

การเขียนโปรแกรม

ภาษา C

ขั้นพื้นฐาน

ครอบคลุมเนื้อหาสำคัญ
ทั้งหมด พร้อมตัวอย่างโค้ด
จำนวนมาก



เขียนโค้ดภาษา C จากเริ่มต้น
จนถึงระดับการใช้งานได้จริง



อธิบายเป็นขั้นตอน มีภาพประกอบ
และสามารถเรียนรู้ได้ด้วยตนเอง



ดาวน์โหลดโค้ดหนังสือเล่มนี้ได้ที่
<http://www.developerthai.com>

บัญชา ปะสีละเตสัง

ผู้เขียนหนังสือขายดีระดับ Best Seller
ด้าน Programming

การเขียนโปรแกรมภาษา C ขั้นพื้นฐาน

โดย บัญชา ปะสีละเตลัง

สงวนลิขสิทธิ์ตามกฎหมาย โดย บัญชา ปะสีละเตลัง © พ.ศ. 2566

ห้ามคัดลอก ลอกเลียน ดัดแปลง ทำซ้ำ จัดพิมพ์ หรือกระทำการอื่นใด โดยวิธีการใดๆ ในรูปแบบใดๆ
ไม่ว่าส่วนหนึ่งส่วนใดของหนังสือเล่มนี้ เพื่อเผยแพร่ในสื่อทุกประเภท หรือเพื่อวัตถุประสงค์ใดๆ
นอกจากจะได้รับอนุญาต

ข้อมูลทางบรรณานุกรมของหอสมุดแห่งชาติ

บัญชา ปะสีละเตลัง.

การเขียนโปรแกรมภาษา C ขั้นพื้นฐาน.-- กรุงเทพฯ : ซีเอ็ดยูเคชั่น, 2566.

392 หน้า.

1. ซี (ภาษาคอมพิวเตอร์).

I. ชื่อเรื่อง.

005.133

Barcode (e-book) : 9786160847518

SE-ED

inspiration starts here

ผลิตและจัดจำหน่ายโดย



บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)
SE-EDUCATION PUBLIC COMPANY LIMITED

เลขที่ 1858/87-90 ถนนเทพรัตน แขวงบางนาใต้ เขตบางนา กรุงเทพฯ 10260

โทรศัพท์ 0-2826-8000

[หากมีคำแนะนำหรือติชม ติดต่อที่ comment@se-ed.com]

คำนำ

ภาษา C เป็นหนึ่งในภาษาคอมพิวเตอร์ที่มีผู้ใช้งานเป็นจำนวนมากในปัจจุบัน เนื่องจากลักษณะโครงสร้างที่ไม่ซับซ้อนจนเกินไป จึงเรียนรู้ได้ไม่ยาก รวมทั้งความเร็วในการประมวลผลและความสามารถที่โดดเด่นในอีกหลายๆ ด้าน ดังนั้น สถาบันการศึกษาส่วนใหญ่จึงมักเลือกภาษา C ในการเรียนการสอนสำหรับวิชาการเขียนโปรแกรมระดับเริ่มต้น และแม้ว่าภาษา C จะถูกสร้างมานานมากแล้วก็ตาม แต่ซอฟต์แวร์หลักๆ ที่เราใช้งานกันอยู่ในทุกวันนี้ ล้วนถูกสร้างจากพื้นฐานของภาษา C แทบทั้งสิ้น เช่น ระบบปฏิบัติการต่างๆ โปรแกรมด้านฐานข้อมูลและธุรกิจอุตสาหกรรม การคำนวณทางวิทยาศาสตร์ Game Engine และอื่นๆ อีกมากมาย ดังนั้น การเขียนโปรแกรมด้วยภาษา C จึงเป็นสิ่งสำคัญอีกอย่างหนึ่งที่ผู้ต้องการก้าวสู่สายงานด้านการพัฒนาซอฟต์แวร์ระดับมืออาชีพควรต้องเรียนรู้เอาไว้

หนังสือเล่มนี้ได้รวบรวมเนื้อหาที่จำเป็นต้องรู้ในขั้นพื้นฐานทั้งหมด พร้อมตัวอย่างประกอบ เพื่อเสริมความเข้าใจอีกเป็นจำนวนมาก โดยใช้เครื่องมือยอดนิยมอย่าง Visual Studio Code (VS Code) ร่วมกับตัวแปลภาษา MinGW ซึ่งผู้เริ่มต้นศึกษาการเขียนโปรแกรม สามารถเรียนรู้ด้วยตนเอง รวมถึงเป็นแนวทางสำหรับศึกษาเพิ่มเติมจากแหล่งข้อมูลอื่นๆ ต่อไปได้

บัญชา ปะลีละเตลัง

banchar_pa@yahoo.com

facebook.com/DeveloperThai



```
... = modifier_ob.modifiers.new("mirror",  
... object to mirror_ob  
..._mod.mirror_object = mirror_ob  
... == "MIRROR_X":  
..._mod.use_x = True  
..._mod.use_y = False  
..._mod.use_z = False  
... == "MIRROR_Y":  
..._mod.use_x = False  
..._mod.use_y = True  
..._mod.use_z = False  
... == "MIRROR_Z":  
..._mod.use_x = False  
..._mod.use_y = False  
..._mod.use_z = True  
... at the end -add back the deselected  
... select 1  
... select 1  
... context.scene.objects.active = modifier_ob  
... " + str(modifier_ob)) # modifier  
... context.selected_objects[0]  
... objects[one.name].select = 1  
... ("please select exactly two objects, the  
... OPERATOR CLASSES
```

```
... Operator):  
... mirror to the selected object""  
... _mod.mirror_mirror_x"  
... mirror X"  
... object is not None
```

SE-ED

inspiration starts here

สารบัญ

บทที่ 1	การจัดเตรียมเครื่องมือ.....	13
	การติดตั้ง VS Code.....	13
	การติดตั้ง MinGW	15
	การเซตตำแหน่งของตัวแปลภาษา.....	17
	การติดตั้ง C/C++ Package สำหรับ VS Code.....	18
	การติดตั้ง Code Runner สำหรับ VS Code.....	19
	การทดสอบโค้ดภาษา C	20
	• สร้างโฟลเดอร์สำหรับเก็บโค้ด.....	20
	• การสร้างไฟล์และทดสอบโค้ดภาษา C	22
	การใช้เครื่องมือพื้นฐานของ VS Code.....	25
	• หน้าจอเริ่มต้น (Get Started)	25
	• การเลือกรูปแบบสี (Theme).....	26
	• การปรับเปลี่ยนฟอนต์.....	27
	• แนวทางการติดตั้งส่วนเสริมเพิ่มเติม	28
	การนำโค้ดของหนังสือมาใช้งาน	30
บทที่ 2	การเขียนโค้ดภาษา C ในเบื้องต้น	31
	ประวัติโดยย่อของภาษา C.....	31
	ข้อดีและข้อเสียของภาษา C.....	32
	คำสวณของภาษา C.....	34
	บทวนการสร้างไฟล์ภาษา C	35

องค์ประกอบของการเขียนโค้ดภาษา C ในเบื้องต้น.....	35
• คำสั่ง #include.....	35
• ฟังก์ชัน main	37
• การกำหนดบล็อกคำสั่งด้วยวงเล็บ {}	38
• การกำหนดจุดสิ้นสุดคำสั่งด้วยเครื่องหมาย ;.....	39
• การเขียนคำอธิบายโค้ด (Comment)	40
การเขียนคำอธิบายแบบบล็อก (Block Comment).....	40
การเขียนคำอธิบายแบบบรรทัด (Line Comment).....	40
การแสดงข้อความในเบื้องต้น.....	41
• การแสดงข้อความด้วยฟังก์ชัน puts().....	41
• การแสดงข้อความด้วยฟังก์ชัน printf()	42
• อักขระพิเศษสำหรับขึ้นบรรทัดใหม่ (\n).....	44

บทที่ 3 ชนิดข้อมูลและตัวแปร..... 47

ชนิดข้อมูลในภาษา C.....	47
• ข้อมูลประเภทเลขจำนวนเต็ม	48
• ข้อมูลประเภทเลขทศนิยม.....	49
• ข้อมูลประเภทอักขระ.....	49
ลักษณะของตัวแปรในภาษา C.....	51
การประกาศตัวแปร	52
การกำหนดค่าให้กับตัวแปร	53
พื้นฐานการแปลงชนิดข้อมูล.....	55
• Implicit Conversion	56
• Explicit Conversion	56
ค่าคงที่.....	57
• การกำหนดค่าคงที่ด้วยคำสั่ง const.....	58
• การกำหนดค่าคงที่ด้วยคำสั่ง define.....	58

บทที่ 4 สตริง การรับและแสดงผลข้อมูล 61

อักขระและสตริง.....	61
การกำหนดและแก้ไขค่าของตัวแปรสตริง.....	65

• การใช้ฟังก์ชัน strcpy()	65
• การใช้ฟังก์ชัน strcat()	67
อักขระพิเศษ (Escape Sequence)	68
การจัดรูปแบบสตริงสำหรับแสดงผล	69
• การจัดรูปแบบในเบื้องต้นสำหรับ printf()	70
• การกำหนดความกว้างของสตริง	73
• การอ่านค่าสตริงที่จัดรูปแบบด้วย sprintf()	76
• การจัดรูปแบบตัวเลขโดยมี , คั่นหลักพัน	77
การรับข้อมูลทางคีย์บอร์ด	78
• การรับข้อมูลชนิดอักขระด้วย getchar()	79
• การรับข้อมูลแบบสตริงด้วย gets()	80
• การรับข้อมูลหลายชนิดด้วย scanf()	81
การแปลงชนิดข้อมูลระหว่างสตริงและตัวเลข	86
• การแปลงสตริงเป็นตัวเลขด้วยฟังก์ชัน atoi()	86
• การแปลงตัวเลขเป็นสตริงด้วยฟังก์ชัน sprintf()	88

บทที่ 5 ตัวเลขและการคำนวณ 91

เครื่องหมายสำหรับการกำหนดค่า	91
เครื่องหมายสำหรับการคำนวณทางคณิตศาสตร์	93
เครื่องหมายสำหรับการคำนวณและกำหนดค่า	99
เครื่องหมายสำหรับเพิ่มและลดค่า	103
ลำดับการประมวลผลของเครื่องหมาย	105
ฟังก์ชันทางด้านคณิตศาสตร์	107
การสร้างเลขสุ่ม	109
การแปลงสูตรคณิตศาสตร์เป็นโค้ดภาษา C	114

บทที่ 6 การเปรียบเทียบและกำหนดเงื่อนไข 119

ข้อมูลชนิดบูลีน (_Bool และ bool)	119
เครื่องหมายสำหรับการเปรียบเทียบ	121
การเปรียบเทียบสตริง	124
ลักษณะพื้นฐานของคำสั่ง if	124

การกำหนดเงื่อนไขด้วยเครื่องหมายเปรียบเทียบ.....	127
การใช้คำสั่ง if-else.....	131
การใช้คำสั่ง if-<else if>	133
การใช้คำสั่ง if-<else if>-else	137
การเปรียบเทียบทางตรรกะ	139
การกำหนดหลายเงื่อนไขด้วย Logical Operator	140
ตัวดำเนินการแบบ Ternary.....	145
การใช้คำสั่ง switch-case.....	146
การตรวจสอบอักขระด้วยฟังก์ชันที่คืนค่าแบบบูลีน	151

บทที่ 7 การทำซ้ำแบบวนรอบ..... 153

การใช้รูปแบบ for ขั้นพื้นฐาน.....	153
การกำหนดลูป for ซ้อนกัน	157
การใช้คำสั่ง break.....	160
การใช้คำสั่ง continue.....	161
การกำหนดลูป for แบบไม่รู้จบ (Infinite Loop).....	161
ลักษณะที่ควรรู้เพิ่มเติมเกี่ยวกับลูป for	162
การใช้รูปแบบ while.....	165
การใช้รูปแบบ do-while.....	169

บทที่ 8 รวมตัวอย่างโค้ดเพิ่มเติม ชุดที่ 1 175

บทที่ 9 การสร้างและใช้งานฟังก์ชัน 201

ลักษณะของฟังก์ชัน.....	201
พารามิเตอร์และอาร์กิวเมนต์	203
ฟังก์ชันแบบไม่ส่งค่ากลับ.....	204
• การสร้างฟังก์ชันแบบไม่ส่งค่ากลับ	205
• ฟังก์ชัน main() และตำแหน่งการเขียนฟังก์ชันที่สร้างเอง.....	206
• การเรียกใช้ฟังก์ชันแบบไม่ส่งค่ากลับ.....	207
ฟังก์ชันแบบส่งค่ากลับ	210
• การสร้างฟังก์ชันแบบส่งค่ากลับ.....	210
• การเรียกฟังก์ชันแบบส่งค่ากลับ.....	212

ตัวแปรแบบ global และ local.....	216
ตัวแปรแบบ static	218
การเรียกฟังก์ชันแบบ Recursion.....	219

บทที่ 10 อาร์เรย์และสตริง (เพิ่มเติม)..... 225

ลักษณะพื้นฐานของอาร์เรย์.....	225
การสร้างและกำหนดค่าสมาชิกของอาร์เรย์.....	227
การเข้าถึงสมาชิกของอาร์เรย์	230
• การอ่านค่าสมาชิกของอาร์เรย์.....	230
• การหาขนาดของอาร์เรย์	234
การใช้อาร์เรย์เป็นพารามิเตอร์ของฟังก์ชัน	238
อาร์เรย์แบบ 2 มิติ.....	240
• ลักษณะของอาร์เรย์ 2 มิติ.....	240
• การหาขนาดของอาร์เรย์ 2 มิติ	244
• การเข้าถึงสมาชิกในอาร์เรย์โดยใช้ลูป	244
อาร์เรย์ของสตริง	245

บทที่ 11 การชี้ตำแหน่งด้วยพอยน์เตอร์..... 253

หลักการเบื้องต้นของพอยน์เตอร์.....	253
การประกาศตัวแปรประเภทพอยน์เตอร์.....	255
การกำหนดตำแหน่งให้พอยน์เตอร์ชี้ไป.....	256
การอ่านและกำหนดค่าของตำแหน่งที่พอยน์เตอร์ชี้ไป	258
การใช้พอยน์เตอร์ร่วมกับฟังก์ชัน	260
• การรับพารามิเตอร์ในแบบพอยน์เตอร์.....	260
• การส่งค่าแบบพอยน์เตอร์กลับจากฟังก์ชัน.....	262
การใช้พอยน์เตอร์ร่วมกับอาร์เรย์.....	267
• การชี้ตำแหน่งของสมาชิกด้วยพอยน์เตอร์.....	267
• การเลื่อนตำแหน่งของพอยน์เตอร์.....	268
• การใช้พอยน์เตอร์ร่วมกับเครื่องหมาย ++ และ --	269
การใช้พอยน์เตอร์กับสตริง	272
• การใช้พอยน์เตอร์แทนตัวแปรแบบสตริง	273
• การใช้พอยน์เตอร์ร่วมกับอาร์เรย์ของสตริงในแบบ 2 มิติ.....	274

บทที่ 12 รวมตัวอย่างโค้ดเพิ่มเติม ชุดที่ 2 279**บทที่ 13 สตรักเจอร์ ยูเนียน และเอนัม 313**

