



คู่มือ Coding ภาษา C

ฉบับสมบูรณ์ 3rd EDITION

- อ่านเข้าใจง่าย มีแบบฝึกหัดและตัวอย่างพร้อมคำอธิบายทุกหัวข้อ
- มีตัวอย่างการพัฒนาโปรแกรมด้วยภาษา C
- แนะนำการเขียนโปรแกรมเกี่ยวกับ Arduino เบื้องต้น
- เรียนรู้จากอาจารย์ที่มีประสบการณ์สอนมากกว่า 20 ปี
- เหมาะสำหรับนักเรียน นักศึกษา และผู้ที่สนใจเขียนโปรแกรมภาษา C





มีเพียง “ความรู้” เท่านั้นที่มนุษย์ใช้พลิก “โลก”
และเปลี่ยน “ชีวิต” เราจึงสร้างสรรค์ และส่งมอบ “ความรู้”
ในรูปแบบที่ดีกว่า เพื่อให้คนไทย “เรียนรู้” ได้ตลอดชีวิต

Only “Knowledge” can help human
change “The World” and “Their Lives”.
With this truth, it drives us to deliver
“Knowledge” for Thai being able to
“Learn” better everyday.



Think
Beyond



คู่มือ Coding ภาษา C

ฉบับสมบูรณ์ (3rd Edition)

Authors

ผศ.สุดา เขียวมนตรี, ไกรศร ตั้งธนาโอภากุล, กิตินันท์ พลสวัสดิ์
thiansuda@gmail.com

Editorial

กิตินันท์ พลสวัสดิ์
kitinan_p@idcpremier.com

Graphic Designer

ชวรินทร์ รัตนะ

Page Layout

ศรัณย์ คมขำ

Proofreader

สุนทรี บรรลือศักดิ์

Publishing Coordinators

วรพล ธนิกุล, สุพัตรา อาจปรุ, วัชรพงศ์ ยงปัญญาสกุล

Visual Studio เป็นเครื่องหมายการค้าของบริษัท Microsoft และเครื่องหมายการค้าอื่นๆ ที่อ้างถึงเป็นของบริษัทนั้นๆ

สงวนลิขสิทธิ์ตามพระราชบัญญัติลิขสิทธิ์ พ.ศ. 2537 โดยบริษัท ไอดีซี พรีเมียร์ จำกัด ห้ามลอกเลียนไม่ว่าส่วนใดส่วนหนึ่งของหนังสือเล่มนี้ ไม่ว่าในรูปแบบใดๆ นอกจากจะได้รับอนุญาตเป็นลายลักษณ์อักษรจากผู้จัดพิมพ์เท่านั้น

บริษัท ไอดีซี พรีเมียร์ จำกัด จัดตั้งขึ้นเพื่อเผยแพร่ความรู้ที่มีคุณภาพสู่ผู้อ่านชาวไทย เรายินดีรับงานเขียนของนักวิชาการและนักเขียนทุกท่าน ท่านผู้สนใจกรุณาติดต่อผ่านทางอีเมลที่ infopress@idcpremier.com หรือทางโทรศัพท์หมายเลข 0-2962-1081 (อัตรานาที 10 คู่มือ) โทรสาร 0-2962-1084

PUBLISHED AND DISTRIBUTED BY



พิมพ์ครั้งที่ 1 กรกฎาคม 2564

บริษัท ไอดีซี พรีเมียร์ จำกัด

200 หมู่ 4 ชั้น 19 ห้อง 1901 อาคารจัสมินอินเตอร์เนชั่นแนล

ทาวเวอร์ ถ.แจ้งวัฒนะ อ.ปากเกร็ด จ.นนทบุรี 11120

โทรศัพท์ 0-2962-1081 (อัตรานาที 10 คู่มือ)

โทรสาร 0-2962-1084

สมาชิกสัมพันธ์

โทรศัพท์ 0-2962-1081-3 ต่อ 121

โทรสาร 0-2962-1084

ร้านค้าและตัวแทนจำหน่าย

โทรศัพท์ 0-2962-1081-3 ต่อ 112-114

โทรสาร 0-2962-1084

ข้อมูลทางบรรณานุกรม

ผศ.สุดา เขียวมนตรี

คู่มือ Coding ภาษา C ฉบับสมบูรณ์ (3rd Edition)

นนทบุรี : ไอดีซี, 2564

416 หน้า

1. การเขียนโปรแกรมโดยใช้ภาษาโปรแกรมเฉพาะชนิด

I ไกรศร ตั้งธนาโอภากุล

II กิตินันท์ พลสวัสดิ์ (ผู้แต่งร่วม)

III ชื่อเรื่อง

005.262

ISBN 885-916-100-908-5

ราคา 330 บาท

คำนำ

หนังสือ คู่มือ Coding ภาษา C ฉบับสมบูรณ์ (3rd Edition) เล่มนี้ได้นำเสนอเนื้อหาสำหรับผู้คนที่ต้องการเรียนรู้การเขียนโปรแกรมภาษา C ตั้งแต่ระดับเริ่มต้นไปถึงการเขียนโปรแกรมเพื่อประยุกต์ใช้งาน ผู้เขียนได้นำเสนอความรู้พื้นฐานและหลักการเขียนโปรแกรมอย่างเป็นลำดับขั้นตอน พร้อมทั้งอธิบายตัวอย่างโปรแกรมไว้อย่างครบถ้วน ซึ่งผู้อ่านสามารถศึกษาเรียนรู้และฝึกเขียนโปรแกรมด้วยตนเองได้โดยง่าย ผู้เขียนได้อธิบายและนำเสนอรูปแบบการเขียนโปรแกรมในแบบต่างๆ เริ่มต้นจากการเขียนโปรแกรมเชิงโครงสร้าง การเขียนฟังก์ชันส่วนการทำงานย่อยของโปรแกรม รวมไปถึงการเขียนโปรแกรมประยุกต์ใช้งาน เช่น โปรแกรมจัดการข้อมูลนักศึกษา และโปรแกรมควบคุมการทำงานของวงจรมicrocontroller Arduino โดยผู้เขียนได้นำเสนอเนื้อหาพร้อมตัวอย่างโปรแกรม ที่สามารถนำไปประยุกต์เพื่อการต่อยอดได้ไว้อย่างครบถ้วน ผู้เขียนมีความตั้งใจที่จะให้หนังสือเล่มนี้เป็นจุดเริ่มต้นสำหรับผู้อ่าน ที่เริ่มต้นศึกษาการเขียนโปรแกรมเพื่อเป็นการต่อยอดสำหรับการเรียนรู้ในภาษาโปรแกรมอื่นต่อไป และเป็นแนวทางในการเขียนโปรแกรมประยุกต์เพื่อการก้าวสู่การเป็นนักเขียนโปรแกรมมืออาชีพ

