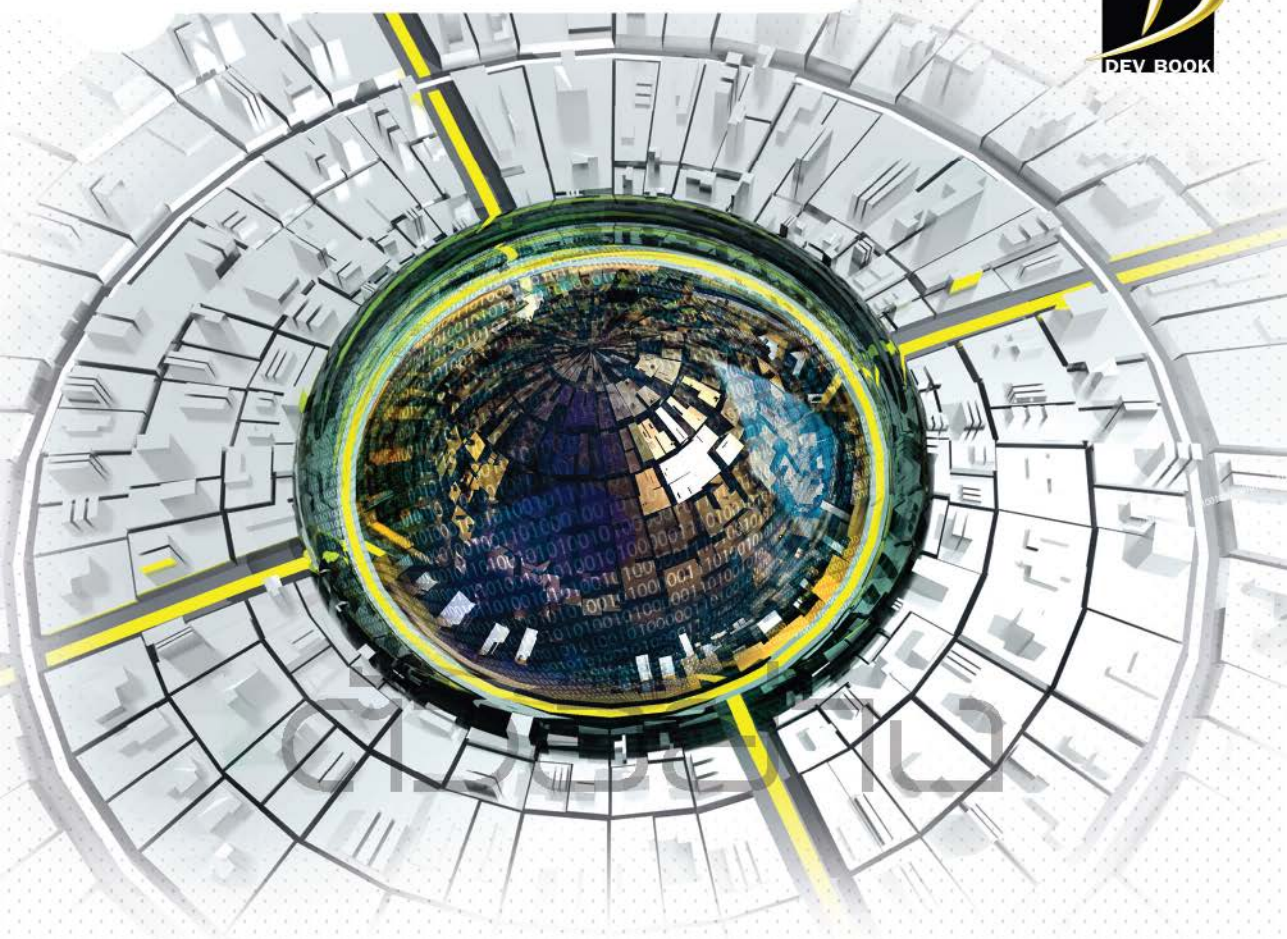




คู่มือเรียนเขียนโปรแกรม

ภาษา C

ฉบับสมบูรณ์

เรียนรู้

ตั้งแต่พื้นฐานอย่างละเอียด
เป็นขั้นตอน

อ่านเข้าใจง่าย

มีแบบฝึกหัดและตัวอย่าง
พร้อมคำอธิบายทุกหัวข้อ

เหมาะสำหรับ

นักเรียน นักศึกษา และผู้ที่สนใจ
เขียนโปรแกรมภาษา C

ไทรศร ตั้งโอภากุล, กิตตินันท์ พลสวัสดิ์

10,000,000

“กว่า 10 ล้านเล่มที่เราผลิต และยังมุ่งมั่นต่อไป”

“หนังสือคุณภาพ เกบสิทธิ์
หนังสือคอมพิวเตอร์
ที่มีผู้อ่านมากที่สุด”

คู่มือเรียนเขียนโปรแกรมภาษา C ฉบับสมบูรณ์

ผู้แต่ง

ไกรศร ตั้งโสภากุล

guru_book@hotmail.com

กิตินันท์ พลสวัสดิ์

neverdie7@windowsslive.com

บรรณาธิการ

กิตินันท์ พลสวัสดิ์

neverdie7@windowsslive.com

ออกแบบปก

อนัน วาโตะ

ออกแบบและจัดรูปเล่ม

วุฒิพันธ์ สมพระเมธ

พิสูจน์อักษร

สุนทรี บรรลือศักดิ์, มนฤดี ศรีอุทโยภาส

ประสานงานการผลิต

สุพัตรา อาจปฐ, ชัตรชนก แก้วจันทร์

Visual Studio 2010 เป็นเครื่องหมายการค้าของบริษัท Microsoft Corp. และเครื่องหมายการค้าอื่นๆ ที่อ้างถึงเป็นของบริษัทนั้นๆ

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

บริษัท ไอดีซี พรีเมียร์ จำกัด จัดตั้งขึ้นเพื่อเผยแพร่ความรู้เทคโนโลยีสารสนเทศทั้งในระดับพื้นฐานและระดับสูง เรายินดีรับงานเขียนของนักวิชาการและนักเขียนทุกท่าน โดยเฉพาะงานที่เกี่ยวข้องกับสารสนเทศ ท่านผู้สนใจกรุณาติดต่อผ่านทางอีเมลที่ editor@infopress.co.th หรือโทรศัพท์หมายเลข 0-2962-1081 (อัตราในมิถุนายน 10 คู่สาย) โทรสาร 0-2962-1084

ข้อมูลทางบรรณานุกรม



ไกรศร ตั้งโสภากุล

คู่มือเรียนเขียนโปรแกรมภาษา C ฉบับสมบูรณ์

นนทบุรี : ไอดีซี, 2556

384 หน้า

1. ภาษาซี (ภาษาโปรแกรมเฉพาะชนิด)

