

# เรียนรู้ภาษา C++ .....

## ในรูปแบบภาษาเชิงโครงสร้าง

วัชระ รอดสัมฤทธิ์  
ฟิสิกส์ราชมงคล  
กุมภาพันธ์ 2553

## บทที่ 0

### Warm up

ก่อนที่จะเริ่มศึกษาภาษา C++ ต้องเตรียมเครื่องมือให้พร้อมเสียก่อน จากนั้นทดลองใช้เครื่องมือต่าง ๆ โดยเขียนโปรแกรมสั้น ๆ ขึ้นมาสักโปรแกรมหนึ่ง แล้วดูผลการทำงานของโปรแกรมนั้น เป็นเสมือนการอุ่นเครื่องก่อนที่จะลงสนามจริง ๆ

เครื่องมือที่ใช้ประกอบในการเรียน ภาษา C++ (ไม่นับรวมคอมไพเลอร์ที่ต้องมีอยู่แล้ว) ได้แก่ ตัวแปลภาษา C++ ให้เป็นภาษาเครื่องที่คอมพิวเตอร์สามารถเข้าใจ เรียกสั้น ๆ ว่า คอมไพเลอร์ (Compiler) ในที่นี้จะเลือกใช้คอมไพเลอร์ของบริษัทบอร์แลนด์ ซึ่งเป็นของฟรี ดาวน์โหลดมาใช้โดยไม่ต้องเสียสตางค์และไม่ต้องเกรงเรื่องลิขสิทธิ์ เครื่องมืออีกชิ้นหนึ่งคือโปรแกรมประเภท editor ที่ถนัด มีไว้สำหรับพิมพ์ต้นฉบับโปรแกรมหรือที่เรียกว่า Source code

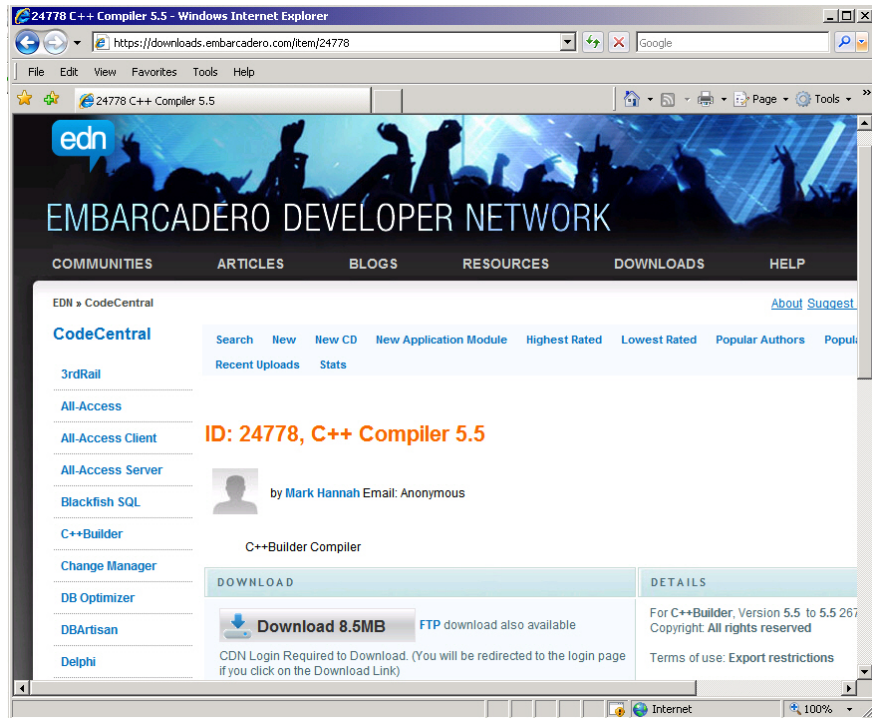
แบ่งการเตรียมการเป็น 5 ขั้นตอนดังนี้

1. download คอมไพเลอร์ Borland C ++ เวอร์ชัน 5.5
2. ติดตั้งคอมไพเลอร์
3. สร้าง configuration file สำหรับ คอมไพเลอร์
4. ติดตั้ง Editor
5. ทดสอบโดยการเขียนโปรแกรมสั้น ๆ ชื่อ Hello.cpp

เริ่มทำตามขั้นตอนดังต่อไปนี้

1. download คอมไพเลอร์ Borland C ++ เวอร์ชัน 5.5

เชื่อมต่ออินเทอร์เน็ต แล้วไปที่ <http://www.codegear.com/downloads/free/cppbuilder> ให้ทำตามคำแนะนำที่บอกไว้ในเว็บไซต์ อาจต้องลงทะเบียนหรือใส่ email ของเรา

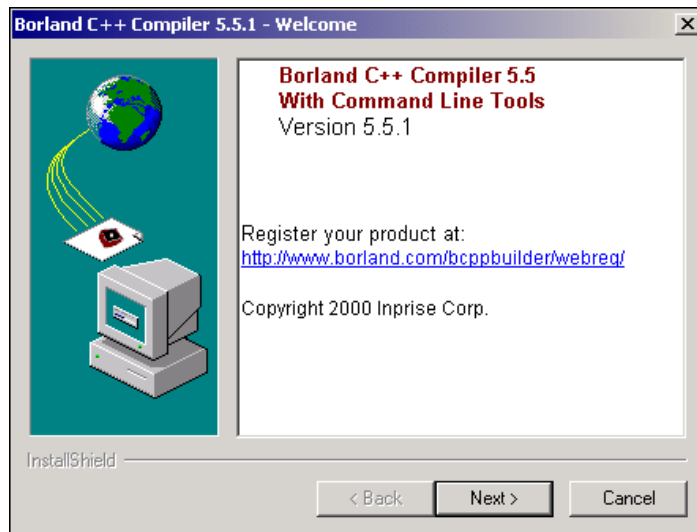


รูปที่ 1 หน้าตาของเว็บไซต์ที่จะดาวน์โหลด Borland C++ compiler 5.5 เข้าถึงเมื่อ 15 ม.ค. 2553  
 จากนั้นทำการดาวน์โหลดไฟล์ที่ชื่อว่า FreeCommandLineTools.exe ขนาดของไฟล์ประมาณ 8.5 MB  
 มาเก็บไว้ในเครื่องของเรา



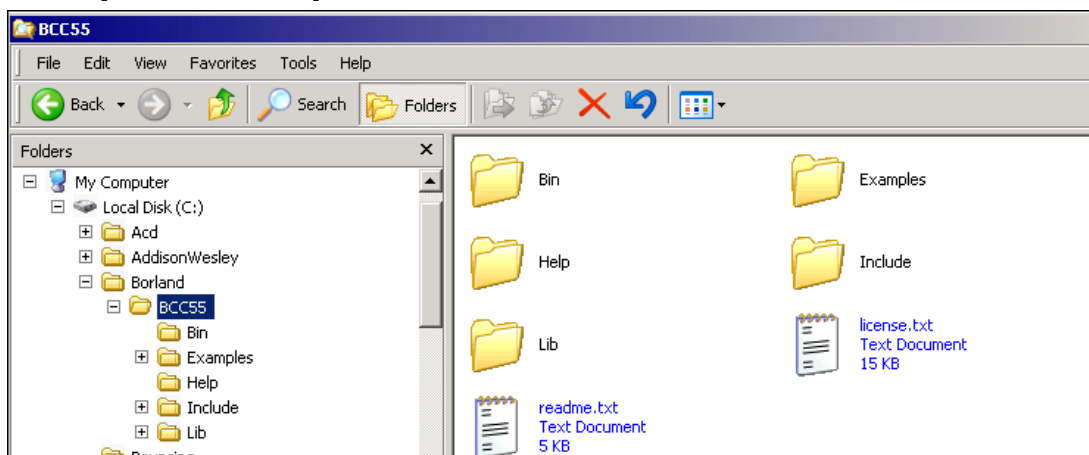
รูปที่ 2 ไฟล์ freecommandLinetools.exe ที่ดาวน์โหลดมาได้

2. ทำการติดตั้ง Boland C++ Compiler (ต่อไปจะขอเรียกสั้น ๆ ว่า BCC) โดยปล่อยให้โปรแกรมใช้ค่าที่ได้กำหนดไว้แล้ว



รูปที่ 3 เริ่มติดตั้ง Borland C++ Compiler

ตัวโปรแกรมคอมไพเลอร์ที่ถูกติดตั้งจะอยู่ที่ ไดรฟ์ C ในโฟลเดอร์ C:\Borland\BCC55 และมีโฟลเดอร์ย่อยแยกกันอยู่อีกหลายโฟลเดอร์ดังรูปที่ 4

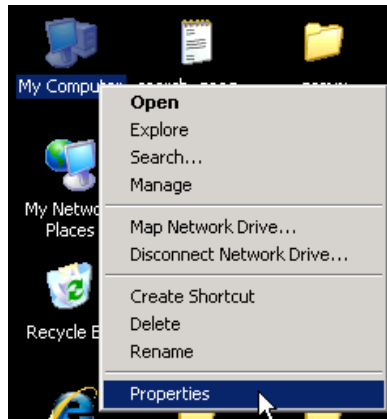


รูปที่ 4 เมื่อติดตั้งโปรแกรม Borland C++ compiler โดยใช้ค่า default จะได้โฟลเดอร์ย่อยดังรูป

3. กำหนดค่าการใช้งานต่าง ๆ ให้กับระบบ (configuration)
  - 3.1 กำหนด path เพื่อที่ระบบปฏิบัติการจะได้มองเห็นคอมไพเลอร์ที่เราใช้งาน
 

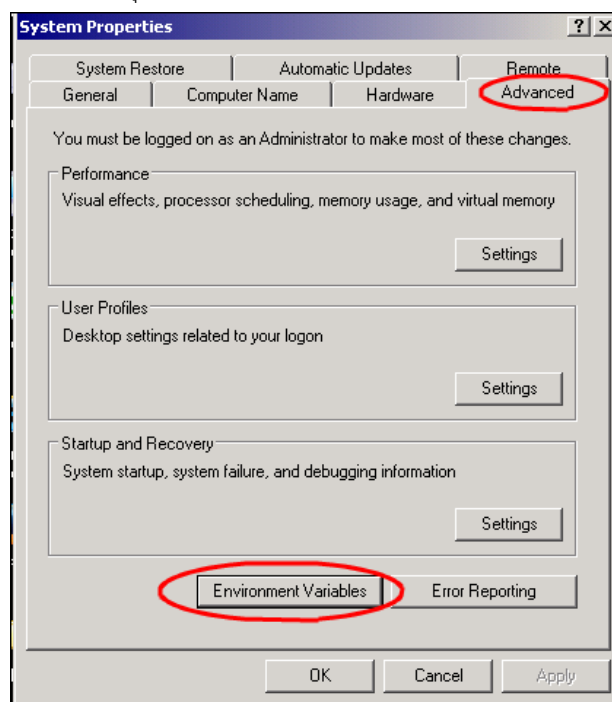
ถ้าคอมพิวเตอร์นั้นใช้ Windows XP หรือ windows 2000 ให้ทำดังนี้

    - คลิกเมาส์ ที่ปุ่มขวา ที่ไอคอน "My computer" ที่อยู่บน desktop จะเห็นเมนู pop up ขึ้นมา ให้เลือก "Properties" ซึ่งอยู่ด้านล่างสุด



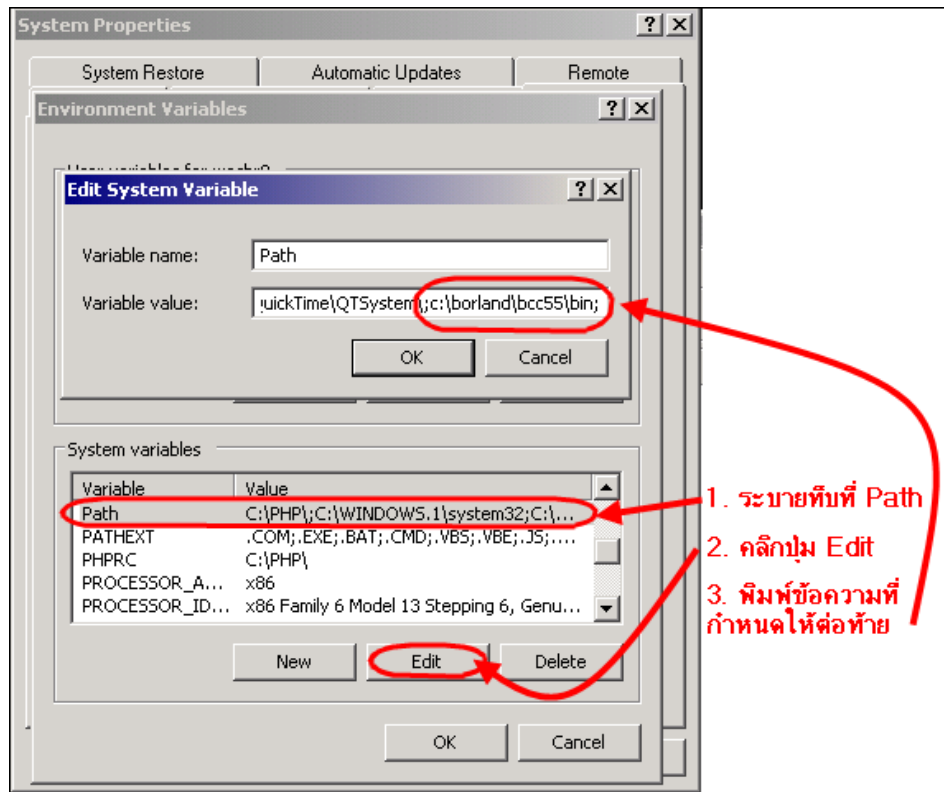
รูปที่ 5 เมื่อคลิกเมาส์ปุ่มขวา ที่ My Computer

- เมื่อคลิกที่ “Properties” จะมีหน้าต่าง ชื่อ System Properties ปรากฏขึ้นมา ให้เลือก tab ที่ชื่อว่า “Advanced” ในแท็บนี้ คลิกเมาส์ปุ่ม “Environmental Variables” ที่วงไว้ด้วยสีแดง ในรูปที่ 6



รูปที่ 6 การกำหนดค่า Path ใน Environment Variables

- ให้ระบายหรือทำ Highlight ที่ “Path” อยู่ตรงบริเวณด้านล่างของหน้าต่าง คลิกปุ่ม Edit พิมพ์ข้อความ “;C:\BORLAND\BCC55\BIN;” ต่อท้ายประโยคเดิม ให้สังเกตว่ามีเครื่องหมาย semi colon ปิดหัวท้าย และไม่ต้องพิมพ์เครื่องหมาย “ “



รูปที่ 7 การเพิ่มค่า Path เพื่อให้วินโดวส์รู้จัก คอมไพเลอร์

- คลิกปุ่ม OK ในหน้าต่าง “Edit System Variable” และ คลิกปุ่ม OK ในหน้าต่าง “Environment Variables” และปุ่ม OK ในหน้าต่าง “System Properties”

ถ้าคอมพิวเตอร์นั้นใช้ Windows 98/Me ให้ทำดังนี้

- กดปุ่ม Start คลิกที่เมนู Run พิมพ์คำสั่ง “cmd” ลงไปในกล่องข้อความ แล้วกด Enter
- ไปที่ root ของไดรฟ์ C โดยพิมพ์คำว่า “cd \” จากนั้น พิมพ์คำว่า “edit autoexec.bat”
- เพิ่มข้อความบรรทัดใหม่ดังนี้ “PATH=C:\BORLAND\BCC55\BIN;%PATH%”
- Save ไฟล์ autoexec.bat ที่เราแก้ไข โดยกดปุ่ม Alt + F และ กดปุ่ม S
- ออกจากโปรแกรม edit โดยกดปุ่ม Alt + F และ X

3.2 สร้าง Configuration files โดยจะสร้างไฟล์นี้เก็บไว้ในโฟลเดอร์ C:\BORLAND\BCC55\BIN

จะสร้างขึ้นมา 2 ไฟล์ ไฟล์หนึ่งสำหรับ คอมไพเลอร์ อีกไฟล์หนึ่งสำหรับการ link กับ library ในระหว่างการทำ execute file

- คลิกที่ปุ่ม Start แล้วคลิกที่ เมนู Run พิมพ์คำว่า “cmd” ลงในกล่องข้อความพร้อม กด Enter
- ไปที่โฟลเดอร์ C:\BORLAND\BCC55\BIN โดยใช้คำสั่งต่อไปนี้

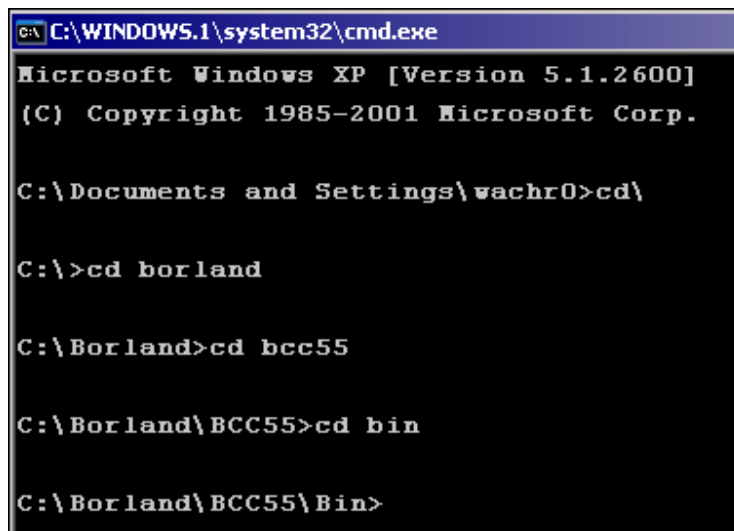
cd\ กด Enter

cd Borland กด Enter

cd bcc55 กด Enter

cd bin กด Enter

จะได้หน้าต่างดังรูปที่ 8



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\wachr0>cd\

C:\>cd borland

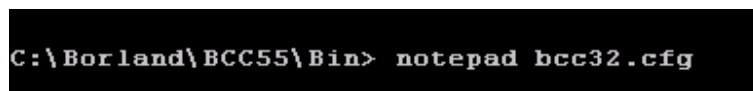
C:\Borland>cd bcc55

C:\Borland\BCC55>cd bin

C:\Borland\BCC55\Bin>
```

รูปที่ 8 การเข้าไปที่โฟลเดอร์ C:\Borland\Bcc55\bin

- สร้าง configuration file หรือ ไฟล์ cfg อันแรก ชื่อว่า bcc32.cfg โดยพิมพ์ คำสั่ง Notepad bcc32.cfg



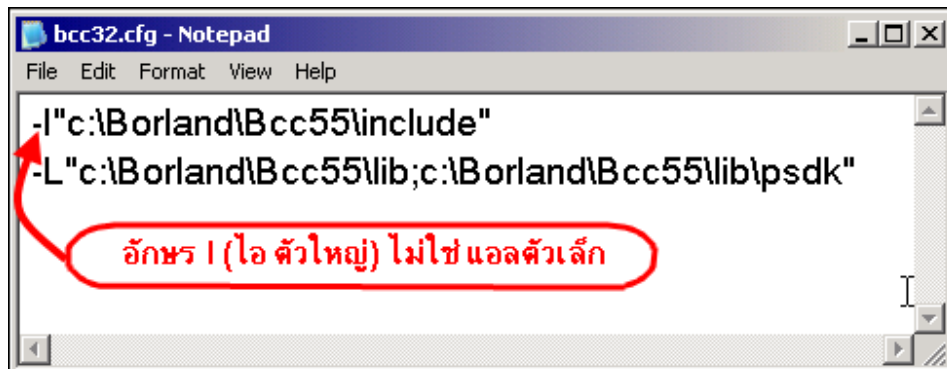
```
C:\Borland\BCC55\Bin> notepad bcc32.cfg
```

รูปที่ 9 การเรียกใช้ Notepad พิมพ์ configuration file

- โปรแกรม Notepad จะสร้างไฟล์ว่าง ๆ ที่ยังไม่มีข้อความใด ๆ ขึ้นมา 1 ไฟล์ จากนั้นให้พิมพ์ข้อความต่อไปนี้ลงไปในพื้นที่ว่าง

- I"c:\Borland\Bcc55\include"
- L"c:\Borland\Bcc55\lib;c:\Borland\Bcc55\lib\psdk"

บรรทัดแรก จะเป็นประโยคที่บอกว่า include file ทั้งหมดนั้นเก็บไว้ที่ใด บรรทัดที่ 2 จะบอกถึง Library file ที่เก็บไว้



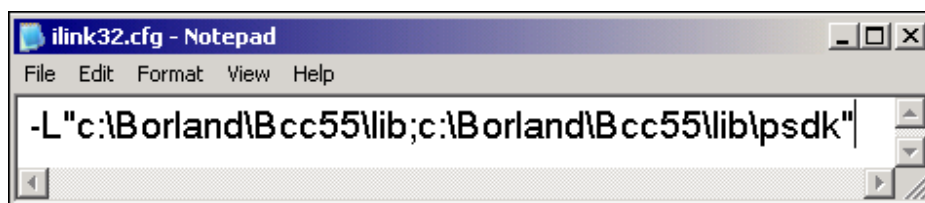
รูปที่ 10 ข้อความใน bcc32.cfg

- Save ไฟล์ bcc32.cfg จากนั้นออกจากโปรแกรม Notepad
- สร้าง configuration file หรือไฟล์ cfg ไฟล์ที่ 2 ชื่อ ilink32.cfg โดยใช้ Notepad ให้พิมพ์คำสั่งดังนี้

```
C:\Borland\BCC55\Bin>notepad ilink32.cfg
```

รูปที่ 11 การเรียกใช้ Notepad สร้างไฟล์ ilink32.cfg

- พิมพ์ข้อความต่อไปนี้ลงในพื้นที่ว่างของ Notepad
- L\"c:\Borland\Bcc55\lib;c:\Borland\Bcc55\lib\psdk\"



รูปที่ 12 ข้อความในไฟล์ ilink32.cfg

- Save ไฟล์ ilink32.cfg จากนั้นออกจากโปรแกรม Notepad
4. ติดตั้ง Editor ที่ตนเองถนัด ในที่นี้ผู้เขียน เลือกใช้ Editorplus เวอร์ชัน 2 (ปัจจุบัน มีถึง เวอร์ชัน 3)



แต่ editor ตัวนี้เป็น shareware สามารถดาวน์โหลดทดลองใช้ได้ 30 วัน เราสามารถใช้ google ค้นหา editor ที่เป็นของฟรีโดยใช้คำว่า “free editor” ก็จะได้โปรแกรม editor ที่เป็น freeware หรือ Open source เป็นจำนวนมากต่อไปนี้เป็น Editor ที่เป็นของฟรี

NotePad++ ดาวน์โหลดได้ที่ <http://notepad-plus.sourceforge.net/uk/site.htm>

Crimson editor ดาวน์โหลดได้ที่ [www.crimsoneditor.com](http://www.crimsoneditor.com)

VIM ดาวน์โหลดได้ที่ [www.vim.org](http://www.vim.org)

PSPad ดาวน์โหลดได้ที่ [www.psPad.com/en/](http://www.psPad.com/en/)

EditPad classic ดาวน์โหลดได้ที่ [www.jgsoft.com](http://www.jgsoft.com)

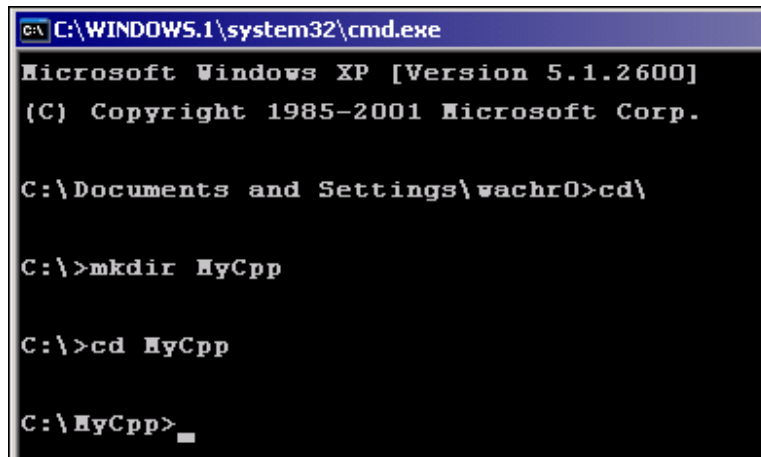
ConTEXT ดาวน์โหลดได้ที่ [www.context.com](http://www.context.com)

ถ้าหาไม่ได้จริง ๆ จะใช้ Notepad ที่ติดมากับวินโดวส์เป็น editor ก็ได้ แต่จะทำให้การเขียนโปรแกรมอาจไม่ค่อยสะดวกมากนัก จึงไม่อยากจะแนะนำให้ใช้

5. ทดสอบคอมไพเลอร์ ที่เราเพิ่งติดตั้ง ทำตามขั้นตอนต่อไปนี้

- สร้างโฟลเดอร์สำหรับเก็บ source code ที่เราจะเขียนไว้โดยเฉพาะสักโฟลเดอร์หนึ่งในที่นี้ จะใช้ชื่อว่า MyCpp โดยสร้างไว้ที่ C:\MyCpp ให้เข้าไปที่ console box ดังที่เคยทำ(คลิกที่ Start และคลิกที่เมนู Run พิมพ์คำว่า “cmd” แล้วกดปุ่ม Enter)

- พิมพ์คำสั่ง “cd \” [Enter]
- พิมพ์ คำสั่ง “mkdir MyCpp” [Enter]
- พิมพ์คำสั่ง “cd MyCpp” [Enter]



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\wachr0>cd\

C:\>mkdir MyCpp

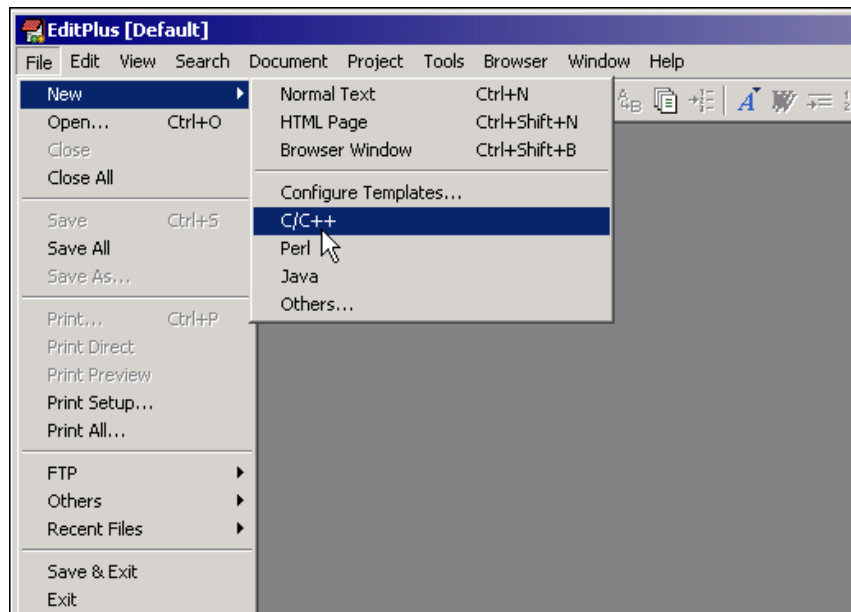
C:\>cd MyCpp

C:\MyCpp>
```

