรอบเพื่อพิมพ์ค่าตัวเลข

```
#include <stdio.h>

int main() {
   // พิมพ์ตัวเลข 0 ถึง 9
   for (int i = 0; i < 10; i++) {
      printf("%d\n", i);
   }
}
```

Computer Programming 15

คำสั่งวนรอบ for

การนับขึ้น (count up) และนับลง (count down)

```
// count up
int i,n;
for (i=0; i<=n; i++)
  { printf("%d \n",i); }
```

```
// count down
int i,n;
for (i=n; i>=0; i--)
  { printf("%d \n",i); }
```

Computer Programming 14

คำสั่ง

จงเขียนโปรแกรมโดยใช้ For Loop แสดงสูตรคูณแม่ 3 แบบถอยหลัง โดยการลดค่าตัวแปร index ที่ใช้ในการวนรอบของ for

3 x 12 = 36
3 x 11 = 33
3 x 10 = 30
3 x 9 = 27
3 x 8 = 24
3 x 7 = 21
3 x 6 = 18
3 x 5 = 15
3 x 4 = 12
3 x 3 = 9
3 x 2 = 6
3 x 1 = 3

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 5.3

เรื่อง สูตรคูณแม่ฉัน

วัตถุประสงค์

1. สามารถใช้โครงสร้าง for ในการวนรอบอย่างถูกต้อง
2. สามารถแสดงผลสูตรคูณด้วยตัวแปร index ของ for loopได้
3. สามารถประยุกต์คำสั่ง scanf ในการเลือกแม่สูตรคูณ ที่ต้องนำมาใช้กำหนดและคำนวณในการวนรอบของ for

ทบทวนความรู้

คำสั่งวนรอบ for

ตัวอย่าง วนรอบเพื่อพิมพ์ค่าตัวเลข

```
#include <stdio.h>

int main() {
    // พิมพ์ตัวเลข 0 ถึง 9
    for (int i = 0; i < 10; i++) {
        printf("%d\n", i);
    }
}
```

คำสั่ง

จงเขียนโปรแกรมโดยใช้ For Loop แสดงสูตรคูณ โดยต้องรับแม่สูตรคูณโดยการกดคีย์บอร์ดจากผู้ใช้

Enter multiply formula : 4 (ใช้ scanf รับค่าแม่สูตรคูณ)

4 x 1 = 4

4 x 2 = 8

4 x 3 = 12

.....

4 x 12 = 48

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 5.4

เรื่อง สูตรคูณแม่ฉัน กำหนด จุดเริ่มต้นและจุดจบ

วัตถุประสงค์

1. สามารถใช้โครงสร้าง for ในการวนรอบอย่างถูกต้อง
2. สามารถแสดงผลสูตรคูณด้วยตัวแปร index การกำหนดค่าเริ่มต้น และเงื่อนไขจบโดยตัวแปรใน for loop ได้
3. สามารถประยุกต์คำสั่ง scanf ในการเลือกแม่สูตรคูณ ที่ต้องนำมาใช้กำหนดและคำนวณในการวนรอบของ for

ทบทวนความรู้

คำสั่งวนรอบ for

ตัวอย่าง วนรอบเพื่อพิมพ์ค่าตัวเลข

```
#include <stdio.h>

int main() {
    // พิมพ์ตัวเลข 0 ถึง 9
    for (int i = 0; i < 10; i++) {
        printf("%d\n", i);
    }
}
```

Computer Programming 15

คำสั่ง

จงเขียนโปรแกรมโดยใช้ For Loop แสดงสูตรคูณ โดยต้องรับแม่สูตรคูณโดยการกดคีย์บอร์ดจากผู้ใช้
**โดยสามารถกำหนดจุดเริ่มต้นและจุดสิ้นสุดของสูตรคูณโดยการกดคีย์บอร์ดจากผู้ใช้ได้
Enter multiply formula : 4 (ใช้ scanf รับค่าแม่สูตรคูณ)
Enter start number : 2 (ใช้ scanf รับตัวเริ่มต้น)
Enter end number : 6 (ใช้ scanf รับตัวสุดท้าย)
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน.....

ใบฝึกปฏิบัติการที่ 5.5

เรื่อง แสดง a-z

วัตถุประสงค์

1. สามารถใช้โครงสร้าง for ในการวนรอบอย่างถูกต้อง
2. สามารถแสดงผล a-z ได้

ทบทวนความรู้

คำสั่งวนรอบ for

ตัวอย่าง วนรอบเพื่อพิมพ์ค่าตัวเลข

```
#include <stdio.h>

int main() {
    // พิมพ์ตัวเลข 0 ถึง 9
    for (int i = 0; i < 10; i++) {
        printf("%d\n", i);
    }
}
```

คำสั่ง

จงใช้ for loop แสดง a-z โดยบวกเพิ่มค่าจากตัวแปร char

DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII
1	☺	32	space	64	@	96	`	128	Ç	160	á	192	└	224	Ó
2	☻	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
3	♥	34	"	66	B	98	b	130	è	162	ó	194	┬	226	Ô
4	♦	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
5	♣	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
6	♠	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
7	•	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
8	◘	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
9	○	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
10	◙	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
11	♂	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
12	♀	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
13	♪	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
14	♫	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
15	☼	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
16	►	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
17	◄	48	0	80	P	112	p	144	È	176	░	208	ð	240	-
18	↕	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
19	‼	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
20	¶	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
21	§	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
22	▬	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
23	↨	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
24	↑	55	7	87	W	119	t	151	ù	183	À	215	Î	247	¸
25	↓	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
26	→	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
27	←	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
28	∟	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
29	↔	60	<	92	\	124	|	156	£	188	╝	220	▄	252	³
30	▲	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
31	▼	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
		63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	space

```
#include <stdio.h>
int main()
{
	char alphabet ='a';
	int i;
	for(____เติมโครงสร้างfor____)
	{
		printf("%c \n",alphabet);
	}
}
```

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน.....

ใบฝึกปฏิบัติการที่ 5.6

เรื่อง ดาวกระชากใจ

วัตถุประสงค์

1. สามารถใช้โครงสร้าง for แบบซ้อนหลายชั้นวนรอบได้อย่างถูกต้อง
2. สามารถแสดงผลตามสั่งได้ถูกต้อง

ทบทวนความรู้

การวนรอบแบบซ้อนหลายชั้น (Nested Loop)

การวนรอบแบบซ้อนหลายชั้น (Nested Loop) คือการใช้วนรอบหนึ่งซ้อนอยู่ภายในอีกวนรอบหนึ่ง โดยเมื่อวนรอบภายนอกทำงานแต่ละครั้ง วนรอบภายในจะทำงานซ้ำจนครบทุกครั้งที่กำหนดไว้ การใช้ลูปซ้อนมักพบในงานที่มีการทำซ้ำหลายมิติ เช่น การจัดการข้อมูลในอาร์เรย์ 2 มิติ การแสดงตารางข้อมูล หรือการทำงานที่มีขั้นตอนหลายระดับ

```
for (initialization1; condition1; increment1) {
    // ลูปภายนอก
    for (initialization2; condition2; increment2) {
        // ลูปภายใน
        // คำสั่งที่จะทำซ้ำ
    }
}
```

Computer Programming 32

การวนรอบแบบซ้อนหลายชั้น (Nested Loop)

ตัวอย่าง วนรอบเพื่อแสดงตารางหมากรุก (ลูปซ้อนกัน)

```
#include <stdio.h>

int main() {
   // วนรอบแสดงตารางหมากรุกขนาด 8x8
   for (int i = 0; i < 8; i++) {
      for (int j = 0; j < 8; j++) {
         if ((i + j) % 2 == 0)
            printf("■");
         else
            printf(" ");
      }
      printf("\n");
   }
}
```


การวนรอบแบบซ้อนหลายชั้น (Nested Loop)

ตัวอย่าง การแสดงค่าในอาร์เรย์ 2 มิติ

```
#include <stdio.h>

int main() {
   int matrix[3][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };

   // วนรอบเพื่อพิมพ์ค่าทั้งหมดในอาร์เรย์ 2 มิติ
   for (int i = 0; i < 3; i++) {
      for (int j = 0; j < 3; j++) {
         printf("%d ", matrix[i][j]);
      }
      printf("\n");
   }
}
```


คำสั่ง

ให้นศ.เขียนโปรแกรมรับค่าจำนวนบรรทัดจากผู้ใช้

บรรทัดที่ 1 แสดง * จำนวน 1ตัว

บรรทัดที่ 2 แสดง * จำนวน 2ตัว

บรรทัดที่ 3 แสดง * จำนวน 3ตัว

บรรทัดที่ 4 แสดง * จำนวน 4ตัว

บรรทัดที่ 5 แสดง * จำนวน 5ตัว

บรรทัดที่ 6 แสดง * จำนวน 6ตัว

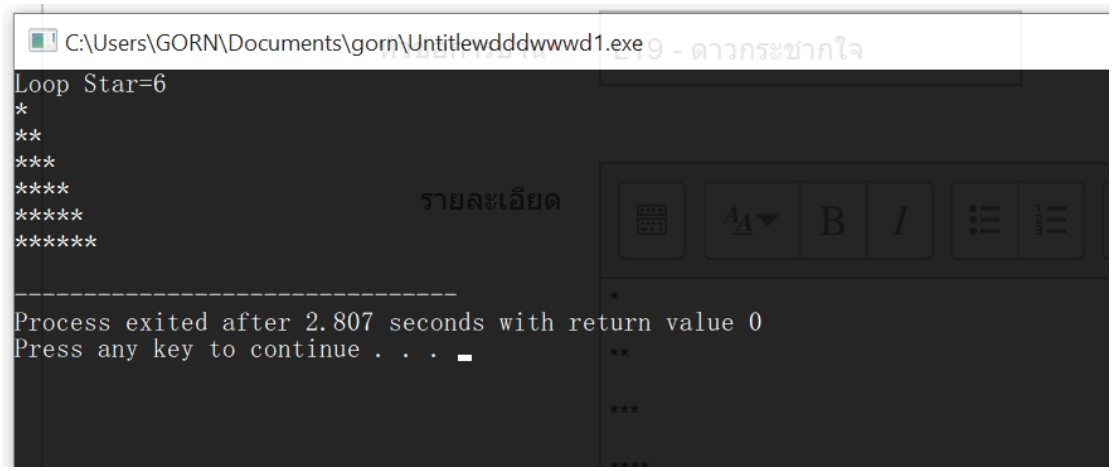
...

บรรทัดที่ N แสดง * จำนวน Nตัว

ตัวอย่าง

Loop Star=6

```
*
**
***
****
*****
******
```



ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 5.7

เรื่อง รอบไหนเหนื่อยก็พักนะ ไม่ต้องทำ

วัตถุประสงค์

1. สามารถใช้โครงสร้าง for วนรอบได้อย่างถูกต้อง
2. สามารถแสดงผลตามสั่งได้ถูกต้อง
3. สามารถใช้คำสั่งควบคุม break และ continue ร่วมกับ if ได้อย่างถูกต้อง

ทบทวนความรู้

3 โครงสร้างการกระโดด (Jump Structure)

break; ใช้เพื่อออกจากลูปหรือ switch-case

continue; ใช้เพื่อข้ามคำสั่งที่เหลือในลูป แล้ววนซ้ำรอบถัดไป

goto label_xx; ใช้กระโดดไปยังตำแหน่งที่กำหนด (ไม่นิยมใช้เพราะอาจทำให้โค้ดอ่านยาก)

Computer Programming 10

คำสั่ง

จงเขียนโปรแกรมคูณเลขตั้งแต่ 1 ถึง 10 โดยใช้การวนรอบ โดยยกเว้นไม่คูณกับเลข 4 7 และ 9

1*2*3*5*6*8*9*10

คำใบ้ continue; เจอเมื่อไหร่จะไม่ทำในloopรอบนั้นๆ

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน.....

ใบฝึกปฏิบัติการที่ 5.8

เรื่อง หยุดตรงนี้ที่เธอ ไม่ไปไกลแล้วใจ จะหยุดทั้งใจไว้ที่เธอ

วัตถุประสงค์

1. สามารถใช้โครงสร้าง for วนรอบได้อย่างถูกต้อง
2. สามารถแสดงผลตามสั่งได้ถูกต้อง
3. สามารถใช้คำสั่งควบคุม break และ continue ร่วมกับ if ได้อย่างถูกต้อง

ทบทวนความรู้

3 โครงสร้างการกระโดด (Jump Structure)

break; ใช้เพื่อออกจากลูปหรือ switch-case

continue; ใช้เพื่อข้ามคำสั่งที่เหลือในลูป แล้ววนซ้ำรอบถัดไป

goto label_xx; ใช้กระโดดไปยังตำแหน่งที่กำหนด (ไม่นิยมใช้เพราะอาจทำให้โค้ดอ่านยาก)

Computer Programming 10

คำสั่ง

จงหาว่า 1 * 2 * 3 * 4 * 5 ..... ผลคูณสะสมไปถึงเลขใดแล้วค่ามากกว่า 36,000,000 และผลคูณได้ค่าเท่าไหร่ และบอกด้วยนะว่าคูณไปถึงเลขเท่าไหร่

**ต้องใช้เงื่อนเพื่อตรวจสอบ และ break; เพื่อหยุดและออกจากLoop

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน.....

ใบฝึกปฏิบัติการที่ 5.9

เรื่อง while แม่ 2

วัตถุประสงค์

1. สามารถใช้โครงสร้าง while วนรอบแทน for ได้อย่างถูกต้อง
2. สามารถแสดงผลสูตรคูณด้วยตัวแปร index ของ while loopได้

ทบทวนความรู้

คำสั่งวนรอบ while

เปรียบเทียบโครงสร้าง FOR และ WHILE

For Loop: ใช้เมื่อรู้จำนวนครั้งที่ต้องการให้ทำซ้ำ	**While Loop**: ใช้เมื่อไม่ทราบจำนวนครั้งที่แน่นอน แต่ทำซ้ำจนกว่าเงื่อนไขจะเป็นเท็จ

for (initialization; condition; increment) {
  // คำสั่งที่จะทำซ้ำ
}
	while (condition) {
	  // คำสั่งที่จะทำซ้ำ
	}

ตัวอย่าง
for (int i = 0; i < 5; i++) {
  printf("รอบที่ %d\n", i);
}
	ตัวอย่าง
	int count = 0;
	while (count < 5) {
	  printf("รอบที่ %d\n", count);
	  count++;
	}

Computer Programming 22

คำสั่ง

จงเขียน while loop เพื่อแสดงสูตรคูณแม่ 2

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 5.10

เรื่อง ทายเลขกันเถอะ

วัตถุประสงค์

1. สามารถใช้โครงสร้าง while วนรอบได้อย่างถูกต้อง
2. สามารถกำหนดเงื่อนไขการหยุดวน while หรือใช้การ break ได้อย่างถูกต้อง
3. สามารถค้นคว้า random function ด้วยตัวเองได้

ทบทวนความรู้

ตัวอย่างวนรอบ while

วนรอบตรวจสอบเงื่อนไขเพื่อรับค่าข้อมูลที่ต้องการ

```
int menu = 0;
// วนรอบจนกว่าผู้ใช้จะเมนูที่เป็นเลข 1-4
while (menu < 1 || menu >4) {
    printf("กรุณาเลือกเมนู(1-4): ");
    scanf("%d", &num);
}
```

Computer Programming 25

คำสั่ง

ให้นศ. สุ่มเลขมาตัวหนึ่ง จากนั้นสร้าง LOOP เพื่อรับค่าให้ผู้ใช้ทายเลขที่เราได้สุ่มขึ้นมา ตัวอย่างโปรแกรมภาษาPythonข้างล่างนี้

โปรแกรมทายเลข

ให้นศ. สุ่มเลขมาตัวหนึ่ง จากนั้นสร้าง LOOP เพื่อรับค่าให้ผู้ใช้ทายเลขที่เราได้สุ่มขึ้นมา

Import Random Lib

```
import random
```

Random a Number

```
[15] n = random.randint(1,10)
     n
```

```
3
```

create infinity loop, input number, check value

```
[14] while(True):
       guessed_number = int(input("Guess a Number(1-10) :"))
       if guessed_number==n :
         print(f"Correct! number is {n}" )
         break
       elif guessed_number<n :
         print(f"Number is too low" )
       elif guessed_number>n :
         print(f"Number is too high" )
```

```
Guess a Number(1-10) :10
Number is too high
Guess a Number(1-10) :9
Number is too high
Guess a Number(1-10) :1
Number is too low
Guess a Number(1-10) :2
Number is too low
Guess a Number(1-10) :4
Number is too high
Guess a Number(1-10) :5
Number is too high
Guess a Number(1-10) :3
Correct! number is 3
```

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 5.11

เรื่อง while โจทย์เด็กๆ

วัตถุประสงค์

1. สามารถใช้โครงสร้าง while วนรอบได้อย่างถูกต้อง
2. สามารถกำหนดเงื่อนไขการหยุดวน while หรือใช้การ break ได้อย่างถูกต้อง

ทบทวนความรู้

ตัวอย่างวนรอบ while

วนรอบตรวจสอบเงื่อนไขเพื่อรับค่าข้อมูลที่ต้องการ

```
int menu = 0;
// วนรอบจนกว่าผู้ใช้จะเมนูที่เป็นเลข 1-4
while (menu < 1 || menu >4) {
    printf("กรุณาเลือกเมนู(1-4): ");
    scanf("%d", &num);
}
```

Computer Programming 25

คำสั่ง

จงเขียนโปรแกรม ถามคำถามและรับคำตอบไปจนกว่าจะได้คำตอบที่ถูกต้อง โดยโปรแกรมแสดงผลดังตัวอย่างต่อไปนี้ **ด้วย while**

Problem: 2*3=?

ANS=7 //<-- เลข 7 เป็นค่าที่รับจากผู้ใช้

Wrong !!!

Problem: 2*3=?

ANS=9 //<-- เลข 9 เป็นค่าที่รับจากผู้ใช้
Wrong !!!

Problem: 2*3=?
ANS=6 //<-- เลข 6 เป็นค่าที่รับจากผู้ใช้
Yeah !!! \\(^o^)//

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

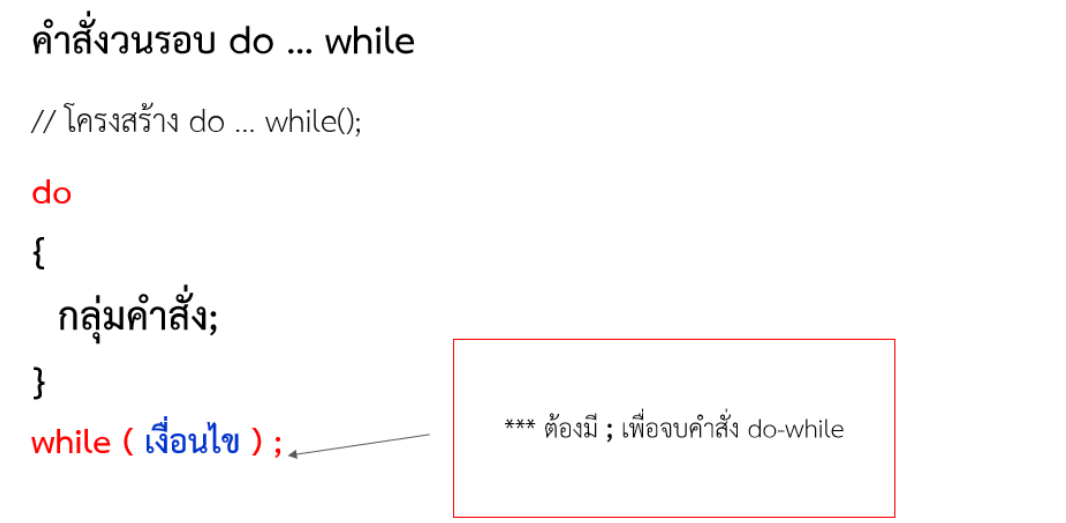
ใบฝึกปฏิบัติการที่ 5.12

เรื่อง while โจทย์เด็กๆ

วัตถุประสงค์

1. สามารถใช้โครงสร้าง do...while วนรอบได้อย่างถูกต้อง
2. สามารถแปลง while เป็น do...while ได้

ทบทวนความรู้



คำสั่ง

จงเขียนโปรแกรม ถามคำถามและรับคำตอบไปจนกว่าจะได้คำตอบที่ถูกต้อง โดยโปรแกรมแสดงผลดังตัวอย่างต่อไปนี้ **ด้วย do...while**

Problem: 2*3=?

ANS=7 //<-- เลข 7 เป็นค่าที่รับจากผู้ใช้

Wrong !!!

Problem: 2*3=?

ANS=9 //<-- เลข 9 เป็นค่าที่รับจากผู้ใช้
Wrong !!!

Problem: 2*3=?
ANS=6 //<-- เลข 6 เป็นค่าที่รับจากผู้ใช้
Yeah !!! \\(^o^)//

แผนการสอนเรื่อง ฟังก์ชัน

สัปดาห์ที่ 10-12

จำนวนชั่วโมงที่ใช้สอน 15 ชั่วโมง

วัตถุประสงค์ เพื่อให้นักศึกษาสามารถ

1. บอกความหมายและประเภทของฟังก์ชันในภาษา C ได้
2. อธิบายประโยชน์ของการใช้ฟังก์ชันในการเขียนโปรแกรมได้
3. เขียนฟังก์ชันที่กำหนดเองในภาษา C เพื่อแก้ไขปัญหาเฉพาะทางโปรแกรมได้
4. วิเคราะห์ความแตกต่างระหว่างตัวแปรภายใน (Local Variable) และตัวแปรภายนอก (Global Variable) ในการเขียนโปรแกรมได้
5. ใช้งานฟังก์ชันแบบเรียกตัวเอง (Recursive Function) และฟังก์ชันทั่วไปเพื่อแก้ไขปัญหาต่าง ๆ ได้อย่างเหมาะสม
6. ออกแบบและสร้างโปรแกรมที่ใช้ฟังก์ชันในการแก้ไขปัญหาซับซ้อนได้

หัวข้อบรรยาย

บทที่ 6 ฟังก์ชัน

6.1 ความหมายและประโยชน์ของฟังก์ชัน

6.2 ฟังก์ชันในภาษา C

6.3 การสร้างฟังก์ชันโดยผู้เขียนโปรแกรม

6.4 การเรียกใช้งานฟังก์ชัน

6.5 ฟังก์ชันแบบเรียกตัวเอง

6.6 ขอบเขตของตัวแปร

กิจกรรมการเรียนการสอน

1. การบรรยายในชั้นเรียน
2. การให้ผู้เรียนทดลองเขียนรหัสต้นฉบับและทดลองรันโปรแกรมระหว่างบรรยาย
3. การให้ผู้เรียนทำใบฝึกปฏิบัติการ
4. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)

สื่อการสอน

1. เอกสารประกอบการสอนแบบภาพเลื่อนหรือภาพนิ่ง
2. ใบฝึกปฏิบัติการ
3. โปรแกรมพัฒนาโปรแกรมภาษาซี เช่น DevC++, VS Code เป็นต้น
4. การเข้าถึงข้อมูลจากอินเทอร์เน็ต

การประเมินผลการเรียนรู้

ผลการเรียนรู้	วิธีการประเมิน	สัปดาห์ที่ประเมิน	สัดส่วนของการประเมินผล
1. คุณธรรมจริยธรรม	1.1 ประเมินจากการเข้าชั้นเรียน การส่งแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการตามกำหนดเวลาที่มอบหมาย 1.2 ประเมินจากพฤติกรรมของนักศึกษาในการปฏิบัติตามกฎระเบียบและข้อบังคับต่าง ๆ ทั้งในข้อบังคับในชั้นเรียน ข้อบังคับของมหาวิทยาลัย 1.3 ประเมินจากการคัดลอกแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการ 1.4 ประเมินจากการมีส่วนร่วมในห้องเรียนและการปิดคอมพิวเตอร์ เก็บขยะ เก็บเก้าอี้หลังจากใช้งานเสร็จ	ตลอดภาคการศึกษา	10%
2. ความรู้	2.1 แบบฝึกหัดท้ายบท 2.2 การสอบกลางภาคเรียนและปลายภาคเรียน	2-8, 11-16 9, 17	20% 40%
3. ทักษะทางปัญญา	3.1 ประเมินจากใบฝึกปฏิบัติการ	ตลอดภาคการศึกษา	20%
4. ทักษะความสัมพันธ์ระหว่างบุคคลและความรับผิดชอบ	4.1 พิจารณาจากการทำใบฝึกปฏิบัติการ 4.3 ประเมินจากพฤติกรรมในการทำงานกลุ่ม การระดมความคิดเห็น การตอบคำถามในชั้นเรียน	ตลอดภาคการศึกษา	5%
5. ทักษะวิเคราะห์เชิงตัวเลข การสื่อสารและการใช้เทคโนโลยีสารสนเทศ	5.1 การทดสอบหรือการประเมินจากงานที่มอบหมาย 5.2 สังเกตการใช้คอมพิวเตอร์ในการเขียนโปรแกรม การทำใบฝึกปฏิบัติการ และการค้นหาข้อมูลเพิ่มเติมจากอินเทอร์เน็ต	ตลอดภาคการศึกษา	5%
	รวม		100%

อ่าน Slide เวอร์ชันล่าสุดได้ที่ Google Slide

https://s.coecore.com/scp6

การโปรแกรมคอมพิวเตอร์

บทที่ 6 ฟังก์ชัน

(Function)

ผู้สอน ดร.พงศกร เจริญเนตรกุล

Outline

- ความหมายและประโยชน์ของฟังก์ชัน
- ฟังก์ชันในภาษา C
- การสร้างฟังก์ชันโดยผู้เขียนโปรแกรม
- การเรียกใช้งานฟังก์ชัน
- ฟังก์ชันแบบเรียกตัวเอง
- ขอบเขตของตัวแปร

ความหมายและประโยชน์ของฟังก์ชัน

ความหมายของฟังก์ชัน

ฟังก์ชัน (Function)

- กลุ่มของคำสั่งที่ทำงานร่วมกันเพื่อทำงานเฉพาะอย่าง
- ฟังก์ชันช่วยในการแบ่งโปรแกรมออกเป็นส่วนย่อย ๆ (Divide and Conquer)
- ทำให้โค้ดมีความเป็นระเบียบ เข้าใจง่าย และสามารถนำมาใช้ซ้ำได้
- หากต้องการใช้งานฟังก์ชัน จะใช้การเรียกชื่อ และส่งค่า(พารามิเตอร์)ที่ต้องการประมวลผลไปยังฟังก์ชัน

ประโยชน์ของฟังก์ชัน

1. การใช้ซ้ำ (Reusability)

- ฟังก์ชันสามารถเรียกใช้งานได้หลายครั้งภายในโปรแกรมเดียวกันหรือในโปรแกรมอื่น ๆ ทำให้ไม่ต้องเขียนโค้ดซ้ำ หากมีงานที่ต้องทำหลายครั้ง เช่น การคำนวณหรือการแสดงผล ฟังก์ชันจะช่วยลดการเขียนโค้ดซ้ำซ้อน

2. การแบ่งโปรแกรมออกเป็นส่วนย่อย (Modularity)

- ฟังก์ชันช่วยในการแยกโปรแกรมออกเป็นส่วนย่อย ๆ ที่ทำงานเฉพาะด้าน ซึ่งช่วยให้การพัฒนาโปรแกรมเป็นไปอย่างมีระบบ สามารถพัฒนาและทดสอบแต่ละฟังก์ชันแยกกันได้ ทำให้โค้ดมีความเป็นระเบียบ เข้าใจง่าย และแก้ไขได้ง่ายขึ้น

3. การแก้ไขและการบำรุงรักษาโปรแกรมง่ายขึ้น (Maintainability)

- หากโค้ดมีการใช้ฟังก์ชันอย่างเป็นระบบ เมื่อเกิดข้อผิดพลาดหรือมีการปรับปรุง ก็สามารถแก้ไขได้ในจุดเดียว (ที่ฟังก์ชันนั้น ๆ) แทนที่จะต้องแก้ไขโค้ดในทุก ๆ ที่ที่ใช้งาน ซึ่งช่วยลดความซับซ้อนและลดโอกาสเกิดข้อผิดพลาดในระหว่างการแก้ไข

ประโยชน์ของฟังก์ชัน

5. การดีบัก (Debugging) ง่ายขึ้น

- เนื่องจากฟังก์ชันเป็นส่วนย่อยที่แยกจากกัน การตรวจสอบและทดสอบการทำงานของฟังก์ชันแต่ละตัวจึงง่ายกว่าการดีบักโค้ดยาว ๆ การสามารถโฟกัสที่การทำงานของแต่ละฟังก์ชันได้จะช่วยทำให้การแก้ปัญหาหรือหาจุดบกพร่องง่ายขึ้น

6. การจัดการโปรแกรมขนาดใหญ่

- เมื่อโปรแกรมมีขนาดใหญ่และซับซ้อน การเขียนโค้ดทั้งหมดในฟังก์ชันเดียวจะทำให้โค้ดดูยุ่งเหยิงและยากต่อการจัดการ การแบ่งออกเป็นฟังก์ชันย่อย ๆ ทำให้โปรแกรมมีความชัดเจนและง่ายต่อการปรับปรุงหรือพัฒนาเพิ่มเติม

7. การทำงานร่วมกันในทีม (Collaboration)

- เมื่อเขียนโปรแกรมเป็นทีม การใช้ฟังก์ชันช่วยให้สมาชิกในทีมสามารถทำงานในส่วนของตนเองโดยไม่ต้องพึ่งพาการทำงานของผู้อื่นมากเกินไป เพราะสามารถแยกงานกันได้ เช่น คนหนึ่งเขียนฟังก์ชันคำนวณ อีกคนหนึ่งเขียนฟังก์ชันแสดงผล การทำงานแยกส่วนเช่นนี้ทำให้การทำงานร่วมกันมีประสิทธิภาพมากขึ้น

ประโยชน์ของฟังก์ชัน

8. หลักการ Single Responsibility Principle (SRP)

- หลักการ SRP เป็นหนึ่งในหลักการพื้นฐานของการออกแบบโค้ดที่ดี ฟังก์ชันควรทำหน้าที่อย่างเดียวเท่านั้น เช่น ฟังก์ชันหนึ่งอาจจะรับผิดชอบการคำนวณค่าบางอย่าง ฟังก์ชันอีกตัวอาจจะแสดงผลข้อมูล การแบ่งงานให้ฟังก์ชันแต่ละตัวทำงานเฉพาะเจาะจงจะช่วยให้โปรแกรมมีโครงสร้างที่ชัดเจนและแก้ไขง่าย

ตัวอย่างฟังก์ชัน

ตัวอย่างที่ 1 โปรแกรมคำนวณพื้นที่สี่เหลี่ยมผืนผ้าและวงกลม **โดยไม่มีฟังก์ชัน**

```
#include <stdio.h>

int main() {
    float width, height, radius;
    const float PI = 3.14159;
    float rectangleArea, circleArea;

    // คำนวณพื้นที่สี่เหลี่ยมผืนผ้า
    printf("กรุณาใส่ความกว้างของสี่เหลี่ยมผืนผ้า: ");
    scanf("%f", &width);
    printf("กรุณาใส่ความสูงของสี่เหลี่ยมผืนผ้า: ");
    scanf("%f", &height);
    rectangleArea = width * height;
    printf("พื้นที่ของสี่เหลี่ยมผืนผ้าคือ: %.2f\n", rectangleArea);

    // คำนวณพื้นที่วงกลม
    printf("กรุณาใส่รัศมีของวงกลม: ");
    scanf("%f", &radius);
    circleArea = PI * radius * radius;
    printf("พื้นที่ของวงกลมคือ: %.2f\n", circleArea);
}
```

ตัวอย่างฟังก์ชัน

ตัวอย่างที่ 2 โปรแกรมคำนวณพื้นที่สี่เหลี่ยมผืนผ้าและวงกลม **โดยมีฟังก์ชัน**

```
#include <stdio.h>

// ฟังก์ชันสำหรับคำนวณพื้นที่สี่เหลี่ยมผืนผ้า
float calculateRectangleArea(float width, float height) {
    return width * height;
}

// ฟังก์ชันสำหรับคำนวณพื้นที่วงกลม
float calculateCircleArea(float radius) {
    const float PI = 3.14159;
    return PI * radius * radius;
}

// ฟังก์ชันแสดงผลลัพธ์
void displayResult(char shape[], float area) {
    printf("พื้นที่ของ %s คือ: %.2f\n", shape, area);
}

int main() {
    float width, height, radius;
    float rectangleArea, circleArea;

    // คำนวณพื้นที่สี่เหลี่ยมผืนผ้า
    printf("กรุณาใส่ความกว้างของสี่เหลี่ยมผืนผ้า: ");
    scanf("%f", &width);
    printf("กรุณาใส่ความสูงของสี่เหลี่ยมผืนผ้า: ");
    scanf("%f", &height);
    rectangleArea = calculateRectangleArea(width, height);
    displayResult("สี่เหลี่ยมผืนผ้า", rectangleArea);

    // คำนวณพื้นที่วงกลม
    printf("กรุณาใส่รัศมีของวงกลม: ");
    scanf("%f", &radius);
    circleArea = calculateCircleArea(radius);
    displayResult("วงกลม", circleArea);
}
```


ตัวอย่างฟังก์ชัน

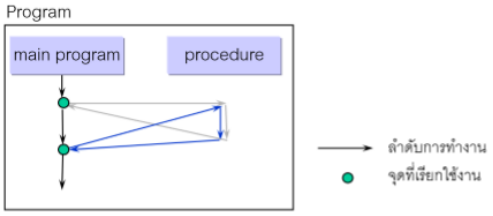
ตัวอย่างที่ 3

```
#include <stdio.h>
main()
{
        int first, second, third;
        printf("\n F(X) = 3X + 10 if X > 0\n");
        printf("\n F(X) = 10 if X = 0\n");
        printf("\n Enter 3 values\n");
        scanf("%d %d %d", &first, &second, &third);
        if (first > 0)
                printf("F(%d) is %d", first, 3*first + 10);
        else
                printf("F(%d) is 10", first);
        if (second > 0)
                printf("F(%d) is %d", second, 3*second + 10);
        else
                printf("F(%d) is 10", second);
        if (third > 0)
                printf("F(%d) is %d", third, 3*third + 10);
        else
                printf("F(%d) is 10", third);
}
```

Code ที่ซ้ำซ้อนสามารถนำไปรวมกันเป็นฟังก์ชันได้

Procedural Programming

รูปแบบ : ชุด statements สามารถรวมไว้ในที่ที่หนึ่งแยกจากส่วนของ main และมีการใช้ procedure call เพื่อเรียกใช้งาน ชุด statements นั้น



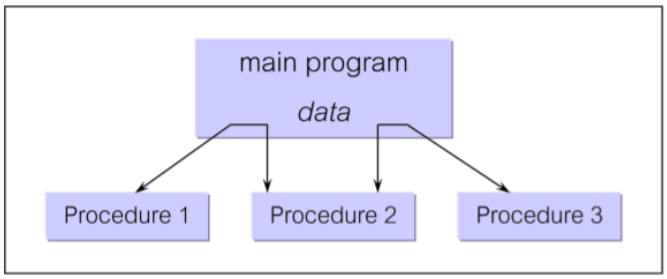
ข้อดีของ Procedural Programming

- มีการซ่อนรายละเอียดซับซ้อนไว้ใน Procedure ทำให้โปรแกรมหลักง่ายในการทำความเข้าใจ
- ลดความซ้ำซ้อนของส่วนโปรแกรมที่เหมือนๆกันไว้ใน Procedure เดียวกัน
- การหาข้อผิดพลาดของโปรแกรมง่ายขึ้น โดยแยกหาระหว่างส่วน main กับ procedures ใดๆ

Computer Programming 11

Procedural Programming

การทำงาน main เรียกใช้งาน procedure โดยอาจมีการส่งค่าให้กับการเรียกแต่ละครั้ง เราอาจมองได้ว่า main เป็นตัวประสานงานระหว่างหลายๆ procedures และเป็นส่วนคอยควบคุมข้อมูลหลัก



สรุป Procedural Programming มีหนึ่งโปรแกรมหลักที่ประกอบไปด้วยส่วนย่อยๆ เรียกว่า procedures หรือฟังก์ชัน

Computer Programming 12

ตัวอย่างฟังก์ชัน

```
ตัวอย่างที่ 4
#include <stdio.h>
void get_Fx(int x);
main()
{
 int first, second, third;
 printf("\n F(X) = 3X + 10 if X > 0\n");
 printf("\n F(X) = 10 if X = 0\n");
 printf("\n Enter 3 values\n");
 scanf("%d %d %d", &first, &second, &third);
 get_Fx(first);
 get_Fx(second);
 get_Fx(third);
}
void get_Fx(int x)
{
 if (x > 0)
   printf("F(%d) is %d", x, (3*x) + 10);
 else
   printf("F(%d) is 10", x);
}
```

➔ สามารถเรียกใช้โดยใช้ชื่อ และส่งค่าที่ต้องการคำนวณไปยังฟังก์ชัน

➔ รวมการทำงานแบบเดียวกันไว้ด้วยกัน

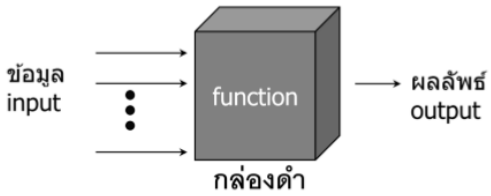
➔ เปลี่ยนแปลงตัวแปร xในการเรียกใช้งานแต่ละครั้ง

ฟังก์ชันในภาษาซี

ฟังก์ชันในภาษาซี

ฟังก์ชัน (Function) ในภาษาซี

- เป็นกลุ่มของคำสั่งที่ทำงานร่วมกันเพื่อทำงานเฉพาะอย่าง โดยฟังก์ชันช่วยในการแบ่งโปรแกรมออกเป็นส่วนย่อย ๆ ซึ่งทำให้โค้ดมีความเป็นระเบียบ เข้าใจง่าย และสามารถนำมาใช้ซ้ำได้
- ต้องมีชื่อเรียกเฉพาะ
- ผู้ใช้งานเพียงแค่รู้ input output และชื่อเรียก ก็สามารถใช้งานฟังก์ชันที่มีอยู่ในระบบได้แล้ว เปรียบได้ว่าฟังก์ชันเป็นกล่องดำ หรือ เครื่องจักรที่เราไม่ต้องรู้กลไกภายใน เพียงแต่รู้แค่ต้องทำยังไงถึงเปิดเครื่องติด เครื่องนั้นทำอะไรได้ และต้องใส่อะไรเข้าไปในเครื่อง



ฟังก์ชันในภาษาซี

เช่น เราต้องการทำการคำนวณค่ารากที่สองของจำนวน โดยใช้งานฟังก์ชัน sqrt() จากไลบรารี math.h ในภาษา C ใช้สำหรับคำนวณค่ารากที่สองของตัวเลขที่เป็นจำนวนจริง (float หรือ double)

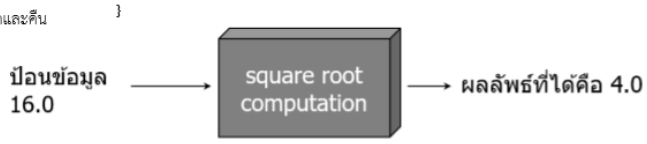
วิธีการใช้งาน

1. ต้องใส่คำสั่ง #include <math.h> เพื่อใช้งานฟังก์ชันจากไลบรารี math.h
2. ฟังก์ชัน sqrt() รับค่าเป็นจำนวนจริงบวกและคืนค่ารากที่สองของตัวเลขนั้น

```
#include <stdio.h>
#include <math.h>

int main() {
   double num = 25.0;
   double result = sqrt(num);

   printf("รากที่สองของ %.2f คือ %.2f\n", num, result);
}
```



ฟังก์ชันในภาษาซี

โปรแกรมภาษา C ต้องมี**ฟังก์ชันหลัก (main) ซึ่งเป็นฟังก์ชันแรกที่โปรแกรมจะทำงานเสมอ**

แต่ละฟังก์ชันต้องประกอบไปด้วยหนึ่งคำสั่ง (Statement) หรือมากกว่า

ภาษา C แบ่งฟังก์ชันเป็น 3 แบบ

1 ฟังก์ชันมาตรฐานใน C (C STANDARD LIBRARY)

2 ฟังก์ชันที่สามารถติดตั้งเพิ่มเติมใน C (C EXTERNAL LIBRARY)

3 ฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรม (PROGRAMMER DEFINED FUNCTION)

1 ฟังก์ชันมาตรฐานใน C (C Standard Library Function)

ฟังก์ชันมาตรฐานในภาษา C คือฟังก์ชันที่มาพร้อมกับไลบรารีมาตรฐานของ C ซึ่งสามารถใช้งานได้ทันทีหลังจากรวมไฟล์เฮดเดอร์ที่เกี่ยวข้อง เช่น stdio.h, stdlib.h, หรือ math.h โดยไม่ต้องติดตั้งอะไรเพิ่มเติม

ตัวอย่าง printf() จากไลบรารี stdio.h ใช้สำหรับแสดงผลข้อความหรือตัวแปรลงบนหน้าจอ

```
#include <stdio.h>

int main() {

    printf("Hello, World!\n");

}
```

1 ฟังก์ชันมาตรฐานใน C (C Standard Library Function)

โดยฟังก์ชันที่ผู้ผลิต C compiler เป็นผู้เขียนขึ้น โดยเก็บไว้ใน C library
ประกอบด้วยฟังก์ชันเกี่ยวกับ

- disk I/O (input/output), standard I/O ex. printf(), scanf(), ...
- string manipulation ex. strlen(), strcpy(), ...
- mathematics ex. sqrt(), sin(), cos(), ...
- etc..