I กิตินันท์ พลสวัสดิ์ ผู้แต่งร่วม II ชื่อเรื่อง

005.133

ISBN

885-916-100-439-4

พิมพ์ครั้งที่ 1

มกราคม 2556

จัดพิมพ์และจัดจำหน่ายโดย



บริษัท ไอดีซี พรีเมียร์ จำกัด

200 หมู่ 4 ชั้น 19 ห้อง 1901 อาคารจัสมินอินเตอร์เนชั่นแนลทาวเวอร์

ถ.แจ้งวัฒนะ อ.ปากเกร็ด จ.นนทบุรี 11120

โทรศัพท์ 0-2962-1081 (อัตราในมิถุนายน 10 คู่สาย) โทรสาร 0-2962-1084

ราคา 250 บาท

สำหรับร้านค้าและตัวแทนจำหน่าย

โทรศัพท์ 0-2962-1081-3

โทรสาร 0-2962-1084

บรรณาธิการ

หนังสือเล่มนี้เป็นหนังสือเขียนเขียนโปรแกรมภาษา C ที่ปรับปรุงมาจากหนังสือ “คู่มือเขียนโปรแกรมภาษา C ฉบับผู้เริ่มต้น” ที่เรียบเรียงโดย คุณประภาพร ช่างไม้ ของ DEV BOOK ซึ่งใช้ในการเรียนการสอนในระดับอุดมศึกษาหลายๆ สถาบันมากกว่า 10 ปี โดยมีวัตถุประสงค์เพื่อปรับปรุงเนื้อหาและคุณภาพของหนังสือให้ดียิ่งขึ้นตามคำติชมจากผู้อ่าน

ภายในเล่มนี้ยังคงรูปแบบโครงสร้างเนื้อหา ตามหลักสูตรการเรียนการสอนในระดับอุดมศึกษา ตั้งแต่ชนิดข้อมูล ตัวดำเนินการ การรับและแสดงผลข้อมูล คำสั่งการทำงานของโปรแกรม ยังรวมไปถึงการสร้างและใช้งานฟังก์ชัน การเขียนโปรแกรมทำงานกับชนิดข้อมูลต่างๆ ไม่ว่าจะเป็นอาร์เรย์ พอยน์เตอร์ หรือโครงสร้างข้อมูลแบบต่างๆ อีกทั้งยังมีตัวอย่างประกอบการอธิบายอย่างละเอียดทุกหัวข้อ พร้อมแบบฝึกหัดท้ายบทให้ผู้อ่านได้ฝึกทักษะการเขียนโปรแกรม ซึ่งจะเห็นได้ว่าเหมาะสำหรับใช้ในการเรียนการสอนการเขียนโปรแกรมภาษา C เป็นอย่างยิ่ง

อย่างไรก็ตามทาง DEV BOOK ขอขอบคุณผู้อ่านทุกๆ ท่านที่ติดตามหนังสือของเสมาเสมอมา และขอขอบคุณทุกๆ คำติชมที่ทำให้เราได้ปรับปรุงและพัฒนาอยู่เสมอ ทาง DEV BOOK หวังเป็นอย่างยิ่งว่า หนังสือเล่มนี้จะเป็นหนังสืออีกเล่มหนึ่งที่ช่วยให้ผู้อ่านทุกท่านเข้าใจและเรียนรู้การเขียนโปรแกรมภาษา C ได้อย่างดีเยี่ยม

สุดท้ายนี้หากหนังสือเล่มนี้มีข้อผิดพลาดประการใด ต้องขออภัยมา ณ ที่นี้ด้วย

ด้วยความเคารพยิ่ง
กิตินันท์ พลสวัสดิ์

คำนำ

ภาษาคอมพิวเตอร์ไม่ว่าจะเป็นภาษา Assembly, C, Java, Visual Basic และอื่นๆ อีกมากมาย ทุกๆ ภาษาที่มีจุดประสงค์หลักอย่างเดียวกันคือ ให้นักพัฒนาโปรแกรมหรือโปรแกรมเมอร์สามารถเขียนโปรแกรมให้ได้ผลลัพธ์หรือออกมาตามต้องการได้ โดยแต่ละภาษาอาจจะมีแนวคิดในการพัฒนาที่แตกต่างกันไปบ้าง แต่อย่างไรก็ตามทุกภาษาจะได้ผลสำเร็จเหมือนกันคือ ผลลัพธ์การทำงานของโปรแกรม

ภาษา C เป็นภาษาระดับสูงที่ถูกพัฒนาขึ้นเพื่อนำไปใช้ในการสร้างระบบปฏิบัติการ ต่อมาได้พัฒนาต่อเพื่อช่วยให้โปรแกรมเมอร์พัฒนาโปรแกรมตามต้องการได้ง่ายยิ่งขึ้น และถือได้ว่าเป็นภาษาที่เหมาะสมสำหรับผู้เริ่มต้นเรียนรู้การเขียนโปรแกรม เนื่องจากเป็นภาษาเชิงโครงสร้างที่เข้าใจง่าย ไม่ซับซ้อน โดยเน้นที่ขั้นตอนการทำงานของโปรแกรมเป็นหลักสำคัญ ทำให้เข้าใจแนวคิดการพัฒนาโปรแกรมได้เป็นอย่างดี

ดังนั้น หนังสือเล่มนี้ผู้เขียนเรียบเรียงขึ้นสำหรับผู้สนใจเริ่มหัดเขียนโปรแกรม โดยเริ่มจากให้ผู้อ่านเข้าใจการทำงานของภาษาโปรแกรม รู้จักเครื่องมือสำหรับใช้พัฒนาโปรแกรม และหลักการเขียนโปรแกรมพื้นฐานที่ควรรู้ เช่น ชนิดข้อมูล ตัวดำเนินการ การรับและแสดงผลข้อมูล คำสั่งการทำงานของโปรแกรม เป็นต้น นอกจากนี้ยังรวมไปถึงการสร้างและใช้งานฟังก์ชัน การเขียนโปรแกรมทำงานกับชนิดข้อมูลต่างๆ ไม่ว่าจะเป็นอาร์เรย์ พอยน์เตอร์ และโครงสร้างข้อมูลแบบต่างๆ ซึ่งถือได้ว่าเหมาะสำหรับผู้เริ่มต้นเขียนโปรแกรมภาษา C หรือใช้ประกอบการเรียนการสอนเป็นอย่างยิ่ง

สุดท้ายนี้ผู้เขียนหวังเป็นอย่างยิ่งว่า หนังสือเล่มนี้จะช่วยให้ผู้อ่านทุกๆ ท่านเข้าใจการเขียนโปรแกรมด้วยภาษา C และสามารถนำไปประยุกต์ใช้งานต่อไปได้ และหากหนังสือเล่มนี้มีข้อผิดพลาดประการใด ต้องขออภัยมา ณ ที่นี้ด้วย

ขอขอบพระคุณบิดามารดา ที่ให้กำเนิดและเลี้ยงดูจนเติบโต

ขอขอบพระคุณครู อาจารย์ ที่ท่านประทานความรู้มาให้

ด้วยความเคารพยิ่ง
ไกรสร ตั้งโอภากุล

Contents

บทที่ 1 รู้จักภาษา C กับการพัฒนาโปรแกรม..... 1

ความเป็นมาของภาษา C	1
จุดเด่นของภาษา C	2
โครงสร้างของภาษา C	3
ความรู้พื้นฐานการพัฒนาซอฟต์แวร์	6
วงจรชีวิตของการพัฒนาซอฟต์แวร์ (Software Development Life Cycle-SDLC)	6
ระเบียบวิธีการพัฒนาซอฟต์แวร์	8
สรุปเนื้อหาบทที่ 1	9