ท้ายที่สุดนี้ ผู้เขียนขอขอบคุณบรรณาธิการ ทีมงาน DEV BOOK และผู้สนับสนุนทุกท่านที่มีส่วนร่วมในการจัดทำหนังสือเล่มนี้ให้สำเร็จได้เป็นอย่างดี

ผศ.สุดา เจริญมนตรี

thiansuda@gmail.com

NOTE

สารบัญ

บทที่ 1	รู้จักภาษา C กับการพัฒนาโปรแกรม	1
	ความเป็นมาของภาษา C	1
	จุดเด่นของภาษา C	3
	โครงสร้างของภาษา C	3
	ความรู้พื้นฐานการพัฒนาซอฟต์แวร์	8
	วงจรชีวิตของการพัฒนาซอฟต์แวร์	
	(Software Development Life Cycle – SDLC)	8
	ระเบียบวิธีการพัฒนาซอฟต์แวร์	10
	สรุปเนื้อหาบทที่ 1	12
บทที่ 2	เตรียมพร้อมก่อนเขียนโปรแกรม	15
	Visual Studio Community 2019 for Windows	15
	ดาวน์โหลด Visual Studio Community 2019	16
	ติดตั้งโปรแกรม Visual Studio Community 2019	17
	เริ่มต้นเขียนโปรแกรมด้วย Visual Studio Community 2019	19
	การบันทึกโปรเจกต์ (Save Project)	27
	การเปิดโปรเจกต์ (Open Project)	27
	การเขียนคำอธิบาย (Comment)	28
	C-Free 5.0 Professional for Windows	29
	ดาวน์โหลด C-Free 5.0 Professional	29
	ติดตั้งโปรแกรม C-Free 5.0 Professional	30
	เริ่มต้นเขียนโปรแกรมด้วย C-Free 5.0 Professional	35
	การบันทึกโปรแกรม	36
	การเปิดใช้งานโปรแกรม	37
	Visual Studio Code for Linux	38
	ดาวน์โหลด Visual Studio Code	39
	ติดตั้งโปรแกรม Visual Studio Code	40



	เริ่มต้นเขียนโปรแกรมด้วย Visual Studio Code	41
	การบันทึกโปรแกรม	44
	การเปิดใช้งานโปรแกรม	45
	สรุปเนื้อหาบทที่ 2	47
บทที่ 3	ตัวแปรและชนิดของข้อมูล	49
	ตัวแปร (Variable)	49
	กฎการตั้งชื่อ	50
	การประกาศตัวแปร	51
	ชนิดของข้อมูล (Data Type)	51
	Integer Type (ชนิดข้อมูลแบบจำนวนเต็ม)	52
	Character Type (ชนิดข้อมูลแบบตัวอักษร)	57
	String Type (ชนิดข้อมูลแบบข้อความ)	59
	Floating Point Type (ชนิดข้อมูลแบบจำนวนเลขทศนิยม)	61
	ค่าคงที่ (Constants)	65
	กฎของการแปลงชนิดของข้อมูล (Data Type Conversion)	67
	Implicit Type Conversion	67
	Explicit Type Conversion (Casting)	70
	รู้จักกับ Null character และ Empty string	72
	สรุปเนื้อหาบทที่ 3	74
บทที่ 4	นิพจน์และตัวดำเนินการ (Expressions and Operator)	75
	นิพจน์ (Expression)	75
	ตัวดำเนินการ (Operator)	76
	ตัวดำเนินการกำหนดค่า (Assignment Operators)	76
	ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)	80
	ตัวดำเนินการยูนารี (Unary Operators)	84
	ตัวดำเนินการเปรียบเทียบ (Comparison Operators)	87
	ตัวดำเนินการตรรกะ (Logical Operators)	91
	ตัวดำเนินการแบบมีเงื่อนไข (Conditional Operators)	93
	ตัวดำเนินการบอกขนาดชนิดข้อมูล (sizeof Operators)	95
	ตัวดำเนินการระดับบิต (Bitwise Operators)	96

Bitwise Shift Left (<<).....	101
Bitwise Shift Right (>>).....	103
One's Complement (~)	106
ลำดับความสำคัญของตัวดำเนินการ (Operator of Precedence).....	109
สรุปเนื้อหาบทที่ 4.....	112

บทที่ 5 การแสดงผลและการรับข้อมูล 113

การแสดงผลข้อมูล	113
การแสดงผลทีละตัวอักษรด้วยคำสั่ง putchar()	113
การแสดงผลเป็นข้อความด้วยคำสั่ง puts()	114
การแสดงผลข้อมูลทุกชนิดด้วยคำสั่ง printf().....	116
รหัสรูปแบบการแสดงผลของคำสั่ง printf()	117
การจัดรูปแบบการแสดงผลข้อมูลของคำสั่ง printf().....	120
การกำหนดการแสดงผลข้อมูลของคำสั่ง printf().....	122
การจัดการพื้นที่แสดงผลข้อมูลของคำสั่ง printf().....	124
การรับข้อมูล	127
การรับข้อมูลทีละตัวอักษรด้วยคำสั่ง getch() และ getchar().....	127
การรับข้อมูลชนิดข้อความด้วยคำสั่ง gets_s()	130
การรับข้อมูลทุกชนิดด้วยคำสั่ง scanf() หรือ scanf_s()	132
การกำหนดลำดับการรับข้อมูลของคำสั่ง scanf() หรือคำสั่ง scanf_s()	136
การกำหนดจำนวนตัวอักษรในการรับค่าข้อมูลชนิดข้อความ ของคำสั่ง scanf() หรือคำสั่ง scanf_s().....	138
การกำหนดรูปแบบการรับค่าข้อมูลของคำสั่ง scanf() หรือคำสั่ง scanf_s()	139
การนับจำนวนตัวอักษรที่รับค่าข้อมูลด้วยคำสั่ง scanf() หรือคำสั่ง scanf_s()	141
สรุปเนื้อหาบทที่ 5.....	142

บทที่ 6 ควบคุมทิศทางการทำงานของโปรแกรมด้วยคำสั่งตัดสินใจ..... 145

คำสั่ง if - คำสั่งตัดสินใจแบบหนึ่งทางเลือก	145
คำสั่ง if...else - คำสั่งตัดสินใจแบบสองทางเลือก	148
คำสั่ง if...else if - คำสั่งตัดสินใจแบบหลายทางเลือก.....	151
คำสั่ง switch-case - คำสั่งตัดสินใจแบบหลายทางเลือก	155
สรุปเนื้อหาบทที่ 6.....	160