สามารถเรียกใช้งานได้เลย แต่ต้องมีการ include header file ที่นิยามฟังก์ชันนั้นๆไว้ เช่น

- จะใช้ printf(), scanf() ต้องเขียน #include <stdio.h>
- จะใช้ sqrt(), sin(), cos() ต้องเขียน #include <math.h>
- etc.

1 ฟังก์ชันมาตรฐานใน C (C Standard Library Function)

เช่น เราต้องการทำการคำนวณค่ารากที่สองของจำนวน โดยใช้งานฟังก์ชัน sqrt() จากไลบรารี math.h ในภาษา C ใช้สำหรับคำนวณค่ารากที่สองของตัวเลขที่เป็นจำนวนจริง (float หรือ double)

วิธีการใช้งาน

1. ต้องใส่คำสั่ง #include <math.h> เพื่อใช้งานฟังก์ชันจากไลบรารี math.h
2. ฟังก์ชัน sqrt() รับค่าเป็นจำนวนจริงบวกและคืนค่ารากที่สองของตัวเลขนั้น

```
#include <stdio.h>
#include <math.h>

int main() {
   double num = 16.0;
   double result = sqrt(num);

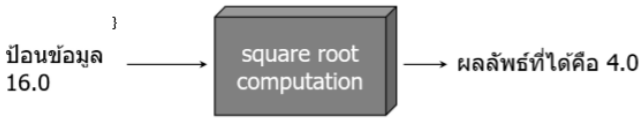
   printf("%.2f\n", sqrt(16.0));
   printf("%.2f\n", sqrt(sqrt(16.0)));

}
```

Output:

4.00

2.00



1 ฟังก์ชันมาตรฐานใน C (C Standard Library Function)

โปรแกรมแสดง 10 ยกกำลัง 1 ถึง 5 โดยเรียกใช้ฟังก์ชัน pow() ใน math.h

ต้นแบบ double pow(double base, double exp);

โดยกำหนดให้ $base^{exp}$

```
#include <stdio.h>
#include <math.h>
int main()
{
    double x=10.0, y =1.0;

    do {
        printf("%.2f\n", pow(x,y));
        y++;
    } while (y < 6);
}
```



Output:

10.00

100.00

1000.00

10000.00

100000.00

2 ฟังก์ชันที่สามารถติดตั้งเพิ่มเติมใน C (C External Library Function)

ฟังก์ชันในไลบรารีเพิ่มเติมหรือไลบรารีภายนอก **จะเป็นฟังก์ชันที่ไม่ได้รวมอยู่ในไลบรารีมาตรฐานของภาษา C แต่สามารถนำมาติดตั้งและใช้งานได้ตามความต้องการ** โดยปกติจะเป็นไลบรารีที่เขียนโดยนักพัฒนาภายนอกและสามารถดาวน์โหลดหรือติดตั้งเพิ่มเติม

ตัวอย่างเช่น

curl_easy_init() จากไลบรารี libcurl ใช้สำหรับการทำ HTTP requests โดยต้องติดตั้งไลบรารี libcurl ก่อนใช้งาน

```
#include <stdio.h>
#include <curl/curl.h>

int main() {
    CURL *curl = curl_easy_init();
    if(curl) {
        curl_easy_cleanup(curl);
    }
}
```

3 ฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรม (Programmer Defined Function)

ฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรมคือฟังก์ชันที่ผู้พัฒนาเขียนขึ้นเองเพื่อตอบสนองความต้องการเฉพาะในโปรแกรม ฟังก์ชันเหล่านี้สามารถรับพารามิเตอร์ ส่งคืนค่า และมีโครงสร้างคล้ายฟังก์ชันมาตรฐาน แต่ถูกออกแบบโดยนักพัฒนาเอง

ตัวอย่าง ฟังก์ชันคำนวณพื้นที่วงกลม

```
#include <stdio.h>
#include <math.h> // รวมไลบรารี math.h เพื่อใช้ค่าคงที่ PI

// ฟังก์ชันคำนวณพื้นที่วงกลม
float calculateCircleArea(float radius) {
    const float PI = 3.14159;
    return PI * radius * radius;
}
```


การสร้างฟังก์ชันโดยผู้เขียนโปรแกรม

การสร้างฟังก์ชันโดยผู้เขียนโปรแกรม

ฟังก์ชันจะมีองค์ประกอบ 2 ส่วน ได้แก่

1. **ต้นแบบฟังก์ชัน (Function Prototype)**

```
return-type  function-name (parameter-list) ;
```

1. **นิยามฟังก์ชัน (Function Definition)**

```
return-type function-name (parameter-list) // header เหมือน function Prototype
{
        รายละเอียดการทำงาน (Algorithm)
}
```


การสร้างฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรม

ขั้นตอนการสร้างฟังก์ชัน

1. วิเคราะห์วัตถุประสงค์ของฟังก์ชัน ซึ่งฟังก์ชันควรมีหน้าที่แค่อย่างใดอย่างหนึ่งเท่านั้นตามหลักการของ Single Responsibility Principle (SRP)
2. วิเคราะห์และกำหนด **ชื่อฟังก์ชัน / Input / Output / Process** ของฟังก์ชัน
3. เขียน**ต้นแบบฟังก์ชัน (Function Prototype)** จากสั่งที่ได้กำหนด
4. เขียน**นิยามของฟังก์ชัน (Function Definition)**
5. **ทดสอบฟังก์ชันโดยการเรียกใช้งานฟังก์ชันและตรวจสอบความถูกต้องในกรณีต่างๆ**

1. **ต้นแบบฟังก์ชัน (Function Prototype)**

ในภาษาซี (C) **มีความสำคัญเพราะช่วยให้คอมไพเลอร์สามารถทราบถึงข้อมูลของฟังก์ชันก่อนที่จะทำการเรียกใช้งานในโค้ดหลัก** การมีต้นแบบฟังก์ชันจะช่วยลดข้อผิดพลาด และทำให้โปรแกรมทำงานได้อย่างถูกต้องและมีประสิทธิภาพมากขึ้น

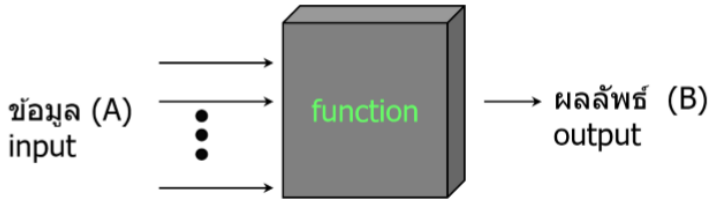
ต้นแบบฟังก์ชันจำเป็นต้องเขียนหรือไม่ ขึ้นกับตำแหน่งที่เรานิยามฟังก์ชันในโปรแกรม

- **ถ้าฟังก์ชันนิยามก่อน main (in-line function) ไม่จำเป็นต้องมีต้นแบบฟังก์ชัน**
- **ถ้าฟังก์ชันนิยามหลัง main จำเป็นต้องมีการประกาศต้นแบบฟังก์ชันก่อนฟังก์ชัน main**

1. **ต้นแบบฟังก์ชัน (Function Prototype)**

ต้นแบบของฟังก์ชันเป็นตัวบอกให้คอมไพเลอร์ (Compiler) รู้ถึง

1. **ชนิดข้อมูลที่จะส่งค่ากลับ (Output) (A)**
2. **จำนวนพารามิเตอร์ที่ฟังก์ชันต้องการ (B)**
3. **ชนิดของพารามิเตอร์ (Input) แต่ละตัว รวมทั้งลำดับของพารามิเตอร์**



1. ต้นแบบฟังก์ชัน (Function Prototype)

โครงสร้างต้นแบบฟังก์ชัน

```
return-type  function-name (parameter-list) ;
```

return-type หรือชนิดข้อมูลที่จะส่งค่ากลับได้แก่ void, int, double, char, ...

function-name ชื่อของฟังก์ชัน

parameter-list จำนวนพารามิเตอร์ที่ฟังก์ชันต้องการ (Input ซึ่งเป็นตัวแปรที่ต้องการส่งไปให้ฟังก์ชันคำนวณ/ประมวลผล)

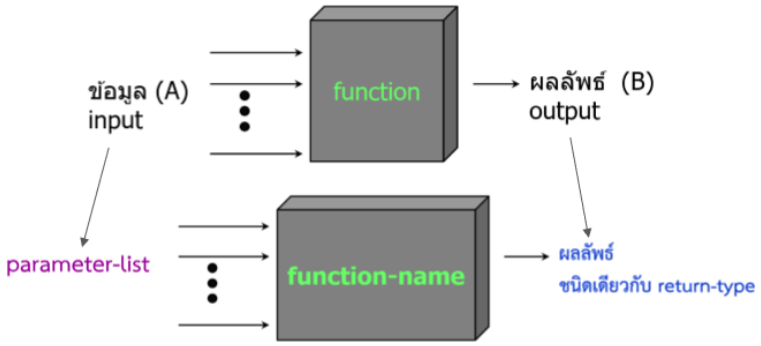
- แต่ละพารามิเตอร์ประกอบด้วย ชนิดตัวแปรและชื่อตัวแปร
- แต่ละพารามิเตอร์แยกด้วยเครื่องหมาย " , "

ตัวอย่าง `int add (int a, int b) ;`

1. ต้นแบบฟังก์ชัน (Function Prototype)

โครงสร้างต้นแบบฟังก์ชัน

```
return-type  function-name (parameter-list) ;
```



2. นิยามฟังก์ชัน (Function Definition)

นิยามฟังก์ชันเป็นการเขียนรายละเอียดการทำงานของฟังก์ชันนั้นๆ

ประกอบด้วยส่วนของ header และ algorithm

โดย header **จะมีการเขียนเหมือน ต้นแบบฟังก์ชัน แต่จะไม่มี ;**

algorithm เขียนอยู่ภายใน { }

```
return-type function-name (parameter-list) // header เหมือน function Prototype
{
        รายละเอียดการทำงาน (Algorithm)
}
```

***ถ้า return-type ไม่ใช่ void ต้องมีการส่งค่ากลับเสมอ โดยใช้คำสั่ง return ตามด้วยค่าที่จะส่งกลับ

2. นิยามฟังก์ชัน (Function Definition)

ตัวอย่างฟังก์ชันที่มี return

```
int add (int a, int b)
{
    int result;
    result = a+b;
    return result;
}
```

ตัวอย่างฟังก์ชันที่ไม่มี return

```
void checkresult (int result)
{
  if(result < 0)
        printf("result is negative");
  if(result == 0)
        printf("result is zero");
  if(result > 0)
        printf("result is positive")
}
```

***ถ้า return-type ไม่ใช่ void ต้องมีการส่งค่ากลับเสมอ โดยใช้คำสั่ง return ตามด้วยค่าที่จะส่งกลับ

ตัวอย่างโครงสร้างของฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรม

```
#include <stdio.h>
void func2();
void func1();
int main() {
  func2();
  printf("3 ");
  return 0;
}
void func2() {
  func1();
  printf("2 ");
}
void func1() {
  printf("1 ");
}
```

ส่วนของการ include header file(s) หรือ Preprocessor

ส่วนที่ประกาศว่ามีฟังก์ชันอะไรบ้างที่สร้างขึ้นใช้ในโปรแกรมนี้ (Function Prototype)

โปรแกรมหลัก (Main Function)

โครงสร้างที่ฟังก์ชันนิยามหลังการเรียกใช้

จำเป็นต้องมี ต้นแบบฟังก์ชัน

ส่วนการนิยามรายละเอียดการทำงานของฟังก์ชัน func2 (Function Definition)

ส่วนการนิยามรายละเอียดการทำงานของฟังก์ชัน func1 (Function Definition)

ตัวอย่างโครงสร้างของฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรม

```
#include <stdio.h>
void func2() {
  func1();
  printf("2 ");
}
void func1() {
  printf("1 ");
}
int main() {
  func2();
  printf("3 ");
  return 0;
}
```

ส่วนของการ include header file(s) หรือ Preprocessor

ส่วนการนิยามรายละเอียดการทำงานของฟังก์ชัน func2 (Function Definition)

ส่วนการนิยามรายละเอียดการทำงานของฟังก์ชัน func1 (Function Definition)

โครงสร้างที่ฟังก์ชันนิยามก่อนเรียกใช้

ไม่จำเป็นต้องมี ต้นแบบฟังก์ชัน

โปรแกรมหลัก (Main Function)

ตัวอย่างการสร้างฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรม

ตัวอย่างขั้นตอนการสร้างฟังก์ชัน

การคำนวณพื้นที่สี่เหลี่ยมผืนผ้า

1. วิเคราะห์วัตถุประสงค์ของฟังก์ชัน (SRP: Single Responsibility Principle)

วัตถุประสงค์: ฟังก์ชันที่เรากำลังจะสร้างต้องทำงานเพียงแค่การคำนวณพื้นที่สี่เหลี่ยมผืนผ้าเท่านั้น ซึ่งเป็นไปตามหลักการ SRP โดยที่ไม่ควรรวมการทำงานอื่นๆ (เช่น การพิมพ์ผลลัพธ์หรือรับข้อมูล) ฟังก์ชันนี้ควรทำหน้าที่เฉพาะในการรับค่าและคำนวณผลลัพธ์ของพื้นที่

ตัวอย่างการสร้างฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรม

ตัวอย่างขั้นตอนการสร้างฟังก์ชัน

การคำนวณพื้นที่สี่เหลี่ยมผืนผ้า

2. วิเคราะห์และกำหนด ชื่อฟังก์ชัน / Input / Output / Process

ชื่อฟังก์ชัน: calRectArea เพื่อบอกชัดเจนว่าฟังก์ชันนี้ทำหน้าที่คำนวณพื้นที่

Input: ความกว้างและความยาวของสี่เหลี่ยมผืนผ้า ซึ่งจะรับเป็น float สองค่าคือ width และ height

Output: ส่งคืนค่าพื้นที่สี่เหลี่ยมเป็น float

Process: คำนวณพื้นที่โดยใช้สูตร พื้นที่ = ความกว้าง * ความยาว

ตัวอย่างการสร้างฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรม

ตัวอย่างขั้นตอนการสร้างฟังก์ชัน

การคำนวณพื้นที่สี่เหลี่ยมผืนผ้า

3. เขียนต้นแบบฟังก์ชัน (Function Prototype)

ต้นแบบฟังก์ชันจะกำหนดประเภทข้อมูลที่รับเข้าและส่งออก

```
float calRectArea(float width, float height);
```


ตัวอย่างการสร้างฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรม

ตัวอย่างขั้นตอนการสร้างฟังก์ชัน

การคำนวณพื้นที่สี่เหลี่ยมผืนผ้า

4. เขียนนิยามของฟังก์ชัน (Function Definition)

นิยามฟังก์ชันจะบอกว่าฟังก์ชันนี้จะทำอะไร โดยในที่นี้จะคำนวณพื้นที่ตามสูตรและส่งค่าผลลัพธ์กลับ

```
float calRectArea(float width, float height) {
   return width * height;
}
```

ตัวอย่างการสร้างฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรม

ตัวอย่างขั้นตอนการสร้างฟังก์ชัน

การคำนวณพื้นที่สี่เหลี่ยมผืนผ้า

5. ทดสอบฟังก์ชันโดยการเรียกใช้งานฟังก์ชันและตรวจสอบความถูกต้องในกรณีต่างๆ

จะสร้างโปรแกรมหลักเพื่อเรียกใช้งานฟังก์ชัน calculate_area และตรวจสอบผลลัพธ์ด้วยการพิมพ์ออกทางหน้าจอ หรือเขียน Test Case ซึ่งเป็นเฉลยและมีฟังก์ชันพิเศษเพื่อเรียกฟังชัน พร้อมเปรียบเทียบกับเฉลย

```c
#include <stdio.h>

// ต้นแบบฟังก์ชัน (Function Prototype)
float calRectArea(float width, float height);

int main() {
    float width, height, area;

    // รับค่าจากผู้ใช้
    printf("กรุณาใส่ค่าความกว้าง: ");
    scanf("%f", &width);
    printf("กรุณาใส่ค่าความยาว: ");
    scanf("%f", &height);

    // เรียกใช้ฟังก์ชันคำนวณพื้นที่
    area = calRectArea(width, height);

    // แสดงผลลัพธ์
    printf("พื้นที่ของสี่เหลี่ยมผืนผ้าคือ: %.2f\n", area);

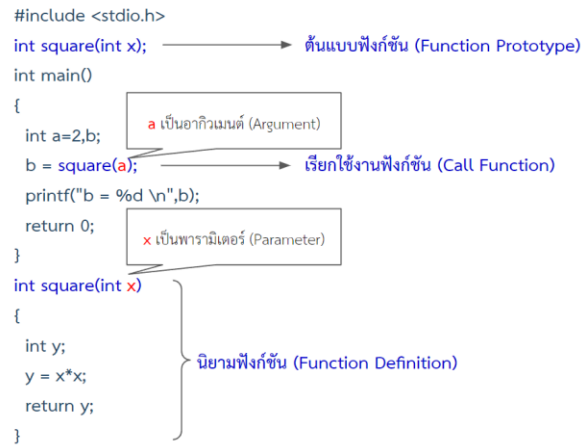
    return 0;
}

// นิยามฟังก์ชัน (Function Definition)
float calRectArea(float width, float height) {
    return width * height;
}
```


ตัวอย่างฟังก์ชัน

```c
#include <stdio.h>
int square(int x);
int main()
{
  int a=2,b;
  b = square(a);
  printf("b = %d \n",b);
  return 0;
}
int square(int x)
{
  int y;
  y = x*x;
  return y;
}
```

ตัวอย่างฟังก์ชัน



การเรียกใช้งานฟังก์ชัน (Call Function)

การเรียกใช้งานฟังก์ชัน (Call Function)



1. Boss สั่งให้ทำงาน
(Call Function)

3. ส่งงานให้ Boss
(Return)

2. ทำงาน ทำงาน ทำงาน
(Do in Function)



BOSS
(calling function)

Worker
(called function)

BOSS อาจเป็นโปรแกรม main หรือเป็นฟังก์ชันอื่น
WORKER เป็นฟังก์ชันที่ถูกเรียกใช้งาน

อ้างอิงรูปภาพ: Generate AI Image from DALL-E

การเรียกใช้งานฟังก์ชัน (Call Function)

ขั้นตอนและการทำงานของการเรียกฟังก์ชัน

1. **การเรียกฟังก์ชันด้วยชื่อ** (Function Call) **จะเรียกใช้งานฟังก์ชันด้วยการระบุชื่อฟังก์ชัน**
2. **การส่งพารามิเตอร์** โดยการส่งพารามิเตอร์ (Input) สามารถส่งผ่านไปยังฟังก์ชันเพื่อให้ฟังก์ชันประมวลผลข้อมูลนั้น หากฟังก์ชันไม่รับพารามิเตอร์ ก็สามารถเรียกใช้งานได้โดยไม่ต้องส่งค่าใด ๆ
3. **การส่งคืนค่า (ถ้ามี)** โดยหากฟังก์ชันมีการคืนค่าผลลัพธ์ (return value) ค่าผลลัพธ์นั้นจะถูกส่งกลับไปยังจุดที่เรียกฟังก์ชัน และสามารถนำไปใช้งานต่อได้ เราต้องมีตัวแปรมารับ เช่น int ret_value = call_func1();
4. **การดำเนินการต่อในโปรแกรม** โดยหลังจากที่ฟังก์ชันทำงานเสร็จแล้ว การทำงานของโปรแกรมจะดำเนินต่อจากจุดที่มีการเรียกฟังก์ชัน เพื่อนำผลลัพธ์ที่ได้จากฟังก์ชันไปประมวลผลต่อ

การเรียกใช้งานฟังก์ชัน (Call Function)

รูปแบบการเรียกใช้งานฟังก์ชัน (Function Call Syntax)

```
result = function_name(arguments);
```

function_name: ชื่อของฟังก์ชันที่ต้องการเรียกใช้งาน

arguments: พารามิเตอร์ที่ส่งให้ฟังก์ชันประมวลผล ถ้าไม่มีให้เว้นว่างไว้

result: ตัวแปรที่ใช้เก็บค่าผลลัพธ์ที่ฟังก์ชันส่งคืน (กรณีที่ฟังก์ชันมีการ return ค่า)

การเรียกใช้งานฟังก์ชัน (Call Function)

ประเภทของการเรียกฟังก์ชัน

1. **การเรียกฟังก์ชันแบบทั่วไป**: เรียกใช้ฟังก์ชันแล้วให้ทำงานตามที่นิยามไว้ โดยอาจมีการส่งค่าพารามิเตอร์และรับค่าผลลัพธ์กลับมา (ถ้ามี)

   เช่น `int sum = add(5, 10);`

1. **การเรียกฟังก์ชันแบบไม่สนใจผลลัพธ์**: บางครั้งเราอาจเรียกใช้งานฟังก์ชันเพื่อให้ทำงานโดยไม่จำเป็นต้องใช้ค่าที่ฟังก์ชันส่งกลับ

   เช่น `showErrorMsg("ERR:NO Response from Server",align,level);`

การเรียกใช้งานฟังก์ชัน (Call Function)

ประเภทของการเรียกฟังก์ชันตามการคืนค่า

1. **การเรียกฟังก์ชันแบบทั่วไป**: เรียกใช้ฟังก์ชันแล้วให้ทำงานตามที่นิยามไว้ โดยอาจมีการส่งค่าพารามิเตอร์และรับค่าผลลัพธ์กลับมา (ถ้ามี)

   เช่น `int sum = add(5, 10);`

1. **การเรียกฟังก์ชันแบบไม่สนใจผลลัพธ์**: บางครั้งเราอาจเรียกใช้งานฟังก์ชันเพื่อให้ทำงานโดยไม่จำเป็นต้องใช้ค่าที่ฟังก์ชันส่งกลับ

   เช่น `showErrorMsg("ERR:NO Response from Server",align,level);`

การเรียกใช้งานฟังก์ชัน (Call Function)

ประเภทของการเรียกฟังก์ชันตามการรับส่งข้อมูล

ประเภทที่ 1 - **ฟังก์ชันที่ไม่รับผ่านค่า และไม่ส่งผ่านค่ากลับ**

```
void func1();
```

ประเภทที่ 2 - **ฟังก์ชันที่มีการรับผ่านค่า แต่ไม่ส่งผ่านค่ากลับ**

```
void func2(int a);
```

ประเภทที่ 3 - **ฟังก์ชันที่มีการรับผ่านค่า และส่งผ่านค่ากลับ**

```
int func3(int a);
```

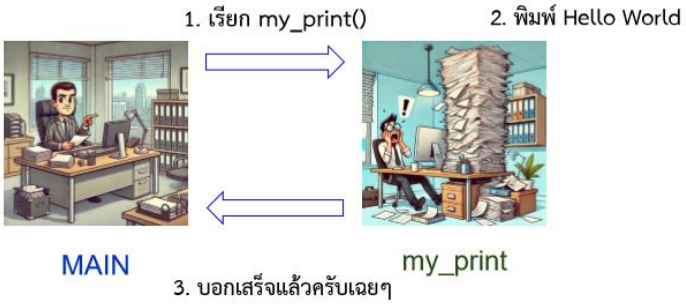
ประเภทที่ 4 - **ฟังก์ชันที่ไม่รับผ่านค่า แต่มีการส่งผ่านค่ากลับ**

```
int func4();
```

ตัวอย่างการเรียกใช้งานฟังก์ชัน (Call Function)

ฟังก์ชันที่ไม่รับผ่านค่า และไม่ส่งผ่านค่ากลับ

```
#include <stdio.h>
void my_print();
int main()
{
    my_print();
}
void my_print()
{
    printf("Hello world");
}
```

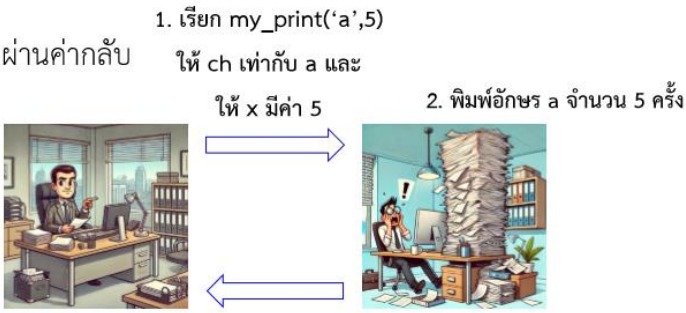


 อ้างอิงรูปภาพ: Generate AI Image from DALL-E

ตัวอย่างการเรียกใช้งานฟังก์ชัน (Call Function)

ฟังก์ชันที่มีการรับผ่านค่า แต่ไม่ส่งผ่านค่ากลับ

```
#include <stdio.h>
void my_print(char ch, int x);
int main()
{
    my_print('a', 5);
}
void my_print(char ch, int x)
{
    while (x > 0)
    {
        printf("%c", ch);
        x--;
    }
}
```



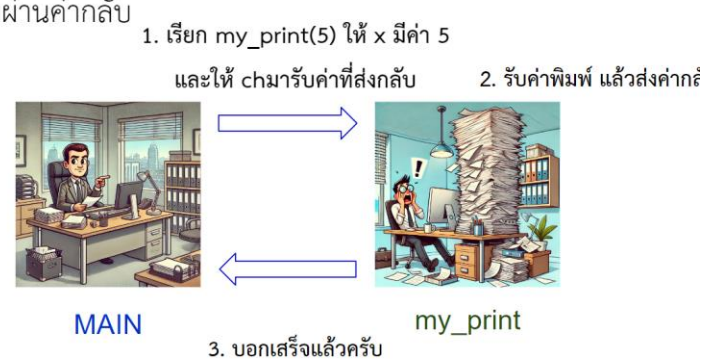
Note :
ฟังก์ชันต้องการพารามิเตอร์ 2 ตัว ตอนเรียกใช้งานต้องให้ใส่อากิวเมนต์ให้ถูกต้องตามจำนวน ชนิด และลำดับตามที่ฟังก์ชันต้องการ

 อ้างอิงรูปภาพ: Generate AI Image from DALL-E

ตัวอย่างการเรียกใช้งานฟังก์ชัน (Call Function)

ฟังก์ชันที่มีการรับผ่านค่า และส่งผ่านค่ากลับ

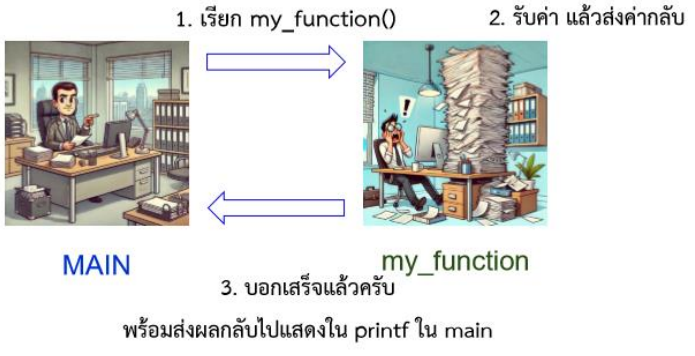
```
#include <stdio.h>
char my_print(int x);
int main()
{
    char ch;
    ch = my_print(5);
    printf("%c", ch);
}
char my_print(int x)
{
    char lch;
    scanf("%c", &lch);
    while (x > 0)
    {
        printf("%c", lch);
        x--;
    }
   return lch;
}
```



ตัวอย่างการเรียกใช้งานฟังก์ชัน (Call Function)

ฟังก์ชันที่ไม่รับผ่านค่า แต่มีการส่งผ่านค่ากลับ

```
#include <stdio.h>
int my_function();
int main()
{
    printf("%d", my_function());
}
int my_function()
{
    int a;
    scanf("%d", &a);
    return a*5;
}
```



สรุปการเรียกใช้งานฟังก์ชัน (Call Function)

ฟังก์ชันที่ไม่มีการส่งค่ากลับ สามารถเรียกชื่อฟังก์ชันได้เลย โดยต้องให้อากิวเมนต์ให้ถูกต้อง

เช่น my_print();

```
void my_print()
{
    printf("Hello world");
}
```

my_print('a',5);

```
void my_print(char ch, int x)
{
    while (x > 0)
    {
        printf("%c", ch);
        x--;
    }
}
```


สรุปการเรียกใช้งานฟังก์ชัน (Call Function)

ฟังก์ชันที่มีการส่งค่ากลับ ให้มองเสมือนว่าเป็นตัวแปรตัวหนึ่งที่มีชนิดเดียวกับชนิดของข้อมูลที่ส่งค่ากลับ

วิธีเรียกใช้งาน

☑ เอาตัวแปรมารับค่า เช่น ch = my_print(5);

☑ ใช้เป็นส่วนหนึ่งในนิพจน์ เช่น printf("%d", my_function());

☑ ใช้เป็นอาร์กิวเมนต์ของฟังก์ชันอื่น เช่น มีฟังก์ชัน int add(int a, int b);

int x = add(add(1,2),4);

คำนวณ add(1,2) ก่อน แล้วผลลัพธ์ที่ได้ ให้เป็นอาร์กิวเมนต์ของ add(... ,4)

ตัวอย่างการออกแบบต้นแบบฟังก์ชัน

จงเขียนโปรโตไทป์ให้ฟังก์ชันชื่อ script ที่มีพารามิเตอร์ 3 ตัว

1 จำนวนช่องว่างที่ต้องการให้แสดงเมื่อเริ่มบรรทัด

2 ตัวอักษรที่ต้องการให้แสดงหลังจากช่องว่างในพารามิเตอร์แรก

3 จำนวนครั้งที่ต้องการให้แสดงตัวอักษรในพารามิเตอร์ที่ 2

โดยฟังก์ชัน script ไม่มีการส่งค่ากลับ

```
void script(int space, char c, int time);
```


ตัวอย่างการเขียนนิยามฟังก์ชัน

จงเขียนนิยามฟังก์ชันชื่อ script ที่มีพารามิเตอร์ 3 ตัว

1 จำนวนช่องว่างที่ต้องการให้แสดงเมื่อเริ่มบรรทัด

2 ตัวอักษรที่ต้องการให้แสดงหลังจากช่องว่างในพารามิเตอร์แรก

3 จำนวนครั้งที่ต้องการให้แสดงตัวอักษรในพารามิเตอร์ที่ 2 โดยฟังก์ชัน script ไม่มีการส่งค่ากลับ

```
void script(int space, char c, int time)
{
        int i;
    for(i=0; i< space; i++)
        printf(" ");
    for(i=0; i< time; i++)
        printf("%c", c);
}
```

ตัวอย่างการเขียนฟังก์ชัน

จงเขียนโปรแกรมสำหรับหาค่าศักย์ไฟฟ้าซึ่งมีสมการดังนี้

V = I*R

โดยที่ V คือ ค่าศักย์ไฟฟ้า

I คือ ค่ากระแสไฟฟ้า

R คือ ค่าความต้านทาน

โดยทั้งสามค่านี้เป็นจำนวนจริง

กำหนดให้ส่วนที่คำนวณค่า V อยู่ในฟังก์ชัน get_volt สำหรับส่วนที่รับค่า I และ R จากผู้ใช้ รวมทั้งส่วนที่แสดงผลลัพธ์ของค่า V ให้อยู่ในฟังก์ชัน main