โครงสร้างข้อมูลแบบสตรักเจอร์ (Structure).....	313
การกำหนดและอ่านข้อมูลจากสตรักเจอร์	315
การสร้างอาร์เรย์ของสตรักเจอร์	320
การใช้พอยน์เตอร์ร่วมกับสตรักเจอร์.....	324
การใช้สตรักเจอร์ร่วมกับฟังก์ชัน.....	324
• การใช้สตรักเจอร์เพื่อเป็นพารามิเตอร์	325
• การส่งค่าแบบสตรักเจอร์กลับจากฟังก์ชัน	325
การสร้างสตรักเจอร์ซ้อนกัน.....	327
โครงสร้างข้อมูลแบบยูเนียน (Union)	329
ชุดข้อมูลแบบเอนัม (Enum)	332

บทที่ 14 การเขียนและอ่านไฟล์ 335

การเปิดและปิดไฟล์.....	335
• การระบุตำแหน่งไฟล์.....	335
• โหมดในการเปิดไฟล์.....	336
• การเปิดไฟล์.....	337
• การปิดไฟล์.....	339
การเขียนไฟล์.....	339
• ฟังก์ชัน fprintf().....	340
• ฟังก์ชัน fputs()	342
• ฟังก์ชัน fputc()	342
การอ่านไฟล์.....	345
• ฟังก์ชัน fgets().....	345
• ฟังก์ชัน fgetc()	348
• การเลื่อนตำแหน่งของพอยน์เตอร์.....	350
• ฟังก์ชัน fscanf()	351
เก็บข้อมูลสตรักเจอร์ด้วยไฟล์ไบนารี.....	355
• โหมดในการเปิดไฟล์ไบนารี.....	355
• การจัดแบ่งข้อมูลแบบ Record.....	356

• การเขียนข้อมูลสตรักเจอร์ลงในไฟล์ไบนารี	356
• การอ่านข้อมูลสตรักเจอร์จากไฟล์ไบนารี	358
• การเข้าถึงข้อมูลแบบเจาะจง	361

บทที่ 15 ฟรีโปรแกรมเมอร์และมาโคร 365

เกี่ยวกับฟรีโปรแกรมเมอร์และโดเร็กทีฟ	365
กลุ่มโดเร็กทีฟ Macro Definition	366
ค่าคงที่ในกลุ่ม Predefined Macro	370
กลุ่มโดเร็กทีฟ Conditional Compilation	371

บทที่ 16 รวมตัวอย่างโค้ดเพิ่มเติม ชุดที่ 3 373



SE-ED
inspiration starts here



```
... = modifier_ob.modifiers.new("Mirror",  
... object to mirror_ob  
..._mod.mirror_object = mirror_ob  
... == "MIRROR_X":  
..._mod.use_x = True  
..._mod.use_y = False  
..._mod.use_z = False  
... == "MIRROR_Y":  
..._mod.use_x = False  
..._mod.use_y = True  
..._mod.use_z = False  
... == "MIRROR_Z":  
..._mod.use_x = False  
..._mod.use_y = False  
..._mod.use_z = True
```

```
... at the end -add back the deselected  
... select 1  
..._b.select 1  
...context.scene.objects.active = modifier_ob  
..._b["edit" + str(modifier_ob)] # modifier  
..._b.context.selected_objects[0]  
..._b.objects[one.name].select = 1  
...("please select exactly two objects, the  
... OPERATOR CLASSES -----
```

```
..._b.Operator):  
... on & mirror to the selected object""  
..._b.mirror_mirror_x"  
... mirror X"  
... object is not None
```

SE-ED

inspiration starts here

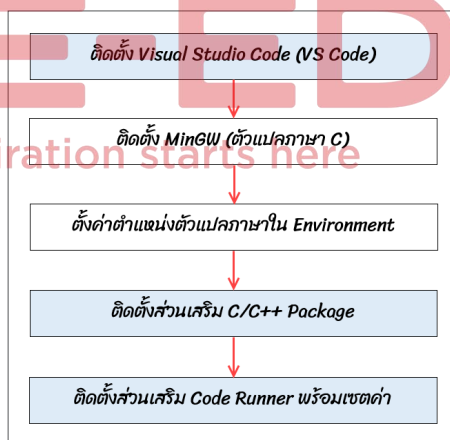
1

การจัดเตรียมเครื่องมือ

ก่อนที่จะเราจะเข้าสู่ขั้นตอนการเขียนโค้ดภาษา C ตามที่จะกล่าวถึงในบทต่อไป สิ่งที่เราจำเป็นต้องจัดเตรียมเอาไว้ล่วงหน้าก็คือ เครื่องมือในการเขียนโค้ดและทดสอบเพื่อผลลัพธ์ ซึ่งแม้จะมีตัวเลือกอยู่มากมาย แต่ในหนังสือเล่มนี้ จะเลือกมานำเสนอเฉพาะเครื่องมือที่ได้รับความนิยมสูงสุดในปัจจุบัน โดยเครื่องมือที่เลือกนี้ ไม่ใช่เฉพาะกับการเขียนโค้ดภาษา C เท่านั้น แต่ยังสามารถนำไปใช้สำหรับการเขียนโค้ดภาษาคอมพิวเตอร์อื่นๆ ได้เกือบทั้งหมด จึงถือเป็นข้อได้เปรียบอย่างหนึ่งเมื่อเราศึกษาเรียนรู้การเขียนโปรแกรมภาษาอื่นๆ เพิ่มเติมในอนาคต

การติดตั้ง VS Code

การเขียนโค้ดภาษา C นั้น แม้เราจะมีเครื่องมือให้เลือกใช้มากมาย แต่ในที่นี้ผู้เขียนจะแนะนำวิธีการที่ได้รับความนิยมสูงสุดในปัจจุบันเพียงวิธีเดียว นั่นก็คือ การใช้งานผ่าน Visual Studio Code (เรานิยมเรียกสั้นๆ ว่า VS Code) อย่างไรก็ตาม เนื่องจาก VC Code ไม่ได้ถูกสร้างมาสำหรับการเขียนโค้ดภาษา C โดยตรง แต่สามารถใช้กับภาษาคอมพิวเตอร์ต่างๆ ได้เกือบทั้งหมด ดังนั้น เรายังต้องติดตั้งเครื่องมืออื่นๆ เพิ่มเติมอีก โดยขั้นตอนที่จำเป็นต้องทำสามารถสรุปได้ดังนี้

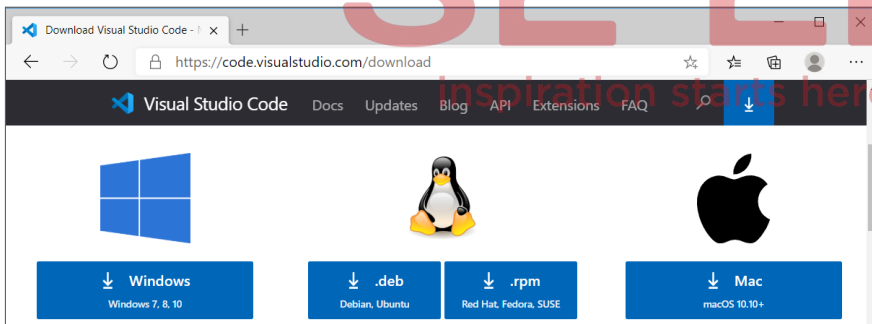


1. ติดตั้ง **Visual Studio Code (VS Code)** เพื่อใช้เป็นเครื่องมือหลักในการเขียนและแสดงผลพร้อมทั้งจัดการไฟล์ของโค้ด
2. ติดตั้ง **MinGW** เพื่อเป็นตัวแปลภาษา (Compiler) ของภาษา C
3. เซตตำแหน่งที่ติดตั้ง MinGW ลงใน **Environment Variables** ของระบบ เพื่อให้โปรแกรมอื่นๆ รู้ตำแหน่งของตัวแปลภาษา
4. ติดตั้งส่วนเสริม **C/C++ Package** ลงใน VS Code เพื่อเป็นตัวช่วยในการเขียนโค้ดภาษา C
5. ติดตั้งส่วนเสริม **Code Runner** ลงใน VS Code เพื่อเชื่อมโยงระหว่าง VS Code กับ MinGW ซึ่งจะช่วยให้เราสามารถรันทดสอบโค้ดได้ง่ายขึ้น รวมทั้งตั้งค่าการแสดงผลผลลัพธ์อีกด้วย

ทั้งหมดที่กล่าวมา จะแยกกล่าวรายละเอียดเป็นหัวข้อต่างๆ ในลำดับต่อไป แต่ในหัวข้อนี้จะกล่าวถึงขั้นตอนแรกนั่นก็คือ การติดตั้ง VS Code ซึ่งเป็นโปรแกรมประเภท IDE (Integrated Development Environment) ที่ Microsoft สร้างขึ้นมาสำหรับใช้ในการเขียนโค้ดทั่วไป และได้รับความนิยมสูงสุดในปัจจุบัน เนื่องจากมีลักษณะที่โดดเด่นหลายประการ เช่น ให้ใช้งานฟรี รองรับการเขียนโค้ดได้หลายภาษา และที่สำคัญคือ มีส่วนเสริมการทำงานให้เลือกใช้มากมาย

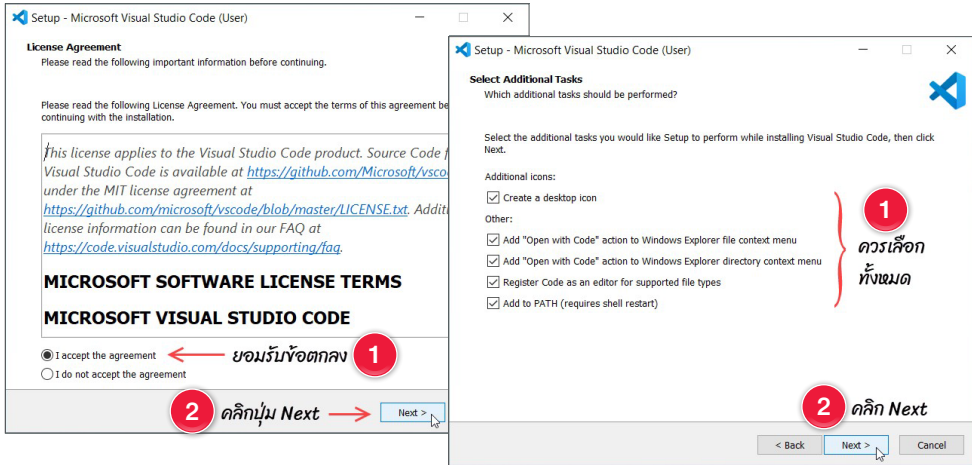
สำหรับการติดตั้ง VS Code ลงในเครื่องที่เราจะใช้งาน มีขั้นตอนโดยสังเขปคือ

1. สามารถดาวน์โหลดชุดติดตั้งได้ที่ <https://code.visualstudio.com/download>

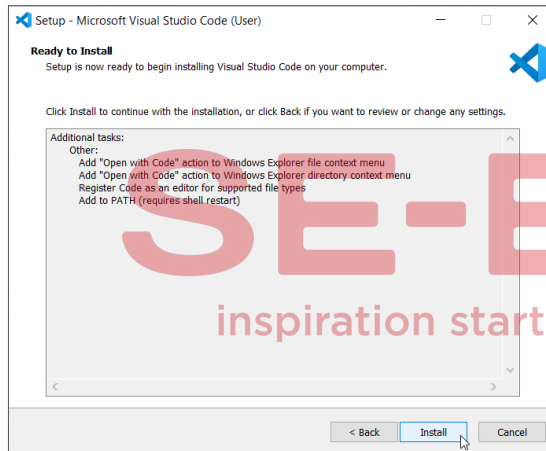


2. เลือกดาวน์โหลดให้ตรงกับระบบปฏิบัติการที่เราจะใช้งาน (ในที่นี้จะกล่าวถึงเฉพาะระบบ Windows เท่านั้น)
3. หลังจากดาวน์โหลดเสร็จ ให้ดับเบิลคลิกไฟล์ติดตั้งที่ได้มา และในหน้าจอถัดไป ก็ยอมรับเงื่อนไขแล้วคลิกปุ่ม Next

4. ขั้นตอนต่อไป จะมีย่อปชั่นให้เลือก ก็แนะนำให้เลือกทั้งหมด แล้วคลิกปุ่ม Next



5. ต่อจากนั้น คลิกปุ่ม Install เพื่อเริ่มการติดตั้งไฟล์ จากนั้นก็รอจนกว่าจะแล้วเสร็จ



การติดตั้ง MinGW

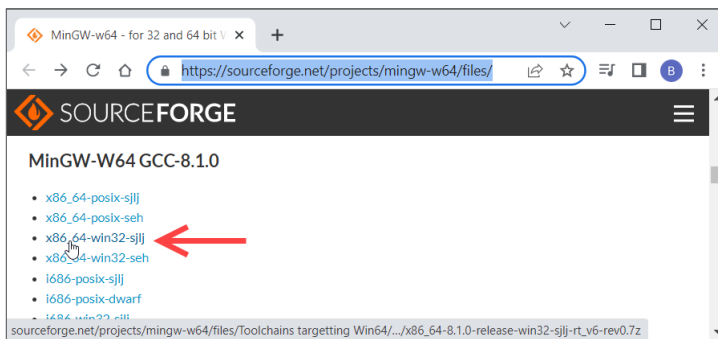
การที่ภาษาคอมพิวเตอร์จะทำงานได้นั้น ต้องอาศัยตัวแปลภาษา (Compiler) ซึ่งสิ่งนี้ ไม่ได้ถูกติดตั้งมาพร้อมกับ VS Code โดยเราต้องติดตั้งเพิ่มเติมลงไปต่างหาก ซึ่งตัวแปลภาษา C มีผู้สร้างให้เลือกใช้มากมาย สำหรับในที่นี้จะใช้ตัวแปลภาษาของ GCC (GNU Compiler Collection) ที่ได้รับความนิยมสูงสุด แต่จะใช้ในเวอร์ชัน MinGW (Minimalist GCC for Windows) ซึ่งเป็นการลดขนาดของตัวแปลภาษาลงสำหรับใช้งานบนระบบ Windows โดยตรง ทั้งนี้ MinGW มีทั้ง

แบบติดตั้งด้วยไฟล์ Installer และแบบคัดลอกไฟล์มาไว้ในเครื่องก็ใช้ได้เลย โดยแบบ Installer นั้นจะมีขั้นตอนการติดตั้งปลั๊กย่อยซึ่งค่อนข้างยุ่งยากเล็กน้อย ในที่นี้ผู้เขียนจึงจะกล่าวถึงเฉพาะแบบคัดลอกไฟล์ เพราะสะดวก รวดเร็วกว่า โดยมีแนวทางการนำมาใช้งานดังนี้