รูปที่ 13 การสร้างโฟลเดอร์ MyCpp สำหรับเก็บ source code

ขณะนี้เรากำลังอยู่ในโฟลเดอร์ C:\MyCpp

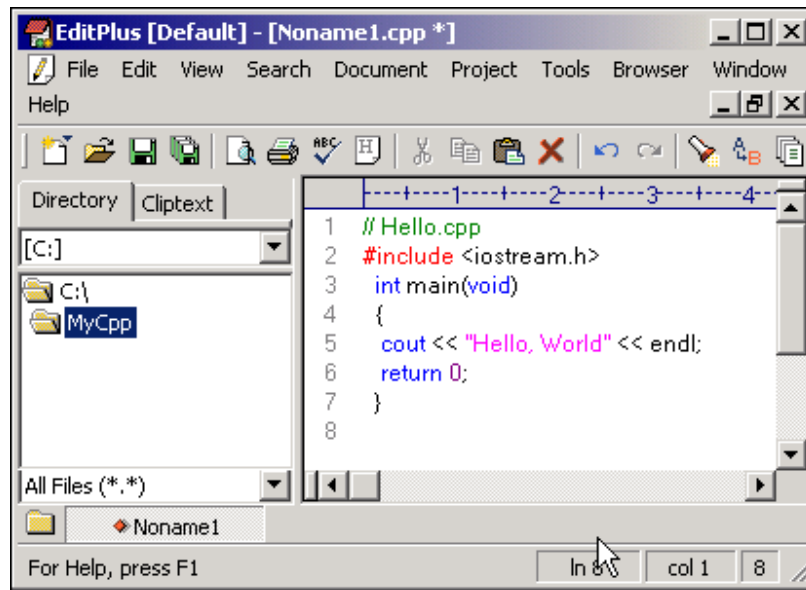
- เรียกใช้โปรแกรม editor ในที่นี้คือ Editplus คลิกที่เมนู File >> New >> C/C++



รูปที่ 14 การสร้างไฟล์ใหม่ใน Editorplus เพื่อเขียนโปรแกรม

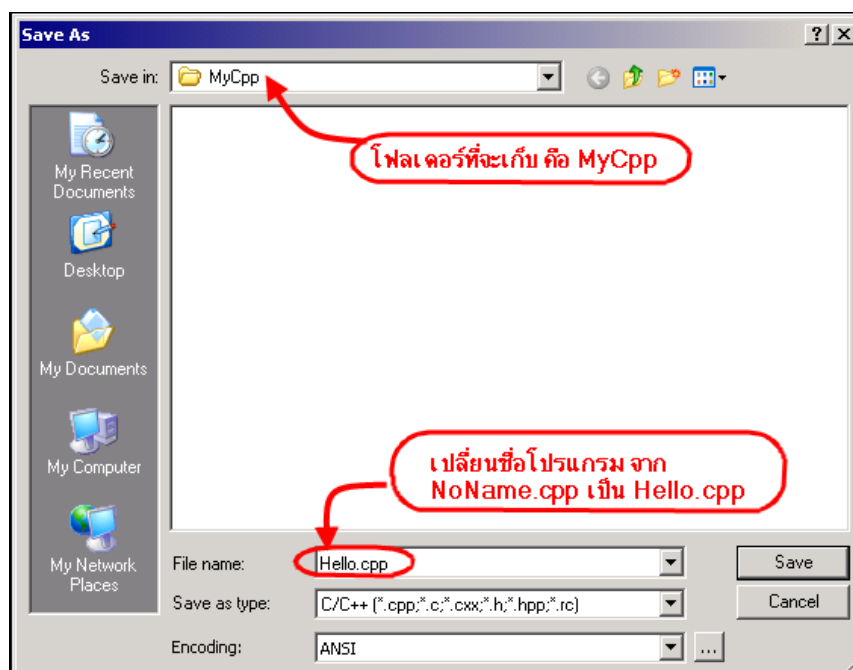
- พิมพ์ข้อความต่อไปนี้ลงในพื้นที่ว่าง

```
// Hello.cpp
#include <iostream.h>
int main(void)
{
    cout << "Hello, World" << endl;
    return 0;
}
```



รูปที่ 15 โปรแกรมแรก Hello.cpp

- Save โปรแกรมภาษา C++ โปรแกรมแรกของเรา เก็บไว้ในโฟลเดอร์ C:\MyCpp โดยตั้งชื่อว่า Hello.cpp



รูปที่ 16 แสดงโฟลเดอร์ที่จะเก็บไฟล์ Hello.cpp

- กลับไปที่หน้าต่าง Console box พิมพ์คำสั่ง เพื่อทำการคอมไพล์โปรแกรกดั้งนี้ "Bcc32 hello.cpp" แล้ว  
เคาะ Enter โปรแกรมจะทำการคอมไพล์และลิงค์ Hello.cpp ถ้าไม่มีข้อผิดพลาดใดๆ ในการพิมพ์โปรแกรม จะได้  
ข้อความที่หน้าจอ ดังรูปที่ 17

```
C:\MyCpp>bcc32 Hello.cpp
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
Hello.cpp:
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
C:\MyCpp>
```

รูปที่ 17 การคอมไพล์ Hello.cpp

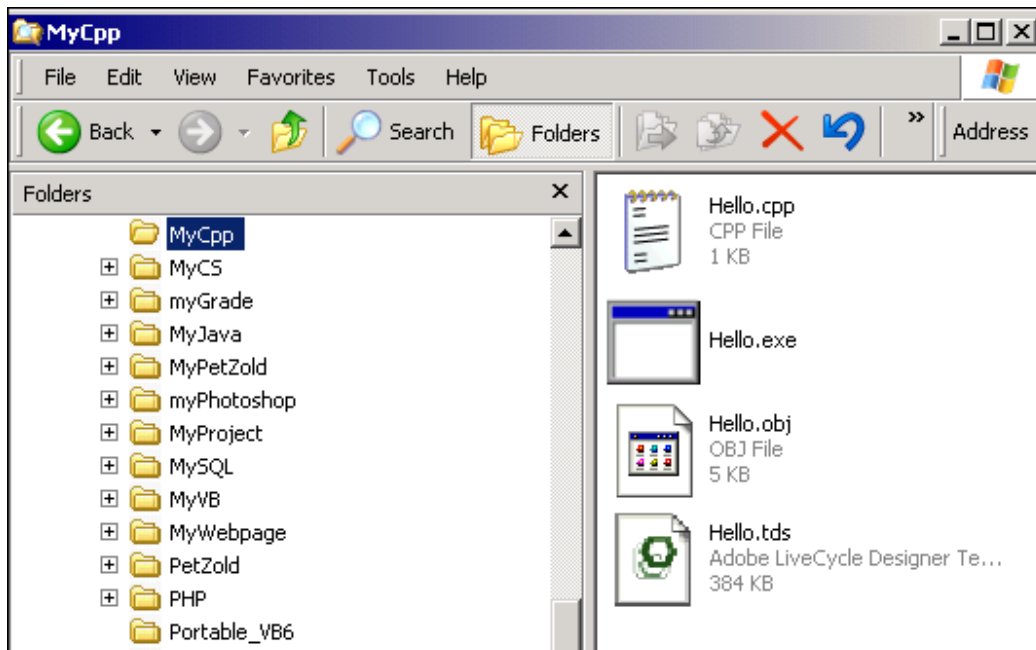
- ทดลองให้โปรแกรมทำงาน โดยพิมพ์คำสั่ง hello จะเห็นข้อความ "Hello, World" แสดงที่หน้าจอ

```
C:\MyCpp>hello
Hello, World
C:\MyCpp>
```

← ผลจากการ run โปรแกรม Hello.exe

รูปที่ 18 แสดงการรัน Hello.exe

- เมื่อดูที่ Folder C:\MyCpp พบว่านอกจากไฟล์ Hello.cpp แล้ว ยังมีไฟล์อื่นเกิดขึ้นอีก 3 ไฟล์



รูปที่ 19 ไฟล์ที่เกิดขึ้น ในระหว่างการคอมไพล์และลิงค์ Hello.cpp

ไฟล์ Hello.obj เป็น Object file ที่ได้จากการที่คอมไพเลอร์แปลโปรแกรมต้นฉบับ Hello.cpp ให้เป็นภาษาเครื่อง จากนั้นโปรแกรม Link จะนำไฟล์ Hello.obj ไปเชื่อมโยงกับ ไฟล์ Library ที่เก็บไว้ใน โฟลเดอร์ LIB แล้วสร้างเป็น ไฟล์ที่คอมพิวเตอร์สามารถนำไปใช้งานได้ เรียกว่า executed file ในที่นี้ คือ Hello.exe เราสามารถนำ Hello.exe ไปรันที่คอมพิวเตอร์เครื่องใดก็ได้ที่ใช้ระบบปฏิบัติการวินโดวส์เหมือนกัน ส่วน ไฟล์ Hello.tds ย่อมาจาก Turbo Debugger Symbols File เป็นไฟล์ที่สร้างขึ้นระหว่างการคอมไพล์และ Link มีประโยชน์สำหรับการ Debug (การแก้ไขโปรแกรมเมื่อมีข้อผิดพลาด) โดยใช้โปรแกรม Turbo debugger เมื่อโปรแกรม Hello.exe ทำงานได้ตามจุดประสงค์ที่เราต้องการแล้ว สามารถลบไฟล์ Hello.obj และ Hello.tds ทั้งได้

มาถึงขั้นนี้ เครื่องมือต่าง ๆ ที่เราเตรียมไว้ก็พร้อมแล้วที่จะตลุย นำเราไปสู่การเรียนรู้ภาษา C++

## บทที่ 1

C++ จัดเป็นภาษาคอมพิวเตอร์เชิงวัตถุ (Object oriented language) แต่ยังคงสภาพเป็นภาษาเชิงโครงสร้าง (Structure or Procedure oriented language) ไว้ด้วยในขณะเดียวกัน ภาษาคอมพิวเตอร์เชิงวัตถุ บางภาษาเช่น Java, Smalltalk ไม่มีคุณสมบัติเช่นนี้ ปัญหาที่มีลักษณะไม่ซับซ้อนมากนัก การออกแบบ โปรแกรมให้มีลักษณะเชิงโครงสร้าง สามารถทำได้ง่ายกว่าและรวดเร็วกว่า แต่เมื่อถึงคราวที่จะต้องทำซอฟต์แวร์ ขนาดใหญ่ ประกอบด้วยคำสั่งไม่น้อยกว่า สองแสนบรรทัดหรือมากกว่า หรือมีโปรแกรมน้อยขนาดเล็กลง ๆ เป็นจำนวนมาก ต้องใช้โปรแกรมเมอร์หลายคน แบ่งหน้ากันทำงาน การเขียนโปรแกรมเชิงวัตถุจะเหมาะสมกว่า เพราะง่ายและสะดวกในการจัดการและบริหารตัวซอฟต์แวร์ที่มีขนาดใหญ่นั้น

โปรแกรมต้นฉบับ (Source code) ที่เขียนด้วยภาษา C++ จะต้องถูกแปลงให้เป็นภาษาเครื่อง กลายเป็น executed file ก่อนนำไปใช้งาน และเป็นไปตามวัฏจักรการพัฒนาโปรแกรม (Developed cycle) ซึ่ง ประกอบด้วยพิมพ์ (เขียน) โปรแกรมที่ออกแบบไว้แล้ว (Editing) แปลโปรแกรมให้เป็นภาษาเครื่อง (Compiling) เชื่อมโยงโปรแกรมที่อยู่ในรูปภาษาเครื่องกับโปรแกรมย่อยอื่น ๆ ที่เก็บไว้ในคลังโปรแกรม (Library) เรียกว่า Linking แล้วนำโปรแกรมไปทดสอบการทำงาน (Testing or Running)

ถ้าการทำงานของโปรแกรมยังไม่บรรลุวัตถุประสงค์ที่ตั้งไว้ ก็ต้องหาข้อบกพร่องหรือข้อผิดพลาด (Debug) จากนั้นกลับไปเริ่มต้นวัฏจักรใหม่อีกครั้ง จนกว่าจะได้โปรแกรมที่สามารถนำไปใช้งานได้

ทำไมต้องเป็น C++

ความเป็นมาของภาษา C++

โปรแกรมแรก

ตัวแปรและค่าคงที่ (Variable and Constant)

ชนิดของข้อมูล (Data type)

นิพจน์ทางคณิตศาสตร์ (Mathematical Expression)

การรับข้อมูลทางแป้นพิมพ์

## ทำไมต้องเป็น C++

C++ มีลักษณะเฉพาะตัวที่มีความโดดเด่นและเห็นได้ชัดกว่าภาษาคอมพิวเตอร์อื่น ๆ คือ

Object-oriented programming การโปรแกรมเชิงวัตถุทำให้โปรแกรมเมอร์สามารถออกแบบแอปพลิเคชันในมุมมองแบบวัตถุ การ Coding เป็นการกำหนดให้วัตถุที่สร้างสื่อสารซึ่งกันและกัน สามารถนำโค้ดที่สร้างขึ้นกลับมาใช้ใหม่ ทำให้เกิดความผิดพลาดน้อยและสร้างผลงานได้อย่างรวดเร็ว

Portability ในทางปฏิบัติสามารถนำโปรแกรมต้นฉบับของภาษา C++ ไปคอมไพล์ในเครื่องคอมพิวเตอร์ในแพลตฟอร์มใด ๆ หรือระบบปฏิบัติการใด ๆ โดยไม่ต้องแก้ไขดัดแปลงใหม่

Brevity คำสั่งในภาษา C++ มีลักษณะสั้น กระชับรัดกุมกว่าภาษาอื่น ๆ (ทำให้คีย์บอร์ด อายุยืนกว่า : < )

Modular Programming สามารถแบ่งโปรแกรมออกเป็นส่วน ๆ คอมไพล์และนำมาลิงค์ด้วยกัน ช่วยประหยัดเวลาการคอมไพล์ ส่วนใดผิดก็สามารถคอมไพล์แล้วแก้ไขเฉพาะส่วนนั้น นอกจากนี้ยังสามารถนำไปเชื่อมต่อกับภาษาอื่น ๆ เช่น แอสเซมบลี ปาสคาล เป็นต้น

C Compatibility โปรแกรมที่เขียนด้วยภาษา C สามารถนำไปใช้ใน C++ โดยไม่ต้องดัดแปลงคำสั่งใด ๆ

Speed โปรแกรมไบนารีที่คอมไพล์โดย C++ จะมีขนาดเล็ก มีประสิทธิภาพ เพราะ C++ เป็นทั้งภาษาชนิด High level และ low level

## ความเป็นมาของภาษา C++

ประมาณ ปีค.ศ. 1969 ห้องปฏิบัติการ Bell ได้พัฒนาระบบปฏิบัติการขึ้นมาใหม่ ต่อมาชื่อเป็น UNIX โดย Ken Thomson ได้ใช้ภาษาแอสเซมบลีเขียน UNIX บนเครื่อง PDP-7 ต่อมา Martin Richard พัฒนาภาษาระดับสูงชื่อ BCPL ( Basic Combined Programming Language) เพื่อใช้เขียนระบบปฏิบัติการและคอมพิวเตอร์ ใช้กับเครื่อง Multics ทอมสันได้ยืมส่วน BCPL ให้มีขนาดเล็กลง เพื่อให้ทำงานได้บนเครื่อง DEC-PDP-7 ซึ่งมีหน่วยความจำเพียง 8 kB เรียกภาษานี้ว่า ภาษา B ทั้ง BCPL และ B เป็นภาษาที่ไม่มีชนิดข้อมูล

ค.ศ. 1971 ได้มีการพัฒนาระบบ UNIX บนเครื่อง PDP-11 ด้วยภาษาแอสเซมบลี และ B

ค.ศ. 1972 Denis Ritchie ได้พัฒนาภาษา B ให้มีความสามารถมากขึ้น โดยนำลักษณะบางส่วนจากภาษา Algo 68 มาใส่รวมด้วย เรียกว่า ภาษา C ในปีถัดมา ได้ใช้ภาษา C เขียนระบบปฏิบัติการ UNIX เกือบ 90 %

ค.ศ. 1978 Brian Kernighan & Denis Ritchie เขียนหนังสือ The C Programming Language จัดเป็นคัมภีร์ภาษา C

ค.ศ. 1980 Bjarne Stroustrup ขยายขีดความสามารถของ C ให้เขียนเป็นภาษาเชิงวัตถุได้ เรียกว่า C with class ต่อมาในปี 1983 Rich Mascitti ได้ให้ชื่อใหม่เป็น C++

ค.ศ. 1983 สำนักงานมาตรฐานแห่งชาติอเมริกา ( American National Standard Institute, ANSI) ได้ตั้งคณะกรรมการ ชุด X3J11 กำหนดมาตรฐานภาษา C และในปีค.ศ. 1989 ได้ตั้งคณะกรรมการชุด X3J16 กำหนดมาตรฐาน C++

ค.ศ. 1989 กำหนดมาตรฐาน ANSI C ประกอบด้วย K & R C และรวมบางส่วนของภาษา C++ และรวม Standard Library ประกาศเป็น ISO/ IEC 9699:1990 เมื่อ ปีค.ศ. 1990 (ISO – International Standardization Organization, IEC – International Electro technical Commission)

ค.ศ. 1991 คณะกรรมการ ANSI X3J16 รวมตัวกลายเป็นส่วนหนึ่งของ ISO/.IEC JTC1/SC22/WG21

ค.ศ. 1998 ประกาศมาตรฐาน ภาษา C++ (ISO/IEC 14882: 1998)

ค.ศ. 1999 ประกาศมาตรฐาน ภาษา C (ISO/IEC 9899: 1999) เรียกย่อว่า ISO C99

( ข้อมูลจาก [www.hitmill.com/programming/cpp](http://www.hitmill.com/programming/cpp))

## เริ่มต้นโปรแกรมแรก กับภาษา C++

ในที่นี้จะเริ่มต้นโดยนำ C++ ไปใช้ในงานคำนวณ เป็นการหาพื้นที่ของรูปสี่เหลี่ยมผืนผ้า ซึ่งหาได้จากสูตร กว้าง X ยาว source code ของโปรแกรมมีดังนี้

```

1:  /* =====
2:  Program 1.1: FirstPrg.cpp
3:      Calculating an area of rectangle.
4:  Author: Wachara R.
5:  Date: 03 Oct 2003
6:  Last Update: -
7:  Note: The first program in C++
8:  ----- */
9:  #include <iostream.h>
10: int main() {
11:     // define variables
12:     float width, length, area;
13:     // initialize variables
14:     width = 30.0; // width of rectangle
15:     length= 120.0; // length of rectangle
16:     //Compute area of rectangle.
17:     area = width*length;
18:     //Print area on standard output
19:     cout << " Area of this rectangle = "
20:         << area << " sqr.meter" << endl;
21:     // exit program
22:     return 0;
23: }
24: // =====

```

โปรแกรมสั้น ๆ 24 บรรทัดสามารถแยกออกเป็นส่วนประกอบย่อย ๆ โดยเรียงตามลำดับที่ปรากฏใน  
โปรแกรกดังนี้

- หมายเหตุของโปรแกรม (Comment)
- ส่วนกำกับในการแปลโปรแกรม (Preprocessor Directive)
- ฟังก์ชันที่มีชื่อว่า main
- ตัวแปร (Variable)
- Statement

หมายเหตุของโปรแกรม เป็นส่วนที่ผู้เขียนโปรแกรมบันทึกข้อความเช่น บอกชื่อโปรแกรม จุดประสงค์ของ  
โปรแกรม เงื่อนไขการทำงานของโปรแกรม เพื่อสะดวกในการตรวจแก้ หรือเตือนความจำเมื่อกลับมาอ่าน  
ภายหลัง คอมไพเลอร์จะไม่ทำการคอมไพล์ในส่วนที่เป็น comment นี้ ในภาษา C++ จะมีหมายเหตุอยู่สอง  
ลักษณะคือ comment ที่เกิดจากรวมกันของบรรทัดหลาย ๆ บรรทัด หรือ comment เฉพาะบรรทัดนั้นเพียง  
บรรทัดเดียว

บรรทัดที่ 1-8 จะใช้สัญลักษณ์ /\* (ในบรรทัดที่ 1) และ \*/ ( บรรทัด ที่ 8) ข้อความที่อยู่ระหว่างเครื่องหมาย  
/\*.....\*/ คือส่วนที่เป็น comment

บรรทัดที่ 14 และ 15 ข้อความที่อยู่ท้ายเครื่องหมาย // จนจบบรรทัดจะเป็น comment

ในการแก้ไขโปรแกรม บางครั้งจะกำหนดมิให้คำสั่งบางบรรทัดทำงาน โดยการใส่ // นำหน้าบรรทัดนั้น หรือถ้า  
ต้องการให้ประโยคคำสั่งหลาย ๆ บรรทัด ไม่ให้ทำงานอาจครอบด้วยเครื่องหมาย /\*.....\*/ แต่ต้องระวัง  
คอมไพเลอร์บางบริษัท ๆ อาจไม่ยอมรับการใช้คอมเมนต์ซ้อนคอมเมนต์



ส่วนกำกับในการแปลโปรแกรม (Preprocessor directive) คือคำสั่งที่ใช้บอกให้คอมไพเลอร์จะต้องทำก่อนที่จะทำการคอมไพล์โปรแกรม คำว่า preprocessing หมายถึงขั้นตอนแรกสุดก่อนที่คอมไพเลอร์จะแปลภาษา C++ ให้อยู่ในรูปภาษาเครื่อง บรรทัดที่ 9 `#include <iostream.h>` จัดเป็น preprocessor directive คำสั่งหนึ่ง ซึ่งจะใส่ไว้ตรงส่วนหัวของโปรแกรม เป็นคำสั่งให้คอมไพเลอร์นำไฟล์ที่อยู่ในวงเล็บรูปเหลี่ยมในที่นี้คือ `iostream.h` มาแทรกตรงส่วนนี้ของโปรแกรมในระหว่างการคอมไพล์ ไฟล์นี้มีคำสั่งที่ช่วยให้เราสามารถใส่คำสั่ง `cout` ในบรรทัดที่ 19 ซึ่งใช้แสดงผลลัพธ์บนหน้าจอได้ ถ้าลบบรรทัดนี้ออก คอมไพเลอร์จะแจ้งข้อผิดพลาดบนหน้าจอว่าไม่รู้จักคำสั่ง `cout`

ประโยคที่เป็น preprocessor directive จะไม่มีเครื่องหมาย ; ตามหลัง

สัญลักษณ์ `<.....>` ที่ใช้คร่อมไฟล์จะเป็นการกำหนดว่า header file นั้นถูกเก็บไว้ในโฟลเดอร์ที่คอมไพเลอร์กำหนดไว้ โดยปกติจะเก็บไว้ภายใต้โฟลเดอร์ที่ชื่อ `include` ในเทอร์โบซี เวอร์ชัน 3 ไฟล์นี้จะถูกเก็บไว้ใน `c:\tc\include\` ถ้าเราสร้าง include file ของเราเองเก็บไว้ที่เดียวกับโปรแกรมต้นฉบับในโฟลเดอร์ MyC++ จะต้องแจ้งให้คอมไพเลอร์ทราบ โดยเขียนดังนี้

```
#include "myheader.h"
```

ระวังอย่าให้มีช่องว่างในเครื่องหมาย `<....>` และ `"...."`

Preprocessor directive จะขึ้นต้นคำสั่งด้วย `#` (pound sign) โดยทั่วไปจะถูกนำมาใช้ในลักษณะต่อไปนี้

1. ใช้ตัวกำกับ `#include` เพิ่มหรือแทรกส่วนต่าง ๆ ของโปรแกรม เราสามารถนำตัวแปรต่าง ๆ ไปประกาศไว้ในแฟ้ม `include` เพียงทีเดียว เพื่อลดความเสี่ยงที่ฟังก์ชันในแฟ้มต้นฉบับอื่น ๆ หาตัวแปรไม่พบ
2. ตัวกำกับที่มีเงื่อนไข (conditional directive) เราสามารถใช้ตัวกำกับที่มีเงื่อนไข เช่น `#if`, `#ifdef`, `#ifndef`, `#else`, `#elseif` และ `#endif`
3. ใช้ควบคุมการแปลโปรแกรมต้นฉบับให้อยู่ภายใต้เงื่อนไขต่าง ๆ เช่น แปลเพียงส่วนใดส่วนหนึ่งของโปรแกรม หรือนำไปคอมไพล์บนระบบปฏิบัติการที่ต่างกัน

ข้อแนะนำ ถ้าต้องการ debug โปรแกรมเป็นช่วง ๆ สามารถใช้ preprocessor แทน `/*.....*/` เพื่อป้องกันการเกิด comment ซ้อน comment เช่น

```
#if (0)
    while (I < 10) {
        sum = sum + i;
    /* summation of each preparation for average calculation */
    }
    average = sum/n;
#endif
```

ฟังก์ชัน `main()` บรรทัดที่ 10 เป็นฟังก์ชันที่มีอยู่เสมอในโปรแกรมภาษา C++ ทุก ๆ โปรแกรม เมื่อเรียกใช้โปรแกรมนี้ให้ทำงาน ระบบปฏิบัติการจะใช้ฟังก์ชัน `main()` เป็นจุดเริ่มต้นของโปรแกรม จากตัวอย่าง `Int main()` หมายถึงเมื่อจบการรันโปรแกรม ฟังก์ชัน `main()` จะส่งค่ากลับเป็นจำนวนเต็ม (integer) ไปยังระบบปฏิบัติการ ในที่นี้คือ 0 ดูได้จากบรรทัดที่ 16 `return 0;` `main` เป็นชื่อฟังก์ชัน () ในวงเล็บว่างเปล่า แสดงว่าไม่มีพารามิเตอร์ใด ๆ ที่ใช้ในฟังก์ชัน `main`

