



ใบความรู้ที่ 2.3

เริ่มต้นกับภาษาซี

ภาษา C เป็นภาษาที่เก่าแก่อีกาเนิดมายาวนาน โดยแต่เดิมนั้นภาษา C ถูกพัฒนาขึ้นเพื่อให้เป็นภาษาสำหรับการสร้างระบบปฏิบัติการยูนิกซ์ (Unix) เนื่องจากในขณะนั้นระบบปฏิบัติการยูนิกซ์เขียนด้วยภาษาแอสเซมบลี (Assembly) ซึ่งเป็นภาษาที่ยึดติดกับฮาร์ดแวร์ของเครื่อง ดังนั้น การที่ย้ายระบบปฏิบัติการไปใช้กับเครื่องอื่นจึงเป็นเรื่องที่ลำบากมาก ซึ่งนับเป็นข้อเสียที่สำคัญของภาษาแอสเซมบลี

ดังนั้น ภาษา C ซึ่งเป็นภาษาที่ไม่ยึดติดกับฮาร์ดแวร์จึงถูกพัฒนาขึ้นมา ในปัจจุบันภาษา C ไม่ได้จำกัดอยู่เพียงแค่การสร้างระบบปฏิบัติการเท่านั้น แต่สามารถนำไปใช้สร้างโปรแกรมเพื่องานในทุกประเภท ยกตัวอย่างเช่น งานเกี่ยวกับการคำนวณ ควบคุมการทำงานของอุปกรณ์ฮาร์ดแวร์ชนิดต่าง ๆ การจัดการฐานข้อมูล หรือสร้างโปรแกรมต่าง ๆ เป็นต้น

ประวัติของภาษา C



ภาษา C คิดค้นขึ้นมาเป็นครั้งแรกโดย เดนิส ริทชี (Dennis Ritchie) ที่ห้องแล็บเบล (Bell Labs) ในปี ค.ศ.1972 โดยได้แนวคิดมาจากภาษา BCPL ซึ่งพัฒนาขึ้นโดย มาร์ติน ริชาร์ด (Martin Richards) และภาษา B ที่เขียนขึ้นโดย เคน ทอมป์สัน (Ken Thompson) เพื่อนำมาพัฒนาต่อจนได้ภาษาใหม่ที่มีประสิทธิภาพการทำงานสูง หลังจากนั้นในปี ค.ศ. 1978 ภาษา C จึงได้รับการเผยแพร่อย่างเป็นทางการโดย เคอร์นิกันเฮน (Kernighan) และเดนิส ริทชี



ขั้นตอนการพัฒนาโปรแกรม

โปรแกรมที่เขียนขึ้นโดยใช้ภาษาโปรแกรมต่างๆ นั้นเราเรียกว่า รหัสต้นฉบับ (source code) ซึ่งอยู่ในรูปของข้อความตามหลักการเขียนโปรแกรมของภาษาโปรแกรมที่สามารถอ่าน และทำความเข้าใจได้โดยมนุษย์เท่านั้น ดังนั้น เพื่อให้เครื่องคอมพิวเตอร์สามารถเข้าใจโปรแกรมและปฏิบัติได้ จึงต้องนำรหัสต้นฉบับมาผ่านกระบวนการแปลงให้อยู่ในรูปของอ็อบเจกต์โค้ดที่ประกอบด้วยรหัสตัวเลข 0 และ 1 ก่อน เราเรียกกระบวนการแปลงดังกล่าวว่า การคอมไพล์โปรแกรม

การสร้างโปรแกรมที่สามารถใช้งานได้ขึ้นมาโปรแกรมหนึ่ง ในภาษาซีมีขั้นตอนดังนี้

ขั้นตอนที่ 1 เขียนโปรแกรม (source code)

สร้างตัวโปรแกรมที่เป็นตัวอักษรหรือเรียกว่า **ซอร์สไฟล์ (Source file)** โดยมีนามสกุลเป็น .c หรือ .cpp ขึ้นมาก่อน โดยใช้โปรแกรมที่สามารถเขียนไฟล์ที่เก็บอักขระ (Editor) ใดๆ ก็ได้ อักขระหรืออักขระใดๆ นั้น จะต้องอยู่ในรูปแบบของการโปรแกรมภาษา (ขั้นตอนนี้คือการสร้างโปรแกรมที่เป็นภาษามนุษย์นั่นเอง)

ขั้นตอนที่ 2 คอมไพล์โปรแกรม (compile)

คอมไพเลอร์ของภาษาซี (C Compiler) จะทำการแปลงซอร์สไฟล์ จากอักขระใดๆ ให้เป็นรหัสที่เครื่องคอมพิวเตอร์สามารถเข้าใจได้เก็บไว้ในอีกไฟล์หนึ่งเรียกว่าไฟล์วัตถุประสงค์ (Object file) ที่มีนามสกุล .obj (ขั้นตอนนี้เรียกว่า **การคอมไพล์** เป็นการแปลงภาษามนุษย์เป็นภาษาเครื่องนั่นเอง)

- หากเกิดข้อผิดพลาด จะแจ้งให้ผู้เขียนโปรแกรมทราบ ผู้เขียนโปรแกรมจะต้องกลับไปแก้ไขโปรแกรม และทำการคอมไพล์โปรแกรมใหม่อีกครั้ง
- หากไม่พบข้อผิดพลาด คอมไพเลอร์จะแปลไฟล์ source code จากภาษาซีไปเป็นภาษาเครื่อง (ไฟล์นามสกุล .obj) เช่นถ้าไฟล์ source code ชื่อ work.c ก็จะถูกแปลไปเป็นไฟล์ work.obj ซึ่งเก็บภาษาเครื่องไว้เป็นต้น

