บทที่ 1
แนะนำภาษา C++ เบื้องต้น

♦ ประวัติภาษา C , C++

ค.ศ. 1970 มีการพัฒนาภาษา B โดย Ken Thompson ซึ่งทำงานบนเครื่อง DEC PDP-7 ซึ่งทำงานบนเครื่องไมโครคอมพิวเตอร์ไม่ได้ และยังมีข้อจำกัดในการใช้งานอยู่ (ภาษา B สืบทอดมาจากภาษา BCPL ซึ่งเขียนโดย Marth Richards)

ค.ศ. 1972 Dennis M. Ritchie และ Ken Thompson ได้สร้างภาษา C เพื่อเพิ่มประสิทธิภาพภาษา B ให้ดียิ่งขึ้น ในระยะแรกภาษา C ไม่เป็นที่นิยมแก่นักโปรแกรมเมอร์โดยทั่วไปนัก

ค.ศ. 1978 Brian W. Kernighan และ Dennis M. Ritchie ได้เขียนหนังสือเล่มหนึ่งชื่อว่า The C Programming Language และหนังสือเล่มนี้ทำให้บุคคลทั่วไปรู้จักและนิยมใช้ภาษา C ในการเขียนโปรแกรมมากขึ้น

แต่เดิมภาษา C ใช้ Run บนเครื่องคอมพิวเตอร์ 8 bit ภายใต้ระบบปฏิบัติการ CP/M ของ IBM PC ซึ่งในช่วงปี ค. ศ. 1981 เป็นช่วงของการพัฒนาเครื่องไมโครคอมพิวเตอร์ ภาษา C จึงมีบทบาทสำคัญในการนำมาใช้บนเครื่อง PC ตั้งแต่นั้นเป็นต้นมา และมีการพัฒนาต่อมาอีกหลาย ๆ ค่าย ดังนั้นเพื่อกำหนดทิศทางการใช้ภาษา C ให้เป็นไปแนวทางเดียวกัน ANSI (American National Standard Institute) ได้กำหนดข้อตกลงที่เรียกว่า 3J11 เพื่อสร้างภาษา C มาตรฐานขึ้นมา เรียนว่า ANSI C

ค.ศ. 1983 Bjarne Stroustrup แห่งห้องปฏิบัติการเบล (Bell Laboratories) ได้พัฒนาภาษา C++ ขึ้นรายละเอียดและความสามารถของ C++ มีส่วนขยายเพิ่มจาก C ที่สำคัญ ๆ ได้แก่ *แนวความคิดของการเขียนโปรแกรมแบบกำหนดวัตถุเป้าหมายหรือแบบ OOP (Object Oriented Programming)* ซึ่งเป็นแนวการเขียนโปรแกรมที่เหมาะกับการพัฒนาโปรแกรมขนาดใหญ่ที่มีความสลับซับซ้อนมาก มีข้อมูลที่ใช้ในโปรแกรมจำนวนมาก จึงนิยมใช้เทคนิคของการเขียนโปรแกรมแบบ OOP ในการพัฒนาโปรแกรมขนาดใหญ่ในปัจจุบันนี้

♦ ข้อดีของภาษา C และ C++

โปรแกรมเมอร์โดยทั่วไปในปัจจุบันนิยมพัฒนาโปรแกรมด้วยภาษา C และ C++ ด้วยเหตุผลดังนี้

1. โปรแกรมเมอร์สามารถสร้างโปรแกรมที่ควบคุมการทำงานของคอมพิวเตอร์และการโต้ตอบระหว่างผู้ใช้กับคอมพิวเตอร์ได้อย่างเต็มประสิทธิภาพ เช่น การเขียนโปรแกรมในลักษณะที่ผู้ใช้ควบคุมโปรแกรมในสภาพแวดล้อม ที่เป็น Event-Driven คือ ผู้ใช้สามารถควบคุมเหตุการณ์ต่าง ๆ ของโปรแกรม

ในขณะทำงานได้ไม่ใช่ผู้ใช้ถูกควบคุมโดยโปรแกรม ลักษณะการทำงานแบบ Event-Driven ได้แก่ โปรแกรมที่ทำงานในสภาพแวดล้อมภายใต้ระบบปฏิบัติการวินโดวส์ เป็นต้น

2. ภาษา C และ C++ มีประสิทธิภาพของภาษาอยู่ในระดับที่ใกล้เคียงกับภาษา Assembly มากที่สุด แต่มีความยืดหยุ่นในยึดติดกับฮาร์ดแวร์คอมพิวเตอร์หรือ Microprocessor รุ่นใดรุ่นหนึ่ง ทำให้สามารถนำโปรแกรมที่สร้างขึ้นไปทำงานได้กับเครื่องคอมพิวเตอร์ได้ทุกรุ่น

4. ภาษา C++ สนับสนุนการเขียนโปรแกรมในลักษณะเชิงวัตถุหรือ OOP (Object Oriented Programming) ซึ่งเป็นเทคนิคการเขียนโปรแกรมที่นิยมใช้เขียนโปรแกรมขนาดใหญ่ที่มีจำนวนข้อมูลในโปรแกรมมาก

5. โปรแกรมเมอร์ส่วนใหญ่จะนิยมใช้ภาษา C, C++ พัฒนาโปรแกรมประยุกต์ในงานด้านต่าง ๆ เป็นจำนวนมากในปัจจุบัน เพราะประสิทธิภาพของภาษาที่ได้เปรียบภาษาอื่น ๆ

♦ ขั้นตอนการพัฒนาโปรแกรมด้วย C++

การพัฒนาโปรแกรมด้วยภาษา C++ มีขั้นตอนในการสร้างคล้ายกับภาษาระดับสูงทั่วไป แต่ภาษา C++ ได้จัดเตรียมเครื่องมือในการพัฒนาโปรแกรมในสภาพแวดล้อมที่รวมไว้ด้วยกันแบบเบ็ดเสร็จ ที่เรียกว่า **IDE (Integrated Development Environment)** คือ ได้นำเครื่องมือที่จำเป็นทั้งหมดในการพัฒนาโปรแกรมมารวมไว้ ด้วยกัน ทั้ง Editor, Compiler, Link Library และ Help เพื่อความสะดวกของผู้ใช้ในขณะทำการพัฒนาโปรแกรม

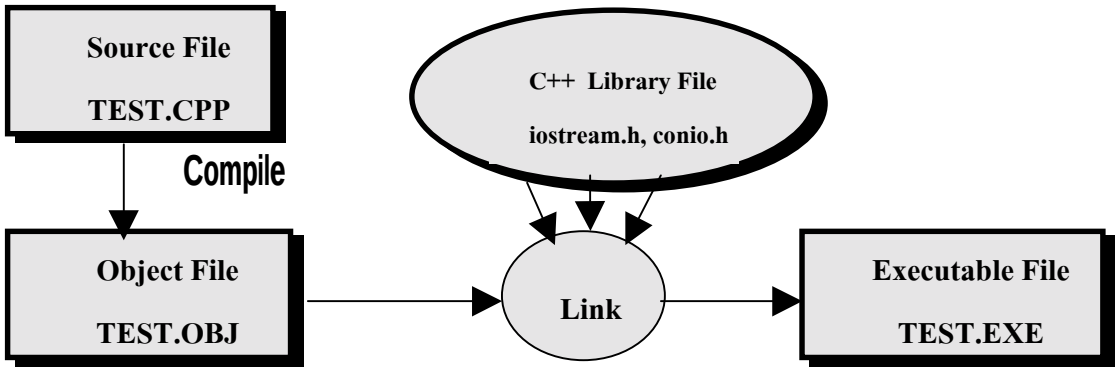
การพัฒนาโปรแกรมด้วยภาษา C++ มีขั้นตอนตามลำดับ ดังนี้