ตัวอย่างการเขียนฟังก์ชัน

จงเขียนโปรแกรมสำหรับหาค่าศักย์ไฟฟ้าซึ่งมีสมการดังนี้

V = I*R

โดยที่ V คือ ค่าศักย์ไฟฟ้า

I คือ ค่ากระแสไฟฟ้า

R คือ ค่าความต้านทาน

โดยทั้งสามค่านี้เป็นจำนวนจริง

กำหนดให้ส่วนที่คำนวณค่า V อยู่ในฟังก์ชัน get_volt สำหรับส่วนที่รับค่า I และ R จากผู้ใช้ รวมทั้งส่วนที่แสดงผลลัพธ์ของค่า V ให้อยู่ในฟังก์ชัน main

```c
#include <stdio.h>

float get_volt(float I,float R);
int main()
{   float i,r,v;
    printf("Enter  I  : ");
    scanf("%f", &i);
    printf("Enter  R  : ");
    scanf("%f", &r);
    v = get_volt(i,r);
    printf("Volt = %f \n",v);
}

float get_volt(float I,float R)
{
  float V = I*R;
  return V;
}
```

ตัวอย่างการเขียนฟังก์ชัน

จงเขียนโปรแกรมสำหรับหาค่า F(x) ซึ่งมีสมการดังนี้

F(x) = x2 + 2x + 1 ถ้า x มีค่าไม่ต่ำกว่า 0

= 0 ถ้า x มีค่าต่ำกว่า 0

กำหนดให้ส่วนที่ใช้ในการคำนวณค่า F(x) อยู่ในฟังก์ชัน get_fx สำหรับส่วนที่รับค่า x จากผู้ใช้ และแสดงผลลัพธ์ของค่า F(x) อยู่ในฟังก์ชัน main (x และ F(x) เป็นจำนวนเต็ม)

ตัวอย่างการเขียนฟังก์ชัน

จงเขียนโปรแกรมสำหรับหาค่า F(x) ซึ่งมีสมการดังนี้

F(x) = x2 + 2x + 1 ถ้า x มีค่าไม่ต่ำกว่า 0

= 0 ถ้า x มีค่าต่ำกว่า 0

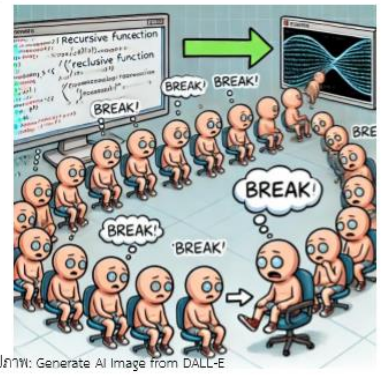
กำหนดให้ส่วนที่ใช้ในการคำนวณค่า F(x) อยู่ในฟังก์ชัน get_fx สำหรับส่วนที่รับค่า x จากผู้ใช้ และแสดงผลลัพธ์ของค่า F(x) อยู่ในฟังก์ชัน main (x และ F(x) เป็นจำนวนเต็ม)

```
#include <stdio.h>
int get_fx(int a);
int main()
{
        int x,fx;
   printf("Enter  x  : ");
   scanf("%d", &x);
   fx = get_fx(x);
   printf("f(%d) = %d \n",x,fx);
}
int get_fx(int a)
{
   if(a>=0)
        return (a*a)+(2*a)+1;
   else
        return 0;
}
```

ฟังก์ชันแบบเรียกตัวเอง (Recursive Function)

ฟังก์ชันแบบเรียกตัวเอง (Recursive Function)

ฟังก์ชันแบบเรียกตัวเอง เป็น ฟังก์ชันที่เรียกใช้งานตัวเองภายในฟังก์ชัน (Call Itself) โดยจะทำการดำเนินการซ้ำจนกว่าจะถึงเงื่อนไขหยุด (Base Case) เพื่อป้องกันการวนลูปไปเรื่อยๆ จนทำให้เกิดข้อผิดพลาดอย่างเช่น stack overflow จะมีวัตถุประสงค์คล้ายๆกับการวนรอบทำซ้ำ



อ้างอิงรูปภาพ: Generate AI Image from DALL-E

ฟังก์ชันแบบเรียกตัวเอง (Recursive Function)

วิธีการสร้างฟังก์ชันแบบเรียกตัวเอง

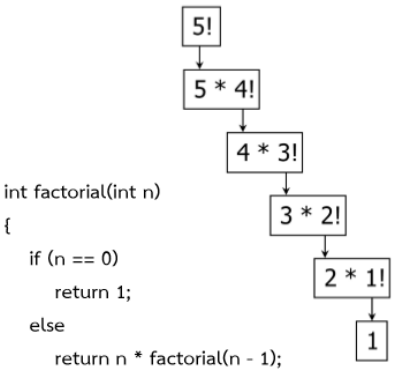
การสร้างฟังก์ชันแบบเรียกตัวเองในภาษาซีทำได้โดยการเขียนฟังก์ชันและให้มีการเรียกใช้งานฟังก์ชันนั้นๆ ภายในตัวมันเอง **ตัวอย่างเช่นการหาค่าของ factorial**

```
int factorial(int n)
{
    if (n == 0) {
        return 1;  // Base case ถ้า n = 0, ให้คืนค่า 1
    } else {
        return n * factorial(n - 1);  // Recursive case = เรียกฟังก์ชันตัวเองโดยลดค่า n ลง 1
    }
}
```

ในตัวอย่างนี้ ฟังก์ชัน factorial จะทำการเรียกตัวเองเพื่อคำนวณค่าของแฟกทอเรียล (Factorial) โดยมีเงื่อนไขหยุดเมื่อ n == 0

ฟังก์ชันแบบเรียกตัวเอง (Recursive Function)

อธิบายการทำงานของการหาค่าของ factorial ด้วยฟังก์ชันแบบเรียกตัวเอง




```
int factorial(int n)
{
    if (n == 0)
        return 1;
    else
        return n * factorial(n - 1);
}
```

Step 1 : factorial(5); ไปเรียก 5 x factorial(4);
Step 2 : factorial(4); ไปเรียก 4 x factorial(3);
Step 3 : factorial(3); ไปเรียก 3 x factorial(2);
Step 4 : factorial(2); ไปเรียก 2 x factorial(1);
Step 5 : factorial(1); return 1
Step 6 : 1 ที่ return มา จะไปคูณกับ 2 ใน Step 4 โดยจะ return 2
Step 7 : 2 ที่ return มา จะไปคูณกับ 3 ใน Step 3 โดยจะ return 6
Step 8 : 6 ที่ return มา จะไปคูณกับ 4 ใน Step 2 โดยจะ return 24
Step 9 : 24 ที่ return มา จะไปคูณกับ 5 ใน Step 1 โดยจะ return 120

ฟังก์ชันแบบเรียกตัวเอง (Recursive Function)

จงเขียนฟังก์ชันสำหรับหาค่า F(x) ซึ่งมีสมการดังนี้

F(x) = 0 ถ้า x เท่ากับ 0

= 1 ถ้า x เท่ากับ 1

= F(x-1) + F(x-2) ถ้า x มากกว่า 1

กำหนดให้ x และ F(x) เป็นจำนวนเต็ม และให้ตั้งชื่อฟังก์ชันว่า fib (อนุกรม Fibonacci)

ฟังก์ชันแบบเรียกตัวเอง (Recursive Function)

จงเขียนฟังก์ชันสำหรับหาค่า F(x) ซึ่งมีสมการดังนี้

F(x) = 0 ถ้า x เท่ากับ 0

= 1 ถ้า x เท่ากับ 1

= F(x-1) + F(x-2) ถ้า x มากกว่า 1

กำหนดให้ x และ F(x) เป็นจำนวนเต็ม และให้ตั้งชื่อฟังก์ชันว่า fib (อนุกรม Fibonacci)

```
int fib(int x)
{
  if(x == 0)
    return 0;
  else if(x == 1)
    return 1;
  else if(x > 1)
    return fib(x-1) + fib(x-2);
}
```

ฟังก์ชันแบบเรียกตัวเอง (Recursive Function)

การใช้งานฟังก์ชันแบบเรียกตัวเอง

ฟังก์ชันแบบเรียกตัวเองมักถูกนำไปใช้ในปัญหาที่สามารถแบ่งเป็นปัญหาย่อยๆ และแก้ไขได้ด้วยวิธีการเดิมซ้ำๆ ตัวอย่างการใช้งานที่พบบ่อยๆ ได้แก่

- การหาค่าแฟกทอเรียล
- การหาลำดับฟีโบนัชชี
- การทำงานเกี่ยวกับโครงสร้างข้อมูลต้นไม้ (Tree Traversal)
- การแก้ปัญหาแบบแบ่งครึ่ง เช่น Binary Search
- การแก้ปัญหาแบบ Backtracking เช่น การเดินหมากรุก หรือปัญหา N-Queens

ประโยชน์ของฟังก์ชันแบบเรียกตัวเอง

ความเรียบง่ายและกระชับ: ฟังก์ชันแบบเรียกตัวเองมักทำให้โค้ดสั้นและเข้าใจได้ง่าย โดยเฉพาะเมื่อปัญหาที่เราพยายามแก้มีโครงสร้างแบบเรียกซ้ำ (Recursive structure)

เหมาะกับปัญหาที่แบ่งได้เป็นส่วนย่อยๆ: เช่น การทำงานกับโครงสร้างข้อมูลแบบ Tree, Graph ที่มีโครงสร้างแบบเป็นลำดับขั้น (hierarchical)

ขอบเขตของตัวแปร (Scope of Variable)

ขอบเขตของตัวแปร (Scope of Variable)

ตัวแปรภายใน (Local Variable) เป็นตัวแปรที่ถูกใช้งานเฉพาะภายในฟังก์ชันหรือบล็อค (ภายในเครื่องหมาย {}) ที่มันถูกประกาศ ทำให้ไม่สามารถเข้าถึงได้จากส่วนอื่นของโปรแกรม จะรู้จักเฉพาะในฟังก์ชันนั้นๆ

ตัวแปรภายนอก (Global Variable) ถูกประกาศนอกฟังก์ชัน (รวมถึงฟังก์ชัน main) โดยจะรู้จัก สามารถเข้าถึง และแก้ไขในทุกฟังก์ชันที่เขียนถัดจากการประกาศตัวแปร

ตัวแปรภายใน (Local Variable)

ตัวแปรภายใน (Local Variable) หมายถึง ตัวแปรที่ถูกประกาศและใช้งานภายในฟังก์ชัน, บล็อคโค้ด, หรือขอบเขตการทำงานใดๆ (Scope) ที่จำกัด ตัวแปรประเภทนี้จะมีอายุการใช้งานเฉพาะในขอบเขตที่มันถูกประกาศ เมื่อฟังก์ชันหรือบล็อคโค้ดนั้นสิ้นสุดการทำงาน ตัวแปรภายในนั้นก็จะถูกทำลายไปด้วย

ลักษณะสำคัญของตัวแปรภายใน

ขอบเขตการใช้งาน(Scope): ใช้ได้เฉพาะภายในฟังก์ชันหรือบล็อคที่มันถูกประกาศ

อายุการใช้งาน(Persistance): จะถูกสร้างขึ้นเมื่อเข้าใช้งานบล็อคหรือฟังก์ชัน และจะถูกทำลายเมื่อออกจากบล็อคหรือฟังก์ชันนั้น

ความเป็นส่วนตัว(Privacy): ตัวแปรภายในไม่สามารถถูกเข้าถึงได้จากฟังก์ชันหรือโค้ดอื่นๆ ที่อยู่นอกขอบเขตของมัน

ตัวแปรภายใน (Local Variable)

ตัวอย่างตัวแปรภายใน

ในตัวอย่างนี้ localVar ถูกประกาศภายในฟังก์ชัน function() ทำให้ไม่สามารถถูกเข้าถึงได้จากฟังก์ชัน main()

```
#include <stdio.h>
void function() {
    int localVar = 10; // ประกาศตัวแปรภายใน
    printf("Local variable inside function: %d\n", localVar);
}
int main() {
    function();
    // printf("%d", localVar); // ถ้าเรียกใช้จะเกิดข้อผิดพลาด เพราะ localVar อยู่นอกขอบเขต
}
```


ตัวแปรภายนอก (Global Variable)

ตัวแปรภายนอก (Global Variable) หมายถึง ตัวแปรที่ถูกประกาศนอกฟังก์ชันใดๆ และสามารถถูกเข้าถึงได้จากทุกฟังก์ชันภายในโปรแกรม ตัวแปรประเภทนี้มีอายุการใช้งานตลอดช่วงเวลาที่โปรแกรมทำงาน และสามารถถูกแก้ไขค่าจากฟังก์ชันต่างๆ ในโปรแกรม

ลักษณะสำคัญของตัวแปรภายนอก

ขอบเขตการใช้งาน(Scope): สามารถเข้าถึงได้จากทุกฟังก์ชันภายในโปรแกรม

อายุการใช้งาน(Persistance): มีอายุการใช้งานตลอดช่วงเวลาที่โปรแกรมทำงาน (จนกว่าโปรแกรมจะสิ้นสุด)

ความเป็นส่วนตัว(Privacy): สามารถใช้เพื่อเก็บค่าที่ต้องการแบ่งปันระหว่างฟังก์ชันต่างๆ ภายในโปรแกรมได้อย่างสะดวก แต่อาจจะถูกเปลี่ยนแปลงจากที่ใดก็ได้ อาจจะทำให้เกิดความผิดพลาดได้งาน

ตัวแปรภายนอก (Global Variable)

ในตัวอย่างนี้ globalVar ถูกประกาศเป็นตัวแปรภายนอก ทำให้สามารถถูกใช้งานและแก้ไขค่าได้จากฟังก์ชันต่างๆ ภายในโปรแกรม

```
#include <stdio.h>
int globalVar = 20; // ประกาศตัวแปรภายนอก
void function1() {
    printf("Global variable in function1: %d\n", globalVar);
}
void function2() {
    globalVar = 30; // แก้ไขค่าตัวแปรภายนอก
    printf("Global variable in function2: %d\n", globalVar);
}
int main() {
    function1();
    function2();
    printf("Global variable in main: %d\n", globalVar); // ค่าเปลี่ยนไปตามที่แก้ใน function2
}
```


ข้อแตกต่างระหว่างตัวแปรภายในและตัวแปรภายนอก

คุณสมบัติ	ตัวแปรภายใน (Local Variable)	ตัวแปรภายนอก (Global Variable)
ขอบเขตการใช้งาน	ใช้ได้เฉพาะในฟังก์ชันหรือบล็อคที่ประกาศ	ใช้ได้ทุกที่ในโปรแกรม
อายุการใช้งาน	มีอายุการใช้งานเฉพาะเมื่ออยู่ในบล็อคหรือฟังก์ชัน	มีอายุการใช้งานตลอดช่วงเวลาของโปรแกรม
การเข้าถึง	เข้าถึงได้เฉพาะภายในฟังก์ชันที่ประกาศ	เข้าถึงได้จากทุกฟังก์ชันในโปรแกรม
การใช้งานในโปรแกรมขนาดใหญ่	ป้องกันการแก้ไขโดยไม่ได้ตั้งใจ เพราะเข้าถึงได้เฉพาะภายใน	อาจทำให้เกิดปัญหาได้หากมีการแก้ไขจากหลายฟังก์ชัน

ตัวอย่าง 1: ตัวแปรภายในและตัวแปรภายนอก

```
#include <stdio.h>
void my_func();
int main()
{
  double x = 1.1;
  my_func();
  printf("In main, x = %d \n",x);
}
void my_func()
{
  double x;
  x = 2.5;
  printf("In my_func, x = %d \n",x);
}
```

Output:

In my_func, x = 2.5

In main, x = 1.1

ตัวอย่าง 2: ตัวแปรภายในและตัวแปรภายนอก

```
#include <stdio.h>
double x;
void my_func();
int main()
{
  x = 1.1;
  my_func();
  printf("In main, x = %d \n",x);
}
void my_func()
{
  x = 2.5;
  printf("In main, x = %d \n",x);
}
```

Output:

In my_func, x = 2.5

In main, x = 2.5

ทดสอบความเข้าใจเรื่องขอบเขตของตัวแปร

```
#define MAX 950
void one(int anarg,
                double second)
{
   int onelocal;
   ...
}
#define LIMIT 200
int fun_two(int one, char anarg)
{               int localvar;
   ...
}
int main(void)
{               int localvar;
                ...
}
```

ตัวแปร	รู้จักใน one	รู้จักใน fun_two	รู้จักใน main
MAX	✓	✓	✓
anarg(int)	✓	✗	✗
second	✓	✗	✗
onelocal	✓	✗	✗
LIMIT	✗	✓	✓
one (parameter)	✗	✓	✗
anarg(char)	✗	✓	✗
localvar(fun_two)	✗	✓	✗
localvar(main)	✗	✗	✓

Computer Programming 77

ทดสอบความเข้าใจเรื่องขอบเขตของตัวแปร

```
#define MAX 950
void one(int anarg,
                double second)
{
   int onelocal;
   ...
}
#define LIMIT 200
int fun_two(int one, char anarg)
{               int localvar;
   ...
}
int main(void)
{               int localvar;
                ...
}
```

ฟังก์ชัน	รู้จักใน one	รู้จักใน fun_two	รู้จักใน main
one()	✓	✓	✓
fun_two()	✗	✓	✓

Computer Programming 78

เอกสารอ้างอิง

คณาจารย์ผู้สอนวิชา Introduction to Computer Programming, เอกสารประกอบการบรรยายวิชา 241-101 Introduction to Computer Programming, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยสงขลานครินทร์, 2551

นราธร สังข์ประเสริฐ, เอกสารประกอบการบรรยายวิชา 04-511-101 การโปรแกรมคอมพิวเตอร์, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย , 2558

แหล่งศึกษาเพิ่มเติม

ผศ.นราธร สังข์ประเสริฐ. (2563) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-3945-2

โอภาส เอี่ยมสิริวงศ์. (2559) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-2740-4

รศ.ธีรวัฒน์ ประกอบผล. (2562) คู่มือการเขียนโปรแกรมภาษา C ฉบับสมบูรณ์, ซิมพลิฟาย, ISBN: 978-616-26-24896

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase1] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=aDBmXMF9jOU

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase2] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=wslM0Aqepxs

Mooc การเขียนโปรแกรมคอมพิวเตอร์พื้นฐาน ด้วยภาษาซี 1 | Fundamentals of the C Programming Language https://lms.thaimooc.org/courses/course-v1:KMUTT+KMUTT013+2019/about

Mooc ทักษะการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี | Computer programming skills with C language https://lms.thaimooc.org/courses/course-v1:SU+SU011+2021/about

Coursera: Programming in C, ผู้สอน: Charles Russell Severance https://www.coursera.org/learn/programming-c

Coursera: ความเชี่ยวชาญพิเศษ การเขียนโปรแกรม C เบื้องต้น, ผู้สอน: Andrew D. Hilton et al. https://www.coursera.org/specializations/c-programming

Playlist Youtube บันทึกการสอนในห้องเรียน, วิชาการโปรแกรมคอมพิวเตอร์, ดร.พงศกร เจริญเนตรกุล 2566 https://www.youtube.com/watch?v=ewEfmiXaRi0&list=PL322g2itJn4mAE1utyOzF-C0SP0BRkej3&pp=gAQBiAQB

แบบฝึกหัดท้ายบท

1. ฟังก์ชันในภาษา C คืออะไร
2. ฟังก์ชันช่วยในการแบ่งโปรแกรมออกเป็นส่วนย่อย ๆ ได้อย่างไร
3. ประโยชน์หลักของการใช้ฟังก์ชันในโปรแกรมมีอะไรบ้าง
4. ฟังก์ชันแบบใดที่ไม่คืนค่าในภาษา C
5. ฟังก์ชันที่รับค่าพารามิเตอร์แต่ไม่คืนค่าจะใช้คำใดบอกว่าไม่มี
6. การใช้ฟังก์ชันช่วยลดการเขียนโค้ดซ้ำได้อย่างไร
7. ตัวแปรภายใน (Local Variable) คืออะไร
8. ตัวแปรภายนอก (Global Variable) ต่างจากตัวแปรภายในอย่างไร
9. ฟังก์ชันมาตรฐานในภาษา C มาจากที่ไหน
10. ยกตัวอย่างฟังก์ชันจาก C Standard Library ที่เกี่ยวกับการคำนวณทางคณิตศาสตร์
11. อะไรคือฟังก์ชันแบบ Recursive Function
12. Recursive Function ทำงานอย่างไร
13. โปรแกรมควรประกาศฟังก์ชันก่อน main หรือไม่
14. Function Prototype มีประโยชน์อย่างไรในการเขียนโปรแกรม
15. อะไรคือ Single Responsibility Principle (SRP)
16. ฟังก์ชันที่มีการคืนค่าจากการคำนวณเรียกว่าอะไร
17. การเรียกฟังก์ชันด้วยการส่งพารามิเตอร์มีรูปแบบอย่างไร
18. ฟังก์ชันแบบใดที่ไม่รับค่าและไม่คืนค่า
19. อธิบายวิธีการใช้ฟังก์ชัน sqrt() จากไลบรารี math.h
20. การประกาศฟังก์ชันในภาษา C มีกี่แบบ
21. ตัวอย่างการเรียกฟังก์ชันที่มีพารามิเตอร์รับค่าอย่างน้อย 2 ตัวคืออะไร
22. การส่งค่าเข้าและรับค่ากลับจากฟังก์ชันมีรูปแบบอย่างไร

23. อธิบายการทำงานของฟังก์ชัน printf()
24. อธิบายการทำงานของฟังก์ชัน scanf()
25. อะไรคือโครงสร้างของฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรม
26. อะไรคือเงื่อนไขในการหยุดทำงานของฟังก์ชันแบบ Recursive Function
27. อธิบายการทำงานของฟังก์ชัน factorial แบบ Recursive
28. ฟังก์ชันแบบใดที่ไม่รับพารามิเตอร์แต่มีการส่งค่ากลับ
29. ฟังก์ชันใดที่ถูกเรียกใช้งานเสมอเมื่อโปรแกรมภาษา C เริ่มทำงาน
30. การทำงานของตัวแปรภายในและภายนอกมีผลอย่างไรต่อการพัฒนาโปรแกรมขนาดใหญ่

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 6.1

เรื่อง Function My Name

วัตถุประสงค์

1. สามารถเขียนฟังก์ชันต้นแบบอย่างถูกต้อง
2. สามารถเขียนนิยามฟังก์ชันได้อย่างถูกต้อง
3. สามารถเรียกใช้งานฟังก์ชันได้อย่างถูกต้อง

ทบทวนความรู้

ตัวอย่างโครงสร้างของฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรม

```
#include <stdio.h>
void func2();
void func1();
int main() {
  func2();
  printf("3 ");
  return 0;
}
void func2() {
  func1();
  printf("2 ");
}
void func1() {
  printf("1 ");
}
```

ส่วนของการ include header file(s) หรือ Preprocessor

ส่วนที่ประกาศว่ามีฟังก์ชันอะไรบ้างที่สร้างขึ้นใช้ในโปรแกรมนี้ (Function Prototype)

โปรแกรมหลัก (Main Function)

โครงสร้างที่ฟังก์ชันนิยามหลังการเรียกใช้

จำเป็นต้องมี ต้นแบบฟังก์ชัน

ส่วนการนิยามรายละเอียดการทำงานของฟังก์ชัน func2 (Function Definition)

ส่วนการนิยามรายละเอียดการทำงานของฟังก์ชัน func1 (Function Definition)

Computer Programming 33

คำสั่ง

จงสร้างฟังก์ชันแสดงชื่อตนเอง โดยฟังก์ชันมีลักษณะดังต่อไปนี้

1. ไม่มีการคืนค่า (void)
2. ฟังก์ชันชื่อ myName
3. ฟังก์ชันไม่รับพารามิเตอร์ใดๆ โดยฟังก์ชันแสดงชื่อของนักศึกษาเอง

4.เขียนฟังก์ชันต้นแบบ (Function Prototype) ด้วย

5.เรียกใช้งานฟังก์ชัน myName เพื่อแสดงชื่อของนศ.3 x 10 = 30

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน.....

ใบฝึกปฏิบัติการที่ 6.2

เรื่อง Function My Name หลายๆครั้งเด้อ

วัตถุประสงค์

1. สามารถเขียนฟังก์ชันต้นแบบอย่างถูกต้อง
2. สามารถเขียนนิยามฟังก์ชันได้อย่างถูกต้อง
3. สามารถเรียกใช้งานฟังก์ชันได้อย่างถูกต้อง
4. สามารถเขียนคำสั่ง loop ในนิยามฟังก์ชันได้
5. สามารถส่งพารามิเตอร์ระหว่างฟังก์ชันได้

ทบทวนความรู้

ตัวอย่างโครงสร้างของฟังก์ชันที่สร้างโดยผู้เขียนโปรแกรม

```
#include <stdio.h>
void func2();
void func1();
int main() {
  func2();
  printf("3 ");
  return 0;
}
void func2() {
  func1();
  printf("2 ");
}
void func1() {
  printf("1 ");
}
```

ส่วนของการ include header file(s) หรือ Preprocessor

ส่วนที่ประกาศว่ามีฟังก์ชันอะไรบ้างที่สร้างขึ้นใช้ในโปรแกรมนี้ (Function Prototype)

โปรแกรมหลัก (Main Function)

โครงสร้างที่ฟังก์ชันนิยามหลังการเรียกใช้

จำเป็นต้องมี ต้นแบบฟังก์ชัน

ส่วนการนิยามรายละเอียดการทำงานของฟังก์ชัน func2 (Function Definition)

ส่วนการนิยามรายละเอียดการทำงานของฟังก์ชัน func1 (Function Definition)

Computer Programming 33

คำสั่ง

จงเขียนฟังก์ชันแสดงชื่อตัวเอง หลายครั้ง โดยรับจำนวนครั้งมาจากผู้ใช้

โดยลักษณะฟังก์ชันเป็นดังนี้

1. ไม่ส่งค่าใดๆ กลับ (void)
2. ชื่อฟังก์ชัน คือ myNamesss
3. รับพารามิเตอร์เป็นจำนวนครั้งที่ต้องการพิมพ์ เป็นชนิดจำนวนเต็ม
4. ใช้ for loop สำหรับการทำซ้ำ

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

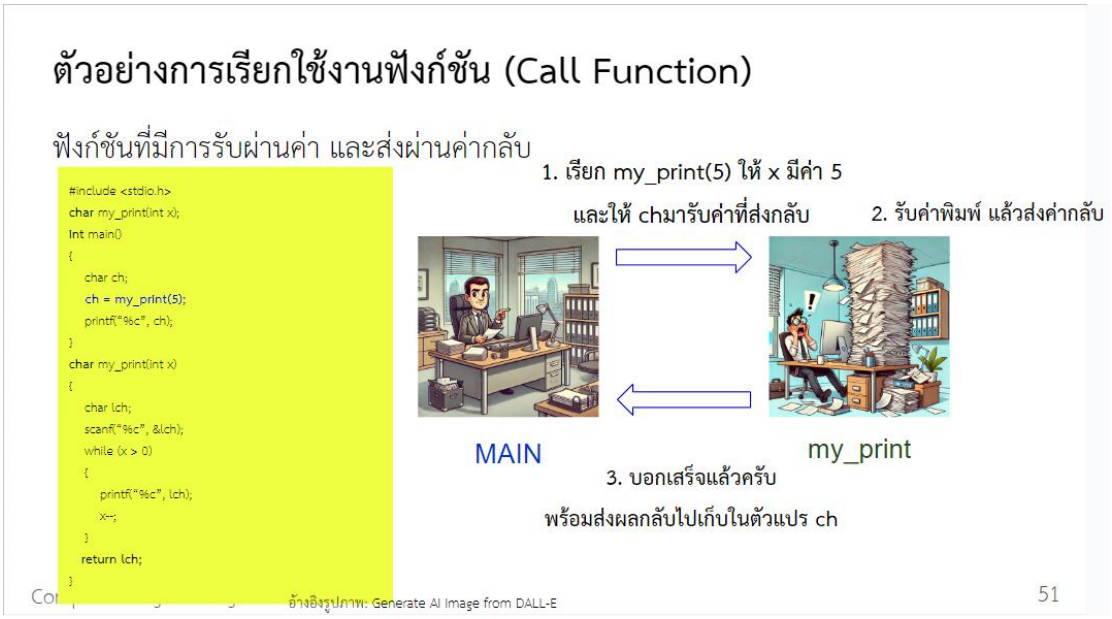
ใบฝึกปฏิบัติการที่ 6.3

เรื่อง Inch to cm

วัตถุประสงค์

1. สามารถเขียนฟังก์ชันต้นแบบอย่างถูกต้อง
2. สามารถเขียนนิยามฟังก์ชันได้อย่างถูกต้อง
3. สามารถเรียกใช้งานฟังก์ชันได้อย่างถูกต้อง
4. สามารถเขียนคำสั่ง loop ในนิยามฟังก์ชันได้
5. สามารถส่งพารามิเตอร์ระหว่างฟังก์ชันได้
6. สามารถคืนค่าไปยังส่วนที่เรียกใช้งานฟังก์ชันได้

ทบทวนความรู้



คำสั่ง

จงเขียนโปรแกรมเพื่อรับค่านิ้ว (โดยรับค่าจากพิมพ์คีย์บอร์ด) เพื่อแปลงเป็น cm โดยเรียกใช้ฟังก์ชันเพื่อแปลงหน่วย inch เป็น cm โดยกำหนดให้ 1 inch = 2.54 cm

1. ฟังก์ชัน ส่งค่ากลับเป็น cm ชนิดข้อมูลเป็น ทศนิยม
2. ฟังก์ชัน ชื่อ inch2cm
3. ฟังก์ชั่น รับพารามิเตอร์เป็น inch(นิ้ว) ชนิดข้อมูลเป็น ทศนิยม

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 6.4

เรื่อง เขียนฟังก์ชันต้นแบบ (function prototype)

วัตถุประสงค์

1. สามารถเขียนฟังก์ชันต้นแบบอย่างถูกต้อง
2. สามารถเขียนนิยามฟังก์ชันได้อย่างถูกต้อง
3. สามารถทดสอบความถูกต้องของฟังก์ชันต้นแบบและนิยามฟังก์ชันได้อย่างถูกต้อง

ทบทวนความรู้

1. ต้นแบบฟังก์ชัน (Function Prototype)

โครงสร้างต้นแบบฟังก์ชัน

```
return-type  function-name (parameter-list) ;
```

return-type หรือชนิดข้อมูลที่จะส่งค่ากลับได้แก่ void, int, double, char, ...

function-name ชื่อของฟังก์ชัน

parameter-list จำนวนพารามิเตอร์ที่ฟังก์ชันต้องการ (Input ซึ่งเป็นตัวแปรที่ต้องการส่งไปให้ฟังก์ชันคำนวณ/ประมวลผล)

- แต่ละพารามิเตอร์ประกอบด้วย ชนิดตัวแปรและชื่อตัวแปร
- แต่ละพารามิเตอร์แยกด้วยเครื่องหมาย " , "

ตัวอย่าง `int add (int a, int b) ;`

Computer Programming 29

2. นิยามฟังก์ชัน (Function Definition)

ตัวอย่างฟังก์ชันที่มี return

```
int add (int a, int b)
{
      int result;
      result = a+b;
      return result;
}
```

ตัวอย่างฟังก์ชันที่ไม่มี return

```
void  checkresult (int result)
{
   if(result < 0)
         printf("result is negative");
   if(result == 0)
         printf("result is zero");
   if(result > 0)
         printf("result is positive")
}
```

***ถ้า return-type ไม่ใช่ void ต้องมีการส่งค่ากลับเสมอ โดยใช้คำสั่ง return ตามด้วยค่าที่จะส่งกลับ

Computer Programming 32

คำสั่ง

จงเขียนฟังก์ชันต้นแบบ (function prototype) และ นิยามฟังก์ชัน (function definition) ตามโจทย์ต่อไปนี้

1. ฟังก์ชันคำนวณพื้นที่วงกลม

คืนค่า float

ชื่อ CircleCal

พารามิเตอร์

1. รัศมี ชนิด float

สูตรพื้นที่วงกลม

พื้นที่ = PI * (รัศมี ยกกำลัง 2)

กำหนดให้ค่า PI = 3.14159

2. ฟังก์ชันคำนวณพื้นที่สี่เหลี่ยมผืนผ้า

คืนค่า float

ชื่อ RectCal

พารามิเตอร์

1. ด้านที่ 1 ชนิด float
2. ด้านที่ 2 ชนิด float

สูตรพื้นสี่เหลี่ยมผืนผ้า

พื้นที่ = ด้านที่ 1 * ด้านที่ 2

3. ฟังก์ชันคำนวณพื้นที่สี่เหลี่ยมด้านขนาน

คืนค่า float

ชื่อ RectParaCal

พารามิเตอร์

1. ฐาน ชนิด float
2. สูง ชนิด float

4. ฟังก์ชันคำนวณพื้นที่สามเหลี่ยม

คืนค่า float

ชื่อ TriangleCal

พารามิเตอร์

1. ฐาน ชนิด float
2. สูง ชนิด float

พร้อมทั้งทดสอบการเรียกใช้งานฟังก์ชันทั้งหมดข้างต้น

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 6.5

เรื่อง คำนวณพื้นที่ทั้งหมดในภาพโดยการเรียกใช้งานฟังก์ชัน

วัตถุประสงค์

1. สามารถเขียนฟังก์ชันต้นแบบอย่างถูกต้อง
2. สามารถเขียนนิยามฟังก์ชันได้อย่างถูกต้อง
3. สามารถเรียกฟังก์ชันเพื่อคำนวณค่าได้

ทบทวนความรู้

1. ต้นแบบฟังก์ชัน (Function Prototype)

โครงสร้างต้นแบบฟังก์ชัน

```
return-type  function-name (parameter-list) ;
```

return-type หรือชนิดข้อมูลที่จะส่งค่ากลับได้แก่ void, int, double, char, ...

function-name ชื่อของฟังก์ชัน

parameter-list จำนวนพารามิเตอร์ที่ฟังก์ชันต้องการ (Input ซึ่งเป็นตัวแปรที่ต้องการส่งไปให้ฟังก์ชันคำนวณ/ประมวลผล)

- แต่ละพารามิเตอร์ประกอบด้วย ชนิดตัวแปรและชื่อตัวแปร
- แต่ละพารามิเตอร์แยกด้วยเครื่องหมาย " , "

ตัวอย่าง `int add (int a, int b) ;`

Computer Programming 29

2. นิยามฟังก์ชัน (Function Definition)

ตัวอย่างฟังก์ชันที่มี return

```
int add (int a, int b)
{
      int result;
      result = a+b;
      return result;
}
```

ตัวอย่างฟังก์ชันที่ไม่มี return

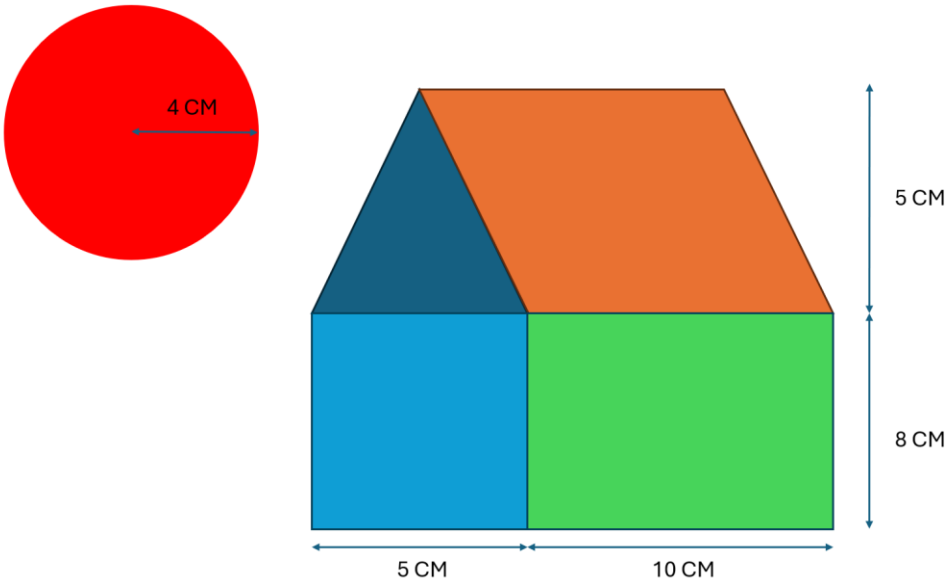
```
void  checkresult (int result)
{
   if(result < 0)
         printf("result is negative");
   if(result == 0)
         printf("result is zero");
   if(result > 0)
         printf("result is positive")
}
```