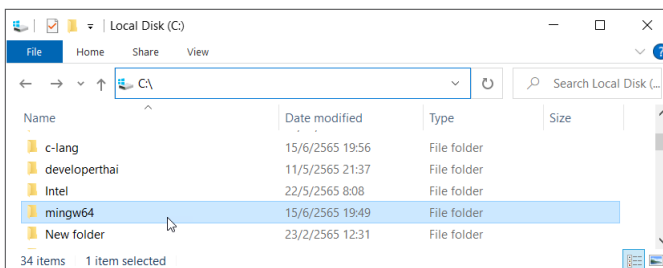
1. สามารถดาวน์โหลด MinGW ได้ที่

<https://sourceforge.net/projects/mingw-w64/files/>

แล้วเลื่อนลงมาคลิกที่ลิงก์ **x86_64-win32-sjlj** ดังภาพ (ชื่อไฟล์อาจถูกเปลี่ยนแปลงไปเรื่อยๆ แต่ควรเลือกไฟล์ชื่อขึ้นต้นด้วย x86_64-win32 อันแรกสุด ซึ่งตามปกติ จะเป็นเวอร์ชันล่าสุด)



2. หลังจากดาวน์โหลดเสร็จ ปกติจะได้ไฟล์ที่ถูกบีบอัดเอาไว้ ก็ให้ขยายไฟล์ดังกล่าว ซึ่งหลังการขยายจะได้โฟลเดอร์ชื่อ mingw64 พร้อมกับไฟล์จำนวนมากในโฟลเดอร์นั้น ก็ให้เราคัดลอกโฟลเดอร์ mingw64 ไปเก็บไว้ยังตำแหน่งที่เราอ้างอิงได้สะดวก ซึ่งโดยทั่วไป เรานิยมเก็บไว้ที่ไดรฟ์ C ดังนั้น ตำแหน่งของมันจะเป็น C:\mingw64 (หรือจะขยายไฟล์มาไว้ที่นี้โดยตรงเลยก็ได้)

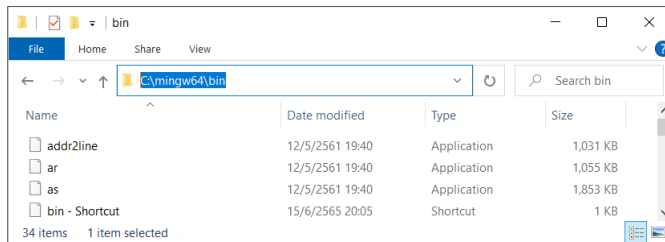


วิธีการตามที่คุณเขียนเลือกใช้ดังที่กล่าวมานี้ เราเพียงแค่คัดลอกไฟล์มาก็ใช้งานได้แล้ว แต่อย่าลืมตำแหน่งที่เราเก็บไฟล์เอาไว้คือ **C:\mingw64** ซึ่งต้องใช้อ้างอิงในหัวข้อต่อไป

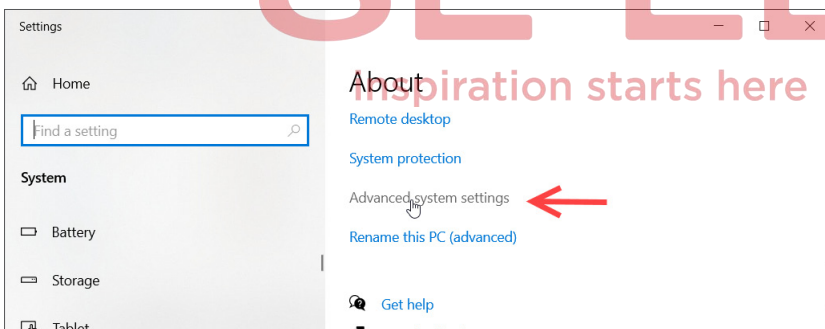
การเซตตำแหน่งของตัวแปลภาษา

แม้เราจะนำชุดไฟล์ของ MinGW มาจัดเก็บในเครื่องแล้ว แต่ระบบหรือโปรแกรมอื่นๆ ก็ยังไม่รู้จักตำแหน่งของมัน ดังนั้น เราต้องเซตตำแหน่งของ MinGW ลงใน Environment ของระบบ เพื่อให้โปรแกรมอื่นๆ รู้จักตำแหน่งของมัน และเข้าถึงได้ โดยมีขั้นตอนดังนี้

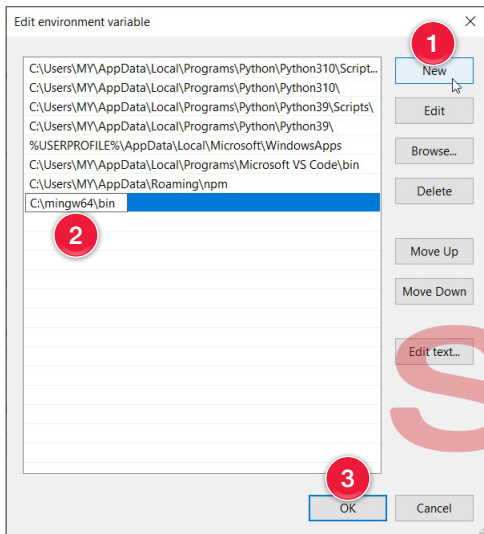
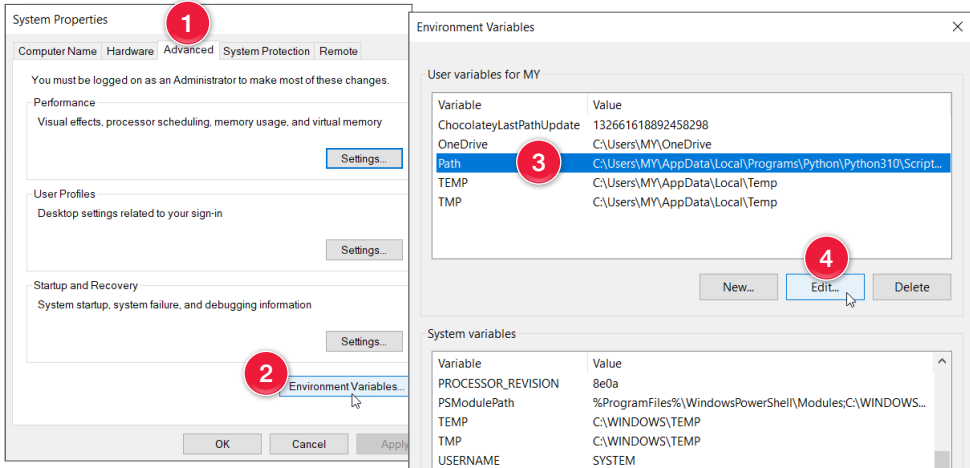
1. จากขั้นตอนการติดตั้งในหัวข้อที่แล้ว ผู้เขียนได้บอกไว้ว่า เราต้องจำไว้ด้วยว่าติดตั้ง MinGW เอาไว้ที่ใด ซึ่งตามปกติหากใช้ตามค่าที่ผู้เขียนแนะนำ ตำแหน่งของตัวแปลภาษาก็จะอยู่ที่ **C:\mingw64\bin** ถ้าไม่แน่ใจ ให้เปิดไปตรวจสอบอีกครั้งก็ได้



2. ที่ระบบ Windows ให้คลิกขวาที่ปุ่ม Start แล้วเลือก **System** (หรือคลิกขวาที่ไอคอน My Computer หรือ This PC จากนั้นเลือกเมนู Properties หรือใช้วิธีอื่นๆ นอกเหนือจากนี้ก็ได้) แล้วในขั้นตอนถัดไปให้คลิกที่ลิงก์หรือเมนู **Advanced system settings** (ระบบปฏิบัติการแต่ละเวอร์ชันอาจจัดวางตำแหน่งที่แตกต่างกัน)



3. ในขั้นตอนถัดไป เมื่อปรากฏหน้าจอ System Properties ให้เลือกแท็บ **Advanced** จากนั้นคลิกที่ปุ่ม **Environment Variables...**
4. เมื่อปรากฏหน้าจอ Environment Variables ให้คลิกที่ตัวแปร (Variable) **Path** จากนั้นคลิกปุ่ม **Edit...**



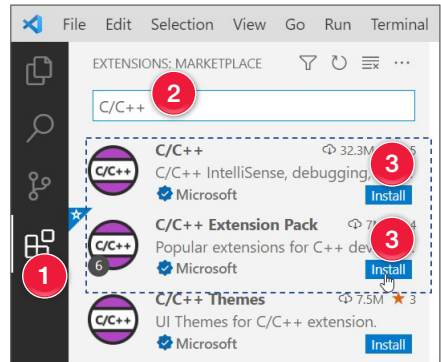
5. เมื่อปรากฏหน้าจอ Edit environment variable ให้คลิกปุ่ม **New** แล้วใส่ตำแหน่งของ MinGW ลงไปตามที่ได้จัดเก็บไฟล์เอาไว้ ซึ่งในกรณีของผู้เขียนคือ C:\mingw64\bin ทั้งนี้หากผู้อ่านจัดเก็บที่ตำแหน่งอื่น ก็ให้ระบุไปตามนั้น เสร็จแล้วคลิกปุ่ม OK

6. เมื่อย้อนกลับมายังหน้าจอในขั้นตอนก่อนนี้ ก็ให้คลิกปุ่ม OK ทั้งหมด ก็ถือเป็นอันเสร็จสิ้นขั้นตอนการเซตค่า

การติดตั้ง C/C++ Package สำหรับ VS Code

จากที่ได้กล่าวไปบ้างแล้วว่า VS Code เป็นเครื่องมือสำหรับเขียนโค้ดทั่วไป ที่ไม่เจาะจงว่าจะใช้กับภาษาคอมพิวเตอร์อันใดอันหนึ่ง ดังนั้น หากเราจะใช้ในการเขียนโค้ดภาษาใด ก็อาจต้องติดตั้งส่วนเสริม (Extension) เพื่อเป็นตัวช่วยสำหรับใช้งานร่วมกับภาษานั้นๆ สำหรับในกรณีของภาษา C ก็ให้เราติดตั้งส่วนเสริมที่ชื่อ C/C++ Package ที่พัฒนาโดยทีมงานของ Microsoft โดยตรง ดังขั้นตอนต่อไปนี้

1. เปิดเข้าสู่ VS Code แล้วคลิกที่ไอคอน Extensions ที่แถบด้านซ้ายมือ (การติดตั้งส่วนเสริมต้องเชื่อมต่ออินเทอร์เน็ต)
2. ในช่องค้นหา ให้พิมพ์คำว่า **C/C++** ลงไป
3. หลังจากปรากฏรายชื่อส่วนเสริม ซึ่งมีอยู่ค่อนข้างมาก แต่อันที่แนะนำให้ติดตั้งโดยคลิกปุ่ม Install ตรงส่วนเสริมอันนั้น (หมายเลข 3 ในภาพ) คือ
 - 1) **C/C++**
 - 2) **C/C++ Extension Pack**

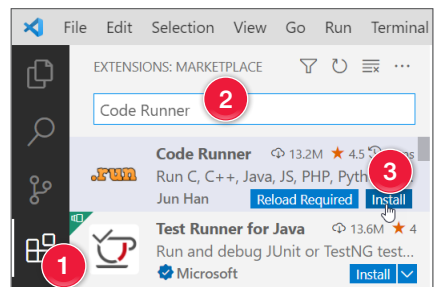


ส่วนเสริม C/C++ เราติดตั้งตั้งขั้นตอนที่ผ่านมา เป็นเพียงตัวช่วยในการเขียนโค้ดเท่านั้น แต่หากเราต้องการความสะดวกสบายในการทดสอบโค้ดที่เขียน ก็อาจต้องติดตั้งส่วนเสริมอื่นๆ เพิ่มลงไปอีก ตามที่จะกล่าวถึงในหัวข้อต่อไป

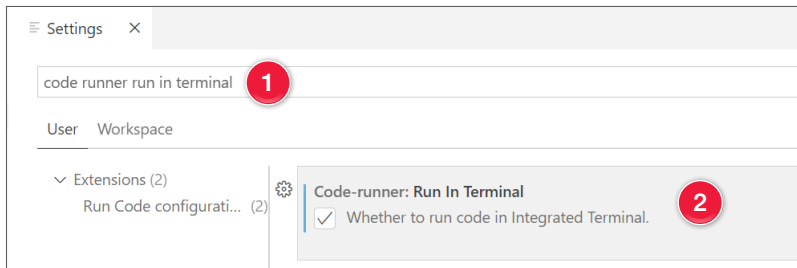
การติดตั้ง Code Runner สำหรับ VS Code

การติดตั้ง VS Code และ MinGW รวมทั้งส่วนเสริมของ C/C++ ตั้งขั้นตอนที่ได้กล่าวมาแล้ว แม้จะเพียงพอที่จะเขียนโค้ดภาษา C บน VS Code ได้แล้ว แต่ก็ยังมีความยุ่งยากในการรันทดสอบ เพราะหากต้องการดูผลลัพธ์ของโค้ดที่เขียน เราต้องพิมพ์คำสั่งลงไปเอง ดังนั้น เพื่อลดความยุ่งยากดังกล่าว เราอาจติดตั้งส่วนเสริมของ VS Code ที่ชื่อ **Code Runner** เป็นตัวช่วยในขั้นตอนการรันทดสอบเพื่อดูผลลัพธ์ ตั้งขั้นตอนต่อไป

1. เปิดเข้าสู่ VS Code แล้วคลิกที่ไอคอน Extensions ที่แถบด้านซ้ายมือ (ต้องเชื่อมต่ออินเทอร์เน็ต)
2. ในช่องค้นหา ให้พิมพ์คำว่า **Code Runner** ลงไป
3. หลังจากปรากฏรายชื่อส่วนเสริม Code Runner ก็ให้คลิกปุ่ม Install
4. หลังติดตั้งเสร็จ ให้ปิดโปรแกรม VS Code แล้วค่อยเปิดขึ้นมาใหม่



- ต่อไปเราจะตั้งค่า เพื่อให้ Code Runner นำผลลัพธ์ไปแสดงที่ Terminal ของ VS Code โดยเลือกเมนู **File > Preferences > Settings**
- การค้นหาตัวเลือกเป้าหมาย วิธีที่สะดวกรวดเร็วที่สุดคือ เมื่อเปิดเข้าสู่แท็บหรือหน้าจอ Settings ให้พิมพ์ค้นหาด้วยคำว่า **"code runner run in terminal"** ก็จะปรากฏตัวเลือกเป้าหมายคือ **Code-runner: Run In Terminal** (รายการนี้จะปรากฏเมื่อติดตั้ง Code Runner เอาไว้แล้วเท่านั้น) ต่อไปเราก็เลือก (คลิก) ที่รายการนั้น ดังภาพ



- หลังจากทำตามขั้นตอนดังที่กล่าวมาจนครบแล้ว ต้องบันทึกการเปลี่ยนแปลงเสมอ เช่น เลือกเมนู **File > Save** หรือกด **<Ctrl + S>** จากนั้นก็สามารถปิดแท็บ Settings ได้เลย และถือเป็นอันเสร็จสิ้นขั้นตอนการตั้งค่า

การเลือกที่ Run In Terminal ตามที่เราเซตค่าในขั้นตอนดังกล่าวมานี้ ก็เพื่อให้ Code Runner แสดงผลลัพธ์และรับข้อมูลจากคีย์บอร์ดผ่านทาง Terminal ของ VS Code โดยตรงนั่นเอง