1. ขั้นตอนการสร้าง Source File หรือแฟ้มต้นฉบับเป็น Text File โดยการใช้ส่วน Editor ของ IDE (หรือสร้างจาก Editor ของโปรแกรมอื่น ๆ ก็ได้) เมื่อสร้างเสร็จ แล้วจึงบันทึก Source File ไว้ โดยกำหนดส่วนขยายเป็น CPP เช่น TEST.CPP (C Plus Plus) โดย Source File นี้จะต้องสร้างให้ถูกต้องตามโครงสร้างและไวยากรณ์ของภาษา C++ ทั้งหมดก่อน

2. การคอมไพล์ (Compile) คือการใช้ตัวโปรแกรมหรือ Compiler ของ C++ ในการแปล Source File ให้เป็นไฟล์ภาษาเครื่องที่เรียกว่า Object File หรือ Object Code จะได้ไฟล์ที่มีส่วนขยายเป็น OBJ เพิ่มขึ้นมาอีกหนึ่งไฟล์ เช่น TEST.OBJ

3. การเชื่อมโยง (Linking) เป็นขั้นตอนการเชื่อมโยงไฟล์ประเภท OBJ เข้ากับแฟ้มจากคลัง (Library) ของภาษา C++ จำนวน 1 แฟ้มหรือมากกว่า ซึ่งไฟล์ใน Library นี้จัดเตรียมไว้โดยผู้สร้างภาษา C++ ผลก็คือจะได้ผลลัพธ์เป็นไฟล์ที่สามารถนำไปทำงาน หรือ Run ได้โดยอิสระ หรือที่เรียกว่า Executable File มีส่วนขยายเป็น EXE เช่น TEST.EXE เป็นต้น

ขั้นตอนของการสร้าง Source File , การ Compile และการ Link ทั้งหมดจะดำเนินการได้ใน IDE ของ C++ ที่จัดเตรียมไว้ให้แล้วอย่างอัตโนมัติ ทำให้ผู้เขียนโปรแกรมสามารถทำสร้างโปรแกรมด้วยภาษา C++ สะดวกยิ่งขึ้น



รูปแสดง ขั้นตอนการพัฒนาโปรแกรมด้วย C++

♦ การใช้ IDE ของ C++

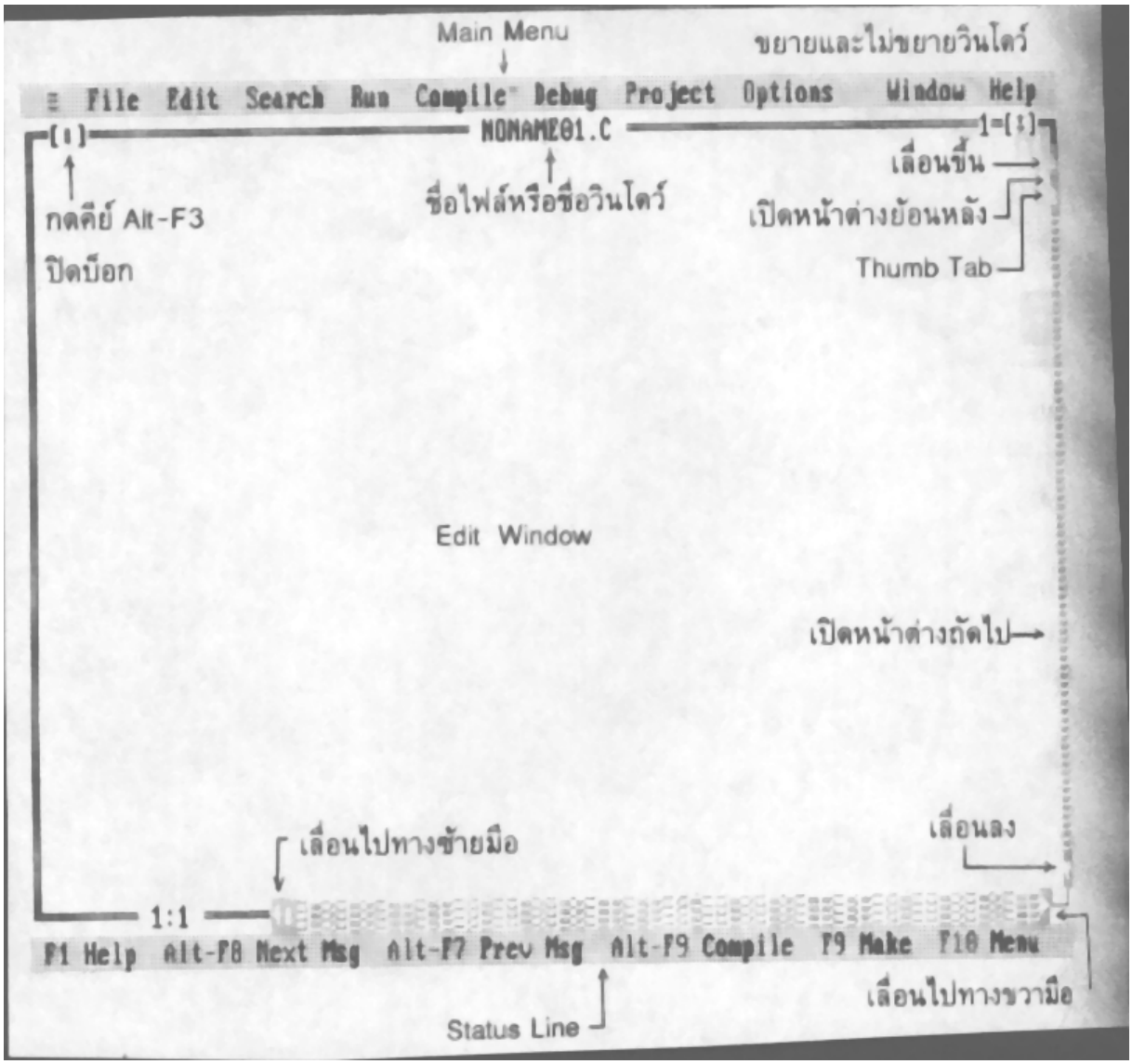
ในการพัฒนาโปรแกรมด้วยภาษา C++ โดยใช้ Turbo C++ Version 3.0 มี ซึ่งได้จัดเตรียมเครื่องมือในการพัฒนาโปรแกรมที่เรียกว่า IDE มาให้แล้ว ลักษณะการใช้งานใน IDE มีความคล้ายคลึงกับ Editor ของภาษาระดับสูงอื่น ๆ เช่น BASIC, PASCAL

การเรียกใช้ IDE ของ C++ ให้เข้าเรียกใช้ไฟล์ TC.EXE ดังนี้

C:\TC>TC แล้วกดแป้น Enter หรือ

C:\TC\BIN>TC แล้วกดแป้น Enter (กรณี TC.EXE อยู่ใน path C:\TC\BIN)