บทที่ 2 เตรียมพร้อมก่อนเขียนโปรแกรม 11

Microsoft Visual C++ 2010 Express Edition	11
เตรียมตัวก่อนติดตั้ง Microsoft Visual C++ 2010 Express	11
ดาวน์โหลด Microsoft Visual C++ 2010 Express	12
ติดตั้งโปรแกรม Microsoft Visual C++ 2010 Express	13
เริ่มต้นเขียนโปรแกรมภาษา C ด้วย Microsoft Visual C++ 2010 Express	15
การบันทึกโปรเจกต์ (Save Project)	20
การเปิดโปรเจกต์ (Open Project)	21
การเขียนคำอธิบาย (Comment)	21
Borland Turbo C++ 4.5 for Win	22
ติดตั้งโปรแกรม Borland Turbo C++ 4.5 for Win	22
เริ่มต้นเขียนโปรแกรมภาษา C ด้วย Turbo C++ 4.5	25
การบันทึกโค้ดโปรแกรม	28
การเปิดใช้งานโปรแกรม	29
สรุปเนื้อหาบทที่ 2	30

บทที่ 3 ตัวแปรและชนิดของข้อมูล..... 31

ตัวแปร (Variable)	31
กฎการตั้งชื่อ	32
การประกาศตัวแปร	32
ชนิดของข้อมูล (Data Type)	33
ชนิดข้อมูลแบบจำนวนเต็ม (Integer type)	33
ชนิดข้อมูลแบบตัวอักษร (Character type)	37
ชนิดข้อมูลแบบข้อความ (String type)	38
ชนิดข้อมูลแบบจำนวนเลขทศนิยม (Floating point type)	39
ค่าคงที่ (Constants)	42
กฎของการแปลงชนิดของข้อมูล (Data Type Conversion)	44
Implicit Type Conversion	44
Explicit Type Conversion (Casting)	46
รู้จักกับ Null character และ Empty string	48
สรุปเนื้อหาบทที่ 3	49
แบบฝึกหัดท้ายบทที่ 3	50

บทที่ 4 ตัวดำเนินการและนิพจน์ (Operator and Expression) 51

นิพจน์ (Expression)	51
ตัวดำเนินการ (Operator).....	52
ตัวดำเนินการกำหนดค่า (Assignment Operator).....	52
ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operator)	54
ตัวดำเนินการยูนารี (Unary Operator)	57
ตัวดำเนินการเปรียบเทียบ (Comparison Operator).....	60
ตัวดำเนินการตรรกะ (Logical Operator)	62
ตัวดำเนินการแบบมีเงื่อนไข (Conditional Operator)	64
ตัวดำเนินการบอกขนาดชนิดข้อมูล (Sizeof Operator).....	65
ตัวดำเนินการระดับบิต (Bitwise Operator)	66
Bitwise Shift Left (<<).....	70
Bitwise Shift Right (>>).....	72
Bitwise One's Complement (~).....	74
ลำดับความสำคัญของตัวดำเนินการ (Operator of Precedence)	76
สรุปเนื้อหาบทที่ 4.....	79
แบบฝึกหัดท้ายบทที่ 4	80

บทที่ 5 การแสดงผลและการรับข้อมูล..... 83

การแสดงผลข้อมูล (Display Data).....	83
การแสดงผลทีละตัวอักษรด้วยคำสั่ง putchar().....	83
การแสดงผลเป็นข้อความด้วยคำสั่ง puts()	84
การแสดงผลข้อมูลทุกชนิดด้วยคำสั่ง printf()	85
รหัสรูปแบบการแสดงผลข้อความของคำสั่ง printf()	86
จัดรูปแบบการแสดงผลข้อมูลของคำสั่ง printf().....	88
การกำหนดการแสดงผลข้อมูลของคำสั่ง printf().....	89
การจัดการพื้นที่ที่แสดงผลข้อมูลของคำสั่ง printf()	91
การรับข้อมูล (Input Data).....	93
การรับข้อมูลที่ละตัวอักษรด้วยคำสั่ง getch() และ getchar().....	93
การรับข้อมูลชนิดข้อความด้วยคำสั่ง gets()	96
การรับข้อมูลทุกชนิดด้วยคำสั่ง scanf()	97
การกำหนดลำดับการรับข้อมูลของคำสั่ง scanf().....	100
การกำหนดจำนวนตัวอักษรในการรับค่าข้อมูลชนิดข้อความของคำสั่ง scanf().....	101
การกำหนดรูปแบบการรับค่าข้อมูลของคำสั่ง scanf().....	102
การนับจำนวนตัวอักษรที่รับค่าข้อมูลด้วยคำสั่ง scanf().....	103
สรุปเนื้อหาบทที่ 5.....	104
แบบฝึกหัดท้ายบทที่ 5	105

บทที่ 6 ควบคุมทิศทางการทำงานของโปรแกรมด้วยคำสั่งตัดสินใจ

(Decision Control Statement).....107

คำสั่งตัดสินใจแบบเลือกทำหรือไม่ทำด้วยคำสั่ง if.....	107
คำสั่งตัดสินใจแบบสองทางเลือกด้วยคำสั่ง if...else	109

คำสั่งตัดสินใจแบบหลายทางเลือกด้วยคำสั่ง if...else if.....	111
คำสั่งตัดสินใจแบบหลายทางเลือกด้วยคำสั่ง switch-case	114
สรุปเนื้อหาบทที่ 6.....	117
แบบฝึกหัดท้ายบทที่ 6	119

บทที่ 7 ควบคุมการทำงานของโปรแกรมด้วยคำสั่งวนลูป

(Repetition Control Statement)121

วนลูการทำงานด้วยจำนวนรอบที่แน่นอนด้วยคำสั่ง for	121
วนลูการทำงานเมื่อเงื่อนไขเป็นจริงด้วยคำสั่ง while	123
วนลูการทำงานอย่างน้อย 1 ครั้งด้วยคำสั่ง do...while.....	125
สรุปเนื้อหาบทที่ 7.....	130
แบบฝึกหัดท้ายบทที่ 7	131

บทที่ 8 การเขียน Flowchart133

หลักการเขียน Flowchart.....	133
แนวคิดการเขียน Flowchart	134
การทำงานแบบตามลำดับ (Sequence).....	134
การเลือกทำงานตามเงื่อนไข (Decision)	136
การทำซ้ำ (Loop).....	140
สรุปเนื้อหาบทที่ 8.....	143
แบบฝึกหัดท้ายบทที่ 8	144

บทที่ 9 ตัวแปรอาร์เรย์ (Array)147

ตัวแปรอาร์เรย์ (Array Variable)	147
ตัวแปรอาร์เรย์ 1 มิติ (One Dimension Array).....	147
ตัวแปรอาร์เรย์หลายมิติ (Multi-Dimension Array).....	148
การเขียนโปรแกรมกับตัวแปรอาร์เรย์	150
การประกาศตัวแปรอาร์เรย์	150
การกำหนดค่าข้อมูลให้ตัวแปรอาร์เรย์	150
การอ้างถึงข้อมูลในตัวแปรอาร์เรย์	151
สรุปเนื้อหาบทที่ 9.....	157
แบบฝึกหัดท้ายบทที่ 9	158