นอกจากคอมไพเลอร์แล้ว ยังมีตัวแปลภาษาอีกรูปแบบหนึ่งที่เรียกว่า อินเตอร์พรีเตอร์ การอ่านและแปลโปรแกรมทีละบรรทัด เมื่อแปลผลบรรทัดหนึ่งเสร็จก็จะทำงานตามคำสั่งในบรรทัดนั้น แล้วจึงทำการแปลผลตามคำสั่งในบรรทัดไป หลักการที่อินเตอร์พรีเตอร์ใช้เรียกว่า อินเตอร์พรีต (Interpret)



ข้อดีและข้อเสียของตัวแปลภาษาทั้งสองแบบมีดังนี้

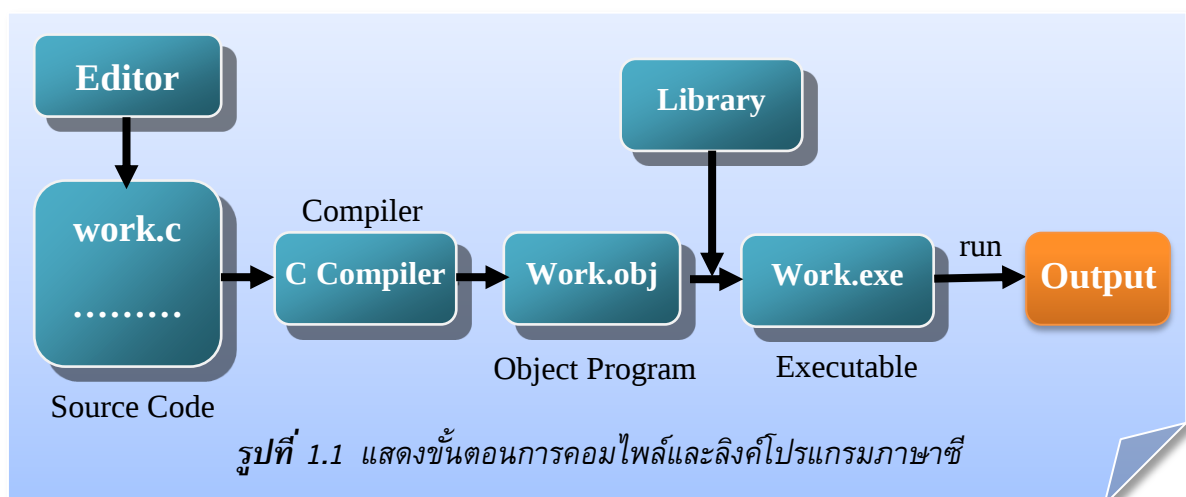
	ข้อดี	ข้อเสีย
คอมไพเลอร์	■ ทำงานได้เร็ว เนื่องจากทำการแปลผลทีเดียว แล้วจึงทำงานตามคำสั่งของโปรแกรมในภายหลัง ■ เมื่อทำการแปลผลแล้ว ในครั้งต่อไปไม่จำเป็นต้องการแปลผลใหม่อีก เนื่องจากภาษาเครื่องที่แปลได้จะถูกเก็บไว้ที่หน่วยความจำ สามารถเรียกใช้งานได้ทันที	■ เมื่อเกิดข้อผิดพลาด ขึ้นกับโปรแกรมจะตรวจสอบหาข้อผิดพลาดได้ยาก เพราะทำการแปลผลทีเดียวทั้งโปรแกรม
อินเตอร์พรีเตอร์	■ หาข้อผิดพลาดของโปรแกรมได้ง่ายเนื่องจากทำการแปลผลทีละบรรทัด ■ เนื่องจากทำงานทีละบรรทัดดังนั้นจึงสั่งให้โปรแกรมทำงานตามคำสั่งเฉพาะจุดที่ต้องการได้ ■ ไม่เสียเวลารอการแปลโปรแกรมเป็นเวลานาน	■ ช้าเนื่องจากที่ทำงานทีละบรรทัด

ขั้นตอนที่ 3 เชื่อมโยงโปรแกรม (link)

ตัวเชื่อม (Linker) จะทำการตรวจสอบว่าในโปรแกรมที่เขียนขึ้นนั้น มีการเรียกใช้งานฟังก์ชันมาตรฐานใดจากห้องสมุดของภาษาซี (C Library) บ้างหรือไม่ ถ้ามี ตัวเชื่อมจะทำการรวมเอาฟังก์ชันเหล่านั้นเข้ากับไฟล์วัตถุประสงค์ แล้วจะได้ไฟล์ที่สามารถทำงานได้ โดยมีนามสกุลเป็น .exe (ขั้นตอนนี้เรียกว่า การลิงค์ เป็นการรวมฟังก์ชันสำเร็จรูปเข้าไป แล้วสร้างไฟล์ที่ทำงานได้)

ขั้นตอนที่ 4 ประมวลผล (run)

เมื่อนำ executable program จากขั้นตอนที่ 3 มาประมวลผลก็จะได้ผลลัพธ์ (output) ของโปรแกรมออกมา (ถ้ามี)