โปรแกรมสภาพแวดล้อมของ IDE ดังนี้จอภาพต่อไปนี้

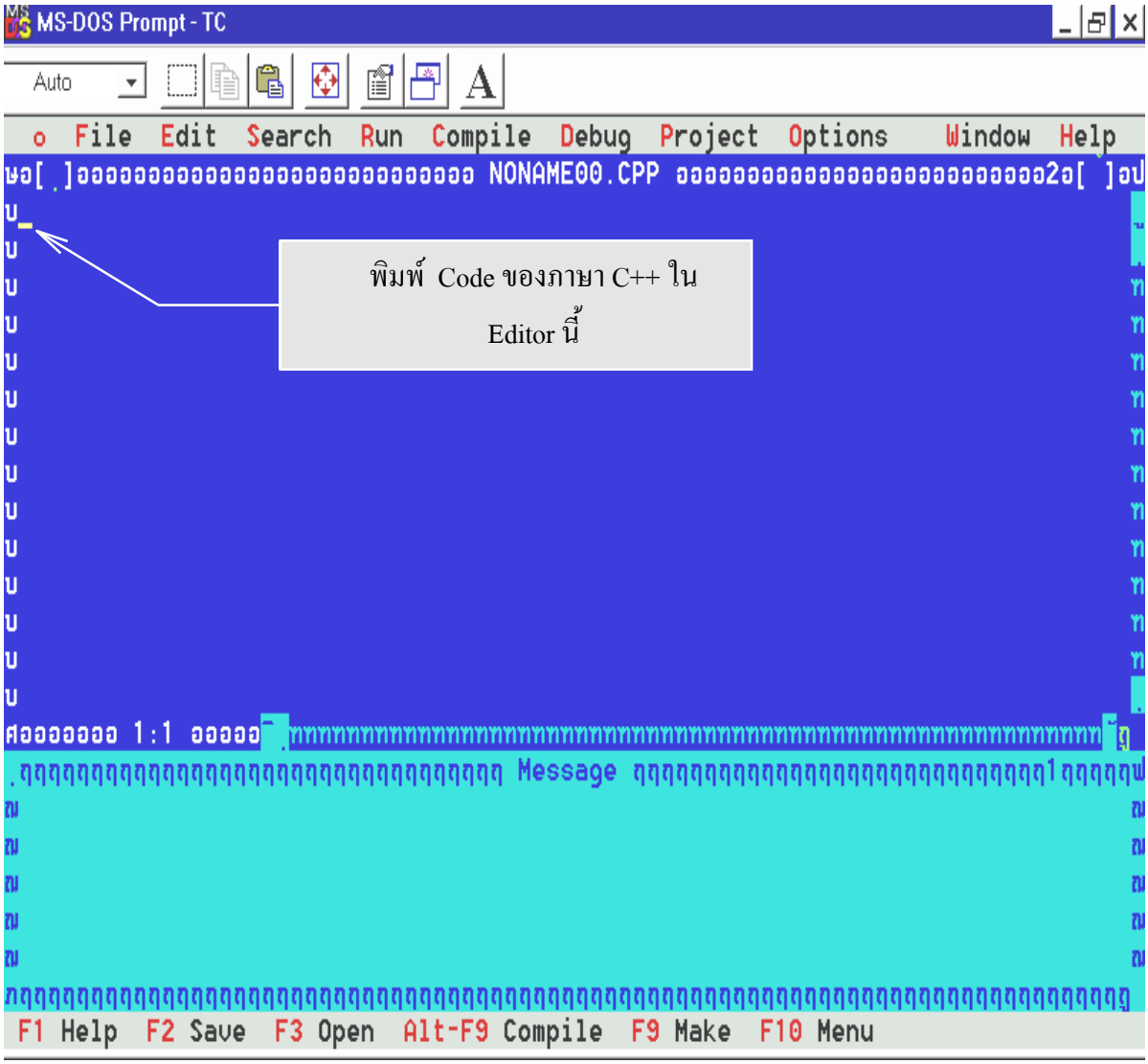


การเปิด IDE ของ TURBO C++ version 3.0 ในระบบปฏิบัติการ Windows'95 ในลักษณะหน้าต่างแบบ Graphic ก็สามารถทำงานได้เช่นเดียวกัน โดยการเลือกโปรแกรม **MS-DOS Prompt** จากเมนูของ Windows'95 เมื่อเครื่องหมาย prompt ของ DOS ปรากฏขึ้นแล้ว ก็ให้เปลี่ยน directory ไปยัง path ของ Turbo C++ ดังนี้

C:\>cd\tc [enter]

C:\TC>TC [enter] เพื่อ run program

จะปรากฏ IDE ของ Turbo C++ ดังภาพข้างล่างในรูปแบบ Graphic windows (ถ้าไม่เป็นดังภาพให้ กดแป้น **Alt+Enter** เพื่อสลับการแสดงผลระหว่าง Graphic window กับ IDE แบบเดิม



♦ ตัวอย่างวิธีการเขียนโปรแกรมและ Compile ด้วย C++

จากตัวอย่าง Source File ต่อไปนี้ ให้ทดลองสร้างโปรแกรมภาษา C++ โดยใช้ IDE ของ Turbo C++ 3.0 ของบริษัท Borland International, Inc. (หรืออาจใช้ C++ ของบริษัทอื่น ๆ)

```
/*Program :   First.CPP
  Written by: Mr.Sirichai
  Date : 10/1997 */
  #include <iostream.h>
  void main(void)
  {
    cout << "My name is Mr.Sirichai Namburi \n";
    cout << "Office : Computer Department,RIPA";
  }
```

ขั้นที่ 1 สร้าง Source File ใน Editor โดยการ Click Mouse ในหน้าต่าง Editor แล้วพิมพ์ รหัสคำสั่งของภาษา C++ ตามตัวอย่างให้ถูกต้อง

ขั้นที่ 2 บันทึกแฟ้ม Source File โดยใช้คำสั่งเมนู File, Save เลือกหรือพิมพ์ชื่อไดรฟ์และไดเรกทอรี่ จากนั้นพิมพ์ชื่อไฟล์ First.CPP แล้ว Click ปุ่มคำสั่ง OK หรือใช้ฟังก์ชันคีย์ F2 เพื่อสั่งบันทึกแฟ้มก็ได้ (ถ้าต้องการเขียนเป็นภาษา C ให้บันทึกเป็นไฟล์เป็นประเภท C เช่น First.C)

ขั้นที่ 3 ใช้คำสั่งสั่ง Run, Run หรือ Ctrl+F9 เพื่อทำการ Compile , Link และทดลอง Run โปรแกรมที่สร้างขึ้นเพื่อทำงาน จากนั้นกดแป้น **Alt+F5** เพื่อดูผลการทำงานของโปรแกรมในหน้าต่างผลลัพธ์ (User Screen)

ผลการทำงานจากโปรแกรม หรือผลการ RUN ได้ดังนี้

My name is Mr.Sirichai Namburi
Office : Computer Department,RIPA

ขั้นที่ 4 กรณีมีข้อผิดพลาด IDE ของ C++ จะไม่สามารถทำการ Compile ได้ จะแจ้งข่าวสารข้อผิดพลาดไว้ในกรอบหน้าต่าง Message จะต้องแก้ไขให้ถูกต้องก่อน แล้วจึงดำเนินการในขั้นที่ 3 ใหม่

♦โครงสร้างของโปรแกรม C++

การเขียนโปรแกรมด้วย C++ มีโครงสร้างของโปรแกรมพื้นฐานดังตัวอย่าง

```
/*Program :   First.CPP
  Written by: Mr.Sirichai
  Date : 10/1997 */
```
← 1. หมายเหตุหรือคำอธิบาย (comments)

```
#include <iostream.h>
```
← 2. Preprocessor, Directive, header file

```
void main()
```
← 3. Function main()

```
{
  cout << "My name is Mr.Sirichai Namburi \n";
  cout << "Office : Computer Department,RIPA";
}
```
← 4. Statement ของคำสั่งอยู่ใน block ของเครื่องหมาย{ …. }แต่ละ Statement จบด้วย ;