บทที่ 10 พอยน์เตอร์ (Pointer)159

ตัวแปรพอยน์เตอร์.....	159
การใช้งานพอยน์เตอร์	160
การดำเนินการกับพอยน์เตอร์	163
การใช้งานพอยน์เตอร์กับอาร์เรย์.....	165
การจัดการพื้นที่หน่วยความจำแบบไดนามิก (Dynamic Memory Allocation)	170
รู้จักกับ Pointers to functions.....	175
สรุปเนื้อหาบทที่ 10.....	177
แบบฝึกหัดท้ายบทที่ 10	178

บทที่ 11 ฟังก์ชัน (Function)179

ฟังก์ชันคืออะไร	179
ฟังก์ชันที่สร้างขึ้นเอง (User-Defined Functions)	180
ฟังก์ชันที่ไม่มีการคืนค่ากลับและไม่มีการรับค่าพารามิเตอร์ (Void Functions with No Parameters)	180
ฟังก์ชันที่ไม่มีการคืนค่ากลับ แต่มีการรับค่าพารามิเตอร์ (Void Functions with Parameters)	183
ฟังก์ชันที่มีการคืนค่ากลับ แต่ไม่มีการรับค่าพารามิเตอร์ (Function Return Value with No Parameters)	186
ฟังก์ชันที่มีการคืนค่ากลับและมีการรับค่าพารามิเตอร์ (Function Return Value with Parameters)	188
การส่งค่าผ่านพารามิเตอร์ (Parameter Passing)	191
ฟังก์ชันประเภทส่งผ่านค่า (Call by Value)	191
ฟังก์ชันประเภทส่งผ่านตัวอ้างอิง (Call by Reference)	192
ขอบเขตการทำงานของตัวแปร	194
ฟังก์ชันเรียกตัวเอง (Recursive Function)	196
ฟังก์ชันที่มาตรฐานภาษา C ได้สร้างมาให้แล้ว (Standard Library Function)	197
ไลบรารีฟังก์ชันการคำนวณทางคณิตศาสตร์ (Math Library)	197
ไลบรารีฟังก์ชันสำหรับอักขระและข้อความ (String Library)	199
สรุปเนื้อหาบทที่ 11	200
แบบฝึกหัดท้ายบทที่ 11	201

บทที่ 12 โครงสร้างข้อมูล (Structuring Data).....203

รู้จักโครงสร้างข้อมูลแบบ struct	203
การประกาศโครงสร้างข้อมูลแบบ struct	204
การประกาศโครงสร้างข้อมูลแบบ struct ด้วยคำสั่ง typedef	205
การใช้งานตัวแปร struct	207
การกำหนดค่าข้อมูลให้กับ struct	207
การอ่านค่าข้อมูลจาก struct	208
อาร์เรย์กับโครงสร้างข้อมูล	210
พอยน์เตอร์กับโครงสร้างข้อมูล	213
โครงสร้างข้อมูลซ้อนโครงสร้างข้อมูล	215
การใช้งานโครงสร้างข้อมูลกับฟังก์ชัน	217
ฟังก์ชันที่ไม่มีการรับค่าและมีการส่งค่ากลับ	217
ฟังก์ชันที่มีการรับค่าและไม่มีการส่งค่ากลับ	219
ฟังก์ชันที่มีการรับค่าและมีการส่งค่ากลับ	222
ยูเนียน (union)	224
ค่าคงที่ enum (Enumeration)	228
สรุปเนื้อหาบทที่ 12	230
แบบฝึกหัดท้ายบทที่ 12	230

บทที่ 13 รู้จักและใช้งาน Linked List.....233

โครงสร้างข้อมูลแบบเชิงเส้น (Linear Data Structure)	233
Dense List	234
Linked List.....	234
สแตติกลิงคัลลิสต์ (Static Linked List)	234
ไดนามิกลิงคัลลิสต์ (Dynamic Linked List).....	235
เริ่มต้นเขียนโปรแกรมกับ Linked List	236
สรุปเนื้อหาบทที่ 13.....	243
แบบฝึกหัดบทที่ 13.....	243

บทที่ 14 เพิ่มข้อมูล (File)245

รู้จักกับแฟ้มข้อมูล.....	245
การเปิดและปิดไฟล์ (File Open/Close)	246
การเปิดไฟล์ (File Open)	246
การปิดไฟล์ (File Close).....	248
การทำงานกับ Text File	248
การอ่านข้อมูลจากไฟล์ทีละตัวอักษรด้วยฟังก์ชัน <code>getc()</code> และฟังก์ชัน <code>fgetc()</code>	248
การเขียนข้อมูลลงไฟล์ทีละตัวอักษรด้วยฟังก์ชัน <code>putc()</code> และฟังก์ชัน <code>fputc()</code>	252
การอ่านข้อมูลจากไฟล์ทีละบรรทัดด้วยฟังก์ชัน <code>fgets()</code>	255
การเขียนข้อความลงไฟล์ด้วยฟังก์ชัน <code>fputs()</code>	257
การอ่านข้อมูลจากไฟล์โดยใช้ฟังก์ชัน <code>fscanf()</code>	259
การเขียนข้อมูลลงไฟล์ด้วยฟังก์ชัน <code>fprintf()</code>	260
การทำงานกับ Binary File	265
การอ่านข้อมูลจากไฟล์ด้วยฟังก์ชัน <code>fread()</code>	265
การเขียนข้อมูลลงไฟล์ด้วยฟังก์ชัน <code>fwrite()</code>	266
การตรวจสอบสถานะของไฟล์ (File Status Function)	272
การตรวจสอบ EOF (End Of File) ด้วยฟังก์ชัน <code>feof()</code>	272
การตรวจสอบ Error ที่เกิดขึ้นกับไฟล์ด้วยฟังก์ชัน <code>ferror()</code>	272
การหาตำแหน่งที่อยู่ของไฟล์พอยน์เตอร์ด้วยฟังก์ชัน <code>ftell()</code>	272
การย้ายตำแหน่งไฟล์พอยน์เตอร์ไปยังตำแหน่งต้นไฟล์ด้วยฟังก์ชัน <code>rewind()</code>	273
การย้ายตำแหน่งไฟล์พอยน์เตอร์ด้วยฟังก์ชัน <code>fseek()</code>	275
การดำเนินการเกี่ยวกับไฟล์ (File Operation)	278
การลบไฟล์	278
การเปลี่ยนชื่อไฟล์	279
สรุปเนื้อหาบทที่ 14.....	280
แบบฝึกหัดท้ายบทที่ 14.....	281

บทที่ 15 รู้จักกับ Preprocessor Directive.....285

Preprocessor Directive.....	285
Includes.....	286
Conditional Compilation	291
มาโคร (Macro).....	296

Macro Definition.....	296
Standard Predefined Positioning Macros	299
Multiple Lines.....	300
Precedence.....	301
สรุปเนื้อหาบทที่ 15.....	302
แบบฝึกหัดท้ายบทที่ 15.....	303

บทที่ 16 รู้จักกับ Command line argument305

มาสนุกกับการสร้าง Command line argument	306
การประยุกต์ใช้งาน.....	309
สรุปท้ายเล่ม.....	311
แบบฝึกหัดบทที่ 16	311