คำสั่งโปรแกรมจะอยู่ระหว่างเครื่องหมายปีกกาเสมอ ดังรูปแบบต่อไปนี้

```

Introductory comment
Preprocessor directive
int main( ) {
    definition - define variables used within main
    executable statement - specify the computing operation to
    be carried out
}

```

ถึงแม้จะไม่มีคำว่า int นำหน้า main (และฟังก์ชันอื่น ๆ ก็เช่นกัน) จะถือว่าฟังก์ชันนั้นส่งค่ากลับเป็นแบบจำนวนเต็ม

รูปแบบเต็มของฟังก์ชัน main มีดังนี้

```

int main (int argc, char **argv, char ** env) {
}

```

ในระบบปฏิบัติการวินโดวส์ จุดเริ่มต้นของโปรแกรมที่ทำงานภายใต้สภาพแวดล้อมของวินโดวส์จะเป็น

```

int WinMain ( ) { }

```

ฟังก์ชัน main ( ) จะส่งค่ากลับไปยังระบบปฏิบัติการ เพื่อให้ตรวจการประมวลผลของโปรแกรมว่ามีข้อผิดพลาดบ้างหรือไม่ ถ้าโปรแกรมส่งค่า “ศูนย์” กลับมา แสดงว่าการประมวลผลเป็นไปอย่างเรียบร้อยไม่มีปัญหา ถ้าค่าที่ส่งกลับมาไม่ใช่ศูนย์ แสดงว่ามีปัญหาเกิดขึ้น โปรแกรมส่วนใหญ่จะส่งค่าระหว่าง 1 ถึง 5 ถ้ามีปัญหาค่อนข้างร้ายแรงระหว่างโปรแกรมทำงาน ระบบปฏิบัติการสามารถใช้ตัวเลขเหล่านี้เป็น error code ในระหว่างการประมวลผล batch file ซึ่งอาจใช้ให้ระบบปฏิบัติการยกเลิกการทำงานหรือทำงานส่วนอื่นขึ้นขึ้นอยู่กับความต้องการของผู้เขียน batch file

ตัวแปร (variable) คือพื้นที่หน่วยความจำที่ใช้เก็บข้อมูล โดยการระบุชื่อ (identified) สมบัติที่สำคัญของตัวแปรมีอยู่ 3 อย่างคือ ชนิด (type) ขอบเขต (scope) และ ลักษณะการจัดเก็บข้อมูล (storage class) ว่าเป็นแบบ auto หรือ register หรือ extern หรือ static ซึ่งจะกล่าวอย่างละเอียดในหัวข้อตัวแปร

จากบรรทัดที่ 12 **float width, length, area;**

float เป็นชนิดของตัวแปร มีตัวแปรชนิดนี้อยู่ 3 ตัวคือ width, length และ area

Statement หรือ ถ้อยแถลง หรือ กระพริบความ อาจเป็นประโยค หรือวลี หรือคำเพียงหนึ่งคำ ปิดท้ายด้วยเครื่องหมาย semi colon เสมอ

บรรทัดที่ 13 14 และ 17

```

14:      width = 30.0;
15:      length= 120.0;
16:      //Compute area of rectangle.
17:      area = width*length;

```

เป็นการกำหนดค่า (assign) ให้ตัวแปร width และ length

บรรทัดที่ 19 เป็น statement ที่ใช้พิมพ์ผลลัพธ์ของโปรแกรมออกที่จอภาพ

```

19:      cout << " Area of this rectangle = "

```

20: << area << "sqr.meter" << endl;

cout ย่อมาจากคำว่า Console Output อ่านว่า ซี – เอช Cout จะนำข้อความ **Area of this rectangle =** แสดงออกที่จอภาพ สัญลักษณ์ << หมายถึง Output operator หรือ insertion operator เป็นการนำข้อความส่งตามมาด้วย ซึ่งเป็นค่า area ที่คำนวณได้จากบรรทัดที่ 17 การส่งข้อความทั้งหลายไปยัง cout จะต่อเนื่องเหมือนกับเป็นกระแสข้อความ ค่าในตัวแปรทั้งหลายที่ส่งนี้เรียกว่า stream cout จะแสดงผลเรียงตามลำดับก่อนหลังที่ปรากฏอยู่หลังเครื่องหมาย <<

endl หมายถึง end of line เป็นการกำหนดให้ขึ้นบรรทัดใหม่

return 0; เป็นการส่งค่ากลับของฟังก์ชัน main() ไปยังระบบปฏิบัติการ มีค่าเป็นศูนย์เมื่อการทำงานของโปรแกรมเป็นไปอย่างราบรื่น

ข้อควรระวัง ประโยคที่นิยามตัวแปร และ statement ต้องปิดท้ายด้วย ; เสมอ

คอมไพเลอร์ภาษา C++ ในยุคหลัง ๆ รวมทั้ง Borland C ++ Compiler ที่กำลังใช้อยู่นี้ด้วย ได้เพิ่มการใช้งาน “namespace” รวมเข้าไว้ด้วย namespace จะช่วยแบ่งขอบเขตการเรียกใช้คลาส ตัวแปร เมธอดต่าง ๆ มิให้โปรแกรมสับสน นิยมใช้กับโปรแกรมที่มีขนาดใหญ่ มีผู้ร่วมเขียนโปรแกรมหลายคน มีการเรียกใช้ Library จากที่แหล่งต่าง ๆ เป็นจำนวนมาก เพราะบางครั้งอาจมีการตั้งชื่อคลาส ตัวแปร หรือเมธอดซ้ำกัน การกำหนดขอบเขตให้ชัดเจน จะทำให้โปรแกรมสามารถเรียกใช้คลาส ตัวแปร หรือ เมธอดได้อย่างถูกต้อง

เนื่องจากโปรแกรมของเรามีขนาดสั้น ตัวแปรที่ใช้ และฟังก์ชันต่าง ๆ จึงมีไม่มากนัก ในที่นี้จึงไม่ใช้ namespace ใส่ไว้ในโปรแกรม

พารามิเตอร์ของฟังก์ชัน main ( )

โปรแกรม 1-1 เป็นการคำนวณหาพื้นที่สี่เหลี่ยมผืนผ้า (สนามฟุตบอล) จะเห็นว่าเรากำหนดความกว้าง ความยาวไว้ในโปรแกรม นั่นหมายถึงถ้าจะใช้โปรแกรมนี้คำนวณพื้นที่สี่เหลี่ยมผืนผ้ารูปอื่น ๆ จะต้องเปลี่ยนค่า width และ length ในโปรแกรมต้นฉบับ จากนั้นจึงคอมไพล์ สร้าง execute file ขึ้นมาใหม่ และทำซ้ำเช่นนี้ทุกครั้ง ที่เปลี่ยนค่าความกว้างและความยาว

จะเป็นการสะดวกถ้าเราจะใช้โปรแกรมนี้คำนวณหาพื้นที่สี่เหลี่ยมผืนผ้าใด ๆ เช่น ต้องการคำนวณหาพื้นที่ของสี่เหลี่ยมผืนผ้า กว้าง 45 เมตร ยาว 72 เมตร สามารถพิมพ์คำสั่งต่อไปนี้ที่ระบบปฏิบัติการ

C: > findarea 45 72 ( กด enter)

จะได้ผลลัพธ์

Rectangle area = 3240 sqr meter

```
C:\MyCpp\c01>bcc32 findarea.cpp
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland

FINDAREA.CPP:
Warning W8057 FINDAREA.CPP 27: Parameter 'env' is never used
in function main(int,char * *,char * *)
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland

C:\MyCpp\c01>FindArea 45 72
Area of this rectangle = 3240 sqr.meter

C:\MyCpp\c01>
```

ผลลัพธ์จากโปรแกรม

เราสามารถทำเช่นนี้ได้ โดยอาศัยพารามิเตอร์ของฟังก์ชัน main ดังโปรแกรม 1.2 ดังนี้

```
1:  /* =====
2:  Program 1.2: FindArea.cpp
3:      Find rectangular area at command prompt
4:  Author: Wachara R.
5:  Date: 03 Oct 2003
6:  Last Update: -
7:  ----- */
8:  #include <iostream.h>
9:  #include <math.h>
10: int main(int argc, char** argv, char**env) {
11:     float width, length, area;
12:
13:     if (argc < 3) {
14:         cout << "How to use this program " << endl
15:         << "C:\>testarg width length (press ENTER)" << endl;
16:     } else {
17:         width = atof(argv[1]); // width of rectangle
18:         length= atof(argv[2]); // length of rectangle
19:         //Compute area of rectangle.
20:         area = width*length;
21:         //Print area on standard output
22:         cout << " Area of this rectangle = "
23:         << area << " sqr.meter" << endl;
24:     }
25:     // exit program
26:     return 0;
27: }
28:  // =====
```

เมื่อเราพิมพ์ C: > findarea 45 72 ( กด enter) ค่าความกว้าง 45 เมตรและความยาว 72 เมตร

ระบบปฏิบัติการ จะเก็บค่านี้ไว้ในตัวแปร argv[1] และ argv[2] นำค่าที่ได้ไปเก็บไว้ในตัวแปร width และ

length จากนั้นคำนวณหาพื้นที่สี่เหลี่ยมผืนผ้าเก็บค่าที่ได้ไว้ใน area คำสั่ง cout จะพิมพ์ คำตอบ 3240 ปรากฏบนจอภาพ

โปรแกรมต่อไปนี้จะตรวจสอบพารามิเตอร์ของฟังก์ชัน main() ที่รับค่าต่าง ๆ จากระบบปฏิบัติการ

```
1:  /* =====
2:  Program 1.2A: TestArg.cpp
3:      Test arguments that passed to function main()
4:      Author: Wachara R.
5:      Date: 03 Oct 2003
6:      Last Update: -
7:  ----- */
8:  #include <iostream.h>
9:  int main(int argc, char** argv, char**env) {
10:     int i;
11:     cout << " argc = " << argc << endl;
12:     for (i = 0; i < argc ; i++ ) {
13:         cout << "argv[" << i << "]" ---> " << argv[i] << endl;
14:     }
15:     cout << endl;
16:     i = 0;
17:     while (env[i]) {
18:         cout << "env[" << i << "]" ---> " << env[i] << endl;
19:         i++;
20:     }
21:
22:
23:     // exit program
24:     return 0;
25: }
26: // =====
```

เมื่อพิมพ์คำสั่ง

C:> testarg test1 test2 test3 test4 (กด enter)

ฟังก์ชัน main( ) ของโปรแกรม testarg จะรับค่าพารามิเตอร์มา 5 ค่า ค่าต่าง ๆ จะถูกเก็บไว้ใน argv [ ] ดังรูป

```
argc = 5
argv[0] ---> C:\MyCpp\chap01\TESTARG.exe
argv[1] ---> test1
argv[2] ---> test2
argv[3] ---> test3
argv[4] ---> test4
```

```
env[0] ---> =::=:\
env[1] ---> =C:=C:\MyCpp\chap01
env[2] ---> =ExitCode=00000000
:
:
:
```

ตัวแปร env[ ] จะเก็บค่าสภาพแวดล้อมของระบบปฏิบัติการ ซึ่งจะแตกต่างกันไปในแต่ละเครื่อง ขึ้นอยู่กับค่าที่กำหนดไว้ใน autoexec.bat และ config.sys ในระบบปฏิบัติการ DOS เราสามารถใช้คำสั่ง set ตรวจสอบสภาพแวดล้อมนี้ได้

ลองทดลองพิมพ์ C:> testarg one two "three four" five six (กด enter)

```
C:\MyCpp\c01>testarg one two "three four" five six
argc = 6
argv[0] ---> C:\MyCpp\c01\TESTARG.exe
argv[1] ---> one
argv[2] ---> two
argv[3] ---> three four
argv[4] ---> five
argv[5] ---> six
```

## ตัวแปรและค่าคงที่ (Variables and Constants)

ตัวแปรคือเนื้อหาของหน่วยความจำที่ใช้เก็บข้อมูล ข้อมูลที่เก็บนี้อาจมีการเปลี่ยนแปลงค่าในระหว่าง การประมวลผล เราจะกำหนดชื่อ (identifier or variable name) ให้กับตัวแปรเพื่อที่สะดวกในการอ้างถึงใน การเขียนโปรแกรม

การกำหนดชื่อตัวแปรมีหลักดังนี้

- มีความยาวที่ตัวอักษรก็ได้ ต้องขึ้นด้วยตัวอักษร A ถึง Z หรือ a ถึง z และเครื่องหมาย \_ (under score) เท่านั้น ขึ้นต้นด้วยตัวเลขไม่ได้
- อักษรถัดจากตัวแรกจะเป็นตัวอักษร หรือ ตัวเลข หรือ เครื่องหมาย \_
- ห้ามมีช่องว่างหรือเครื่องหมายทางคณิตศาสตร์ เช่น +, -, \*, /, ^, <<, >>, (,), [, ], {, }, %, \$, &
- อักษรตัวใหญ่ตัวเล็ก มีผลต่อชื่อตัวแปร เช่น Area กับ area ภาษา C++ จะถือว่าเป็นตัวแปรคนละตัวกัน
- ชื่อของตัวแปรจะต้องไม่ซ้ำกับคำที่สงวนไว้ ( Reserved word or token) ในภาษา C++ มี 109 คำดังนี้

and	and_eq	asm	auto	bitand	bitor	bool	break
case	catch	char	class	const	const_cast	continue	default
delete	do	double	dynamic_cast	else	enum	explicit	export
extern	false	float	for	friend	goto	if	inline
int	long	mutable	namespace	new	not	not_eq	operator
or	or_eq	private	protected	public	register	reinterpret_cast	return
short	signed	sizeof	static	static_cast	struct	switch	template
this	throw	true	try	typedef	typeid	typename	union
unsigned	using	virtual	void	volatile	wchar_t	while	Xor
xor_eq							

คำสั่งบางคำ คอมไพเลอร์รุ่นเก่าไม่ได้ับรวมหรือจัดไว้ว่าเป็นคำต้องห้าม ถึงกระนั้นก็ตามเราควรหลีกเลี่ยงชื่อตัวแปรโดยไม่ใช้ชื่อซ้ำกับคำเหล่านี้ คอมไพเลอร์บางบริษัทมีการเพิ่มคำสั่งเฉพาะเข้าไปอีก เช่น Borland compilers for the PC มีการเพิ่มคำว่า near, far, huge, cdecl และ pascal

จะเห็นว่า main, cin และ cout ไม่ใช่เป็นคำสั่ง แต่ในทางปฏิบัติไม่ควรตั้งชื่อตัวแปรให้ซ้ำกับคำเหล่านี้เช่นกัน

- การตั้งชื่อตัวแปรควรสื่อความหมาย เพื่อประโยชน์ในการทำความเข้าใจการทำงานของโปรแกรม และสะดวกในการตรวจแก้ไขโปรแกรม
- ในระบบปฏิบัติการวินโดวส์ ได้เสนอแนะให้ใช้การตั้งชื่อตัวแปรแบบ Hungarian notation ซึ่งในชื่อตัวแปรนั้นจะบอกชนิดของข้อมูล วัตถุประสงค์ของตัวแปรว่าจะนำไปใช้ทำอะไร
- หลักการตั้งชื่อตัวแปรสามารถนำไปใช้กับชื่อฟังก์ชัน คลาส และ structure ได้อีกด้วย

แบบฝึกหัด ชื่อตัวแปร ต่อไปนี้ ถูกหรือผิด ถ้าผิดบอกเหตุผลด้วย

Perimeter	x_sum	3dimension	f(x)	fx	operator	tax_payment
Km/hr	subtotal	variable1	counter	far	null	Net_Pay
Next_character	_x	get-result	H2O	_CLOCK_		x&y
_in_\$	4mypet	PuZzle	tip4you	myCash		while
Percent	%rate					

## ชนิดของข้อมูล (Data type)

จะเป็นตัวบอกให้คอมไพเลอร์รู้ว่าข้อมูลที่เก็บไว้ในหน่วยความจำ จะนำไปประมวลผลในลักษณะใดบ้าง แบ่ง ข้อมูลที่สามารถนำไปคำนวณได้ (numeric data type) หรือเป็นข้อมูลที่ไม่สามารถนำไปคำนวณได้ เช่น อักขระ ต่าง ๆ หรือข้อมูลตรรกะ (Logical data type)

ขนาดของหน่วยความจำที่ใช้เก็บข้อมูลจะแปรเปลี่ยนไปตามระบบปฏิบัติการ หรือตามแพลตฟอร์มของคอมพิวเตอร์ที่ใช้ประมวลผล คอมไพเลอร์ที่ถูกสร้างขึ้น ในระบบปฏิบัติการ DOS และใน Unix จะใช้เนื้อที่ในการเก็บข้อมูลต่างกัน ความแม่นยำ ความถูกต้องของจำนวนเลข ค่าสูงสุด ค่าต่ำสุดที่เก็บได้จะต่างกันไปด้วย แต่โดยทั่วไปขนาดของหน่วยความจำที่ใช้เก็บข้อมูลจะเป็นไปตามตารางข้างล่างนี้

ตารางจำแนกชนิดของข้อมูลใน C++

ชนิดข้อมูล	รายละเอียด	จำนวน byte	ค่าที่เป็นไปได้ต่ำสุด	ค่าที่เป็นไปได้สูงสุด
ข้อมูลที่เก็บแบบจำนวนเต็ม				
char	ข้อมูลชนิดตัวอักษรโดยเก็บรหัสASCII ของตัวอักษรนั้น	1	-128	127
unsigned	ข้อมูลชนิด	1	0	255

char	ตัวอักษรไม่คิดเครื่องหมาย			
short	จำนวนเต็ม	2	-32,768	32,767
unsigned short	จำนวนเต็มไม่คิดเครื่องหมาย	2	0	65,535
long	จำนวนเต็มทั้งบวกและลบ	4	-2,147,483,648	2,147,483,647
unsigned long	จำนวนเต็มบวก	4	0	4,294,967,295
int	เช่นเดียวกับ long	2	-32,768	32,767
unsigned int	เช่นเดียวกับ unsigned long	2	0	65,535
int64	จำนวนเต็ม	8	-9,223,372,036,854,775,807	9,223,372,036,854,775,808

ชนิดข้อมูล	รายละเอียด	จำนวน byte	ค่าที่เป็นไปได้ต่ำสุด	ค่าที่เป็นไปได้สูงสุด
ข้อมูลที่เก็บแบบเลขทศนิยม (Floating Point)				
float	จำนวนจริงแบบ single precision	4	1.75494E-38	3.402823E+38
double		8	2.2250738585E-308	1.7976931348E+308
long double	จำนวนจริงแบบ double precision	10	3.3621031431E-4932	1.1897314953E+4932
ข้อมูลชนิดตรรกะ (Boolean data type)				
bool	ข้อมูลชนิดตรรกะ	1	true	false

ถ้าเราต้องการทราบว่าขอบเขตของชนิดข้อมูลในแพลตฟอร์มที่เราใช้งานอยู่ ทำได้โดยพิมพ์ค่าขีดจำกัดบนและล่างของข้อมูลแต่ละชนิด โดยอาศัยค่าคงที่ ที่กำหนดไว้ใน header file ที่ชื่อว่า limits.h และ float.h ดังตัวอย่างต่อไปนี้

```

1: // Program 1.3 testlimit.cpp
2: //      Print the limits to the various data type
3: //-----
4: #include <iostream.h>
```





```

14: long int li2;
15: unsigned long uli;
16: float f;
17: double d;
18: long double ld;
19: cout << " Data Type\t\tSize in Byte" << endl
20: << "-----" << endl;
21: cout << "char\t\t\t" << sizeof (ch) << endl
22: << "unsigned char\t\t" << sizeof (uch) << endl
23: << "int\t\t\t" << sizeof (i) << endl
24: << "unsigned int\t\t" << sizeof (ui) << endl
25: << "short\t\t\t" << sizeof (si1) << endl
26: << "short int\t\t" << sizeof (si2) << endl;
27: cout << "unsigned short int\t\t" << sizeof (usi) << endl
28: << "long\t\t\t" << sizeof (li1) << endl
29: << "long int\t\t" << sizeof (li2) << endl
30: << "unsigned long int\t\t" << sizeof (uli) << endl
31: << "float\t\t\t" << sizeof (f) << endl
32: << "double\t\t\t" << sizeof (d) << endl
33: << "long double\t\t" << sizeof (ld) << endl;
34:
35: return 0;
36: }

```

ข้อสังเกต sizeof ถือว่าเป็น operator ตัวหนึ่ง ใช้บอกขนาดของหน่วยความจำ ที่ใช้เก็บข้อมูลชนิดนั้น sizeof

(ch) อาจเขียนเป็น sizeof ch ก็ได้ โดยที่ไม่ต้องใส่วงเล็บ

ผลของการรันโปรแกรมที่ 1.4 จะเป็นดังนี้

Data Type	Size in Byte
char	1
unsigned char	1
int	4
unsigned int	4
short	2
short int	2
unsigned short int	2
long	4
long int	4
unsigned long int	4
float	4
double	8
long double	10

## ค่าคงที่ (Symbolic Constants)

ถ้าข้อมูลนั้นไม่มีการเปลี่ยนแปลงค่าตั้งแต่ต้นโปรแกรมจนจบการทำงาน เรายินยอมกำหนดข้อมูลนั้นให้เป็นค่าคงที่ การกำหนดชื่อค่าคงที่ จะนำหน้าด้วย const ตามด้วยชนิดของข้อมูล และชื่อค่าคงที่ ปิดท้ายประโยคด้วย ; เสมอ เช่น

```
Const double PI = 3.141592653;
```

เมื่อนำไปใช้ในโปรแกรม เช่น area = PI \* radius \* radius ; ค่า PI จะถูกแทนที่ด้วย 3.141592653..

ตัวอย่างค่าคงที่อื่น ๆ ที่ใช้บ่อยในการคำนวณทางวิทยาศาสตร์ กำหนดให้เป็นค่าคงที่จะได้

```
const double EARTH_GRAVITY      = 9.81; // acceleration due to gravity
const double ELECTRIC_CHARGE    = 1.602177e-19 ; // an Electric charge
const double LIGHT_SPEED        = 2.999792e+8; // speed of light in vacuum
const int     DAY_IN_A_YEAR      = 365;
const char    TAB                 = '\t';
const float   VATRATE            = 0.07;
const         SECOND_IN_HOUR     = 3600;
```

บรรทัดสุดท้าย ไม่มีการระบุชนิดข้อมูล จะถือว่าเป็นข้อมูลชนิด int

คอมไพเลอร์จะมองเห็นค่าคงที่เป็นเลขฐานสิบเสมอ เราสามารถเขียนค่าคงที่ให้คอมไพเลอร์มองเห็นเป็นเลขฐานแปด หรือ ฐานสิบหกได้ จำนวนเลขโดดที่ขึ้นต้นด้วย 0 (ศูนย์) นำหน้า C++ จะคิดว่าเป็นเลขฐานแปด เช่น

023 หมายถึง 19 ในเลขฐานสิบ

010 หมายถึง 8 ในเลขฐานสิบ

จำนวนเลขโดดที่ขึ้นต้นด้วย 0x (ศูนย์ กับ เอกซ์) จะมองว่าเลขฐานสิบหก เช่น

0xa หรือ 0xA หมายถึง 10 ในเลขฐานสิบ

0x2f หรือ 0x2F หมายถึง 47 ในเลขฐานสิบ

จำนวนจริง เขียนได้ในรูปแบบสัญกรณ์นิยม หรือใช้ตัวอักษร e แทนการเขียนตัวเลขยกกำลัง เช่น

0.0

32.68

-3e5 หมายถึง  $-3 \times 10^5$

ถ้าต่อท้ายตัวเลขด้วย f หรือ F จำนวนเลขนั้นจะถูกเก็บเป็นแบบข้อมูลชนิด float ถ้าต่อท้ายตัวเลขด้วย l หรือ L จำนวนเลขนั้นจะถูกเก็บเป็นแบบข้อมูลชนิด long double จำนวนจริงที่ไม่มีตัวอักษร f(F) หรือ l(L) ต่อท้าย

คอมไพเลอร์จะมองเห็นเป็น double

ข้อมูลชนิด char อักขระ 1 ตัวจะอยู่ในเครื่องหมาย single quote เช่น 'a', '3', '\n' เมื่อมีอักขระหลาย ๆ ตัว หรือ วลี จะอยู่ในเครื่องหมาย double quote เช่น "Area of rectangle", "Hello, world". เรียกอักขระหรือวลีเหล่านี้ว่า String

ข้อมูลชนิด char ยังสามารถเก็บสัญลักษณ์พิเศษที่ไม่สามารถแสดงผลบนจอภาพ เรียกอักขระเหล่านี้ว่าเป็น escape sequence ประกอบด้วยตัวอักษร 1 ตัว นำหน้าด้วย \ ( back slash)

Escape sequence	ความหมาย
\a	alert character (bell)
\b	backspace
\f	form feed
\n	New line
\r	Carriage return
\t	Horizontal tab

\v	Vertical tab
\\	Back slash
\?	Question mark
\'	Single quote
\"	Double quote
\xff	Character constant in hex

การกำหนดค่าให้กับตัวแปร (Assignment)

การกำหนดค่าให้ตัวแปรใด ๆ คือ การนำค่าคงที่หรือชื่อของตัวแปรหรือ ผลที่เกิดจากการคำนวณทางคณิตศาสตร์ ไปใส่ไว้ในหน่วยความจำที่มีชื่อเป็นของตัวแปรนั้น เช่น

Float width, length, area;

```
width = 30.0;
length = 120.0;
area = width*length;
```

เป็นการกำหนดค่าตัวแปรชื่อ width และ length มีค่าเป็น 30.0 และ 120.0 (30.0 และ 120.0 เป็นค่าคงที่ที่เป็นตัวเลขแบบทศนิยม) และกำหนดค่าตัวแปร area มีค่าเท่ากับผลคูณของตัวแปร width และ length จากตัวอย่าง ถ้าเปลี่ยนการกำหนดค่าเป็นดังนี้

```
width = 30.0;
length = width;
```