การทดสอบโค้ดภาษา C

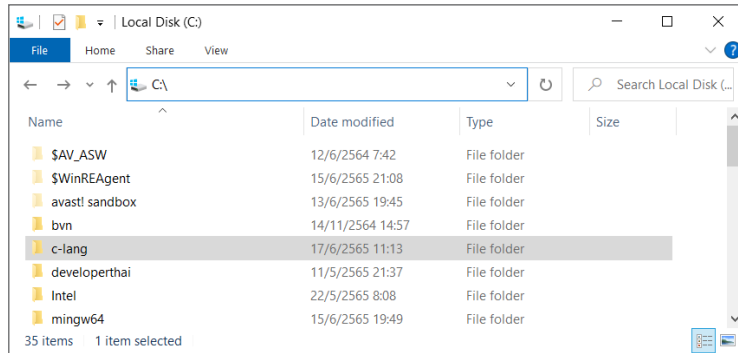
inspiration starts here

หลังจากที่เราได้จัดเตรียมเครื่องมือที่ต้องใช้สำหรับการทดสอบภาษา C จนครบทั้งหมดแล้ว ดังหัวข้อที่ผ่านๆ มา ขั้นตอนต่อไปเราก็จะทดสอบเครื่องมือเหล่านั้นว่าใช้งานร่วมกันได้จริงหรือไม่ โดยมีแนวทางดังต่อไปนี้

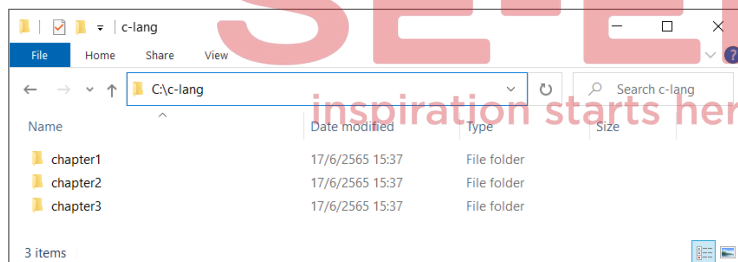
สร้างโฟลเดอร์สำหรับเก็บโค้ด

ตามหลักการที่ถูกต้องของ VS Code นั้น เราควรสร้างโฟลเดอร์สำหรับจัดเก็บไฟล์ของโค้ดให้เป็นสัดส่วน เพราะการดำเนินการบางอย่างเราต้องทำกับโฟลเดอร์ที่บรรจุไฟล์เท่านั้น ซึ่งแนวทางที่ผู้เขียนจะเลือกใช้ประกอบในหนังสือเล่มนี้คือ

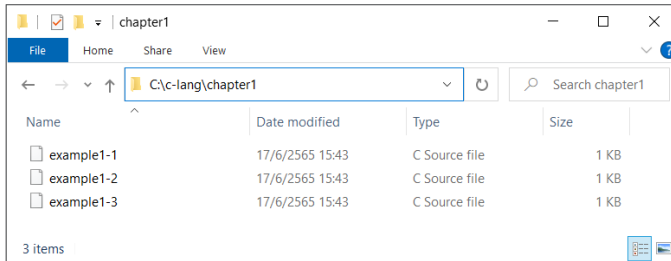
- กำหนดโฟลเดอร์ชื่อ c-lang สำหรับเก็บโค้ดรวมของทุกบท โดยโฟลเดอร์นี้จะสร้างไว้ที่ C:\c-lang



- เนื่องจากในแต่ละบท จะมีไฟล์ย่อยๆ ของแต่ละตัวอย่างเป็นจำนวนมาก ดังนั้น เพื่อให้ง่ายต่อการเข้าถึงไฟล์เป้าหมาย เราจะสร้างโฟลเดอร์ย่อยๆ ซ้อนไว้ใน C:\c-lang เพื่อจัดเก็บโค้ดของแต่ละบท เช่น
 - โฟลเดอร์ chapter1 สำหรับเก็บโค้ดของบทที่ 1 จะอยู่ที่ C:\c-lang\chapter1\
 - โฟลเดอร์ chapter2 สำหรับเก็บโค้ดของบทที่ 2 จะอยู่ที่ C:\c-lang\chapter2\
 - โฟลเดอร์ chapter3 สำหรับเก็บโค้ดของบทที่ 3 จะอยู่ที่ C:\c-lang\chapter3\
 - โฟลเดอร์อื่นๆ ก็ใช้หลักการเดียวกัน ...

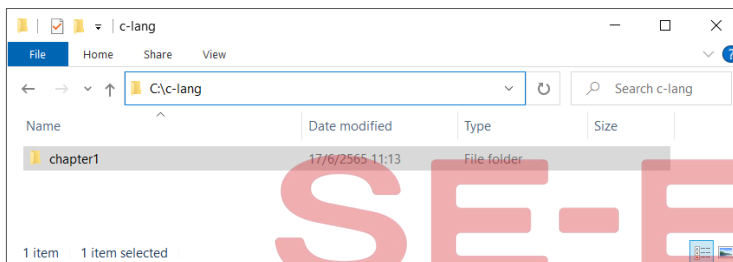


- ในแต่ละบท จะสร้างไฟล์ตามชื่อบท-ลำดับตัวอย่าง เช่น
 - โค้ดของตัวอย่าง 1-1 จะอยู่ที่ไฟล์ C:\c-lang\chapter1\example1-1.c
 - โค้ดของตัวอย่าง 1-2 จะอยู่ที่ไฟล์ C:\c-lang\chapter1\example1-2.c
 - โค้ดของตัวอย่าง 2-1 จะอยู่ที่ไฟล์ C:\c-lang\chapter2\example2-1.c
 - โค้ดของตัวอย่าง 3-5 จะอยู่ที่ไฟล์ C:\c-lang\chapter3\example3-5.c
 - โค้ดของตัวอย่างอื่นๆ ก็ใช้หลักการเดียวกัน ...



หลังจากที่เราทราบหลักการโดยสังเขปแล้ว ต่อไปเราจะลองทดสอบการใช้งาน โดยในขั้นตอนแรก ถ้าเรายังไม่มีโฟลเดอร์รวม ให้สร้างขึ้นมาก่อน ดังนี้

1. เปิด File Explorer ของระบบ Windows แล้วไปที่ C:\
2. สร้างโฟลเดอร์ใหม่ชื่อ c-lang (เช่น คลิกขวาแล้วเลือก New > Folder)
3. เปิดเข้าไปในโฟลเดอร์ c-lang แล้วสร้างโฟลเดอร์ชื่อ chapter1 สำหรับเก็บไฟล์ของบทที่ 1 (ขั้นตอนนี้อาจไปสร้างใน VS Code ก็ได้)

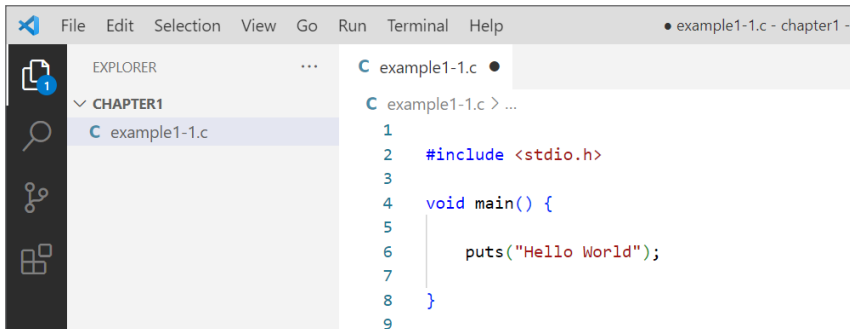
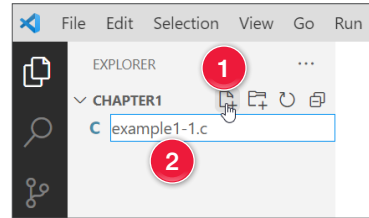


การสร้างไฟล์และทดสอบโค้ดภาษา C

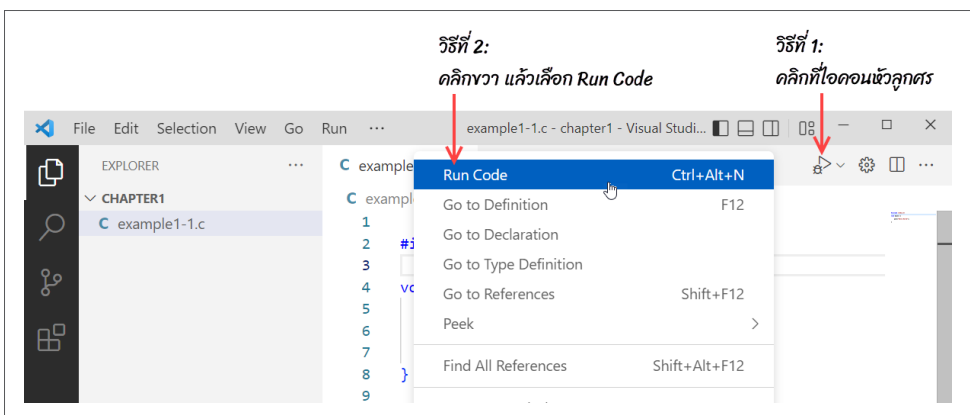
หลังจากที่เราสร้างโฟลเดอร์สำหรับจัดเก็บโค้ดเอาไว้แล้ว ดังหัวข้อที่ผ่านมา ต่อไปเราจะสร้างไฟล์ภาษา C เพื่อทดสอบการทำงาน ดังนี้

1. เปิดเข้าสู่ VS Code
2. คลิกเมนู File > Open Folder หรือเลือกจากลิงก์ Open Folder บนหน้าจอ Get Started ก็ได้ จากนั้นเลือกโฟลเดอร์ของบทที่ 1 ตามที่เราได้สร้างเอาไว้ นั่นคือ C:\c-lang\chapter1

3. คลิกที่ไอคอน New File ดังภาพ แล้วกำหนดชื่อไฟล์ โดยให้มีส่วนขยายหรือนามสกุลของไฟล์เป็น .c เท่านั้น (หมายถึงไฟล์ภาษา C) เช่นในที่นี้กำหนดชื่อเป็น example1-1.c
4. ให้ลองพิมพ์โค้ดง่ายๆ ลงไป ดังนี้ (รายละเอียดของโค้ดจะกล่าวถึงในบทต่อไป)



5. หลังการเขียนหรือแก้ไขโค้ด ต้องบันทึกการเปลี่ยนแปลงทุกครั้ง เช่น เลือกเมนู File > Save หรือกด <Ctrl + S>
6. รันทดสอบโค้ดที่เขียนด้วยวิธีใดวิธีหนึ่งคือ
 - **วิธีที่ 1** คลิกที่ปุ่มไอคอนหัวลูกศรที่อยู่บริเวณมุมขวาบนของพื้นที่เขียนโค้ด
 - **วิธีที่ 2** คลิกขวาบริเวณพื้นที่เขียนโค้ด แล้วเลือกเมนู Run Code (เมนูรายการนี้จะปรากฏเฉพาะกรณีที่เรากดติดตั้งส่วนเสริม Code Runner เอาไว้แล้วเท่านั้น) หรือกด <Alt + Ctrl + N>



7. เมื่อสั่งรันตามที่กล่าวมา ผลลัพธ์ที่ได้จะถูกนำมาแสดงบน Terminal (จะปรากฏโดยอัตโนมัติ) ดังภาพ

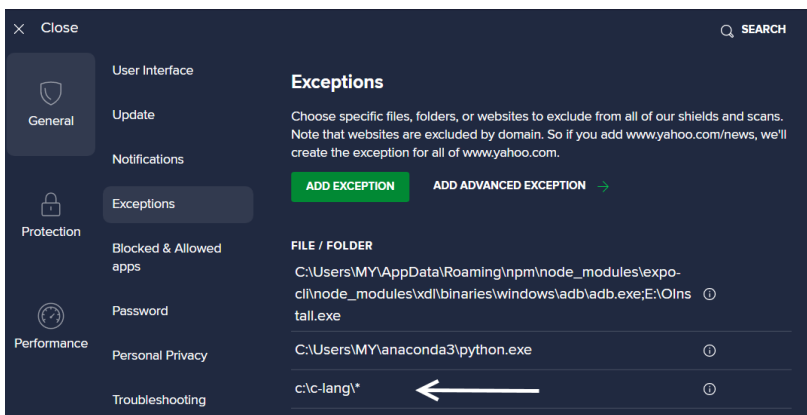
```

C:\c-lang\chapter1>cd "c:\c-lang\chapter1" && gcc example1-1.c -o example1-1 && "c:\c-lang\chapter1\example1-1"
Hello World
C:\c-lang\chapter1>

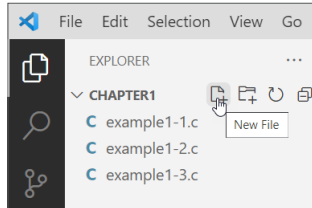
```

อย่างไรก็ตาม เนื่องจากการรันโค้ดภาษา C จะได้ไฟล์ .exe เพิ่มขึ้นอีกไฟล์ ซึ่งโปรแกรมประเภท Anti-Virus อาจมองว่าไม่ปลอดภัย และอาจบล็อกการรันก็เป็นได้ ซึ่งเราอาจแก้ปัญหาด้วยวิธีใดวิธีหนึ่งคือ

- **วิธีที่ 1** อาจหยุดการทำงานของโปรแกรม Anti-Virus ชั่วคราวในระหว่างที่กำลังทดสอบการรันภาษา C แต่วิธีนี้อาจไม่ปลอดภัย หากในขณะนั้นเราใช้อินเทอร์เน็ตควบคู่กันไปด้วย
- **วิธีที่ 2** อาจสร้าง Exception ภายในโปรแกรม Anti-Virus เพื่อให้มันยกเว้นการตรวจสอบตำแหน่งไฟล์เดอรัที่เรাজัดเก็บโค้ด เช่น ในที่นี้คือ C:\c-lang ดังในภาพถัดไป เป็นการสร้าง Exception บนโปรแกรม Avast (เลือกไอคอน Menu > Settings > Exceptions > Add Exception)



หากเราต้องการสร้างไฟล์สำหรับตัวอย่างต่อไป ก็ทำเหมือนเดิม นั่นคือ เปิดเข้าสู่ VS Code แล้วเข้าสู่โฟลเดอร์ C:\c-lang\chapterX ที่ต้องการจัดเก็บไฟล์นั้น แล้วก็คลิกไอคอน New File จากนั้นก็กำหนดชื่อที่มีส่วนขยายเป็น .c ตามเดิม ซึ่งเราจะสร้างเพิ่มกี่ไฟล์ก็ได้



หลังจากที่เราเสร็จสิ้นการใช้งาน อาจปิดโฟลเดอร์นั้น เช่น เลือกที่เมนู File > Close Folder แต่กรณีนี้ หากจะใช้งานเมื่อเข้าสู่ VS Code ครั้งต่อไป ต้องเปิดโฟลเดอร์ก่อน ตามหลักการเดิมทั้งหมด