ภาคผนวก ก313

ภาคผนวก ข.....319

เฉลยแบบฝึกหัดท้ายบทที่ 3.....	319
เฉลยแบบฝึกหัดท้ายบทที่ 4.....	320
เฉลยแบบฝึกหัดท้ายบทที่ 5.....	322
เฉลยแบบฝึกหัดท้ายบทที่ 6.....	323
เฉลยแบบฝึกหัดท้ายบทที่ 7.....	325
เฉลยแบบฝึกหัดท้ายบทที่ 8.....	327
เฉลยแบบฝึกหัดท้ายบทที่ 9.....	329
เฉลยแบบฝึกหัดท้ายบทที่ 10.....	332
เฉลยแบบฝึกหัดท้ายบทที่ 11.....	333
เฉลยแบบฝึกหัดท้ายบทที่ 12.....	336
เฉลยแบบฝึกหัดท้ายบทที่ 13.....	337
เฉลยแบบฝึกหัดท้ายบทที่ 14.....	343
เฉลยแบบฝึกหัดท้ายบทที่ 15.....	348
เฉลยแบบฝึกหัดท้ายบทที่ 16.....	350

ภาคผนวก ค แนะนำภาษา Objective C.....353

ชนิดของข้อมูลในภาษา Objective C (Data Type).....	355
ควบคุมทิศทางการทำงานของโปรแกรมด้วยคำสั่งตัดสินใจ (Decision Control Statement) ...	356
คำสั่ง if.....	356
คำสั่ง if...else	358
คำสั่ง Nested if.....	360
คำสั่ง switch-case.....	362
ควบคุมการทำงานของโปรแกรมด้วยคำสั่งวนลูป (Repetition Control Statement).....	364
คำสั่ง for.....	364
คำสั่ง while.....	366
คำสั่ง do...while	368

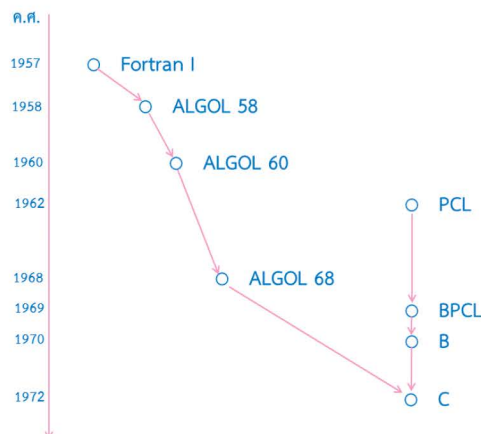
รู้จักภาษา C กับการพัฒนาโปรแกรม

ภาษา C เป็นภาษาระดับสูง ที่ถูกสร้างขึ้นมาเพื่อให้การเขียนโปรแกรมง่ายขึ้น ปัจจุบันถือว่าเป็นภาษาคอมพิวเตอร์สำหรับผู้เริ่มต้นที่ต้องการเรียนรู้การเขียนโปรแกรม เนื่องจากเป็นภาษาเชิงโครงสร้างที่มีหลักการทำงานไม่ซับซ้อน เข้าใจง่าย

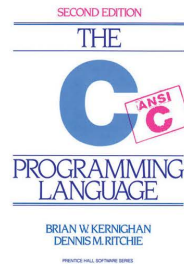
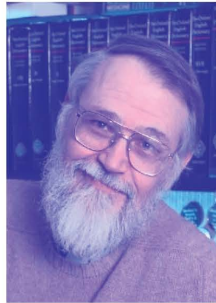
เนื้อหาในบทนี้ผู้อ่านจะได้ทราบประวัติความเป็นมา, จุดเด่น, โครงสร้างของภาษา C และนอกจากนี้ผู้เขียนจะแนะนำเครื่องมือที่ใช้ในการเขียนโปรแกรมภาษา C ที่ง่ายและสะดวกให้กับผู้อ่านด้วย

ความเป็นมาของภาษา C

ภาษา C เป็นภาษาคอมพิวเตอร์ที่ได้พัฒนาขึ้นในปี ค.ศ. 1972 (พ.ศ. 2515) โดยเดนนิส ริตชี (Dennis Ritchie) แห่ง Bell Telephone Laboratories, Inc. (ในปัจจุบันก็คือ AT & T Bell Laboratories) ซึ่งภาษา C มีการพัฒนามาจากภาษา B ในช่วงแรกภาษา C ได้ถูกนำมาใช้เพื่อสร้างระบบปฏิบัติการ Unix หากนำวิวัฒนาการของภาษา C มาแสดงออกเป็นแผนภาพจะได้ดังนี้



ในปี ค.ศ. 1978 (พ.ศ. 2521) เดนนิส ริชชี และนายเบรอน เครนิกันฮาน (Dennis Ritchie and Brian W. Kernighan) ได้แต่งหนังสือชื่อ “The C Programming Language” โดยนำเสนอภาษา C ที่สามารถนำมาปรับใช้กับคอมพิวเตอร์ในรูปแบบต่างๆ ได้มากยิ่งขึ้น และทำให้ภาษา C ได้รับความนิยมอย่างมาก จนกระทั่งในปี ค.ศ. 1988 (พ.ศ. 2531) ได้มีการสร้างมาตรฐานของภาษา C ขึ้นมาในชื่อของ ANSI C ภายใต้ความร่วมมือระหว่างสถาบัน ANSI (American National Standards Institute) กับนายเดนนิส ริชชี และนายเบรอน เครนิกันฮาน



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

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

จุดเด่นของภาษา C

ในภาษาโปรแกรมทุกภาษามีจุดเด่นในการประมวลผลของภาษาแตกต่างกัน ในข้อหัวนี้ผู้อ่านจะได้รู้จักกับจุดเด่นของภาษา C ซึ่งมีดังนี้

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

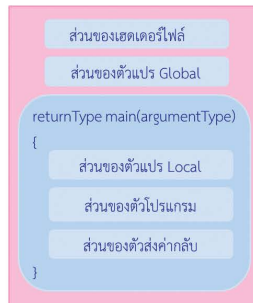
TIPS



คอมไพเลอร์ (Compiler) จะทำหน้าที่แปลงภาษาโปรแกรมทั้งหมดที่เขียนให้เป็นภาษาเครื่อง พร้อมกันนี้จะทำหน้าที่ตรวจสอบไวยากรณ์ของภาษา หากมีข้อผิดพลาดก็จะส่งข้อความหรือคำเตือนออกมา เมื่อกระบวนการทำงานของคอมไพเลอร์เสร็จสมบูรณ์ โปรแกรมจึงเริ่มทำงาน ซึ่งแตกต่างกับหลักการทำงานของ อินเตอร์พรีเตอร์ (interpreter) ซึ่งการทำงานจะแปลงภาษาโปรแกรมและประมวลผลคำสั่งทีละคำสั่ง

โครงสร้างของภาษา C

ในโปรแกรมที่พัฒนาด้วยภาษา C ทุกโปรแกรมจะมีโครงสร้างการพัฒนาที่ไม่แตกต่างกัน ซึ่งประกอบด้วย 6 ส่วนหลักๆ โดยแต่ละส่วนจะมีหน้าที่แตกต่างกันดังนี้