***ถ้า return-type ไม่ใช่ void ต้องมีการส่งค่ากลับเสมอ โดยใช้คำสั่ง return ตามด้วยค่าที่จะส่งกลับ

Computer Programming 32

คำสั่ง

คำนวณพื้นที่เฉพาะ 2 มิติของบ้านและวงกลม(แยกกัน) โดยการเรียกใช้งานฟังก์ชันที่สร้างขึ้นในใบฝึกปฏิบัติการที่ 6.4



ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 6.6

เรื่อง ฟังก์ชันยกกำลัง

วัตถุประสงค์

1. สามารถเขียนฟังก์ชันต้นแบบอย่างถูกต้อง
2. สามารถเขียนนิยามฟังก์ชันได้อย่างถูกต้อง
3. สามารถเรียกฟังก์ชันเพื่อคำนวณค่าได้
4. สามารถเรียกใช้ loop เพื่อคำนวณค่าที่ซับซ้อนได้

ทบทวนความรู้

1. ต้นแบบฟังก์ชัน (Function Prototype)

โครงสร้างต้นแบบฟังก์ชัน

```
return-type  function-name (parameter-list) ;
```

return-type หรือชนิดข้อมูลที่จะส่งค่ากลับได้แก่ void, int, double, char, ...

function-name ชื่อของฟังก์ชัน

parameter-list จำนวนพารามิเตอร์ที่ฟังก์ชันต้องการ (Input ซึ่งเป็นตัวแปรที่ต้องการส่งไปให้ฟังก์ชันคำนวณ/ประมวลผล)

- แต่ละพารามิเตอร์ประกอบด้วย ชนิดตัวแปรและชื่อตัวแปร
- แต่ละพารามิเตอร์แยกด้วยเครื่องหมาย " , "

ตัวอย่าง

```
int add (int a, int b) ;
```

2. นิยามฟังก์ชัน (Function Definition)

ตัวอย่างฟังก์ชันที่มี return

```
int add (int a, int b)
{
        int result;
        result = a+b;
        return result;
}
```

ตัวอย่างฟังก์ชันที่ไม่มี return

```
void  checkresult (int result)
{
   if(result < 0)
            printf("result is negative");
   if(result == 0)
            printf("result is zero");
   if(result > 0)
            printf("result is positive")
}
```

***ถ้า return-type ไม่ใช่ void ต้องมีการส่งค่ากลับเสมอ โดยใช้คำสั่ง return ตามด้วยค่าที่จะส่งกลับ

Computer Programming 32

คำสั่ง

จงเขียนฟังก์ชันยกกำลังเลียนแบบ math.h โดยสามารถคำนวณตัวชี้กำลัง (exponent) ที่เป็นจำนวนเต็มบวก ศูนย์ และจำนวนเต็มลบได้ โดยมีฟังก์ชันต้นแบบดังนี้ double pow(double base,int exp);

แผนการสอนเรื่อง อาร์เรย์

สัปดาห์ที่ 13-15

จำนวนชั่วโมงที่ใช้สอน 15 ชั่วโมง

วัตถุประสงค์ เพื่อให้นักศึกษาสามารถ

1. บอกความหมายของอาร์เรย์และความแตกต่างระหว่างตัวแปรอาร์เรย์กับตัวแปรธรรมดาได้
2. อธิบายวิธีการประกาศและการเข้าถึงข้อมูลในตัวแปรอาร์เรย์แบบ 1 มิติและหลายมิติได้อย่างถูกต้อง
3. เขียนโค้ดโปรแกรมที่ใช้ตัวแปรอาร์เรย์ในการจัดเก็บและประมวลผลข้อมูล เช่น การคำนวณผลรวมและค่าเฉลี่ยในอาร์เรย์
4. เปรียบเทียบการส่งผ่านตัวแปรอาร์เรย์ไปยังฟังก์ชันแบบ Call-by-value และ Call-by-reference ได้
5. ตรวจสอบความถูกต้องของโปรแกรมที่ใช้อาร์เรย์ในสถานการณ์ต่าง ๆ เช่น การเข้าถึงค่าที่อยู่นอกขอบเขตของอาร์เรย์
6. ออกแบบโปรแกรมที่ใช้ตัวแปรอาร์เรย์หลายมิติสำหรับการจัดเก็บและประมวลผลข้อมูลเชิงซับซ้อนได้ เช่น ตารางข้อมูลหรือเมทริกซ์

หัวข้อบรรยาย

บทที่ 7 อาร์เรย์

7.1 นิยามของอาร์เรย์

7.2 ตัวแปรอาร์เรย์แบบ 1 มิติ

7.3 การเข้าถึงค่าในอาร์เรย์

7.4 การกำหนดค่าเริ่มต้นให้กับอาร์เรย์

7.5 การส่งผ่านอาร์เรย์ไปยังฟังก์ชัน

7.6 ตัวแปรอาร์เรย์แบบหลายมิติ

7.7 สายอักขระ (String)

7.8 ข้อผิดพลาดที่พบบ่อยในการใช้งานอาร์เรย์

กิจกรรมการเรียนการสอน

1. การบรรยายในชั้นเรียน
2. การให้ผู้เรียนทดลองเขียนรหัสต้นฉบับและทดลองรันโปรแกรมระหว่างบรรยาย
3. การให้ผู้เรียนทำใบฝึกปฏิบัติการ
4. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)

สื่อการสอน

1. เอกสารประกอบการสอนแบบภาพเลื่อนหรือภาพนิ่ง
2. ใบฝึกปฏิบัติการ
3. โปรแกรมพัฒนาโปรแกรมภาษาซี เช่น DevC++, VS Code เป็นต้น
4. การเข้าถึงข้อมูลจากอินเทอร์เน็ต

การประเมินผลการเรียนรู้

ผลการเรียนรู้	วิธีการประเมิน	สัปดาห์ที่ประเมิน	สัดส่วนของการประเมินผล
1. คุณธรรมจริยธรรม	1.1 ประเมินจากการเข้าชั้นเรียน การส่งแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการตามกำหนดเวลาที่มอบหมาย 1.2 ประเมินจากพฤติกรรมของนักศึกษาในการปฏิบัติตามกฎระเบียบและข้อบังคับต่างๆ ทั้งในข้อบังคับในชั้นเรียน ข้อบังคับของมหาวิทยาลัย 1.3 ประเมินจากการคัดลอกแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการ 1.4 ประเมินจากการมีส่วนร่วมในห้องเรียนและการปิดคอมพิวเตอร์ เก็บขยะ เก็บเก้าอี้หลังจากใช้งานเสร็จ	ตลอดภาคการศึกษา	10%
2. ความรู้	2.1 แบบฝึกหัดท้ายบท 2.2 การสอบกลางภาคเรียนและปลายภาคเรียน	2-8, 11-16 9, 17	20% 40%
3. ทักษะทางปัญญา	3.1 ประเมินจากใบฝึกปฏิบัติการ	ตลอดภาคการศึกษา	20%
4. ทักษะความสัมพันธ์ระหว่างบุคคลและความรับผิดชอบ	4.1 พิจารณาจากการทำใบฝึกปฏิบัติการ 4.3 ประเมินจากพฤติกรรมในการทำงานกลุ่ม การระดมความคิดเห็น การตอบคำถามในชั้นเรียน	ตลอดภาคการศึกษา	5%
5. ทักษะวิเคราะห์เชิงตัวเลข การสื่อสารและการใช้เทคโนโลยีสารสนเทศ	5.1 การทดสอบหรือการประเมินจากงานที่มอบหมาย 5.2 สังเกตการใช้คอมพิวเตอร์ในการเขียนโปรแกรม การทำใบฝึกปฏิบัติการ และการค้นหาข้อมูลเพิ่มเติมจากอินเทอร์เน็ต	ตลอดภาคการศึกษา	5%
	รวม		100%

อ่าน Slide เวอร์ชันล่าสุดได้ที่ Google Slide

https://s.coecore.com/scp7

การโปรแกรมคอมพิวเตอร์

บทที่ 7 อาร์เรย์

(Array)

ผู้สอน ดร.พงศกร เจริญเนตรกุล

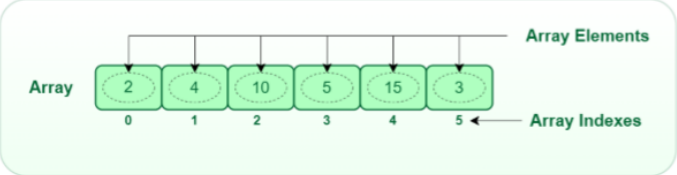
Outline

- นิยามของอาร์เรย์
- ตัวแปรอาร์เรย์แบบ 1 มิติ
- การเข้าถึงค่าในอาร์เรย์
- การกำหนดค่าเริ่มต้นให้กับอาร์เรย์
- การส่งผ่านอาร์เรย์ไปยังฟังก์ชัน
- ตัวแปรอาร์เรย์แบบหลายมิติ
- สายอักขระ (String)
- ข้อผิดพลาดที่พบบ่อยในการใช้งานอาร์เรย์

นิยามของตัวแปรชุด

นิยามของอาร์เรย์(ตัวแปรชุด)

- **ตัวแปรชุด (array อ่านว่า อาร์เรย์) ในภาษาซี** หมายถึง ชุดของตัวแปรที่มีชนิดข้อมูลเดียวกัน (ใช้ชื่อตัวแปรเดียวกัน)
- ถูกเก็บไว้ในหน่วยความจำติดกันเป็นลำดับ โดยสามารถเข้าถึงแต่ละตัวแปรในชุดได้โดยใช้ดัชนี (index) ซึ่ง**เริ่มต้นจาก 0** ตัวแปรชุดจึงช่วยให้เราสามารถจัดเก็บและจัดการข้อมูลจำนวนมากที่มีชนิดข้อมูลเหมือนกันได้สะดวก เช่น การจัดเก็บตัวเลขหลายตัวหรือชุดตัวอักษร



ที่มาของรูปภาพ https://medium.com/javarevisitedian-introduction-to-arrays-applications-advantages-and-disadvantages-c3153adfb40e

ความแตกต่างระหว่างตัวแปรชุด (array) กับตัวแปรธรรมดา (variable)

ขนาดและจำนวนข้อมูล

- **ตัวแปรธรรมดา** **จะเก็บข้อมูลได้เพียงค่าเดียว** เช่น int a = 10; ตัวแปรนี้จะเก็บค่า 10 ไว้เท่านั้น
- **ตัวแปรชุด (array)** **จะเก็บข้อมูลหลายค่า(ต้องชนิดเดียวกัน)** เช่น int a[5]; ตัวแปรชุดนี้สามารถเก็บจำนวนเต็มได้ 5 ค่า

การเข้าถึงข้อมูล

- **ตัวแปรธรรมดา** เข้าถึงได้โดยตรงโดยใช้ชื่อของตัวแปร เช่น a
- **ตัวแปรชุด** เข้าถึงแต่ละค่าผ่านดัชนี เช่น a[0], a[1] **โดยดัชนี (Index) จะเริ่มที่** 0

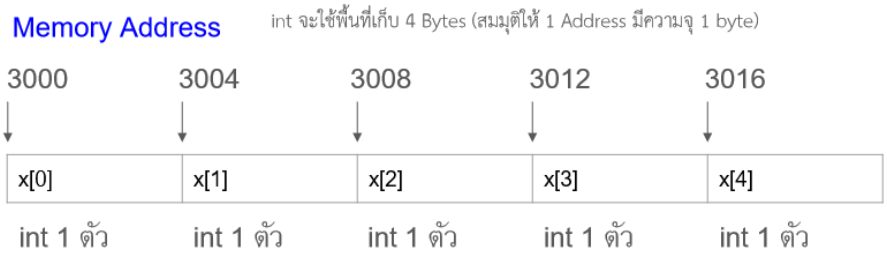
การจัดการในหน่วยความจำ

- **ตัวแปรธรรมดา** ใช้ที่อยู่ในหน่วยความจำเพียงที่เดียวสำหรับค่าเดียว
- **ตัวแปรชุด** ใช้ที่อยู่ในหน่วยความจำติดต่อกันตามขนาดของชุดข้อมูล เช่น ตัวแปรชุดแบบ int a[5] จะใช้หน่วยความจำ 5 ตำแหน่งต่อกัน

นิยามของอาร์เรย์(ตัวแปรชุด)

int x[5]; // ประกาศตัวแปร x เป็นตัวแปรชุด(อาร์เรย์) ชนิด int จำนวน 5 ช่อง

สมาชิก (Element/member) ของ x จะมี **x[0]** ถึง **x[4]**



ตัวแปรอาร์เรย์แบบ 1 มิติ

ตัวแปรอาร์เรย์แบบ 1 มิติ

- อาร์เรย์หนึ่งมิติ มีโครงสร้างเทียบเท่าเมตริกซ์ขนาด nx1
- **การประกาศตัวแปรอาร์เรย์** จะใช้เครื่องหมาย [ ] ล้อมค่าตัวเลขจำนวนเต็ม เพื่อบอกจำนวนหน่วยข้อมูลที่ต้องการได้ในรูป

  ชนิดของตัวแปร ชื่อตัวแปร[จำนวนสมาชิกที่ต้องการ];

  data_type variable_name [number-of-elements];

  เช่น int a[11];

  double x, y[10], z[3];

Note: จำนวนสมาชิกที่ต้องการ(ขนาดของอาร์เรย์) ต้องระบุเป็นค่าคงที่เท่านั้น (จำนวนนับ)

ตัวแปรอาร์เรย์แบบ 1 มิติ

ขนาดของตัวแปรอาร์เรย์ และความจุในหน่วยไบต์

เราสามารถหาความจุของอาร์เรย์ที่ถูกจัดเก็บที่หน่วยความจำ (memory) ในหน่วยไบต์ได้

โดยใช้ sizeof(ชื่อตัวแปร)

ตัวอย่างเช่น

```
int numbers[10];
printf (“%d”, sizeof (numbers) ); //หาขนาดทั้งหมด
printf (“%d”, sizeof (numbers) / sizeof(int) ); //หาจำนวนช่อง
```

Size of int = 4 bytes
Size of float = 4 bytes
Size of char = 1 bytes

ขนาดความจุ(ไบต์) ของตัวแปรชนิดอาร์เรย์ชื่อ numbers คือ 10 * 4 = 40 ไบต์

ตัวแปรอาร์เรย์แบบ 1 มิติ

```
// ex การดูขนาดอาร์เรย์ 1 มิติในหน่วยไบต์
//arrayex1.c
#include <stdio.h>
int main()
{       int ages[10];    char name[50];
        double scores[20];
        printf("Size of ages =  %d\n",sizeof(ages));
                printf("Size of name = %d\n" ,sizeof(name));
                printf("Size of scores = %d\n",sizeof(scores));
                printf("Size of name[0]=%d\n" ,sizeof(name[0]));
                printf("Size of ages[0]=%d\n",sizeof(ages[0]));
}
```

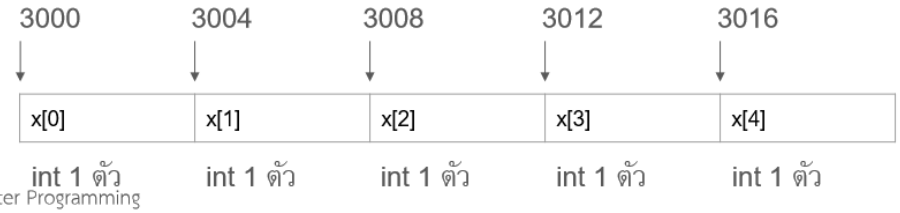
Output:
Size of ages = 40
Size of name = 50
Size of scores = 160
Size of names[0] = 1
Size of ages[0] = 4

การอ้างถึงอาร์เรย์ในภาษาซี

การอ้างชื่อของตัวแปรอาร์เรย์ พร้อม [] เช่น x[1] จะเป็นการเข้าถึงค่าใน elementต่างๆ

แต่ * การอ้างชื่อของตัวแปรอาร์เรย์เพียงอย่างเดียว เช่น x (โดยที่ประกาศ int x[5]; ไว้) จะหมายถึง ค่าตำแหน่งแอดเดรสของหน่วยความจำเริ่มต้นของกลุ่มข้อมูลอาร์เรย์**

ในตัวอย่างนี้หากเรียก x จะส่งค่า 3000 กลับมาให้ ซึ่งไม่ใช่ค่าของ x[0] และไม่ใช่ค่าสมาชิกทุกตัวในอาร์เรย์



ตัวแปรอาร์เรย์แบบ 1 มิติ

- โดยเมื่อมีการประกาศตัวแปร ระบบจะทำการจองหน่วยความจำเพื่อเก็บข้อมูล โดยจะอ้างอิงกับ Address หรือ ตำแหน่งในหน่วยความจำ ทำให้โปรแกรมเมอร์ไม่ต้องไปจัดการหน่วยความจำโดยตรง เพียงแต่ใช้ชื่อตัวแปรในการอ้างอิงแทน
- การประกาศตัวแปร จะทำให้ระบบจองหน่วยความจำสำหรับตัวแปรชื่อนั้น ๆ โดยขนาดที่จองขึ้นอยู่กับชนิดข้อมูล เช่น int a; จะจองพื้นที่ในหน่วยความจำสำหรับเก็บจำนวนเต็ม
- การใช้ scanf() จะต้องใช้เครื่องหมาย & (address-of operator) หน้าชื่อตัวแปร เช่น scanf("%d", **&a**); เพราะ scanf() ต้องการที่อยู่ของหน่วยความจำที่ตัวแปรนั้นถูกจัดเก็บ เพื่อจะนำค่าที่ป้อนเข้ามาเก็บในที่อยู่ของตัวแปรโดยตรง สรุปคือ & ใช้เพื่อบอกตำแหน่งของตัวแปรในหน่วยความจำให้ scanf() รู้ว่าจะเก็บค่าที่ไหน

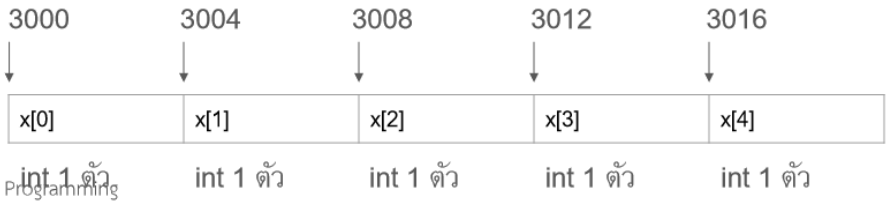
การเข้าถึงค่าในอาร์เรย์

การเข้าถึงค่าในอาร์เรย์

เมื่อมีการประกาศอาร์เรย์แล้ว ค่าตำแหน่งหมายเลขลำดับข้อมูลสำหรับใช้เข้าถึงตัวแปรย่อยต่างๆ ในอาร์เรย์ **จะถูกกำหนดโดยอัตโนมัติ**

โดยหากกำหนดอาร์เรย์ด้วยขนาด n ข้อมูล **หน่วยแรกจะมีค่าตำแหน่งลำดับแรกเป็น 0** ไปจนถึง **ข้อมูลหน่วยสุดท้าย ที่มีค่าตำแหน่งลำดับเป็น n-1**

เช่น int val[10]; **ตัวแรก -> val[0] ตัวสุดท้าย -> val[9]**



การเข้าถึงค่าในอาร์เรย์

ถ้าต้องการอ่านหรือเขียนข้อมูลในหน่วยต่างๆ ของตัวแปรอาร์เรย์ จะต้องอ้างชื่อตัวแปรตามด้วยค่าลำดับของหน่วยในกลุ่มข้อมูลอาร์เรย์ **ล้อมด้วยเครื่องหมาย [ ] ซึ่งเรียกว่า subscript (หรือดัชนี index)**

ค่าดัชนี อาจอยู่ในรูป ค่าคงที่ ของตัวแปร นิพจน์ หรือฟังก์ชันที่สามารถให้ค่า**เป็นค่าจำนวนเต็มบวก**ก็ได้ (positive integer >=0)

ของเขตของ index หรือ subscript มีค่าตั้งแต่ 0 ถึง n-1 (n คือขนาดของอาร์เรย์)

ตัวอย่างการเข้าถึงค่าในอาร์เรย์

```
#include <stdio.h>
int main() {
    int scores[5]; // ประกาศตัวแปร array ขนาด 5
    int sum, x = 2, y = 4; // ตัวแปรอื่น ๆ สำหรับใช้ในตัวอย่าง

    // กำหนดค่าให้กับสมาชิกใน array
    scores[0] = 85;
    scores[1] = scores[0] + 10;
    scores[2] = 3 * (scores[0] - 5);
    scores[3] = 70;
    scores[4] = (scores[2] + scores[3] - 8) / 2;

    // คำนวณผลรวมของทุกค่าที่อยู่ใน array
    sum = scores[0] + scores[1] + scores[2] + scores[3] + scores[4];

    // ตัวอย่างการเข้าถึง array
    printf("scores[x]: %d\n", scores[x]);       // scores[2] = 240
    printf("scores[2*x]: %d\n", scores[2*x]);   // scores[4] = 151
    printf("scores[y-x]: %d\n", scores[y-x]);   // scores[2] = 240
}
```

การเข้าถึงค่าในอาร์เรย์

สามารถหาผลรวมแทนการบวกทีละตัว (sum = scores[0] + scores[1] + scores[2] + scores[3] + scores[4]; **) ด้วยการวน Loop ได้ ดังนี้**

```
sum = 0; // กำหนดค่าตัวแปรให้เป็น  0 เพราะหากไม่กำหนดให้เป็น 0 อาจมีค่าเริ่มต้นที่ไม่ใช่ 0

int i;

for (i=0 ; i<=4 ; i++) // สามารถใช้ (i=0 ; i<5 ; i++) ได้ด้วย

{
        sum += scores[i];

}
```

ใช้ i เป็นตัวบอก index ในแต่ละการวน Loop

ตัวอย่างการเข้าถึงค่าในอาร์เรย์

```
int grades[3];
double price[6];
char code[2];
float a[5];
int count;

price[5] = 10.69; // กำหนดค่า 10.69 ให้กับสมาชิกตำแหน่งที่ 5 ของ array price

scanf("%d  %lf", &grades[0], &price[2]); // รับค่าจำนวนเต็มและจำนวนทศนิยมสองตัว (double) จากผู้ใช้แล้วเก็บใน grades[0] และ price[2]

scanf("%c", &code[0]); // รับตัวอักษรหนึ่งตัวจากผู้ใช้แล้วเก็บใน code[0]

scanf("%d %d %d", &grades[0], &grades[1], &grades[2]); // รับค่าจำนวนเต็มสามตัวจากผู้ใช้แล้วเก็บใน grades[0], grades[1], และ grades[2]

for (count = 0; count < 5; count++)
    scanf("%f", &a[count]); // รับค่าทศนิยม (float) จำนวน 5 ค่าแล้วเก็บใน array a ตั้งแต่ตำแหน่ง 0 ถึง 4
```

ข้อควรระวังในการเข้าถึงค่าในอาร์เรย์

หากคุณเข้าถึง index ของอาร์เรย์เกินกว่าที่ประกาศไว้ จะเกิด การเข้าถึงหน่วยความจำที่ไม่ถูกต้อง ซึ่งอาจนำไปสู่เหตุการณ์ต่อไปนี้

- ค่าผิดปกติ: อาจได้รับค่าที่ไม่คาดคิด เนื่องจากคุณเข้าถึงหน่วยความจำที่ไม่ได้จัดสรรให้กับอาร์เรย์นั้น ซึ่งอาจมีข้อมูลที่ไม่เกี่ยวข้องอยู่
- โปรแกรมล้มเหลว (Crash): หากระบบตรวจพบการเข้าถึงหน่วยความจำที่ไม่ได้รับอนุญาต โปรแกรมอาจหยุดทำงานทันทีหรือเกิดข้อผิดพลาด (Segmentation fault)
- ช่องโหว่ด้านความปลอดภัย: ในบางกรณี การเข้าถึงข้อมูลที่อยู่นอกอาร์เรย์อาจถูกใช้เป็นช่องโหว่ในการโจมตีโปรแกรม (เช่น buffer overflow) ทำให้ข้อมูลรั่วไหลหรือถูกแก้ไข

หากสามารถใช้ c++ ได้ ควรใช้ตัวแปร std::vector แทน เนื่องจากมีการป้องกันการเข้าถึง index ที่เกินขอบเขต เพื่อให้โปรแกรมทำงานได้ถูกต้องและปลอดภัยจากข้อผิดพลาดและช่องโหว่

การเข้าถึงค่าในอาร์เรย์ด้วยการวนรอบ for

สามารถใช้คำสั่งวนรอบ for ในการวนรอบรับค่าที่ป้อนเข้ามาและใช้ในการคำนวณ โดยการใช้ตัวแปรในการวนรอบและใช้ตัวแปรเดียวกัน เพื่อกำหนดลำดับของข้อมูลที่จะใช้ในอาร์เรย์ ดังตัวอย่างต่อไปนี้

```
#include <stdio.h>
int main() {
    int i, a[5];  // ประกาศตัวแปร i สำหรับใช้เป็นตัวนับ และ a เป็นอาร์เรย์ขนาด 5 สำหรับเก็บจำนวนเต็ม

    // ลูปสำหรับรับค่าจากผู้ใช้ 5 ค่าและเก็บในอาร์เรย์ a
    for (i = 0; i < 5; i++) {
        printf("Enter value for a[%d]:", i);  // แสดงข้อความ เพื่อขอให้ผู้ใช้ป้อนค่าของ a ในแต่ละตำแหน่ง
        scanf("%d", &a[i]);  // รับค่าจำนวนเต็มจากผู้ใช้แล้วเก็บใน a[i]
    }

    printf("Show all values\n");  // แสดงข้อความเพื่อบอกว่ากำลังจะแสดงค่าทั้งหมดในอาร์เรย์
    // ลูปสำหรับแสดงค่าทั้งหมดที่เก็บในอาร์เรย์ a
    for (i = 0; i < 5; i++) {  // แก้ไขจาก "x" เป็น "i" ในเงื่อนไขของลูป
        printf("a[%d] = %d\n", i, a[i]);  // แสดงค่าที่เก็บใน a[i] สำหรับแต่ละตำแหน่ง
    }
}
```

การเข้าถึงค่าในอาร์เรย์ด้วยการวนรอบ for

จากโปรแกรมนี้ หากใส่ 10, 20, 30, 40 จะมีผลลัพธ์เป็นอย่างไร

```
#include <stdio.h>
int main()
{
  int num[4],i;
  for(i=0;i<4;i++)
  {
    printf("Enter num[%d]: ",i);
    scanf("%d",&num[i]);
  }
  for(i=0;i<4;i++)
    num[i] = 2*num[i];
  for(i=0;i<4;i++)
    printf("num[%d] = %d\n",i,num[i]);
}
```

Input:
Enter num[0]: 10
Enter num[1]: 20
Enter num[2]: 30
Enter num[3]: 40

Output:
num[0] = 20

num[1] = 40

num[2] = 60

num[3] = 80

การกำหนดค่าเริ่มต้นให้กับอาร์เรย์

การกำหนดค่าเริ่มต้นให้กับอาร์เรย์ (Array Initialization)

การกำหนดค่าเริ่มต้นให้กับอาร์เรย์ (Array Initialization) ในภาษาซี เป็นการให้ค่าเริ่มต้นแก่สมาชิกแต่ละตัวของอาร์เรย์ในขณะที่ประกาศอาร์เรย์นั้น โดยสามารถทำได้หลายวิธีตามขนาดและความต้องการของโปรแกรม ตัวอย่างของการกำหนดค่าเริ่มต้นมีดังนี้

1. การกำหนดค่าเริ่มต้นทั้งหมดในขณะที่ประกาศ สามารถกำหนดค่าให้กับสมาชิกทุกตัวของอาร์เรย์ในขณะที่ประกาศ

ตัวอย่างเช่น int numbers[5] = {10, 20, 30, 40, 50};

numbers[0] จะมีค่าเป็น 10
numbers[1] จะมีค่าเป็น 20
numbers[2] จะมีค่าเป็น 30
numbers[3] จะมีค่าเป็น 40
numbers[4] จะมีค่าเป็น 50

การกำหนดค่าเริ่มต้นให้กับอาร์เรย์ (Array Initialization)

2. การกำหนดค่าเริ่มต้นบางส่วน

หากคุณกำหนดค่าเริ่มต้นเพียงบางส่วน ระบบจะตั้งค่าให้สมาชิกที่เหลือเป็น 0 โดยอัตโนมัติ

ตัวอย่างเช่น int numbers[5] = {10, 20};

numbers[0] จะมีค่าเป็น 10
numbers[1] จะมีค่าเป็น 20
numbers[2] จะมีค่าเป็น 0
numbers[3] จะมีค่าเป็น 0
numbers[4] จะมีค่าเป็น 0

การกำหนดค่าเริ่มต้นให้กับอาร์เรย์ (Array Initialization)

3. การกำหนดค่าเริ่มต้นโดยไม่ระบุขนาดอาร์เรย์

สามารถให้คอมไพเลอร์คำนวณขนาดของอาร์เรย์จากจำนวนสมาชิกที่กำหนดค่าเริ่มต้นไว้ได้

ตัวอย่างเช่น int numbers[] = {10, 20, 30, 40, 50};

คอมไพเลอร์จะสร้างอาร์เรย์ numbers ที่มีขนาด 5 โดยอัตโนมัติจากค่าที่กำหนด

การกำหนดค่าเริ่มต้นให้กับอาร์เรย์ (Array Initialization)

4. การกำหนดค่าเริ่มต้นสำหรับอาร์เรย์หลายมิติ

สำหรับอาร์เรย์หลายมิติ สามารถกำหนดค่าเริ่มต้นให้กับแต่ละมิติได้

ตัวอย่างเช่น int matrix[2][3] = {{1, 2, 3}, {4, 5, 6}};

matrix[0][0] จะมีค่าเป็น 1matrix[0][1] จะมีค่าเป็น 2matrix[0][2] จะมีค่าเป็น 3
matrix[1][0] จะมีค่าเป็น 4matrix[1][1] จะมีค่าเป็น 5matrix[1][2] จะมีค่าเป็น 6

ข้อควรระวัง
ขนาดของอาร์เรย์ต้องไม่เกินขอบเขตที่ระบบจัดการหน่วยความจำสามารถรองรับได้
การเข้าถึงสมาชิกที่ไม่ได้ประกาศหรืออยู่นอกขอบเขตอาร์เรย์จะทำให้เกิดข้อผิดพลาด (undefined behavior)

การกำหนดค่าเริ่มต้นให้กับอาร์เรย์ (Array Initialization)

สรุป

ในการกำหนดค่าเริ่มต้นให้กับตัวแปรอาร์เรย์นั้น สามารถกำหนดไปพร้อมกับการประกาศตัวแปรได้เช่นเดียวกันกับตัวแปรเดี่ยว แต่จะใช้กลุ่มของค่าคงที่ในการกำหนดค่าเริ่มต้น และใช้เครื่องหมาย { } ครอบกลุ่มของข้อมูล

ชนิดของตัวแปรอาร์เรย์ ชื่ออาร์เรย์[จำนวนข้อมูล] = {ค่าคงที่,ค่าคงที่,...};

ตัวอย่าง การกำหนดค่าเริ่มต้นให้กับอาร์เรย์

```
int x[5] = {12,7,0,5,-10};
```

x[0]=12	x[1]=7	x[2]=0	x[3]=5	x[4]=-10

```
char codes[6] = {'s', 'a', 'm', 'p', 'l', 'e'};

double width[7] = {10.96, 6.43, 2.58, 0.86, 5.89, 7.56, 8.22};

float temp[4] = {98.6, 97.2, 99.0 , 101.5};


int gallons[20] = { 19, 16, 14, 19, 20,
                    18, 12, 10, 22, 15,
                    18, 17, 16, 24, 23,
                    19, 15, 18, 21, 5  };
```

ตัวอย่าง การกำหนดค่าเริ่มต้นให้กับอาร์เรย์

ค่าคงที่ที่ใช้ในการกำหนดค่าเริ่มต้น จะต้องมีชนิดสอดคล้องกับชนิดอาร์เรย์ ค่าคงที่แต่ละค่าจะถูกนำไปกำหนดให้กับสมาชิกแต่ละตัวตามลำดับ

หากเราใช้ค่าคงที่จำนวนน้อยกว่าสมาชิกของอาร์เรย์ ตำแหน่งที่เหลือจะถูกกำหนดให้เป็น 0

เช่น

```
int a[5] = {12 , 7 };
```

มีค่าเท่ากับ

```
int a[5] = {12 , 7 , 0 , 0 , 0 };
```


ตัวอย่าง การกำหนดค่าเริ่มต้นให้กับอาร์เรย์

การประกาศตัวแปรแบบอาร์เรย์พร้อมทั้งกำหนดค่าเริ่มต้น สามารถทำได้โดยไม่จำเป็นต้องกำหนดขนาดของอาร์เรย์ก็ได้ คอมไพเลอร์จะหาจำนวนสมาชิกในกลุ่มค่าคงที่เอง เช่น

```
char codes[6] = {'s', 'a', 'm', 'p', 'l', 'e'};
char codes[] = {'s', 'a', 'm', 'p', 'l', 'e'};
```

ทั้งสองบรรทัดทำงานเหมือนกัน โดยคอมไพล์เลอร์จะรู้ได้เองว่าขนาดของอาร์เรย์จะมีขนาดเป็น 6 แต่ห้ามเว้นว่างไว้โดยไม่กำหนดอะไรเลย

เช่น `char codes[ ]; // <-` ห้ามทำแบบนี้

การกำหนดค่าเริ่มต้นให้กับอาร์เรย์ แบบ char []

ในกรณีให้ค่าเริ่มต้นแก่ตัวแปรอาร์เรย์แบบ char สามารถกระทำได้ โดยใช้สตริง(string)ได้ด้วยดังนี้

char code[] = "sample"; **// มีสมาชิก 7 ตัวอักษร**

คอมไพล์เลอร์จะใส่ ศูนย์ หรือ ' \0 ' หรือเรียกว่า null character ไว้ท้ายข้อความเสมอ

's'	'a'	'm'	'p'	'l'	'e'	'\0'
code[0]	code[1]	code[2]	code[3]	code[4]	code[5]	code[6]

ตัวอย่าง โปรแกรมสำหรับหาค่าต่ำสุดในอาร์เรย์ (min finding)

Output:
Minimum value is -1

```
#include <stdio.h>
int main() {
    int a[4] = {-1, 6, 9, 2};      // กำหนดค่าเริ่มต้นให้กับอาร์เรย์ a
    int i, min = a[0];                    // กำหนดให้ค่าต่ำสุดเริ่มต้นเป็นสมาชิกแรกของ
อาร์เรย์
    // วนลูปเพื่อหาค่าต่ำสุด
    for (i = 1; i < 4; i++) {
        if (a[i] < min)                   // ถ้าค่าที่ตำแหน่ง i น้อยกว่าค่าต่ำสุดในปัจจุบัน
            min = a[i];                   // ปรับค่าต่ำสุดเป็นค่าที่ตำแหน่งนั้น
    }
    printf("Minimum value is %d\n", min);  // แสดงค่าต่ำสุดที่หาได้
}
```

ตัวอย่าง โปรแกรมสำหรับหาค่าสูงสุดในอาร์เรย์ (max finding)

```
#include <stdio.h>
int main()
{
  int a[4] = {-1,6,9,2};
  int i,max = a[0];
  for(i=1;i<4;i++) {
    if(a[i] > max)
      max = a[i];
  }
  printf("Maximum value is %d\n",max);
}
```

Output:
Maximum value is 9

ตัวอย่าง โปรแกรมสำหรับหาค่าผลรวมและค่าเฉลี่ยในอาร์เรย์

```
#include <stdio.h>
int main() {
    int num[] = {1, 4, 5, 7};  // กำหนดค่าเริ่มต้นให้กับอาร์เรย์ num
    int i, total = 0;
    float average;  // ประกาศตัวแปรสำหรับเก็บค่าเฉลี่ย
    // วนลูปเพื่อหาผลรวมของสมาชิกในอาร์เรย์
    for (i = 0; i < (sizeof(num) / sizeof(int)) ; i++) {
        total = total + num[i];
    }
    // คำนวณค่าเฉลี่ย
    average = (float)total / (sizeof(num) / sizeof(int));
    printf("Sum of all elements = %d\n", total);   // แสดงผลรวม
    printf("Average value = %.2f\n", average);     // แสดงค่าเฉลี่ย (2 ตำแหน่งทศนิยม)
}
```