ส่วนประกอบเบื้องต้นของ C++ มีดังนี้

1. Comments or Remark หมายถึงส่วนที่เป็นการอธิบายหรือหมายเหตุในโปรแกรม เขียนอธิบายไว้ในเครื่องหมาย /* ……*/ หรือเขียนตามหลังเครื่องหมาย // ก็ได้ ในขณะที่แปล Compiler ของ C++ จะไม่นำไปแปลด้วย แต่ต้องเขียน Comments อยู่ภายในเครื่องหมายให้ถูกต้อง โดยที่ /*…..*/

มักใช้กับ Comment หลาย ๆ บรรทัด ส่วน // ใช้กับการ Comments ตามหลัง Statement เป็นส่วนใหญ่ เช่น

```
/*Program :   First.CPP
 Written by: Mr.Sirichai
Date : 10/1997 */
```

หมายเหตุหรือคำอธิบาย

หรือ

```
cout << "My name is Mr.Sirichai Namburi \n";       // display text to screen
```

2. #include <iostream.h> บรรทัดที่ขึ้นต้นด้วย # นี้จะต้องมีเสมอในทุกโปรแกรม เรียกว่า preprocessor เรียกคำว่า include ที่ตามเครื่องหมาย # นี้ว่า directive และชื่อไฟล์ที่อยู่ในเครื่องหมาย <…..> (จะใช้เครื่องหมาย "……" แทนก็ได้) เรียกว่า header file หรือ include file ซึ่งเป็นไฟล์ที่เก็บไว้ในคลังคำสั่ง (Library File) ของ C++

<u>ข้อสังเกต</u> การเขียน preprocessor directive จะต้องเขียนรายการละ 1 บรรทัด และไม่ต้องมีเครื่องหมาย ; ที่ท้ายประโยค

#include <iostream.h> หมายถึง การสั่งให้ Compiler นำสิ่งที่อยู่ในไฟล์ที่กำหนดชื่อมาให้ คือไฟล์ iostream.h มารวมกับ source file ขณะทำการ link เพื่อให้ได้ Executable file นั่นหมายความว่า ในโปรแกรมที่เราสร้างขึ้น ได้มีการเรียกใช้ฟังก์ชันที่ถูกเก็บไว้ใน Header File นั้น

3. void main() เป็นการเรียกใช้ฟังก์ชันหลักของโปรแกรมคือ ฟังก์ชัน main() ซึ่งจะต้องมีชื่อฟังก์ชันนี้เสมอ ฟังก์ชัน main() เป็นฟังก์ชันหลัก จะประกอบไปด้วยวงเล็บเปิด { เป็นการเริ่มต้นภายในมีการประกาศตัวแปร มีประโยคคำสั่งของภาษา C++ มีชื่อฟังก์ชันอื่น ๆ ที่ผู้เขียนสร้างขึ้นแล้วเรียกใช้ภายในฟังก์ชัน main() แล้วจบฟังก์ชันด้วยวงเล็บปิด }

คำว่า void เป็นชื่อ ***ประเภทข้อมูล(data type)*** ที่ให้ค่าว่าง จะทำให้ฟังก์ชันไม่มีการส่งค่าใด ๆ กลับไปยังชื่อฟังก์ชันที่ถูกเรียกใช้ ทั้งนี้ เนื่องจากใน C++ เมื่อมีการเรียกใช้ฟังก์ชันใดฟังก์ชันหนึ่ง เมื่อฟังก์ชันทำงานเสร็จแล้ว จะต้องส่งค่าคืนกลับมายังจุดที่เรียกใช้ชื่อฟังก์ชันเสมอ เพื่อไม่ให้ส่งคืนค่าใด ๆ กลับมา จึงใช้คำว่า **void** เพื่อกำหนด main() ให้เป็นฟังก์ชันที่ไม่ต้องคืนค่ากลับมา ณ จุดเรียกใช้หรือเป็นฟังก์ชันประเภทไม่มีค่านั่นเอง

4. cout << "My name is Mr.Sirichai Namburi \n";

cout << "Office : Computer Department,RIPA";

เป็นส่วนของประโยคคำสั่งหรือ Statement ในภาษา C++ ซึ่งต้องเขียนให้ถูกต้องตามไวยากรณ์ของภาษา ทุกประโยคต้องจบด้วยเครื่อง semicolon (;) เสมอ สำหรับคำว่า cout เป็น object ซึ่งถูกเก็บไว้ในไฟล์ iostream.h ดังนั้นจึงต้องกำหนดชื่อไฟล์ iostream.h ไว้ในส่วนของ preprocessor directive ด้วย

header file ที่สามารถใช้ร่วมกับ #include ใน C++ เพื่อให้สามารถเรียกใช้คำสั่งหรือฟังก์ชันต่าง ๆ ที่ผู้ใช้ต้องการได้ ได้แก่รายชื่อไฟล์ในตารางต่อไปนี้ โดยก่อนที่จะเรียกใช้ header file ใดนั้น ผู้ใช้จะต้องตรวจสอบก่อนว่าฟังก์ชันที่ต้องเรียกใช้ถูกสร้างไว้ใน header file ใด เช่น ถ้ามีการเรียกใช้ฟังก์ชัน getch() ในโปรแกรม จะต้องเขียน preprocessor directive เรียกใช้ header file ที่ชื่อ conio.h เนื่องจากฟังก์ชัน getch() ถูกเก็บไว้ในไฟล์ conio.h ซึ่งเป็นคลังคำสั่ง (Library) ของ C++ มีรูปแบบการเรียกใช้ดังนี้

#include <conio.h>

ตาราง แสดงรายชื่อ Header file ใน C++ และรายละเอียดกลุ่มฟังก์ชันที่เรียกใช้ได้ในแต่ละ herder file

Header File	Groups of Functions in Header file
alloc.h	Declares memory management functions (allocation, deallocation, etc.).
assert.h	Defines the assert debugging macro.
Bcd.h	Declares the C++ class bcd and the overloaded operators for bcd and bcd math functions.
Bios.h	Declares various functions used in calling IBM-PC ROM BIOS routines.
complex.h	Declares the C++ complex math functions.
Conio.h	Declares various functions used in calling the DOS console I/O routines.
ctype.h	Contains information used by the character classification and character conversion macros.
Dir.h	Contains structures, macros, and functions for working with directories and path names.
Direct.h	Defines structures, macros, and functions for working with directories and path names.
Dirent.h	Declares functions and structures for POSIX directory operations.
Dos.h	Defines various constants and gives declarations needed for DOS and 8086-specific calls.
Errno.h	Defines constant mnemonics for the error codes.
Fcntl.h	Defines symbolic constants used in connection with the library routine open.
Float.h	Contains parameters for floating-point routines.
fstream.h	Declares the C++ stream classes that support file input and output.

Header File	Groups of Functions in Header file
generic.h	Contains macros for generic class declarations.
graphics.h	Declares prototypes for the graphics functions.
io.h	Contains structures and declarations for low-level input/output routines.
iomanip.h	Declares the C++ streams I/O manipulators and contains macros for creating parameterized manipulators.
iostream.h	Declares the basic C++ (version 2.0) streams (I/O) routines.
Limits.h	Contains environmental parameters, information about compile-time limitations, and ranges of integral quantities.
locale.h	Declares functions that provide country- and language-specific information.
malloc.h	Memory management functions and variables.
math.h	Declares prototypes for the math functions, defines the macro HUGE_VAL, and declares the exception structure used by matherr.
mem.h	Declares the memory-manipulation functions. (Many of these are also defined in string.h.)
memory.h	Memory manipulation functions.
new.h	Access to operator new and newhandler.
process.h	Contains structures and declarations for the spawn... and exec... functions.
search.h	Declares functions for searching and sorting.
setjmp.h	Defines a type used by longjmp and setjmp.
share.h	Defines parameters used in functions that use file-sharing.
signal.h	Defines constants and declarations for signal and raise.
stdarg.h	Defines macros used for reading the argument list in functions declared to accept a variable number of arguments.
stddef.h	Defines several common data types and macros.
stdio.h	Defines types and macros needed for the Standard I/O Package defined in Kernighan and Ritchie and extended under UNIX System V. Defines the standard I/O predefined streams stdin, stdout, stdprn, and stderr, and declares stream-level I/O routines.