1. **ส่วนของเฮดเดอร์ไฟล์ (Header File or Processing Directive)** ส่วนนี้จะมีจุดสังเกตที่สำคัญคือ จะขึ้นต้นด้วยเครื่องหมาย # เสมอ การทำงานของคอมไพเลอร์จะทำงานในส่วนนี้เป็นส่วนแรก ในส่วนของเฮดเดอร์ไฟล์จะเป็นส่วนที่เก็บไลบรารีมาตรฐานของภาษา C ซึ่งจะถูกรวบรวมเข้ามารวมกับโปรแกรมในขณะที่กำลังคอมไพล์ โดยใช้คำสั่ง #include ซึ่งสามารถเขียนได้เป็น 2 รูปแบบคือ

รูปแบบที่ 1

```
#include<HeaderName>
```

รูปแบบที่ 2

```
#include "HeaderName"
```

โดยที่ HeaderName เป็นชื่อ Header File

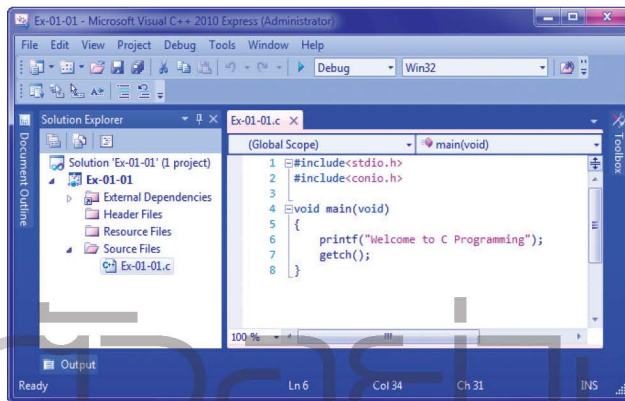
ผู้อ่านคงสงสัยแล้วว่า ทั้งสองแบบนี้มีความแตกต่างกันอย่างไร ทั้งสองแบบนี้ต่างกันที่การทำงานคือแบบที่ใช้เครื่องหมาย <...> คอมไพเลอร์จะค้นหาเฮดเดอร์ไฟล์จากไลบรารีของภาษา C เพียงทีเดียวเท่านั้น ส่วนที่ใช้เครื่องหมาย “...” คอมไพเลอร์จะค้นหาเฮดเดอร์ไฟล์จากไลบรารีที่เก็บ Source Code ของเราก่อน ถ้าหากไม่เจอก็จะไปค้นหาที่ไลบรารีของภาษา 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) โดยในขอบเขตของฟังก์ชันจะเริ่มต้นด้วยเครื่องหมาย { และสิ้นสุดด้วยเครื่องหมาย }

4. **ส่วนของตัวแปร Local** เป็นส่วนประกาศตัวแปรที่สามารถใช้ได้เฉพาะภายในฟังก์ชันของตนเองเท่านั้น ซึ่งส่วนนี้จะไม่มีหรือไม่มีก็ได้
5. **ส่วนของตัวโปรแกรม** เป็นส่วนคำสั่งการทำงานของโปรแกรม โดยที่คำสั่งในแต่ละคำสั่งจะต้องจบด้วยเครื่องหมาย ; เสมอ
6. **ส่วนของตัวส่งค่ากลับ** เป็นส่วนของการส่งค่าข้อมูลกลับเมื่อฟังก์ชันจบการทำงาน โดยค่าที่ส่งกลับนั้น จะต้องเป็นค่าที่มีชนิดของข้อมูลตรงกับชนิดของข้อมูลที่ฟังก์ชันคืนค่ากลับ (return type) ในกรณีที่ไม่ต้องการให้ฟังก์ชันมีการส่งค่ากลับ สามารถกำหนดได้โดยใช้คีย์เวิร์ด void

ตัวอย่างที่ 1.1 เป็นตัวอย่างการเขียนโปรแกรมภาษา C อย่างง่าย เพื่อแสดงส่วนต่างๆ ของโปรแกรมในกรณีประกาศฟังก์ชัน main() ที่ไม่มีการคืนค่าใดๆ (void) ดังรูป



เมื่อทดสอบการทำงานของโปรแกรมจะได้ผลลัพธ์ดังนี้



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 Welcome to C Programming |
| บรรทัดที่ 7 | เป็นคำสั่งรับอักขระจากแป้นพิมพ์ ในที่นี้เพื่อกำหนดไม่ให้โปรแกรมปิดหน้าต่างผลลัพธ์เมื่อแสดงผลลัพธ์ที่ต้องการแล้ว |
| บรรทัดที่ 8 | ส่วนสิ้นสุดของฟังก์ชัน main() |

NOTE

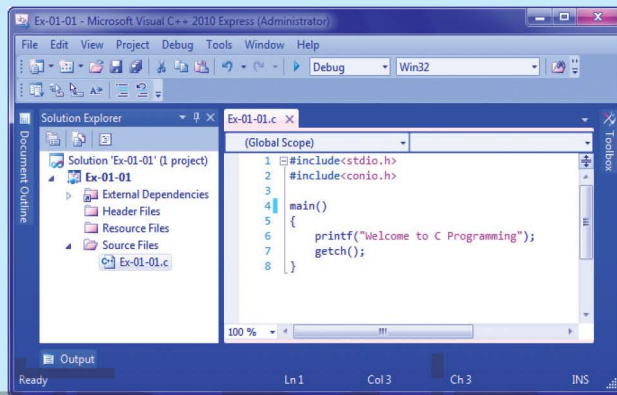


เมื่อผู้อ่านได้อ่านมาถึงจุดนี้ ผู้อ่านอาจจะไม่เข้าใจความหมายของคำบางคำ เช่น void, return type เป็นต้น แต่ผู้อ่านไม่ต้องเป็นกังวล เนื่องจากส่วนนี้ผู้เขียนมีเป้าหมายให้ผู้อ่านมองเห็นภาพรวมของการเขียนโปรแกรมภาษา C เท่านั้น โดยในส่วนต่างๆ ที่ผู้อ่านเป็นกังวล ผู้เขียนจะอธิบายในบทต่อไป

TIPS



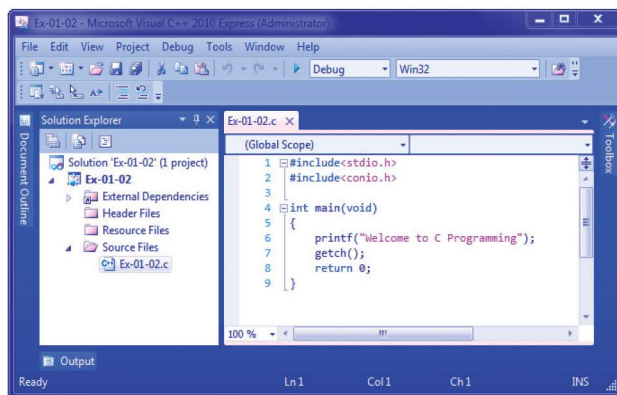
ในกรณีประกาศฟังก์ชัน main() โดยไม่ระบุค่าส่งกลับจะมีค่า Default ของค่าส่งกลับเป็น integer ดังรูป



```
1 #include<stdio.h>
2 #include<conio.h>
3
4 int main()
5 {
6     printf("Welcome to C Programming");
7     getch();
8 }
```

ตัวอย่างที่ 1.2

เป็นตัวอย่างการเขียนโปรแกรมภาษา C อย่างง่าย เพื่อแสดงส่วนต่างๆ ของโปรแกรมกรณีประกาศฟังก์ชัน main() ที่มีการส่งค่าข้อมูลกลับดังรูป



```
1 #include<stdio.h>
2 #include<conio.h>
3
4 int main(void)
5 {
6     printf("Welcome to C Programming");
7     getch();
8     return 0;
9 }
```