Output:
Sum of all elements = 17
Average value = 4.25

การส่งผ่านอาร์เรย์ไปยังฟังก์ชัน

การส่งผ่านอาร์เรย์ไปยังฟังก์ชัน

การส่งผ่านอาร์เรย์ไปยังฟังก์ชันสามารถแบ่งได้เป็น 2 ลักษณะ

1.การ**ส่งผ่านค่าอีลีเมนต์อาร์เรย์ให้กับฟังก์ชัน** เป็นการเรียกใช้ฟังก์ชันแบบ **Call-by-value**

2.การ**ส่งอาร์เรย์ทุกอีลีเมนต์ให้กับฟังก์ชัน** เป็นการเรียกใช้ฟังก์ชันแบบ **Call-by-reference**

การส่งผ่านอาร์เรย์ไปยังฟังก์ชัน

Call-by-value และ Call-by-reference เป็นวิธีการส่งค่าหรือตัวแปรเข้าสู่ฟังก์ชันในภาษาซี (และภาษาโปรแกรมอื่น ๆ) โดยมีความแตกต่างหลัก ๆ ในการจัดการกับข้อมูลดังนี้

1. Call-by-value เป็นวิธีการส่ง สำเนา ของค่าของตัวแปรเข้าสู่ฟังก์ชัน ฟังก์ชันจะทำงานกับสำเนาของตัวแปรนั้น ดังนั้น การเปลี่ยนแปลงค่าภายในฟังก์ชันจะไม่ส่งผลต่อค่าของตัวแปรต้นฉบับในฟังก์ชันที่เรียก ใช้ในการส่งค่าทั่วไป เช่น int, float, char เป็นต้น

ตัวอย่าง

```
#include <stdio.h>

void modifyValue(int x) {
    x = 10;  // เปลี่ยนแปลงสำเนาของตัวแปร x
}

int main() {
    int a = 5;
    modifyValue(a);  // ส่ง a เข้าฟังก์ชันโดยใช้ call-by-value
    printf("a = %d\n", a);  // ค่า a ยังเท่าเดิมคือ 5
}
```


การส่งผ่านอาร์เรย์ไปยังฟังก์ชัน

Call-by-value VS Call-by-reference เป็นวิธีการส่งค่าหรือตัวแปรเข้าสู่ฟังก์ชันในภาษาซี (และภาษาโปรแกรมอื่น ๆ) โดยมีความแตกต่างหลัก ๆ ในการจัดการกับข้อมูลดังนี้

1. Call-by-value เป็นวิธีการส่ง สำเนา ของค่าของตัวแปรเข้าสู่ฟังก์ชัน ฟังก์ชันจะทำงานกับสำเนาของตัวแปรนั้น ดังนั้น การเปลี่ยนแปลงค่าภายในฟังก์ชันจะไม่ส่งผลต่อค่าของตัวแปรต้นฉบับในฟังก์ชันที่เรียก ใช้ในการส่งค่าทั่วไป เช่น int, float, char เป็นต้น

ตัวอย่าง

```
#include <stdio.h>

void modifyValue(int x) {
    x = 10;  // เปลี่ยนแปลงสำเนาของตัวแปร x
}

int main() {
    int a = 5;
    modifyValue(a);  // ส่ง a เข้าฟังก์ชันโดยใช้ call-by-value
    printf("a = %d\n", a);  // ค่า a ยังเท่าเดิมคือ 5
}
```

การส่งผ่านอาร์เรย์ไปยังฟังก์ชัน

2. Call-by-reference **เป็นวิธีการส่งที่อยู่ของตัวแปร (address) หรือการส่งตัวแปรต้นฉบับ เข้าไปในฟังก์ชันโดยตรง ฟังก์ชันจะทำงานกับตัวแปรต้นฉบับผ่านการอ้างอิงถึงที่อยู่ ดังนั้น การเปลี่ยนแปลงค่าภายในฟังก์ชันจะส่งผลต่อค่าของตัวแปรต้นฉบับ** ใช้เมื่อส่ง pointer หรือการอ้างอิง เช่น การส่ง array หรือใช้ pointer เป็น argument **ตัวอย่างเช่น**

```
#include <stdio.h>

void modifyValue(int *x) {
   *x = 10;  // เปลี่ยนแปลงค่าของตัวแปรต้นฉบับ
}

int main() {
   int a = 5;
   modifyValue(&a);  // ส่งที่อยู่ของ a เข้าไปในฟังก์ชันโดยใช้ call-by-reference
   printf("a = %d\n", a);  // ค่า a จะเปลี่ยนเป็น 10
```


การส่งผ่านอาร์เรย์ไปยังฟังก์ชัน

การส่งผ่านอาร์เรย์ไปยังฟังก์ชันผ่านการเรียกใช้แบบ Call-by-value

- ใช้วิธีการส่งค่าของตัวแปร (value) ให้กับฟังก์ชัน โดยผ่านพารามิเตอร์
- ไม่สามารถแก้ไขค่าของอาร์กิวเมนต์(หรือพารามิเตอร์) ภายในฟังก์ชันได้ = การแก้ไขค่าต่างๆในฟังก์ชัน ไม่มีผลต่อตัวแปรที่ส่งค่ามา
- ใช้กับฟังก์ชันที่รับค่าเข้าเป็นตัวแปรธรรมดา (int, float, char,...) **ตัวอย่างเช่น**

```
void triple(int x)
        {  x=x*3;  printf("x = %d",x); }
int main() {
                int x=5, y[2]={10,11};
                triple(x);  triple(y[0]);
}
```

คำอธิบาย

triple(x) ส่งค่า 5 ให้กับฟังก์ชัน ในฟังก์ชัน x เริ่มต้นเป็น 5 และถูกทำให้กลายเป็น 15 หลังจบฟังก์ชัน ค่า x นอกฟังก์ชันจะไม่เปลี่ยนแปลง

triple(y[0]) ส่งค่า 10 ให้กับฟังก์ชัน ในฟังก์ชัน x เริ่มต้นเป็น 10 และถูกทำให้กลายเป็น 30 หลังจบฟังก์ชัน ค่า y[0] ยังเหมือนเดิม

การส่งผ่านอาร์เรย์ไปยังฟังก์ชัน

การส่งผ่านอาร์เรย์ไปยังฟังก์ชันผ่านการเรียกใช้แบบ Call-by-reference

- ใช้วิธีการส่งค่า แอดเดรส (Address)*** ของตัวแปรไปให้ฟังก์ชัน
- ใช้กับฟังก์ชันที่รับค่าเข้าเป็นอาร์เรย์
- สามารถแก้ไขค่าของอาร์กิวเมนต์ภายในฟังก์ชันได้ = การแก้ไขค่าตัวแปรอาร์เรย์ภายในฟังก์ชัน **มีผลการเปลี่ยนแปลงต่อตัวแปรที่ส่งค่ามา** เพราะจะจัดการค่าของหน่วยความจำในตำแหน่งเดียวกัน
- ***แอดเดรส (Address) คือ ค่าที่ใช้อ้างถึงตัวข้อมูลภายในหน่วยความจำ เหมือนกับหมายเลขบ้านเลขที่***

การส่งผ่านอาร์เรย์ไปยังฟังก์ชัน

ตัวอย่างการส่งผ่านอาร์เรย์ไปยังฟังก์ชันผ่านการเรียกใช้แบบ Call-by-reference

```
#include <stdio.h>
void modifyArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        arr[i] = arr[i] * 2;  // เปลี่ยนแปลงค่าของสมาชิกในอาร์เรย์
    }
}
int main() {
    int numbers[5] = {1, 2, 3, 4, 5};  // ประกาศและกำหนดค่าให้กับอาร์เรย์
    int size = 5;
    printf("Before modification:\n");
    for (int i = 0; i < size; i++) {
        printf("%d ", numbers[i]);  // แสดงค่าก่อนการเปลี่ยนแปลง
    }
    modifyArray(numbers, size);  // ส่งอาร์เรย์ไปยังฟังก์ชัน (Call-by-reference)
    printf("\nAfter modification:\n");
    for (int i = 0; i < size; i++) {
        printf("%d ", numbers[i]);  // แสดงค่าหลังจากการเปลี่ยนแปลง
    }
}
```

Output:

```
Before modification:
1 2 3 4 5
After modification:
2 4 6 8 10
```

การเขียนฟังก์ชันที่มีการรับค่าเข้าเป็นอาร์เรย์

ฟังก์ชันสามารถที่จะรับค่าเข้าเป็นอาร์เรย์ได้ ซึ่งรูปแบบของการเขียนต้นแบบของฟังก์ชันดังนี้

ชนิดข้อมูล ชื่อฟังก์ชัน(**ชนิดข้อมูล ชื่อตัวแปร[ขนาดอาร์เรย์]**);

ในกรณีฟังก์ชันมีการรับค่าเข้าเป็นอาร์เรย์ 1 มิติ อาจจะไม่ต้องกำหนดขนาดของอาร์เรย์ก็ได้

ตัวอย่างเช่น

```
int sum_arr(int num[10]);
void print_arr(int a[5]);
float average(int num[]);
```


ตัวอย่าง การส่งค่าหนึ่งอีลีเมนต์ในอาร์เรย์ให้กับฟังก์ชัน

```
#include <stdio.h>
void check_val(int x);
int main()
{
  int a[3] = {2,-1,5};
  check_val(a[0]);
  return 0;
}
void check_val(int x)
{
  if(x >= 0)
    printf("%d : Positive\n",x);
  else
    printf("%d : Negative\n",x) ;
}
```

Output:
2 : Positive

ตัวอย่าง การส่งค่าแต่ละอีลีเมนต์ในอาร์เรย์ให้กับฟังก์ชัน

```
#include <stdio.h>
void check_val(int x);
int main()
{
  int i,a[3] = {2,-1,5};
  for(i=0;i<3;i++) check_val(a[i]);
}
void check_val(int x)
{
 if(x >= 0)
              printf("%d : Positive\n",x);
  else
    printf("%d : Negative\n",x);
}
```

Output:
2 : Positive
-1 : Negative
5 : Positive

ตัวอย่าง การส่งค่าแต่ละอีลีเมนต์ในอาร์เรย์ให้กับฟังก์ชัน

```
#include <stdio.h>
void check_val(int x);
int main()
{
  int i,a[3] = {2,-1,5};
  for(i=0;i<3;i++) check_val(a[i]);
}
void check_val(int x)
{
 if(x >= 0)
              printf("%d : Positive\n",x);
  else
    printf("%d : Negative\n",x);
}
```

Output:
2 : Positive
-1 : Negative
5 : Positive

การส่งทั้งอาร์เรย์(ทุกอีลีเมนต์ของอาร์เรย์)ให้กับฟังก์ชัน

การส่งอาร์เรย์ในกรณีนี้ ใช้แค่ชื่อตัวแปรอาร์เรย์เท่านั้น เช่น หากใน main มีการประกาศอาร์เรย์ดังนี้

```
int num[10];
```

และฟังก์ชัน print_arr มีต้นแบบฟังก์ชันดังนี้

```
void print_arr(int a[10]);
```

การส่งอาร์เรย์ num ทุกอีลีเมนต์ไปให้ฟังก์ชัน print_arr สามารถเขียนได้ดังนี้

```
print_arr(num);
```


ตัวอย่าง การส่งทั้งอาร์เรย์(ทุกอีลีเมนต์ของอาร์เรย์)ให้กับฟังก์ชัน

```
#include <stdio.h>
void print_arr(int a[4]);
int main() {
  int num[4] = {5,2,-1,8};
  print_arr(num);
}

void print_arr(int a[4])
{
  int i;
  for(i =0;i<4;i++)
    printf("%d ", a[i]);
}
```

```
Output:
5  2  -1  8
```

ตัวอย่าง การส่งทั้งอาร์เรย์(ทุกอีลีเมนต์ของอาร์เรย์)ให้กับฟังก์ชัน

```
#include <stdio.h>
void find_max(int vals[5]); /* function prototype */
void main()
{
        int nums[5] = {2, 18, 1, 27, 16};
        find_max(nums);
}

void find_max(int vals[5]) /* find the maximum value */
{
        int i, max = vals[0];
        for ( i = 1; i < 5; ++i)
                if (max < vals[i])
                max = vals[i];
        printf("The maximum value is %d\n", max);
}
```

Output:
The maximum value is 27

จงเขียนโปรแกรมที่รับจำนวนเต็มจากผู้ใช้ 10 ตัว จากนั้นให้นับว่าในจำนวนเต็มทั้ง 10 ตัวนั้นมีตัวประกอบของ 40 อยู่กี่ตัว

กำหนดให้ส่วนที่รับค่าจำนวนเต็มจากผู้ใช้และส่วนที่แสดงผลลัพธ์อยู่ใน main และให้ส่วนที่นับตัวประกอบของ 40 อยู่ใน ฟังก์ชันชื่อ count_factor โดยให้ส่งอาร์เรย์ที่เก็บจำนวนเต็มทั้ง 10 จาก main มาให้กับฟังก์ชัน count_factor



```
#include <stdio.h>
int count_factor(int x[10]);
int main()
{               int num[10],num_factor;
                for(int i =0;i<10;i++)
                {               printf("Enter integer %d : ",i+1);
                                scanf("%d",&num[i]);
                }
        num_factor = count_factor(num);
        printf("Found factor of 40 : %d",num_factor);
}

int count_factor(int x[10])
{
  int count = 0;
  for(int i=0;i<10;i++)
  {
    if(40%x[i] == 0)
    count++;
  }
  return count;
}
```

```
Output:
Enter integer 1: 2
Enter integer 2: 15
Enter integer 3: 20
Enter integer 4: 10
Enter integer 5: 3
Enter integer 6: 1
Enter integer 7: 5
Enter integer 8: 4
Enter integer 9: 25
Enter integer 10: 30
Found factor of 40 : 6
```


จงเขียนโปรแกรมสำหรับรับค่าคะแนนของนักศึกษาจำนวน 10 คน และให้พิมพ์ค่าเฉลี่ยของคะแนนทั้งหมด

โดยกำหนดให้ส่วนรับคะแนนจากผู้ใช้และส่วนที่แสดงค่าเฉลี่ยอยู่ใน main สำหรับส่วนที่คำนวณค่าเฉลี่ยให้อยู่ฟังก์ชันชื่อ average โดยให้ส่งอาร์เรย์ที่เก็บคะแนนทั้งของ 10 คนจาก main มาให้ฟังก์ชัน average (คะแนนสามารถเป็นทศนิยมได้)



```
#include <stdio.h>
float average(float num[10]);
int main()
{ int i;  float score[10],avg_score;
  for(i=0;i<10;i++)
  {  printf("Enter score %d : ",i+1);
     scanf("%f",&score[i]);
  }
  avg_score = average(score);
  printf("Average score is %.2f\n",avg_score);
}
float average(float num[10])
{   float total,avg;
    int i;
    total = 0.0;
    for(i=0;i<10;i++)
                total = total + num[i];
    avg = total/10.0;
    return avg;
}
```

```
Output:
Enter score 1: 10
Enter score 2: 20
Enter score 3: 30
Enter score 4: 40
Enter score 5: 50
Enter score 6: 60
Enter score 7: 70
Enter score 8: 80
Enter score 9: 90
Enter score 10: 100
Average score is : 55.00
```


การแก้ไขค่าของอาร์เรย์ภายในฟังก์ชัน

ฟังก์ชันที่มีการรับพารามิเตอร์เป็นอาร์เรย์ หากมีการแก้ไขค่าภายในอาร์เรย์ดังกล่าว จะส่งทำให้อาร์เรย์ที่ถูกส่งมาเป็น อาร์กิวเมนต์ของฟังก์ชันมีการเปลี่ยนแปลงด้วย

ตัวอย่างเช่น

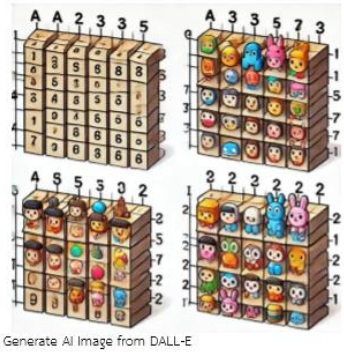
```
#include <stdio.h>
void edit_arr(int a[4]);
int main() {
  int i,num[4] = {2,5};
  edit_arr(num);
  for(i=0;i<4;i++)
    printf("%d ",num[i]);
  return 0;
}
void edit_arr(int a[4]) {
  int i;
  for(i=0;i<4;i++)
    a[i] = a[i]*a[i];
}
```

```
Output:
4   25   0   0
```

ตัวแปรอาร์เรย์แบบหลายมิติ

ตัวแปรอาร์เรย์หลายมิติ

ตัวแปรอาร์เรย์แบบหลายมิติในภาษาซี (Multidimensional Array) เป็นตัวแปรอาร์เรย์ที่มีมากกว่า 1 มิติ โดยทั่วไปแล้วจะใช้สำหรับการจัดเก็บข้อมูลใน**รูปแบบตารางหรือเมทริกซ์** (เช่น 2 มิติ) หรือ**ในรูปแบบที่ซับซ้อนมากขึ้น**เช่น 3 มิติหรือมากกว่า



อ้างอิงรูปภาพ: Generate AI Image from DALL-E

ตัวแปรอาร์เรย์ 2 มิติ

การประกาศและใช้งานอาร์เรย์สองมิติ (2D Array)

อาร์เรย์สองมิติสามารถถูกมองว่าเป็นตารางที่มีแถวและคอลัมน์ (เช่น ตารางข้อมูล หรือ เมทริกซ์) การประกาศจะระบุจำนวนแถวและคอลัมน์

ชนิดตัวแปร ชื่อตัวแปร[จำนวนสมาชิก(ROW)][จำนวนสมาชิกในหลัก(COL)];

ตัวอย่างเช่น

```
int matrix[3][4];  // ประกาศอาร์เรย์ 2 มิติ ขนาด 3 แถว และ 4 คอลัมน์
```


ตัวแปรอาร์เรย์ 3 มิติ

การประกาศและใช้งานอาร์เรย์สามมิติ (3D Array)

อาร์เรย์สามมิติสามารถถูกมองว่าเป็นชุดของ **"ตารางหลายๆ ตาราง"** หรือ **"ลูกบาศก์ข้อมูล"** ซึ่งจะมีการจัดเก็บข้อมูลที่ซับซ้อนยิ่งขึ้น เช่น ในการจัดเก็บข้อมูลสามมิติ (เช่น **แถว, คอลัมน์, และชั้น**)

ชนิดตัวแปร ชื่อตัวแปร[จำนวนสมาชิกในหลัก(LAYER)][จำนวนสมาชิก(ROW)][จำนวนสมาชิกในหลัก(COL)];

ตัวอย่างเช่น

```
int cube[2][3][4]; // ประกาศอาร์เรย์ 3 มิติ ขนาด 2 ชั้น, 3 แถว และ 4 คอลัมน์
```

ตัวอย่างตัวแปรอาร์เรย์

อาร์เรย์ที่มีสมาชิกอยู่ 3 ตัว และสมาชิกแต่ละตัวก็เป็นอาร์เรย์เก็บข้อมูลชนิด int มีจำนวนสมาชิก 2 ตัว จะกำหนดได้ดังนี้

```
int x[3][2];
```

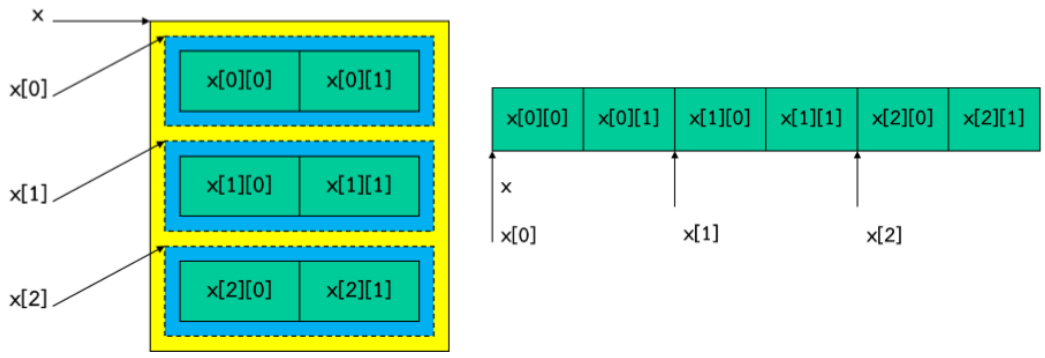
ขนาดของอาร์เรย์ 3 x 2

จำนวนไบต์ ที่ใช้ในการเก็บอาร์เรย์

sizeof (a) คำนวณจาก 3 x 2 x sizeof (int) = 3x2x4 = 24 ไบต์

ตัวอย่างตัวแปรอาร์เรย์

ตัวอย่างโครงสร้างของ int x[3][2];



ตัวอย่าง การหาขนาดตัวแปรอาร์เรย์ 2 มิติ

```
#include <stdio.h>
int main()
{
  int x[2][5];
  char names[3][4];
  float nums[2][2];
  printf("Size of x = %d\n",sizeof(x));
  printf("Size of names = %d\n",sizeof(names));
  printf("Size of nums = %d\n",sizeof(nums));
}
```

Output:
Size of x = 40
Size of names = 12
Size of nums = 16

การเข้าถึงอาร์เรย์ 2 มิติ

- int x[3][2]; จะประกอบไปด้วย 3 แถว 2 หลัก หรือ มี x[0] x[1] และ x[2] โดยแต่ละตัวจะประกอบไปด้วยสมาชิก 2 ตัว เช่น x[0] มีสมาชิก ได้แก่ x[0][0] และ x[0][1]
- การเข้าถึงอาร์เรย์ 2 มิติ จะต้องระบุ index ทั้งสอง ได้แก่ แถวและหลักตามลำดับ จึงจะเข้าถึงค่าที่อยู่ใน element ต่างๆได้

x[3][2] จะมีสมาชิกดังนี้

x[0][0]	x[0][1]
x[1][0]	x[1][1]
x[2][0]	x[2][1]

สรุป

เมื่ออ้างชื่ออาร์เรย์พร้อมสมาชิกหนึ่งอันดับ เช่น x[0] จะหมายถึงตำแหน่งเริ่มต้นของอาร์เรย์ย่อยภายในอาร์เรย์ 1 มิติ

เมื่ออ้างชื่ออาร์เรย์พร้อมค่าสองอันดับ เช่น x[0][0] จะหมายถึง **ข้อมูล(ค่าที่เก็บ)**ภายในอาร์เรย์

ตัวอย่างการประกาศตัวแปรอาร์เรย์ 2 มิติ

```
int val[3][4];                  // 3 แถว 4 หลัก
double prices[10][5];           // 10 แถว 5 หลัก
char code[6][4];                // 6 แถว 4 หลัก
```


ตัวอย่างการเข้าถึงตัวแปรอาร์เรย์ 2 มิติ

```
char code[6][4];        // 6 แถว 4 หลัก
```

code[3] →

***หากเรียก code[3] ในภาษาซีจะ return Address ของ code[3][0]

	column 0	column 1	column 2	column 3
row 0	code[0][0]	code[0][1]	code[0][2]	code[0][3]
row 1	code[1][0]	code[1][1]	code[1][2]	code[1][3]
row 2	code[2][0]	code[2][1]	code[2][2]	code[2][3]
row 3	code[3][0]	code[3][1]	code[3][2]	code[3][3]
row 4	code[4][0]	code[4][1]	code[4][2]	code[4][3]
row 5	code[5][0]	code[5][1]	code[5][2]	code[5][3]

การใช้ for วนรอบเข้าถึงตัวแปรอาร์เรย์ 2 มิติ

ในภาษา C การเข้าถึงอาร์เรย์ 2 มิติสามารถทำได้โดยใช้ลูป for ซึ่งเราจะใช้ลูปซ้อน (nested loop) เพื่อเข้าถึงแต่ละแถวและแต่ละคอลัมน์ในอาร์เรย์ ตัวอย่างเช่น

```
// ประกาศอาร์เรย์ 2 มิติขนาด 3x3
int array[3][2] = {
    {1, 2},
    {4, 5},
    {7, 8}
};
// ใช้ for loop ซ้อนกันเพื่อเข้าถึงแต่ละตำแหน่งของอาร์เรย์
for (int i = 0; i < 3; i++) { // ลูปภายนอกใช้วนแถว (row)
    for (int j = 0; j < 2; j++) { // ลูปภายในใช้วนคอลัมน์ (column)
        printf("array[%d][%d] = %d\n", i, j, array[i][j]);
    }
}
```

	col 0	col 1
row 0	1	2
row 1	4	5
row 2	7	8

ตัวอย่างการใช้ for วนรอบเข้าถึงตัวแปรอาร์เรย์ 2 มิติ

```
#include<stdio.h>
int main(void)
{
  int i, j, val[3][4] = {8,16,9,52,3,15,27,6,14,25,2,10};
  printf("\n Display all elements of array");
  printf("\n%2d %2d %2d %2d",  val[0][0],val[0][1],val[0][2],val[0][3]);
  printf("\n%2d %2d %2d %2d",  val[1][0],val[1][1],val[1][2],val[1][3]);
  printf("\n%2d %2d %2d %2d\n",val[2][0],val[2][1],val[2][2],val[2][3]);
  for ( i = 0; i < 3; ++i)
  {
          printf("\n");          /* start a new line for each row */
          for ( j = 0; j < 4; ++j)   printf("%2d  ", val[i][j]);
  }
}
```

```
Output:
Display all elements of array
 8 16  9 52
 3 15 27  6
14 25  2 10

 8 16  9 52
 3 15 27  6
14 25  2 10
```

ตัวอย่างการใช้ for วนรอบเข้าถึงตัวแปรอาร์เรย์ 2 มิติ

```
#include <stdio.h>
int main()
{  int i, j, val[3][4] = {8,16,9,52,3,15,27,6,14,25,2,10};
  /* multiply each element by 10 and display it */
  printf("\n\nDisplay of multiplied elements\n");
  for(i=0 ; i < 3 ; ++i )
  {
        printf("\n");  /* start a new line */
        for(j = 0; j < 4; ++j)
        {
              val[i][j] = val[i][j] * 10;
              printf("%3d ", val[i][j]);
        } /* End for j */
  } /* End for i */
}
```

Output:
Display all multiplied elements
80 160 90 520
30 150 270 60
140 250 20 100

ตัวอย่างการใช้ for วนรอบรับค่าตัวแปรอาร์เรย์ 2 มิติ

```
#include <stdio.h>
int main() {
  int i,j,a[2][3];
  for(i=0;i<2;i++) {
                for(j=0;j<3;j++) {
                printf("Enter a[%d][%d] : ",i,j);
                scanf("%d", &a[i][j]);
   }
  }
  for(i=0;i<2;i++) {
                for(j=0;j<3;j++)
                printf("%d",a[i][j]);
                printf("\n");
  }
}
```

Input:
Enter a[0][0]: 10
Enter a[0][1]: 20
Enter a[0][2]: 30
Enter a[1][0]: 40
Enter a[1][1]: 50
Enter a[1][2]: 60
Output:
10 20 30
40 50 60

การกำหนดค่าเริ่มต้น (Array Initialization) ของอาร์เรย์ 2 มิติ

สามารถกำหนดค่าเริ่มต้นของอาร์เรย์ 2 มิติทั้งหมดได้โดยใช้เครื่องหมายปีกกา {} โดยระบุค่าทั้งหมดในรูปแบบที่แบ่งแยกตามแถวและคอลัมน์ ดังตัวอย่าง

// ประกาศและกำหนดค่าเริ่มต้นให้กับอาร์เรย์ 2 มิติขนาด 3x3

```
int array[3][3] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

หรือ

```
int array[3][3] = {
    1, 2, 3,
    4, 5, 6,
    7, 8, 9
};
```

ก็ได้ โดยจะจับชุดละ 3 ไปลง collumn ในแต่ละแถวอัตโนมัติ

การกำหนดค่าเริ่มต้น (Array Initialization) ของอาร์เรย์ 2 มิติ

หากไม่มีการกำหนดจำนวนแถว คอมไพเลอร์จะกำหนดจำนวนแถวโดยนับจากที่กำหนดในค่าเริ่มต้น แต่จะต้องมีการกำหนดจำนวนหลักเสมอ เช่น

```
int a[][2] = {{5,8},{9},{-1}};
```

	col 0	col 1
row 0	5	8
row 1	9	0
row 2	-1	0

การกำหนดค่าเริ่มต้น (Array Initialization) ของอาร์เรย์ 2 มิติ

ในการกำหนดค่าเริ่มต้นของอาร์เรย์ 2 มิติ สามารถละเครื่องหมายปีกกาที่ใช้แบ่งแถวได้ โดยใช้จำนวนหลักในการจัดว่าอีลีเมนต์ใดอยู่แถวใด

เช่น float x[2][3] = { 2.5, 1.25, 8.6,4.1, 6.9, 7.2 };

	col 0	col 1	col 2
row 0	2.5	1.25	8.6
row 1	4.1	6.9	7.2

โจทย์ฝึก อาร์เรย์สองมิติ

จงเขียนโปรแกรมสำหรับรับ**เมตริกซ์ของเลขจำนวนเต็มขนาด 2x2 จำนวน 2 เมตริกซ์** จากนั้นแสดงผลบวกของเมตริกซ์ทั้งสอง โดยตัวอย่างผลการรันเป็นดังนี้ (ข้อความสีแดงคือค่าที่รับจากผู้ใช้)

```
Enter matrix A(2x2):
2 -1
15 9
Enter matrix B(2x2):
10 8
3 -7
Matrix A+B:
12 7
18 2
```

เฉลยโจทย์ฝึก อาร์เรย์สองมิติ

```
#Include <stdlo.h>
int main()
{
  int A[2][2],B[2][2],i,j;
  printf("Enter matrix A:\n");
  for(i=0;i<2;i++) {
                        // for(j=0;j<2;j++) scanf("%d",&A[i][j]); รับทีละ element
                        scanf("%d %d", &A[i][0], &A[i][1]);
  }
  printf("Enter matrix B:\n" );
  for(i=0;i<2;i++) {
                        // for(j=0;j<2;j++) scanf("%d",&B[i][j]); รับทีละ element
                        scanf("%d %d", &B[i][0], &B[i][1]);
  }
   printf("Matrix A+B:\n");
   for(i=0;i<2;i++) {
                          for(j=0;j<2;j++)
    printf("%d  ",A[i][j] + B[i][j]);
                        printf("\n");
  }
}
```


ฟังก์ชันที่รับค่าเข้าเป็นอาร์เรย์ 2 มิติ

ต้นแบบของฟังก์ชัน:

```
void display(int val[3][4]);
```

ฟังก์ชันมีชื่อว่า display ไม่มีการส่งค่ากลับ แต่รับค่าเข้าเป็นอาร์เรย์ของ int ขนาด 3 แถว 4 หลัก

สามารถละจำนวนแถวก็ได้ แต่ต้องระบุจำนวนหลักเสมอ เช่น

```
void display(int val[ ][4]);
```

ตัวอย่าง ฟังก์ชันที่รับค่าเข้าเป็นอาร์เรย์ 2 มิติ

```
#include <stdio.h>
void display(int nums[3][4]);   /* function prototype */
int main()
{
   int i, j, val[3][4] = {8,16,9,52,3,15,27,6,14,25,2,10};
   display(val);
}
void display (int nums[3][4])
{    int row_num, col_num;
     for(row_num = 0 ; row_num < 3 ; row_num++)
     {
        for(col_num = 0; col_num < 4; ++col_num)  printf("%4d", nums[row_num][col_num]);
        printf("\n");
  } /* End for col_num */
}
```

```
Output:
 8   16   9   52
 3   15  27    6
14   25   2   10
```


ตัวแปรอาร์เรย์ 3 มิติ

ตัวแปรอาร์เรย์ 3 มิติ (3D Array) มักถูกนำไปใช้ในงานที่ต้องจัดการกับข้อมูลที่มีความซับซ้อนมากกว่า 2 มิติ โดยสามารถจัดเก็บข้อมูลในลักษณะที่มี 3 มิติ (เช่น แถว, คอลัมน์, และชั้น) งานที่ใช้อาร์เรย์ 3 มิติมักเกี่ยวข้องกับการจัดการข้อมูลหลายมิติ

ตัวอย่างเช่น

1. การประมวลผลภาพ (Image Processing)

อาร์เรย์ 3 มิติถูกใช้ในการจัดเก็บข้อมูลของภาพสี โดยภาพแต่ละภาพจะมีพิกเซลที่แบ่งออกเป็นแถวและคอลัมน์ และในแต่ละพิกเซลจะมีข้อมูลสีที่ประกอบด้วยค่า RGB (Red, Green, Blue) ซึ่งทำให้เป็นอาร์เรย์ 3 มิติ โดยมิติแรกคือตำแหน่งแถว มิติที่สองคือตำแหน่งคอลัมน์ และมิติที่สามคือตัวแทนของสี

เช่น การเก็บภาพขนาด 100x100 พิกเซล ที่แต่ละพิกเซลมีค่า RGB สามค่า จะใช้ int image[100][100][3]

2. การสร้างแบบจำลองทางกายภาพ (Physical Simulations)

การจำลองแบบจำลอง 3 มิติ เช่น การเคลื่อนที่ของวัตถุในพื้นที่สามมิติ การจำลองอุณหภูมิหรือความดันในอวกาศสามมิติ หรือการคำนวณการไหลของของไหลในพื้นที่สามมิติ อาร์เรย์ 3 มิติจะถูกใช้เพื่อเก็บค่าของตัวแปรต่าง ๆ เช่น ความเร็ว, ความดัน, และอุณหภูมิ ที่ตำแหน่งต่าง ๆ ในพื้นที่

เช่น การจำลองกล่องขนาด 10x10x10 หน่วย สามารถใช้ double simulation[10][10][10] เพื่อเก็บค่า

ตัวแปรอาร์เรย์ 3 มิติ

3. เกม 3 มิติและกราฟิกคอมพิวเตอร์ (3D Games and Computer Graphics)

อาร์เรย์ 3 มิติมักถูกใช้ในการเก็บข้อมูลที่เกี่ยวข้องกับตำแหน่งของวัตถุในเกม 3 มิติ เช่น ตำแหน่งของตัวละคร, สภาพแวดล้อม, หรือการเรนเดอร์ภาพสามมิติ การใช้ 3D array สามารถช่วยในการจัดการพื้นที่ในเกมหรือกราฟิกได้อย่างมีประสิทธิภาพ

เช่น การเก็บข้อมูลตำแหน่งวัตถุในเกมสามารถใช้ int game_world[100][100][100] ที่แทนแต่ละพิกเซลหรือโวลุมในโลกของเกม

4. การจัดการข้อมูลทางวิทยาศาสตร์ (Scientific Data Management)

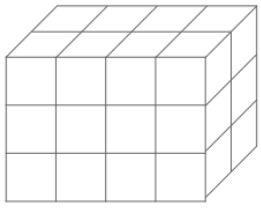
อาร์เรย์ 3 มิติมักถูกใช้ในการจัดเก็บข้อมูลที่เกิดจากการทดลองทางวิทยาศาสตร์ที่ซับซ้อน เช่น การวัดค่าต่าง ๆ ที่เกิดขึ้นในจุดต่าง ๆ ในพื้นที่สามมิติ หรือการเก็บข้อมูลจากการสำรวจหรือทดลองในอวกาศ (เช่น ข้อมูลดาวเทียม)

เช่น การจัดเก็บข้อมูลเกี่ยวกับอุณหภูมิและความดันในพื้นที่ 3 มิติของมหาสมุทร สามารถใช้ float ocean_data[depth][latitude][longitude]

ตัวอย่าง ตัวแปรอาร์เรย์ 3 มิติ

ตัวอย่าง อาร์เรย์ 3 มิติ ที่ใช้เก็บค่าปริมาณแต่ละบล๊อคในลูกบาศก์ (Cube)