เป็นการกำหนดค่าให้ length มีค่าเท่ากับ width คือ 30.0 อาจเขียนให้อยู่ในบรรทัดเดียวกันได้ดังนี้

```
width = length = 30.0;
```

แต่เราไม่สามารถเขียนการกำหนดค่า 20 = width เพราะกำหนดค่าตัวแปรให้กับค่าคงที่ไม่ได้

ถ้าการกำหนดค่าตัวแปรให้ไม่ตรงกับชนิดของข้อมูล ข้อมูลจะถูกแปลงค่าให้เหมาะสมกับชนิดของข้อมูลของตัวแปรนั้น เช่น

```
int x;
x = 4.5;
```

x เป็นข้อมูลชนิด จำนวนเต็ม ไม่สามารถเก็บเลขทศนิยม ดังนั้นตัวแปร x จะเก็บค่า x = 4 ไว้ในหน่วยความจำเท่านั้น ข้อมูลจะไม่ถูกต้องสมบูรณ์ ถ้าเราใส่ข้อมูลชนิด float ให้กับตัวแปรชนิด integer

ข้อมูลที่เรากำหนดให้ตัวแปรจะมีการสูญเสียหรือไม่ตรงกับความเป็นจริงหรือไม่ ให้ตรวจสอบกับลำดับของชนิดข้อมูลโดยเรียงจากขนาดหน่วยความจำและตัวเลขสูงสุดที่เก็บได้ดังนี้

สูง	long double
	double
	float
	long int

	int
	short int
ต่ำ	char

กรณีแรก ถ้าข้อมูลที่มีชนิดสูงกว่าถูกกำหนดให้แก่ตัวแปรที่มีชนิดข้อมูลต่ำกว่า ข้อมูลจะมีการสูญหาย และผลลัพธ์ที่ได้อาจไม่ถูกต้อง กรณีที่ 2 ถ้ากำหนดข้อมูลที่มีชนิดต่ำกว่าให้กับตัวแปรข้อมูลที่มีชนิดข้อมูลสูงกว่า ข้อมูลจะถูกตัดทิ้ง ไม่เกิดปัญหา

จะเห็นว่าไม่ได้นำข้อมูลชนิด unsigned integer เข้ามาไว้ในรายการ เพราะจะทำให้เกิดความคลาดเคลื่อนเสมอ ทั้งสองกรณี อย่างไรก็ตามควรกำหนดค่าให้ถูกต้องกับชนิดของข้อมูลจะเป็นการดีที่สุด

เมื่อกำหนดค่าคงที่ให้แก่ข้อมูล คอมไพเลอร์จะรู้ว่าเป็นข้อมูลชนิดใดจะดูจากรูปแบบการเขียนจำนวนเลขดังต่อไปนี้

int จะเป็นตัวเลขที่ไม่มีทศนิยม เช่น 1, 0, -8, 123, -6385, 0xff (เลขฐานสิบหก)

long int เป็นเลขจำนวนเต็มที่มี L ต่อท้าย เช่น 1L, 0L, -8L, 123L, -6385L, 0xffL (ใช้ l ก็ได้ แต่จะดูคล้ายเลขหนึ่ง)

double ตัวเลขจำนวนจริงที่อยู่ในรูปทศนิยม หรือเลขตรรกยะ เช่น 1.0, 0.0, 1234.56, 5e2 ( $5 \times 10^2$ ), 3.2e6 ( $3.2 \times 10^6$ ), 6.23e-34 ( $6.23 \times 10^{-34}$ )

โปรแกรม 1.5 จะเป็นการทดสอบว่า เมื่อไม่กำหนดค่าให้กับตัวแปร และบางแห่งมีการกำหนดค่าให้กับตัวแปร คอมไพเลอร์จะแสดงผลการคอมไพล์อย่างไรบ้าง

```

1: //Program 1.5 TestDataTypes.cpp
2: //    Test Basic data type
3: //
4: #include <iostream.h>
5: main () {
6: //definition without initialization
7:     char character;
8:     int integer;
9:     float floatnum;
10:    double dfloatnum;
11: // definition with initialization
12:    char ch='A', people='p';
13:    int i=2, j=6;
14:    float height= 1.8, base = 7.0;
15:    double charge = 1.6e-19;
16: //Print out uninitialized variables
17:    cout << "\ncharacter (char Not initialized) = " << character;
18:    cout << "\ninteger ( int not initialized)  = " << integer;
19:    cout << "\nfloatnum (float not initialized) = " << floatnum;
20:    cout << "\ndfloatnum (double not initialized)= " << dfloatnum;
21: // Print out initialized variable
22:
23:    cout << "\n ch (char data type) = " << ch
24:    << "\n i (integer type)  = " << i
25:    << "\n height (float type) = " << height
26:    << "\n charge (double type) = " << charge

```

```

27:                                     << endl;
28:
29:             return 0;
30:     }

```

ผลลัพธ์การรันโปรแกรม 1.5 จะเป็นดังนี้

```

character (char Not initialized) =
integer ( int not initialized) = 256
floatnum (float not initialized) = 1.4013e-45
dfloatnum (double not initialized)= 2.01678e-307
ch (char data type) = A
i (integer type) = 2
height (float type) = 1.8
charge (double type) = 1.6e-19

```

ก่อนนำตัวแปรไปใช้งานจะต้องกำหนดค่าเริ่มต้นให้ก่อนเสมอ เพราะคอมพิวเตอร์จะไม่เตือนเรื่องการกำหนดค่าเบื้องต้นให้ทราบ สิ่งที่โปรแกรมนำมาแสดงคือ ข้อมูลขยะ ผู้เขียนโปรแกรมจะต้องเป็นฝ่ายรับผิดชอบและรอบคอบในการกำหนดค่าเริ่มต้นให้กับตัวแปรเอง

#### การแปลงชนิดข้อมูล (Type Casting)

เมื่อกำหนดค่าให้แก่ตัวแปรซึ่งเป็นชนิดข้อมูลต่างกัน เช่น กำหนดตัวแปรไว้เป็นจำนวนเต็ม เช่น

```
int i;
```

```
i= 3.14;
```

ตัวแปร i จะเก็บค่าไว้เพียง i = 3; เพราะ i เป็นตัวแปรแบบจำนวนเต็ม จะไม่เก็บค่าทศนิยม

จากตัวอย่างโปรแกรม 1.1 เรากำหนดข้อมูล width และ length เป็น float แต่เมื่อป้อนจำนวนเต็ม เช่น width=20 length = 15 ตัวแปรชนิด float สามารถรับจำนวนเต็มได้ โดย C++ จะแปลงจำนวนเต็มให้เป็นจำนวนจริง ค่าที่เก็บไว้ใน width และ length จึงเป็น 20.0 และ 15.0 ตามลำดับ

เมื่อกำหนดค่าข้อมูลที่มีชนิดต่ำกว่า ให้แก่ตัวแปรที่มีข้อมูลชนิดสูงกว่า ไม่จำเป็นต้องใส่ type cast operator ดังตัวอย่างโปรแกรมที่ 1.6 เป็นการแปลงข้อมูลชนิด char ให้เป็น short ให้เป็น int ให้เป็น long ไปเรื่อย ๆ จนถึง double

```

1:    //Program 1.6 TestCast.cpp
2:    //    Test order of variables
3:    //
4:    #include <iostream.h>
5:    main () {
6:    char ch = 'a'; cout << "\n ch (char type)  = " << ch ;
7:    short sh = ch; cout << "\n sh (short type) = " << sh;
8:    int i  = sh; cout << "\n i (int type)    = " << i;
9:    long n  = i; cout << "\n n (long type)   = " << n;
10:   float f  = n; cout << "\n f (float type)  = " << f;
11:   double df = f; cout << "\n df (double type) = " << df;
12:   return 0;
13:   }

```

ผลลัพธ์จากการกำหนดค่า ch ซึ่งเป็นตัวแปรชนิด char ให้มีค่าเป็น "a" ซึ่งมีค่ารหัสอักขระเท่ากับ 97 เมื่อ  
กำหนดค่านี้ให้กับข้อมูลที่มีชนิดสูงกว่า พบว่าข้อมูลที่เก็บไม่มีความคลาดเคลื่อน ผลของการทำงานของ  
โปรแกรม 1.6 จะเป็นดังนี้

```
ch (char type)  = a
sh (short type) = 97
i (int type)    = 97
n (long type)   = 97
f (float type)  = 97
df (double type) = 97
```

ถ้าการคำนวณทางคณิตศาสตร์เกี่ยวข้องกับข้อมูลต่างชนิดกัน ผลลัพธ์ที่ได้อาจเกิดความ  
คลาดเคลื่อน ตัวอย่างเช่น

```
int i=12, j=4, k=3;
int m;
```

```
m = i*(j/k);
```

ค่าของที่คำนวณได้คือ 12 ไม่ใช่ 16 เพราะ  $j/k = 4/3 = 1$  (เก็บเฉพาะจำนวนเต็ม ตัดเลข  
ทศนิยมทิ้ง) ผลลัพธ์จึงได้ 12

เราสามารถใส่คำสั่ง ที่มีรูปแบบ ดังนี้ type(value) เมื่อ type คือชนิดของข้อมูลที่ต้องการ  
value คือข้อมูลของอีกชนิดหนึ่งซึ่งต้องการเก็บค่าไว้ เช่น

```
x = 6.3124;
```

ต้องการเก็บค่าไว้ในตัวแปร i ซึ่งเป็นข้อมูลชนิดจำนวนเต็ม สามารถเขียนได้เป็น

```
i = int (x); หรือ i = (int) x
```

ตัวอย่างผลลัพธ์ที่เกิดจากการแปลงชนิดข้อมูล

```
int (4.5) = 4
float (2) = 2.0
float (3/2) = 1.0
float(3)/float(2) = 1.5
```

```
1: //Program 1.7 TestCas2.cpp
2: // Test order of variables
3: //
4: #include <iostream.h>
5: main () {
6:     double x = 6.3214;
7:     int m,n;
8:
9:         m = 6.3214;
10:        n = int(x); // or n = (int) x
11:        cout << "m = " << m << endl;
12:        cout << "n = " << n << endl;
13: }
```

ผลลัพธ์จากการรันโปรแกรม จะเห็นว่า ถึงแม้จะใส่คำสั่ง casting ข้อมูลในบรรทัดที่ 9 ผลลัพธ์ที่ได้มีค่าเท่ากับ  
กำหนดค่าโดยตรง

```
m = 6
n = 6
```

เมื่อนำข้อมูลชนิดสูงกว่า เช่น double ไปเก็บไว้ในข้อมูลชนิดต่ำกว่า เช่น integer ดังในตัวอย่าง 1.7 พบว่าเลข  
ทศนิยมจะถูกตัดทิ้ง (truncate) ไม่มีการปัดเศษ

เครื่องหมาย หรือตัวปฏิบัติการ (Operator)

จะเป็นตัวบอกกับโปรแกรมว่าจะให้จัดการกับข้อมูลทั้งนั้นอย่างไร เช่น นำข้อมูลมาคำนวณ นำมาตรวจสอบว่าข้อมูลทั้งสองชุดเท่ากันหรือไม่ หรือสลับค่าข้อมูลในตัวแปร

ในโปรแกรม 1.1 เราได้พบกับเครื่องหมาย = หมายถึงการกำหนดค่าให้กับตัวแปร และ  $width * length$  เครื่องหมาย \* หมายถึงนำตัวแปรทั้งสองมาคูณกัน

ในภาษา C++ มีเครื่องหมายให้ใช้เป็นจำนวนมาก สามารถจำแนกเครื่องหมายที่ใช้บ่อย ๆ ให้เป็นหมวดหมู่ดังนี้

สัญลักษณ์	ความหมาย	ตัวอย่าง	หมายเหตุ
+ หรือ -	บอกว่าเป็นจำนวนบวกหรือลบ	$+x, +3, +5.2, -x, -3, -5.2$	เป็น unary operator
+	การบวก (Addition)	$z = x + y$	
-	การลบ (Subtraction)	$z = x - y$	
*	การคูณ (multiplication)	$z = x * y$	
/	การหาร (division)	$z = x / y$	
%	การหารเก็บเศษ (modulus)	$z = x \% y$ 5 % 2 จะได้ผลลัพธ์ 1 เก็บเฉพาะเศษไว้	ไม่สามารถใช้กับเลขทศนิยมได้

ภาษา C++ มีเครื่องหมายที่เกี่ยวข้องกับการคำนวณและใช้บ่อย ได้แก่

$++$  หมายถึงการเพิ่มค่าครั้งละ 1 เช่น  $x++$  หมายถึง  $x = x + 1$  (เพิ่มค่าข้อมูลในตัวแปร  $x$  อีก 1 จากนั้นนำค่าที่ได้ไปเก็บไว้ในตัวแปร  $x$  เช่นเดิม

$--$  หมายถึงการลดค่าครั้งละ 1 เช่น  $x--$  หมายถึง  $x = x - 1$  (ลดค่าข้อมูลในตัวแปร  $x$  อีก 1 จากนั้นนำค่าที่ได้ไปเก็บไว้ในตัวแปร  $x$

### นิพจน์ทางคณิตศาสตร์ (Arithmetic expression)

เราสามารถนำตัวแปร เครื่องหมายทางคณิตศาสตร์ ค่าคงที่ มาเรียบเรียงเป็นนิพจน์เพื่อนำไปใช้ในการคำนวณได้ สิ่งที่ต้องพึงระวังเป็นอย่างยิ่งในการเขียนข้อความทางคณิตศาสตร์ คือ ต้องรู้ลำดับการทำงานของเครื่องหมาย มิฉะนั้นจะได้ผลการคำนวณออกมาคลาดเคลื่อน โดยที่คอมพิวเตอร์ไม่แสดงข้อความเตือนหรือแจ้งข้อผิดพลาดให้ทราบ เช่น

$$X = 5 + 2 * 3$$

จะได้ค่า  $x$  เท่ากับ 11 ไม่ใช่ 21 เพราะการคูณมีลำดับการทำงานก่อนเครื่องหมายบวก เพื่อให้ดูเข้าใจง่าย ควรเขียนเป็น

$$X = 5 + (2 * 3)$$

ลำดับการทำงานก่อนหลังของเครื่องหมายทางคณิตศาสตร์เป็นไปดังตารางต่อไปนี้ เครื่องหมายที่อยู่ส่วนบนของตารางจะมีลำดับความสำคัญกว่าลำดับที่อยู่รองลงไป เครื่องหมายที่มีลำดับความสำคัญเท่ากัน จะ

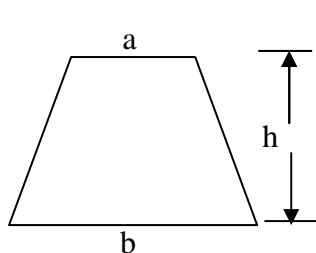


เริ่มกระทำจากซ้ายไปขวา หรือจากขวาไปซ้ายตามที่กำหนดไว้ในตาราง การคูณจะเริ่มคำนวณจากทางซ้ายไปขวามือ  $x*y*z$  หมายถึง  $(x*y)*z$  ไม่ใช่  $x*(y*z)$

ลำดับ	เครื่องหมายหรือตัวปฏิบัติการ (operator)	เริ่มประมวลผลจาก(Associativity)
1	( )	parentheses
2	+, -, ++, --, cast	unary operator
3	*, /, %	การคูณ การหาร และการหารเก็บเศษ
4	+, -	การบวก การลบ
5	=, +=, -=, *=, /=, %=	assignment

ตัวอย่างการเขียนนิพจน์ทางคณิตศาสตร์

ก. พื้นที่สี่เหลี่ยมคางหมู =  $\frac{1}{2} \times \text{สูง} \times \text{ผลบวกของด้านคู่ขนาน}$



$$\text{Area} = 0.5 * h * (a + b);$$

ข. ระยะกระจัดของก้อนหินที่ตกลงมาอย่างอิสระภายใต้สนามโน้มถ่วงของโลก ด้วยความเร็วต้น  $u$  ที่เวลา  $t$

ใด ๆ คือ  $y = ut + \frac{1}{2}gt^2$  เมื่อ  $g$  คือความเร่งเนื่องจากแรงดึงดูดของโลก

$$y = u * t + 0.5 * g * t * t;$$

ค. แรงไฟฟ้าเนื่องจากจุดประจุขนาด  $q_1$  และ  $q_2$  วางห่างกันเป็นระยะ  $r$  ใด ๆ หาได้จาก

$$F = \frac{kq_1q_2}{r^2} \quad \text{เมื่อ } k \text{ คือค่าคงที่}$$

เขียนเป็นนิพจน์ทางคณิตศาสตร์ในภาษา C++ ได้ดังนี้

$$F = k * q1 * q2 / (r * r);$$

โปรแกรมที่ 1.8 จะเป็นการตรวจสอบลำดับการคำนวณของเครื่องหมายและตัวปฏิบัติการ

```

1: //Program 1.8 Testprecedence.cpp
2: //
3: //
4: #include <iostream.h>
5: main () {
6: int i=12, j=4, k=3;
7:     cout << "\n i + j * k = " << i+j*k;
8:     cout << "\n i*j + k = " << i*j +k;
```

```

9:      cout << "\n i * j / k = " << i*j/k;
10:     cout << "\n i* (j + k)  = " << i*(j /k);
11:     return 0;
12:
13:}

```

คำตอบ ในบรรทัดที่ 10 ไม่ตรงกับที่คาดคะเนไว้ เพราะว่า เรากำหนดตัวแปร i, j, k เป็นชนิดจำนวนเต็ม เมื่อนำ  $j/k = 4/3 = 1$  ส่วนที่เป็นทศนิยมจะถูกตัดทิ้งไป ทำให้ผลลัพธ์ในบรรทัดที่ 9 และ 10 ไม่เท่ากัน

กำหนดให้ x เป็นตัวแปรชนิดจำนวนเต็ม จงหาผลลัพธ์ของนิพจน์ต่อไปนี้

$$\begin{aligned}
 x &= 8 * 10 - 9 \% 4 * 3 + 9 &= 80 - (9 \% 4) * 3 + 9 \\
 & &= 80 - 2 * 3 + 9 \\
 & &= 80 - 6 + 9 \\
 & &= 83 \\
 x &= (8 * (10 - 9) \% 4) * 3 + 9 &= (8 * 1 \% 4) * 3 + 9 \\
 & &= (0) * 3 + 9 \\
 & &= 9 \\
 x &= (8 + 9) \% 5 / 2 &= 17 \% 5 / 2 \\
 & &= 2 / 2 \\
 & &= 1 \\
 x &= -6 * 8 / -5 &= -48 \% -3 \\
 & &= 0
 \end{aligned}$$

เราสามารถเขียนการกำหนดค่าทางคณิตศาสตร์ให้สั้นลงกว่าเดิมได้

ข้อความเต็ม	ข้อความย่อ
identifier = identifier operator expression;	identifier operator=identifier;
y = y+1;	y++; หรือ y +=1;
y = y-1;	y--; หรือ y -=1;
z=z-2;	z-=2;
d=d/4.5;	d/=4.5;
r=r%3;	r%=3;

ตัวอย่างอื่นๆ

$x = x + 1;$

$z = x + y;$

นิพจน์ทางคณิตศาสตร์ 2 บรรทัดนี้ เขียนให้รวบรัดอยู่ในบรรทัดเดียวกันได้ดังนี้

$z = x++ + y;$

การเขียนนิพจน์  $x++$  หรือ  $x--$ —เช่นนี้เรียกว่าเขียนแบบต่อท้าย (postfix) หมายถึงค่า x ถูกนำไปใช้ในการคำนวณก่อนแล้วจึงมีการเพิ่มค่าหรือลดค่า

ถ้าเขียนแบบนำหน้า(prefix) ++x หรือ -x หมายถึงมีการเพิ่มค่าหรือลดค่าก่อน แล้วจึงนำค่า x ไปคำนวณ  
ตัวอย่าง

Int i=4; j=2;

I= i\*j++;                      จะได้ i=8 และ j =3 นำ j ไปคูณกับ i ก่อน แล้วจึงเพิ่มค่า j

Int i=4; j=2;

I= i\*++j;                      จะได้ i=12 และ j =3 เพิ่มค่า j ก่อน แล้วจึงนำ j ไปคูณกับ i

Int i=4; j=2;

I= i\*j--;                      จะได้ i=8 j =2

Int i=4; j=2;

I= i\*--j;                      จะได้ i=4 j =2

แบบฝึกหัด

จงตรวจสอบว่านิพจน์ต่อไปนี้ถูกต้องตามหลักของภาษา C++ หรือไม่ ถ้าถูกต้องให้หาผลลัพธ์

- ก. 24/8\*3;                      เฉลย 9
- ข. 24/8/3;                      เฉลย 3
- ค. 32%3\*2 ;                      เฉลย 4
- ง. 6(8+4);                      ไม่ถูกต้อง ต้องมีเครื่องหมาย \* หลังเลข 6
- จ. 18%5%2                      เฉลย 1
- ฉ. 18%(5%2)                      เฉลย 0

จงเขียนคำสั่งต่อไปนี้ให้จบในบรรทัดเดียว

- ก.    price=price+1;  
             sum = sum + price;  
(เฉลย sum += +price;)
- ข.        a= a-(b+c);  
             c = c+1;  
(เฉลย a -= (b+c++); )

กำหนดให้ int x=2, y=3; ในแต่ละข้อ เมื่อโปรแกรมทำคำสั่งแต่ละบรรทัดจะได้ค่า x, y เท่าใด

- ก.    y +=--x;                      เฉลย x= 1, y=4
- ข.    y \*=x++;                      เฉลย x= 3, y=6

กำหนดให้ int x=2, y = 4; ในแต่ละนิพจน์ค่า z (ซึ่งเป็นตัวแปรชนิด int เช่นกัน) จะมีค่าเท่าใด

- ก.    z = ++x\*y;                      เฉลย z = 12

ข.  $z = x++*y$ ; เฉลย  $z = 8$

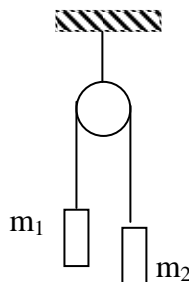
จงเขียนนิพจน์คณิตศาสตร์ต่อไปนี้ให้อยู่ในรูปประโยคคำสั่งของภาษา C++

ก.  $x^3 - 2x^2 + 5x + 7$

ข.  $\frac{x^2 - 1}{x^3 + 1}$

ค. ความเร่งของระบบมวล  $m_1$  และ  $m_2$  คล้องกับเชือกเบาและรอกไม่มีความฝืดมีค่าเป็น

$$a = \frac{m_1 m_2}{m_1 + m_2} g$$



ง. สมการสถานะแก๊สของแวนเดอร์วาลส์ (Van der Waals) มีรูปสมการเป็น

$$RT = \left(P - \frac{a}{V^2}\right)\left(\frac{V}{n} - b\right)$$

เมื่อ  $T$  คืออุณหภูมิสมบูรณ์

$P$  คือความดันของแก๊ส

$n$  คือจำนวนโมลของแก๊ส

$a, b$  คือค่าคงที่

$R$  คือค่าคงที่สากลของแก๊ส

$V$  คือปริมาตรของแก๊ส

จ. ความเร็วของทรงกระบอกมวล  $m$  รัศมี  $r$  ที่กลิ้งลงมาถึงปลายพื้นเอียงสูง  $h$  หาได้จาก

$$v = \sqrt{\frac{2gh}{1 + \frac{I}{mr^2}}}$$

เมื่อ  $I$  คือโมเมนต์ความเฉื่อยของทรงกระบอก

จงเขียนโปรแกรมต่อไปนี้

- ระยะทางระหว่างกรุงเทพฯ ถึงเชียงราย เท่ากับ 838 กิโลเมตร จงแปลงระยะทางนี้ให้เป็นไมล์ โดยที่ 1 ไมล์ เท่ากับ 1.6093440 กิโลเมตร
- แปลงน้ำหนักนกยูง 110 ปอนด์ ให้เป็นกิโลกรัม 1 กิโลกรัม เท่ากับ 2.205 ปอนด์
- แปลงอุณหภูมิ 30 องศาเซลเซียส ให้เป็นอุณหภูมิฟาเรนไฮต์ (Fahrenheit) โดยใช้สูตร

$$\frac{C}{5} = \frac{F - 32}{9}$$

- หาพื้นที่ของเซกเตอร์ของวงกลม ซึ่งมีมุมระหว่างรัศมี ( $d$ , degree) 30 องศา รัศมี ( $r$ ) ยาว 2 เมตร พื้นที่ของเซกเตอร์คือ  $\frac{1}{2}r^2\theta$  เมื่อ  $\theta$  คือมุมที่วัดเป็นเรเดียน
- จงหาปริมาตร ( $V$ , volume) และพื้นที่ผิว ( $A$ , Surface area) ของทรงกลมที่มีรัศมี 15 เซนติเมตร

คำถาม ถ้ากำหนดค่า unsigned int num = -1; การคอมไพล์จะเป็นอย่างไร

คำตอบ คอมไพเลอร์จะเตือน ค่า -1 จะถูกตีความเป็นตัวเลขที่ไม่คิดเครื่องหมาย -1 = 0xff ในเลขฐานสิบหก หรือ 65535 ของเลขฐานสิบ

คำถาม ถ้ากำหนดค่า int number = 6.3;

คำตอบ คอมไพเลอร์จะแจ้งเตือนเช่นเดียวกัน ตัวเลขที่เก็บไว้จะถูกตัดให้เหลือเฉพาะจำนวนเต็มคือ 6

จงตั้งชื่อตัวแปรและเลือกชนิดของข้อมูลให้เหมาะสมกับค่าของข้อมูลที่จะเก็บ

- ก. พื้นที่สนามฟุตบอล
- ข. อายุของนักศึกษา
- ค. รายได้ต่อหัวของประชากรไทย
- ง. จำนวนสุนัขจรจัด ในกรุงเทพ ฯ

ประโยคต่อไปนี้ใช้ comment ถูกหรือไม่

```
cout << "First try in C++ // my first time in C++";
```

โปรแกรมต่อไปนี้เป็นคอมไพล์ผ่านหรือไม่

```
main() {  
    2000; // y2k year  
    return 0;  
}
```

ในแต่ละข้อต่อไปนี้เป็นประกาศตัวแปรที่ผิดหรือไม่ ถ้าผิด แก้ไขให้ถูกต้อง

- ก. int old = 60; new = 20;
- ข. int a = b = 5;
- ค. double float c;

โปรแกรมต่อไปนี้เป็นคอมไพล์ผ่านหรือไม่

```
main() {  
    c = 35;  
    cout << c << endl;  
}
```

## การรับข้อมูลทางแป้นพิมพ์ (Keyboard)

เมื่อต้องการกำหนดค่าตัวแปรใด โดยวิธีป้อนข้อมูลผ่านแป้นพิมพ์ ภาษา C++ มีคำสั่ง cin (console input -อ่านว่า ซี-อิน) รับข้อมูลจากแป้นพิมพ์ นำไปเก็บไว้ในตัวแปรที่ต้องการ รูปแบบคำสั่งทั่วไปมีดังนี้

```
cin >> ชื่อตัวแปร1 >> ชื่อตัวแปร 2 >> .....
```

เครื่องหมาย >> เรียกว่า input operator ระวังอย่าสับสนกับ << ของ cout

จากโปรแกรม 1.1 ถ้าต้องการรับค่าความกว้างและความยาวของสี่เหลี่ยมผืนผ้าทางแป้นพิมพ์ ใช้คำสั่ง cin โปรแกรมจะเปลี่ยนไปดังนี้

```
1:  /* =====
2:  Program 1.9: FindArea2.cpp
3:      Find rectangular area at command prompt
4:  Author: Wachara R.
5:  Date: 03 Oct 2003
6:  Last Update: -
7:  ----- */
8:  #include <iostream.h>
9:  #include <math.h>
10: int main() {
11:     float width, length, area;
12:
13:     cout << "Input width and length : ";
14:     cin >> width >> length;
15:     //Compute area of rectangle.
16:     area = width*length;
17:     //Print area on standard output
18:     cout << " Area of this rectangle = "
19:          << area << " sqr.meter" << endl;
20:
21:     // exit program
22:     return 0;
23: }
24:  // =====
```

เมื่อให้โปรแกรมทำงาน โปรแกรมจะหยุดให้ป้อนค่า ความกว้าง ความยาว ให้เราป้อนตัวเลข 2 จำนวน แยกจากกันโดยใช้เว้นวรรค (เคาะแป้น spacebar)

Input width and length : 22.5 30.2 (กด enter)

เมื่อกด enter จะปรากฏผลลัพธ์บนจอภาพ

Area of rectangle = 679.5 sqr.meter.