Header File	Groups of Functions in Header file
stdiostr.h	Declares the C++ (version 2.0) stream classes for use with stdio FILE structures.
stdlib.h	Declares several commonly used routines: conversion routines, search/sort routines, and other miscellany.
stream.h	Declares the C++ (version 1.2) streams (I/O) routines.
string.h	Declares several string- and memory-manipulation routines.
strstrea.h	Declares the C++ stream classes for use with byte arrays in memory.
sys\locking.h	Definitions for mode parameter of locking function.
sys\stat.h	Defines symbolic constants used for opening and creating files.
sys\timeb.h	Declares the function ftime and the structure timeb that ftime returns.
sys\types.h	Declares the type time_t used with time functions.
time.h	Defines a structure filled in by the time-conversion routines, and a type used by other time routines; also provides prototypes for these routines.
utime.h	Declares the functions utime and the structure utimbuf
values.h	Defines important constants, including machine dependencies; provided for UNIX System V compatibility.
varargs.h	Defines old style marcos for processing variable argumnet lists. Superceded by stdarg.h

หมายเหตุ ถ้าต้องการทราบว่า header file ประกอบด้วยฟังก์ชั่นใดบ้าง ให้ใช้คำสั่ง Help, Index จากเมนูใน IDE ของภาษา C++ เพื่อค้นหารายละเอียดและตัวอย่างการใช้ฟังก์ชัน

♦ไอเด็นติฟายเออร์ (identifier) ใน C++

ไอเด็นติฟายเออร์ (identifier) หมายถึง ชื่อที่มีอยู่ในส่วนต่าง ๆ ของโครงสร้างโปรแกรม C++ ซึ่งได้แก่ ชื่อของ เลเบล (label) คอนสแตนต์ (constant) แวเรียเบิลหรือตัวแปร (variable) ฟังก์ชัน (function) และชนิดของข้อมูล (data type)

ประเภทของไอเด็นติฟายเออร์ มี 3 ประเภท คือ keyword , standard identifier และ user-defined identifier มีรายละเอียด ดังนี้

1. Keyword เป็นชื่อที่มีความหมายและวิธีการใช้แน่นอน ได้กำหนดไว้ในภาษา C++ แล้ว คอมไพเลอร์จะไม่ยอมให้เราใช้ชื่อนี้ในลักษณะที่แตกต่างไปจากที่กำหนดไว้ ตัวอย่างของ keyword เช่น void if else int char float case auto return

2. Standard Identifier หมายถึง ชื่อที่กำหนดขึ้นในคอมไพเลอร์ ชื่อเหล่านี้มีความหมายและวิธีใช้ตามเงื่อนไขที่คอมไพเลอร์กำหนดไว้ แต่เราสามารถเปลี่ยนแปลงวิธีการทำงานและเงื่อนไขการใช้ชื่อเหล่านี้ได้ โดยคอมไพเลอร์จะยกเลิกเงื่อนไขเดิมและเปลี่ยนมาใช้เงื่อนไขที่เรากำหนดขึ้นใหม่ standard identifier ส่วนใหญ่จะเป็นชื่อฟังก์ชันที่มีอยู่ใน C++ เช่น abort, abs, arc, ftime, getch, open, rename เป็นต้น

3. User-defined identifier หมายถึง ชื่อที่เรากำหนดความหมายและเงื่อนไขในการใช้ขึ้นเองโดยผู้ใช้ แต่ตั้งกำหนดขึ้นตามกฎเกณฑ์ของ C++ ซึ่งมีรายละเอียดของกฎการตั้งชื่อ ดังนี้

- อักขระตัวแรกต้องเป็นตัวอักษรหรือ underscore (_) จะเป็นตัวเลขไม่ได้ ตัวอักขระตัวต่อไปจะเป็นตัวอักษร ตัวเลข หรือเครื่องหมาย _ ก็ได้ เรียงกันโดยห้ามมีช่องว่างภายในชื่อ
- ชื่อห้ามซ้ำกับคีย์เวิร์ด (Keywords) ของภาษา C++ เช่น main void if
- คอมไพเลอร์จะถือว่าอักษรพิมพ์เล็กและพิมพ์ใหญ่ ***มีความแตกต่างกัน*** ดังนั้น Identifier ชื่อFIRST_PROGRAM กับ first_program จะถือว่าเป็นชื่อต่างกันและเป็นคนละชื่อกัน
- ชื่อมีความยาวไม่จำกัด แต่จะมีความหมายเฉพาะอักขระ 32 ตัวแรกเท่านั้น แต่ควรตั้งชื่อให้มีความหมายสอดคล้องกับวัตถุประสงค์การนำชื่อนั้นไปใช้ภายในโปรแกรม เพื่อความสะดวกในการจดจำในขณะเขียนโปรแกรม

♦ ชนิดของข้อมูล(Data Type)

ข้อมูลใน C++ แบ่งชนิดข้อมูลออกเป็น 2 ประเภท คือ

1. Simple data type เป็นชนิดข้อมูลที่ใช้แสดงค่าของสิ่งใดสิ่งหนึ่งเพียงรายการเดียว เช่น ค่าความสูง น้ำหนัก จำนวนนักเรียน อุณหภูมิ ระดับคะแนน เป็นต้น

2. Structure เป็นข้อมูลชนิดใช้แสดงค่าของสิ่งใดสิ่งหนึ่งหลายรายการ เช่น ความสูงของนักเรียนใน ชั้น ม. 6, อุณหภูมิของแต่ละวันในเดือนตุลาคม, รายชื่อนักเรียนใน 1 กลุ่ม ต้องกำหนดเป็นข้อมูลชนิดโครงสร้างแบบ อาร์เรย์ (array) แบบโครงสร้าง(structure) หรือแบบยูเนียน(union) เป็นต้น

ข้อมูล Simple data type รายละเอียดชนิดของมูลและช่วงของข้อมูลประเภท Simple data type แสดงได้ดังตารางต่อไปนี้

ตาราง แสดงชื่อชนิดของข้อมูล ช่วงของค่าข้อมูลและขนาดหน่วยความจำที่ใช้

ชนิดข้อมูล	ค่าต่ำสุด	ค่าสูงสุด	ใช้พื้นที่หน่วยความจำ
char	-128	127	1 byte
unsigned char	0	255	1 byte
int	-32,768	32,767	2 byte
unsigned int	0	65,535	2 byte
short int	-32,768	32,767	2 byte
long	-2,147,483,648	2,147,483,647	4 byte
unsigned long	0	4,294,967,295	4 byte
float	$3.4x10^{-38}$	$3.4x10^{+38}$	4 byte
double	$1.7x10^{-308}$	$1.7x10^{+308}$	8 byte
long double	$3.4x10^{-4932}$	$3.4x10^{+4932}$	10 byte

หมายเหตุ ชื่อชนิดของข้อมูลได้แก่ char, unsigned char , int , unsigned int, short in, long , unsigned long , float , double , long double เป็น keyword ที่นำไปกำหนดประเภทข้อมูลที่จะใช้ในโปรแกรม