บทที่ 7	ควบคุมการทำงานของโปรแกรมด้วยคำสั่งวนลูป	163
	คำสั่ง for – คำสั่งวนซ้ำด้วยจำนวนรอบที่แน่นอน	163
	คำสั่ง while – คำสั่งวนซ้ำด้วยจำนวนรอบที่ไม่แน่นอน	166
	คำสั่ง do...while – คำสั่งวนซ้ำโดยตรวจสอบเงื่อนไขหลังการทำงาน	169
	สรุปเนื้อหาบทที่ 7	176
บทที่ 8	การเขียน Flowchart	177
	หลักการเขียน Flowchart	177
	แนวคิดการเขียน Flowchart	179
	การทำงานแบบตามลำดับ (Sequence)	179
	การทำงานแบบใช้คำสั่งตัดสินใจ (Decision)	182
	การทำงานแบบใช้คำสั่งวนลูป (Loop)	188
	สรุปเนื้อหาบทที่ 8	192
บทที่ 9	ตัวแปรอาร์เรย์	193
	ตัวแปรอาร์เรย์ (Array Variable)	193
	ตัวแปรอาร์เรย์ 1 มิติ (One Dimensional Array)	193
	ตัวแปรอาร์เรย์หลายมิติ (Multidimensional Array)	194
	การเขียนโปรแกรมกับตัวแปรอาร์เรย์	196
	การประกาศตัวแปรอาร์เรย์	196
	การกำหนดค่าให้ตัวแปรอาร์เรย์	197
	การอ้างถึงข้อมูลในตัวแปรอาร์เรย์	198
	สรุปเนื้อหาบทที่ 9	206
บทที่ 10	พอยน์เตอร์	207
	ตัวแปรพอยน์เตอร์ (Pointer Variable)	207
	การใช้งานพอยน์เตอร์	209
	การดำเนินการกับพอยน์เตอร์	213
	การใช้งานพอยน์เตอร์กับอาร์เรย์	215
	การจัดการพื้นที่หน่วยความจำแบบไดนามิก (Dynamic Memory Allocation)	222
	รู้จักกับ Pointers to functions	229
	สรุปเนื้อหาบทที่ 10	231

บทที่ 11 ฟังก์ชัน (Function) 233

รู้จักกับฟังก์ชัน พารามิเตอร์ และอาร์กิวเมนต์ 233

ฟังก์ชันที่สร้างขึ้นเอง (User-Defined Function)..... 235

ฟังก์ชันที่ไม่มีการส่งค่ากลับและไม่มีการรับค่าพารามิเตอร์

(void function with no parameter) 236

ฟังก์ชันที่ไม่มีการส่งค่ากลับแต่มีการรับค่าพารามิเตอร์

(void function with parameter)..... 239

ฟังก์ชันที่มีการส่งค่ากลับแต่ไม่มีการรับค่าพารามิเตอร์

(return value function with no parameter) 243

ฟังก์ชันที่มีการส่งค่ากลับและมีการรับค่าพารามิเตอร์

(return value function with parameter) 245

การส่งค่าผ่านพารามิเตอร์ (Parameter Passing)..... 249

ฟังก์ชันประเภทส่งผ่านค่า (Call by Value) 249

ฟังก์ชันประเภทส่งผ่านตัวอ้างอิง (Call by Reference) 251

ขอบเขตการทำงานของตัวแปร 253

ฟังก์ชันเรียกตัวเอง (Recursive Function) 256

ฟังก์ชันที่มาตรฐานภาษา C ได้สร้างมาให้แล้ว (Standard Library Function)..... 258

ไลบรารีฟังก์ชันการคำนวณทางคณิตศาสตร์ 258

ไลบรารีฟังก์ชันสำหรับตัวอักษร..... 259

ไลบรารีฟังก์ชันสำหรับข้อความ 260

สรุปเนื้อหาบทที่ 11 262

บทที่ 12 ตัวแปรประเภทโครงสร้างข้อมูล..... 263

โครงสร้างข้อมูลแบบ struct..... 263

การประกาศโครงสร้างข้อมูลแบบ struct 264

การประกาศโครงสร้างข้อมูลแบบ struct ด้วยคำสั่ง typedef 266

การใช้งานตัวแปร struct..... 268

การกำหนดค่าข้อมูลให้กับ struct..... 269

การอ่านค่าข้อมูลจาก struct..... 270

อาร์เรย์กับโครงสร้างข้อมูล 273

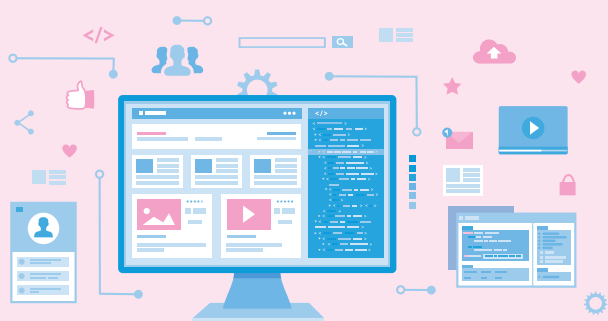
พอยน์เตอร์กับโครงสร้างข้อมูล 277

โครงสร้างข้อมูลซ้อนโครงสร้างข้อมูล 279



การใช้งานโครงสร้างข้อมูลกับฟังก์ชัน.....	282
การใช้งานโครงสร้างข้อมูลกับฟังก์ชันที่มีการส่งค่ากลับแต่ไม่มีการรับค่าพารามิเตอร์.....	282
การใช้งานโครงสร้างข้อมูลกับฟังก์ชันที่ไม่มีการส่งค่ากลับแต่มีการรับค่าพารามิเตอร์.....	286
การใช้งานโครงสร้างข้อมูลกับฟังก์ชันที่มีการส่งค่ากลับและมีการรับค่าพารามิเตอร์.....	289
โครงสร้างข้อมูลแบบยูเนียน (union)	293
การประกาศโครงสร้างข้อมูลแบบ union.....	294
ประเภทข้อมูลแบบ enum (Enumeration Constants).....	298
สรุปเนื้อหาบทที่ 12	300
บทที่ 13 รู้จักและใช้งาน Linked List	301
โครงสร้างข้อมูลแบบเชิงเส้น (Linear Data Structure).....	301
Dense List	302
Linked List	302
สแตติกลิงก์ลิสต์ (Static Linked List).....	302
ไดนามิกลิงก์ลิสต์ (Dynamic Linked List).....	304
เริ่มต้นเขียนโปรแกรมกับ Linked List	305
สรุปเนื้อหาบทที่ 13	314
บทที่ 14 แฟ้มข้อมูล (File)	315
รู้จักกับแฟ้มข้อมูล	315
การเปิดและปิดไฟล์ (File Open/Close)	316
การเปิดไฟล์ (File Open)	316
การปิดไฟล์ (File Close)	319
การทำงานกับ Text File.....	319
การอ่านข้อมูลจากไฟล์ทีละตัวอักษรด้วยฟังก์ชัน getc() และฟังก์ชัน fgetc()	319
การเขียนข้อมูลลงไฟล์ทีละตัวอักษรด้วยฟังก์ชัน putc() และฟังก์ชัน fputc().....	324
การอ่านข้อมูลจากไฟล์ทีละบรรทัดด้วยฟังก์ชัน fgets()	328
การเขียนข้อความลงไฟล์ด้วยฟังก์ชัน fputs().....	331
การอ่านข้อมูลจากไฟล์โดยใช้ฟังก์ชัน fscanf().....	333