การใช้โปรแกรม Bloodshed Dev-C++

ชุดพัฒนาหรือเครื่องมือที่ช่วยในการพัฒนาโปรแกรม ภาษาอังกฤษเรียกว่า IDE (Integrel Devenlopment Environment) เป็นโปรแกรมที่ออกแบบมาเพื่อช่วยให้ผู้ที่ทำการเขียนโปรแกรมใช้ในการสร้างโปรแกรม โดยจะมี อีดิเตอร์ (Editor) สำหรับเขียนโค้ดของโปรแกรมและมีตัวแปลภาษามาให้พร้อม ปัจจุบันมีการออกชุดพัฒนามาหลายรุ่น และเปลี่ยนแปลงรวดเร็วมาก เช่น Turbo C++ , Borland C++, Microsoft C/C++, Microsoft Visual C++, Microsoft Visual C#, Microsoft Visual C++.NET ซึ่งชุดพัฒนาแต่ละตัวมีวิธีการนำไปใช้งานที่แตกต่างกัน เพราะมันเป็นผลิตภัณฑ์ที่พัฒนามาจากบริษัทต่างกัน แต่อย่างไรก็ตามการเขียนโปรแกรมภาษาซีไม่ว่าจะเป็น IDE ใด ก็มีหลักการและวิธีการในการเขียนที่คล้ายคลึงกัน จะต่างกันบ้างตรงรายละเอียดบางอย่างที่เพิ่มขึ้น หรือพัฒนาให้ง่ายในการเขียนโปรแกรม

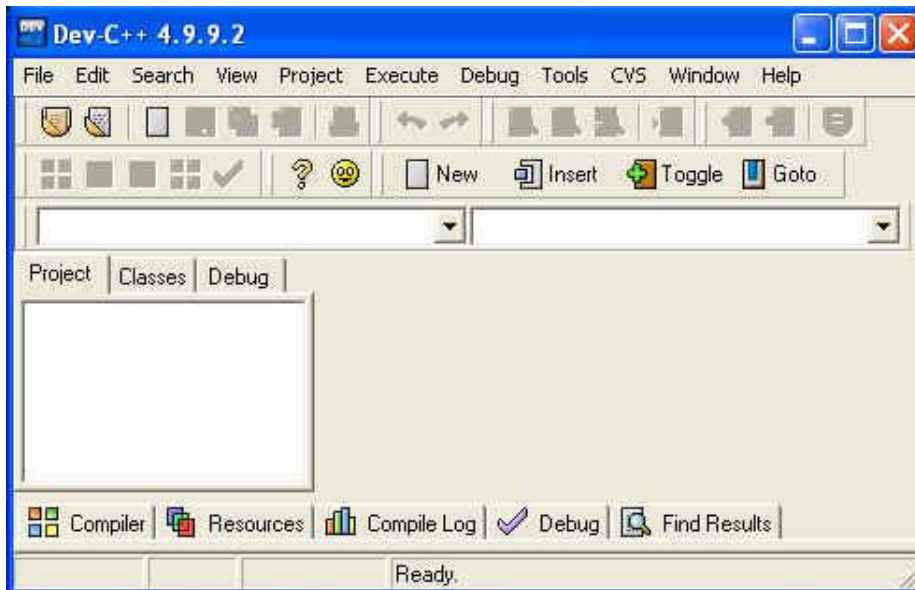
ที่จะกล่าวถึงต่อไปนี้เป็น IDE ของ Bloodshed-Dev C++ ซึ่งเป็นชุดพัฒนาขึ้นมาเพื่อใช้เป็นฟรีแวร์และทำงานภายใต้ระบบปฏิบัติการ Windows ใช้งานได้กับ Windows ทุกรุ่น ซึ่งมีวิธีการเรียกโปรแกรมขึ้นมาใช้งานได้ดังนี้

• การเรียกใช้โปรแกรม

1. คลิกที่ Start > All Program > Bloodshed Dev-C++ > Dev-C++

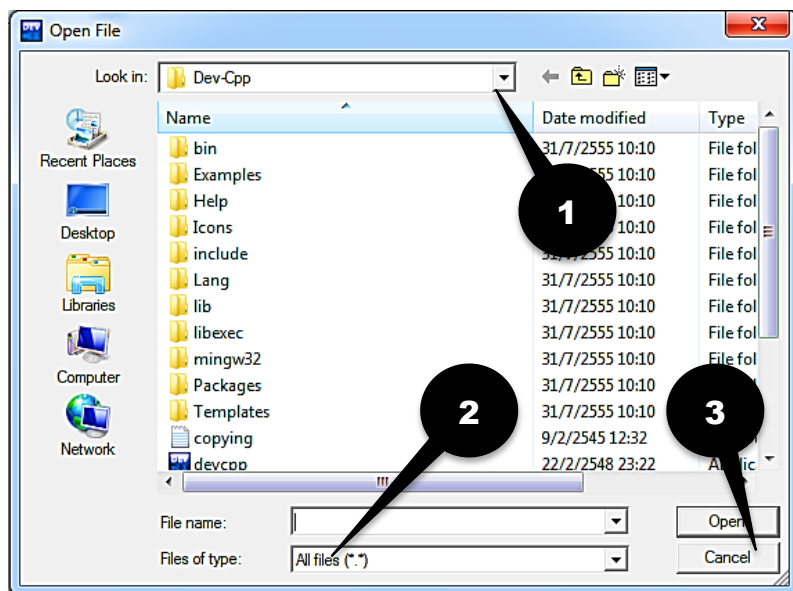


2. หน้าต่างโปรแกรม Bloodshed Dev-C++ จะปรากฏขึ้นมาบนหน้าจอ ดังรูป



• การเปิดแฟ้มโปรแกรม

ผู้ใช้สามารถเปิดแฟ้มโปรแกรมขึ้นมาแสดงบนหน้าต่างโปรแกรมเพื่อแก้ไข แปลหรือสั่งให้
กระทำการ โดยเลือกคลิกที่เมนู File > Open Project or File... จะได้หน้าต่าง Open File ดังรูป