## บทที่ 2

### การควบคุมลำดับคำสั่ง (Control Structure)

ภาษา C++ มีการควบคุมลำดับคำสั่งของโปรแกรมเช่นเดียวกับภาษาคอมพิวเตอร์อื่น ๆ มีการควบคุมโดยใช้เงื่อนไข (if, else) การวนรอบ (for, while, do while) หรือเลือกทำคำสั่งจากชุดคำสั่งหลาย ๆ ชุด (switch)

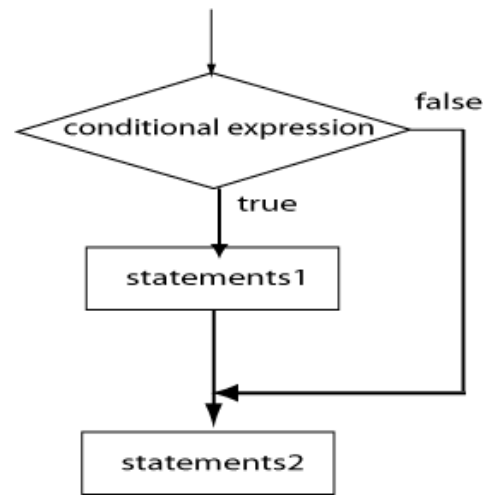
#### การควบคุมชุดคำสั่งโดยใช้เงื่อนไข (if)

ประโยค if ใช้ในการควบคุมการทำงานของโปรแกรม ให้เป็นไปตามเงื่อนไขที่กำหนดไว้

##### รูปแบบการใช้ if แบบที่ 1

```
If (conditional_expression) {  
    ชุดคำสั่งที่ 1;  
}  
ชุดคำสั่งที่ 2 ;
```

ถ้าประโยคเงื่อนไข conditional\_expression เป็นจริง คำสั่งชุดที่ 1 ภายในวงเล็บปีกกาจะทำงาน แล้วจึงออกจากวงเล็บมาทำคำสั่งชุดที่ 2 ถ้า conditional\_expression เป็นเท็จ โปรแกรมจะข้ามมาทำชุดคำสั่งที่ 2

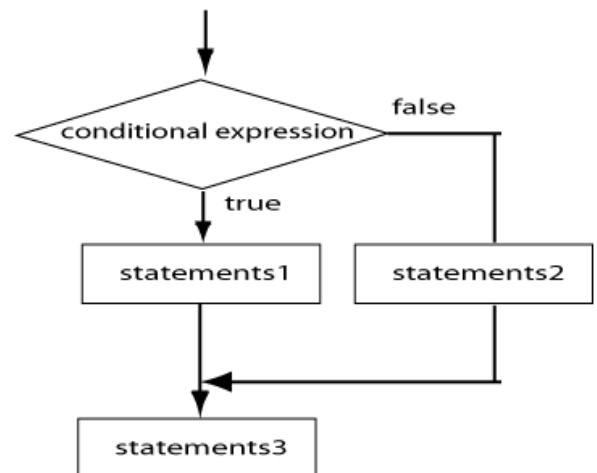


##### รูปแบบการใช้ if แบบที่ 2

```
If (conditional_expression) {  
    ชุดคำสั่งที่ 1;  
} else {  
    ชุดคำสั่งที่ 2 ;  
}  
ชุดคำสั่งที่ 3;
```

ถ้าประโยคเงื่อนไข conditional\_expression เป็นจริง โปรแกรมจะทำ คำสั่งชุดที่ 1 ภายในวงเล็บปีกกาจากนั้นจะกระโดดข้ามมาทำชุดคำสั่งที่ 3 ถ้า

conditional\_expression เป็นเท็จ โปรแกรมจะทำชุดคำสั่งที่ 2 แล้วต่อด้วยชุดคำสั่งที่ 3



##### รูปแบบการใช้ if แบบที่ 3

```
If (conditional_expression1) {  
    ชุดคำสั่งที่ 1;
```

```

} else if{ conditional_expression2) {
    ชุดคำสั่งที่ 2 ;
} else {
    ชุดคำสั่งที่ 3;
}
ชุดคำสั่งที่ 4;

```

ถ้าประโยคเงื่อนไข conditional\_expression1 เป็นจริง ชุดคำสั่งที่ 1 จะทำงานแล้วข้ามมาทำชุดคำสั่งที่ 4 ถ้า conditional\_expression1 เป็นเท็จ โปรแกรมจะตรวจสอบประโยคเงื่อนไขที่ 2 conditional\_expression 2 ถ้าเป็นจริงชุดคำสั่งที่ 2 จะถูกประมวลผล แล้วข้ามไปทำชุดคำสั่งที่ 4 ถ้า conditional\_expression 2 เป็นเท็จ ชุดคำสั่งที่ 3 จะทำงานแล้วต่อดำเนินชุดคำสั่งที่ 4

กรณีที่ if ..else ซ้อนกันหลาย ๆ ชั้น การใส่วงเล็บปีกกาจะช่วยให้ทราบว่า if กับ else ตัวไหนจับคู่กัน เช่น

```

If ( expression1) {
    คำสั่งที่ 1;
    If (expression 2) คำสั่งที่ 2;
} else
    คำสั่งที่ 3;

```

แสดงว่า else จับคู่กับ if ของบรรทัดแรก ถ้าไม่ใส่วงเล็บปีกกา else จะจับคู่กับ if ของบรรทัดที่ 2 โดยทั่วไป else จะจับคู่กับ if ที่อยู่ติดกันเสมอ

การใช้ if ในรูปแบบที่ 3 ซึ่งเป็นการใช้ if ซ้อน if เรียกแบบนี้ว่า nested if statement ในภาษา C++ สามารถใช้ if ซ้อน if ได้ไม่มีขีดจำกัด ในทางปฏิบัติการใช้ if ซ้อนกันหลายชั้น ทำให้เกิดการสับสนในการตรวจสอบการทำงานของโปรแกรม

```

1: //Program 2.1 Calculating the square root of positive number
2: // Program will display some message if you input negative number.
3: #include <iostream.h>
4: #include <math.h>
5: int main() {
6:     int x;
7:     cout << "Finding square root of an integer\n" << endl;
8:     cout << " =====\n" << endl;
9:     cout << "Enter an integer :";
10:    cin >> x;
11:    if ( x < 0)
12:        cout << "Don't input negative number" << endl;
13:    else
14:        cout << "The square root of "<< x << " is " <<sqrt(x) << endl;
15:
16:    cout << "program terminated normally " << endl;
17:    return 0;
18: }
19:

```



เมื่อให้โปรแกรม 2.1 ทำงาน เมื่อป้อนค่า -4 คำสั่งตรวจเงื่อนไข if พบว่า x มีค่าน้อยกว่าศูนย์ เงื่อนไขเป็นจริง คำสั่งในบรรทัดที่ 12 จะแสดงผลบนจอภาพว่า Don't input negative number แล้วเข้ามาทำบรรทัดที่ 16 แสดงผล Program terminated normally และจบการทำงาน

เมื่อให้โปรแกรมทำงานอีกครั้ง ทดลองป้อน +4 คำสั่ง if จะตรวจสอบพบว่า x มีค่ามากกว่าศูนย์ เงื่อนไขเป็นเท็จ โปรแกรมจะทำคำสั่งตามหลัง else (บรรทัดที่ 14) หาค่ารากที่สองของ x แล้วแสดงผลบนจอภาพเป็น 2 ( $\sqrt{4} = \pm 2$ )

ทดลองให้โปรแกรมทำงานอีกครั้ง แล้วป้อนค่า x ที่เป็นเลขทศนิยม เช่น 4.3 แล้วสังเกตผลลัพธ์ที่ได้ หรืออาจป้อนค่าจำนวนเต็ม เช่น 5 รากที่สองที่ได้จะแสดงผลเป็นจำนวนเต็มหรือไม่

ข้อสังเกต ถ้าประโยคคำสั่งที่ตามหลัง if หรือ else มีเพียงคำสั่งเดียว ไม่จำเป็นต้องใส่เครื่องหมายวงเล็บปีกกาปิดล้อมคำสั่ง แต่การใส่เครื่องหมายวงเล็บปีกกาจะทำให้ง่ายต่อการตรวจสอบ

<pre> If (x &lt; 0 ) {     cout &lt;&lt; "don't input negative number \n"; } else {     cout &lt;&lt; " Yes, positive number \n "; } </pre>	<pre> If (x &lt; 0 )     cout &lt;&lt; "don't input negative number \n"; else     cout &lt;&lt; " Yes, positive number \n "; </pre>
---	---

โปรแกรม 2.2 เป็นการหาค่าฟังก์ชัน  $y = \frac{4}{x^2 - 4}$  เมื่อป้อนค่า x ใด ๆ โปรแกรมจะคำนวณหาค่า y พร้อม

แสดงผลที่จอภาพ เมื่อป้อนค่า x = 2 หรือ x = -2 โปรแกรมจะแจ้งเตือนว่าไม่สามารถหาค่า ได้ เพราะค่า x ดังกล่าว ทำให้เกิดกรณีการหารด้วยศูนย์

```

1:  //Program Calculate y = 4/(x^2 -4)
2:  //      Program will stop if x = +2 or x = -2
3:  #include <iostream.h>
4:  int main() {
5:      double x, y;
6:          cout << "Enter a number: " ;
7:          cin >> x;
8:          if ( x == 2 || x == -2)
9:              cout << "Zero Division, stop calculating" << endl;
10:         else {
11:             y = 4/(x*x-4);
12:             cout << "When x = " << x << "y = 4/(x^2 -4) = " << y << endl;
13:         }
14:         cout << "program ended normally " << endl;
15:         return 0;
16:     }

```

ผลการรันโปรแกรมจะเป็นดังรูป

```

C:\MyCpp\chap02>iffunc
Enter a number: 5
When x = 5 y = 4/(x^2 -4) = 0.190476
program ended normally

C:\MyCpp\chap02>iffunc
Enter a number: 2
Zero Division, stop calculating
program ended normally

```

นิพจน์เงื่อนไข (Conditional expression)

นิพจน์เงื่อนไขที่ตามหลัง if (หรือ while หรือ do while หรือ for) ต้องให้ค่าเป็น “จริง” หรือ “เท็จ”  
 ข้อความในนิพจน์อาจจะอยู่ในรูปแสดงความสัมพันธ์กัน เช่น เท่ากับ มากกว่า น้อยกว่า หรืออยู่ในรูปของตรรกะ  
 เครื่องหมายที่ใช้แสดงความสัมพันธ์ (relation operator) เพื่อเปรียบเทียบระหว่างนิพจน์ 2 นิพจน์มี  
 ดังนี้

สัญลักษณ์	ความหมาย	ตัวอย่าง
==	เท่ากับ	if ( x == 5) { ... }
!=	ไม่เท่ากับ	if (x != 5) {...}
<	น้อยกว่า	if ( x < 5) { ... }
>	มากกว่า	If (x > 5) { ... }
<=	น้อยกว่าหรือเท่ากับ	if ( x <=5) { ... }
>=	มากกว่าหรือเท่ากับ	if ( x >=5) { ... }

เครื่องหมายเชิงตรรกะ (logical operator) มีดังนี้

สัญลักษณ์	ความหมาย	ตัวอย่าง
&&	และ (and)	A && B
	หรือ (or)	A    B
!	ค่าความจริงที่ตรงข้าม (not)	!A

A && B , A || B และ !A จะมีค่าเป็นจริงหรือเท็จ ให้ดูจากตารางค่าความจริงดังต่อไปนี้ F หมายถึง false

T หมายถึง true

A	B	A && B
F	F	F
F	T	F
T	F	F
T	T	T

A	B	A    B
F	F	F
F	T	T
T	F	T
T	T	T

A	!A
F	T
T	F

ลำดับความสำคัญ(Precedence)ของเครื่องหมายแสดงความสัมพันธ์และเครื่องหมายตรรกะจะต่ำกว่าเครื่องหมายทางคณิตศาสตร์ แต่จะสูงกว่าเครื่องหมายกำหนดค่า ดังตารางต่อไปนี้

ลำดับความสำคัญ	เครื่องหมาย	ลักษณะการกระทำเริ่มจาก
1	( )	เริ่มจากวงเล็บในสุด
2	+ - ++ -- cast ! (unary operator)	ขวาไปซ้าย
3	* / %	ซ้ายไปขวา
4	+ -	ซ้ายไปขวา
5	< <= > >=	ซ้ายไปขวา
6	== !=	ซ้ายไปขวา
7	&&	ซ้ายไปขวา
8		ซ้ายไปขวา
9	= += -= *= /= %=	ขวาไปซ้าย

### ตัวอย่างประโยคเงื่อนไข

ให้ a = 3, b = 2, c = -1 ประโยคเงื่อนไขของแต่ละข้อต่อไปนี้ จะให้ค่าจริงหรือเท็จ

ก. a < b + c      ประโยคนี้เป็นเท็จ เพราะ 3 < 1 ไม่เป็นจริง

ข. a + c >= b      ประโยคนี้เป็นจริง เพราะ 2 เท่ากับ 2

ค. -c < c + 10      ประโยคนี้เป็นจริง เพราะ -1 < 9

ง. a < 6 && a > 1      ประโยคนี้เป็นจริง เพราะ 3 < 6 และ 3 > 1

จ. a == b - c      ประโยคนี้เป็นจริง เพราะ 3 เท่ากับ 2 - (-1)

ฉ. a == b || b == c      ประโยคนี้เป็นเท็จ เพราะ 3 == 2 หรือ 2 == -1

ข้อผิดพลาดที่พบบ่อยในการใช้ประโยคเงื่อนไข

ผู้เริ่มต้นเขียนโปรแกรมภาษา C++ มักจะตกหลุมพรางต่อไปนี้เสมอ ประโยคเงื่อนไขต่อไปนี้ถูกหลักไวยากรณ์ของ C++ และเมื่อคอมไพล์ก็จะได้ไม่แจ้งข้อความเตือนหรือผิดพลาด แต่ผลลัพธ์ของการทำงานของโปรแกรมไม่ตรงกับจุดประสงค์ที่ต้องการ

1. สับสนเครื่องหมาย = (การกำหนดค่า) กับเครื่องหมาย == (เท่ากับ)

ถ้าต้องการตรวจสอบว่า i มีค่าเท่ากับ 10 หรือไม่ ประโยคที่ถูกคือ

```
If ( i == 10 ) cout << "i equals 10 \n"; // test exactly equality
```

แต่ถ้าเขียนแบบนี้

```
If ( i = 10 ) cout << "i equals 10 \n"; // test of the value assigned to i (zero or nonzero)
```

จะเป็นการตรวจสอบว่ามีการนำค่าคงที่ (ในที่นี้คือ 10) ใส่ไว้ในตัวแปร i หรือไม่ ประโยคนี้จะได้เงื่อนไขเป็นจริงเสมอ เพราะ i ไม่เท่ากับศูนย์

2. ประโยคทางคณิตศาสตร์  $a \leq x \leq b$  หมายถึง x มีค่ามากกว่าหรือเท่ากับ a แต่มีค่าน้อยกว่าหรือเท่ากับ b เมื่อเขียนในโปรแกรมมักจะเขียนเป็น

```
if ( a <= x <= b ) cout << " value of x = " << x;
```

ถ้ากำหนดค่าให้  $a = 2$   $b = 5$  และ  $x = 6$  ประโยคเงื่อนไขนี้จะได้ค่าที่ถูกต้องคือ เท็จ แต่ในการหาค่าจริงเท็จของประโยคนี้นั้นจะเริ่มจากซ้ายไปขวา เริ่มต้นด้วยเปรียบเทียบ  $a$  กับ  $x$  ( $a \leq x$ ) จะได้ค่าเป็นจริงคือ 1 แล้วนำ 1 นี้ไปเปรียบเทียบกับ  $b$  ( $1 \leq 5$ ) ค่าที่ได้จะเป็นจริงคือ 1

3. เปรียบเทียบโดยใช้เครื่องหมาย  $==$  หรือ  $!=$  กับตัวแปรที่มีชนิดข้อมูลเป็น float หรือ double ในทางทฤษฎีเป็นไปได้ที่จำนวนจริง 2 จำนวนมีค่าเท่ากัน แต่การเก็บเลขจำนวนจริงของคอมพิวเตอร์ ความเที่ยงตรง (precision) ของจำนวนเลข จะถูกจำกัดด้วยขนาดของหน่วยความจำที่ใช้เก็บ เลขจำนวนจริงจึงถูกปัดเศษ ทำให้เกิดความคลาดเคลื่อนไปจากค่าจริง ไม่ตรงกับที่เราต้องการ

การเปรียบเทียบควรตรวจสอบกับค่าความแม่นยำที่ยอมรับได้มากกว่า เช่นต้องการให้ค่าที่ได้ผิดพลาดได้ไม่เกิน  $10^{-6}$  ควรเปรียบเทียบดังนี้

```
If (fab(x-y) < 1.0e-6) cout << " x = y";
```

ไม่ควรใช้

```
If ( x == y) cout << " x = y";
```

ลองดูตัวอย่างต่อไปนี้

```
{ float x,y;
  x = 0.123456780;
  y=0.123456785;
  if ( x == y )
      cout << " x equals to y";
  else
      cout << " x not equals to y "
  :
  :
}
```

พบว่า จะได้เงื่อนไขเป็นจริง เนื่องจาก  $x$  และ  $y$  เป็นตัวแปรชนิด float ซึ่งขาดความเที่ยงตรงของทศนิยมตำแหน่งหลัง ๆ แต่ถ้ากำหนดให้  $x$  และ  $y$  เป็นตัวแปรชนิด double พบว่าประโยคเงื่อนไขนี้จะให้ค่าเป็นเท็จ แต่ถ้านำประโยคนี้นี้ไปคอมไพล์บนเครื่องที่กำหนดให้ตัวแปรชนิด float เป็นแบบ 32 บิต เงื่อนไขนี้จะให้ค่าจริงเช่นกัน

4. ประโยคเงื่อนไขจะมีค่าเป็นจริง(true) เมื่อค่าที่ได้ไม่เป็นศูนย์ นั่นหมายถึงจำนวนลบก็จะทำให้ได้ค่าจริงเช่นกัน การสร้างประโยคเงื่อนไขควรให้ค่าที่ได้ออกมาเป็นไปได้เพียงค่าเดียว คือควรตรวจสอบกรณีเป็นศูนย์จะดีกว่า เพราะจะลดความเสี่ยงของค่าต่าง ๆ ที่อาจเกิดขึ้นได้อย่างคาดไม่ถึง ค่าต่าง ๆ ต่อไปนี้ทำให้ประโยคเงื่อนไขให้ค่าเป็นเท็จเสมอ

ชนิดข้อมูล	false value
char	'\0'
short	0
int	0
long	0L
float	0.0
double	0.0

pointer

null or 0

คำสั่งย่อสำหรับเงื่อนไข if ... else

เราสามารถใช้เครื่องหมาย ? : เขียนประโยคคำสั่ง if... else ให้ดูกะทัดรัด โดยมีรูปแบบทั่วไปดังนี้

expression1 ? expression2 : expression3;

มีความหมายเหมือนกับ

```
if ( expression1)
```

```
    expression2;
```

```
else
```

```
    expression3;
```

ตัวอย่าง เปรียบเทียบเลข 2 จำนวน แล้วพิมพ์เลขที่มีค่ามาก

<pre>if ( a &gt; b)     cout &lt;&lt; a; else     cout &lt;&lt; b;</pre>	<pre>(a &gt; b) ? cout &lt;&lt; a : cout &lt;&lt; b;</pre>
--	--

ตัวอย่าง ถ้า x มากกว่าหรือเท่ากับ 0 ให้  $z = x+2$  ถ้าน้อยกว่า 0 ให้  $z = x^2$

<pre>if ( x &gt;= 0)     z = x+2; else     z= x*x;</pre>	<pre>z = (x &gt;= 0) ? x+2 :x*x;</pre>
--	--

ตัวอย่าง ตัวแปรที่ชื่อ bit\_val ต้องการพิมพ์ค่าบิตที่ 0 (บิตสุดท้ายทางขวามือ) ให้เป็นเลขฐานสองคือ 0 หรือ 1 ทำได้ดังนี้

<pre>if (( bit_val &amp; 1)== 0)     cout &lt;&lt; '0'; else     cout &lt;&lt; '1';</pre>	<pre>cout &lt;&lt; (bit_val &amp; 1) ? '0' : '1';</pre> <p>หรือ</p> <pre>cout &lt;&lt; (char)('0' + (bit_val &amp; 1));</pre> <p>หมายเหตุ char เป็น casting operator</p>
---	--

การวนรอบโดยใช้ while และ do... while

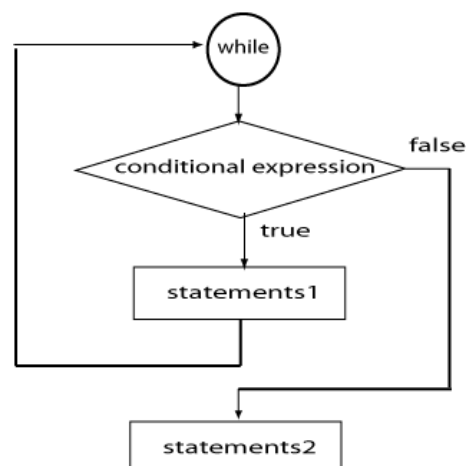
การวนรอบโดยใช้คำสั่ง while หรือ do .. while เป็นการกระทำซ้ำชุดคำสั่งที่อยู่ในประโยค while ตราบเท่าที่ประโยคเงื่อนไขหลัง while มีค่าเป็นจริง (หรือไม่เป็นศูนย์) มีรูปแบบทั่วไปดังนี้

```
while (conditional_expression) {
```

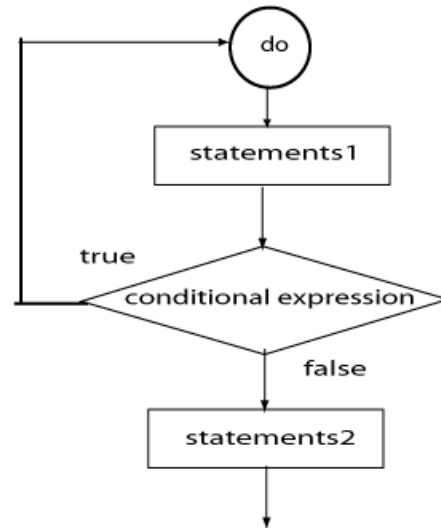
```
    Statements1;
```

```
}
```

```
statements2;
```



```
do {
    statements1;
} while (conditional_expression);
statements2;
```



การทำงานของวงรอบ while จะตรวจสอบประโยคเงื่อนไข (conditional\_expression) ก่อน ถ้าเป็นจริงชุดคำสั่ง statements1 จะทำงาน ถ้าประโยคเงื่อนไขมีค่าเป็นเท็จ จะกระโดดเข้ามาทำ statements2 ทันที นั่นคือถ้าเริ่มต้นการทำงาน ประโยคเงื่อนไขเป็นเท็จ statements1 จะไม่มีการทำงานเลย

ส่วนการทำงานของ do .. while loop statements1 จะถูกประมวลผลในการวนรอบครั้งแรกเสมอ นั่นคือ statements1 ใน do .. while จะทำงานอย่างน้อยหนึ่งครั้ง

ตัวอย่าง ต้องการหาค่าเฉลี่ยของเลขชุดหนึ่ง โดยการป้อนตัวเลขผ่านแป้นพิมพ์ เมื่อป้อนครบเสร็จแล้วให้ป้อน -99 โปรแกรมจะคำนวณหาค่าเฉลี่ยให้ทันที

```
1: // Program: FindAverage.cpp
2: //Loop reading real number from keyboard until finding the end of file (Ctrl+z)
3: // then calculate the average value of those numbers.
4: #include <iostream>
5: using namespace std;
6: int main(void) {
7:     int n=0;
8:     float x, // number of values read
9:         sum =0.0f, //sum of values start with zero
10:         average; // average value of numbers
11:
12:     cout << "Input a series of real numbers :";
13:     while (cin >> x) {
14:         n++;
15:         sum = sum + x;
16:     }
17:     average = sum / n;
18:     cout << "Number of data = " << n << " average = " << average << endl;
19:     return 0;
20: }
```