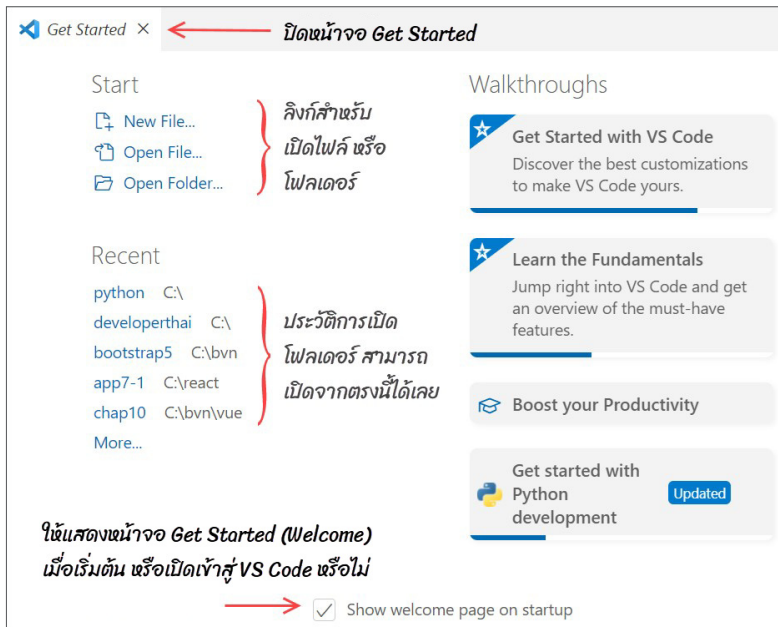
การใช้เครื่องมือพื้นฐานของ VS Code

ก่อนที่เราจะเข้าสู่ขั้นตอนการเขียนโค้ดภาษา C อย่างเต็มรูปแบบในบทต่อไป ก็ควรรู้จักกับการใช้เครื่องมือพื้นฐานบางส่วนของ VS Code เพิ่มเติม เพื่อจำเป็นต้องใช้งานในบางโอกาส โดยสิ่งที่เราควรรู้จักในเบื้องต้นมีดังนี้

หน้าจอรเริ่มต้น (Get Started)

ตามปกติ เมื่อเราเปิดเข้าสู่ VS Code จะปรากฏหน้าจอร Get Started เป็นอันดับแรก ซึ่งลักษณะที่สำคัญเกี่ยวกับหน้าจอรนี้คือ

- อาจมีการแจ้งข้อมูลข่าวสารที่น่าสนใจเกี่ยวกับ VS Code
- มีลิงก์สำหรับให้เราคลิกเพื่อเปิดไฟล์หรือโฟลเดอร์เป้าหมายที่ต้องการได้อีกหนึ่งวิธี นอกเหนือจากการเลือกที่เมนู
- มีลิงก์ที่แสดงประวัติการเปิดโฟลเดอร์ที่เราเข้าสู่ VS Code ในครั้งก่อนๆ ซึ่งหากมีโฟลเดอร์เป้าหมายที่เราต้องการรวมอยู่ด้วย ก็สามารถคลิกเปิดจากตรงนี้ได้ทันที

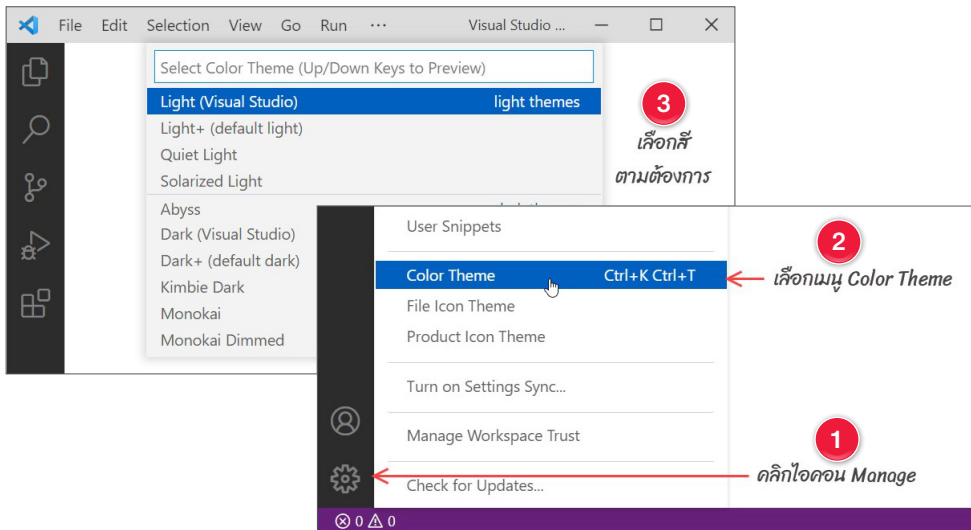


หากเราต้องการปิดหน้าจอนี้ ก็คลิกปุ่ม X ที่แท็บด้านบน หรือหากไม่ต้องการให้ปรากฏหน้าจอ Get Started ต่อไปอีก ก็ให้เราเอาเครื่องหมายถูกตรงตัวเลือก Show welcome page on startup ที่บริเวณขอบด้านล่างออก หรือหลังจากปิดไปแล้ว หากเราต้องการให้หน้าจอ Get Started กลับมาแสดงอีกครั้ง ก็ให้เลือกที่เมนู **Help > Get Started**

การเลือกรูปแบบสี (Theme)

ตามปกติ VS Code จะเลือกธีม (Theme) เป็นสีดำ (Dark) เอาไว้ล่วงหน้า ซึ่งหากเราชอบแบบนี้ก็ไม่จำเป็นต้องปรับเปลี่ยน แต่ถ้าเราต้องการสีอื่นในรูปแบบอื่น ก็มีตัวเลือกอยู่จำนวนหนึ่ง โดยใช้ขั้นตอนดังต่อไปนี้

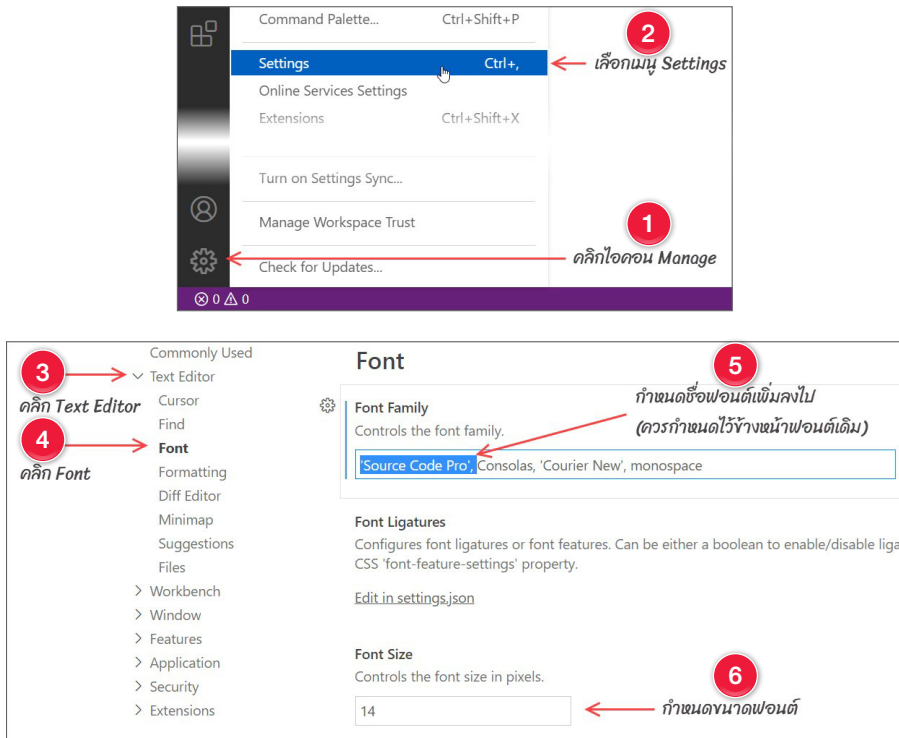
1. ที่มุมล่างขวาของโปรแกรม VS Code ให้คลิกที่ไอคอน Manage (รูปเฟือง)
2. เลือกที่เมนู Color Theme
3. จากนั้นจะปรากฏรายชื่อสีให้เลือก ถ้าเรายังไม่ทราบว่าแต่ละสีมีลักษณะเป็นยังไง ก็อาจลองเลือกดูก่อนก็ได้ ถ้าไม่ถูกใจก็ค่อยเลือกสีใหม่ สำหรับรูปแบบที่จะใช้ประกอบในหนังสือเล่มนี้ ผู้เขียนเลือกธีมสีแบบ Light (Visual Studio)



การปรับเปลี่ยนฟอนต์

ฟอนต์ (Font: ในที่นี้คือตัวอักษรที่ใช้เขียนโค้ดและแสดงผลลัพธ์) ที่ VS Code กำหนดเอาไว้ล่วงหน้า แม้จะเป็นรูปแบบมาตรฐานที่นิยมใช้งานกันทั่วไป แต่หากต้องการใช้ฟอนต์ที่เราถูกใจ ก็สามารถเปลี่ยนแปลงได้ด้วยขั้นตอนดังต่อไปนี้

1. ที่มุมล่างขวาของโปรแกรม VS Code ให้คลิกที่ไอคอน Manage (รูปเฟือง)
2. เลือกที่เมนู Settings
3. คลิกแท็บ Text Editor
4. คลิกแท็ก Font
5. กำหนดชื่อฟอนต์ลงในช่อง Font Family ซึ่งตามปกติ จะมีฟอนต์เดิมที่ VS Code กำหนดไว้ให้ล่วงหน้า หากเราต้องการเปลี่ยนไปใช้ชนิดอื่น ควรเพิ่มชื่อฟอนต์ไว้ข้างหน้าฟอนต์เดิม (ชื่อที่อยู่ก่อนจะถูกพิจารณาก่อน และไม่แนะนำให้ลบชื่อฟอนต์เดิมออก) โดยค้นด้วยเครื่องหมาย , และถ้าชื่อฟอนต์ชนิดนั้นมีช่องว่าง ให้เขียนไว้ในเครื่องหมาย '' ซึ่งชื่อฟอนต์ที่ระบุต้องถูกติดตั้งเอาไว้ในเครื่องนั้นอยู่ก่อนแล้ว (การติดตั้งฟอนต์ ให้คัดลอกไฟล์ของฟอนต์ไปไว้ที่ C:\Windows\Fonts แล้วเริ่มโปรแกรมที่จะใช้ฟอนต์ใหม่)
6. ถ้าต้องการกำหนดขนาด ก็ระบุตัวเลขลงในช่อง Font Size และเมื่อปิดหน้าจอ Setting ฟอนต์ที่ตั้งค่าเอาไว้ก็จะมีผลทันที



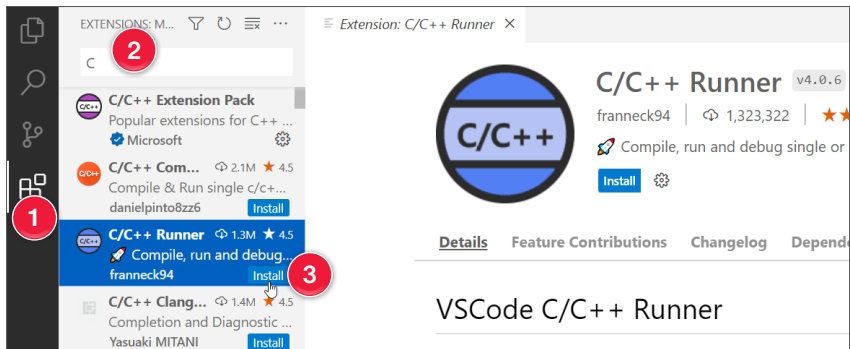
นอกจากนี้ ยังมีเทคนิคเพิ่มเติมอีกอย่างคือ การย่อหรือขยาย (Zoom) ขนาดของฟอนต์ ในโปรแกรม VS Code ให้ใหญ่ขึ้นหรือเล็กลง ซึ่งกรณีนี้ จะมีผลกับทั้งโค้ด เมนู และข้อความอื่นๆ ทั้งหมดบน VS Code โดยใช้วิธีการง่ายๆ คือ

- หากต้องการขยายขนาดเพิ่มขึ้น (Zoom In: คล้ายกับการเข้าไปมองใกล้ๆ ซึ่งจะเห็นขนาดที่ใหญ่ขึ้น) ให้กดคีย์บอร์ดปุ่ม <Ctrl => หรือเลือกเมนู **View > Appearance > Zoom In** ซึ่งหากทำซ้ำเช่นนี้อีก ขนาดของฟอนต์จะเพิ่มขึ้น 1 ระดับ ไปเรื่อยๆ
- หากต้องการลดขนาดให้เล็กลง (Zoom Out: คล้ายกับการออกมามองไกลๆ ซึ่งเราจะเห็นขนาดที่เล็กลง) ให้กดคีย์บอร์ดปุ่ม <Ctrl -> หรือเลือกเมนู **View > Appearance > Zoom Out** ซึ่งหากทำซ้ำเช่นนี้อีก ขนาดของฟอนต์จะลดลง 1 ระดับ ไปเรื่อยๆ

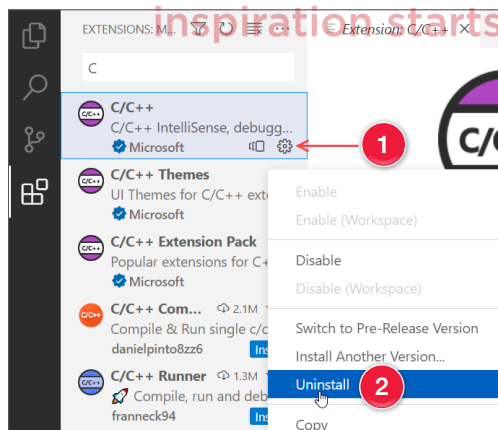
แนวทางการติดตั้งส่วนเสริมเพิ่มเติม

การเขียนโค้ดในบางกรณี เราอาจต้องใช้ส่วนเสริม (Extension) สำหรับการทำงานร่วมกับภาษานั้นๆ ซึ่งในหัวข้อที่ผ่านมา เราก็ได้ติดตั้งส่วนของภาษา C/C++ ไปแล้ว จึงขอลำดับถึงอีกครั้งหนึ่ง เนื่องจากส่วนเสริมของ VS Code ยังมีอีกเป็นจำนวนมาก ซึ่งเราสามารถตรวจสอบและนำส่วนเสริมอื่นๆ มาใช้งานเพิ่มเติมได้ดังแนวทางต่อไปนี้

1. ในขณะนั้น ต้องเชื่อมต่อกับอินเทอร์เน็ต แล้วคลิกที่ไอคอน Extensions ที่ขอบด้านซ้าย (รูปภาพถัดไป) หรือเลือกที่เมนู **View > Extensions**
2. ใส่คีย์เวิร์ดลงไป หลังจากนั้นจะปรากฏรายชื่อส่วนขยายที่อาจเกี่ยวข้องกับคีย์เวิร์ดดังกล่าว
3. ถ้าเราต้องการติดตั้งส่วนเสริมใด ก็คลิกปุ่ม Install บนรายการนั้น หรือหากต้องการดูรายละเอียดเพิ่มเติมเกี่ยวกับส่วนเสริมใด ก็คลิกที่รายการดังกล่าว ดังภาพ