การเขียนข้อมูลลงไฟล์ด้วยฟังก์ชัน fprintf()	334
การทำงานกับ Binary File	341
การอ่านข้อมูลจากไฟล์ด้วยฟังก์ชัน fread()	341
การเขียนข้อมูลลงไฟล์ด้วยฟังก์ชัน fwrite()	341
การตรวจสอบสถานะของไฟล์ (File Status Function)	348
การตรวจสอบ EOF (end of file) ด้วยฟังก์ชัน feof()	349
การตรวจสอบ error ที่เกิดขึ้นกับไฟล์ด้วยฟังก์ชัน ferror()	349
การหาตำแหน่งที่อยู่ของไฟล์พอยน์เตอร์ด้วยฟังก์ชัน ftell()	349
การย้ายตำแหน่งไฟล์พอยน์เตอร์ไปยังตำแหน่งต้นไฟล์ด้วยฟังก์ชัน rewind()	350
การย้ายตำแหน่งไฟล์พอยน์เตอร์ด้วยฟังก์ชัน fseek()	352
การดำเนินการเกี่ยวกับไฟล์ (File Operation)	356
การลบไฟล์	356
การเปลี่ยนชื่อไฟล์	357
สรุปเนื้อหาบทที่ 14	359
บทที่ 15 กรณีศึกษา ระบบบันทึกข้อมูลนักศึกษา	361
รายละเอียดระบบบันทึกข้อมูลนักศึกษา	362
ส่วนฟังก์ชันหลักและเมนูต่างๆ	363
ส่วนฟังก์ชันสำหรับเพิ่มข้อมูล	366
ส่วนฟังก์ชันสำหรับแก้ไขข้อมูล	368
ส่วนฟังก์ชันสำหรับลบข้อมูล	373
ส่วนฟังก์ชันสำหรับค้นหาข้อมูล	377
ส่วนฟังก์ชันสำหรับแสดงข้อมูลทั้งหมด	380
สรุปเนื้อหาบทที่ 15	383
บทที่ 16 แนะนำการประยุกต์ใช้ภาษา C กับ Arduino	385
รู้จักกับ Arduino	385
การใช้งานโปรแกรม Tinkercad	387
ขั้นตอนการสมัครสมาชิก	387
โครงสร้างภาษา C Arduino เบื้องต้น	389
สรุปเนื้อหาบทที่ 16	399



บทที่ 1

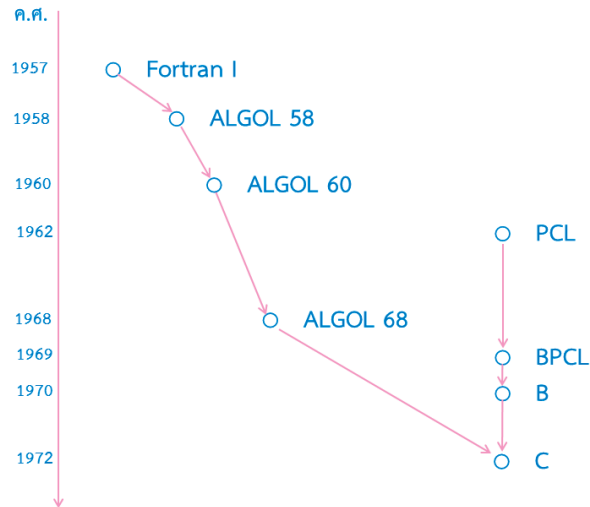
รู้จักภาษา C กับการพัฒนาโปรแกรม

ภาษา C เป็นภาษาระดับสูงที่ถูกสร้างขึ้นมาเพื่อให้การเขียนโปรแกรมง่ายขึ้น ปัจจุบันถือว่าเป็นภาษาคอมพิวเตอร์สำหรับผู้เริ่มต้นที่ต้องการเรียนรู้การเขียนโปรแกรม เนื่องจากเป็นภาษาเชิงโครงสร้างที่มีหลักการทำงานไม่ซับซ้อน เข้าใจง่าย

เนื้อหาในบทนี้ผู้อ่านจะได้ทราบประวัติความเป็นมา, จุดเด่น, โครงสร้างของภาษา C และนอกจากนี้ผู้เขียนจะแนะนำเครื่องมือที่ใช้ในการเขียนโปรแกรมภาษา C ที่ง่ายและสะดวกให้กับผู้อ่านด้วย

ความเป็นมาของภาษา C

ภาษา C เป็นภาษาคอมพิวเตอร์ที่ได้พัฒนาขึ้นในปี ค.ศ. 1972 (พ.ศ. 2515) โดย เดนนิส ริสซี (Dennis Ritchie) แห่ง Bell Telephone Laboratories, Inc. (ในปัจจุบันก็คือ AT&T Bell Laboratories) ซึ่งภาษา C มีการพัฒนามาจากภาษา B ในช่วงแรกภาษา C ได้ถูกนำมาใช้เพื่อสร้างระบบปฏิบัติการ Unix หากนำวิวัฒนาการของภาษา C มาแสดงออกเป็นแผนภาพ จะได้ดังนี้



ในปี ค.ศ. 1978 (พ.ศ. 2521) เดนนิส ริสชี และเบรน เคนนิกัน (Dennis Ritchie and Brian W. Kernighan) ได้แต่งหนังสือชื่อ “The C Programming Language” โดยนำเสนอภาษา C ที่สามารถนำมาปรับใช้กับคอมพิวเตอร์ในรูปแบบต่างๆ ได้ดียิ่งขึ้น และทำให้ภาษา C ได้รับความนิยมอย่างมาก จนกระทั่งในปี ค.ศ. 1988 (พ.ศ. 2531) ได้มีการสร้างมาตรฐานของภาษา C ขึ้นมา ในชื่อของ ANSI C ภายใต้ความร่วมมือระหว่างสถาบัน ANSI (American National Standards Institute) กับเดนนิส ริสชี และเบรน เคนนิกัน



รูปซ้ายมือคือ Mr. Dennis Ritchie, รูปตรงกลาง Mr. Brian W. Kernighan และรูปขวามือคือ หนังสือที่ทั้งคู่ได้แต่งขึ้นร่วมกัน

ในปี ค.ศ. 1990 (พ.ศ. 2533) องค์กรมาตรฐานสากล หรือ ISO (International Standards Organization) ได้ยอมรับมาตรฐานที่ได้สร้างขึ้นมานี้ ภายใต้ชื่อ ANSI/ISO C

จุดเด่นของภาษา C

ในภาษาโปรแกรมทุกภาษามีจุดเด่นในการทำงานของภาษาแตกต่างกัน ในข้อห้วนนี้ผู้อ่านจะได้รู้จักกับจุดเด่นของภาษา C ซึ่งมีดังนี้