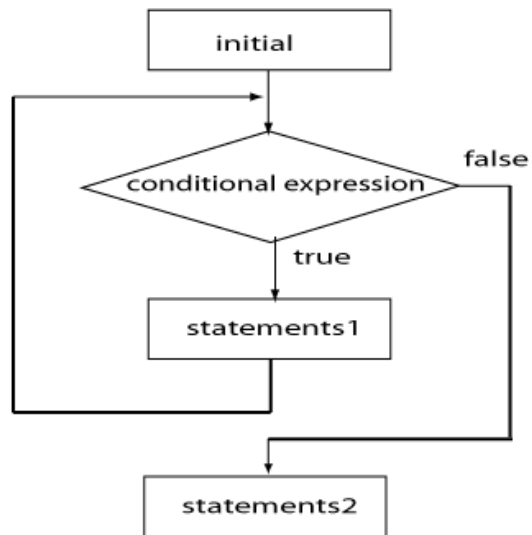
ใน BCC5.5 ไม่สามารถใช้ Ctrl + z ได้ ต้องพิมพ์ /0 เป็นข้อมูลสุดท้าย โปรแกรมจึงจะหาค่าเฉลี่ยให้  
การวนรอบโดยใช้คำสั่ง for

ภาษาคอมพิวเตอร์อื่น ๆ จะใช้คำสั่ง for ในการวนรอบเมื่อรู้จำนวนรอบที่จะวน แต่ for ในภาษา C++ มีความยืดหยุ่นมากกว่านั้น สามารถตรวจสอบประโยคเงื่อนไขได้ เหมือนกับคำสั่ง while ประโยค for ประกอบด้วย 3 ส่วน แต่ละส่วนคั่นด้วยเครื่องหมาย ; ประกอบด้วย ส่วนที่เป็นค่าเริ่มต้น (initial) ส่วนที่ตรวจสอบเงื่อนไข (conditional\_expression) และส่วนที่ใช้ในการเพิ่มค่าหรือลดค่า (adjust)

```

For ( initial; conditional_expression ; adjust) {
    Statements1;
}
statements2;

```



ตัวอย่าง แสดงการหาผลรวมของ  $1 + 2 + 3 + \dots + 50$

```

1:  //Program Sum1to50.cpp
2:  // Finding summation of number from 1 to 50
3:  #include <iostream.h>
4:  int main() {
5:      int sum = 0;
6:      for (int i=1; i<=50; i++) {
7:          sum +=i;
8:      }
9:      cout << " 1 + 2 + 3 +...+ 50 = " << sum << endl;
10: }
11:

```

ตัวอย่าง แสดงการคำนวณแฟกทอเรียลของเลขจำนวนเต็มบวก  $n$  โดยที่

$n! = n(n-1)(n-2)\dots 1$  เมื่อ  $n$  มากกว่าหรือเท่ากับ 0

และ  $0! = 1$

```

1:  // Program Factorial n
2:  #include <iostream.h>
3:  int main() {
4:      //int n,fac;
5:      //long n,fac;
6:      unsigned long n, fac;
7:      //double n, fac;
8:      cout << "Enter a positive number for calculating Factorial : ";
9:      cin >> n;
10:     fac = n;
11:     if ( n != 0) {
12:         for (unsigned long i=n-1; i > 1; i--) {
13:             fac = fac * i;
14:         }
15:     } else if ( n==0)
16:         fac =1;
17:     else cout << "can't calculating Factorial" << endl;
18:     cout << n << "!= " << fac << endl;
19:     return 0;
20: }

```

ทดลองให้โปรแกรมทำงาน แล้วทดสอบดูว่า เมื่อกำหนดให้ตัวแปร n เป็นชนิด int ค่า n มีค่าได้ไม่เกินเท่าใด ค่า n! จะยังคงไม่ผิดพลาด

ใน เทอร์โบ ซีพลัส พลัส พบว่า  $8! = -25216$  ค่าที่ถูกต้องคือ 40320 ค่านี้เกิน 32767 ซึ่งเป็นค่ามากที่สุดที่ตัวแปรชนิด int เก็บได้

ถ้าเปลี่ยนชนิดข้อมูลของตัวแปร n ให้เป็น unsigned long พบว่าหาค่าได้ถูกต้องถึง 32!

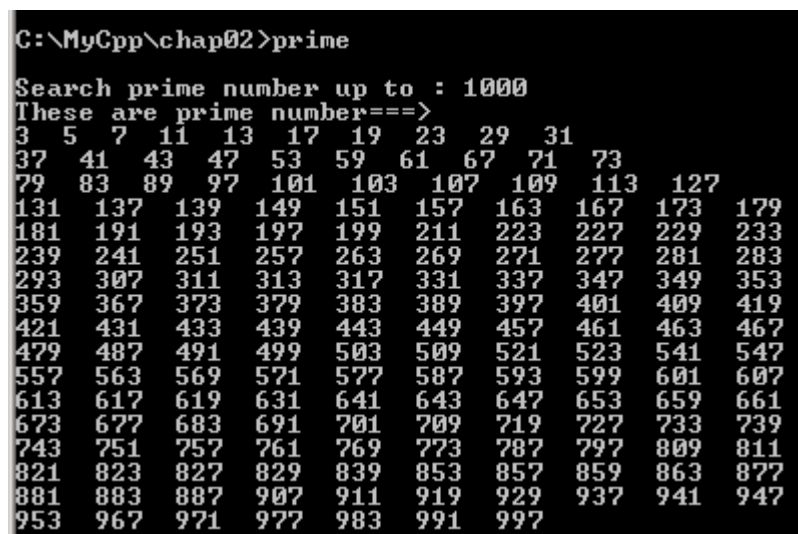
$32! = 2147483648$

ตัวอย่าง การหา จำนวนเฉพาะ (Prime number) จำนวนเฉพาะคือ จำนวนเลขที่มากกว่า 1 และไม่มีเลขอื่นใดหารได้ลงตัวเลย ยกเว้นตัวมันเอง และ 1 จำนวนเฉพาะได้แก่

2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67, ...

```
1: //Program Prime.cpp
2: // Find positive integers for prime number.
3: //A prime number is greater than 1 and is
4: // divisible only by itself and 1
5: #include <iostream.h>
6: void main() {
7:     unsigned long max_num,j;
8:     int line = 0;
9:     cout << "\nSearch prime number up to : ";
10:    cin >> max_num;
11:    cout << "These are prime number==> " << endl;
12:    for (unsigned long i=3; i<=max_num; i+=2) {
13:        //divide number i by add integer j ranging from 3 to i/2
14:        for ( j=3;i%j !=0 && (j < (i/2)); j+=2)
15:            /* leave as null statement intentionally */ ;
16:        if ( j >= (i/2)) {
17:            line += 1;
18:            if (line%10 == 0) cout << i << endl;
19:            else cout << i << " ";
20:        } // if
21:    } //for i loop
22: }
```

ตัวอย่างการหาจำนวนเฉพาะตั้ง 1 – 1000 จะเห็นว่าโปรแกรมนี้นี้ไม่รวม 2 เป็นจำนวนเฉพาะด้วย



```
C:\MyGpp\chap02>prime
Search prime number up to : 1000
These are prime number==>
3 5 7 11 13 17 19 23 29 31
37 41 43 47 53 59 61 67 71 73
79 83 89 97 101 103 107 109 113 127
131 137 139 149 151 157 163 167 173 179
181 191 193 197 199 211 223 227 229 233
239 241 251 257 263 269 271 277 281 283
293 307 311 313 317 331 337 347 349 353
359 367 373 379 383 389 397 401 409 419
421 431 433 439 443 449 457 461 463 467
479 487 491 499 503 509 521 523 541 547
557 563 569 571 577 587 593 599 601 607
613 617 619 631 641 643 647 653 659 661
673 677 683 691 701 709 719 727 733 739
743 751 757 761 769 773 787 797 809 811
821 823 827 829 839 853 857 859 863 877
881 883 887 907 911 919 929 937 941 947
953 967 971 977 983 991 997
```



continue และ break

ในบางครั้งเราอาจต้องการให้การวนรอบที่เกิดจากการใช้คำสั่ง for หรือ while หรือ do .. while มีการเปลี่ยนแปลง เช่น ลื่นสุดลงก่อนที่จะครบตามเงื่อนไขที่กำหนด หรืองดเว้นการทำคำสั่งบางคำสั่งในระหว่างการวนรอบ

เมื่อการวนรอบของ for หรือ while หรือ do .. while พบกับคำสั่ง continue จะไม่มีการทำคำสั่งที่อยู่หลัง continue จะกระโดดข้ามไปวนรอบค่าต่อไปทันที ตัวอย่างเช่น การบวกเลขจำนวนคู่ตั้งแต่ 2 ถึง 10

```
1: #include <iostream>
2: // using namespace std;
3:
4: int main( )
5: {
6:     int sum, i ;
7:     for( sum=0, i=0; i <= 10; i++) {
8:         if (i & 1) continue;
9:         sum +=i;
10:
11:     }
12:     cout << "2 + 4 + 6 + 8 + 10 = " << sum << endl;
13:     return 0;
14: }
```

คำสั่ง break ใช้เพื่อเป็นการออกจากการวนรอบกลางคัน โปรแกรมจะประมวลผลคำสั่งถัดไปที่ต่อจากการวนรอบ

การวนรอบแบบไม่รู้จบ (infinite loop)

บางครั้งเราต้องการบังคับโปรแกรมให้ทำงานวนรอบแบบไม่สิ้นสุด ตัวอย่างเช่น ในระบบควบคุม โปรแกรมอาจจะวนรอบเพื่อคอยรับค่าที่ได้จากหัววัด หรือระบบตรวจจับอื่น ๆ อย่างต่อเนื่องตลอดเวลาแล้วนำค่าเหล่านี้ไปประมวลผล ซึ่งโปรแกรมไม่สามารถจะหยุดทำงานได้ด้วยตัวมันเอง การหยุดการทำงานต้องอาศัยระบบปฏิบัติการที่เรียกว่า killing process ในระบบยูนิกซ์ ทำได้โดยกดปุ่ม Ctrl + c หรือ Ctrl + Q หรือใช้วิธี reset เครื่องคอมพิวเตอร์ เราสามารถสร้างการวนรอบแบบอนันต์ได้ดังนี้

```
while (1) {
    statements;
    :
    :
}
```

สำหรับ for loop

```
for ( ; ; ) {
    statements;
    :
    :
}
```

สำหรับ do ... while loop

```
do {  
    statements;  
    :  
    :  
} while (1);
```

การควบคุมคำสั่ง โดยใช้ switch

การตรวจสอบเงื่อนไขโดยใช้ if ... else ซ้อนกันหลาย ๆ ชั้น อาจเลื่อยมาใช้ switch จะทำให้การตรวจสอบดูง่าย โอกาสที่จะเกิดความผิดพลาดมีน้อยกว่า ตัวอย่างเช่น

<pre>if ( n == 1)     statement1; else if ( n==2)     statement2; else if ( n==3)     statement3; else if ( n==4)     statement4;     : else default_statements;</pre>	<pre>switch ( n ) {     case 1 : { statement1 ;               break; }     case 2 : { statement2;               break; }     case 3 : { statement3;               break; }     case 4 : { statement4;               break; }     :     default : default_statements; }</pre>
--	--

บางครั้งเป็นการบังคับให้โปรแกรมทำงานตามอักขระที่เรากดแป้นพิมพ์ เช่น

<pre>cin &lt;&lt; ch; if ( ch == 'r')     readdata(); else if ( ch=='s')     savedata(); else if ( ch=='c')     calculatedata(); else if ( ch=='p')     printdata();     :</pre>	<pre>cin &lt;&lt; ch; switch ( ch) {     case 'r' : { readdata() ;      break; }     case 's' : { savedata();      break; }     case 'c' : { calculatedata();  break; }     case 'p' : { printdata();      break; }     : }</pre>
--	---

## รูปแบบของคำสั่ง switch

```
switch (expression) {  
    case constant1 : { statements1;  
                        break; }  
    case constant2 : { statements2;  
                        break; }  
    case constant3 : { statements3;  
                        break; }  
    :  
    :  
    default : { default_statements;  
               }  
}  
next_statements;
```

ตัวแปร expression ในวงเล็บของ switch จะต้องเป็นข้อมูลชนิด int หรือ char การใช้ข้อมูลแบบ string ดังต่อไปนี้ ไม่ถูกต้อง คอมไพเลอร์จะแสดงข้อความผิดพลาดขณะคอมไพล์

```
switch (stringvar) {  
    case "one" : { .... };  
    case "two" : { ... };  
}
```

คำสั่ง switch จะทำการตรวจสอบตัวแปร expression ว่าจะไปทำคำสั่ง case ที่ลำดับใด ถ้าค่าใน expression เท่ากับ constant2 โปรแกรมจะกระโดดมาทำคำสั่ง statement2 ที่อยู่ในบล็อกของ case constant2 เมื่อถึงคำสั่ง break โปรแกรมจะเข้ามาทำคำสั่ง next\_statements

ถ้าค่าใน expression ไม่ตรงกับค่าคงที่ใด ๆ ที่ตามหลัง case เลย โปรแกรมจะทำคำสั่งที่อยู่ในบล็อกของ default แล้วต่อด้วย next\_statements

### ข้อสังเกต

1. คำสั่ง case จะสลับลำดับก่อนหลังอย่างไรก็ได้ เช่นเดียวกับ default จะนำไปเขียนตรงส่วนใดก็ได้ ในระหว่างคำสั่ง switch { ... }
2. คำสั่ง break จะเป็นการกำหนดให้โปรแกรมทำเฉพาะชุดคำสั่งที่อยู่ใน case นั้น ถ้าตัดคำสั่ง break ออก โปรแกรมจะประมวลผลไล่ตามลำดับลงมาเรื่อย ๆ รวมทั้งทำคำสั่งของ default ด้วย
3. case แต่ละค่า สามารถเขียนเรียงกันในบรรทัดเดียวกันได้ เช่น  
case 1 : case 2 : statement ;

ตัวอย่าง ในการจัดกลุ่มการสอบย่อย ซึ่งมีคะแนนเต็ม 10 คะแนน นักศึกษาที่ได้คะแนน 8 – 10 คะแนนจัดอยู่ในกลุ่มเก่ง (Good) นักศึกษาที่ได้คะแนน 5-7 คะแนน จัดอยู่ในกลุ่มพอใช้ (Not bad) นักศึกษาที่ได้คะแนน 0 -4 คะแนน ต้องแก้ไขปรับปรุง (More practice)

```

1:  #include <iostream>
2:  using namespace std;
3:
4:  int main( ) {
5:      int score;
6:      cout << "Input your score (from 0 to 10) : ";
7:      cin >> score;
8:      switch (score)      {
9:          case 10: case 9: case 8: { cout << "Good " << endl; break; }
10:         case 7 :case 6: case 5: { cout << "Not bad " << endl ; break; }
11:         case 4 : case 3 : case 2 : case 1 : case 0 : {
12:             cout << " You must practice more ! " << endl; break;
13:         }
14:         default : { cout << "Please input an integer from 0 to 10 \n"; }
15:     }
16:
17:     return 0;
18: }

```

มาถึงตอนนี้ เราได้เรียนรู้ การใช้ชนิดข้อมูลแบบต่าง ๆ ให้เหมาะสมกับงาน การใช้ตัวปฏิบัติการต่าง ๆ การควบคุมคำสั่งให้โปรแกรมวนรอบตามที่เรต้องการ ต่อไปนี้จะเป็นการนำความรู้เหล่านี้มาเขียนเป็นโปรแกรมสั้น ๆ เพื่อคำนวณและแก้ปัญหาทางคณิตศาสตร์ หรือทางวิทยาศาสตร์

**ปัญหา** (ดัดแปลงจาก Schaum's outline หน้า 79) กำหนดตัวตั้ง (numerator) และตัวหาร (denominator) ทั้งคู่เป็นเลขจำนวนเต็มบวก โปรแกรมจะแสดงผลหารซึ่งเป็นจำนวนเต็ม (integer quotient) และเศษเหลือ (remainder)

- ก. ใช้เครื่องหมาย / และ %
- ข. ห้ามใช้เครื่องหมาย / และ %

ก. ใช้เครื่องหมาย / (quotient operator) และ % (remainder operator)

```

1:  // Program division.cpp -- Finding quotient and remainder
2:  // by using / and % operator
3:  // 01 Jan 2004
4:  #include <iostream>
5:  using namespace std;
6:  int main() {
7:      int n ; // numerator
8:      int d; // denominator
9:      cout << "Input Numerator = "; cin >> n;
10:     cout << "Input Denominator = ";cin >> d;
11:     if ( n <=0 ||d <= 0) {
12:         cout << "Your number should greater than zero \n";
13:         return 1;
14:     }
15:     cout << "Quotient = " << n/d;
16:     cout << "\tRemainder= " << n%d;

```

```

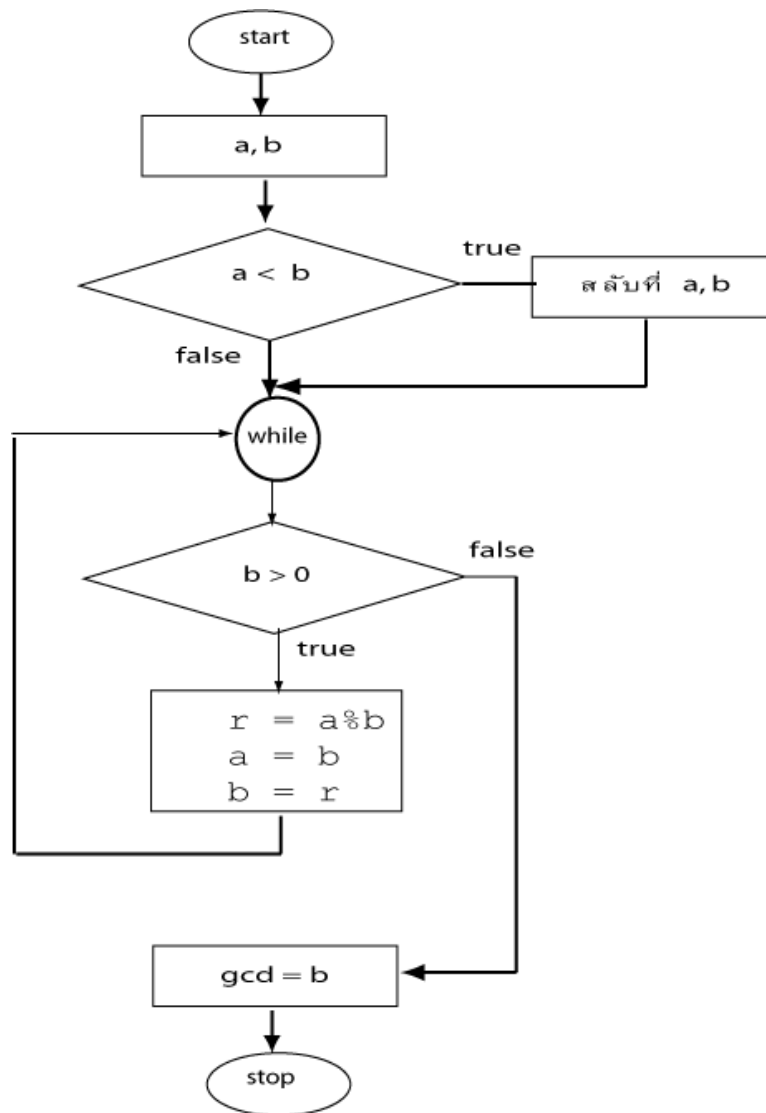
17:         return 0;
18:     }
    ข. ห้ามใช้เครื่องหมาย / และ % แต่จะการใช้การวนรอบแก้ปัญหแทน
1:     // Program divis2.cpp -- Finding quotient and remainder
2:     // by using for loop
3:     // 01 Jan 2004
4:     #include <iostream>
5:     using namespace std;
6:
7:     main() {
8:         int n ; // numerator
9:         int d; // denominator
10:        int i; // number of iteration
11:        int r; // remainder
12:        cout << "Input Numerator = "; cin >> n;
13:        cout << "Input Denominator = ";cin >> d;
14:        if ( n <=0 || d <= 0) {
15:            cout << "Your number should greater than zero \n";
16:            return 1;
17:        }
18:        r = n;
19:        for ( i = 0; r > d; i++)
20:            r = r - d;
21:
22:        cout << "Quotient = " << n/d << endl;
23:        cout << "Remainder= " << n%d << endl;
24:        return 0;
25:    }

```

ปัญหา (จากหนังสือ Schaum's outline หน้า 80) การหาตัวหารร่วมมาก (ห.ร.ม.)(GCD, The Greatest

Common Divisor) โดยใช้ Euclidean algorithm

อัลกอริทึมของ ยูคลิด สามารถเขียนเป็น flow chart ได้ดังนี้



ตัวอย่างเช่น จงหา ห.ร.ม. ของ 52, 128

1. สลับที่ : 128, 52
2. นำ  $128 \% 52$  จะเหลือเศษ 24 จะได้ตัวเลขเก็บไว้เป็น 52, 24
3. นำ  $52 \% 24$  จะเหลือเศษ 4 จะได้ตัวเลขเป็น 24, 4
4. นำ  $4 \% 2$  จะเหลือเศษ 0 จะได้ตัวเลขเป็น 4, 0
5. ห.ร.ม. ของ 52, 128 คือ 4

ตัวอย่าง จงหา ห.ร.ม. ของ 532, 112

$(532, 112) \rightarrow (112, 84) \rightarrow (84, 28) \rightarrow (28, 0)$  ห.ร.ม. ของ 532, 112 คือ 28

ตัวอย่าง จงหา ห.ร.ม. ของ 540, 432

$(540, 432) \rightarrow (432, 108) \rightarrow (108, 0)$  ห.ร.ม. ของ 540, 432 คือ 108

```

1: //Program gcd.cpp -- Calculate the greatest common divisor (g.c.d.)
2: // Jan 01, 2004
3: #include <iostream>
4: using namespace std;
5:
6: main() {
7:     long a,b;
8:     long remainder;
9:     long temp; // temporal variable
10:    cout << "Input two positive integer ";
11:    cin >> a >> b;
12:    if (a < b) {
13:        temp = a;
14:        a = b;
15:        b = temp;
16:    }
17:    cout << "The great common divisor of " << a << " and " << b << " = ";
18:    while ( b > 0) {
19:        remainder = a % b;
20:        a = b;
21:        b = remainder;
22:    }
23:    cout << a << endl;
24:    return 0;
25:
26: }

```

ปัญหา ( Schaum's outline หน้า 80) จงเรียงตัวเลขจำนวนเต็มให้ย้อนกลับทิศทางเดิม เช่น 1234 จะแสดงผล  
4321

```

1: //Program Reverse.cpp -- Reverse your input number
2: // Jan 01, 2004
3: #include <iostream>
4: using namespace std;
5:
6: main() {
7:     long Number;
8:     long digit;
9:     long NewNumber=0;
10:    cout << "What is your number ? ";
11:    cin >> Number;
12:    if (Number <= 0) {
13:        cout << "Please input positive integer only ";
14:        return 1;
15:    }
16:    while (Number > 0) {
17:        digit = Number % 10; //find the right-most digit
18:        Number = Number / 10; // remove the right-most digit
19:        NewNumber = 10*NewNumber + digit; // reverse them
20:    }
21:    cout << " The reverse Number is " << NewNumber <<endl;
22:    return 0;
23:
24: }

```

ตัวอย่างการทำงาน

```
C:\MyCpp\chap02>reverse
What is your number ? 100564
The reverse Number is 465001

C:\MyCpp\chap02>reverse
What is your number ? 123456789
The reverse Number is 987654321

C:\MyCpp\chap02>reverse
What is your number ? 7500
The reverse Number is 57

C:\MyCpp\chap02>reverse
What is your number ? 007
The reverse Number is 7
```

ข้อสังเกต 007 จะได้เป็น 7

7500 จะได้เป็น 57

การสร้างตัวเลขสุ่ม

การใช้โปรแกรมคอมพิวเตอร์จำลองสถานการณ์ (simulation) เช่น การทอดลูกเต๋า จับสลากออกรางวัล โดยให้คอมพิวเตอร์สร้างข้อมูลโดยสุ่มตัวเลขขึ้นมาเอง ตัวเลขที่คอมพิวเตอร์สุ่มขึ้นมา นั้น ต้องไม่จำเพาะเจาะจง หรือคาดเดาตัวเลขที่จะสุ่มออกมาได้โดยวิธีใดวิธีหนึ่ง หรือมีรูปแบบที่แน่นอนตายตัว

ในภาษา C++ มีฟังก์ชัน rand() ใช้สร้างตัวเลขสุ่มที่มีค่าเป็นบวก ตั้งแต่ 0 ถึง RAND\_MAX การใช้ฟังก์ชันนี้จะต้องใส่ header file ชื่อ stdlib.h ไว้ที่ต้นโปรแกรมด้วย

ทดลองสุ่มตัวเลขมา 10 ค่าโดยเขียนโปรแกรมหดังนี้

```
1: // Program random1.cpp --Generate Random number
2: // Jan 01, 2004
3: #include <iostream>
4: #include <stdlib>
5: using namespace std;
6:
7: main() {
8:     for (int i = 1 ; i <= 10 ; i++)
9:         cout << rand() << endl;
10:    cout << "Maximum random number = " << RAND_MAX;
11:    return 0;
12: }
13:
```

ให้โปรแกรมทำงาน จะได้เลขสุ่ม ออกมา 10 ค่า โดยเขียนโปรแกรม ดังนี้

```
346
130
10982
1090
11656
7117
17595
6415
22948
31126
Maximum random number = 32767
```



จะเห็นเลขสุ่มที่ได้จะมีค่าไม่ซ้ำกันเลย และไม่สามารถเดาเลขสุ่มค่าถัดไปด้วยว่ามีค่าเท่าใด เพราะถูกสุ่มออกมาโดยไม่รูปแบบที่แน่นอน เมื่อให้โปรแกรมนี้ทำงานกับเครื่องคอมพิวเตอร์ต่างเครื่องกัน พบว่าอนุกรมเลขสุ่มอาจจะไม่เหมือน ดังที่เขียนไว้ในตัวอย่างนี้