```
int volume[3][4][2];
int i,j,k;
for(i=0;i<3;i++)
        for(j=0;j<4;j++)
        for(k=0;k<2;k++)
        val[i][j][k] = 0;
```



อีลีเมนต์แรก -> val[0][0][0] อีลีเมนต์สุดท้าย -> val[2][3][1]

สายอักขระ (String)

สายอักขระ (String)

ในภาษาซี (C) ไม่มีชนิดข้อมูลเฉพาะสำหรับสายอักขระ (String) อ่านว่า “สตริง” เหมือนในบางภาษาระดับสูงอื่น ๆ เช่น Python หรือ Java **แต่ string ในภาษาซีจะถูกจัดการผ่านอาร์เรย์ของตัวอักษร (character array) โดย string ในภาษาซีต้องสิ้นสุดด้วยอักขระพิเศษ '\0' (null terminator) เพื่อบ่งบอกว่าจบสตริง**

การประกาศและการใช้งานสตริงในภาษาซี

1. **การประกาศสตริงด้วยอาร์เรย์ของตัวอักษร** สามารถประกาศอาร์เรย์ของตัวอักษรเพื่อเก็บสตริง

ตัวอย่าง `char str[6] = "Hello";`

ในตัวอย่างนี้ อาร์เรย์ str จะมีขนาด 6 ตำแหน่ง เพราะสตริง "Hello" ประกอบด้วยตัวอักษร 5 ตัว (H, e, l, l, o) และต้องใช้ตำแหน่งพิเศษสำหรับ '\0' เพื่อสิ้นสุดสตริง

ค่าที่ถูกจัดเก็บในอาร์เรย์ str จะเป็น: ['H', 'e', 'l', 'l', 'o', '\0']

สายอักขระ (String)

2. **การประกาศสตริงโดยไม่ระบุขนาดอาร์เรย์** เราสามารถให้คอมไพเลอร์คำนวณขนาดอาร์เรย์โดยอัตโนมัติจากสตริงที่กำหนด

ตัวอย่าง char str[] = "Hello";

คอมไพเลอร์จะสร้างอาร์เรย์ที่มีขนาดเพียงพอ (6 ตัวอักษร) เพื่อเก็บ "Hello" รวมถึงอักขระ '\0'

3. **การประกาศสตริงเปล่า** โดยเราสามารถประกาศสตริงเปล่า (ไม่มีข้อมูล) แล้วกำหนดค่าให้ในภายหลัง

ตัวอย่าง char name[100];

4. **การเข้าถึงและปฏิบัติเปลี่ยนค่าแต่ละตำแหน่งในสตริง** โดยสามารถเข้าถึงหรือเปลี่ยนแปลงสมาชิกของอาร์เรย์ (ตัวอักษร) ได้โดยใช้ดัชนี (Index)

ตัวอย่าง

```
name[0] = 'J';
name[1] = 'o';
name[2] = 'h';
name[3] = 'n';
name[4] = '\0';  // สิ้นสุดสตริงที่นี่
```


สายอักขระ (String)

ตัวอย่างการประกาศ string

```
char str1[]  ={'A', 'B', 'C', 'D', '\0'};
char str2[10]={'A', 'B', 'C', 'D', '\0'};
char str3[]  = "ABCD";
```

ตัวอย่างการประกาศที่ผิด

```
char str4[2] ={'A', 'B', 'C', 'D', '\0'};
```

สายอักขระ (String)

ตัวอย่างการประกาศ string

```
char msg[10] = "Error!";
```

ถูกเซ็ตเป็นศูนย์อัตโนมัติ เพราะ เป็น element ส่วนที่เหลือที่ยังไม่ได้การกำหนดค่าเริ่มต้นให้

msg[0]	msg[1]	msg[2]	msg[3]	msg[4]	msg[5]	msg[6]	msg[7]	msg[8]	msg[9]
'E'	'r'	'r'	'o'	'r'	'!'	'\0'	0	0	0

สิ้นสุดด้วยอักขระพิเศษ '\0' (null terminator) เพื่อบ่งบอกว่าจบสตริง

สายอักขระ (String)

การใช้ฟังก์ชันสตริงมาตรฐาน

ภาษาซีมีฟังก์ชันในไลบรารี <string.h> ที่ช่วยในการจัดการกับสตริง เช่น การคำนวณความยาวของสตริง, การคัดลอกสตริง, และการต่อสตริง

1. **strlen()** – ใช้ในการคำนวณความยาวของสตริง (ไม่รวมอักขระ '\0')

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[] = "Hello";
    int length = strlen(str);  // คำนวณความยาวของสตริง
    printf("Length of the string: %d\n", length);  // แสดงความยาวสตริง
}
```

Output:
Length of the string: 5

สายอักขระ (String)

การใช้ฟังก์ชันสตริงมาตรฐาน

2. strcpy() – ใช้ในการคัดลอกสตริงจากแหล่งที่มาไปยังปลายทาง เนื่องจากหลัง initial value แล้วไม่สามารถกำหนดค่าใหม่ด้วย = ตรงๆได้ จึงจำเป็นต้องใช้strcpy()

```
#include <stdio.h>
#include <string.h>
int main() {
    char source[] = "Hello";
    char destination[10];
    strcpy(destination, source);  // คัดลอกสตริงจาก source ไปยัง destination
    printf("Copied string: %s\n", destination);
}
```

```
char message[20];
message = "Hello World!";// Error

strcpy(message,"Hello"); // OK
```

Output:
Copied string: Hello

สายอักขระ (String)

การใช้ฟังก์ชันสตริงมาตรฐาน

3. strcat() – ใช้ในการต่อสตริง (concatenate) สองสตริงเข้าด้วยกัน

```
#include <stdio.h>
#include <string.h>
int main() {
    char first[20] = "Hello ";
    char second[] = "World";
    strcat(first, second);  // ต่อสตริง first และ second เข้าด้วยกัน
    printf("Concatenated string: %s\n", first);
}
```

Output:
Concatenated string: Hello World

สายอักขระ (String)

การใช้ฟังก์ชันสตริงมาตรฐาน

4. strcmp() – ใช้ในการเปรียบเทียบสตริงสองสตริง

```
#include <stdio.h>
#include <string.h>

int main() {
    char str1[] = "Apple";
    char str2[] = "Banana";
    int result = strcmp(str1, str2);  // เปรียบเทียบสตริง str1 และ str2

    if (result == 0) {
        printf("Strings are equal.\n");
    } else if (result > 0) {
        printf("str1 is greater than str2.\n");
    } else {
        printf("str1 is less than str2.\n");
    }
}
```

Output:
str1 is less than str2.

สายอักขระ (String)

การรับสตริงจากผู้ใช้

ในภาษาซี คุณสามารถใช้ scanf() หรือ gets() ในการรับสตริงจากผู้ใช้ แต่ควรระมัดระวังเรื่องขนาดของอาร์เรย์ที่ใช้เก็บสตริงเพื่อหลีกเลี่ยงการเขียนเกินขอบเขต (buffer overflow)

ตัวอย่างการใช้ scanf() เพื่อรับสตริง

```
#include <stdio.h>
int main() {
    char name[100];  // ประกาศอาร์เรย์ขนาด 100 สำหรับเก็บสตริง
    printf("Enter your name: ");
    scanf("%s", name);  // รับสตริงจากผู้ใช้ (scanf จะรับสตริงจนกว่าจะพบช่องว่าง หรือ กด Enter)
    // หากต้องการพิมพ์ช่องว่างระหว่าง string ได้ ต้องระบุ Terminate Char -> scanf("%[^\n]s", name);
    printf("Hello, %s!\n", name);  // แสดงข้อความทักทาย
}
```

ข้อควรระวัง

ขนาดของอาร์เรย์: ต้องแน่ใจว่าขนาดของอาร์เรย์มีขนาดเพียงพอที่จะเก็บสตริงรวมถึงอักขระ '\0' ด้วย

Buffer Overflow: ระวังการเขียนข้อมูลเกินขนาดของอาร์เรย์ เพราะอาจทำให้เกิดข้อผิดพลาดหรือช่องโหว่ด้านความปลอดภัย

รับสตริงที่มีช่องว่าง: ถ้าต้องการรับสตริงที่มีช่องว่าง (เช่น ชื่อเต็ม) ควรใช้ฟังก์ชันที่รองรับการรับข้อมูลที่มีช่องว่าง เช่น gets() (แต่ควรระวังเรื่องความปลอดภัย) หรือ fgets() (ปลอดภัยกว่า gets())

อาร์เรย์ของสายอักขระ (String)

เนื่องจาก String จัดเป็นอาร์เรย์ 1 มิติ

```
char name[6] = "Nesta";
```

ดังนั้น อาร์เรย์ของ String จึงต้องมีรูปแบบเป็นอาร์เรย์ 2 มิติ เช่น

```
char names[2][20] ={ "JohnWick","Thor" };
```

names[0]	'J'	'o'	'h'	'n'	'W'	'i'	'c'	'k'	'\0'	0
names[1]	'T'	'h'	'o'	'r'	'\0'	0	0	0	0	0

จงเขียนโปรแกรมสำหรับรับข้อความจากผู้ใช้ 4 ข้อความ โดยแต่ละข้อความมีความยาวไม่ถึง 20 ตัวอักษร จากนั้นพิมพ์ค่าความยาวของแต่ละข้อความออกทางหน้าจอ

(ฟังก์ชันที่ใช้ในการหาความยาวของข้อความคือ strlen)

ตัวอย่างผลการรันเป็นดังนี้

```
Enter text 1: Maths
Enter text 2: Physics
Enter text 3: Chemistry
Enter text 4: Biology
Maths: 5
Physics: 7
Chemistry: 9
Biology: 7
```

จงเขียนโปรแกรมสำหรับรับข้อความจากผู้ใช้ 4 ข้อความ โดยแต่ละข้อความมีความยาวไม่ถึง 20 ตัวอักษร จากนั้นพิมพ์ค่าความยาวของแต่ละข้อความออกทางหน้าจอ

(ฟังก์ชันที่ใช้ในการหาความยาวของข้อความคือ strlen)

```
#include <stdio.h>
#include <string.h>
int main()
{
  char texts[4][20];
  int i;
  for(i=0;i<4;i++) {
    printf("Enter text %d : ",i+1);
    scanf("%s",&texts[i]);
  }
  for(i=0;i<4;i++)
    printf("%s : %d\n",texts[i],strlen(texts[i]));
}
```


จงเขียนโปรแกรมสำหรับรับเมตริกซ์จากผู้ใช้ 1 เมตริกซ์ (ให้ชื่อว่าเมตริกซ์ A) โดยผู้ใช้สามารถกำหนดจำนวนหลักและแถวของเมตริกซ์ได้ (แต่สมมุติว่าผู้ใช้ใส่ค่าจำนวนแถวและจำนวนหลักไม่เกิน 10) จากนั้นให้แสดงผลลัพธ์ของเมตริกซ์ 2A

ตัวอย่างผลการรันโปรแกรมเป็นดังนี้ (ข้อความสีแดงคือค่าที่รับจากผู้ใช้)

Enter number of rows: 2

Enter number of columns: 3

Enter matrix A:

4 3 5

2 9 7

Matrix 2A:

8 9 10

4 18 14

จงเขียนโปรแกรมสำหรับรับเมตริกซ์จากผู้ใช้ 1 เมตริกซ์ (ให้ชื่อว่าเมตริกซ์ A) โดยผู้ใช้สามารถกำหนดจำนวนหลักและแถวของเมตริกซ์ได้ (แต่สมมุติว่าผู้ใช้ใส่ค่าจำนวนแถวและจำนวนหลักไม่เกิน 10) จากนั้นให้แสดงผลลัพธ์ของเมตริกซ์ 2A

การคูณเมตริกซ์ Matrix Multiplication

Matrix ที่คูณกันได้ ตัวตั้ง (A) ต้องมีจำนวนหลัก เท่ากับจำนวนแถว ของตัวคูณ (B) และ Matrix ผลคูณ (C) จะมีจำนวนแถวเท่ากับตัวตั้งและจำนวนหลักเท่ากับตัวคูณ

$$C = A \times B$$

$$C_{mxp} = A_{mxn} \times B_{nxp}$$

จงเขียนโปรแกรมสำหรับรับเมตริกซ์จากผู้ใช้ 1 เมตริกซ์ (ให้ชื่อว่าเมตริกซ์ A) โดยผู้ใช้สามารถกำหนดจำนวนหลักและแถวของเมตริกซ์ได้ (แต่สมมุติว่าผู้ใช้ใส่ค่าจำนวนแถวและจำนวนหลักไม่เกิน 10) จากนั้นให้แสดงผลลัพธ์ของเมตริกซ์ 2A

การคูณเมตริกซ์ Matrix Multiplication

$$A = \begin{bmatrix} a11 & a12 \\ a21 & a22 \\ a31 & a32 \end{bmatrix} \qquad B = \begin{bmatrix} b11 & b12 & b13 & b14 \\ b21 & b22 & b23 & b24 \end{bmatrix}$$

$$C = \begin{bmatrix} a11*b11+a12*b21 & a11*b12+a12*b22 & a11*b13+a12*b23 & a11*b14+a12*b24 \\ a21*b11+a22*b21 & a21*b12+a22*b22 & a21*b13+a22*b23 & a21*b14+a22*b24 \\ a31*b11+a32*b21 & a31*b12+a32*b22 & a31*b13+a32*b23 & a31*b14+a32*b24 \end{bmatrix}$$

$$c_{ij} = a_{i1}*b_{1j} + a_{i2}*b_{2j} + \ldots + a_{in}*b_{nj}$$

```
void multiplyMatrix(int A[MAX][MAX],int B[MAX][MAX], int C[MAX][MAX], int m, int n, int p )
{
   int i,j,k,sum;
   for (i=0; i<m; i++)  //row of C
   {
       for(j=0; j<p; j++) //column of C
       {
          for(k=0,sum=0; k<n; k++)
                sum+=A[i][k]*B[k][j];
                C[i][j]=sum;
       }
   }
}
```


ความผิดพลาดและข้อควรระวังในการใช้งานอาร์เรย์

ลืมประกาศตัวแปร

ใช้งานตัวแปรอาร์เรย์ นอกเหนือจากขอบเขตที่ประกาศไว้ ตัวอย่างเช่น

```
int a_error[3] , i;
for ( i = 0 ; i <=3 ; i++ ) // มีช่องเพียง [0] [1] [2]
        a_error[i] = i;
```

ความผิดพลาดและข้อควรระวังในการใช้งานอาร์เรย์

ลืมให้ค่าเริ่มต้นกับตัวแปรอาร์เรย์ แล้วนำตัวแปรนั้นไปใช้งาน

```
int a1 = 5, a2 = 8;
int total[2];
total[1] = total[0] + a1 + a2;
```


ความผิดพลาดและข้อควรระวังในการใช้งานอาร์เรย์

ลืมให้ค่าเริ่มต้นกับตัวแปรอาร์เรย์ แล้วนำตัวแปรนั้นไปใช้งาน

```
int a1 = 5, a2 = 8;
int total[2];
total[1] = total[0] + a1 + a2;
```

การให้ค่าเริ่มต้นอย่างง่าย int array[10][5][3]={ }; ทุก element เป็นศูนย์

กำหนดค่าให้กับตัวแปรอาร์เรย์โดยตรง

```
int x[10];
    x = 11;
```

ความผิดพลาดและข้อควรระวังในการใช้งานอาร์เรย์

- เมื่อประกาศตัวแปรอาร์เรย์และกำหนดขนาดของอาร์เรย์แล้ว **หากเขียนหรืออ่านเกินกว่าสมาชิกของอาร์เรย์ได้ โดยคอมไพเลอร์จะไม่แจ้งเตือนแต่อย่างใด**
- การเขียนข้อมูลเกินกว่าขนาดของอาร์เรย์ที่จองไว้ **จะทำให้ข้อมูลที่เขียนลงไปนั้น จะไปทับข้อมูลอื่น หรือไปทับส่วนของโปรแกรม อันจะทำให้ระบบเสียหายได้**

เอกสารอ้างอิง

คณาจารย์ผู้สอนวิชา Introduction to Computer Programming, เอกสารประกอบการบรรยายวิชา 241-101 Introduction to Computer Programming, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยสงขลานครินทร์, 2551

แหล่งศึกษาเพิ่มเติม

ผศ.นราธร สังข์ประเสริฐ. (2563) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-3945-2

โอภาส เอี่ยมสิริวงศ์. (2559) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-2740-4

รศ.ธีรวัฒน์ ประกอบผล. (2562) คู่มือการเขียนโปรแกรมภาษา C ฉบับสมบูรณ์, ซิมพลิฟาย, ISBN: 978-616-26-24896

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase1] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=aDBmXMF9jOU

วีดีโอยูทูป เขียนโปรแกรมภาษา C | สำหรับผู้เริ่มต้น [Phase2] โดย KongRuksiam Tutorial Youtube: https://www.youtube.com/watch?v=wslM0Aqepxs

Mooc การเขียนโปรแกรมคอมพิวเตอร์พื้นฐาน ด้วยภาษาซี 1 | Fundamentals of the C Programming Language https://lms.thaimooc.org/courses/course-v1:KMUTT+KMUTT013+2019/about

Mooc ทักษะการเขียนโปรแกรมคอมพิวเตอร์ด้วยภาษาซี | Computer programming skills with C language https://lms.thaimooc.org/courses/course-v1:SU+SU011+2021/about

Coursera: Programming in C, ผู้สอน: Charles Russell Severance https://www.coursera.org/learn/programming-c

Coursera: ความเชี่ยวชาญพิเศษ การเขียนโปรแกรม C เบื้องต้น, ผู้สอน: Andrew D. Hilton et al. https://www.coursera.org/specializations/c-programming

Playlist Youtube บันทึกการสอนในห้องเรียน, วิชาการโปรแกรมคอมพิวเตอร์, ดร.พงศกร เจริญเนตรกุล 2566 https://www.youtube.com/watch?v=ewEfmiXaRi0&list=PL322g2itJn4mAE1utyOzF-C0SP0BRkej3&pp=gAQBiAQB

แบบฝึกหัดท้ายบท

1. จงอธิบายความหมายของอาร์เรย์ในภาษาซี
2. การเข้าถึงข้อมูลในอาร์เรย์ใช้ดัชนีเริ่มต้นที่ค่าใด
3. วิธีการประกาศอาร์เรย์ 1 มิติในภาษาซีทำอย่างไร
4. อาร์เรย์แบบหลายมิติคืออะไร
5. จงยกตัวอย่างการกำหนดค่าเริ่มต้นให้กับอาร์เรย์แบบ 1 มิติ
6. จงอธิบายความแตกต่างระหว่างตัวแปรธรรมดาและอาร์เรย์
7. การส่งผ่านอาร์เรย์แบบ Call-by-value แตกต่างจาก Call-by-reference อย่างไร
8. ข้อผิดพลาดที่พบบ่อยในการใช้งานอาร์เรย์คืออะไร
9. ในภาษาซี อาร์เรย์จะเก็บข้อมูลในหน่วยความจำอย่างไร
10. ฟังก์ชัน sizeof ใช้หาค่าอะไรกับอาร์เรย์ได้บ้าง
11. จงอธิบายการใช้ลูป for ในการเข้าถึงค่าของอาร์เรย์
12. จงอธิบายวิธีการรับค่าอาร์เรย์จากผู้ใช้โดยใช้ scanf()
13. เมื่อส่งอาร์เรย์เข้าไปในฟังก์ชันแบบส่งทีละ element และส่งทั้งอาร์เรย์ โดยมีการเปลี่ยนแปลงค่าภายในฟังก์ชันจะมีผลต่อตัวแปรอาร์เรย์ต้นฉบับหรือไม่ อย่างไร
14. จงอธิบายการกำหนดค่าเริ่มต้นให้กับอาร์เรย์หลายมิติ
15. จงยกตัวอย่างการใช้งานอาร์เรย์ 2 มิติ
16. การเข้าถึงสมาชิกในอาร์เรย์แบบ 2 มิติต้องใช้ดัชนีอย่างไร
17. จงอธิบายความหมายของการประกาศ int a[10]; ในภาษาซี
18. ในกรณีที่ต้องการคำนวณผลรวมของสมาชิกทั้งหมดในอาร์เรย์ต้องเขียนโค้ดอย่างไร
19. การกำหนดค่าเริ่มต้นเพียงบางส่วนของอาร์เรย์มีผลอย่างไร
20. การใช้ดัชนีเกินขอบเขตของอาร์เรย์จะส่งผลอย่างไร

21. จงยกตัวอย่างการกำหนดค่าเริ่มต้นให้กับอาร์เรย์ 3 มิติ
22. ขนาดของอาร์เรย์ 2 มิติ int matrix[4][3]; มีจำนวนเท่าใด
23. อธิบายการทำงานของฟังก์ชันที่มีพารามิเตอร์เป็นอาร์เรย์
24. จงเขียนตัวอย่างการใช้ฟังก์ชัน strcpy() สำหรับคัดลอกสตริง
25. วิธีการหาขนาดของอาร์เรย์ในหน่วยไบต์ทำอย่างไร
26. จงยกตัวอย่างการใช้งานสายอักขระ (String) ในภาษาซี
27. การส่งผ่านอาร์เรย์เข้าสู่ฟังก์ชันสามารถทำได้กี่วิธี
28. ฟังก์ชัน strlen() ใช้เพื่ออะไร
29. การใช้ลูป for ในการเข้าถึงค่าอาร์เรย์ 2 มิติทำอย่างไร
30. จงอธิบายความหมายของการส่งค่าด้วย Call-by-value และ Call-by-reference

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 7.1

เรื่อง สร้างและแสดง Array

วัตถุประสงค์

1. สามารถเขียนประกาศตัวแปรอาร์เรย์พร้อมกำหนดค่าได้ถูกต้อง
2. สามารถเขียนใช้การวนรอบ for เพื่อแสดงค่าที่อยู่ในตัวแปรอาร์เรย์แต่ละประเภทได้ถูกต้อง

ทบทวนความรู้

การเข้าถึงค่าในอาร์เรย์ด้วยการวนรอบ for

จากโปรแกรมนี้ หากใส่ 10, 20, 30, 40 จะมีผลลัพธ์เป็นอย่างไร

```
#include <stdio.h>
int main()
{
  int num[4],i;
  for(i=0;i<4;i++)
  {
    printf("Enter num[%d]: ",i);
    scanf("%d",&num[i]);
  }
  for(i=0;i<4;i++)
    num[i] = 2*num[i];
  for(i=0;i<4;i++)
    printf("num[%d] = %d\n",i,num[i]);
}
```

Input:
Enter num[0]: 10
Enter num[1]: 20
Enter num[2]: 30
Enter num[3]: 40

Output:
num[0] = 20

num[1] = 40

num[2] = 60

num[3] = 80

Computer Programming 21

คำสั่ง

จงสร้างตัวแปรอาแรย์ดังต่อไปนี้

1. ตัวแปรชนิดจำนวนเต็ม ชื่อ num มีสมาชิก 5 ตัว มีค่าเริ่มต้นเป็น 3 5 7 9 11
2. ตัวแปรชนิดทศนิยม (float) ชื่อ numF มีสมาชิก 5 ตัว มีค่าเริ่มต้นเป็น 0.3 0.5 0.9 0.7 0.11
3. ตัวแปรชนิดทศนิยม (double) ชื่อ numD มีสมาชิก 5 ตัว มีค่าเริ่มต้นเป็น 0.33 0.55 0.77 0.99 0.1111
4. ตัวแปรชนิดตัวอักษร ชื่อ str มีสมาชิก 5 ตัว มีค่าเริ่มต้นเป็น A B C D (แค่ D นะครับ ไม่รวม E)

หลังจากนั้นให้แสดงค่าในตัวแปรอาเรย์ออกมาทั้งหมด **โดยใช้ For Loop

ตัวอย่างผลลัพธ์

```
3 5 7 9 11
0.300000 0.500000 0.900000 0.700000 0.110000
0.330000 0.550000 0.770000 0.990000 0.111100
A B C D
```

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 7.2

เรื่อง โปรแกรมเก็บเกรดเข้า Array - ใช้ loop

วัตถุประสงค์

1. สามารถเขียนประกาศตัวแปรอาร์เรย์พร้อมกำหนดค่าได้ถูกต้อง
2. สามารถเขียนใช้การวนรอบ for เพื่อแสดงค่าที่อยู่ในตัวแปรอาร์เรย์แต่ละประเภทได้ถูกต้อง
3. สามารถเขียนใช้การวนรอบ for เพื่อรับค่าไปเก็บในตัวแปรอาร์เรย์ด้วย scanf ร่วมกับการวนรอบได้ถูกต้อง

ทบทวนความรู้

จงเขียนโปรแกรมสำหรับรับค่าคะแนนของนักศึกษาจำนวน 10 คน และให้พิมพ์ค่าเฉลี่ยของคะแนนทั้งหมด

โดยกำหนดให้ส่วนรับคะแนนจากผู้ใช้และส่วนที่แสดงค่าเฉลี่ยอยู่ใน main สำหรับส่วนที่คำนวณค่าเฉลี่ยให้อยู่ฟังก์ชันชื่อ average โดยให้ส่งอาร์เรย์ที่เก็บคะแนนทั้งของ 10 คนจาก main มาให้ฟังก์ชัน average (คะแนนสามารถเป็นทศนิยมได้)



Computer Programming 52

```
#include <stdio.h>
float average(float num[10]);
int main()
{ int i;  float score[10],avg_score;
  for(i=0;i<10;i++)
  {  printf("Enter score %d : ",i+1);
     scanf("%f",&score[i]);
  }
  avg_score = average(score);
  printf("Average score is %.2f\n",avg_score);
}
float average(float num[10])
{   float total,avg;
    int i;
    total = 0.0;
    for(i=0;i<10;i++)
              total = total + num[i];
    avg = total/10.0;
    return avg;
}
```

```
Output:
Enter score 1: 10
Enter score 2: 20
Enter score 3: 30
Enter score 4: 40
Enter score 5: 50
Enter score 6: 60
Enter score 7: 70
Enter score 8: 80
Enter score 9: 90
Enter score 10: 100
Average score is : 55.00
```

คำสั่ง

จงสร้างตัวแปรอาแรย์ดังต่อไปนี้

1. ตัวแปรชนิดตัวอักษร ชื่อ myName มีสมาชิก 100 ตัว
2. ตัวแปรชนิดทศนิยม (float) ชื่อ myGrade มีสมาชิกเท่ากับจำนวนวิชาที่นักศึกษาลงทะเบียนในเทอมนี้ จำนวน 3 รายวิชา

จากนั้นรับค่าไปเก็บในตัวแปร array ข้างต้น

1. ใช้คำสั่ง scanf("%s",myName); เพื่อรับค่าชื่อนักศึกษเข้าไปเก็บในตัวแปร myName
2. ใช้ loop for ร่วมกับคำสั่ง scanf เพื่อรับเกรดของแต่ละวิชาไปเก็บในตัวแปร myGrade

ตัวอย่างข้อความรับค่าจากผู้ใช้

My grade of course 1 : 4

My grade of course 2 : 3.5

My grade of course 3 : 2

...

[หมายเหตุ] เกรด A ให้ใส่ค่าเป็น 4

เกรด B+ ให้ใส่ค่าเป็น 3.5

เกรด B ให้ใส่ค่าเป็น 3

เกรด C+ ให้ใส่ค่าเป็น 2.5

...

เกรด F ให้ใส่เป็น 0

จากนั้นแสดงค่าตัวแปร array ข้างต้น

1. ใช้คำสั่ง printf("my name is %s \n",myName); เพื่อแสดงชื่อของนักศึกษา
2. ใช้ loop for ร่วมกับคำสั่ง printf เพื่อแสดงเกรดของแต่ละวิชาที่ถูกเก็บใน myGrade โดยมีลักษณะการแสดงผลดังนี้

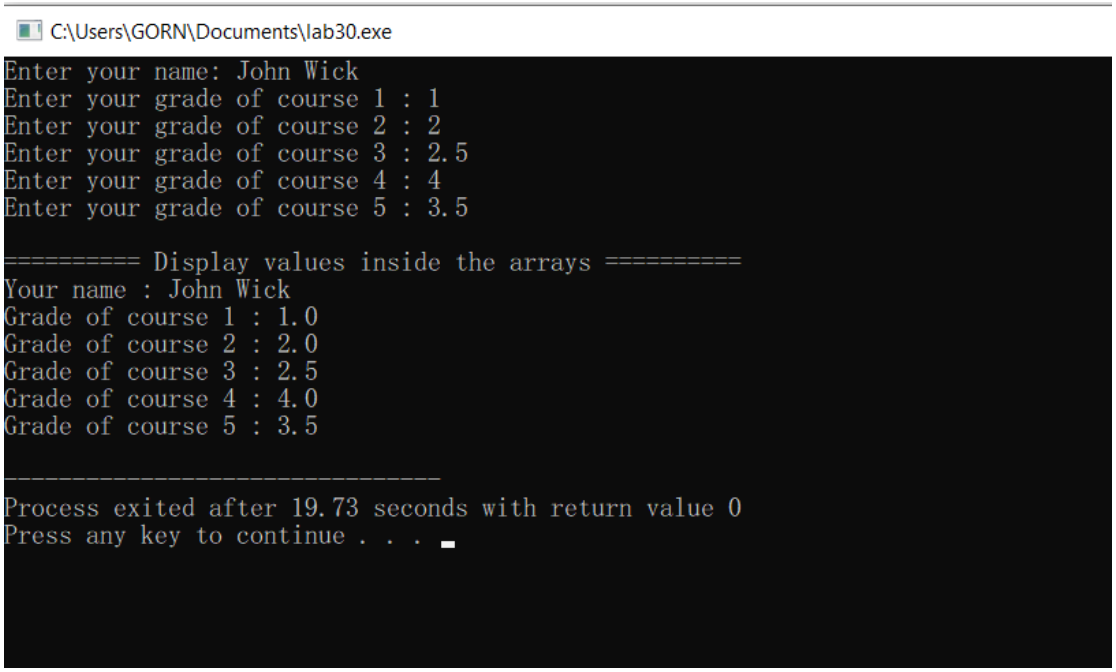
myGrade [course 1] is 4

myGrade [course 2] is 3.5

myGrade [course 3] is 2

...

ตัวอย่างผลลัพธ์



ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

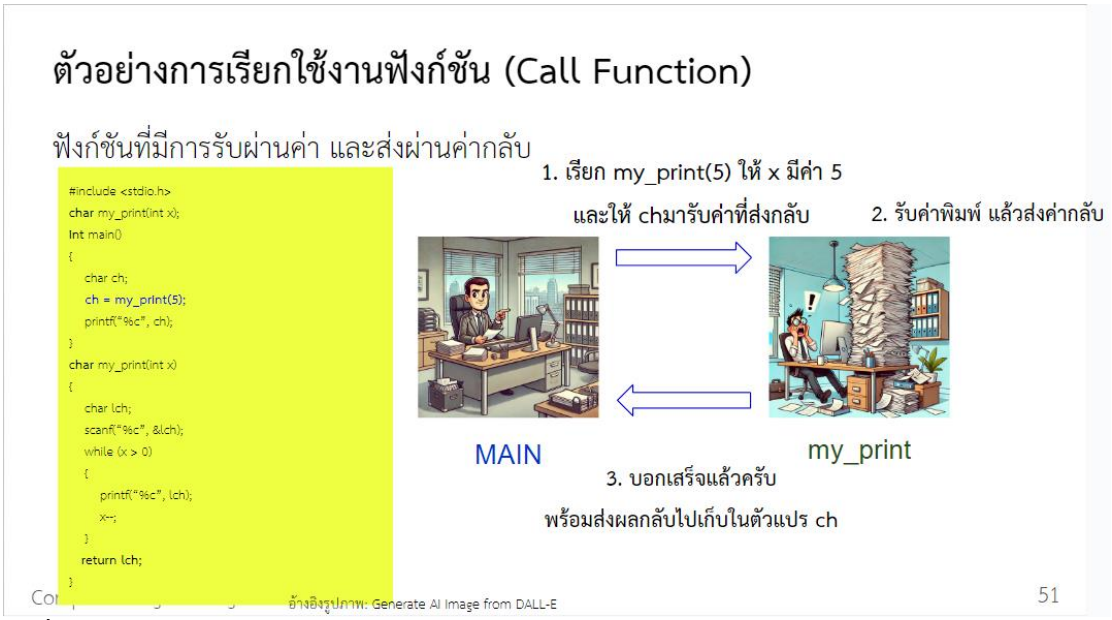
ใบฝึกปฏิบัติการที่ 7.3

เรื่อง Call-by-Value แสดงเกรดเป็นตัวอักษรโดยเรียกฟังก์ชัน

วัตถุประสงค์

1. สามารถเขียนประกาศตัวแปรอาร์เรย์พร้อมกำหนดค่าได้ถูกต้อง
2. สามารถเขียนใช้การวนรอบ for เพื่อแสดงค่าที่อยู่ในตัวแปรอาร์เรย์แต่ละประเภทได้ถูกต้อง
3. สามารถเขียนใช้การวนรอบ for เพื่อรับค่าไปเก็บในตัวแปรอาร์เรย์ด้วย scanf ร่วมกับการวนรอบได้ถูกต้อง
4. สามารถเรียกใช้ฟังก์ชันเพื่อแปลงเกรดแบบเลขเป็นเกรดแบบตัวอักษรได้อย่างถูกต้อง

ทบทวนความรู้



คำสั่ง

ให้นศ.ใช้โค้ดจากใบฝึกปฏิบัติการที่ 7.2 ที่เป็นโปรแกรมเก็บเกรดเข้า Array - ใช้ loop โดยให้เพิ่มฟังก์ชันการแปลงเกรดจากตัวเลขเป็นตัวอักษรดังเงื่อนไขต่อไปนี้

1. ฟังก์ชัน ส่งค่ากลับเป็นตัวอักษร (char)
2. ฟังก์ชันชื่อ gradeChar
3. ฟังก์ชันรับพารามิเตอร์ 1 ตัว เป็นทศนิยม (float) ให้ตั้งชื่อว่า gradeFloat
4. การแปลงเกรดจากจำนวนเป็นตัวอักษรมีเงื่อนไขดังต่อไปนี้
   4.1 เกรด 4 เป็นอักษร A
   4.2 เกรด 3.5 และ 3 เป็นอักษร B
   4.3 เกรด 2.5 และ 2 เป็นอักษร C

4.4 เกรด 1.5 และ 1 เป็นอักษร D

4.5 เกรด 0 เป็นอักษร F

จากนั้นให้การเรียกฟังก์ชันที่ได้สร้างเพื่อแปลงเกรดแบบเลขเป็นเกรดแบบตัวอักษรดังตัวอย่าง

```
char myGradeChar;
myGradeChar = gradeChar(3.5);
```

โดยให้ปรับปรุงเฉพาะโค้ดส่วนการแสดงผลให้เป็นดังนี้

```
myGrade [course 1] is 4  (A)
myGrade [course 2] is 3.5  (B)
myGrade [course 3] is 2  (C)
```

Challenges พิเศษ :

1. **สามารถแสดง เป็น B+ C+ D+ ได้**
2. **รับค่าชื่อวิชาเป็นข้อความได้**

ตัวอย่างผลลัพธ์

```
Enter your name: JohnWick
Enter your CourseName/Grade/Unit of Course 1 : ComputerProgramming 4 3
Enter your CourseName/Grade/Unit of Course 2 : Physic 3 2
Enter your CourseName/Grade/Unit of Course 3 : LabPhysic 2.5 1
Enter your CourseName/Grade/Unit of Course 4 : Electronic 3 2
Enter your CourseName/Grade/Unit of Course 5 : Basketball 4 2

========== Display values inside the arrays ==========
Your name : JohnWick
1 Grade of ComputerProgramming = 4.0 (A) (3.0 Units)
2 Grade of Physic = 3.0 (B) (2.0 Units)
3 Grade of LabPhysic = 2.5 (C+) (1.0 Units)
4 Grade of Electronic = 3.0 (B) (2.0 Units)
5 Grade of Basketball = 4.0 (A) (2.0 Units)
=========================
Your GPX  : 3.45
-------------------------------
```

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 7.4

เรื่อง คำนวณเกรดเฉลี่ย

วัตถุประสงค์

1. สามารถเขียนประกาศตัวแปรอาร์เรย์พร้อมกำหนดค่าได้ถูกต้อง
2. สามารถเขียนใช้การวนรอบ for เพื่อแสดงค่าที่อยู่ในตัวแปรอาร์เรย์แต่ละประเภทได้ถูกต้อง
3. สามารถเขียนใช้การวนรอบ for เพื่อรับค่าไปเก็บในตัวแปรอาร์เรย์ด้วย scanf ร่วมกับการวนรอบได้ถูกต้อง
4. สามารถเรียกใช้ฟังก์ชันเพื่อแปลงเกรดแบบเลขเป็นเกรดแบบตัวอักษรได้อย่างถูกต้อง
5. สามารถนำค่าที่อยู่ในตัวแปรอาร์เรย์มาคำนวณได้