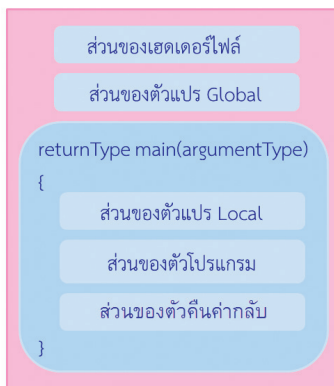
- เป็นภาษาคอมพิวเตอร์ที่มีแนวคิดในการพัฒนาแบบ “โปรแกรมเชิงโครงสร้าง (Structure Programming)” ทำให้ภาษา C เป็นภาษาที่เหมาะสมสำหรับนำมาพัฒนาระบบ
- เป็นภาษาคอมพิวเตอร์ที่เป็นภาษามาตรฐาน ซึ่งการทำงานของภาษาไม่ขึ้นกับฮาร์ดแวร์ ทำให้สามารถนำไปใช้ใน CPU รุ่นต่างๆ ได้
- เป็นภาษาระดับสูงที่ทำงานเหมือนภาษาระดับต่ำ สามารถทำงานแทนภาษา Assembly ได้
- ความสามารถของคอมไพเลอร์ในภาษา C มีประสิทธิภาพสูง ทำงานได้รวดเร็ว โดยใช้รหัสออบเจกต์ (Object) ที่สั้น จึงทำให้เหมาะสำหรับงานที่ต้องการความรวดเร็ว



คอมไพเลอร์ (Compiler) จะทำหน้าที่แปลภาษาโปรแกรมทั้งหมดที่เขียนให้เป็นภาษาเครื่อง พร้อมกันนี้จะทำหน้าที่ตรวจสอบไวยากรณ์ของภาษา หากมีข้อผิดพลาดก็จะส่งข้อความหรือคำเตือนออกมา เมื่อกระบวนการทำงานของคอมไพเลอร์เสร็จสมบูรณ์ โปรแกรมจึงเริ่มทำงาน ซึ่งแตกต่างกับหลักการการทำงานของ**อินเตอร์พรีเตอร์ (Interpreter)** ซึ่งการทำงานจะแปลภาษาโปรแกรมและประมวลผลคำสั่งทีละคำสั่ง

โครงสร้างของภาษา C

ในโปรแกรมที่พัฒนาด้วยภาษา C ทุกโปรแกรมจะมีโครงสร้างการพัฒนาไม่แตกต่างกัน ซึ่งประกอบด้วย 6 ส่วนหลักๆ โดยที่แต่ละส่วนจะมีหน้าที่แตกต่างกันดังนี้





1. **ส่วนของเฮดเดอร์ไฟล์ (header file or processing directive)** ส่วนนี้จะมีจุดสังเกตที่สำคัญคือ จะขึ้นต้นด้วยเครื่องหมาย # เสมอ การทำงานของคอมไพเลอร์จะทำงานในส่วนนี้เป็นส่วนแรก ในส่วนของเฮดเดอร์ไฟล์จะเป็นส่วนที่เก็บไลบรารีมาตรฐานของภาษา C ซึ่งจะถูกดึงเข้ามารวมกับโปรแกรมในขณะที่กำลังคอมไพล์ โดยใช้คำสั่ง **#include** ซึ่งสามารถเขียนได้เป็น 2 รูปแบบคือ

รูปแบบที่ 1

```
#include<HeaderName>
```

รูปแบบที่ 2

```
#include "HeaderName"
```

โดยที่ HeaderName เป็นชื่อ header file

ผู้อ่านคงสงสัยแล้วว่า ทั้งสองแบบนี้มีความแตกต่างกันอย่างไร ทั้งสองแบบนี้ต่างกันที่การทำงานคือ แบบที่ใช้เครื่องหมาย <...> คอมไพเลอร์จะค้นหาเฮดเดอร์ไฟล์จากไลบรารีของภาษา C เพียงที่เดียวเท่านั้น ส่วนที่ใช้เครื่องหมาย “...” คอมไพเลอร์จะค้นหาเฮดเดอร์ไฟล์จากไลบรารีที่เก็บ Source Code ของเราก่อน ถ้าหากไม่เจอก็จะไปค้นหาที่ไลบรารีมาตรฐานของภาษา C และในจุดที่ผู้อ่านควรสังเกตคือ เฮดเดอร์ไฟล์นี้จะมีสกุลไฟล์เป็น .h เท่านั้น

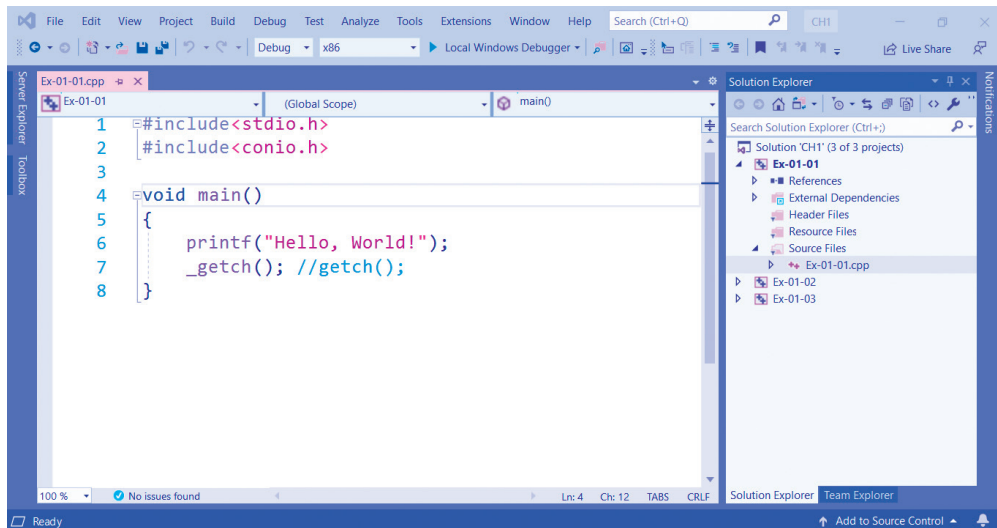
ในการเขียนโปรแกรมภาษา C เฮดเดอร์ไฟล์ที่เก็บไลบรารีมาตรฐานในการจัดการเกี่ยวกับอินพุตและเอาต์พุตของโปรแกรมก็คือ stdio.h ซึ่งถือว่าเป็นส่วนสำคัญที่ขาดไม่ได้