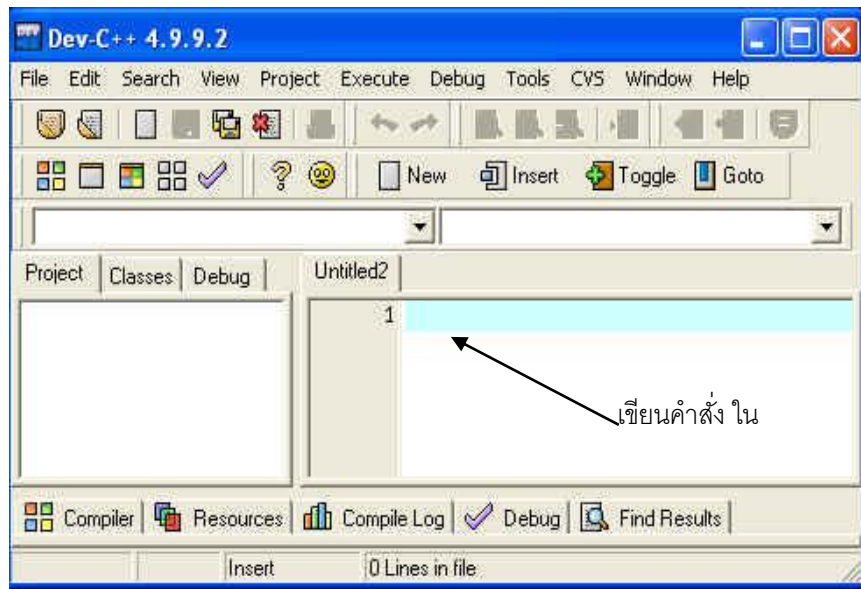


1. เปลี่ยนไดรฟ์ เปลี่ยนโฟลเดอร์
2. พิมพ์ชื่อไฟล์ หรือคลิกเลือกชื่อไฟล์
3. คลิกปุ่ม Open

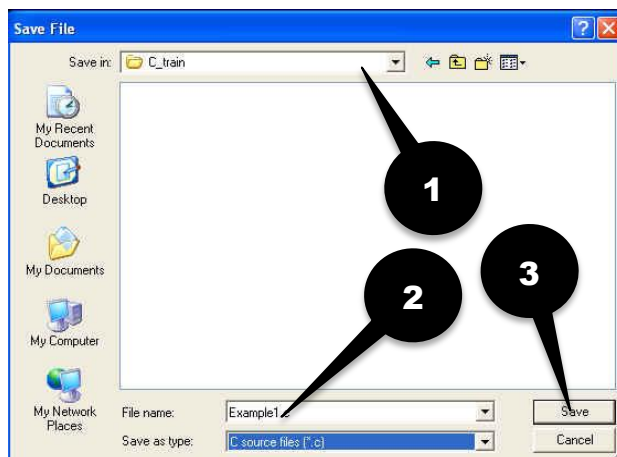


• การสร้างแฟ้มโปรแกรม

การสร้างแฟ้มโปรแกรม ทำได้โดยเลือกคลิกเมนู File > New > Source File หรือกด Ctrl+N จะได้หน้าต่าง Edit ดังรูป



• การบันทึกแฟ้มโปรแกรม



ในการบันทึกโปรแกรมใหม่ ทำได้โดยเลือกเมนู File > Save หรือกด Ctrl+S จะได้หน้าต่างดังรูป

1. เลือกโฟลเดอร์ที่ต้องการเก็บไฟล์
2. ตั้งชื่อไฟล์ตามต้องการ เพื่อความมั่นใจให้ใส่ส่วนขยายของไฟล์เป็น .c จะได้ไฟล์ source code ของโปรแกรมที่จะเป็นภาษา C ถ้าไม่กำหนดให้ถูกต้อง จะมีส่วนขยายเป็น .cpp ซึ่งเป็นไฟล์ ของ Source code ในภาษา C++ ซึ่งมีรายละเอียดบางประการต่างไป จึงต้องระวังในเรื่องนี้
3. คลิกปุ่ม Save

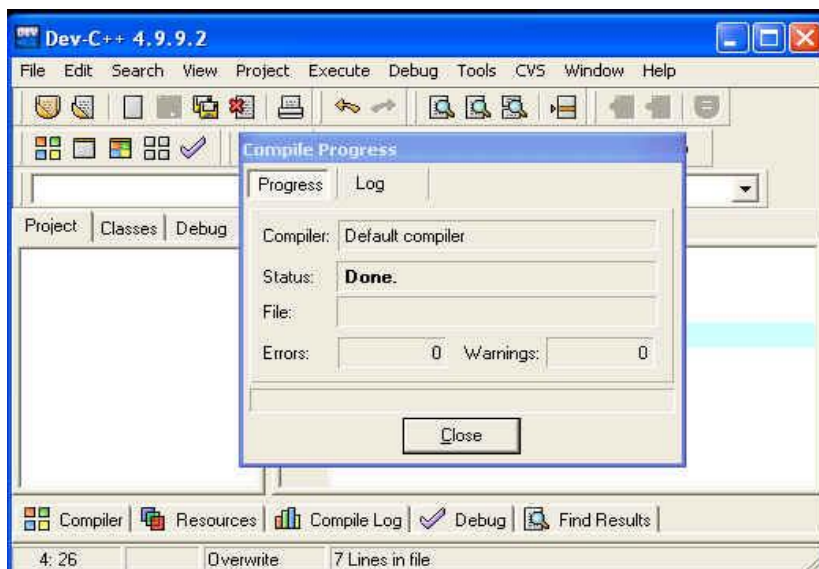


- การแปลโปรแกรม (Compile)