ทบทวนความรู้

```
#include <stdio.h>
float average(float num[10]);
int main()
{ int i;  float score[10],avg_score;
  for(i=0;i<10;i++)
  {  printf("Enter score %d : ",i+1);
     scanf("%f",&score[i]);
  }
  avg_score = average(score);
  printf("Average score is %.2f\n",avg_score);
}
float average(float num[10])
{   float total,avg;
    int i;
    total = 0.0;
    for(i=0;i<10;i++)
                total = total + num[i];
    avg = total/10.0;
    return avg;
}
```

```
Output:
Enter score 1: 10
Enter score 2: 20
Enter score 3: 30
Enter score 4: 40
Enter score 5: 50
Enter score 6: 60
Enter score 7: 70
Enter score 8: 80
Enter score 9: 90
Enter score 10: 100
Average score is : 55.00
```

Computer Programming 53

คำสั่ง

ให้นศ.ใช้โค้ดจากใบฝึกปฏิบัติการที่ 7.3 ที่เป็นโปรแกรมเก็บเกรดเข้า Array - ใช้ loop

จากนั้นสร้างตัวแปรอาเรย์เพิ่มจากโปรแกรมเก็บเกรดและจำนวนหน่วยกิตเข้า Array

1.ตัวแปรอาเรย์ชนิดทศนิยม (float) ชื่อ myGrade (เก็บค่าเกรดของแต่ละรายวิชา 0,1,1.5,2,2.5,3,3.5,4 แทน A B C D + F) มีสมาชิกเท่ากับจำนวนวิชาที่นักศึกษาลงทะเบียนในเทอมนี้

2. ตัวแปรอาเรย์ชนิดทศนิยม (float) ชื่อ myUnit (เก็บค่าหน่วยกิตของแต่ละรายวิชา) มีสมาชิกเท่ากับจำนวนวิชาที่นักศึกษาลงทะเบียนในเทอมนี้

3. ตัวแปรอาเรย์ชนิดทศนิยม (float) ชื่อ myUnitByGrade (เก็บค่าผลคูณของหน่วยกิตและเกรดของแต่ละรายวิชา) มีสมาชิกเท่ากับจำนวนวิชาที่นักศึกษาลงทะเบียนในเทอมนี้

จากนั้นให้คำนวณเกรดเฉลี่ยด้วยวิธีต่อไปนี้

1. ใช้ loop for เพื่อคำนวณหน่วยกิจที่ได้ตามเกรด โดยใช้สมการดังนี้

myUnitByGrade [i] = courseUnit[i] * myGrade[i]

2. ใช้ loop for เพื่อคำนวณผลรวมหน่วยกิจที่ได้ตามเกรด โดยเก็บผลรวมไว้ในตัวแปร sumMyUnitByGrade

3. ใช้ loop for เพื่อคำนวณผลรวมหน่วยกิจของทุกวิชาจากตัวแปร courseUnit โดยเก็บผลรวมไว้ในตัวแปร sumCourseUnit

4. คำนวณและแสดงค่าเกรดเฉลี่ย โดยใช้สมการดังนี้

avgGrade = sumMyUnitByGrade/sumCourseUnit

ตัวอย่างผลลัพธ์

```
Enter your name: JohnWick
Enter your CourseName/Grade/Unit of Course 1 : ComputerProgramming 4 3
Enter your CourseName/Grade/Unit of Course 2 : Physic 3 2
Enter your CourseName/Grade/Unit of Course 3 : LabPhysic 2.5 1
Enter your CourseName/Grade/Unit of Course 4 : Electronic 3 2
Enter your CourseName/Grade/Unit of Course 5 : Basketball 4 2

========== Display values inside the arrays ==========
Your name : JohnWick
1 Grade of ComputerProgramming = 4.0 (3.0 Units)
2 Grade of Physic = 3.0 (2.0 Units)
3 Grade of LabPhysic = 2.5 (1.0 Units)
4 Grade of Electronic = 3.0 (2.0 Units)
5 Grade of Basketball = 4.0 (2.0 Units)
==========================
Your GPX  : 3.45
------------------------------
```

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 7.5

เรื่อง array 2 มิติ - ประกาศตัวแปร รับค่า แสดงค่า

วัตถุประสงค์

1. สามารถเขียนประกาศตัวแปรอาร์เรย์หลายมิติได้ถูกต้อง
2. สามารถเขียนใช้การวนรอบ for เพื่อแสดงค่าที่อยู่ในตัวแปรอาร์เรย์หลายมิติแต่ละประเภทได้ถูกต้อง
3. สามารถเขียนใช้การวนรอบ for เพื่อรับค่าไปเก็บในตัวแปรอาร์เรย์หลายมิติด้วย scanf ร่วมกับการวนรอบได้ถูกต้อง

ทบทวนความรู้

จงเขียนโปรแกรมสำหรับรับข้อความจากผู้ใช้ 4 ข้อความ โดยแต่ละข้อความมีความยาวไม่ถึง 20 ตัวอักษร จากนั้นพิมพ์ค่าความยาวของแต่ละข้อความออกทางหน้าจอ

(ฟังก์ชันที่ใช้ในการหาความยาวของข้อความคือ strlen)

```
#include <stdio.h>
#include <string.h>
int main()
{
  char texts[4][20];
  int i;
  for(i=0;i<4;i++) {
    printf("Enter text %d : ",i+1);
    scanf("%s",&texts[i]);
  }
  for(i=0;i<4;i++)
    printf("%s : %d\n",texts[i],strlen(texts[i]));
}
```

Computer Programming 91

คำสั่ง

1. จงสร้างตัวแปร อาเรย์ 2 มิติ ตามเงื่อนไขดังต่อไปนี้

    1.1 สร้างตัวแปร ชนิด char ชื่อ names ขนาด 5 แถว 100 หลัก
    เพื่อใช้สำหรับเก็บชื่อ คน 5 คน

    1.2 สร้างตัวแปร ชนิด int ชื่อ age_weight_height ขนาด 5 แถว 3 หลัก
    เพื่อใช้เก็บค่า อายุ น้ำหนัก และ ส่วนสูง ของคนทั้ง 5 คน

2. จงใช้ลูป for 1 ชั้น ร่วมกับคำสั่ง scanf เพื่อรับค่า ชื่อ อายุ น้ำหนักและส่วนสูง ของคนทีละคน
    โดยมีรูปแบบการรับค่าดังนี้

Enter name, age, weight and height: Tony 18 50 165

3. จงใช้ลูป for 1 ชั้น ร่วมกับคำสั่ง printf เพื่อแสดงค่า ชื่อ อายุ น้ำหนักและส่วนสูง ของคนทีละคน

โดยมีรูปแบบการแสดงค่าดังนี้

Name: Tony

Age: 18 years

Weight: 50 k.g.

Height: 165 cm.

//ตัวอย่างโปรแกรม

```
#include <stdio.h>
int main(){
	char names[3][30];
	int weight_height[3][2];
	scanf("%s %d %d",	&names[0], &weight_height[0][0], &weight_height[0][1]);
	printf("N:%s W:%d H:%d", names[0], weight_height[0][0], weight_height[0][1]);
	return 0;
}
```

สามารถ copy ข้อความเพื่อทดสอบโปรแกรมได้ >> https://pastebin.com/JEcsvHFm

โดยการ copy และคลิกขวาในหน้าจอ console ของโปรแกรมที่รันรับค่าอยู่

ตัวอย่างผลลัพธ์

```
Enter name age weight height of [0] :SuperMan 250 400 187
Enter name age weight height of [1] :WonderWomen 2500 45 175
Enter name age weight height of [2] :TheFLASH 20 59 175
Enter name age weight height of [3] :BatMann 45 62 189
Enter name age weight height of [4] :AquaMan 220 55 186
_____________________________________________
Name -> SuperMan
        Age -> 250 Weight -> 400 Height -> 187
Name -> WonderWomen
        Age -> 2500 Weight -> 45 Height -> 175
Name -> TheFLASH
        Age -> 20 Weight -> 59 Height -> 175
Name -> BatMann
        Age -> 45 Weight -> 62 Height -> 189
Name -> AquaMan
        Age -> 220 Weight -> 55 Height -> 186
```

แผนการสอนเรื่อง บทที่ 8 การประยุกต์ใช้โปรแกรมสำเร็จรูปทางวิศวกรรม Scilab

สัปดาห์ที่ 15

จำนวนชั่วโมงที่ใช้สอน 5 ชั่วโมง

วัตถุประสงค์ เพื่อให้นักศึกษาสามารถ

1. อธิบายพื้นฐานการใช้งานโปรแกรม Scilab ในการคำนวณเชิงวิศวกรรม คณิตศาสตร์ และวิทยาศาสตร์
2. ระบุขั้นตอนการสร้างและจัดการตัวแปรต่าง ๆ ในโปรแกรม Scilab เช่น Scalar, Matrix และ String
3. เลือกใช้โครงสร้างเงื่อนไขและการวนรอบใน Scilab เพื่อการตัดสินใจและการคำนวณซ้ำอย่างเหมาะสม
4. เขียนฟังก์ชันใน Scilab ที่มีอินพุตและเอาต์พุต
5. สร้างกราฟข้อมูลเชิงวิศวกรรมในรูปแบบ 2D และ 3D โดยใช้โปรแกรม Scilab
6. ประยุกต์ใช้โปรแกรม Scilab ในการจำลองและวิเคราะห์ระบบทางวิศวกรรม เช่น การคำนวณพลังงานจลน์และการคำนวณแรง
7. วิเคราะห์ข้อมูลเชิงวิศวกรรมและนำเสนอ ผ่านการสร้างกราฟและการแสดงผลข้อมูลใน Scilab

หัวข้อบรรยาย

บทที่ 8 การประยุกต์ใช้โปรแกรมสำเร็จรูปทางวิศวกรรม Scilab

8.1 ความรู้พื้นฐานเกี่ยวกับโปรแกรม Scilab

8.2 ความสามารถของ Scilab

8.3 การสร้างและใช้งานตัวแปร

8.4 โครงสร้างเงื่อนไขและการวนรอบของ Scilab

8.5 การเขียนฟังก์ชันใน Scilab

8.6 การวาดกราฟใน Scilab

8.7 การประยุกต์ใช้ Scilab ในงานวิศวกรรม

กิจกรรมการเรียนการสอน

1. การบรรยายในชั้นเรียน
2. การให้ผู้เรียนทดลองเขียนรหัสต้นฉบับและทดลองรันโปรแกรมระหว่างบรรยาย
3. การให้ผู้เรียนทำใบฝึกปฏิบัติการ
4. การเล่นเกมส์ตอบคำถามทบทวนความเข้าใจของทฤษฎีด้วย Quizizz (หากเวลาเรียนพอ)

สื่อการสอน

1. เอกสารประกอบการสอนแบบภาพเลื่อนหรือภาพนิ่ง
2. ใบฝึกปฏิบัติการ
3. โปรแกรม Scilab
4. การเข้าถึงข้อมูลจากอินเทอร์เน็ต

การประเมินผลการเรียนรู้

ผลการเรียนรู้	วิธีการประเมิน	สัปดาห์ที่ประเมิน	สัดส่วนของการประเมินผล
1. คุณธรรม จริยธรรม	1.1 ประเมินจากการเข้าชั้นเรียน การส่งแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการตามกำหนดเวลาที่มอบหมาย 1.2 ประเมินจากพฤติกรรมของนักศึกษาในการปฏิบัติตามกฎระเบียบและข้อบังคับต่างๆ ทั้งในข้อบังคับในชั้นเรียน ข้อบังคับของมหาวิทยาลัย 1.3 ประเมินจากการคัดลอกแบบฝึกหัดท้ายบทและใบฝึกปฏิบัติการ 1.4 ประเมินจากการมีส่วนร่วมในห้องเรียนและการปิดคอมพิวเตอร์ เก็บขยะ เก็บเก้าอี้หลังจากใช้งานเสร็จ	ตลอดภาคการศึกษา	10%
2. ความรู้	2.1 แบบฝึกหัดท้ายบท 2.2 การสอบกลางภาคเรียนและปลายภาคเรียน	2-8, 11-16 9, 17	20% 40%
3. ทักษะทางปัญญา	3.1 ประเมินจากใบฝึกปฏิบัติการ	ตลอดภาคการศึกษา	20%
4. ทักษะความสัมพันธ์ระหว่างบุคคลและความรับผิดชอบ	4.1 พิจารณาจากการทำใบฝึกปฏิบัติการ 4.3 ประเมินจากพฤติกรรมในการทำงานกลุ่ม การระดมความคิดเห็น การตอบคำถามในชั้นเรียน	ตลอดภาคการศึกษา	5%
5. ทักษะวิเคราะห์เชิงตัวเลข การสื่อสารและการใช้เทคโนโลยีสารสนเทศ	5.1 การทดสอบหรือการประเมินจากงานที่มอบหมาย 5.2 สังเกตการใช้คอมพิวเตอร์ในการเขียนโปรแกรม การทำใบฝึกปฏิบัติการ และการค้นหาข้อมูลเพิ่มเติมจากอินเทอร์เน็ต	ตลอดภาคการศึกษา	5%
	รวม		100%

อ่าน Slide เวอร์ชันล่าสุดได้ที่ Google Slide

https://s.coecore.com/scp8

การโปรแกรมคอมพิวเตอร์

บทที่ 8 การประยุกต์ใช้โปรแกรมสำเร็จรูปทางวิศวกรรม Scilab

(Scilab : Application Program for Engineering)

ผู้สอน ดร.พงศกร เจริญเนตรกุล

Outline

- ความรู้พื้นฐานของโปรแกรม Scilab
- การสร้างและใช้งานตัวแปรในโปรแกรมScilab
- การตรวจสอบและแสดงค่าของตัวแปร
- โครงสร้างเงื่อนไขและการวนรอบของ Scilab
- การเขียนฟังก์ชันใน Scilab
- การวาดกราฟใน Scilab

ความรู้พื้นฐานของโปรแกรม Scilab

Scilab เป็น**โปรแกรมคอมพิวเตอร์ที่ใช้สำหรับการคำนวณทางวิทยาศาสตร์ วิศวกรรม และคณิตศาสตร์** มีความสามารถในการประมวลผลสมการคณิตศาสตร์ที่ซับซ้อน การสร้างกราฟ รวมถึงการจำลองระบบต่าง ๆ ซึ่ง Scilab สามารถนำไปใช้ในหลายด้าน

- **Scilab เป็น Open Source ไม่มีค่าใช้จ่าย** ทำให้เป็นทางเลือกที่ดีสำหรับวิศวกรหรือนักศึกษาที่ต้องการเครื่องมือคำนวณทางวิทยาศาสตร์ วิศวกรรม
- รองรับการคำนวณทางวิศวกรรมหลายประเภท รวมถึงการเขียนโปรแกรมแบบอัลกอริธึมเพื่อจำลองและแก้ปัญหาทางคณิตศาสตร์

ความสามารถของ Scilab ด้านวิศวกรรม

การคำนวณเชิงตัวเลข (Numerical Computation)
Scilab รองรับการคำนวณทางคณิตศาสตร์ที่ซับซ้อน เช่น การแก้สมการเชิงเส้นและไม่เชิงเส้น การคำนวณแคลคูลัส การทำงานกับเมทริกซ์ และการคำนวณเชิงฟังชันคณิตเชิงสัญลักษณ์ ซึ่งมีประโยชน์ในการวิเคราะห์เชิงวิศวกรรม

การจำลองระบบ (System Simulation)
สามารถใช้ Scilab ในการจำลองระบบเชิงพลศาสตร์ เช่น การจำลองระบบไฟฟ้า ระบบกลไก และระบบควบคุม ซึ่งช่วยในการทดสอบและพัฒนาระบบก่อนที่จะนำไปใช้งานจริง

การสร้างแบบจำลองทางคณิตศาสตร์ (Mathematical Modeling)
สามารถใช้ Scilab เพื่อสร้างแบบจำลองทางคณิตศาสตร์สำหรับระบบหรือกระบวนการต่าง ๆ เช่น ระบบควบคุมการผลิต ระบบพลังงาน หรือกระบวนการทางเคมี

การควบคุมอัตโนมัติ (Automatic Control)
Scilab มีเครื่องมือและฟังก์ชันที่ช่วยในการออกแบบและวิเคราะห์ระบบควบคุม เช่น PID controllers การวิเคราะห์เสถียรภาพของระบบ และการจำลองการตอบสนองของระบบควบคุม

การวิเคราะห์สัญญาณและข้อมูล (Signal and Data Processing)
Scilab รองรับการวิเคราะห์สัญญาณ เช่น การกรองสัญญาณ การวิเคราะห์ในโดเมนเวลาและความถี่ และการแปลงฟูเรียร์ ซึ่งเป็นพื้นฐานสำคัญในการประมวลผลสัญญาณในงานวิศวกรรมไฟฟ้าและอิเล็กทรอนิกส์

การประมวลผลภาพ (Image Processing)
Scilab สามารถใช้สำหรับการประมวลผลภาพ เช่น การกรองภาพ การตรวจจับขอบ การปรับปรุงคุณภาพภาพ และการวิเคราะห์รูปภาพ ซึ่งเป็นประโยชน์ในงานด้านวิศวกรรมคอมพิวเตอร์และหุ่นยนต์

การแสดงผลข้อมูลและการสร้างกราฟ (Data Visualization and Graph Plotting)
Scilab มีความสามารถในการสร้างกราฟ 2D และ 3D สำหรับการแสดงผลข้อมูลเชิงวิศวกรรม ซึ่งช่วยในการวิเคราะห์และตีความข้อมูล

การเชื่อมต่อกับฮาร์ดแวร์ (Hardware Interface)
Scilab สามารถเชื่อมต่อกับอุปกรณ์ภายนอก เช่น ไมโครคอนโทรลเลอร์ เซนเซอร์ หรือระบบฝังตัวอื่น ๆ ผ่านการสื่อสารโปรโตคอลมาตรฐาน เช่น Serial หรือ CAN ซึ่งเป็นประโยชน์ในงานวิศวกรรม IoT และระบบฝังตัว

การติดตั้ง Scilab

สามารถดูวิธีการติดตั้ง Scilab ได้ใน Youtube: **https://www.youtube.com/watch?v=2qnxLo5uIQo**

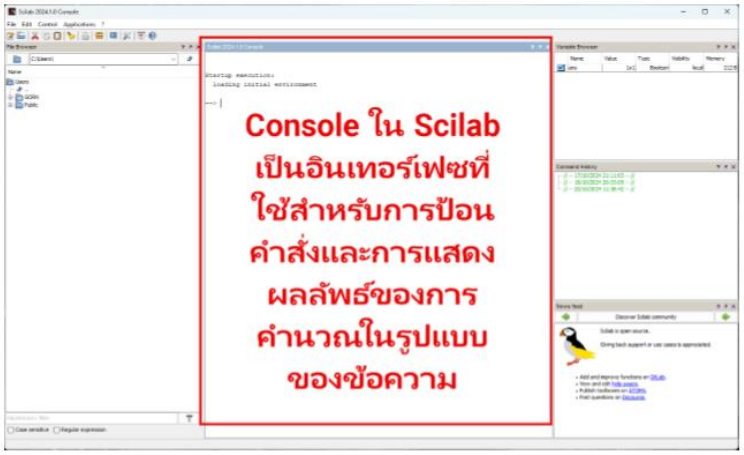


Scan เพื่อดูวิธีการติดตั้ง Scilab

หน้าต่าง Scilab

Scilab Console

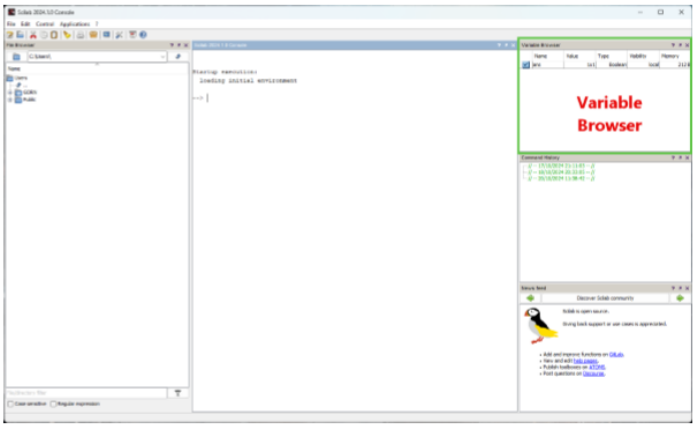
- การป้อนคำสั่งและโค้ด
- การแสดงผลลัพธ์
- การทดสอบโค้ดสั้น ๆ
- การดีบักและตรวจสอบข้อผิดพลาด (Debugging)
- การเรียกใช้งานตัวแปรและฟังก์ชัน (Variable and Function Execution)
- การจัดการไฟล์และสคริปต์



Computer Programming 7

หน้าต่าง Scilab

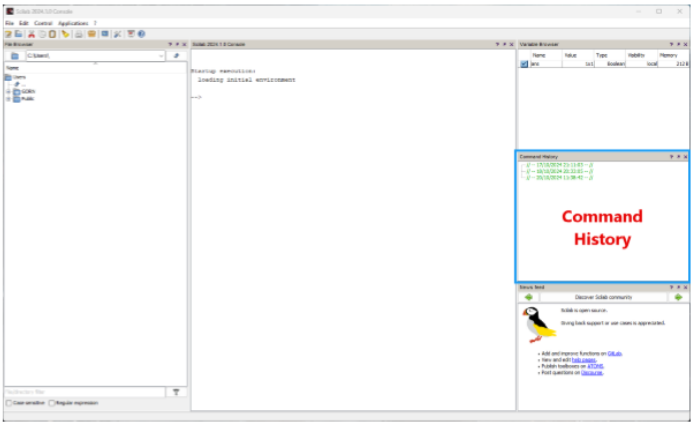
Variable Browser ใน Scilab เป็นเครื่องมือที่ช่วยให้ผู้ใช้สามารถตรวจสอบ แก้ไข ลบ และจัดการตัวแปรต่าง ๆ ได้อย่างมีประสิทธิภาพในระหว่างการทำงาน ทำให้การควบคุมและการติดตามข้อมูลที่เกิดขึ้นใน session ของ Scilab ง่ายขึ้น



Computer Programming 8

หน้าต่าง Scilab

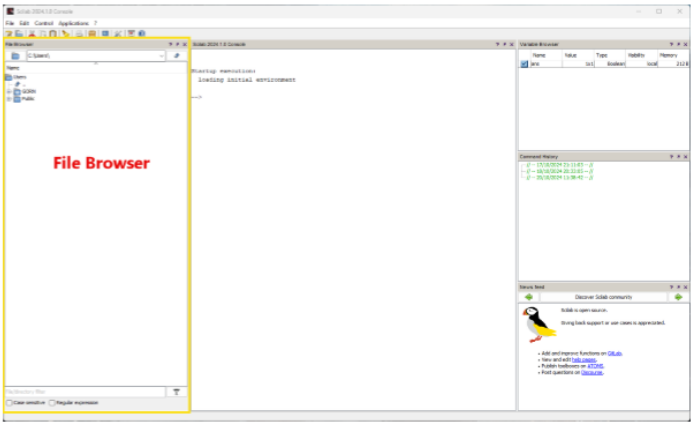
Command History ใน Scilab เป็นเครื่องมือที่บันทึกและแสดงรายการคำสั่งที่ผู้ใช้เคยป้อนใน console ระหว่าง session การทำงาน



หน้าต่าง Scilab

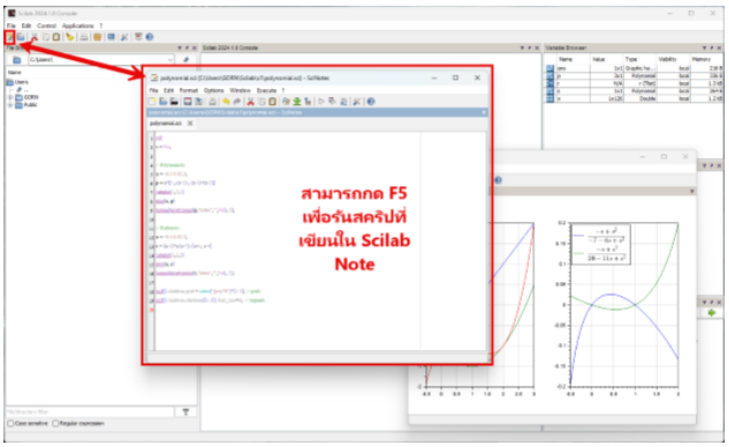
File Browser ใน Scilab File Browser ใน Scilab เป็นเครื่องมือที่ช่วยให้ผู้ใช้สามารถจัดการและเข้าถึงไฟล์และโฟลเดอร์ในระบบไฟล์ของคอมพิวเตอร์ได้โดยตรงจากภายใน Scilab ซึ่งมีหน้าที่และการใช้งานหลัก ๆ ดังนี้

- **การเข้าถึงและจัดการไฟล์** เช่น ไฟล์สคริปต์ (script files), ไฟล์ข้อมูล (data files), หรือไฟล์กราฟิก
- **การโหลดและรันสคริปต์** โดยผู้ใช้สามารถเลือกไฟล์สคริปต์ (.sce หรือ .sci) จาก File Browser และเรียกใช้ไฟล์นั้นโดยตรงใน Scilab โดยไม่ต้องพิมพ์คำสั่งเพื่อระบุตำแหน่งไฟล์
- **การบันทึกไฟล์** โดยผู้ใช้สามารถบันทึกไฟล์ผลลัพธ์ต่าง ๆ เช่น ไฟล์สคริปต์ ไฟล์ข้อมูล หรือไฟล์กราฟิกลงในตำแหน่งที่ต้องการได้โดยตรงผ่าน File Browser รวมถึงการเลือกสถานที่บันทึกไฟล์ใหม่ที่ถูกสร้างขึ้น



Scilab Note

Scilab Note เป็นเครื่องมือที่มีไว้เพื่อการเขียน แก้ไข และจัดการสคริปต์ใน Scilab โดยเฉพาะ โดยช่วยให้การทำงานกับโค้ดง่ายขึ้น ผ่านการเน้นไวยากรณ์ การจัดการไฟล์หลายไฟล์ การดีบักโค้ด และการรันโค้ด



Computer Programming 11

การสร้างและใช้งานตัวแปรในโปรแกรมScilab

Computer Programming 12

การสร้างและใช้งานตัวแปรในโปรแกรม Scilab

ตัวเลข (Scalar)

```
x = 10;      // ตัวแปรชนิดจำนวนเต็ม
y = 3.14;    // ตัวแปรชนิดจำนวนทศนิยม
```

เมทริกซ์ (Matrix)

```
A = [1, 2; 3, 4];  // เมทริกซ์ 2x2
B = ones(3, 3);    // เมทริกซ์ 3x3 ที่มีค่าทุกตัวเป็น 1
C = rand(4, 4);    // เมทริกซ์ 4x4 ที่มีค่าสุ่ม
```


การสร้างและใช้งานตัวแปรในโปรแกรม Scilab

สตริง (String)

```
name = 'Scilab';  // ตัวแปรชนิดสตริง
```

พหุนาม (Polynomial)

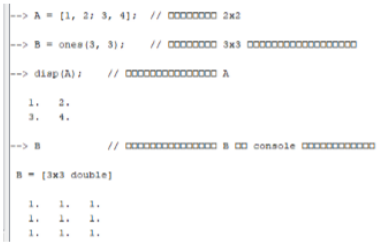
```
p = poly([1, -2, 3], 'x', 'coeff');  // สร้างพหุนาม x^2 - 2x + 3
```

การตรวจสอบและแสดงค่าของตัวแปร

การตรวจสอบและแสดงค่าของตัวแปร

สามารถใช้คำสั่ง disp หรือเพียงพิมพ์ชื่อตัวแปรใน console เพื่อแสดงค่าของตัวแปรได้

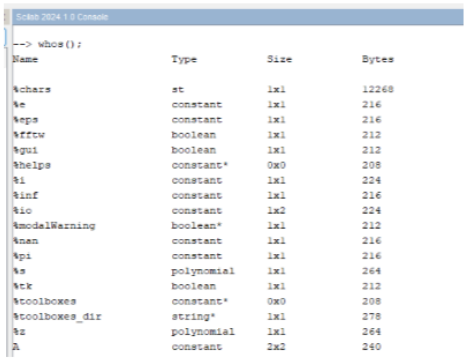
```
A = [1, 2; 3, 4];  // เมทริกซ์ 2x2
B = ones(3, 3);    // เมทริกซ์ 3x3 ที่มีค่าทุกตัวเป็น
disp(A);    // แสดงค่าเมทริกซ์ A
B           // พิมพ์ชื่อตัวแปร B ใน console เพื่อแสดงค่า
```



การตรวจสอบและแสดงค่าของตัวแปร

สามารถใช้คำสั่ง whos() เพื่อดูตัวแปรทั้งหมดที่ถูกสร้างไว้ในปัจจุบันและประเภทของตัวแปรเหล่านั้น

```
whos();  // แสดงรายการตัวแปรทั้งหมดใน workspace
```



```
--> whos();
Name                Type          Size          Bytes

%chars              st            1x1           12268
%e                  constant      1x1           216
%eps                constant      1x1           216
%fftw               boolean       1x1           212
%gui                boolean       1x1           212
%helps              constant*     0x0           208
%i                  constant      1x1           224
%inf                constant      1x1           216
%io                 constant      1x2           224
%modalWarning       boolean*      1x1           212
%nan                constant      1x1           216
%pi                 constant      1x1           216
%s                  polynomial    1x1           264
%tk                 boolean       1x1           212
%toolboxes          constant*     0x0           208
%toolboxes_dir      string*       1x1           278
%z                  polynomial    1x1           264
A                   constant      2x2           240
```


การตรวจสอบและแสดงค่าของตัวแปร

การจัดการตัวแปรในหน่วยความจำ

หากไม่ต้องการใช้งานตัวแปรแล้ว สามารถลบตัวแปรออกจากหน่วยความจำได้ด้วยคำสั่ง clear

```
clear A;   // ลบตัวแปร A

clear;     // ลบตัวแปรทั้งหมด
```

การใช้งานตัวแปรในสมการหรือฟังก์ชัน

```
r = sqrt(a^2 + b^2);  // คำนวณค่า r จากตัวแปร a และ b
```

การตรวจสอบและแสดงค่าของตัวแปร

การใส่คำอธิบายโปรแกรมใช้เครื่องหมาย // หรือที่เรียกว่า comment

```
a = 5;  // กำหนดค่าให้กับตัวแปร a
b = 10; // กำหนดค่าให้กับตัวแปร b
sum = a + b;  // หาผลรวมของ a และ b
// คำสั่งนี้ใช้สำหรับแสดงผลลัพธ์
disp(sum);
```


การตรวจสอบและแสดงค่าของตัวแปร

การใส่เครื่องหมายเชื่อมโยงประโยคระหว่างบรรทัด (Line Continuation)

สามารถแบ่งคำสั่งหรือสมการที่ยาวออกเป็นหลายบรรทัดได้โดยใช้เครื่องหมาย .. ซึ่งเรียกว่า Line Continuation ใน Scilab เมื่อต้องการเชื่อมโยงคำสั่งหรือสมการระหว่างบรรทัด

```
// การคำนวณค่าเฉลี่ยจากหลายค่าโดยใช้เครื่องหมายเชื่อมประโยค
mean_value = (a + b + 15 + 25 + 35 + 45) / ..
             6;  // หาค่าเฉลี่ยจากผลรวมทั้งหมด
// การสร้างเมทริกซ์ที่มีหลายบรรทัด
A = [1, 2, 3; ..
     4, 5, 6; ..
     7, 8, 9];
```

โครงสร้างเงื่อนไขและการวนรอบของ Scilab

โครงสร้างเงื่อนไขและการวนรอบของ Scilab

โครงสร้างเงื่อนไข (Conditional Structures)

Scilab มีโครงสร้างเงื่อนไขที่ใช้ตรวจสอบค่าต่าง ๆ และทำงานตามเงื่อนไขที่กำหนด

1 คำสั่ง if-then-else

ใช้สำหรับตรวจสอบเงื่อนไขและทำงานตามที่กำหนดเมื่อเงื่อนไขนั้นเป็นจริง (true) หรือไม่จริง (false)

```
if condition then
    // คำสั่งที่จะทำงานเมื่อเงื่อนไขเป็นจริง
elseif another_condition then
    // คำสั่งที่จะทำงานเมื่อเงื่อนไขที่สองเป็นจริง
else
    // คำสั่งที่จะทำงานเมื่อเงื่อนไขทั้งหมดไม่เป็นจริง
end
```

Example Code:

```
a = 5;
if a > 10 then
    disp('a มากกว่า 10');
elseif a == 5 then
    disp('a เท่ากับ 5');
else
    disp('a น้อยกว่า 5');
end
```

โครงสร้างเงื่อนไขและการวนรอบของ Scilab

โครงสร้างเงื่อนไข (Conditional Structures)

2 คำสั่ง select-case

คำสั่งนี้ใช้เลือกการทำงานตามค่าที่ตรงกับเงื่อนไขเฉพาะ

```
select variable
   case value1 then
      // คำสั่งเมื่อค่าเป็น value1
   case value2 then
      // คำสั่งเมื่อค่าเป็น value2
   else
      // คำสั่งเมื่อไม่ตรงกับค่าใด ๆ
end
```

Example Code:

```
x = 2;
select x
   case 1 then
      disp('x เท่ากับ 1');
   case 2 then
      disp('x เท่ากับ 2');
   else
      disp('x ไม่เท่ากับ 1 หรือ 2');
end
```


โครงสร้างเงื่อนไขและการวนรอบของ Scilab

โครงสร้างการวนรอบ (Loops)

1 คำสั่ง for

ใช้สำหรับการวนรอบเมื่อทราบจำนวนรอบ

```
for variable = start:step:end
   // คำสั่งที่จะทำซ้ำ
end
```

Example Code:

```
for i = 1:2:10
   disp(i);  // จะแสดงค่า i จาก 1 ถึง 9 โดยเพิ่มทีละ 2
end
```

โครงสร้างเงื่อนไขและการวนรอบของ Scilab

โครงสร้างการวนรอบ (Loops)

2 คำสั่ง while

ใช้เมื่อยังไม่ทราบจำนวนครั้งที่ต้องการวนซ้ำ แต่ต้องการให้วนซ้ำจนกว่าเงื่อนไขจะเป็นเท็จ

```
while condition
    // คำสั่งที่จะทำซ้ำตราบใดที่เงื่อนไขยังเป็นจริง
end
```

Example Code:

```
x = 0;
while x < 5 do
    disp(x);
    x = x + 1;  // เพิ่มค่า x ไปเรื่อย ๆ จนกว่าจะถึง 5
end
```


โครงสร้างเงื่อนไขและการวนรอบของ Scilab

โครงสร้างการวนรอบ (Loops)

3 คำสั่ง break และ continue

break จะใช้สำหรับหยุดการวนรอบเมื่อถึงจุดที่กำหนด

continue จะใช้เพื่อข้ามคำสั่งที่เหลือและเริ่มรอบถัดไปทันที

Example Code:

```
for i = 1:10
    if i == 5 then
        break;  // หยุดการวนรอบเมื่อ i เท่ากับ 5
    end
    disp(i);
end
```

Example Code:

```
for i = 1:10
    if i == 5 then
        continue;  // ข้ามการทำงานเมื่อ i เท่ากับ 5
    end
    disp(i);
end
```

การเขียนฟังก์ชันใน Scilab

การเขียนฟังก์ชันใน Scilab

การเขียนฟังก์ชันใน Scilab เป็นวิธีที่ช่วยในการสร้างโค้ดที่สามารถนำกลับมาใช้ใหม่ได้ ช่วยให้โปรแกรมมีโครงสร้างที่ชัดเจนยิ่งขึ้น

1. โครงสร้างพื้นฐานของฟังก์ชันใน Scilab

การประกาศฟังก์ชันใน Scilab ใช้คำสั่ง function และสิ้นสุดด้วยคำสั่ง endfunction โดยมีโครงสร้างพื้นฐานดังนี้

```
function [output1, output2, ..., outputN] = function_name(input1, input2, ..., inputM)
    // โค้ดภายในฟังก์ชัน
    output1 = ...; // กำหนดค่าผลลัพธ์ที่ต้องการ
    output2 = ...;
    ...
endfunction
```

โดยที่
output1, output2, ..., outputN คือ ค่าที่ฟังก์ชันจะส่งกลับไปหลังจากทำงานเสร็จ (สามารถมีหลายค่าหรือไม่มีเลยก็ได้)
function_name คือ ชื่อของฟังก์ชันที่ผู้ใช้กำหนด
input1, input2, ..., inputM คือ ค่าหรือข้อมูลที่ถูกส่งเข้ามาในฟังก์ชันตอนที่เรียกใช้

การเขียนฟังก์ชันใน Scilab

ตัวอย่างการเขียนฟังก์ชัน

ตัวอย่าง 1 ฟังก์ชันที่ไม่มีอินพุตและเอาต์พุต

```
function greet()
    disp('Hello, welcome to Scilab!');
endfunction
```

ฟังก์ชันนี้ไม่มีอินพุตและเอาต์พุต เมื่อเรียกใช้จะทำการแสดงข้อความ "Hello, welcome to Scilab!"