เมื่อทดสอบการทำงานของโปรแกรมจะได้ผลลัพธ์ดังนี้



```
D:\C\Editor Version\Ex\Ch1\Ex-01-02\Debug\Ex-01-02.exe
Welcome to C Programming_
```


อธิบายการทำงานของโปรแกรม

บรรทัดที่ 1-2	เป็นการเรียกใช้งานส่วนของเฮดเดอร์ไฟล์ สังเกตได้ที่เครื่องหมาย # โดยมีการเรียกใช้ไลบรารี <code>stdio.h</code> ซึ่งเป็นไลบรารีจัดการเกี่ยวกับอินพุตและเอาต์พุต และไลบรารี <code>conio.h</code> ซึ่งเป็นไลบรารีจัดการเกี่ยวกับหน้าจอทั้งหมด
บรรทัดที่ 4	ส่วนของฟังก์ชัน <code>main()</code> โดยประกาศชนิดข้อมูลที่คืนค่ากลับเป็น <code>int</code> และค่าที่รับเข้ามาในฟังก์ชันเป็น <code>void</code> หมายถึง ฟังก์ชันนี้จะมีการส่งค่าข้อมูลกลับออกไป แต่ไม่มีการรับค่าใดๆ เข้ามาในฟังก์ชัน
บรรทัดที่ 5	ส่วนเริ่มต้นของฟังก์ชัน <code>main()</code>
บรรทัดที่ 6	เป็นคำสั่งแสดงผลทางจอภาพ โดยให้พิมพ์คำว่า <code>Welcome to C Programming</code>
บรรทัดที่ 7	เป็นคำสั่งรับอักขระจากแป้นพิมพ์ ในที่นี้เพื่อกำหนดไม่ให้โปรแกรมปิดหน้าต่างผลลัพธ์เมื่อแสดงผลลัพธ์ที่ต้องการแล้ว
บรรทัดที่ 8	เป็นส่วนของการส่งค่ากลับ ในที่นี้ให้ส่งค่ากลับเป็น 0 ซึ่งเป็นค่าตัวเลข (ซึ่งตรงกับชนิดข้อมูลที่ประกาศไว้ในบรรทัดที่ 4)
บรรทัดที่ 9	ส่วนสิ้นสุดของฟังก์ชัน <code>main()</code>

ความรู้พื้นฐานการพัฒนาซอฟต์แวร์

ในหัวข้อที่ผ่านมาผู้อ่านคงได้รู้จักกับภาษา C กันไปบ้างแล้ว ซึ่งเป็นพื้นฐานที่ทำให้ผู้อ่านได้มองเห็นภาพรวมการเขียนโปรแกรมด้วยภาษา C แต่ก่อนที่จะเริ่มเขียนโปรแกรมกัน ผู้เขียนอยากให้ผู้อ่านได้เรียนรู้ขั้นตอนการพัฒนาโปรแกรมหรือซอฟต์แวร์กันเสียก่อน เนื่องจากเราเรียนภาษา C เพื่อนำไปพัฒนาโปรแกรมหรือซอฟต์แวร์ในอนาคต

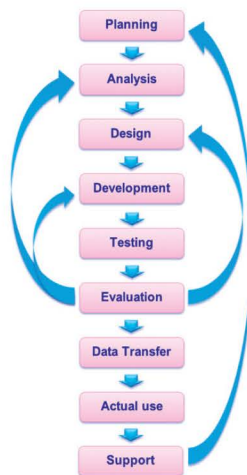
วงจรชีวิตของการพัฒนาซอฟต์แวร์ (Software Development Life Cycle-SDLC)

ในตอนนี้ผู้อ่านจะได้เรียนรู้เกี่ยวกับ**วงจรชีวิตของการพัฒนาซอฟต์แวร์ (Software Development Life Cycle)** หรือเรียกสั้นๆ ว่า **SDLC** ซึ่งเป็นแนวทางหรือขั้นตอนการพัฒนาโปรแกรม ซอฟต์แวร์ หรือระบบสารสนเทศให้สำเร็จ โดยแนวทางหรือขั้นตอนการพัฒนาซอฟต์แวร์แต่ละซอฟต์แวร์อาจมีรูปแบบหรือรายละเอียดในบางขั้นตอนแตกต่างกัน ขึ้นอยู่กับทีมงานพัฒนาว่าต้องการใช้รูปแบบการพัฒนาแบบใด ที่คิดว่าเหมาะสมกับการพัฒนาระบบที่ต้องการมากที่สุด

ดังนั้น ลำดับวงจรชีวิตของการพัฒนาซอฟต์แวร์แต่ละตัวอาจจะแตกต่างกันได้ แต่สามารถสรุปเป็นขั้นตอนพอให้เข้าใจได้ดังนี้

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

4. **เขียนโปรแกรม (Development)** เป็นขั้นตอนการเขียนโปรแกรมตามแนวทางที่ได้ออกแบบไว้ให้โปรแกรมสามารถทำงานได้ตามต้องการ
5. **ทดสอบการทำงานของระบบ (Testing)** ขั้นตอนนี้เป็นการนำระบบหรือซอฟต์แวร์ที่ทำเสร็จแล้วมาทดสอบการใช้งานว่าทำงานถูกต้องตามความต้องการหรือไม่ การทดสอบนี้จะรวมถึงการทดสอบการเชื่อมโยงกับระบบซอฟต์แวร์อื่นๆ ที่เกี่ยวข้องด้วย เช่น การทำงานร่วมกับเครื่องพิมพ์ในเครือข่าย เป็นต้น
6. **ประเมินการทำงานของระบบ (Evaluation)** เมื่อทดสอบการทำงานของระบบแล้ว เราจะมีการประเมินว่าระบบหรือซอฟต์แวร์ที่ทดสอบสามารถทำงานได้ถูกต้องหรือไม่ มีข้อผิดพลาดประการใด เหมาะสมที่จะนำไปใช้งานได้หรือไม่ ในกรณีที่พบข้อผิดพลาดให้กลับมาแก้ไขข้อผิดพลาดที่พบ อย่างไรก็ตามการแก้ไขดังกล่าวจะแก้ไขที่ขั้นตอนใด จะขึ้นอยู่กับรายละเอียดของข้อผิดพลาดที่เกิดขึ้น เช่น หากข้อผิดพลาดที่พบเจอเป็นการเขียนโปรแกรมผิดพลาด ให้แก้ไขที่โค้ดโปรแกรม แต่หากเกิดจากการออกแบบระบบผิดพลาด ให้กลับไปแก้ไขที่ขั้นตอนการออกแบบระบบ เป็นต้น (ในรูปแบบการพัฒนาซอฟต์แวร์บางประเภท ขั้นตอนนี้อาจรวมอยู่ในการทดสอบการทำงานของระบบ)
7. **โอนย้ายข้อมูล (Data Transfer)** ขั้นตอนนี้เป็นการนำข้อมูลของระบบเก่ามาเข้าสู่ระบบใหม่ก่อนนำไปใช้จริง ในขั้นตอนนี้จะทำก็ต่อเมื่อทดสอบการทำงานและประเมินการทำงานของระบบเรียบร้อยแล้ว (ขั้นตอนนี้ อาจจะไม่มีความจำเป็นสำหรับการพัฒนาซอฟต์แวร์บางประเภท)
8. **นำไปใช้งานจริง (Actual use)** เมื่อซอฟต์แวร์พัฒนาสำเร็จและผ่านการทดสอบแล้ว ก็สามารถนำไปใช้งานจริงได้ ซึ่งขั้นตอนนี้อาจรวมไปถึงการแนะนำวิธีการใช้งานแก่ผู้ใช้ด้วย
9. **การให้ความช่วยเหลือ (Support)** ในขั้นตอนนี้เป็นการให้ความช่วยเหลือต่อผู้ใช้ เมื่อพบปัญหาในการใช้งานระบบ แต่ในกรณีที่ไม่สามารถแก้ไขปัญหาได้หรือจะต้องพัฒนาระบบเพิ่มเติมก็จะกลับไปทำตามขั้นตอนการวางแผนงานใหม่