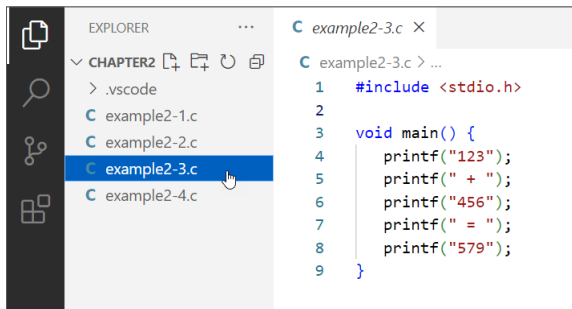
4. ถ้าค้นหาส่วนเสริมใดที่ถูกติดตั้งเอาไว้แล้ว จะมีรูปไอคอน Manage ที่มุมล่างขวาของรายการนั้น ทั้งนี้หากเราต้องการลบส่วนขยายนั้นทิ้งไป จากนั้นทำดังนี้
 - (1) คลิกที่ไอคอน Manage ที่ส่วนเสริมนั้น (รูปเฟือง)
 - (2) เมื่อปรากฏเมนู ให้คลิกที่ Uninstall



การนำโค้ดของหนังสือมาใช้งาน

สำหรับโค้ดของหนังสือเล่มนี้ ผู้อ่านสามารถดาวน์โหลดได้ที่ <https://www.developerthai.com> หลังจากนั้นก็ให้ขยายไฟล์ดังกล่าว ซึ่งจะได้โฟลเดอร์ที่ชื่อ c-lang ก็ให้นำโฟลเดอร์ดังกล่าวไปเก็บไว้ที่ C:\ ดังนั้น ตำแหน่งของมันก็จะ เป็น C:\c-lang โดยภายในโฟลเดอร์ c-lang จะประกอบด้วยไฟล์ย่อยๆ ของบทต่างๆ เช่น chapter1, chapter2, ... ดังหลักการที่ได้กล่าวมาแล้ว

เมื่อต้องการเปิดดูโค้ด และ/หรือ รันทดสอบ ก็ให้เข้าสู่ VS Code แล้วเปิดโฟลเดอร์ C:\c-lang แล้วโฟลเดอร์ของทุกบทก็จะถูกเปิดพร้อมกันทั้งหมด หรือจะเลือกเปิดเฉพาะโฟลเดอร์ของบทที่ต้องการ เช่น C:\c-lang\chapter2 ซึ่งจะเข้าถึงไฟล์ได้ง่ายขึ้น หลังจากนั้นก็คลิกชื่อไฟล์ของตัวอย่างที่ต้องการได้เลย



สิ่งที่เราได้เรียนรู้ในบทนี้ เป็นการเตรียมเครื่องมือก่อนจะเข้าสู่ขั้นตอนการเขียนโค้ดภาษา C อย่างแท้จริงในบทต่อไป ซึ่งหากเรามีเครื่องมือที่ครบถ้วนและรู้จักวิธีการใช้งานที่ดีพอในระดับหนึ่งแล้ว ก็จะมีส่วนช่วยให้เราศึกษาเรียนรู้ได้รวดเร็วยิ่งขึ้นด้วย

inspiration starts here



2

การเขียนโค้ดภาษา C ในเบื้องต้น

ภาษา C มีองค์ประกอบและข้อกำหนดที่สำคัญหลายอย่างซึ่งเราควรรู้จักในเบื้องต้น โดยสิ่งที่จะกล่าวถึงในบทนี้ เช่น คำสงวน คำสั่งที่จำเป็นสำหรับการเขียนโค้ดในเบื้องต้น การแสดงผล การกำหนดบล็อกและคำอธิบายโค้ด เป็นต้น ซึ่งสิ่งเหล่านี้ล้วนเป็นพื้นฐานสำคัญของภาษา C ที่เราต้องใช้งานกันไปตลอด

ประวัติโดยย่อของภาษา C

ภาษา C ถูกสร้างขึ้นเมื่อประมาณปี ค.ศ. 1972 โดย Dennis Ritchie ที่ห้องปฏิบัติการของ Bell Labs (หรือ AT & T ในปัจจุบัน) ซึ่งวัตถุประสงค์เริ่มแรกก็เพื่อใช้ในระบบ Unix อย่างไรก็ตาม Dennis Ritchie ไม่ใช่ผู้บุกเบิกในการสร้างภาษา C มาตั้งแต่เริ่มต้น แต่ภาษานี้ได้จากการพัฒนาเพิ่มเติมต่อจากภาษาคอมไพเลอร์อื่นๆ ที่มีอยู่ก่อนแล้วในขณะนั้น และหลังจากยุคของ Dennis Ritchie ก็มีหลายองค์กรที่ได้นำภาษา C มาพัฒนาเพิ่มเติมเรื่อยมา ซึ่งเราสามารถสรุปวิวัฒนาการที่เกี่ยวข้องกับภาษา C โดยสังเขปตั้งแต่อดีตจนถึงปัจจุบัน ได้ดังนี้

- 1960: ภาษา ALGO
 - สร้างโดยองค์กรระหว่างประเทศ (International Committee)
 - ไม่มีข้อกำหนดการใช้งานที่ชัดเจน เนื่องจากมีลักษณะที่เป็นนามธรรม (Abstract) มากเกินไป
- 1963: ภาษา CPL (Combine Programming Language)
 - สร้างขึ้นที่มหาวิทยาลัยเคมบริดจ์ (Cambridge University)
 - มีลักษณะที่เจาะจงมากเกินไป จึงไม่สะดวกต่อการใช้งานจริง

- 1967: ภาษา **BCPL** (Basic Combine Programming Language)
 - ◎ สร้างโดย Martin Richards ที่มหาวิทยาลัยเคมบริดจ์
 - ◎ ยังมีลักษณะที่เจาะจงเหมือนเดิม จึงเหมาะกับการใช้งานเฉพาะอย่าง เพราะขาดความยืดหยุ่นต่อการใช้งานทั่วไป
- 1970: ภาษา **B**
 - ◎ สร้างโดย Ken Thompson ที่ห้องปฏิบัติการ Bell Labs (AT &T)
 - ◎ สร้างขึ้นจากพื้นฐานของภาษา BCPL (จึงเป็นที่มาของชื่อภาษา B) โดยแก้ไขข้อบกพร่องหลายอย่างและทำให้ภาษามีขนาดเล็กลง เช่น การลดจำนวนคีย์เวิร์ด แต่ก็ยังเหมาะกับการแก้ปัญหาที่มีลักษณะเฉพาะเจาะจงเหมือนเดิม
- 1972: ภาษา **C**
 - ◎ สร้างโดย Dennis Ritchie
 - ◎ สร้างขึ้นจากพื้นฐานของภาษา BCPL และภาษา B ร่วมกัน จึงกลายเป็นที่มาของชื่อภาษา C ว่าพัฒนาต่อจากภาษาในตระกูล B
 - ◎ ภาษา C มีวิวัฒนาการต่อเนื่องมาหลายเวอร์ชัน ซึ่งสามารถสรุปได้ดังนี้
 - ◎ 1972: **กำเนิดภาษา C** (หรือเรียกว่า C72)
 - ◎ 1978: **K&R C** (หรือเรียกว่า C78) โดย K มาจากชื่อของ Ken Thompson และ R มาจากชื่อของ Dennis Ritchie
 - ◎ 1989: **ANSI C** (หรือเรียกว่า C89)
 - ◎ 1990: **ISO C** (C90)
 - ◎ 1999: **C99**
 - ◎ 2011: **C11**
 - ◎ 2017: **C17**
 - ◎ 2023: **C23** (เป็นเพียงการคาดการณ์ในขณะที่เขียนหนังสือเล่มนี้ แต่ยังไม่มีการประกาศใช้งานจริง)

ข้อดีและข้อเสียของภาษา C

ถึงแม้ภาษา C จะถูกสร้างมานานมากแล้ว รวมถึงเป็นต้นแบบให้กับกับภาษาอื่นๆ อีกหลายภาษา หากนับจากอดีตถึงปัจจุบัน ก็ถือว่าเป็นภาษาที่ประสบความสำเร็จและมีอิทธิพลต่อเทคโนโลยีด้านไอทีมากที่สุดอีกภาษาหนึ่ง และมีกลุ่มนักพัฒนาจำนวนมากจากทั่วโลกที่ยังคงเลือกใช้ภาษา C

กันต่อไป ถึงแม้จะมีภาษาอื่น ๆ ที่ถูกพัฒนาขึ้นมาภายหลังและพยายามจะมาทดแทนภาษานี้ก็ตาม แต่ภาษา C ก็ยังเป็น 1 ใน 5 ของภาษาคอมพิวเตอร์ที่ผู้ใช้งานมากที่สุดในโลกไม่เคยเปลี่ยนแปลงถึงอย่างไรก็ตาม ภาษา C ก็มีข้อดีและข้อเสียเช่นเดียวกับภาษาอื่น ๆ ซึ่งเราสามารถสรุปได้ดังนี้

ข้อดีบางส่วนของภาษา C

- มีรูปแบบไวยากรณ์การเขียนโค้ดที่ไม่ยากและเข้าใจได้ง่าย
- สามารถเรียนรู้ได้ในระยะเวลาอันรวดเร็ว จึงเหมาะที่จะศึกษาเป็นภาษาเริ่มแรก ก่อนก้าวไปสู่ภาษาอื่น ๆ ต่อไป เช่น C++, C# เป็นต้น
- ทำงานได้เร็วกว่าเมื่อเทียบกับภาษาอื่น ๆ ถือเป็นจุดเด่นที่สำคัญยิ่งของภาษานี้
- มีไลบรารีให้เลือกใช้จำนวนมากและครอบคลุมการทำงานที่สำคัญทั้งหมด
- รองรับการใช้งานในทุกระบบ เนื่องจากภาษา C เป็นพื้นฐานที่ถูกใช้ในการพัฒนาระบบปฏิบัติการทั้งหมดไม่ว่าจะเป็น Unix/Linux, Windows, Mac, iOS, Android, ... ดังนั้น จึงสามารถนำโค้ดที่เขียนไปใช้กับทุกระบบ ตามคำกล่าวที่ว่า write once, compile everywhere

ข้อเสียบางส่วนของภาษา C

- ภาษา C เป็นภาษาแบบ Procedural (Function) ไม่ใช่ภาษาแบบ OOP (Object Oriented Programming) จึงอาจเกิดความยุ่งยากต่อการนำไปสร้างเป็นแอปพลิเคชันที่ซับซ้อนมากขึ้น เมื่อเทียบกับภาษาที่เป็น OOP เช่น C++, C#, Java, Objective-C เป็นต้น
- ไม่มีระบบจัดการปัญหา Garbage Collection หรือส่วนของหน่วยความจำที่ไม่จำเป็นต้องใช้งานต่อไปอีก อาจทำให้เกิดปัญหาหน่วยความจำเต็ม หรือมีพื้นที่ว่างเหลือไม่พอ
- ไม่มีวิธีการป้องกันและจัดการข้อผิดพลาดในแบบ Exception Handling เหมือนที่มีในภาษาอื่น ๆ จึงอาจทำให้เกิดข้อผิดพลาดในระหว่างรันได้ง่าย
- ภาษา C ยังขาดเทคนิคที่จะช่วยให้การเขียนโค้ดนั้นง่ายขึ้น บางอย่างก็ใช้วิธีการที่ล้าสมัย ถ้าเทียบกับภาษาใหม่ๆ ที่สร้างขึ้นในภายหลัง ซึ่งส่วนใหญ่จะมีแนวคิดและหลักการที่ทันสมัยมากกว่า ทั้งนี้ก็อาจเนื่องมาจากภาษา C นั้น ถูกสร้างมานานมากแล้ว และเป็นรากฐานของภาษาอื่น ๆ อีกหลายภาษา รวมถึงมีหลายมาตรฐาน ดังนั้น การเปลี่ยนแปลงกฎเกณฑ์ต่าง ๆ จึงทำได้ยาก

คำสงวนของภาษา C

คำสงวน (Reserved Words) หรือคีย์เวิร์ด (Keywords) เป็นคำสำหรับใช้เป็นคำสั่งเพื่อควบคุมหรือกำหนดวิธีการทำงานในภาษา C ซึ่งเราจะนำคำเหล่านี้ไปตั้งเป็นชื่อตัวแปร หรือฟังก์ชันไม่ได้ โดยคำที่ถูกใช้เป็นคำสงวนในภาษา C คือ

auto	do	goto	signed	unsigned
break	double	if	sizeof	void
case	else	int	static	volatile
char	enum	long	struct	while
const	extern	register	switch	inline (C99+)
continue	float	return	typedef	restrict (C99+)
default	for	short	union	_Bool (C99+)

นอกจากนี้ ยังมีคำสงวนอีกบางส่วนที่ถูกเพิ่มเติมเข้ามาในเวอร์ชันหลังๆ แต่เรามักไม่ค่อยได้ใช้งานสำหรับการเขียนโปรแกรมทั่วไป กันบ่อยนัก จึงไม่ขอนำมากล่าวถึง โดยวัตถุประสงค์พอสังเขปสำหรับคำสงวนบางส่วน มีดังนี้ (รายละเอียดที่ชัดเจนของคำเหล่านี้ เราจะได้เรียนรู้ในภายหลัง)

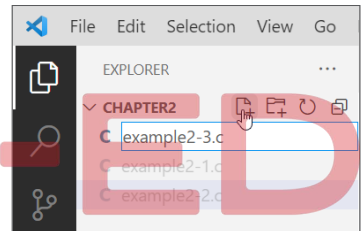
- **int, float, char, double, long, _Bool** : ใช้ในการกำหนดชนิดข้อมูลของตัวแปร
- **if, else, switch, case, default** : ใช้ในการกำหนดเงื่อนไขเพื่อตัดสินใจ หรือควบคุมโครงสร้างการทำงานของโปรแกรมนั่นเอง
- **for, while, do** : รูปแบบการวนลูปเพื่อการทำซ้ำ
- **break** : ใช้เพื่อหยุดการวนรอบของลูป หรือออกจากเงื่อนไขแบบ switch case
- **void** : การส่งค่าเข้าและออกจากฟังก์ชัน
- **goto** : กำหนดตำแหน่งที่จะข้ามไปทำงาน
- **auto, signed, const, extern, register, unsigned, volatile** : ใช้ร่วมกับการประกาศตัวแปรสำหรับกรณีเฉพาะ
- **return** : สำหรับการส่งค่ากลับออกจากฟังก์ชัน หรือหยุดการทำงานของฟังก์ชัน
- **continue** : ให้วนลูปถัดไปทันที โดยไม่ทำคำสั่งที่เหลือภายในลูป

- **enum** : สร้างชุดหรือกลุ่มของค่าคงที่ (Set of constants)
- **sizeof** : สำหรับการตรวจสอบขนาด
- **struct, typedef** : สร้างชนิดให้กับกลุ่มข้อมูลแบบโครงสร้าง (Structures)
- **union** : กำหนดกลุ่มตัวแปรที่จะใช้ตำแหน่งและพื้นที่จัดเก็บข้อมูลบนหน่วยความจำร่วมกัน

บทวนการสร้างไฟล์ภาษา C

ในบทที่แล้ว เราได้ทราบแนวทางการสร้างและทดสอบไฟล์ภาษา C กันไปแล้ว โดยในบทนี้ก็มีหลายกรณีที่เรายังต้องเขียนโค้ดตัวอย่าง เพื่อทดสอบการใช้งานควบคู่กันไป ทั้งนี้ เราจำเป็นต้องแยกแต่ละตัวอย่างไว้คนละไฟล์ จึงขอแนะนำวิธีการสร้างไฟล์ภาษา C ที่เคยกล่าวไว้ในบทที่แล้ว มากล่าวทบทวนอีกครั้งหนึ่ง ดังนี้