- ***ตัวอย่างโปรแกรม size_mem.cpp เป็นตัวอย่างโปรแกรมแสดงขนาดของหน่วยความจำที่ข้อมูลแต่ละชนิดใช้พื้นที่หน่วยความจำ โดยใช้ คีย์เวิร์ด sizeof ซึ่งให้ค่าขนาดหน่วยความจำที่ข้อมูลชนิดนั้นใช้ มีหน่วยเป็น byte***

```
/*Program : size_mem.cpp
  Process : Display size of memory for each simple data type
*/
#include <iostream.h>
#include <conio.h>
void main()
{
  clrscr(); //clear screen in conio.h
  cout<< "Size of char =          "<<sizeof(char)<< " bytes"<<endl;
  cout<< "Size of unsigned char = "<<sizeof(unsigned char)<< " bytes"<<endl;
  cout<< "Size of int =           "<<sizeof(int)<< " bytes"<<endl;
  cout<< "Size of unsigned int =  "<<sizeof(unsigned int)<< " bytes"<<endl;
  cout<< "Size of short int =     "<<sizeof(short int)<< " bytes"<<endl;
  cout<< "Size of long =          "<<sizeof(long)<< " bytes"<<endl;
  cout<< "Size of unsigned long = "<<sizeof(unsigned long)<< " bytes"<<endl;
  cout<< "Size of float =         "<<sizeof(float)<< " bytes"<<endl;
  cout<< "Size of double =        "<<sizeof(double)<< " bytes"<<endl;
  cout<< "Size of long double =   "<<sizeof(long double)<< " bytes"<<endl;
  getch(); //wait for press any key
}
```

♦ การประกาศตัวแปร (Variable Declaration) และการกำหนดค่าให้ตัวแปรใน C++

ตัวแปร (Variables) ในภาษา C++ หมายถึง ชื่อที่กำหนดขึ้นเพื่อใช้เก็บค่าของข้อมูลหรือค่าคงที่ประเภทต่าง ๆ ในขณะโปรแกรมทำงาน ซึ่งผู้เขียนโปรแกรมต้องตั้งชื่อตัวแปรตามกฎเกณฑ์การตั้งชื่อประเภท user defined identifier

การใช้ตัวแปรในภาษา C++ จะต้องมีการประกาศ (Declaration) ชื่อตัวแปรและประเภทของตัวแปร (Data type) ไว้ก่อน จึงจะสามารถนำตัวแปรไปใช้ในโปรแกรมได้ มีรูปแบบการประกาศตัวแปรดังนี้

```
Data_type    variable_name; หรือ
Data_type    variable_name1, variable_name2,varible_name3, … ;
```

โดยที่ Data_type คือ ชื่อของประเภทข้อมูลของตัวแปร ที่สามารถเก็บค่าได้ เช่น int, float

variable_name คือ ชื่อของตัวแปรที่ผู้ใช้กำหนดเอง ถ้าในประเภทนั้นมีมากกว่า 1 ตัวให้ใช้เครื่องหมาย , แยก และจบประโยคด้วยเครื่องหมาย ; (semi-colon)

การประกาศตัวแปรเพื่อใช้ในโปรแกรมของ C++ มี 2 ลักษณะ คือ

1. definition คือ เป็นการประกาศเพื่อกำหนดความหมาย เป็นประโยคคำสั่งที่ประกอบด้วย ชื่อประเภทของตัวแปร และตัวแปร โดยทั่วไปมักจะประกาศไว้ตอนต้น ๆ ของฟังก์ชัน หรือโปรแกรม เช่น

```
#include <iostream.h>
void main()
{  int  number;
   float   sales, purchase;
   char  grade;
   …
}
```

◀—— การประกาศตัวแปรในลักษณะ definition

2. การประกาศแบบกำหนดค่า ณ ตำแหน่งที่ใช้ หมายถึง การประกาศตัวแปร ณ ตำแหน่งที่ต้องการใช้ตัวแปรในโปรแกรม ก็จะประกาศพร้อมกับหนดค่าให้กับตัวแปรทันที ดังตัวอย่างโปรแกรม

ตัวอย่างโปรแกรม pos_var.cpp แสดงวิธีการประกาศตัวแปร ณ จุดที่ต้องการใช้ในโปรแกรม

```
/*Program : pos_var.cpp
  Process : Show declared varaible at any position in program */
#include <iostream.h>
#include <conio.h>
void main()
```

```
{ float sale,price;
 clrscr();
 sale=500.25;
 price=5.25;
 float  total=sale*price;
 cout<< "Total Sale = "<<total;
 getch();
}
```

ประกาศตัวแปร total ณ ตำแหน่งที่ต้องการใช้

วิธีการกำหนดค่าให้แก่ตัวแปรใน C++ ทำได้ ดังนี้

1. การใช้ประโยคคำสั่งเครื่องหมายเท่ากับ (=) โดยกำหนดให้ชื่อตัวแปรที่จะใช้เก็บค่าอยู่ทางซ้ายมือของ **เครื่องหมาย =** ตัวแปรหรือค่าคงที่อยู่ทางด้านขวาของเครื่องหมาย ดังเช่นตัวอย่างโปรแกรมต่อไปนี้

- ***ตัวอย่างโปรแกรม variable.cpp แสดงการกำหนดค่าคงที่ให้กับตัวแปรในโปรแกรม โดยใช้เครื่องหมายเท่ากับ =***

```
/*Program : variable.cpp
  Process : Display set value of varible */
#include <iostream.h>
#include <conio.h>
void main()
{  int number;            //declaration variable
main()
   float sales,purchase;   //declaration variable
  // set value of variable
  number =  50;
  sales = 5000.75;
  purchase = 3500.50;
  clrscr();   // function clear screen from <conio.h>
  //show value from variable
  cout << "Show varlue of variable"<<endl;
  cout << "number = " << number<<endl;
  cout << "sales = " << sales<<endl;
  cout << "purchase = " << purchase;
  getch();   // function getch() for wait to press anykey from <conio.h>
}
```

การประกาศตัวแปรภายในฟังก์ชัน

การกำหนดค่าคงที่ให้กับตัวแปร

<u>หมายเหตุ</u> กรณีเป็นตัวแปรประเภทตัวเลข ค่าคงที่ด้านขวามือห้ามมีเครื่องหมายวรรคตอนใด ๆ ทั้งสิ้น ยกเว้นจุดทศนิยม ส่วนตัวแปรประเภท char กำหนดค่าคงที่ในเครื่องหมาย '_' เช่น ch = 'A';

2. การกำหนดค่าเริ่มต้นให้แก่ตัวแปร เมื่อมีการประกาศใช้ตัวแปรในลักษณะ definition มีการกำหนดค่าคงที่ให้แก่ตัวแปรทันที

- ***ตัวอย่างโปรแกรม varia2.cpp แสดงการกำหนดค่าให้แก่ตัวแปร number, sales, purchase ด้วยเครื่องหมาย = เมื่อประกาศตัวแปรในโปรแกรม***

```
/*Program : varia2.cpp
  Process : Display set value of varible */
#include <iostream.h>
#include <conio.h>
void main()
{ int  number=500;   //declaration and set value variable
  float  sales=500.50,purchase=3500.75;   //declaration and set value variable
  clrscr();   // function clear screen from <conio.h>
  //show value from variable
  cout << "Show varlue of variable"<<endl;
  cout << "number = " << number<<endl;
  cout << "sales = " << sales<<endl;
  cout << "purchase = " << purchase;
  getch();    // function getch() for wait to press anykey from <conio.h>
}
```