◀ วงจรชีวิตของการพัฒนาซอฟต์แวร์
(Software Development Life Cycle-SDLC)

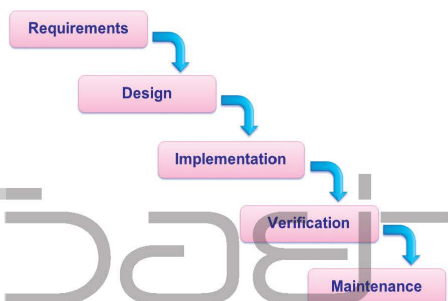
จากรูปจะเห็นได้ว่า ภาษา C ที่เราจะเรียนกันนั้นนำไปใช้ในขั้นตอนการเขียนโปรแกรม (Development) นั่นเอง ซึ่งถือเป็นเรื่องหัวใจของการพัฒนาโปรแกรมหรือซอฟต์แวร์

ระเบียบวิธีการพัฒนาซอฟต์แวร์

จากวงจรชีวิตของการพัฒนาซอฟต์แวร์ ทำให้ปัจจุบันมีผู้นำเสนอระเบียบวิธีการพัฒนาซอฟต์แวร์มากมายหลายรูปแบบ แต่ละรูปแบบมีข้อดีและข้อเสียที่แตกต่างกัน ในที่นี้จะขอยกตัวอย่างเป็นกรณีศึกษาให้ผู้อ่านคือ 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



▲ ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบ Waterfall Model

โครงสร้างแบบน้ำตกจะทำงานให้เสร็จทีละขั้น และจะไม่มีการย้อนกลับมาทำขั้นตอนก่อนหน้านี้อีก ดังนั้น จึงเหมาะสมกับระบบที่มีความต้องการที่ชัดเจน ซึ่งมีข้อดีและข้อเสียดังนี้

ข้อดี

- แบ่งงานง่ายให้เป็นงานเล็กๆ เพื่อให้ง่ายต่อการจัดการ
- มีการกำหนดซอฟต์แวร์ที่ต้องส่งมอบในแต่ละงานอย่างชัดเจน

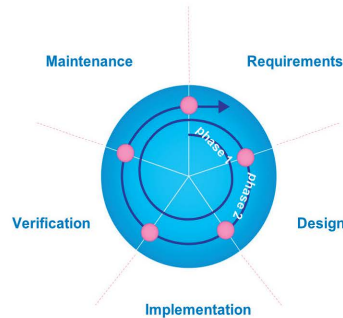
ข้อเสีย

- ถ้าหากพบข้อผิดพลาดของขั้นที่เสร็จสิ้นแล้ว จะไม่สามารถแก้ไขได้ การแก้ไขจำเป็นต้องเริ่มรอบ (Iteration) ใหม่ ซึ่งอาจเกิดความเสียหายอย่างใหญ่หลวง
- ผู้ใช้งานจะได้ใช้ซอฟต์แวร์ก็ต่อเมื่อถึงปลายทางของโครงการเท่านั้น

โครงสร้างแบบก้นหอย (Spiral Model)

โครงสร้างแบบก้นหอย (Spiral Model) เป็นการสร้างต้นแบบกับการทำงานอย่างเป็นขั้นตอน เช่นเดียวกับโครงสร้างแบบน้ำตก แต่จะเป็นลักษณะวนซ้ำการทำงานไปเรื่อยๆ จนพัฒนาซอฟต์แวร์เสร็จสมบูรณ์ ในหนึ่งรอบการทำงานจะเรียกว่า **phase** โดยแต่ละ phase จะมีการวิเคราะห์และจัดการความเสี่ยง เพื่อลดข้อผิดพลาดจากการพัฒนาซอฟต์แวร์ด้วย

โครงสร้างแบบก้นหอยจะมีพัฒนาการของซอฟต์แวร์เป็นเวอร์ชัน โดยเวอร์ชันต่อไปจะมีความสมบูรณ์มากยิ่งขึ้นในช่วง phase ใดๆ ซึ่งสามารถแสดงได้ดังรูป



▲ ระเบียบวิธีการพัฒนาซอฟต์แวร์แบบ Spiral Model

ข้อดี

- เนื่องจากการพัฒนาแบ่งออกเป็นช่วงๆ (phase) เมื่อพบข้อผิดพลาดจึงสามารถแก้ไขได้ทันที
- เป็น Model ที่เหมาะกับงานที่มีโอกาสเปลี่ยนแปลงบ่อย หรือมีความต้องการใหม่ๆ มีลักษณะเป็นเวอร์ชัน

ข้อเสีย

- เป็น Model ที่เหมาะกับซอฟต์แวร์ขนาดใหญ่ เนื่องจากการวิเคราะห์และจัดการความเสี่ยงเป็นค่าใช้จ่าย ซึ่งอาจจะไม่คุ้มสำหรับโครงการขนาดเล็ก และผู้ที่จัดการความเสี่ยงได้ต้องมีประสบการณ์

สรุปเนื้อหาบทที่ 1

ภาษา C ได้ถูกคิดค้นโดย Dennis Ritchie ในปี ค.ศ. 1972 (พ.ศ. 2515) ซึ่งมีการพัฒนามาจากภาษา ALGOL, ภาษา BCPL และภาษา B โดยมีจุดเด่นอยู่ตรงที่

- มีแนวคิดในการพัฒนาโดยอาศัยหลักการ “โปรแกรมเชิงโครงสร้าง (Structure Programming)”
- การทำงานไม่ขึ้นกับฮาร์ดแวร์ สามารถนำไปใช้ใน CPU รุ่นต่างกันได้
- สามารถทำงานแทนภาษา Assembly ได้
- ความสามารถของคอมพิวเตอร์ทำงานได้รวดเร็ว โดยใช้รหัสออบเจกต์ (Object) ที่สั้น
- โครงสร้างของภาษา C จะประกอบด้วยส่วนหลักๆ ดังนี้
 - ส่วนของเฮดเดอร์ไฟล์ (Header File or Processing Directive)
 - ส่วนของตัวแปร Global
 - ส่วนของฟังก์ชัน (Function)
 - ส่วนของตัวแปร Local
 - ส่วนของตัวโปรแกรม
 - ส่วนของตัวส่งค่ากลับ