- สำหรับตัวอย่างในบทที่ 2 เราจะเก็บไฟล์ทั้งหมดเอาไว้ที่ **C:\c-lang\chapter2** ดังนั้น เราจำเป็นต้องสร้างโฟลเดอร์ chapter2 ซ่อนไว้ในโฟลเดอร์ c-lang
- ใน VS Code ให้เปิดโฟลเดอร์ **C:\c-lang\chapter2** ที่ได้สร้างเอาไว้
- เมื่อต้องการสร้างไฟล์ตัวอย่างสำหรับบทนี้ ก็คลิกที่ไอคอน New File แล้วกำหนดชื่อไฟล์ โดยให้มีส่วนขยายเป็น .c เท่านั้น ซึ่งในบทนี้จะตั้งชื่อตัวอย่างต่างๆ เป็น example2-1.c, example2-2.c ... เป็นเช่นนี้ไปเรื่อยๆ



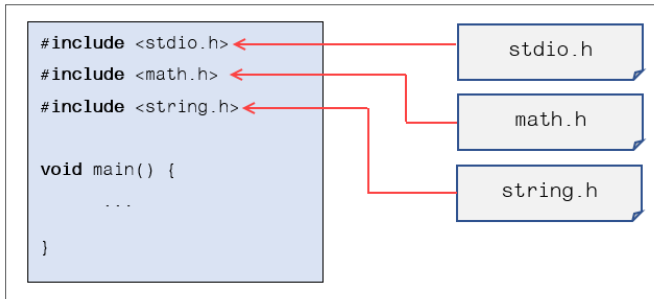
inspiration starts here

องค์ประกอบของการเขียนโค้ดภาษา C ในเบื้องต้น

ไม่ว่าจะเขียนโค้ดเพื่อวัตถุประสงค์ใดก็ตาม เราจำเป็นต้องใช้ข้อกำหนดพื้นฐานของภาษาเหมือนกัน กัน ดังนั้น ก่อนที่เราจะเข้าสู่ขั้นตอนการเขียนโค้ดที่ซับซ้อนมากขึ้นในลำดับต่อไป ก็ควรรู้จักกับองค์ประกอบของการเขียนโค้ดในเบื้องต้นเอาไว้ล่วงหน้า เพราะมันคือสิ่งจำเป็นที่เราต้องนำไปใช้งานอยู่ตลอด โดยจะแยกแต่ละประเด็นที่น่าสนใจออกเป็นหัวข้อต่างๆ ดังต่อไปนี้

คำสั่ง #include

คำสั่ง include ในภาษา C จะใช้สำหรับการอ้างถึงไฟล์ภายนอกที่จัดเก็บชุดคำสั่งพิเศษ ที่ต้องนำเข้ามาใช้งานร่วมกับโค้ดในไฟล์นั้นดังแนวทางในภาพ



ทั้งนี้ สาเหตุที่เราต้องอ้างอิงถึงชุดคำสั่งจากไฟล์ภายนอกก็เนื่องจาก ตามปกติแล้ว ชุดคำสั่งดังกล่าว ไม่ใช่พื้นฐานที่จำเป็นสำหรับการทำงานทุกๆ ไปของภาษา C แต่อาจใช้ในบางกรณี ดังนั้น เมื่อใดที่จำเป็นต้องใช้ ก็จะอ้างอิงด้วยคำสั่ง `include` ในรูปแบบดังนี้

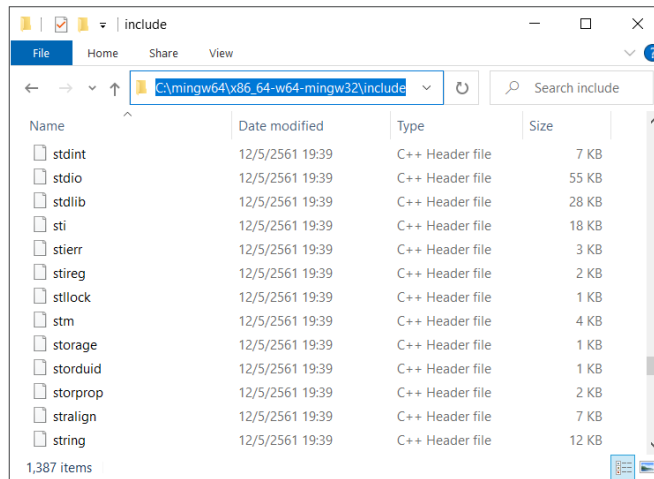
```
#include <ชื่อไฟล์.h>
หรือ #include "ชื่อไฟล์.h"
```

กลุ่มไฟล์ที่จัดเก็บชุดคำสั่งพิเศษดังกล่าว เราเรียกว่า เฮดเดอร์ (header) ด้วยเหตุนี้ จึงมีส่วนขยายเป็น `.h` ทั้งหมด โดยไฟล์เหล่านี้ เราจะได้เรียนรู้เพิ่มเติมต่อไปเรื่อยๆ แต่เฮดเดอร์ที่เราจะได้ใช้งานอยู่บ่อยๆ ก็คือ `stdio.h` ซึ่งจากที่ผ่านมาจะเห็นว่า เราเขียนโค้ดบรรทัดแรกเป็น

```
#include <stdio.h>
```

- **include** เป็นคำสั่งของภาษา C ที่ใช้อ้างถึงไฟล์ภายนอกดังที่กล่าวมาแล้ว โดยในกรณีนี้ต้องเขียนเครื่องหมาย `#` กำกับไว้ข้างหน้า เพื่อให้แตกต่างจากคำสั่งทั่วไป
- **stdio.h** คำว่า `stdio` ย่อมาจาก `standard input/output` ซึ่งเป็นมาตรฐานการรับและแสดงผลข้อมูลทางจอภาพ ถ้าเราไม่อ้างอิงเฮดเดอร์อันนี้ เมื่อเรียกใช้ฟังก์ชันที่เกี่ยวข้องกับการรับและแสดงผลข้อมูล เช่น `puts`, `gets`, `printf`, `scanf` และอีกหลายฟังก์ชัน ก็จะทำให้เกิดข้อผิดพลาดเมื่อรันทดสอบ ซึ่งเราได้เรียนรู้เพิ่มเติมในภายหลัง

ถ้าเราติดตั้งตัวแปลภาษา C ตามวิธีการของหนังสือเล่มนี้ (กล่าวไว้ในบทที่ 1) ตำแหน่งของไฟล์เฮดเดอร์ ก็น่าจะอยู่ที่ `C:\mingw64\x86_64-w64-mingw32\include` ซึ่งถ้าเราลองเปิดเข้าไปดูก็จะพบกับไฟล์จำนวนมาก ในลักษณะดังนี้



การประกาศเฮดเดอร์ ให้เขียนไว้ส่วนหัวของไฟล์ก่อนจะเริ่มต้นโค้ดหลัก เพื่อให้มันถูกประมวลผลก่อนส่วนโค้ดที่เราเขียนในลำดับถัดมา ซึ่งตามปกติ ในการเขียนโค้ดที่ซับซ้อนขึ้น เราอาจต้องอ้างถึงเฮดเดอร์มากกว่า 1 อัน โดยใช้หลักการเดิมทั้งหมด เช่น

```
#include <stdio.h>
#include <math.h>
#include <string.h>
```

ฟังก์ชัน main

ในภาษา C (รวมถึงภาษาคอมไพเลอร์โดยส่วนใหญ่) จะใช้ฟังก์ชันที่ชื่อ `main` เพื่อเป็นจุดเริ่มต้นในการทำงาน ซึ่งความจริงนั้น นอกจากคำว่า `main` แล้วอาจมีองค์ประกอบอื่นๆ ที่เราสามารถเขียนเพิ่มเติมร่วมกับคำว่า `main` ได้ แต่ในเบื้องต้นนี้ จะแนะนำเพียงแบบเดียวไปก่อน นั่นคือ

```
void main() {

}
```

- **void** เป็นคำสงวนในภาษา C ซึ่งใช้เพื่อบ่งชี้ว่าฟังก์ชันนี้ จะไม่มีการส่งข้อมูลใดๆ กลับออกไปจากฟังก์ชัน
- **main** เป็นชื่อฟังก์ชันหลักของโค้ดโปรแกรมนั้น ซึ่งเป็นส่วนแรกที่จะถูกเรียกขึ้นมาทำงาน

การกำหนดบล็อกคำสั่งด้วยวงเล็บ {}

การกระทำหรือดำเนินการบางอย่าง อาจประกอบด้วยคำสั่งย่อยๆ มากกว่า 1 คำสั่ง และเพื่อให้เราทราบขอบเขตว่าการกระทำนั้น ประกอบด้วยคำสั่งย่อยๆ ใดบ้าง ในภาษา C จะใช้วงเล็บ {} เป็นตัวกำหนดขอบเขต โดยนำคำสั่งย่อยๆ ที่จะดำเนินการต่อเนื่องกันไปตามลำดับมาเขียนไว้ในวงเล็บ {} ดังรูปแบบต่อไปนี้

```
{
    คำสั่งที่ 1
    คำสั่งที่ 2
    คำสั่งที่ 3
    ...
}
```

แต่ละคำสั่งภายในบล็อก ควรเยื้องเข้าไปเป็นระยะ 1 แท็บ โดยให้จุดเริ่มต้นของแต่ละบรรทัดอยู่ในแนวเดียวกัน แม้จะไม่ใช่ว่าบังคับ แต่ก็ยังเป็นสิ่งที่นิยมทำกันทั่วไป เพราะช่วยให้โค้ดนั้นอ่านง่ายขึ้น ซึ่งจากที่ผ่านมามาดูจะเห็นว่า เราได้กำหนดวงเล็บ {} ต่อท้ายฟังก์ชัน main เพื่อกำหนดคำสั่งย่อยๆ ที่จะดำเนินการ เช่น

```
void main() {
    กำหนดคำสั่งย่อยๆ ไว้ที่นี่
}
```

การเขียนวงเล็บเปิด { เพื่อเริ่มต้นบล็อกคำสั่งนั้น เราอาจวางไว้ที่บรรทัดถัดไป หรือจะเขียนต่อจากท้ายบล็อกในบรรทัดเดียวกันก็ได้ ดังวิธีการต่อไปนี้

- **วิธีที่ 1** New Line Block โดยเขียนวงเล็บเปิด { เพื่อเริ่มต้นบล็อกคำสั่ง ที่บรรทัดถัดไป เช่น

```
void main()
{
    คำสั่งย่อยๆ
    ...
}
```

- **วิธีที่ 2** In-Line Block โดยเขียนวงเล็บเปิด { ต่อท้ายบล็อกในบรรทัดเดียวกัน เช่น

```
void main() {
    คำสั่งย่อยๆ
    ...
}
```

การเขียนโค้ดภาษา C ในยุคแรกๆ จะนิยมใช้แบบที่ 1 เป็นส่วนใหญ่ เพราะอ่านโค้ดได้ง่ายกว่า แต่ในยุคหลังๆ ที่การเขียนโค้ดจะซับซ้อนและยืดยาวมากขึ้นเรื่อยๆ เราจึงนิยมใช้แบบที่ 2 มากกว่า เพราะใช้พื้นที่ในการเขียนโค้ดน้อยกว่า จึงทำให้โค้ดดูสั้นลง แต่อย่างไรก็ตาม ไม่มีกฎเกณฑ์หรือข้อบังคับว่าเราต้องใช้แบบใด ทั้งนี้ผู้อ่านสามารถใช้แบบใดก็ได้ที่ตนถนัด และสำหรับในหนังสือเล่มนี้ จะเลือกใช้แบบที่ 2 เป็นหลัก ซึ่งเป็นรูปแบบที่พบเห็นในการใช้งานจริงได้มากกว่า

การกำหนดจุดสิ้นสุดคำสั่งด้วยเครื่องหมาย ;

คำสั่งต่างๆ ในภาษา C ที่เราเขียน มักใช้เพื่อทำสิ่งใดสิ่งหนึ่งโดยเฉพาะ ดังนั้น เราอาจต้องนำคำสั่งย่อยๆ หลายอันมาใช้งานร่วมกัน เพื่อดำเนินการอย่างใดอย่างหนึ่ง ด้วยเหตุนี้จึงต้องมีสิ่งที่ใช้ระบุขอบเขตหรือจุดสิ้นสุดในแต่ละคำสั่ง โดยในภาษา C จะใช้เครื่องหมาย Semicolon (;) เพื่อกำหนดจุดสิ้นสุดของแต่ละคำสั่ง โดยวางเครื่องหมายนี้ต่อท้ายคำสั่ง เช่น

```
void main() {
    คำสั่งที่ 1;      <= เขียนเครื่องหมาย ; ปิดท้าย เพื่อป้องกัน ล้นสุดคำสั่งที่ 1
    คำสั่งที่ 2;      <= เขียนเครื่องหมาย ; ปิดท้าย เพื่อป้องกัน ล้นสุดคำสั่งที่ 2
    คำสั่งที่ 3;      <= เขียนเครื่องหมาย ; ปิดท้าย เพื่อป้องกัน ล้นสุดคำสั่งที่ 3
    ...
}
```

และจากการที่เราระบุเครื่องหมาย ; ต่อท้าย เพื่อระบุจุดสิ้นสุดคำสั่ง ดังนั้น จึงสามารถนำคำสั่งถัดไป มาเขียนต่อเนื่องในบรรทัดเดียวกันได้ เพราะมีเครื่องหมาย ; เป็นตัวแบ่งหรือบ่งชี้จุดสิ้นสุดคำสั่งอยู่แล้ว เช่น

```
void main() {
    คำสั่งที่ 1; คำสั่งที่ 2; คำสั่งที่ 3;
    ...
}
```

การเขียนคำอธิบายโค้ด (Comment)

เราสามารถเขียนข้อความแทรกลงในโค้ด เพื่ออธิบายเกี่ยวกับโค้ดส่วนนั้นพอสังเขป หรือจะเขียนข้อความในวัตถุประสงค์อื่นๆ ทั้งนี้ เมื่อเราย้อนกลับมาอ่านโค้ดในภายหลัง หรือเมื่อส่งโค้ดให้กับคนอื่นๆ ต่อไป ก็จะสามารถเข้าใจโค้ดส่วนนั้นได้เร็วขึ้น ซึ่งวิธีการเขียนคำอธิบายโค้ดในภาษา C นั้น อาจทำแบบใดแบบหนึ่ง ดังนี้

การเขียนคำอธิบายแบบบล็อก (Block Comment)

วิธีนี้ จะกำหนดจุดเริ่มต้นและสิ้นสุดบล็อกด้วยสัญลักษณ์ `/* */` ตามลำดับ โดยนับตั้งแต่สัญลักษณ์ `/*` เป็นต้นไป จะถือเป็นคำอธิบายทั้งหมด จนกว่าจะเจอสัญลักษณ์ `*/` จึงจะถือว่าสิ้นสุดคำอธิบาย ดังนั้น เราจึงสามารถเขียนคำอธิบายหลายบรรทัดไว้ในบล็อกเดียวกันได้ เช่น

```
/* example2-0.c */
/* C Programming Language
   Written by Dennis Ritchie
*/

void main() {                                /* main program */
    puts("Hello World");                     /* display message */

    /* puts("Goodbye"); */                  /* โค้ดบรรทัดนี้ไม่ถูกประมวลผล
                                           แม้จะเป็นคำสั่งในภาษา C ก็ตาม
                                           เพราะเขียนไว้ใน Block Comment
                                           */
}
```

inspiration starts here

การเขียนคำอธิบายแบบบรรทัด (Line Comment)