ประกาศตัวแปร พร้อมกำหนดค่า

3. การกำหนดค่าตัวแปรโดยการรับค่าทางแป้นพิมพ์ โดยการใช้ฟังก์ชันในการรับข้อมูล (Input) เข้าไปเก็บไว้ในตัวแปร จะกล่าวรายละเอียดต่อไปในบทที่ 2

ตัวแปรประเภท Global และ Local

การประกาศใช้ตัวแปรใน C++ สามารถทำได้ 2 ลักษณะ คือ

1. Global variable คือ ตัวแปรที่กำหนดหรือประกาศไว้นอกฟังก์ชันใด ๆ ทุกฟังก์ชันสามารถนำตัวแปรประเภท Global ไปใช้ได้ทุกฟังก์ชัน เพราะเป็นลักษณะการประกาศแบบสารธรณะ

2. Local variable คือ ตัวแปรที่กำหนดหรือประกาศไว้ในฟังก์ชันใดฟังก์ชันหนึ่ง สามารถนำ
ตัวแปรนั้นไปใช้ได้เฉพาะในฟังก์ชันนั้น ๆ เท่านั้น ฟังก์ชันอื่นไม่สามารถนำไปใช้ได้ เพราะประกาศในลักษณะส่วนตัวหรือเฉพาะที่

<u>หมายเหตุ</u> กรณีที่มีการตั้งชื่อตัวแปรประเภท Global และ Local ซ้ำกัน จะถือว่าเป็นตัวแปรคนละตัวกัน เนื่องจากใช้พื้นที่ในหน่วยความจำในการเก็บข้อมูลในตำแหน่งที่ต่างกัน แต่ถ้ามีการเรียกใช้ชื่อตัวแปร C++ จะนำตัวแปรประเภท Local มาใช้ก่อนเสมอ

- ***ตัวอย่างโปรแกรม glo_loc.cpp แสดงการประกาศตัวแปรประเภท Global และ Local ในโปรแกรม***

```
/*Program : glo_loc.cpp
  Process : define global and locat variable */
  #include <iostream.h>
  #include <conio.h>
char grade;    //defined global variable
  void main()
  {  int midterm,final,total;  //defined local variable in function main()
Local
    clrscr();
main()
    midterm = 30;
    final=50;
    total=midterm+final;
    grade='A';
    cout<< "You get total score "<<total<<"  And get grade = " <<grade;
    getch();
  }
```

ประกาศตัวแปรประเภท Global Variable

ประกาศตัวแปรประเภท Local Variable ในฟังก์ชัน

♦ การกำหนดตัวแปรคงที่(Constant)

ตัวแปรคงที่หรือคอนสแตนต์ (Constant) หมายถึง ตัวแปรที่เก็บค่าคงที่ที่ไม่สามารถเปลี่ยนแปลงได้ การกำหนดคอนสแตนต์ มีจุดประสงค์เพื่อป้องกันมิให้มีการเปลี่ยนแปลงค่าของตัวแปรนั้นในขณะทำงาน สามารถกำหนดได้ทั้งในลักษณะที่เป็น Global และ Local รูปแบบการกำหนดค่าคอนสแตนต์ ใช้คำว่า const นำหน้าประเภทข้อมูลและชื่อคอนสแตนต์ ดังนี้

```
const  ชนิดข้อมูล  ชื่อคอนสแตนต์  =  ค่าคงที่;
```

ตัวอย่างเช่น

```
const  int  Day = 7;
const int  month = 12;
const float PI = 3.1418926;
const float Amount = 1.0E+2;       // คือ 1.0 * 10^2
const float Rate = 1E-3;   // คือ 1 * 10^-3
const char name = 'A';   // ให้ค่ารหัส ASCII ของ  A คือ 65
const char ch = 'B';   // ให้ค่ารหัส ASCII ของ  B คือ 66
```

- ***ตัวอย่างโปรแกรม contst.cpp แสดงการกำหนดและการใช้ constant ในโปรแกรม มีรายละเอียด ดังนี้***

```
/*Program : const.cpp
  Process : Display constant in program
*/
#include <iostream.h>
#include <conio.h>
const   float rate = 0.05;  // Global constant

void main()
{  const  float tax=0.03;  //Local constant
   float  sales,income,total_tax;
   clrscr();
   sales = 8500.50;
   income = sales*rate;
   total_tax=income*tax;
   cout<< "Income = "<< income <<endl;
   cout<< "Total tax = "<< total_tax <<endl;
   getch();
}
```

♦ การดำเนินการทางคณิตศาสตร์ (Arithmetic Operations)

ในการคำนวณทางคณิตศาสตร์ของภาษา C++ มีการใช้เครื่องหมายสำหรับการคำนวณ ดังนี้

ตาราง แสดงเครื่องหมายคำนวณทางคณิตศาสตร์ใน C++

ตัวดำเนินการ	ความหมาย	ตัวอย่าง	ผลลัพธ์
-	การลบ	10-5	5
+	การบวก	10+5	15
*	การคูณ	10*5	50
/	การหาร	10/5	2
%	การหารคิดเฉพาะเศษ	9%2	1
--	การลดค่าครั้งละ 1	--x หรือ x--	x=x-1
++	การเพิ่มค่าครั้งละ 1	++x หรือ x++	x=x+1
+=	การบวกสะสมในตัวแปร	y+=x	y=y+x
-=	การลบค่าจากตัวแปร	y-=x	y=y-x
*=	การคูณค่าจากตัวแปร	y*=x	y=y*x
/=	การหารจากตัวแปร	y/=x	y=y/x
%=	การหารคิดเศษจากตัวแปร	y%=x	y=y%x

ตาราง แสดงลำดับการประมวลผล (Precedence) ของเครื่องหมายคณิตศาสตร์

เครื่องหมายคณิตศาสตร์	ลำดับการประมวลผล	หมายเหตุ

()	1	*ถ้าเครื่องหมายมีลำดับการประมวลผลระดับเดียวกันให้ดำเนินการจากซ้ายไปขวา*
++, -- (ใช้แบบ prefix)	2	
- (เครื่องหมายหน้าตัวเลข)	3	
* / %	4	
+ -	5	
+= -= *= /= %=	6	

ข้อสังเกต การใช้โอเปอเรเตอร์ (Operator) ในการเพิ่มค่า(increment) ++ และลดค่า(decrement) -- มีข้อควรสังเกต ดังนี้

1. การเพิ่มค่าขึ้นทีละ 1 ได้แก่การเพิ่มค่าให้แก่ตัวแปรครั้งละ +1 เช่น

```
count = count +1;
count +=1;
count++;
++count;
```

สำหรับ ++ มีวิธีการใช้ 2 วิธี คือ แบบ prefix และ แบบ postfix มีข้อแตกต่างกัน ดังตัวอย่างต่อไปนี้เป็นการใช้ ++ แบบ prefix

```
price = 5;
volume = 3;
value = price *  ++volume;   // ค่าของ value คือ 20  ค่าของ volume คือ 4
```

จากประโยคคำสั่งนี้ ค่าของ volume จะเพิ่มขึ้น 1 ก่อนที่จะนำไปคูณกับ price แล้วนำไปเก็บไว้ในตัวแปร value

ต่อไปนี้ ลองพิจารณาการใช้ ++ แบบ postfix

```
price = 5;
volume = 3;
value = price *  volume++;   // ค่าของ value คือ 15  ค่าของ volume คือ 4
```