เมื่อทดสอบให้โปรแกรมทำงานครั้งที่ 2 จะได้ผลลัพธ์ดังนี้

```
346
130
10982
1090
11656
7117
17595
6415
22948
31126
Maximum random number = 32767
```

และครั้งที่ 3

```
346
130
10982
1090
11656
7117
17595
6415
22948
31126
Maximum random number = 32767
```

โปรแกรมทำงานแต่ละครั้งจะสร้างเลขสุ่มตั้งแต่ 0 ถึง RAND\_MAX ( ในที่นี้จะได้ค่า 32767) พบว่าเลขสุ่มที่ลำดับเดียวกันของทั้งสามครั้งจะเป็นเลขตัวเดียวกันหรือการทำงานแต่ละครั้งจะให้เลขสุ่มที่เป็นอนุกรมเดียวกันที่เป็นเช่นนี้ เพราะการทำงานของโปรแกรมแต่ละครั้งจะสร้างอนุกรมของตัวเลขจากจุดเริ่มต้น (seed) เดียวกัน ซึ่งจะกำหนดค่าโดยคอมพิวเตอร์โดยปริยาย และเป็นค่าเดียวกันทุกครั้งที่โปรแกรมนี้ทำงาน

คอมพิวเตอร์ไม่สามารถสร้างเลขสุ่มที่แท้จริง เพราะเมื่อให้ค่าอินพุตค่าเดียวกัน คอมพิวเตอร์จะให้ผลลัพธ์ออกมาเหมือนกันเสมอ แต่จำนวนที่ได้ทั้ง 10 ค่าจัดได้ว่าเป็นเลขสุ่ม เราเรียกเลขสุ่มที่ได้จากคอมพิวเตอร์นี้ว่าเป็นเลขสุ่มเทียม (pseudo-random number)

ถ้ากำหนดจุดเริ่มต้น (seed) ที่ต่างกันไปได้จะได้อนุกรมของเลขสุ่มที่ต่างกัน โดยใช้ฟังก์ชัน srand() กำหนดจุดเริ่มต้นของเราเอง ไม่ต้องให้คอมพิวเตอร์กำหนดให้ จะได้ชุดเลขสุ่มที่ต่างกันไป ดังตัวอย่างต่อไปนี้

```
1: // Program - random2.cpp --Generate Random number
2: // Jan 01, 2004
3: #include <iostream>
4: #include <stdlib>
5: using namespace std;
```

```

6:
7:  main() {
8:      unsigned seed;
9:          cout << " Input seed for random number : ";
10:         cin >> seed;
11:         srand(seed);
12:         cout << "Seed = " << seed << endl;
13:         for (int i = 1 ; i <= 10 ; i++)
14:             cout << rand() << " ";
15:         // cout << "Maximum random number = " << RAND_MAX;
16:         return 0;
17:     }

```

เมื่อกำหนดค่า seed = 1, seed = 123, seed = 500 และ seed = 346 จะได้อนุกรมเลขสุ่มที่มีค่าต่างกันดังนี้

Input seed for random number : Seed = 1

346 130 10982 1090 11656 7117 17595 6415 22948 31126

Input seed for random number : Seed = 123

9827 6588 24191 6712 22732 10409 17951 18683 8409 12981

Input seed for random number : Seed = 500

9312 23588 19347 13071 8345 3124 3206 30808 15588 15732

Input seed for random number : Seed = 346

21517 24031 19326 29074 19611 14009 28003 15860 25345 1571

เมื่อพิจารณาอนุกรมเลขสุ่ม ที่ seed = 1 พบว่าเป็นอนุกรมเดียวกันกับตัวเลขสุ่มที่ได้จากโปรแกรม

random1.cpp แสดงว่าโปรแกรมได้กำหนดค่า seed = 1 เป็นค่าโดยปริยาย

เพื่อที่จะไม่ต้องกำหนดค่า seed ในแต่ละครั้งที่มีการสร้างเลขสุ่ม จะใช้ประโยชน์จากสัญญาณนาฬิกาของระบบคอมพิวเตอร์เป็นผู้สร้างค่า seed ให้ ฟังก์ชัน time ( ) จะนำค่าเวลาของเครื่อง ณ ขณะนั้นส่งกลับเป็นเลขจำนวนเต็มบวก การใช้ฟังก์ชัน time ( ) ต้องกำหนด header file ชื่อ time.h ไว้ที่ส่วนหัวของโปรแกรม

```

1:  // Program random3.cpp -- Generate Random number
2:  // Jan 01, 2004
3:  #include <iostream>
4:  #include <stdlib>
5:  #include <time>
6:  using namespace std;
7:
8:  main() {
9:      unsigned seed;
10:         seed = time(NULL);
11:         cout << "Seed = " << seed << endl;
12:         srand(seed);
13:         for (int i = 1 ; i <= 10 ; i++)
14:             cout << rand() << " ";
15:         // cout << "Maximum random number = " << RAND_MAX;
16:         return 0;
17:     }

```

เมื่อให้โปรแกรมทำงานครั้งแรกจะได้ seed = 38620 จะได้เลขสุ่มออกมา 1 ชุด อีก 11 วินาทีต่อมา รันโปรแกรมอีกครั้งจะได้อนุกรมเลขสุ่มอีกชุดหนึ่ง ดังนี้

Seed = 38620

4974 27386 29523 668 25353 22414 13030 21087 9781 22631

Seed = 38631

8783 25014 17821 22931 10505 5253 29983 29177 27809 14280

ปัญหา การทายตัวเลข : คอมพิวเตอร์จะสุ่มตัวเลขจำนวนหนึ่งมีค่าระหว่าง 1 ถึง 100 จากนั้นให้ผู้เล่นทายว่าเลขนี้คือค่าเท่าใด เมื่อทายผิดในแต่ละครั้ง คอมพิวเตอร์จะบอกให้ว่าเลขที่ทายนั้นมากไปหรือน้อยไป โดยให้จำนวนครั้งที่ทายน้อยที่สุดเท่าที่จะเป็นไปได้

```
1: // Program guessnum.cpp --Guessing a Number
2: // Jan 01, 2004
3: #include <iostream>
4: #include <stdlib>
5: #include <time>
6: using namespace std;
7:
8: main() {
9:     unsigned seed,number,guess;
10:    char ch = 'y';
11:    int min =1, max = 100,n;
12:    int range;
13:        while (ch == 'y' || ch == 'Y') {
14:            seed = time(NULL);
15:            srand(seed);
16:            range = max - min + 1;
17:            number = rand() % range + min;
18:            cout << "\nI picked a number(1 to 100). Guess it? ";
19:            cin >> guess;
20:            n=1;
21:            while ( guess != number) {
22:                cout << n << " Your select is " << guess << endl;
23:                if (guess > number) cout << "Too large..\n";
24:                if (guess < number) cout << "Too small ..\n";
25:                cout << "Try again ---> "; cin >> guess;
26:                n++;
27:            }
28:            cout << "At last you are right in " << n << " time. My number is "
29:                << number << endl;
30:
31:            cout << "Would you like to play again ? ";
32:            cin >> ch;
33:        } //while
34:        return 0;
35:    }
```

จากโปรแกรม guessnum.cpp จะเห็นว่า เราสามารถสร้างเลขสุ่มให้มีค่าอยู่ในช่วงที่ต้องการ เช่น จากโปรแกรมให้มีค่าระหว่าง 1 -100 นั่นคือ max =100 , min = 1 ช่วงของเลขสุ่มคือ range มีค่าเท่ากับ max – min + 1 นำไปหาเลขสุ่มได้จาก

$$\text{number} = \frac{\text{rand}()}{\text{range}} + \text{min}$$

ปัญหา เกมไฮโลว์ มีลูกเต๋ายู่ 3 ลูก เมื่อทอดลูกเต๋าล้วนนับจำนวนแต้มทั้งสามลูกซึ่งมีค่าระหว่าง 3 ถึง 18 ถ้ารวมแล้วน้อยกว่า 11 ถือว่าการทอดครั้งนี้ออก “ต่ำ” ถ้ามากกว่า 11 ถือว่าออก “สูง” ถ้ารวมกันได้ 11 พอดีจะถือว่าออก “ไฮโลว์” (height – low) โปรแกรมนี้จะให้ผู้เล่นทายว่าลูกเต๋าทิ้งสามลูกออกมา ต่ำ สูง หรือ “ไฮโลว์” คอมพิวเตอร์จะสุ่มเลขสุ่ม ตั้งแต่ 1 ถึง 6 ออกมา 3 จำนวนแทนลูกเต๋าลูกแต่ละลูก

```
1: // Hilo.cpp
2: // Jan 01, 2004
3: #include <iostream>
4: #include <stdlib>
5: #include <time>
6: using namespace std;
7:
8: int main() {
9:     unsigned dice1,dice2,dice3;
10:    char ch = 'y';
11:    int min =1, max = 6,range,sum,answer,result;
12:        range = max - min + 1;
13:        while (ch == 'y' || ch == 'Y') {
14:            srand(time(NULL));
15:            dice1 = rand()%range + min;
16:            srand(time(NULL)+5);
17:            dice2 = rand()%range + min;
18:            srand(time(NULL)+10);
19:            dice3 = rand()%range + min;
20:            sum = dice1+ dice2 + dice3;
21:            if (sum < 11 ) result = 1 ;
22:            else if ( sum > 11 ) result = 3 ;
23:            else result = 2;
24:            cout << "I've tossed my dices. Pleas guess.\n"
25:                << " Press 1 if it's low, \n"
26:                << " Press 2 if it's just hi-lo \n"
27:                << " Press 3 if it's high.\n";
28:            cout << "Your select is ==> ";cin >> answer;
29:            if (answer == result)
30:                cout << "Congratulation. You win.....\n";
31:            else cout << "Sorry. You lossed.....\n";
32:            cout << "Dice # 1 = " << dice1 << endl;
33:            cout << "Dice # 2 = " << dice2 << endl;
34:            cout << "Dice # 3 = " << dice3 << endl;
35:            cout << "Sum of 3 dices = " << sum << endl;
36:            cout << "\nPlay again ? ";
37:            cin >> ch;
38:        } //while
39:        return 0;
40:    }
```

ผลของการทำงานของโปรแกรมมีดังนี้

```

C:\MyCpp\chap02>hilo
I've tossed my dices. Pleas guess.
Press 1 if it's low,
Press 2 if it's just hi-lo
Press 3 if it's high.
Your select is ==> 1
Congratulation. You win.....
Dice # 1 = 6
Dice # 2 = 1
Dice # 3 = 3
Sum of 3 dices = 10

Play again ? y
I've tossed my dices. Pleas guess.
Press 1 if it's low,
Press 2 if it's just hi-lo
Press 3 if it's high.
Your select is ==> 3
Sorry. You lossed.....
Dice # 1 = 1
Dice # 2 = 3
Dice # 3 = 5
Sum of 3 dices = 9

Play again ? n
C:\MyCpp\chap02>

```

แบบฝึกหัด (จากหนังสือ teach yourself c++ in 21 days)

1. ต่อไปนี้ให้ค่าจริงหรือเท็จ

- ก. 0
- ข. 1
- ค.  $x = 0$
- ง. -1
- จ.  $x == 0$

2. จงคาดคะเนผลลัพธ์จากโปรแกรมต่อไปนี้

```

#include <iostream>

using namespace std;

int main () {
    int x=1, y =1, z;
    if ( z = (x-y))
        cout << "z = " << z;
    return 0;
}

```

3. จงคาดคะเนผลลัพธ์จากโปรแกรมต่อไปนี้

```

#include <iostream>

using namespace std;

```

```

int main () {
    int i=1;
    int x;

    while ( i < 100) {

        cout << " Input an integer :";

        cin >> x;

    }

    return 0;
}

```

(เฉลี่ย ต้องมีประโยค i++; มิฉะนั้นจะวนรอบไม่สิ้นสุด)

#### 4. จงเขียนผลลัพธ์ของโปรแกรม

```

#include <iostream>

using namespace std;

int main () {

    short int mynumber;

    mynumber = 65535;

    cout << "Now the number = " << mynumber << endl;

    mynumber ++;

    cout << "Now the number = " << mynumber << endl;

    mynumber ++;

    cout << "Now the number = " << mynumber << endl;

    return 0;

}

```

ผลลัพธ์ จะได้เป็น

Now the number = 65535

Now the number = 0

Now the number = 1

จงเขียนโปรแกรมหาค่า  $e^x$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

ให้ผู้ใช้นำป้อนค่า x จากนั้นแสดงผล  $e^x$

### บทที่ 3

#### Standard I/O และ ฟังก์ชันทางคณิตศาสตร์

ภาษาC++ อำนวยความสะดวกให้ผู้เขียนโปรแกรมสามารถนำข้อมูลแสดงออกทาง output (จอภาพ, เครื่องพิมพ์, ฯ) หรือรับข้อมูลจาก input (แป้นพิมพ์, พอร์ตต่าง ๆ , หรือไฟล์) ได้โดยง่าย โดยบรรจุฟังก์ชัน และ คลาสที่ทำหน้าที่เหล่านี้ไว้ใน standard library (คลังโปรแกรมมาตรฐาน) พร้อมนำไปใช้งานได้ทันที

การแสดงผลออกทางจอภาพ เราได้ใช้คำสั่ง cout (console output) มาแล้วในโปรแกรม 1.1 บทที่ 1

```
cout << "Area of rectangle = " << area << endl;
```

จากประโยคนี้มีการส่งผ่านพารามิเตอร์ (parameter) 3 ค่า ไปยัง cout โดยส่งเรียงไปตามลำดับเป็น กระแสข้อมูล (stream – A stream is any flow of data form a producer (source) to a consumer(sink) ) คั่นด้วยเครื่องหมาย << (insertion parameter) cout สามารถแปลงข้อมูลชนิดต่าง ๆ อย่างอัตโนมัติ ไม่ว่าจะเป็นจำนวนเต็ม หรือเลขทศนิยม จะถูกแปลงให้เป็นตัวอักษรแสดงผลเรียงลำดับบนจอภาพ

ก่อนจะใช้ cout แสดงผลบนจอภาพ หรือ cin (console input) รับข้อมูลจากแป้นพิมพ์ จะต้องนำ header file ชื่อ iostream.h ใส่ไว้ตรงส่วนหัวของโปรแกรมเสมอ โดยใช้คำสั่ง #include ในไฟล์ iostream.h จะมีการกำหนดข้อมูลและเครื่องหมายต่าง ๆ สำหรับ cout และ cin นำไปใช้

การแสดงผลอาจใช้ตัวกรองข้อมูล หรือ manipulator ทำหน้าที่กรองกระแสข้อมูลให้ผ่านไปยัง cout ทำให้รู้ว่าจะส่งข้อมูลแสดงผลในลักษณะใด manipulator จะถูกนิยามไว้ใน iostream.h ได้แก่ hex จะเป็นการกำหนดให้พิมพ์ตัวเลขในรูปแบบสิบหก dec พิมพ์จำนวนเลขเป็นเลขฐานสิบ oct พิมพ์จำนวนเลขเป็นเลขฐานแปด

ตัวอย่างการนำไปใช้มีดังนี้

```
1: // Program 3.1 Dec2HexAndOct.cpp
2: // Converse decimal to Hex and Oct
3: // -----
4: #include <iostream>
5: using namespace std;
6:
7: main() {
8:     int i;
9:     cout << "Input an Integer Number : " ;
10:    while ( cin >> i) {
11:        cout << i << " in Hexadecimal = " << hex << i << endl;
12:        cout << i << " in Octal      = " << oct << i << endl;
13:    }
14:    return 0;
15: }
```

การทำงานของโปรแกรมจะเป็นดังนี้

```
C:\MyCpp\chap03>dec2hexo
Input an Integer Number :15
15 in Hexadecimal = f
f in Octal        = 17
256
400 in Hexadecimal = 100
100 in Octal       = 400
^Z
\0
```

manipulator นอกจากจะนิยามไว้ใน iostream แล้ว ยังนิยามไว้ใน iomanip ด้วย ที่ใช้บ่อย ๆ ในการแสดงผลจำนวนจริง ได้แก่

setw หมายถึง set field width บอกจำนวนตำแหน่งที่จะแสดงผลบนหน้าจอ สามารถจัดรูปแบบการแสดงผลโดยใช้แทน tab (\t)

ตัวอย่าง เช่น        `int i = 625;`

`cout << setw(5) << i;`

จะแสดงผลบนจอโดยพิมพ์ตัวเลขชิดขวา เติมช่องว่างด้านหน้าให้อีก 2 ตำแหน่งให้ครบ 5 ตำแหน่ง

`-- 625`

setprecision เป็นการกำหนดจำนวนตัวเลขหลังจุดทศนิยม โดยปกติจะแสดงตัวเลขหลังจุดทศนิยม 6 ตำแหน่ง

setiosflags จะใช้ในการกำหนดรูปแบบหรือตั้งค่าการแสดงผลและรับข้อมูล ข้อมูลจะคงรูปแบบเช่นนี้ตลอดไปในโปรแกรม จนกว่าจะใช้คำสั่ง resetiosflags ทั้งการแสดงผลและการรับข้อมูลจะมองจาก flag ที่ manipulator กำหนดไว้

flag ที่ใช้ในการแสดงผลมีดังนี้

ios :: right        แสดงผลตัวเลขในลักษณะชิดด้านขวา

ios :: left         แสดงผลตัวเลขในลักษณะชิดด้านซ้าย

ios :: dec         แสดงผลตัวเลขในระบบสิบ เช่นเดียวกับ dec ที่อยู่ในไฟล์ iostream

ios :: oct         แสดงผลตัวเลขในระบบแปด เช่นเดียวกับ oct ที่อยู่ในไฟล์ iostream

ios :: hex         แสดงผลตัวเลขในระบบสิบหก เช่นเดียวกับ hex ที่อยู่ในไฟล์ iostream

ios :: scientific   แสดงผลในรูปแบบ exponential เช่น 3.24 จะเขียนในรูปแบบ 3.2 e1

ios :: fixed        แสดงผลเป็นจำนวนจริงที่มีจำนวนตำแหน่งของเลขทศนิยมคงที่

```
1:  // Program 3.2 realFormat.cpp
2:  // How to print a number in different format
3:  // -----
4:  #include <iostream>
5:  #include <iomanip>
6:  using namespace std;
7:  main() {
8:  const double PI = 3.141592653;
9:  const double AVOGADRO = 6.0221e23;
10: const double ELECTRON_MASS = 9.1095e-19;
11: const double FIFTY = 50.0;
```



```

12:         cout << "Print PI in different precision : \n";
13:         for (int i=1; i <=10; i++ ) {
14:             cout << setprecision(i) << PI << endl;
15:         }
16:
17:         cout << "\n\nPrint PI with set field width = 12 " << endl;
18:         cout << setw(12) << setprecision(4) << PI << endl;
19:
20:         cout << "\n\nPrint PI in scientific notation ";
21:         cout << setiosflags(ios::scientific) << PI << endl;
22:
23:         cout << "\n\nPrint 50.0, display in scientific notation, because of \n
24: setting seiosflag(ios::scientific)";
25:         cout << "\n FIFTY = 50.0 but display in Scientific Notation " << FIFTY <<
26:         endl;
27:
28:         cout << "\n\nThen reset Scientific Notation flag" << resetiosflags(ios::
29:         scientific);
30:         cout << "\n FIFTY = 50.0 and display as " << FIFTY << endl;
31:         return 0;
32:
33:     }

```

ผลการรันโปรแกรม ดังรูปข้างล่างนี้

```

C:\MyCpp\chap03>format
Print PI in different precision :
3
3.1
3.14
3.142
3.1416
3.14159
3.141593
3.1415927
3.14159265
3.141592653
Print PI with set field width = 12
3.142
Print PI in scientific notation 3.1416e+00
Print 50.0, display in scientific notation, because of
setting seiosflag(ios::scientific)
FIFTY = 50.0 but display in Scientific Notation 5.0000e+01
Then reset Scientific Notation flag
FIFTY = 50.0 and display as 50

```

The screenshot shows the output of a C++ program. Annotations with arrows point to specific parts of the output:

- 1**: Points to the list of PI values with increasing precision (3, 3.1, 3.14, 3.142, 3.1416, 3.14159, 3.141593, 3.1415927, 3.14159265, 3.141592653).
- 2**: Points to the output "Print PI with set field width = 12" and the resulting formatted value "3.142".
- 3**: Points to the output "Print PI in scientific notation 3.1416e+00".
- 4**: Points to the output "Print 50.0, display in scientific notation, because of setting seiosflag(ios::scientific) FIFTY = 50.0 but display in Scientific Notation 5.0000e+01".
- 5**: Points to the output "Then reset Scientific Notation flag FIFTY = 50.0 and display as 50".

โปรแกรมจะทำงานในขั้นแรก หมายเลข 1 ในรูป จะแสดงผลค่า PI ( $PI = 3.141592653$ ) ที่มีตัวเลขหลังทศนิยมต่างกัน ตามที่กำหนดไว้ `setprecision(i)` โดยกำหนดให้ `i` มีค่าตั้งแต่ 1 ถึง 10

หมายเลข 2 กำหนดให้การพิมพ์ค่า PI โดยกำหนดความกว้างของ `setw(12)`

หมายเลข 3 เป็นการพิมพ์ค่า **PI** ในรูปเลขยกกำลัง

หมายเลข 4 ได้กำหนด flag ไว้เป็น `ios :: scientific` เมื่อพิมพ์จำนวน 50 พบว่าจะอยู่ในเลขยกกำลัง

หมายเลข 5 เมื่อ reset flag พบว่าสามารถพิมพ์ตัวเลข 50 ในรูปแบบปกติที่เราคุ้นเคย

cin และการตรวจสอบข้อมูลที่ป้อนเข้ามาทางแป้นพิมพ์

ข้อมูลที่ป้อนเข้ามาทางแป้นพิมพ์อาจมีการพิมพ์ผิด เช่น ต้องการป้อนข้อมูลตัวเลข อาจมีตัวอักษรปนเข้ามาด้วย เราสามารถตรวจสอบได้โดยใช้คำสั่ง cin ดังนี้

cin.eof() ส่งค่า 1 กลับคืนถ้าพบจุดสิ้นสุดของไฟล์ (end of file)

cin.good() ส่งค่า 1 กลับคืนถ้าข้อมูลที่รับเข้ามาถูกต้อง

cin.fail() ส่งค่า 1 กลับคืน ถ้าพบว่าข้อมูลที่ป้อนเข้ามามีข้อผิดพลาด

ตัวอย่างเช่น

```
int i;

cin >> i;

if ( cin.good() ) cout << " Data ok, value = " << i << endl;

else

    cout << "Data input error !!! ";

.....
```

กรณีเลือกใช้ cin.fail() ในการตรวจสอบจะเป็นดังนี้

```
if ( cin.fail() ) cout << "Data Error \n";

else cout << " Data ok, value = " << i << endl;

.....
```

วิธีการตรวจสอบ อาจใช้สถานะของ stream ในตัวคำสั่งของ cin จะเรียกฟังก์ชัน cin.fail() จะส่งกลับค่า false ถ้ามีข้อผิดพลาด

```
if ( cin ) cout << " Data OK \n " ;

else cout << "Data Error \n";

อาจเขียนคำสั่งตรวจสอบกะทัดรัด พร้อมคำสั่งป้อนข้อมูลก็ได้ดังนี้

if ( cin >> i ) cout << " Data OK = " << i ;

else cout << "Data error \n ";
```

```
1: // Program 3_3. testcin.cpp
2: //          Read a radius of circle from keyboard
3: //          if ok, calculate area of this circle.
4:
5: #include <iostream>
6: using namespace std;
7: const float PI =3.141592653;
8: main() {
9:     float radius, area;
10:    cout << "Enter radius of circle (real number) : ";
11:    if ( cin >> radius)
12:    {      cout << "radius = " << radius << endl;
13:          area = PI * radius *radius;
14:          cout << "Area of this circle is " << area << endl;
```

```

15:         } else {
16:             cout << "Data input error \n ";
17:         }
18:         return 0;
19:     }

```

โปรดสังเกต การป้อนค่า radius ผ่านแป้นพิมพ์ โดยใช้คำสั่ง cin ดังต่อไปนี้

```

C:\MyCpp\chap03>testcin
Enter radius of circle (real number) : 2
radius = 2
Area of this circle is 12.5664

C:\MyCpp\chap03>testcin
Enter radius of circle (real number) : 2e1
radius = 20
Area of this circle is 1256.64

C:\MyCpp\chap03>testcin
Enter radius of circle (real number) : 2.5
radius = 2.5
Area of this circle is 19.635

C:\MyCpp\chap03>testcin
Enter radius of circle (real number) : -2.5
radius = -2.5
Area of this circle is 19.635

C:\MyCpp\chap03>testcin
Enter radius of circle (real number) : 2e-1
radius = 0.2
Area of this circle is 0.125664

C:\MyCpp\chap03>testcin
Enter radius of circle (real number) : abc
Data input error

C:\MyCpp\chap03>testcin
Enter radius of circle (real number) : 2abc
radius = 2
Area of this circle is 12.5664

```

ฟังก์ชันทางคณิตศาสตร์ (Mathematical function)

ในการแก้ปัญหาทางวิทยาศาสตร์และวิศวกรรม สมการที่ได้อาจอยู่ในรูปเลขยกกำลังหรือลอการิทึม ฟังก์ชันตรีโกณมิติ ฟังก์ชันเอ็กซ์โพเนนเชียล ภาษาC++ ได้สร้างฟังก์ชันทางคณิตศาสตร์เตรียมไว้ให้เรียกใช้งาน ได้ทันที ฟังก์ชันเหล่านี้อยู่ใน standard C++ library การเรียกใช้งานฟังก์ชัน ต้องใส่ประโยค `#include <math>` ไว้ที่ส่วนหัวของโปรแกรม

ฟังก์ชันทางคณิตศาสตร์พื้นฐานที่ใช้บ่อย ๆ ในงานคำนวณ ได้แก่

- ceil (x)	ปัดจำนวนจริง x ให้มีค่าเพิ่มขึ้นเป็นจำนวนเต็มที่มีค่าใกล้ x และใกล้ค่า $\infty$ (infinity) มากที่สุด (smallest integer $\geq x$ ) or round up เช่น ceil (5.03) = 6
- exp(x)	คำนวณค่า $e^x$ เมื่อ e เป็นเลขฐานธรรมชาติ $e = 2.718282\dots$ , exp(e) = 7.38906,,,
- fab(x)	หาค่าสมบูรณ์ของ x เช่น fab(-6) = 6
- floor ( x )	ปัดจำนวนจริง x ให้มีค่าลดลงเป็นจำนวนเต็มที่มีค่าใกล้ x และใกล้ค่า $-\infty$ มากที่สุด (largest integer $\leq x$ ) or round down เช่น floor (5.03) = 5
- log ( x )	คำนวณค่า $\ln x$ จะเกิดข้อความผิดพลาด ถ้า x น้อยกว่า หรือเท่ากับ 0 เช่น log (2) = 0.693147
-log10 (x)	คำนวณค่า log ฐานสิบของ x จะเกิดข้อความผิดพลาด ถ้า x น้อยกว่า หรือเท่ากับ 0 เช่น log10 (2) = 0.30103
- pow (x,p)	คำนวณค่า $x^p$ จะเกิดข้อความผิดพลาด ถ้า x เท่ากับศูนย์ และ P น้อยกว่าศูนย์ หรือ ถ้า x น้อยกว่าศูนย์ และ P ไม่ใช่เลขจำนวนเต็ม เช่น pow(3,2) = 9.0
- sqrt ( x )	คำนวณค่ารากที่สองของ x จะเกิดข้อความผิดพลาด ถ้า x เป็นจำนวนลบ เช่น sqrt(3)=1.732...