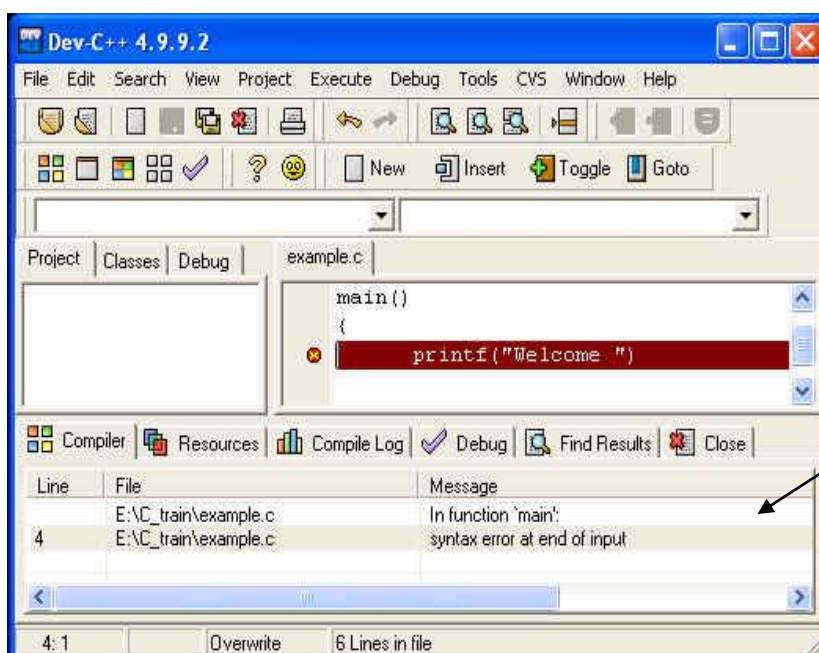
การเรียกใช้คอมไพเลอร์ ทำได้โดยเลือกเมนู Execute > Compile หรือกด Ctrl+F9

ถ้าโปรแกรมที่เขียนสามารถ Compile ได้สำเร็จ แม้จะมีข้อผิดพลาดแต่เป็นข้อผิดพลาดที่ไม่มี

ปัญหาต่อการ Compile จะได้หน้าต่าง Compile Progress ดังรูป

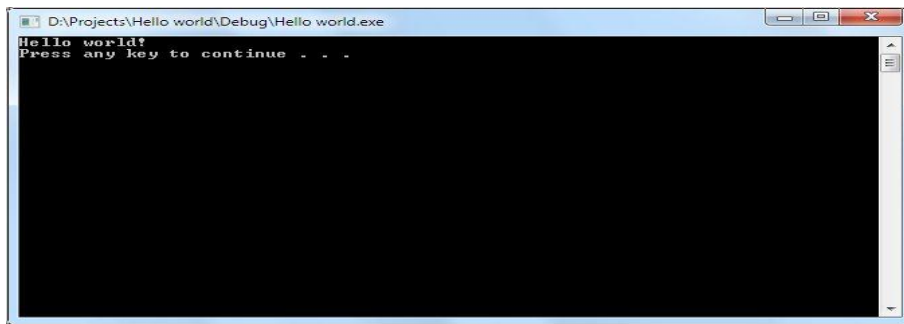


แต่ถ้ามีปัญหาต่าง ๆ ที่ทำให้ไม่ได้โปรแกรมที่สามารถทำงานได้ เช่น เขียนคำสั่งผิด จะแสดงความผิดพลาด ลักษณะดังรูป