คำอธิบายแบบนี้ จะใช้สัญลักษณ์ `//` เป็นตัวกำหนดจุดเริ่มต้น และถือว่าจุดสิ้นสุดบรรทัด (ตำแหน่งที่เรากด <Enter> เพื่อขึ้นบรรทัดใหม่) เป็นจุดสิ้นสุดคำอธิบาย นั้นแสดงว่า เราจะเขียนคำอธิบายที่ยาวได้ไม่เกิน 1 บรรทัด และถ้าต้องการเขียนคำอธิบายที่ยาวหลายบรรทัด ต้องกำหนดด้วยสัญลักษณ์ `//` ไปทุกบรรทัด เช่น

```
//example2-0.c
//C Programming Language
//Written by Dennis Ritchie
```

```
void main() {           //main program
    puts("Hello World"); //display message

    //puts("Goodbye");   //โค้ดบรรทัดนี้ไม่ถูกประมวลผล
                        //แม้จะเป็นคำสั่งในภาษา C ก็ตาม
                        //เพราะเขียนตามหลัง Line Comment
}
```

การแสดงความข้อความในเบื้องต้น

ในการเขียนโค้ดภาษา C นั้น ไม่ว่าจะเป็นการทดสอบหรือเพื่อใช้งานจริง ก็อาจมีกรณีที่ต้องแสดงข้อความบางอย่างแก่ผู้ใช้ ซึ่งอาจเป็นข้อความทั่วไป การแสดงผลลัพธ์ คำแจ้งเตือน รวมถึงการแสดงข้อผิดพลาดต่างๆ โดยการแสดงข้อความตามที่กล่าวมา อาจเป็นเพียงการแสดงข้อความแบบง่ายๆ หรืออาจเป็นข้อความที่มีการจัดรูปแบบก็ได้ สำหรับข้อความที่มีการจัดรูปแบบที่ซับซ้อนขึ้นจะกล่าวถึงในบทที่ 4 แต่ในบทนี้ จะแนะนำเพียงการเขียนข้อความพื้นฐานธรรมดาไปก่อน ดังรายละเอียดต่อไปนี้

การแสดงความข้อความด้วยฟังก์ชัน puts()

วิธีการแสดงข้อความแบบที่ง่ายที่สุดในภาษา C น่าจะเป็นการใช้ฟังก์ชัน puts() ในรูปแบบดังนี้

```
puts("ข้อความที่ต้องการแสดง")
```

- ฟังก์ชัน puts() อยู่ในกลุ่มเฮดเดอร์ stdio.h ดังนั้น เราต้องอ้างอิงเฮดเดอร์นี้ด้วย `#include <stdio.h>` เอาไว้ล่วงหน้าเหมือนที่เคยทำมา จึงจะใช้งานฟังก์ชันนี้ได้
- ข้อความที่จะแสดงด้วย puts() ต้องเขียนไว้ในเครื่องหมายคำพูดแบบ " " เท่านั้น ถึงจะเป็นตัวเลขก็ต้องเขียนไว้ในเครื่องหมายนี้เช่นกัน เช่น "123"
- ฟังก์ชัน puts() แสดงได้เพียงข้อความเดียว ถ้าเราจะแสดงหลายข้อความก็ต้องกำหนดแต่ละข้อความด้วยฟังก์ชัน puts() แยกกัน หรือใช้ฟังก์ชัน puts() หลายครั้งนั่นเอง
- ฟังก์ชัน puts() เป็นการแสดงข้อความ 1 บรรทัด ดังนั้น เมื่อเราใช้ฟังก์ชัน puts() เพื่อแสดงข้อความครั้งถัดไป มันจะถูกนำไปแสดงหรือขึ้นบรรทัดใหม่โดยอัตโนมัติ

- ข้อความที่อยู่ในเครื่องหมาย " " อันเดียวกัน ต้องเขียนให้จบในบรรทัดเดียวกัน จะแยกบางส่วนไปขึ้นบรรทัดใหม่ไม่ได้ (ความจริงมีวิธีที่ทำได้ แต่จะมีปัญหาการเว้นระยะห่าง) ถ้าข้อความนั้นยาวเกินไป ไม่สามารถเขียนจบในบรรทัดเดียวกัน วิธีแก้ปัญหานั้นเบื้องต้น คือ ให้แยกเป็นข้อความย่อยๆ แล้วกำหนดไว้ในเครื่องหมาย " " คนละอัน

สำหรับแนวทางการใช้งาน ให้เราดูจากตัวอย่างต่อไปนี้ ซึ่งวิธีการสร้างไฟล์ตัวอย่างในภาษา C ได้กล่าวเอาไว้แล้ว

ตัวอย่าง 2-1 การแสดงข้อความอย่างง่ายด้วยฟังก์ชัน puts()

```
#include <stdio.h>

void main() {
    puts("C is a real-world, widely available and ...");
    puts("C is a small, efficient, powerful, and ...");
    puts("C has been standardized, portable and ...");
}
```

```
C:\c-lang\chapter2>cd "c:\c-lang\chapter2\" && gcc example2-1.c -o example2-1
er2\example2-1
C is a real-world, widely available and popular language.
C is a small, efficient, powerful, and flexible language
C has been standardized, portable and close to hardware
C:\c-lang\chapter2>|
```

การแสดงผลข้อความด้วยฟังก์ชัน printf()

ตามปกตินั้น เรามักใช้ printf() กับข้อความที่มีการจัดรูปแบบบางอย่าง โดยตัว f ที่เขียนต่อท้ายชื่อฟังก์ชัน มาจากคำว่า format ที่หมายถึงการจัดรูปแบบนั่นเอง แต่อย่างไรก็ตาม ในเบื้องต้นนี้ เราจะใช้ฟังก์ชัน printf() เพื่อแสดงข้อความธรรมดาที่ยังไม่มีการจัดรูปแบบใดๆ เช่นเดียวกับ puts() โดยข้อแตกต่างที่สำคัญในเบื้องต้นของ 2 ฟังก์ชันนี้คือ

- ฟังก์ชัน puts() จะแสดงข้อความโดยขึ้นบรรทัดใหม่โดยอัตโนมัติ
- ฟังก์ชัน printf() หากแสดงข้อความแบบธรรมดา จะแสดงที่บรรทัดเดิมต่อเนื่องไปเรื่อยๆ

สำหรับแนวทางการใช้ฟังก์ชัน printf() ในเบื้องต้น ให้ดูจากตัวอย่างต่อไปนี้

ตัวอย่าง 2-2 การแสดงข้อความอย่างง่ายด้วยฟังก์ชัน printf()

```
#include <stdio.h>
```

```
void main() {  
    printf("C is a real-world, widely available and ...");  
    printf("C is a small, efficient, powerful, and ...");  
    printf("C has been standardized, portable and ...");  
}
```

```
C:\c-lang\chapter2>cd "c:\c-lang\chapter2\" && gcc example2-2.c -o example2-2 && "  
c:\c-lang\chapter2\example2-2  
C is a real-world, widely available and popular language.C is a small, efficient,  
powerful, and flexible language.C has been standardized, portable and close to har  
dware.  
C:\c-lang\chapter2>
```

จากผลลัพธ์จะเห็นได้ว่า printf() จะเขียนข้อความต่อท้ายข้อความเดิมในบรรทัดเดียวกันไปเรื่อยๆ ทั้งนี้ สำหรับวิธีการแก้ไขให้คำสั่ง printf() แสดงข้อความที่บรรทัดใหม่ จะกล่าวถึงในหัวข้อถัดไป

ตัวอย่าง 2-3 การแสดงข้อความด้วยฟังก์ชัน printf()

```
#include <stdio.h>
```

```
void main() {  
    printf("123");  
    printf(" + ");  
    printf("456");  
    printf(" = ");  
    printf("579");  
}
```

```
C:\c-lang\chapter2>cd "c:\c-lang\chapter2\" && gcc example2-3.c  
c:\c-lang\chapter2\example2-3  
123 + 456 = 579  
C:\c-lang\chapter2>
```

อักขระพิเศษสำหรับขึ้นบรรทัดใหม่ (\n)

ถ้าเรามีข้อความย่อยๆ หลายอันที่จะแสดงผล ก็ต้องใช้ฟังก์ชัน puts() หรือ printf() ซ้ำๆ ตามจำนวนข้อความจนครบ ดังตัวอย่างที่ผ่านๆ มานั่นเอง ซึ่งการเรียกฟังก์ชันเดิมซ้ำๆ แบบนี้ถือเป็นข้อบกพร่องที่เราควรแก้ไข โดยวิธีการที่จะแนะนำในเบื้องต้นคือ การแทรกอักขระพิเศษลงในข้อความเพื่อให้เกิดการขึ้นบรรทัดใหม่ โดยไม่จำเป็นต้องใช้ puts() หรือ printf() ซ้ำหลายครั้ง ดังหลักการต่อไปนี้

- ให้แทรกอักขระพิเศษคือ \n (หมายถึง New Line) ลงในข้อความ ณ ตำแหน่งที่ต้องการขึ้นบรรทัดใหม่
- ในข้อความเดียวกัน (ในเครื่องหมาย " " เดียวกัน) จะแทรก \n ลงไปกี่ครั้งก็ได้ โดยจะเกิดการขึ้นบรรทัดใหม่ทุกครั้งในตำแหน่งที่เป็นอักขระ \n
- ความจริงแล้ว อักขระพิเศษที่ขึ้นต้นด้วย \ ยังมีอีกหลายอัน โดยเราเรียกอักขระพิเศษในกลุ่มนี้ว่า Escape Sequence ซึ่งจะกล่าวรายละเอียดเพิ่มเติมในบทที่ 3

เนื่องจากการแทรกอักขระ \n ลงในข้อความ เป็นเพียงเรื่องง่ายๆ จึงขอให้อ่านแนวทางการใช้งานจากตัวอย่างต่อไปนี้ได้เลย

ตัวอย่าง 2-4 การแทรกอักขระพิเศษ \n ลงในข้อความเพื่อขึ้นบรรทัดใหม่

```
#include <stdio.h>

void main() {
    printf("one\n");
    printf("two\n");
    printf("three\n");
    printf("four\n");

    printf("five\nsix\nseven\n");

    puts("\neight");
    puts("\nnine");
    puts("ten\n");
}
```



ตัวอย่าง 2-5 การแทรกอักขระพิเศษ \n ลงในข้อความเดียวกัน (อยู่ในเครื่องหมาย " " คู่เดียวกัน) เพื่อขึ้นบรรทัดใหม่

```
#include <stdio.h>

void main() {
    puts("\n1989: ANSI C\n1990: ISO C\n1999: C99");
    printf("\n2011: C11\n2017: C17\n2023: C23\n");
}
```

```
C:\c-lang\chapter2>cd "c:\c-lang\chapter2\" && gcc example4-1.c -o
example4-1 && "c:\c-lang\chapter2\"example4-1
```

```
1989: ANSI C
1990: ISO C
1999: C99
```

```
2011: C11
2017: C17
2023: C23
```

```
C:\c-lang\chapter2>█
```

สิ่งที่เราได้เรียนรู้ในบทนี้ ถึงจะเป็นเพียงพื้นฐานของภาษา C ในเบื้องต้น แต่มันคือสิ่งสำคัญที่เราต้องใช้งานกันอยู่ตลอด โดยการเขียนโค้ดในบทต่อๆ ไป เราจำเป็นต้องนำสิ่งที่ได้กล่าวถึงในบทนี้ ไปใช้งานร่วมกันแทบทุกกรณี

SE-ED

inspiration starts here



การเขียนโปรแกรม ภาษา C ขั้นพื้นฐาน

ภาษา C เป็นหนึ่งในภาษาคอมพิวเตอร์ที่มีผู้ใช้งานเป็นจำนวนมากในปัจจุบัน เนื่องจากลักษณะโครงสร้างที่ไม่ซับซ้อนจนเกินไป จึงเรียนรู้ได้ไม่ยาก รวมทั้งความเร็วในการประมวลผล และความสามารถที่โดดเด่นในอีกหลาย ๆ ด้าน ดังนั้น สถาบันการศึกษาส่วนใหญ่จึงมักเลือกภาษา C ในการเรียนการสอนสำหรับวิชาการเขียนโปรแกรมระดับเริ่มต้น และแม้ว่าภาษา C จะถูกสร้างมานานมากแล้ว แต่ซอฟต์แวร์หลัก ๆ ที่เราใช้งานกันอยู่ทุกวันนี้ ล้วนถูกสร้างจากพื้นฐานของภาษา C แทบทั้งสิ้น เช่น ระบบปฏิบัติการต่าง ๆ โปรแกรมด้านฐานข้อมูลและธุรกิจอุตสาหกรรม การคำนวณทางวิทยาศาสตร์ Game Engine และอื่น ๆ อีกมากมาย ดังนั้น การเขียนโปรแกรมด้วยภาษา C จึงเป็นสิ่งสำคัญอย่างหนึ่งที่ผู้ต้องการก้าวสู่สายงานด้านการพัฒนาซอฟต์แวร์ระดับมืออาชีพควรต้องเรียนรู้เอาไว้

เนื้อหาในหนังสือประกอบด้วย :

- | | |
|--|---|
| บทที่ 1 : การจัดเตรียมเครื่องมือ | บทที่ 9 : การสร้างและใช้งานฟังก์ชัน |
| บทที่ 2 : การเขียนโค้ดภาษา C ในเบื้องต้น | บทที่ 10 : อาร์เรย์และสตริง (เพิ่มเติม) |
| บทที่ 3 : ชนิดข้อมูลและตัวแปร | บทที่ 11 : การชี้ตำแหน่งด้วยพอยน์เตอร์ |
| บทที่ 4 : สตริง การรับและแสดงผลข้อมูล | บทที่ 12 : รวมตัวอย่างโค้ดเพิ่มเติม ชุดที่ 2 |
| บทที่ 5 : ตัวเลขและการคำนวณ | บทที่ 13 : สตริงเจอร์ยูเนียน และเอ็นเอ็ม |
| บทที่ 6 : การเปรียบเทียบและกำหนดเงื่อนไข | บทที่ 14 : การเขียนและอ่านไฟล์ |
| บทที่ 7 : การทำซ้ำแบบวนรอบ | บทที่ 15 : พร็อพริเตสเซอร์และมาโคร |
| บทที่ 8 : รวมตัวอย่างโค้ดเพิ่มเติม ชุดที่ 1 | บทที่ 16 : รวมตัวอย่างโค้ดเพิ่มเติม ชุดที่ 3 |



www.se-ed.com



sbc.fans



SE-ED Publisher

พร้อมจำหน่ายในรูปแบบ

- | | |
|--|---|
| ☒ e-book (PDF) | ☐ audiobooks |
| ☐ e-book (EPUB) | ☐ audio CD / MP3 |
| <hr/> | |
| ☒ ปกอ่อน | ☐ LARGE PRINT (ตัวอักษรขนาดใหญ่) |

ISBN 978-616-08-4751-8



9 786160 847518
239 บาท