จากประโยคคำสั่งนี้ ค่าของ volume คือ 3 จะถูกนำไปคูณกับ price คือ 5 ก่อนแล้วนำไปเก็บไว้ในตัวแปร value จากนั้นจึงเพิ่มค่าของ volume อีก 1 จึงมีค่าเป็น 4

2. การลดค่าลงทีละ 1 ได้แก่การลดค่าของตัวแปรครั้งละ -1 เช่น

```
count = count -1;
count -=1;
count--;
--count;
```

สำหรับ -- มีวิธีการใช้ 2 วิธี คือ แบบ prefix และ แบบ postfix มีข้อแตกต่างการใช้เช่นเดียวกับการใช้ ++

♦ การเปลี่ยนแปลงชนิดข้อมูลในนิพจน์คณิตศาสตร์

ในนิพจน์ทางคณิตศาสตร์เมื่อมีการใช้เครื่องหมายคำนวณระหว่างข้อมูลหรือตัวแปรที่มีชนิด (Data Type) แตกต่างกัน เช่น ชนิด float บวกกับ ชนิด int ใน C++ จะปรับประเภทข้อมูลของผลลัพธ์ที่ได้จากการคำนวณให้เป็นชนิดข้อมูลที่มีอันดับสูงกว่าโดยอัตโนมัติ ดังแสดงอันดับความสำคัญของชนิดข้อมูลในตารางต่อไปนี้

ตาราง แสดงอันดับความสำคัญ(Precedence) ของข้อมูลประเภท Simple Data Type ชนิดต่าง ๆ

ชนิดข้อมูล	อันดับ	ตัวอย่าง
long double	สูงสุด	int * long = long
double	↓	char + int = int
float	↓	int + float = float
long	↓	int * double = double
int	↓	float + double = double
char	ต่ำสุด	long + long dobule = long double

♦ เครื่องหมายเปรียบเทียบและตรรก (Comparison and Logical Operators)

การเปรียบเทียบ หมายถึง การหาว่าเมื่อนำค่าที่อยู่ทางด้านซ้ายของเครื่องหมายกับค่าที่อยู่ทางขวาของเครื่องหมายเปรียบเทียบ มาเปรียบเทียบกันแล้วจะได้ผลลัพธ์เป็นจริงหรือเท็จ ซึ่งในแต่ละกรณีเมื่อเปรียบเทียบกันแล้ว จะได้ค่าเป็นจริงหรือเป็นเท็จเพียงค่าเดียวเท่านั้น **โดย C++ จะใช้ค่า 0 แทนเท็จ และ 1 แทน จริง** และมีลำดับในการประมวลผลจากซ้ายไปขวา ถ้าไม่มีวงเล็บกำกับไว้ มีเครื่องหมายเปรียบเทียบดังนี้

ตาราง แสดงเครื่องหมายเปรียบเทียบใน C++

เครื่องหมาย	ความหมาย	ตัวอย่าง	ผลลัพธ์
>	มากกว่า	5>1	1 (จริง)
<	น้อยกว่า	5<1	0 (เท็จ)
==	เท่ากับ	5==1	0 (เท็จ)
!=	ไม่เท่ากับ	5!=1	1 (จริง)
>=	มากกว่าหรือเท่ากับ	5>=5	1 (จริง)
<=	น้อยกว่าหรือเท่ากับ	5<=5	1 (จริง)

การหาค่าลอจิก (Logic) หมายถึง การหาผลลัพธ์จาก Logic operator ซึ่งจะให้ค่าผลลัพธ์ออกมาเป็นจริงหรือเท็จกรณีใดกรณีหนึ่งเท่านั้น เครื่องหมายที่เป็น Logical operator ใน C++ แสดงไว้ในตารางดังนี้

ตาราง แสดงเครื่องหมายเปรียบเทียบเชิงตรรก (Logical operator) ใน C++

เครื่องหมาย	ความหมาย	ตัวอย่าง	ผลลัพธ์
&&	AND	(5==5)&&(5>3)	1 (จริง)
||	OR	(5<1)||(5>3)	1 (จริง)
!	NOT	!(5==5)	0 (เท็จ)

ตาราง แสดงค่าเป็นจริงและเท็จในกรณีต่าง ๆ ที่เป็นไปได้ทั้งหมดของ AND, OR , NOT โดยกำหนดให้ A และ B คือประโยคตรรกที่ให้ค่าจริงหรือเท็จ

A	B	!A	A&&B	A||B
จริง	จริง	เท็จ	จริง	จริง
จริง	เท็จ	เท็จ	เท็จ	จริง
เท็จ	จริง	จริง	เท็จ	จริง
เท็จ	เท็จ	จริง	เท็จ	เท็จ

- ***ตัวอย่างโปรแกรม logic_tst.cpp เป็นการแสดงค่าตรรกที่ได้จากการเปรียบเทียบด้วยเครื่องหมายเปรียบเทียบและ ตัวดำเนินการตรรก โดยค่า 0 แทน เท็จ และ 1 แทนจริง ดังนี้***

```
/*Program : logic_tst.cpp
  Process : Test logical value of expression
*/
#include <iostream.h>
#include <conio.h>

void main()
{ clrscr();
  cout<< "Display Logic Operation :\a";
  cout<< "\nLogic value of expression (3==5) :" <<(3==5);
  cout<< "\nLogic value of expression (5==5) :" <<(5==5);
  cout<< "\nLogic value of expression (3<=5) :" <<(3<=5);
  cout<< "\nLogic value of expression (3>=5) :" <<(3>=5);
  cout<< "\nLogic value of expression (3<=5)&&(5>3) :" <<((3<=5)&&(5>3));
  cout<< "\nLogic value of expression ((3<=5)&&(3>5)) :" <<((3<=5)&&(3>5));
  cout<< "\nLogic value of expression ((3<=5)||(3>5)) :" <<((3<=5)||(3>5));
  cout<< "\nLogic value of expression ((8<=5)||(3>=5)) :" <<((8<=5)||(3>=5));
  cout<< "\n\nValue  1 is true,  0 is false ....press any key";
  getch();
}
```

♦ แบบฝึกหัดท้ายบท

1. จงเขียนโปรแกรมเพื่อคำนวณหาผลลัพธ์ของนิพจน์ทางคณิตศาสตร์ต่อไปนี้โดยกำหนดให้ตัวแปร

   A=50 B=30 C=5 D=3 E = 10

   1.1 (A+B)*(E-D)

   1.2 ++D+C*E

   1.3 (25+A)/C+B

   1.4 A*=D

   1.5 20*C+B+++D/2

   1.6 --D+C+B--

   1.7 25*D/5+10

   1.8 A+B--+D

   1.9 C*2+E*5

   1.10 (A*2)+B/C-15

2. จงบอกขนาดของหน่วยความจำที่ตัวแปรชนิดต่าง ๆ ต่อไปนี้ต้องใช้ในการจองพื้นที่ในหน่วยความจำ

   char, int , float, double, long int, long double

3. จงเขียนโปรแกรมเพื่อหาคำตอบว่าประโยคต่อไปนี้เป็นจริงหรือเท็จ ถ้ากำหนดให้ตัวแปร

   A = 20 B=30 C = 2 D = 5 E=50

   3.1 (A>=B) && (A==A)

   3.2 (B+C>A+D)|| (B+C<A+D)&&(D<10)

   3.3 (A==20) && (B>=30)

   3.4 (50==E) || (!(D<=E)

   3.5 !(D!=5)

   3.6 (B+C+20)!=50

   3.7 C==(D-3)

   3.8 (A/C<=B) && (C+D<=A) || (D>A)

   3.9 (A%C+5)==(E/5-10)

   3.10 A<B&&D<E