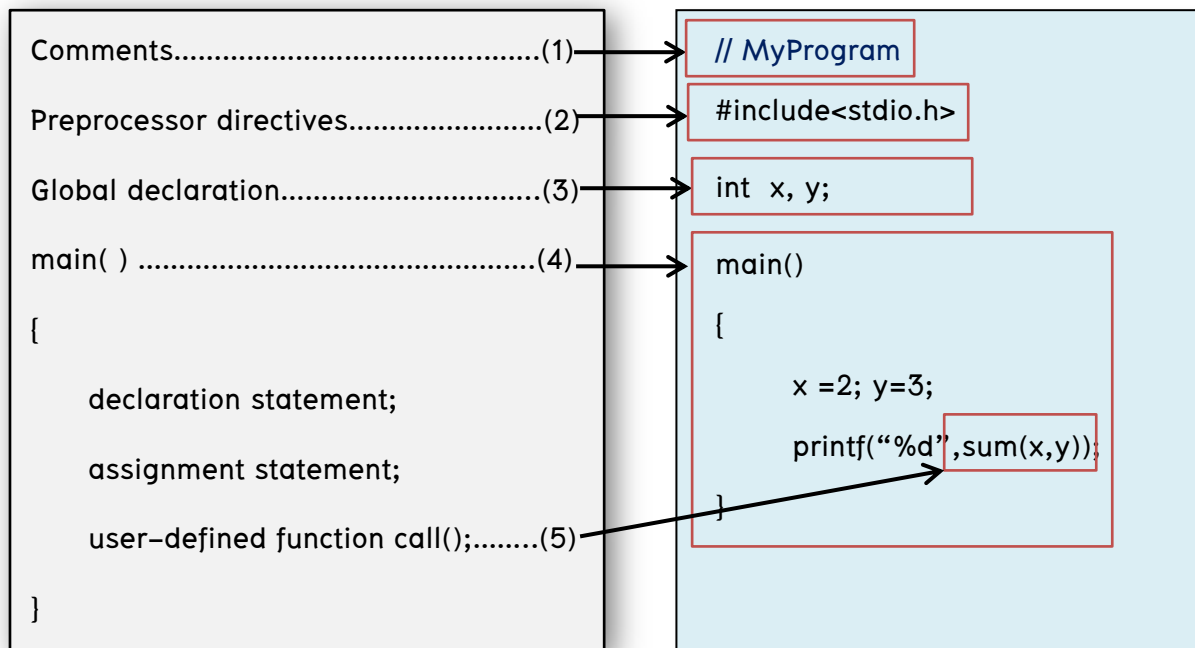


- การสั่งให้โปรแกรมทำงาน (Run)

เมื่อโปรแกรมได้ถูกแปลเป็นโปรแกรมภาษาเครื่องเรียบร้อยแล้ว ผู้ใช้สามารถสั่งให้โปรแกรมที่แปลแล้วกระทำการ โดยเลือกที่เมนู Execute > Run หรือกด Ctrl+F5 จะได้นหน้าต่างผลการ Run โปรแกรมตัวอย่างดังรูป



โครงสร้างพื้นฐานของโปรแกรมภาษาซี



1) ส่วนอธิบายโปรแกรม (Comments)

ใช้สำหรับเขียนอธิบายการทำงานของโปรแกรม ซึ่งข้อความในส่วนนี้ เมื่อผ่านการแปลคำสั่งตัวแปลคำสั่งจะไม่สนใจว่าเป็นส่วนหนึ่งของโปรแกรม เพื่อให้เข้าใจและอ่านโปรแกรมง่ายขึ้น การเขียนส่วนอธิบายโปรแกรมจะใช้เครื่องหมาย `//` หรือ `/*` และ `*/` คร่อมข้อความที่ต้องการอธิบาย ดังนี้

`// ..... ข้อความที่ต้องการอธิบาย`

แต่ถ้าต้องการเขียนอธิบายหลายๆ บรรทัดจะเขียนได้ดังนี้

```
/* .....  
..... ข้อความที่ต้องการอธิบาย.....  
..... */
```

การเขียนส่วนคำอธิบายโปรแกรมจะเขียนไว้ที่ไหนในโปรแกรมก็ได้ โดยมากจะนิยมเขียนอธิบายขั้นตอนการทำงานก่อนเขียนคำสั่ง หรือ อธิบายความหมายของคำสั่งไว้ข้างคำสั่งนั้นก็ได้

2) พรีโปรเซสเซอร์ไดเรกทีฟ (Preprocessor directives)

เป็นส่วนที่ทุกโปรแกรมต้องมี ใช้สำหรับเรียกไฟล์ที่โปรแกรมต้องการ และกำหนดค่าต่างๆ ซึ่งจะต้องเริ่มต้นด้วยเครื่องหมายไดเรกทีฟ (`#`) และตามด้วยชื่อที่ต้องการกำหนดค่า



ตัวอย่างไคเรกทีฟที่ใช้อย

#include เป็นการบอกให้ตัวแปลคำสั่ง (compiler) อ่านไฟล์อื่นเข้ามาพร้อมในการคอมไพล์มีรูปแบบดังนี้

#include<filename> หรือ #include"filename"

เช่น **#include<stdlib.h>**

อ่านไฟล์ **stdlib.h** จากไคเรคทอรีที่กำหนด

#include"sample.h"

อ่านไฟล์ **sample.h** จากไคเรคทอรีปัจจุบันหรือที่กำหนด

- การใช้เครื่องหมาย **<...>** คร่อมชื่อ เพื่อบอกให้คอมไพเลอร์ค้นหาไฟล์จากไคเรคทอรีที่กำหนดใน Option directory แต่ถ้าใช้เครื่องหมาย **"..."** จะเป็นการกำหนดให้ค้นหาจากไคเรคทอรีปัจจุบัน และไคเรคทอรีที่กำหนดเส้นทางไว้

- ชื่อฟังก์ชันการทำงานพื้นฐานของภาษาซี ตัวแปร ค่าคงที่ และมาโครพื้นฐานโดยปกติจะรวมกันเป็นกลุ่มในไฟล์ที่เรียกว่าไฟล์หัว (Header Files) คือไฟล์ ที่มีนามสกุล **.h** ไฟล์หัวแต่ละไฟล์จะเก็บโมดูลแยกกันเป็นส่วนการทำงานแต่ละส่วนเพื่อให้ผู้เขียนโปรแกรมสามารถเรียกใช้งานได้ เช่น ในตัวอย่างที่ 1 จะเห็นว่าการเรียกใช้ไฟล์ **stdio.h** ซึ่งในไฟล์ดังกล่าวมีการกำหนดคำสั่ง **printf** ซึ่งถูกเรียกใช้ในฟังก์ชัน **main()** และ ฟังก์ชัน **process** แต่ถ้าเราต้องการเรียกใช้ฟังก์ชัน **system()** ซึ่งไม่มีในไฟล์นี้ แต่มีในไฟล์ **stdlib.h** เราก็จะต้อง **include** ไฟล์ **stdlib.h** เข้ามาด้วย

ตัวอย่างไคเรกทีฟการใช้งาน

#include	Include text from a file
#define	Define a macro
#undef	Undefined a macro
#if	Test if a compile-time condition holed.
#ifdef	Test if a symbol is defined
#ifndef	Test if a symbol is not defined
#else	Indicate alternative if a test file fail.
#line	Give a line number for compiler message.