การเรียกใช้

```
greet();  // แสดงผล "Hello, welcome to Scilab!"
```


การเขียนฟังก์ชันใน Scilab

ตัวอย่างการเขียนฟังก์ชัน

ตัวอย่าง 2 ฟังก์ชันที่มีอินพุตและเอาต์พุต

```
function result = square_number(x)
    result = x^2;
endfunction
```

ฟังก์ชันนี้รับอินพุตเป็นตัวเลข x และคืนค่า x^2 กลับไปเป็นผลลัพธ์

การเรียกใช้

```
y = square_number(4);  // y จะมีค่าเท่ากับ 16
```

การเขียนฟังก์ชันใน Scilab

ตัวอย่างการเขียนฟังก์ชัน

ตัวอย่าง 3 ฟังก์ชันที่มีหลายอินพุตและหลายเอาต์พุต

```
function [sum, product] = calculate(a, b)
    sum = a + b;
    product = a * b;
endfunction
```

ฟังก์ชันนี้รับอินพุต 2 ค่า a และ b และคืนค่าผลรวม (sum) และผลคูณ (product) กลับมา

การเรียกใช้

```
[s, p] = calculate(3, 5); // s จะเท่ากับ 8 และ p จะเท่ากับ 15
```


การวาดกราฟใน Scilab

การวาดกราฟใน Scilab

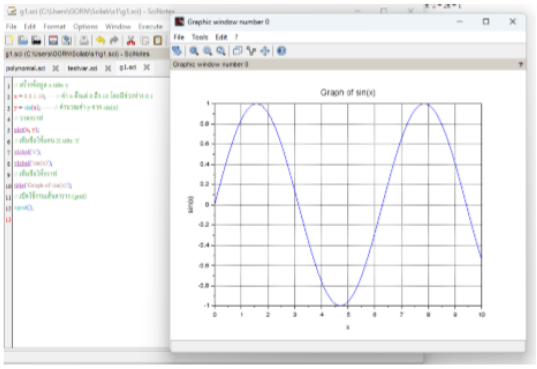
การวาดกราฟใน Scilab เป็นเครื่องมือสำคัญสำหรับการแสดงข้อมูลในรูปแบบภาพ ไม่ว่าจะเป็นการวิเคราะห์ข้อมูล การจำลองระบบ หรือการนำเสนอผลการทดลอง Scilab รองรับการสร้างกราฟทั้ง 2D และ 3D

1. การวาดกราฟพื้นฐาน (2D Plot)

การสร้างกราฟแบบ 2 มิติสามารถทำได้ด้วยฟังก์ชัน plot() โดยกราฟที่สร้างจะเป็นกราฟเส้นที่แสดงความสัมพันธ์ระหว่างค่า x และ y

ตัวอย่าง

```
// สร้างข้อมูล x และ y
x = 0:0.1:10;       // ค่า x ตั้งแต่ 0 ถึง 10 โดยมีช่วงห่าง 0.1
y = sin(x);         // คำนวณค่า y จาก sin(x)
// วาดกราฟ
plot(x, y);
// เพิ่มชื่อให้แกน X และ Y
xlabel('x');
ylabel('sin(x)');
// เพิ่มชื่อให้กราฟ
title('Graph of sin(x)');
// เปิดใช้งานเส้นตาราง (grid)
xgrid(0);
```



การวาดกราฟใน Scilab

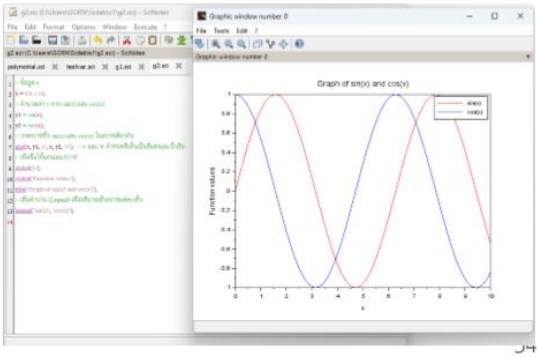
การวาดกราฟใน Scilab เป็นเครื่องมือสำคัญสำหรับการแสดงข้อมูลในรูปแบบภาพ ไม่ว่าจะเป็นการวิเคราะห์ข้อมูล การจำลองระบบ หรือการนำเสนอผลการทดลอง Scilab รองรับการสร้างกราฟทั้ง 2D และ 3D

2. การวาดกราฟหลายเส้นในกราฟเดียว

สามารถวาดกราฟหลายเส้นในกราฟเดียวได้โดยระบุข้อมูล y หลายชุด จาก x ค่าเดียว

ตัวอย่าง

```
// ข้อมูล x
x = 0:0.1:10;
// คำนวณค่า y จาก sin(x) และ cos(x)
y1 = sin(x);
y2 = cos(x);
// วาดกราฟทั้ง sin(x) และ cos(x) ในกราฟเดียวกัน
plot(x, y1, 'r', x, y2, 'b');  // 'r' และ 'b' กำหนดสีเส้นเป็นสีแดงและน้ำเงิน
// เพิ่มชื่อให้แกนและกราฟ
xlabel('x');
ylabel('Function values');
title('Graph of sin(x) and cos(x)');
// เพิ่มตำนาน (Legend) เพื่ออธิบายเส้นกราฟแต่ละเส้น
legend('sin(x)', 'cos(x)');
```



การวาดกราฟใน Scilab

3. การใช้เครื่องหมายและรูปแบบเส้น

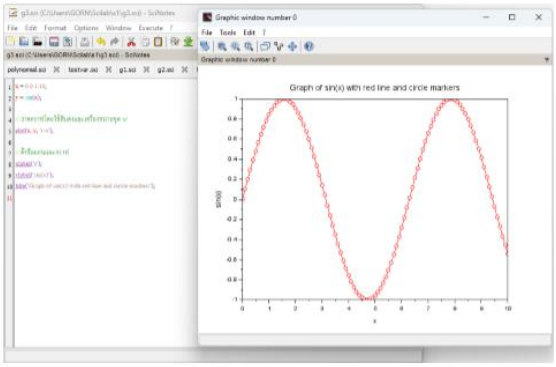
สามารถปรับแต่งกราฟได้โดยการเปลี่ยนสีเส้น รูปแบบเส้น และเครื่องหมายจุดข้อมูล

ตัวอย่าง

```
x = 0:0.1:10;
y = sin(x);

// วาดกราฟโดยใช้สีแดงและเครื่องหมายจุด 'o'
plot(x, y, 'r-o');

// ตั้งชื่อแกนและกราฟ
xlabel('x');
ylabel('sin(x)');
title('Graph of sin(x) with red line and circle markers');
```



การวาดกราฟใน Scilab

4. การวาดกราฟแบบ 3D (3D Plot)

Scilab รองรับการวาดกราฟแบบ 3 มิติ โดยใช้ฟังก์ชัน plot3d() สำหรับการสร้างพื้นผิว หรือ param3d() สำหรับกราฟเส้นสามมิติ

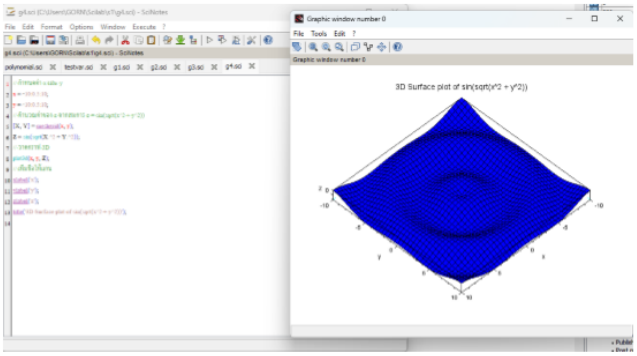
ตัวอย่าง

```
// กำหนดค่า x และ y
x = -10:0.5:10;
y = -10:0.5:10;

// คำนวณค่าของ z จากสมการ z = sin(sqrt(x^2 + y^2))
[X, Y] = meshgrid(x, y);
Z = sin(sqrt(X.^2 + Y.^2));

// วาดกราฟ 3D
plot3d(x, y, Z);

// เพิ่มชื่อให้แกน
xlabel('x');
ylabel('y');
zlabel('z');
title('3D Surface plot of sin(sqrt(x^2 + y^2))');
```



การวาดกราฟใน Scilab

5. การสร้าง Subplots (แบ่งหน้าต่างแสดงกราฟ)

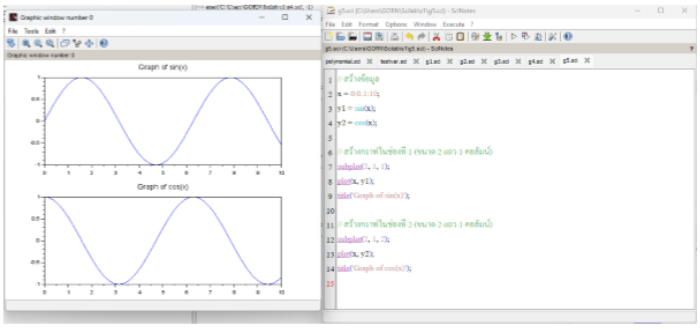
สามารถแสดงหลายกราฟในหน้าต่างเดียวกันโดยใช้ฟังก์ชัน subplot()

ตัวอย่าง

```
// สร้างข้อมูล
x = 0:0.1:10;
y1 = sin(x);
y2 = cos(x);

// สร้างกราฟในช่องที่ 1 (ขนาด 2 แถว 1 คอลัมน์)
subplot(2, 1, 1);
plot(x, y1);
title('Graph of sin(x)');

// สร้างกราฟในช่องที่ 2 (ขนาด 2 แถว 1 คอลัมน์)
subplot(2, 1, 2);
plot(x, y2);
title('Graph of cos(x)');
```



การวาดกราฟใน Scilab

6. การสร้างกราฟแบบกระจาย (Scatter Plot)

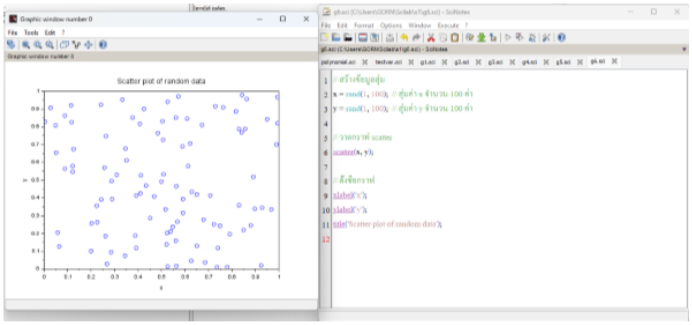
การสร้างกราฟแบบกระจายข้อมูล (Scatter Plot) สามารถใช้ฟังก์ชัน scatter() สำหรับแสดงจุดข้อมูล

ตัวอย่าง

```
// สร้างข้อมูลสุ่ม
x = rand(1, 100);  // สุ่มค่า x จำนวน 100 ค่า
y = rand(1, 100);  // สุ่มค่า y จำนวน 100 ค่า

// วาดกราฟ scatter
scatter(x, y);

// ตั้งชื่อกราฟ
xlabel('x');
ylabel('y');
title('Scatter plot of random data');
```



ตัวอย่าง การประยุกต์โจทย์ทางวิศวกรรมด้วย Scilab

1 การคำนวณค่าพลังงานจลน์

จงเขียนโปรแกรมใน Scilab ที่คำนวณค่าพลังงานจลน์ (Kinetic Energy, KE) ดังสมการพลังงานจลน์

$$KE = \frac{1}{2} mv^2$$

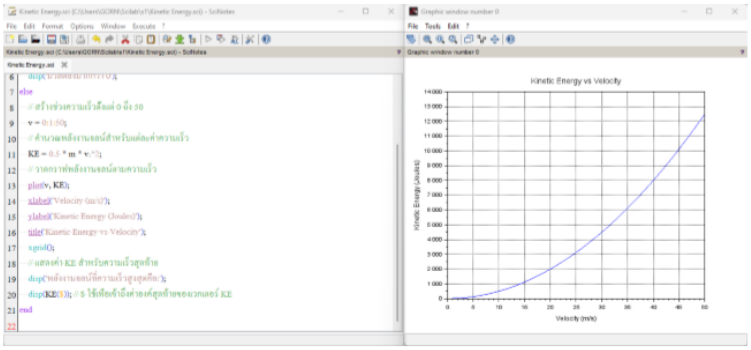
โดยที่ m คือมวลของวัตถุ (kg) และ v คือความเร็วของวัตถุ (เมตรต่อวินาที)

โดยใช้โครงสร้างเงื่อนไขเพื่อตรวจสอบว่ามวลและความเร็วต้องเป็นค่าบวก และวาดกราฟของพลังงานจลน์ที่เปลี่ยนไปตามความเร็วที่เพิ่มขึ้นในช่วง 0 ถึง 50 เมตรต่อวินาที โดยมวลของวัตถุคงที่ที่ 10 กิโลกรัม

ตัวอย่าง การประยุกต์โจทย์ทางวิศวกรรมด้วย Scilab

1 การคำนวณค่าพลังงานจลน์

```
// กำหนดค่า m (มวล) และ v (ความเร็ว)
m = 10;  // กิโลกรัม
// ตรวจสอบว่ามวลต้องเป็นค่าบวก
if m <= 0 then
    disp('มวลต้องมากกว่า 0');
else
    // สร้างช่วงความเร็วตั้งแต่ 0 ถึง 50
    v = 0:1:50;
    // คำนวณพลังงานจลน์สำหรับแต่ละค่าความเร็ว
    KE = 0.5 * m * v.^2;
    // วาดกราฟพลังงานจลน์ตามความเร็ว
    plot(v, KE);
    xlabel('Velocity (m/s)');
    ylabel('Kinetic Energy (Joules)');
    title('Kinetic Energy vs Velocity');
    xgrid();
    // แสดงค่า KE สำหรับความเร็วสุดท้าย
    disp('พลังงานจลน์ที่ความเร็วสูงสุดคือ:');
    disp(KE($)); // $ ใช้เพื่อเข้าถึงค่าองค์สุดท้ายของเวกเตอร์ KE
end
```



ตัวอย่าง การประยุกต์โจทย์ทางวิศวกรรมด้วย Scilab

2 การคำนวณและแสดงผลการเปลี่ยนแปลงแรงที่กระทำต่อวัตถุ

เขียนโปรแกรมที่คำนวณแรงที่กระทำต่อวัตถุโดยใช้สมการ F=m*a

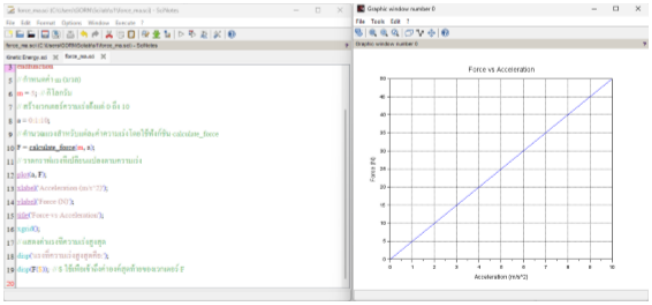
โดยที่ F คือแรง (นิวตัน) m คือมวลของวัตถุ (kg) และ a คือความเร่ง (เมตรต่อวินาที^2)

โดยสร้างฟังก์ชัน calculate_force ที่รับมวลและความเร่งเป็นอินพุต และคืนค่าแรง จากนั้นใช้ฟังก์ชันนี้คำนวณค่าแรงสำหรับวัตถุที่มีมวล 5 กิโลกรัม และความเร่งตั้งแต่ 0 ถึง 10 เมตรต่อวินาที^2 และวาดกราฟแสดงแรงที่เปลี่ยนแปลงตามความเร่ง

ตัวอย่าง การประยุกต์โจทย์ทางวิศวกรรมด้วย Scilab

2 การคำนวณและแสดงผลการเปลี่ยนแปลงแรงที่กระทำต่อวัตถุ

```
// ฟังก์ชันคำนวณแรง
function F = calculate_force(m, a)
    F = m * a;
endfunction
// กำหนดค่า m (มวล)
m = 5;  // กิโลกรัม
// สร้างเวกเตอร์ความเร่งตั้งแต่ 0 ถึง 10
a = 0:1:10;
// คำนวณแรงสำหรับแต่ละค่าความเร่งโดยใช้ฟังก์ชัน calculate_force
F = calculate_force(m, a);
// วาดกราฟแรงที่เปลี่ยนแปลงตามความเร่ง
plot(a, F);
xlabel('Acceleration (m/s^2)');
ylabel('Force (N)');
title('Force vs Acceleration');
xgrid();
// แสดงค่าแรงที่ความเร่งสูงสุด
disp('แรงที่ความเร่งสูงสุดคือ:');
disp(F($));  // $ ใช้เพื่อเข้าถึงค่าองค์สุดท้ายของเวกเตอร์ F
```



เอกสารอ้างอิง

ผศ.นราธร สังข์ประเสริฐ. (2563) การเขียนโปรแกรมด้วยภาษาซี ซีเอ็ดยูเคชั่น ISBN: 978-616-08-3945-2

แหล่งศึกษาเพิ่มเติม

ผศ.ดร.ปิยะ โควินท์ทวีวัฒน์. (2551) คู่มือโปรแกรมภาษา SCILAB สำหรับผู้เริ่มต้น (ฉบับปรับปรุงใหม่), ISBN : 974-620-609-5

ผศ.ดร.ปิยะ โควินท์ทวีวัฒน์. (2552) สัญญาณและระบบ กับการประยุกต์ใช้โปรแกรม SCILAB, ISBN : 978-974-235-415-2

ผศ.นราธร สังข์ประเสริฐ. (2563) การเขียนโปรแกรมด้วยภาษาซี ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-3945-2

Scilab Youtube Channel: https://www.youtube.com/@scilab

แบบฝึกหัดท้ายบท

1. Scilab คืออะไร และมีความสำคัญอย่างไรในงานวิศวกรรม
2. คุณสมบัติของ Scilab ในการคำนวณเชิงตัวเลข (Numerical Computation) คืออะไร
3. อธิบายการใช้งาน Scilab ในการจำลองระบบเชิงพลศาสตร์ เช่น ระบบไฟฟ้าหรือระบบกลไก
4. การสร้างตัวแปรใน Scilab ทำอย่างไร ยกตัวอย่างการสร้างตัวแปรชนิด Scalar และ Matrix
5. อธิบายคำสั่ง if-then-else และยกตัวอย่างการใช้งานใน Scilab
6. ความแตกต่างระหว่างคำสั่ง for และ while loop ใน Scilab คืออะไร
7. อธิบายวิธีการเขียนฟังก์ชันใน Scilab โดยยกตัวอย่างฟังก์ชันที่มีอินพุตและเอาต์พุต
8. คำสั่ง plot() ใน Scilab ใช้ทำอะไร และมีขั้นตอนในการสร้างกราฟ 2D อย่างไร
9. อธิบายการใช้งานฟังก์ชัน plot3d() ในการสร้างกราฟ 3D ใน Scilab
10. การสร้างกราฟแบบกระจายข้อมูล (Scatter Plot) ใน Scilab ทำได้อย่างไร
11. คุณสมบัติของ Scilab ในการประมวลผลภาพ (Image Processing) คืออะไร
12. วิธีการจัดการตัวแปรใน Scilab ผ่าน Variable Browser คืออะไร
13. อธิบายโครงสร้างคำสั่ง select-case ใน Scilab และการใช้งาน
14. วิธีการวาดกราฟหลายเส้นในกราฟเดียวใน Scilab ทำอย่างไร
15. จงเขียนโปรแกรมใน Scilab เพื่อคำนวณค่าพลังงานศักดิ์โน้มถ่วง (Gravitational Potential Energy) ของเครนที่ยกปูน 3 กระสอบ โดยแต่ละกระสอบมีน้ำหนัก 30 กิโลกรัม โดยยกจากพื้นที่ระยะ 2 เมตร จนถึงบนตึก 33 เมตร และวาดกราฟแสดงความสัมพันธ์ระหว่างความสูงกับพลังงานศักดิ์โน้มถ่วงที่เกิดขึ้น โดยใช้สูตรดังนี้ Ep = mgh โดยกำหนดค่า g = 10 m/s^2 M = มวล (kg) และ h = ความสูง

ชื่อ .....รหัสนักศึกษา.....กลุ่มเรียน. .....

ใบฝึกปฏิบัติการที่ 8.1

เรื่อง การคำนวณปริมาตรของถังทรงกระบอกด้วย Scilab

วัตถุประสงค์

1. สามารถแก้โจทย์ปัญหาด้วยการเขียนโปรแกรมใน Scilab ได้
2. สามารถวาดกราฟได้อย่างถูกต้อง

ทบทวนความรู้

การเขียนฟังก์ชันใน Scilab

ตัวอย่างการเขียนฟังก์ชัน

ตัวอย่าง 2 ฟังก์ชันที่มีอินพุตและเอาต์พุต

```
function result = square_number(x)
    result = x^2;
endfunction
```

ฟังก์ชันนี้รับอินพุตเป็นตัวเลข x และคืนค่า x^2 กลับไปเป็นผลลัพธ์

การเรียกใช้

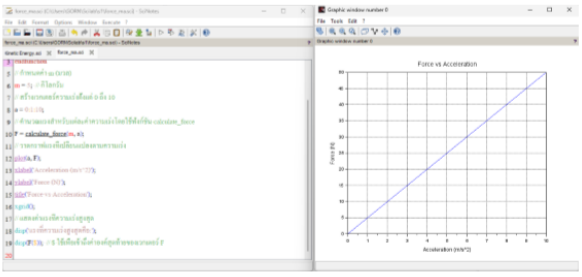
```
y = square_number(4);  // y จะมีค่าเท่ากับ 16
```

Computer Programming 30

ตัวอย่าง การประยุกต์โจทย์ทางวิศวกรรมด้วย Scilab

2 การคำนวณและแสดงผลการเปลี่ยนแปลงแรงที่กระทำต่อวัตถุ

```
// ฟังก์ชันคำนวณแรง
function F = calculate_force(m, a)
    F = m * a;
endfunction
// กำหนดค่า m (มวล)
m = 5;  // กิโลกรัม
// สร้างเวกเตอร์ความเร่งตั้งแต่ 0 ถึง 10
a = 0:1:10;
// คำนวณแรงสำหรับแต่ละค่าความเร่งโดยใช้ฟังก์ชัน calculate_force
F = calculate_force(m, a);
// วาดกราฟแรงที่เปลี่ยนแปลงตามความเร่ง
plot(a, F);
xlabel('Acceleration (m/s^2)');
ylabel('Force (N)');
title('Force vs Acceleration');
xgrid();
// แสดงค่าแรงที่ความเร่งสูงสุด
disp('แรงที่ความเร่งสูงสุดคือ:');
disp(F($));  // $ ใช้เพื่อเข้าถึงค่าองค์สุดท้ายของเวกเตอร์ F
```



Computer Programming 42

คำสั่ง

จงเขียนโปรแกรมที่คำนวณปริมาตรของถังทรงกระบอกโดยใช้สมการ

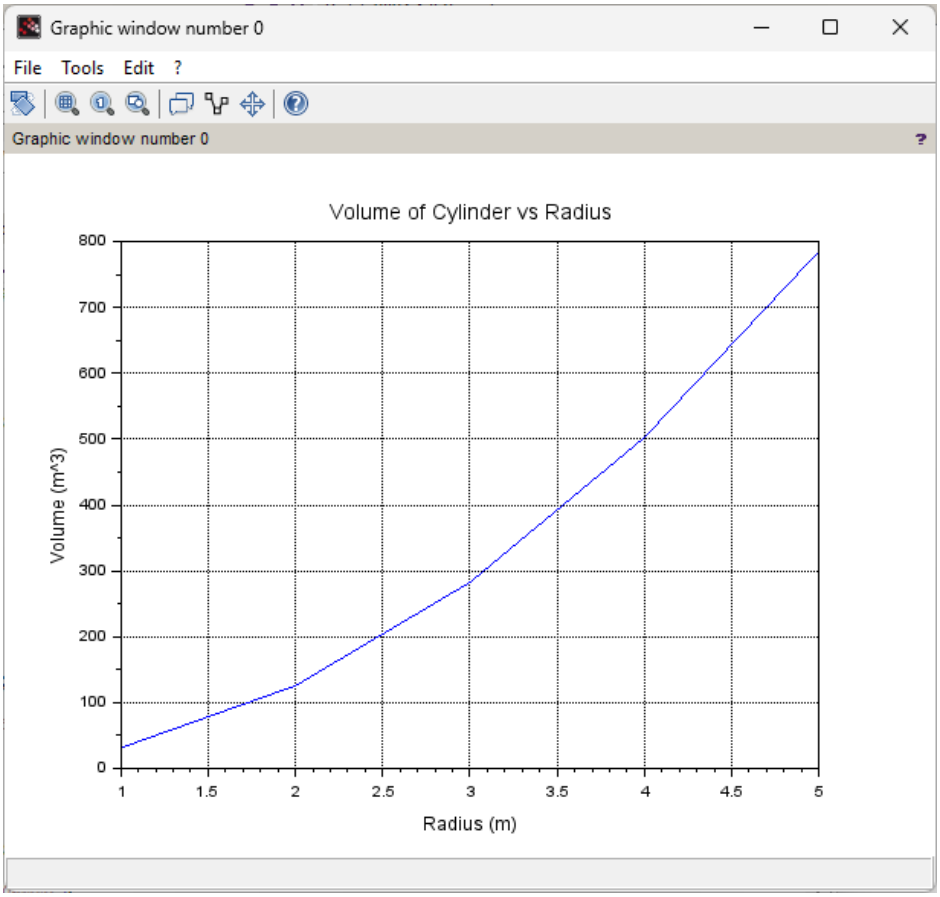
V=**π**r^2h

โดยที่

- V คือปริมาตร (ลูกบาศก์เมตร)
- r คือรัศมีฐานของถัง (เมตร)
- h คือความสูงของถัง (เมตร)

โดยใช้โครงสร้างวนรอบเพื่อคำนวณปริมาตรของถังที่มีรัศมีตั้งแต่ 1 ถึง 5 เมตร และความสูงคงที่ที่ 10 เมตร และวาดกราฟแสดงปริมาตรของถังตามการเปลี่ยนแปลงของรัศมี

ตัวอย่างผลลัพธ์ของกราฟ



ใบฝึกปฏิบัติการที่ 8.2

เรื่อง การวิเคราะห์แรงสปริงด้วย Scilab

วัตถุประสงค์

1. สามารถแก้โจทย์ปัญหาด้วยการเขียนโปรแกรมใน Scilab ได้
2. สามารถวาดกราฟได้อย่างถูกต้อง

ทบทวนความรู้

การเขียนฟังก์ชันใน Scilab

ตัวอย่างการเขียนฟังก์ชัน

ตัวอย่าง 2 ฟังก์ชันที่มีอินพุตและเอาต์พุต

```
function result = square_number(x)
    result = x^2;
endfunction
```

ฟังก์ชันนี้รับอินพุตเป็นตัวเลข x และคืนค่า x^2 กลับไปเป็นผลลัพธ์

การเรียกใช้

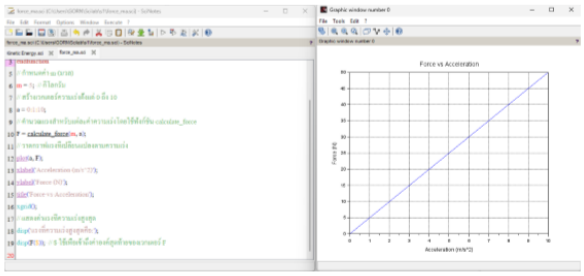
```
y = square_number(4);  // y จะมีค่าเท่ากับ 16
```

Computer Programming 30

ตัวอย่าง การประยุกต์โจทย์ทางวิศวกรรมด้วย Scilab

2 การคำนวณและแสดงผลการเปลี่ยนแปลงแรงที่กระทำต่อวัตถุ

```
// ฟังก์ชันคำนวณแรง
function F = calculate_force(m, a)
    F = m * a;
endfunction
// กำหนดค่า m (มวล)
m = 5;  // กิโลกรัม
// สร้างเวกเตอร์ความเร่งตั้งแต่ 0 ถึง 10
a = 0:1:10;
// คำนวณแรงสำหรับแต่ละค่าความเร่งโดยใช้ฟังก์ชัน calculate_force
F = calculate_force(m, a);
// วาดกราฟแรงที่เปลี่ยนแปลงตามความเร่ง
plot(a, F);
xlabel('Acceleration (m/s^2)');
ylabel('Force (N)');
title('Force vs Acceleration');
xgrid();
// แสดงค่าแรงที่ความเร่งสูงสุด
disp('แรงที่ความเร่งสูงสุดคือ:');
disp(F($));  // $ ใช้เพื่อเข้าถึงค่าองค์สุดท้ายของเวกเตอร์ F
```



Computer Programming 42

คำสั่ง

จงเขียนโปรแกรมที่คำนวณแรงที่เกิดจากการยืดหรือหดสปริงโดยใช้กฎของฮุก (Hooke's Law) ดังสมการ $F=-kx$

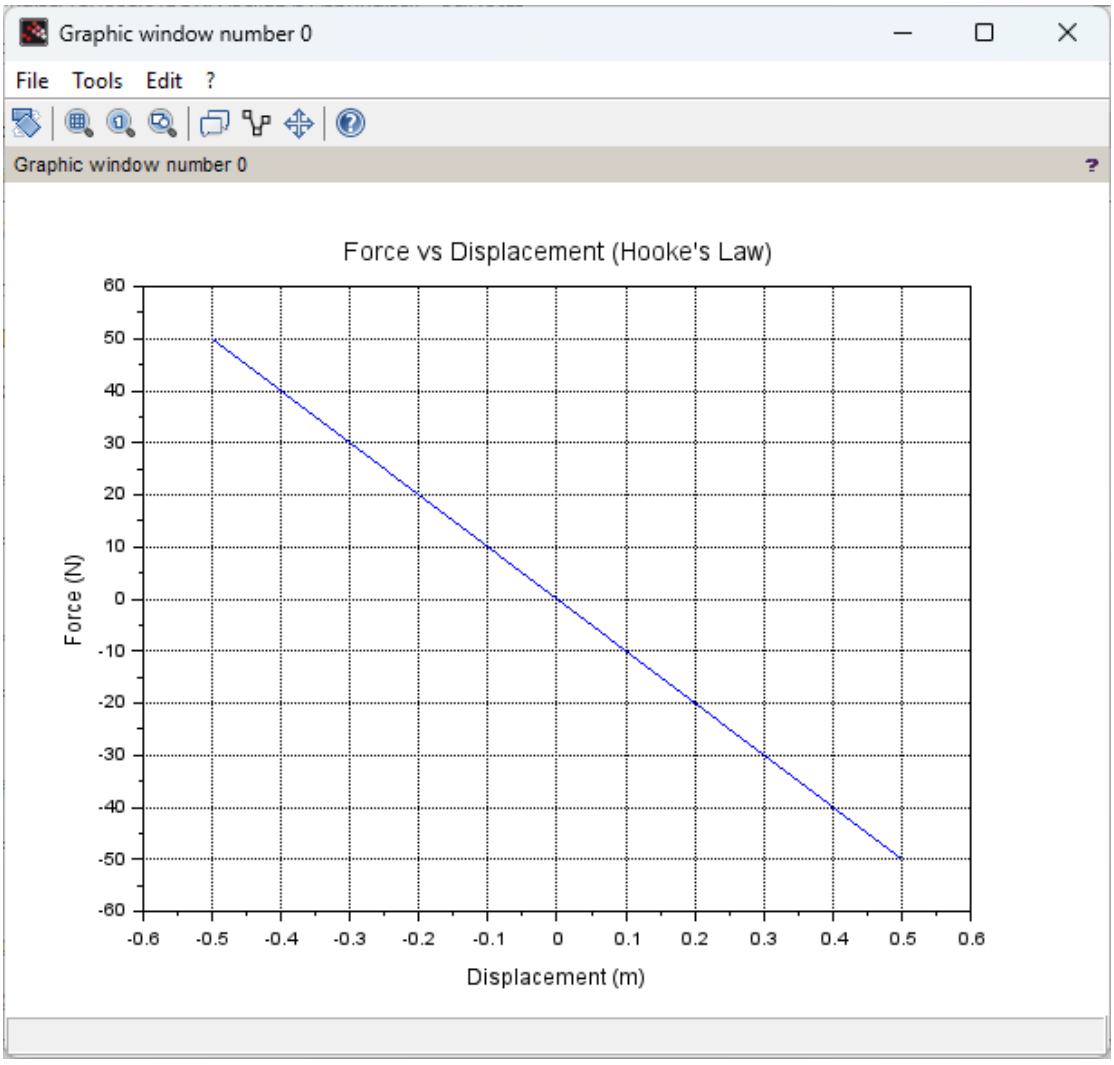
โดยที่ F คือแรง (นิวตัน)

k คือค่าคงที่ของสปริง (นิวตันต่อเมตร)

x คือการยืดหรือหดของสปริง (เมตร)

- สร้างฟังก์ชัน **hookes_law** ที่รับค่าคงที่ของสปริงและการยืด/หดของสปริงเป็นอินพุต และคืนค่าแรง
- คำนวณแรงที่เกิดจากการยืดหรือหดของสปริงที่มีค่าคงที่ 100 นิวตัน/เมตร และการยืด/หดตั้งแต่ -0.5 ถึง 0.5 เมตร
- วาดกราฟแสดงแรงตามการยืดหรือหดของสปริง

ตัวอย่างผลลัพธ์ของกราฟ



บรรณานุกรม

รศ.ดร.จันทนา จันทราพรชัย. (2555) สถาปัตยกรรมคอมพิวเตอร์และการออกแบบ (COMPUTER ARCHITECTURE AND DESING), สำนักพิมพ์ท้อป, ISBN: 978-616-282-009-0

รศ.ธีรวัฒน์ ประกอบผล. (2551) สถาปัตยกรรมคอมพิวเตอร์, สมาคมส่งเสริมเทคโนโลยี, ISBN: 978-974-443-369-5

ผศ.นราธร สังข์ประเสริฐ. (2563) การเขียนโปรแกรมด้วยภาษาซี ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-3945-2

โอภาส เอี่ยมสิริวงศ์. (2559) การเขียนโปรแกรมด้วยภาษาซี, ซีเอ็ดยูเคชั่น, ISBN: 978-616-08-2740-4

รศ.ธีรวัฒน์ ประกอบผล. (2562) คู่มือการเขียนโปรแกรมภาษา C ฉบับสมบูรณ์, ซิมพลิฟาย, ISBN: 978-616-26-24896

นราธร สังข์ประเสริฐ, เอกสารประกอบการบรรยายวิชา 04-511-101 การโปรแกรมคอมพิวเตอร์, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย , 2558

Sites.google.com, "รหัสเทียม Psuedo Code - Programming", 2016. [Online]. Available: https://sites.google.com/site/programmingm42/-rhas-theiym-psuedo-code. [Accessed: 17- Jan- 2016].

Sites.google.com, "การแก้ปัญหา - Programming", 2016. [Online]. Available: https://sites.google.com/site/programmingm42/kar-kae-payha. [Accessed: 17- Jan- 2016].

คณาจารย์ผู้สอนวิชา Introduction to Computer Programming, เอกสารประกอบการบรรยายวิชา 241-101 Introduction to Computer Programming, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยสงขลานครินทร์, 2551

อ.สุธน แซ่ว่อง, เอกสารประกอบการบรรยายอบรมภาษาซีเบื้องต้น, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยสงขลานครินทร์, 30 มีนาคม 2552

Y. D. Liang, Supplement II.C:Dev-C++ Tutorial for Introduction to C++ Programming, http://www.bloodshed.net/dev/devcpp.html

บรรณานุกรม (ต่อ)

ดร. แสงสุรีย์ วสุพงศ์อัยยะ, เอกสารประกอบการบรรยายอบรมภาษาซีเบื้องต้น, คณะวิศวกรรมศาสตร์, มหาวิทยาลัยสงขลานครินทร์, 30 มีนาคม 2552