2. **ส่วนของตัวแปร Global** เป็นส่วนประกาศตัวแปรที่สามารถใช้ร่วมกันได้ทั้งโปรแกรม ซึ่งส่วนนี้จะมีหรือไม่มีก็ได้
3. **ส่วนของฟังก์ชัน (Function)** เป็นส่วนการทำงานของโปรแกรม ในโครงสร้างของภาษา C จะบังคับให้มีอย่างน้อย 1 ฟังก์ชันคือ ฟังก์ชัน main() ซึ่งเป็นฟังก์ชันเริ่มการทำงานของโปรแกรม (คอมไพเลอร์จะประมวลผลที่ฟังก์ชัน main() เป็นฟังก์ชันแรก) โดยขอบเขตของฟังก์ชันจะเริ่มต้นด้วยเครื่องหมาย { และสิ้นสุดด้วยเครื่องหมาย }
4. **ส่วนของตัวแปร Local** เป็นส่วนประกาศตัวแปรที่สามารถใช้ได้เฉพาะภายในฟังก์ชันของตนเองเท่านั้น ซึ่งส่วนนี้จะมีหรือไม่มีก็ได้

5. ส่วนของตัวโปรแกรม เป็นส่วนคำสั่งการทำงานของโปรแกรม โดยคำสั่งในแต่ละคำสั่งจะต้องจบด้วยเครื่องหมาย ; เสมอ

6. ส่วนของตัวคืนค่ากลับ เป็นส่วนของการคืนค่าข้อมูลกลับเมื่อฟังก์ชันจบการทำงาน โดยค่าที่คืนกลับนั้นจะต้องเป็นค่าที่มีชนิดของข้อมูลตรงกับชนิดของข้อมูลที่ฟังก์ชันคืนค่ากลับ (return type) ในกรณีไม่ต้องการให้ฟังก์ชันมีการส่งคืนค่ากลับ สามารถกำหนดได้โดยใช้คีย์เวิร์ด void

ตัวอย่างที่ 1.1 โปรแกรมแสดงส่วนต่างๆ ในกรณีประกาศฟังก์ชัน main() ที่ไม่มีการคืนค่าใดๆ (void) ดังรูป



```
1 #include<stdio.h>
2 #include<conio.h>
3
4 void main()
5 {
6     printf("Hello, World!");
7     _getch(); //getch();
8 }
```

ผลการทำงานของโปรแกรมแสดงได้ดังนี้

Hello, World!

อธิบายการทำงานของโปรแกรม

บรรทัดที่ 1-2 เป็นการเรียกใช้งานส่วนของเฮดเดอร์ไฟล์ สังเกตได้ที่เครื่องหมาย # โดยมีการเรียกใช้ไลบรารี **stdio.h** ซึ่งเป็นไลบรารีจัดการเกี่ยวกับอินพุตและเอาต์พุต และไลบรารี **conio.h** ซึ่งเป็นไลบรารีจัดการเกี่ยวกับหน้าจอทั้งหมด

บรรทัดที่ 4 ส่วนของฟังก์ชัน main() โดยประกาศชนิดข้อมูลที่คืนค่ากลับเป็น void และค่าที่รับเข้ามาในฟังก์ชันเป็น void หมายถึง ฟังก์ชันนี้จะไม่มีการคืนค่าใดๆ กลับออกไป และไม่มีการรับค่าใดๆ เข้ามาในฟังก์ชัน

บรรทัดที่ 5 ส่วนเริ่มต้นของฟังก์ชัน main()



- บรรทัดที่ 6 เป็นคำสั่งแสดงผลทางจอภาพ โดยให้พิมพ์คำว่า “Hello, World!”
- บรรทัดที่ 7 เป็นคำสั่งรับอักขระทางแป้นพิมพ์ ในที่นี้เพื่อกำหนดไม่ให้โปรแกรมปิดหน้าต่างผลลัพธ์ เมื่อแสดงผลพร้อมที่ต้องการแล้ว
- บรรทัดที่ 8 ส่วนสิ้นสุดของฟังก์ชัน main()

Note

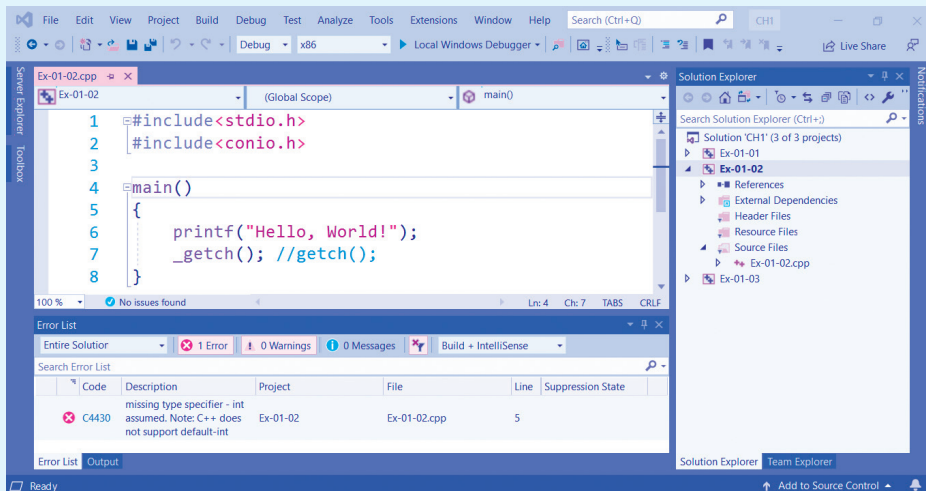
คำสั่งรับอักขระทางแป้นพิมพ์ ภาษา C เวอร์ชัน Microsoft Visual Studio 2019 จะใช้คำสั่ง `_getch` เพื่อให้เป็นไปตามมาตรฐานการอินเตอร์เฟซของ OS POSIX (Portable Operating System Interface)

Note

เมื่อผู้อ่านได้อ่านมาถึงจุดนี้ ผู้อ่านอาจจะไม่เข้าใจความหมายของคำบางคำ เช่น void, return type เป็นต้น แต่ผู้อ่านไม่ต้องเป็นกังวล เนื่องจากส่วนนี้ผู้เขียนมีเป้าหมายให้ผู้อ่านมองเห็นภาพรวมของการเขียนโปรแกรมภาษา C เท่านั้น ในส่วนต่างๆ ผู้เขียนจะอธิบายในบทต่อไป