#define ทำหน้าที่ใช้กำหนดค่าคงที่ ที่เป็นชื่อแทน คำ นิพจน์ คำสั่ง หรือคำสั่งหลายคำสั่ง มีรูปแบบการใช้งานดังนี้



#define ชื่อตัวแปร (ชื่อที่ใช้แทน) ค่าที่ต้องการกำหนด

เช่น #define TEN 10 กำหนดตัวแปร TEN แทนค่า 10

 #define PI 3.141592654 กำหนดตัวแปร PI แทนค่า 3.141592654

3) ส่วนประกาศ (Global Declaration)

ใช้ในการประกาศชื่อตัวแปร หรือฟังก์ชันที่ต้องใช้ในโปรแกรม โดยทุกๆ ส่วนของโปรแกรมสามารถจะเรียกใช้ข้อมูลส่วนนี้ได้

4) ส่วนฟังก์ชันหลัก (main program)

เป็นส่วนที่โปรแกรมภาษาซีทุกโปรแกรมต้องมีในฟังก์ชัน main จะประกอบด้วยคำสั่งต่างๆ ที่ใช้ในการสั่งงานโปรแกรมเรียงต่อกันโดยแต่ละประโยคคำสั่งจะจบลงด้วยเครื่องหมาย semi colon (;)

5) ส่วนของการเรียกใช้ฟังก์ชันที่ผู้ใช้สร้างขึ้นเอง (User defined function call)

เป็นส่วนของการเรียกใช้ฟังก์ชันที่ผู้ใช้สร้างขึ้นเอง ซึ่งนอกเหนือจากฟังก์ชันโปรแกรมภาษาซีมีไว้ให้



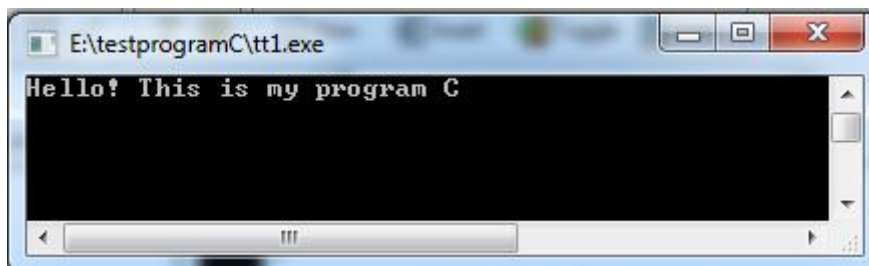


แบบฝึกทักษะที่ 1

1. จงพิจารณาข้อความต่อไปนี้ จัดอยู่ในขั้นตอนการทำงานอะไร

```
#include<stdio.h>
main()
{
    printf("Hello! This is my program C \n");
    getch();
}
```

2. ภาพต่อไปนี้ จัดอยู่ในขั้นตอนการทำงานใด



3. การทำงานในขั้นตอนใดเมื่อเสร็จสิ้นการทำงานจะได้ไฟล์ที่มีนามสกุล .obj

4. การทำงานในขั้นตอนใดเมื่อเสร็จสิ้นการทำงานจะได้ไฟล์ที่มีนามสกุล .c

5. การทำงานในขั้นตอนใดเมื่อเสร็จสิ้นการทำงานจะได้ไฟล์ที่มีนามสกุล .exe



6. ให้นักเรียนเปิดโปรแกรม Dev-C++ และป้อนโปรแกรมต่อไปนี้

```
#include<stdio.h>
main()
{
    printf("Hello! PCCCR \n");
    getch();
}
```

ผลการรัน

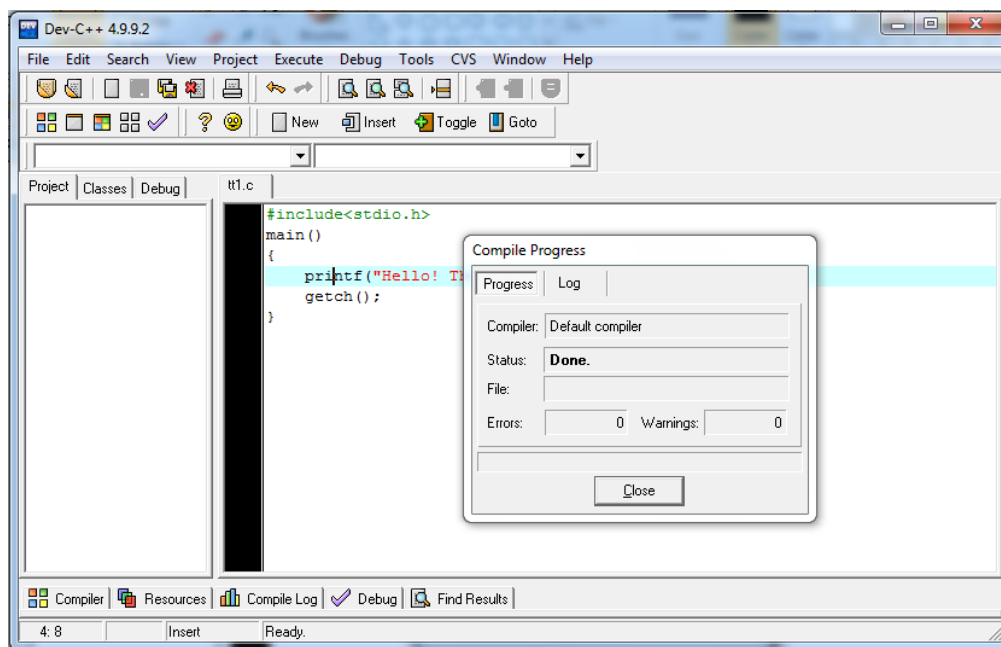
.....
.....
.....
.....
.....

7. บันทึกโปรแกรม (save) โดยเลือกเมนู File->Save หรือกด Ctrl+S

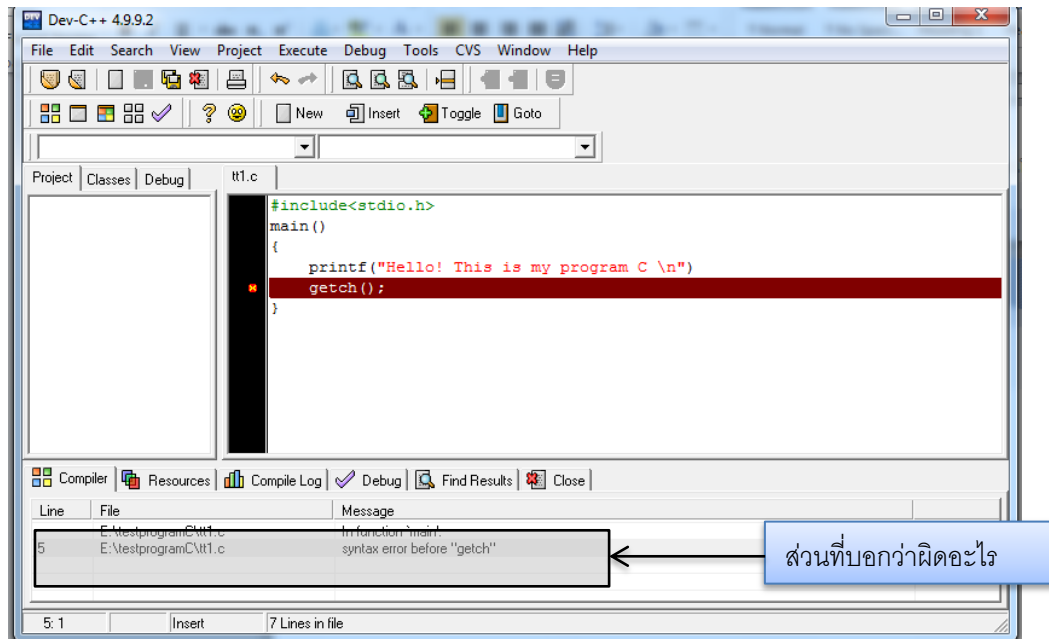
- เปลี่ยนไดรฟ์เป็นไดรฟ์ D
- สร้างไฟล์เดอร์ใหม่ ชื่อ ProgramC-xxx (xxx หมายถึง ระดับชั้น เช่น ม.4/2 -> 402)
- พิมพ์ชื่อไฟล์ program01.c
- คลิกปุ่ม Save

8. แปลโปรแกรม (Compile) โดยเลือกเมนู Execute -> Compile หรือกด Ctrl+F9

ถ้าโปรแกรมไม่ผิดพลาดจะได้ผลดังรูป



ในกรณีที่พิมพ์โปรแกรมผิดจะแสดงรายการข้อผิดพลาดที่ตรวจพบ ดังรูป



9. สั่งให้โปรแกรมทำงาน (Run) เมื่อโปรแกรมได้ถูกแปลเป็นโปรแกรมภาษาเครื่องเรียบร้อยแล้ว ผู้ใช้สามารถสั่งให้โปรแกรมที่แปลแล้วกระทำการโดยเลือกคำสั่ง Execute -> Run หรือกด Ctrl+F10

10. บันทึกผลการรันในข้อ 6

11. โครงสร้างของภาษาซีในส่วนของคำสั่งพรีโพรเซสเซอร์ ทำหน้าที่อะไร

.....

.....

12. ในภาษาซี ฟังก์ชันหลักที่ทุกโปรแกรมต้องมี คือฟังก์ชันอะไร

.....

.....

13. การเขียนคำสั่งพรีโพรเซสเซอร์ ต้องเริ่มต้นด้วยเครื่องหมายอะไร

.....

.....

14. คำสั่งพรีโพรเซสเซอร์ #include ทำหน้าที่อะไร

.....

.....

15. คำสั่งพรีโพรเซสเซอร์ #define ทำหน้าที่อะไร

.....

.....



16.ประโยคคำสั่งในภาษาซีต้องจบด้วยเครื่องหมายอะไร

.....
.....

17.ประโยชน์ของการเขียนหมายเหตุไว้ในโปรแกรม คืออะไร

.....
.....

18.การเขียนหมายเหตุในภาษาซี มีหลักการเขียนอย่างไร

.....
.....

19.รูปแบบการเขียนคำสั่ง #include<ชื่อไฟล์> มีความหมายอย่างไร

.....
.....

20. รูปแบบการเขียนคำสั่ง #include<ชื่อไฟล์> มีความหมายอย่างไร

.....
.....