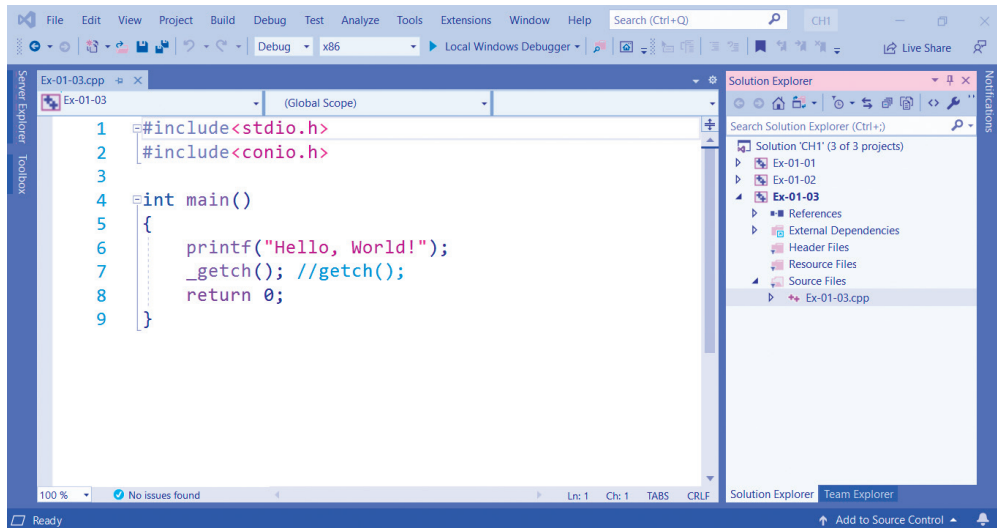
Tip

ในการประกาศฟังก์ชัน main() หรือฟังก์ชันใดๆ ผู้อ่านจะต้องระบุคีย์เวิร์ดในการคืนค่า หากผู้อ่านไม่ต้องการให้ฟังก์ชันมีการคืนค่ากลับ สามารถกำหนดได้โดยใช้คีย์เวิร์ด void ดังตัวอย่างที่นำมา



ตัวอย่างนี้ไม่ได้กำหนดคีย์เวิร์ดในการคืนค่ากลับ เมื่อรันโปรแกรมก็จะมีข้อผิดพลาด (error) เกิดขึ้น

ตัวอย่างที่ 1.2 โปรแกรมแสดงส่วนต่างๆ ในกรณีประกาศฟังก์ชัน main() ที่มีการคืนค่ากลับเป็นชนิดเลขจำนวนเต็ม (int) ดังรูป



```
1 #include<stdio.h>
2 #include<conio.h>
3
4 int main()
5 {
6     printf("Hello, World!");
7     _getch(); //getch();
8     return 0;
9 }
```

ผลการทำงานของโปรแกรมแสดงได้ดังนี้

Hello, World!

อธิบายการทำงานของโปรแกรม

- บรรทัดที่ 1-2 เป็นการเรียกใช้งานส่วนของเฮดเดอร์ไฟล์ สังเกตได้ที่เครื่องหมาย # โดยมีการเรียกใช้ไลบรารี stdio.h ซึ่งเป็นไลบรารีจัดการเกี่ยวกับอินพุตและเอาต์พุต และไลบรารี conio.h ซึ่งเป็นไลบรารีจัดการเกี่ยวกับหน้าจอทั้งหมด
- บรรทัดที่ 4 ส่วนของฟังก์ชัน main() โดยประกาศชนิดข้อมูลที่คืนค่ากลับเป็น int และค่าที่รับเข้ามาในฟังก์ชันเป็น void หมายถึง ฟังก์ชันนี้จะมีการส่งค่าข้อมูลกลับออกไปเป็นจำนวนเต็ม แต่ไม่มีการรับค่าใดๆ เข้ามาในฟังก์ชัน
- บรรทัดที่ 5 ส่วนเริ่มต้นของฟังก์ชัน main()
- บรรทัดที่ 6 เป็นคำสั่งแสดงผลทางจอภาพ โดยให้พิมพ์คำว่า “Hello, World!”
- บรรทัดที่ 7 เป็นคำสั่งรับอักขระทางแป้นพิมพ์ ในที่นี้เพื่อกำหนดไม่ให้โปรแกรมปิดหน้าต่างผลลัพธ์ เมื่อแสดงผลลัพธ์ที่ต้องการแล้ว
- บรรทัดที่ 8 เป็นส่วนของการคืนค่ากลับ ในที่นี้ให้คืนค่ากลับเป็น 0 ซึ่งเป็นค่าตัวเลขจำนวนเต็ม (ซึ่งตรงกับชนิดข้อมูลที่ประกาศไว้ในบรรทัดที่ 4)
- บรรทัดที่ 9 ส่วนสิ้นสุดของฟังก์ชัน main()



การเขียนภาษา C ในส่วนของฟังก์ชัน `main()` นิยมประกาศชนิดข้อมูลคืนค่ากลับเป็น `int` (จำนวนเต็ม) และกำหนดให้ค่าส่งกลับเป็น 0 เพื่อเป็นการบ่งชี้ว่าโปรแกรมนั้นทำงานถูกต้อง ถ้าหากคืนค่ากลับเป็นจำนวนเต็มที่ไม่มากกว่า 0 ก็หมายความว่า โปรแกรมนั้นทำงานล้มเหลวหรือมีข้อผิดพลาดเกิดขึ้น

ความรู้พื้นฐานการพัฒนาซอฟต์แวร์

ในหัวข้อที่ผ่านมาผู้อ่านคงได้รู้จักกับภาษา C กันไปบ้างแล้ว ซึ่งเป็นพื้นฐานที่ทำให้ผู้อ่านได้มองเห็นภาพรวมของการเขียนโปรแกรมด้วยภาษา C แต่ก่อนที่จะเริ่มเขียนโปรแกรมกัน ผู้เขียนอยากให้ผู้อ่านได้เรียนรู้ขั้นตอนการพัฒนาโปรแกรมหรือซอฟต์แวร์กันเสียก่อน เนื่องจากเราเรียนภาษา C เพื่อนำไปพัฒนาโปรแกรมหรือซอฟต์แวร์ในอนาคต

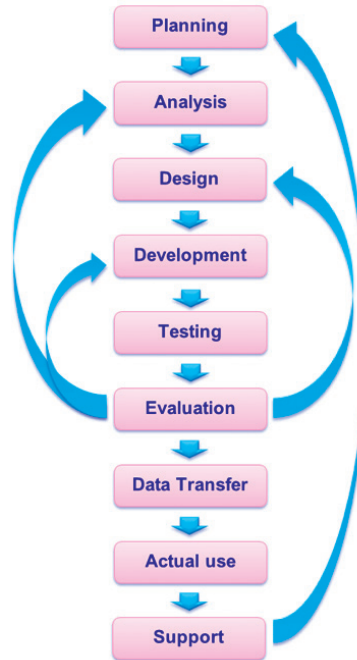
วงจรชีวิตของการพัฒนาซอฟต์แวร์ (Software Development Life Cycle - SDLC)

ในตอนนี้ผู้อ่านจะได้เรียนรู้เกี่ยวกับ**วงจรชีวิตของการพัฒนาซอฟต์แวร์ (Software Development Life Cycle)** หรือเรียกสั้นๆ ว่า **SDLC** ซึ่งเป็นแนวทางหรือขั้นตอนการพัฒนาโปรแกรมซอฟต์แวร์หรือระบบสารสนเทศให้สำเร็จ โดยแนวทางหรือขั้นตอนการพัฒนาซอฟต์แวร์แต่ละซอฟต์แวร์อาจมีรูปแบบหรือรายละเอียดในบางขั้นตอนแตกต่างกัน ขึ้นอยู่กับทีมงานพัฒนาว่าต้องการใช้รูปแบบการพัฒนาแบบใด ที่คิดว่าเหมาะสมกับการพัฒนาระบบที่ต้องการมากที่สุด

ดังนั้น ลำดับวงจรชีวิตของการพัฒนาซอฟต์แวร์แต่ละตัวอาจจะแตกต่างกันได้ แต่สามารถสรุปเป็นขั้นตอนพอเข้าใจได้ดังนี้

1. **วางแผนงาน (Planning)** เป็นเสมือนจุดเริ่มต้นการพัฒนาโปรแกรมหรือระบบงาน เพื่อประมาณการต้นทุนในการพัฒนาซอฟต์แวร์ กำหนดระยะเวลา กำหนดรูปแบบและแนวทางของการพัฒนาซอฟต์แวร์คร่าวๆ ทำให้เราทราบขอบเขตและข้อกำหนดของการพัฒนาในเบื้องต้น
2. **วิเคราะห์ความต้องการของระบบ (Analysis)** ขั้นตอนนี้เป็นการค้นหาความต้องการของซอฟต์แวร์ และวิเคราะห์ความจำเป็นหรือขั้นตอนการทำงานของความต้องการนั้น
3. **ออกแบบระบบ (Design)** เป็นการออกแบบส่วนประกอบต่างๆ ของซอฟต์แวร์ให้ตรงกับความต้องการที่ได้วิเคราะห์มาแล้ว

4. **เขียนโปรแกรม (Development)** เป็นขั้นตอนการเขียนโปรแกรมตามแนวทางที่ได้ออกแบบไว้ ให้โปรแกรมสามารถทำงานได้ตามต้องการ
5. **ทดสอบการทำงานของระบบ (Testing)** ขั้นตอนนี้เป็นการนำระบบหรือซอฟต์แวร์ที่ทำเสร็จแล้วมาทดสอบการใช้งานว่า ทำงานถูกต้องตามความต้องการหรือไม่ การทดสอบนี้จะรวมถึงการทดสอบความเชื่อมโยงกับระบบซอฟต์แวร์อื่นๆ ที่เกี่ยวข้องด้วย เช่น การทำงานร่วมกับเครื่องพิมพ์ในเครือข่าย เป็นต้น
6. **ประเมินการทำงานของระบบ (Evaluation)** เมื่อทดสอบการทำงานของระบบแล้ว เรามีการประเมินว่า ระบบหรือซอฟต์แวร์ที่ทดสอบสามารถทำงานได้ถูกต้องหรือไม่ มีข้อผิดพลาดประการใด เหมาะสมที่จะนำไปใช้งานได้หรือไม่ ในกรณีที่พบข้อผิดพลาดให้กลับมาแก้ไขข้อผิดพลาดที่พบ อย่างไรก็ตาม การแก้ไขดังกล่าวจะแก้ไขที่ขั้นตอนใด จะขึ้นอยู่กับรายละเอียดของข้อผิดพลาดที่เกิดขึ้น เช่น หากข้อผิดพลาดที่พบเจอเป็นการเขียนโปรแกรมผิดพลาด ให้แก้ไขที่ได้โปรแกรม แต่หากเกิดจากการออกแบบระบบผิดพลาด ให้กลับไปแก้ไขที่ขั้นตอนการออกแบบระบบ เป็นต้น (ในรูปแบบการพัฒนาซอฟต์แวร์บางประเภท ขั้นตอนนี้อาจรวมอยู่ในการทดสอบการทำงานของระบบ)
7. **โอนย้ายข้อมูล (Data Transfer)** ขั้นตอนนี้เป็นการนำข้อมูลของระบบเก่าเข้าสู่ระบบใหม่ ก่อนนำไปใช้จริง ในขั้นตอนนี้จะทำก็ต่อเมื่อทดสอบการทำงานและประเมินการทำงานของระบบเรียบร้อยแล้ว (ขั้นตอนนี้อาจจะไม่มีความจำเป็นสำหรับการพัฒนาซอฟต์แวร์บางประเภท)
8. **นำไปใช้งานจริง (Actual use)** เมื่อซอฟต์แวร์พัฒนาสำเร็จและผ่านการทดสอบแล้วก็สามารถนำไปใช้งานจริงได้ ซึ่งขั้นตอนนี้จะรวมไปถึงการแนะนำวิธีการใช้งานแก่ผู้ไปด้วย
9. **การให้ความช่วยเหลือ (Support)** ในขั้นตอนนี้เป็นการให้ความช่วยเหลือต่อผู้ใช้เมื่อพบปัญหาในการใช้งานระบบ แต่ในกรณีที่ไม่สามารถแก้ไขปัญหาได้หรือจะต้องพัฒนาระบบเพิ่มเติม ก็จะกลับไปทำตามขั้นตอนวางแผนงานใหม่



จากรูปจะได้ว่า ภาษา C ที่เราจะเรียนกันนั้นนำไปใช้ในขั้นตอนการเขียนโปรแกรม (Development) นั่นเอง ซึ่งถือเป็นหัวใจของการพัฒนาโปรแกรมหรือซอฟต์แวร์

เปรียบเทียบวิธีการพัฒนาซอฟต์แวร์

จากวงจรชีวิตของการพัฒนาซอฟต์แวร์ ทำให้ปัจจุบันมีผู้นำเสนอระเบียบวิธีการพัฒนาซอฟต์แวร์มากมายหลายรูปแบบ โดยแต่ละรูปแบบมีข้อดีและข้อเสียที่แตกต่างกัน ในที่นี้จะยกตัวอย่างเป็นกรณีศึกษาให้ผู้อ่านคือ โครงสร้างแบบน้ำตก (Waterfall Model) และโครงสร้างแบบก้นหอย (Spiral Model) เป็นต้น

โครงสร้างแบบน้ำตก (Waterfall Model)

ระเบียบวิธีการพัฒนาซอฟต์แวร์โครงสร้างแบบน้ำตกนี้ จะแบ่งกระบวนการทำงานออกเป็นขั้นตอนต่างๆ แต่ละขั้นตอนจะสืบเนื่องกันไปจากขั้นหนึ่งสู่อีกขั้นหนึ่งตามลำดับขั้นเหมือนสายน้ำตก โดยจะเริ่มต้นจากการกำหนดและวิเคราะห์ความต้องการของผู้ใช้ (Requirements) ก้าวไปสู่การสร้างแบบจำลอง และออกแบบวิธีการแก้ปัญหา (Design) ต่อด้วยการสร้างซอฟต์แวร์ (Implementation) จากนั้นก็ทดสอบการทำงานและติดตั้งซอฟต์แวร์ (Verification) และขั้นตอนสุดท้ายคือ การบำรุงรักษาให้ความช่วยเหลือแก่ผู้ใช้ซอฟต์แวร์ (Maintenance) ซึ่งสามารถแสดงได้ดังรูป