ฟังก์ชันทางคณิตศาสตร์จะส่งค่ากลับแบบ double ถ้าค่าที่ส่งผ่านในอาร์กิวเมนต์เป็นจำนวนเต็ม จะถูกเปลี่ยนเป็นจำนวนจริงเสียก่อนที่จะนำไปคำนวณ

การใช้งานฟังก์ชันทางคณิตศาสตร์ ทำได้โดยเขียนชื่อฟังก์ชันตามด้วยวงเล็บ ภายในวงเล็บคือค่าคงที่หรือค่าของตัวแปรที่จะให้ฟังก์ชันนั้นไปใช้ในการคำนวณ อาจจะมีมากกว่า 1 ค่า เช่น pow ( x, y) จะต้องมีความคงที่หรือตัวแปรถึง 2 ตัว หรืออาจไม่มีเป็นวงเล็บว่าง ๆ ก็ได้ ค่าที่อยู่ในวงเล็บเรียกว่า parameter หรือ argument ฟังก์ชันใดที่มีพารามิเตอร์มากกว่า 1 ตัว การใส่ค่าพารามิเตอร์จะต้องเรียงลำดับให้ถูกต้อง และตรงกับชนิดข้อมูลที่กำหนดไว้ด้วย

บางครั้งจะต้องระวังเรื่องหน่วยวัดของค่าพารามิเตอร์ด้วย ฟังก์ชันตรีโกณมิติจะรับค่ามุมที่มีหน่วยเป็นเรเดียนเท่านั้น ถ้าข้อมูลมีหน่วยเป็นองศา จะต้องเปลี่ยนค่ามุมที่เป็นองศาให้เป็นเรเดียนเสียก่อน

ตัวอย่างเช่น ต้องการเก็บค่า  $\sin x$  ไว้ในตัวแปร y โดยที่ x วัดเป็นองศา จะต้องเปลี่ยนค่า x ให้เป็นเรเดียนเสียก่อน โดยใช้ความสัมพันธ์ 180 องศา เท่ากับ  $\pi$  เรเดียน

```
const double PI = 3.141592653;
:
:
double x, y;
    y = sin (x *PI/ 180);
```

ฟังก์ชันสามารถใช้เป็นตัวพารามิเตอร์ของอีกฟังก์ชันหนึ่งก็ได้ บรรทัดต่อไปนี้กำลังหาค่า log ของค่าสมบูรณ์ของ x

```
y          =      log (fabs(x));
```

## ฟังก์ชันตรีโกณมิติ

ฟังก์ชันตรีโกณมิติที่มีอยู่ใน standard C++ library มีดังนี้

sin (x)	คำนวณหาค่า sin x เมื่อ x มีหน่วยเป็นเรเดียน
cos (x)	คำนวณหาค่า cos x เมื่อ x มีหน่วยเป็นเรเดียน
tan (x)	คำนวณหาค่า tan x เมื่อ x มีหน่วยเป็นเรเดียน
asin (x)	คำนวณหาค่า arcsine หรือ inverse sine ของ x x จะมีค่าอยู่ระหว่าง -1 ถึง 1 ฟังก์ชันจะส่งผลลัพธ์กลับเป็นค่ามุมเรเดียน มีค่า $-\pi/2$ ถึง $\pi/2$
acos (x)	คำนวณหาค่า arccosine หรือ inverse cosine ของ x x จะมีค่าอยู่ระหว่าง -1 ถึง 1 ฟังก์ชันจะส่งผลลัพธ์กลับเป็นค่ามุมเรเดียน มีค่า $-\pi/2$ ถึง $\pi/2$
atan(y,x)	คำนวณหาค่า arctan หรือ inverse tan ของ y/ x ฟังก์ชันจะส่งผลลัพธ์กลับเป็นค่ามุมเรเดียน มีค่า $-\pi/2$ ถึง $\pi/2$
atan2(y/x)	คำนวณหาค่า arctan หรือ inverse tan ของ y/ x ฟังก์ชันจะส่งผลลัพธ์กลับเป็นค่ามุมเรเดียน มีค่า $-\pi$ ถึง $\pi$

ฟังก์ชัน atan จะส่งกลับค่ามุมซึ่งอยู่ในควอดแรนต์ที่ 1 และ 3 ในขณะที่ฟังก์ชัน atan2 จะส่งกลับค่ามุมซึ่งอยู่ในควอดแรนต์ใด ๆ ขึ้นอยู่กับเครื่องหมายของ x และ y ในการแก้ปัญหาด้านวิทยาศาสตร์ ควรใช้ฟังก์ชัน atan2

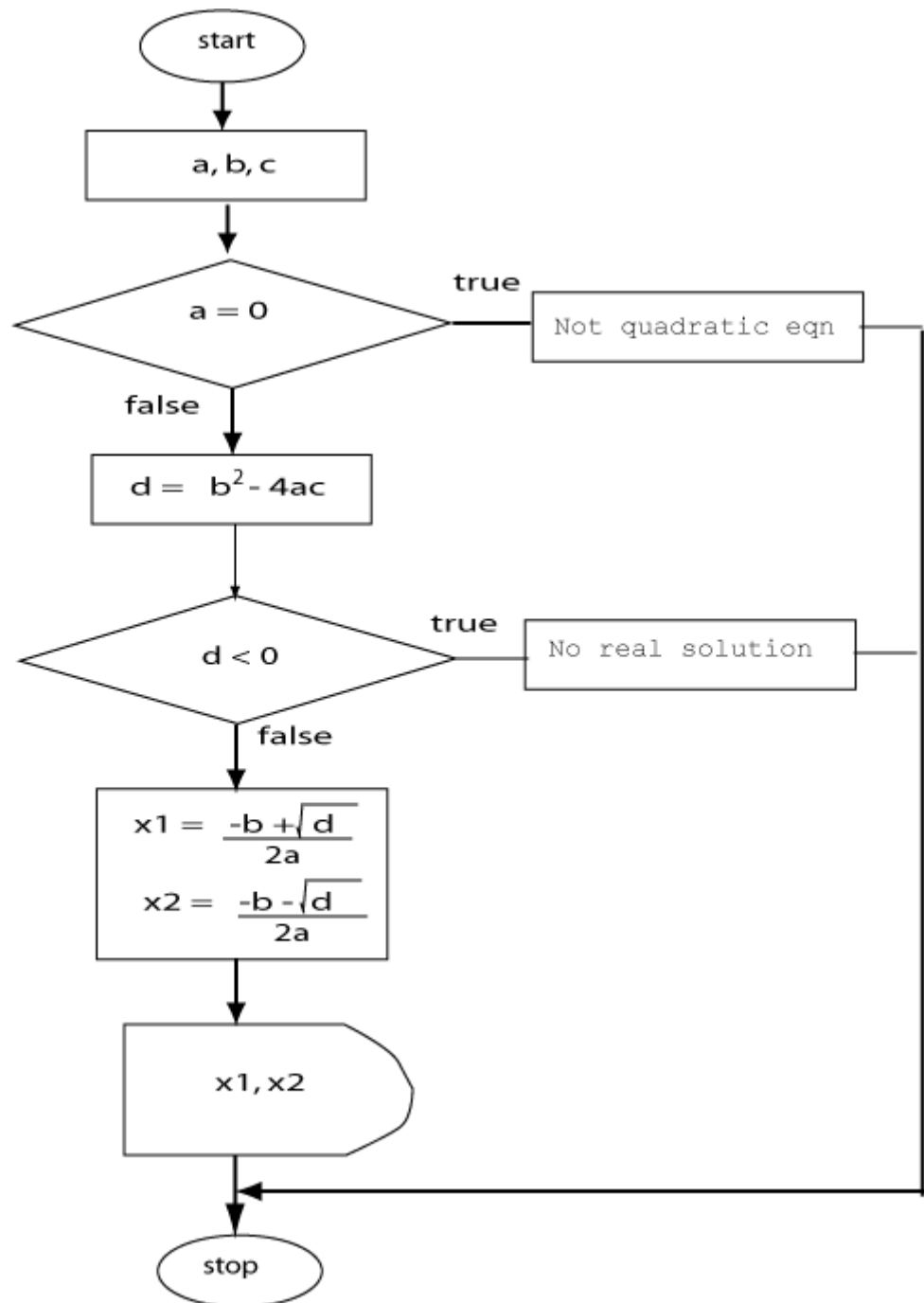
ค่า inverse ของ sine และ cosine จะถูกต้องก็ต่อเมื่อค่าอาร์กิวเมนต์อยู่ในช่วง -1 ถึง 1 เท่านั้น

ตัวอย่าง การใช้ฟังก์ชันคณิตศาสตร์หาคำสมการกำลังสองซึ่งอยู่ในรูป  $ax^2 + bx + c = 0$  เมื่อ a,b และ c เป็นสัมประสิทธิ์ของ x หาคำสมการที่ได้จะมี 2 คำ คือ

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

การทำงานเป็นไปตามโฟลว์ชาร์ต ดังต่อไปนี้



```

1: // Program 3.3 quadratic.cpp
2: // How to print a number in different format
3: // -----
4: #include <iostream>
5: #include <iomanip>
6: using namespace std;
7: main() {
8:     double a, b, c, d, x1, x2;
9:     cout << "Input coefficients of quadratic equation : ";
10:    cin >> a >> b >> c;
11:    if ( a == 0 ) {

```

```

12:             cout << " It's not quadratic equation. \n";
13:             return 0;
14:         }
15:         d = b*b - 4*a*c; // the discriminant
16:         if ( d < 0 ) {
17:             cout << " No real Solution \n";
18:             return 0;
19:         }
20:         x1 = (-b + sqrt(d)) / (2*a);
21:         x2 = (-b - sqrt(d)) / (2*a);
22:
23:         cout << "The quadratic equation is " << a << "x^2 + " << b \
24:             << "x + " << c << " = 0 \n";
25:         cout << "The Solutions are : " << x1 << " and " << x2 << endl;
26:     }
27:

```

ผลการรันโปรแกรม จะได้ดังนี้

```

C:\MyCpp\chap03>quadratic
Input coefficints of quadratic equation : 1 2 1
The quadratic equation is 1x^2 + 2x + 1 = 0
The Solutions are : -1 and -1

C:\MyCpp\chap03>quadratic
Input coefficints of quadratic equation : 1 -7 12
The quadratic equation is 1x^2 + -7x + 12 = 0
The Solutions are : 4 and 3

C:\MyCpp\chap03>quadratic
Input coefficints of quadratic equation : 1 0 -16
The quadratic equation is 1x^2 + 0x + -16 = 0
The Solutions are : 4 and -4

C:\MyCpp\chap03>quadratic
Input coefficints of quadratic equation : 2 6 8
No real Solution

```

## Enumeration Type

นอกจากจะมีชนิดข้อมูลแบบจำนวนเต็ม ได้แก่ int และ char ให้ใช้งานแล้ว C++ ยังยอมให้ผู้เขียนโปรแกรมสร้างชนิดข้อมูลในแบบเฉพาะของตนเอง (user-define type) อีกด้วย ซึ่งทำได้หลายวิธี วิธีที่มีประสิทธิภาพมากที่สุดคือใช้ class (กล่าวถึงต่อไปในบทที่ 9) วิธีที่ง่ายและทำให้โปรแกรมสามารถสื่อความหมายของตัวมันเองได้ชัดเจน คือใช้ชนิดข้อมูลแบบ enumeration type ซึ่งมีรูปแบบดังนี้

```
enum ชื่อชนิดข้อมูล { รายการข้อมูลต่าง ๆ ที่จะเป็นสมาชิกของชนิดข้อมูลนี้ };
```

enum เป็นคำสงวนใน C++ การตั้งชื่อชนิดข้อมูล (typename) ใช้หลักเกณฑ์เดียวกันกับการตั้งชื่อตัวแปร รายการข้อมูล (enumeratorlist) ที่ใส่ในวงเล็บปีกกาจะถูกแทนด้วยจำนวนเต็ม โดยเริ่มต้นที่ศูนย์ ตัวอย่างเช่น

```
enum Grade { F, D, C, B, A };
```

เราสามารถนำชนิดข้อมูล Grade ไปใช้งาน ดังนี้

```

Grade sci_grade, math_grade;
sci_grade = B;
math_grade = A;
if ( sci_grade == math_grade)
    cout << "You got the same grade \n";
else
    cout << " You got different grade\n";

```

ค่าจำนวนเต็มที่ถูกกำหนดไว้ในรายการข้อมูล เรียกว่า enumerator ซึ่งจะคล้ายกับการนิยามค่าคงที่

```

const int F = 0;

const int D = 1;

:

const int A = 4;

```

ค่าที่กำหนดใน enumeratorlist อาจเป็นจำนวนเต็มค่าอื่นที่เราต้องการก็ได้ เช่น

```
enum Base { binary = 2, octal = 8, decimal = 10, hexadecimal = 16 };
```

หรือกำหนดค่าเริ่มต้นเป็นค่าอื่น ที่ไม่ใช่ศูนย์ เช่น

```
enum DayOfWeek { sun=1, mon, tue, wed, thu, fri, sat };
```

sun มีค่าเท่ากับ 1 mon มีค่าเท่ากับ 2 , ..., sat มีค่าเท่ากับ 7

สามารถสร้าง enumerator ที่มีค่าซ้ำกันได้ เช่น

```
enum Answer { false = 0, no=0, ture=1, yes=1, ok=1 };
```

การนำไปใช้ให้ประกาศตัวแปรดังนี้

```
Answer ans;
```

```
:
```

```
if (ans) cout << " Your answer is ok. ";
```

ไม่ว่าค่า ans จะเป็น true หรือ yes หรือ ok ทุกตัวมีค่าเท่ากับ 1 เงื่อนไขนี้จะเป็นจริงเสมอ

ตัวอย่างอื่น ๆ กับการกำหนดข้อมูลชนิด enumeration type

```
enum Sex { female, male};
```

```
enum CardType { clubs, diamonds, hearts, spades };
```

```
enum RainbowColor { violet, blue, green, yellow, orange, red};
```

```
enum RomanNumber { I=1, V=5, X= 10, L=50, C=100, D=500, M=1000};
```

ข้อควรระวัง

1. สมาชิกใน enumeratorlist ไม่สามารถมีเครื่องหมาย + หรือ - ได้เช่น

```
enum Grade { F, D, D+, C, C+, B, B+, A};
```

2. ชื่อสมาชิกของข้อมูลชนิด enumeratorlist ที่มากกว่า 2 ชุดขึ้นจะซ้ำกันไม่ได้ เช่น

```
enum Grade { F, D, C, B, A };
```

```
enum RomanNumber { I=1, V=5, X= 10, L=50, C=100, D=500, M=1000};
```

จากตัวอย่างนี้จะเห็นว่า C และ D จะถูกนิยามสองครั้ง

ตัวอย่างโปรแกรมการสร้างข้อมูลชนิด enum



```

1:  #include <iostream>
2:  using namespace std;
3:  enum number { zero, one, two, three, four, five, six, seven, eight, nine };
4:
5:  int main( ) {
6:
7:      number mynum;
8:      mynum = one;
9:      if (mynum==1) cout << "My number = " << one << endl;
10:     mynum = two;
11:     if (mynum==2) cout << "My number = " << two << endl;
12:     mynum = three;
13:     if (mynum==3) cout << "My number = " << three << endl;
14:
15:     mynum = (number)(two + three);
16:     cout << "Now, My number = " << mynum << endl;
17:     return 0;
18: }

```

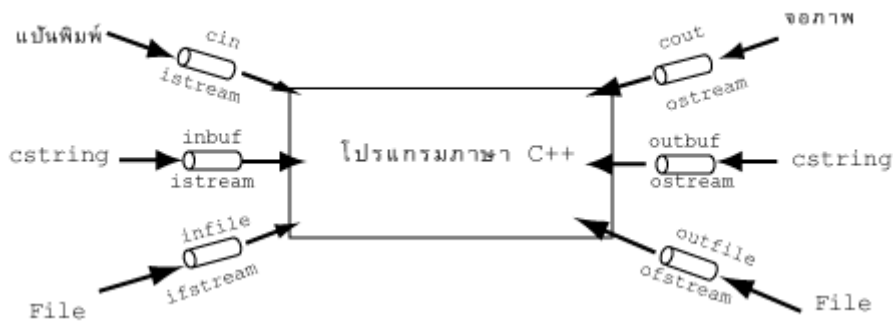
ผลลัพธ์ของ two + three จะมีค่าเหมือน  $2 + 3 = 5$  อย่างไรก็ตาม mynum ไม่ใช่ข้อมูลชนิด int แต่เป็น number จึงต้อง cast ชนิดข้อมูลด้วย number

### แฟ้มข้อมูล

ไฟล์หรือแฟ้มข้อมูลใน C++ จัดเป็น stream คือประกอบด้วยบิต (0 หรือ 1) ถูกจัดเก็บเรียงต่อกันในสื่อบันทึกข้อมูล เช่น แผ่นดิสก์ หรือเทป ระบบปฏิบัติการของคอมพิวเตอร์จะตีความหมายบิตทั้งหลายเหล่านี้ในลักษณะต่าง ๆ กัน เช่น ถ้าบิตถูกจัดกลุ่ม ๆ ละ 8 บิต จะถูกแปลความหมายเป็นรหัสอักขระ (ASCII code) ไฟล์นี้จะจัดเป็น text file สามารถเปิดดูได้โดยอาศัยโปรแกรมประเภท text editor ถ้าบิตเหล่านี้ถูกจัดกลุ่ม ๆ ละ 32 บิตแทนพิกเซลของสี ไฟล์นี้จะเป็ไฟล์ประเภทกราฟิก ต้องใช้โปรแกรมประเภทกราฟิก เช่น paint หรือ photoshop เปิดดูข้อมูล ถ้าไฟล์นั้นเป็นชุดคำสั่งของคอมพิวเตอร์ เช่นมีนามสกุล เป็น exe หรือ com บิตจะถูกแปลความหมายเป็นคำสั่งของคอมพิวเตอร์ หน่วยประมวลผลกลางหรือcpu จะนำไปประมวลผลต่อไป

ถ้าไฟล์นั้นถูกใช้งานสำหรับเขียนข้อมูลหรือแสดงข้อมูลออกทางจอภาพหรือเครื่องพิมพ์ จะเรียกเป็น output file stream ถ้าไฟล์นั้นถูกเปิดขึ้นมาเพื่ออ่านข้อมูลจากไฟล์นำเข้าไปสู่หน่วยความจำของคอมพิวเตอร์จะเรียกว่า input file stream การใช้งานแฟ้มข้อมูลจะต้องอ้างไฟล์ fstream.h ไว้ที่ส่วนต้นของโปรแกรม ในไฟล์ fstream.h นี้จะมีการนิยามคลาส ofstream และ ifstream สำหรับใช้กับ output file stream และ input file stream ตามลำดับ instance ของคลาสเหล่านี้จะใช้เครื่องหมาย << (insertion operator) และ เครื่องหมาย >> (extraction operator) เหมือนกับ stream อื่น ๆ

ภาพต่อไปนี้จะแสดงความสัมพันธ์ระหว่าง stream กับสิ่งที่เกี่ยวข้อง อาจมอง stream เหมือนกับท่อน้ำที่นำข้อมูลเข้าหรือออกจากโปรแกรมไปยังต้นทางหรือปลายทาง (from page 396 Schaum's outline)



ในกรณีข้อมูลที่ใช้ในการคำนวณมีปริมาณมาก การป้อนข้อมูลเข้าทางแป้นพิมพ์ย่อมไม่สะดวก หรือ บางครั้งข้อมูลที่ได้ ได้มาจากเครื่องมือวัด (probe or sensor) ซึ่งเก็บข้อมูลตามระยะเวลาที่กำหนด ข้อมูลที่เก็บ ได้จะถูกบันทึกไว้ในรูปแฟ้มข้อมูล โปรแกรมสามารถดึงข้อมูลจากแฟ้มนั้นมาประมวลผลได้ทันที

การเขียนโปรแกรมที่ใช้งานแฟ้มข้อมูลจะต้องกำหนดไฟล์ส่วนหัว (header file) ไว้ดังนี้

```
#include <fstream>
```

ไฟล์ fstream.h จะเก็บข้อมูลที่เกี่ยวข้องกับการจัดการกับแฟ้มข้อมูล ก่อนที่แฟ้มข้อมูลแต่ละแฟ้มจะถูกสร้างขึ้นมา จะต้องกำหนดเป็นชนิดข้อมูลที่เรียกว่า file object สำหรับใช้ในการติดต่อกับแฟ้มข้อมูล เช่นเรา จะสร้างไฟล์ ชื่อ data1.dat เพื่อไว้เก็บข้อมูล จะต้องกำหนด file object ชื่อว่า myfile (การกำหนดชื่อ file object เป็นไปตามเงื่อนไขเดียวกับ identifier ชื่อสำคัญต้องสื่อความหมายและอ่านง่าย) ไว้ใช้ติดต่อเชื่อมโยงกับ ไฟล์ดังกล่าว โดยกำหนดดังนี้

```
fstream myfile;
```

file object ชื่อ myfile ที่กำหนดขึ้นมานี้ จะถูกนำไปใช้จัดการกับไฟล์ data1.dat โดยใช้ฟังก์ชัน open ซึ่งประกอบด้วยอาร์กิวเมนต์ (argument) 2 ตัวคือ

```
fstream object_name;
```

```
object_name.open ( "filename", access_mode);
```

สามารถเขียนรวมเป็นบรรทัดเดียวกันได้ดังนี้

```
fstream object_name ("filename", access_mode);
```

อาร์กิวเมนต์ตัวแรกเป็นชื่อไฟล์ ปิดหัวท้ายด้วยเครื่องหมาย “..” (double quotes) ตัวที่สอง จะใช้บอกสถานะไฟล์หรือลักษณะการเข้าถึงตัวไฟล์ (access mode) ว่าจะบันทึกข้อมูลลงในไฟล์ หรืออ่านข้อมูลจากไฟล์ นั้น

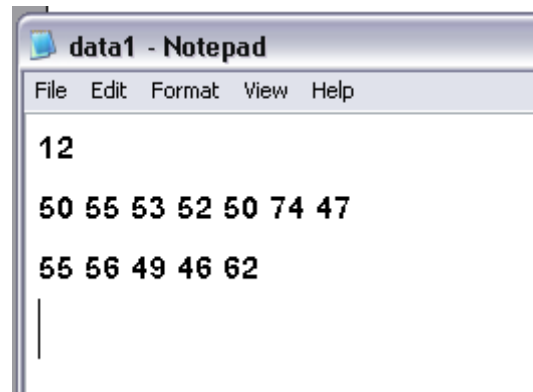
ถ้าต้องการอ่านข้อมูลจากไฟล์ จะต้องกำหนดสถานะเป็น ios:: in ถ้าต้องการจะเขียนข้อมูลลงในไฟล์ จะต้องกำหนดเป็น ios:: out

ถ้าเราต้องการอ่านข้อมูลที่เก็บไว้ใน data1.dat จะต้องเขียนคำสั่งในโปรแกรมดังนี้

```
myfile.open ( "data1.dat", ios :: in);
```

เมื่อกำหนดชื่อไฟล์และสถานะของไฟล์แล้ว เราสามารถนำข้อมูลจากไฟล์มาใช้ในการคำนวณ แทนที่จะต้องป้อนผ่านแป้นพิมพ์ทีละค่า ซึ่งจะทดลองทำดังต่อไปนี้

ขั้นแรกให้เปิดโปรแกรมประเภท editor เช่น Notepad ในวินโดว หรือ edit ใน DOS พิมพ์คะแนนของนักศึกษา 12 คน ดังนี้



ข้อมูลบรรทัดแรกเป็นจำนวนนักศึกษา ข้อมูลในบรรทัดที่ 2 และ 3 จะเป็นคะแนนของนักศึกษาทั้ง 12 คน จัดเก็บข้อมูลนี้ในไฟล์ชื่อ data1.dat โดยเก็บไว้ในโฟลเดอร์ที่เราจะสร้าง execute file จากนั้นเขียนและคอมไพล์โปรแกรมต่อไปนี้

```
1: // Program 3.6: findAv2.cpp
2: // Read data from file , data1.dat
3: #include <iostream>
4: #include <fstream>
5: using namespace std;
6:
7: main() {
8:     int num=0;
9:     double sum,score,average;
10:    fstream myfile;
11:
12:        myfile.open("data1.dat", ios::in);
13:        myfile >> num;
14:        sum = score = average = 0.0;
15:
16:        for (int i=1 ; i <= num; i++) {
17:            myfile >>score;
18:            sum = sum + score;
19:        }
20:        average = sum/num;
21:        cout << "The average of scores = " << average << endl;
22:        myfile.close();
23:        return 0;
24:    }
```

ผลการรันโปรแกรมจะเป็นดังนี้

```
C:\MyCpp\chap03>findav2
The average of scores = 54.0833
```

เราสามารถนำข้อมูลจากไฟล์มาเก็บไว้ในตัวแปรโดยใช้เครื่องหมาย >> (คล้ายกับ cin) ซึ่งเป็นการรับข้อมูล

```
myfile >> num;
```

เป็นการนำข้อมูลค่าแรก (คือ 12) มาเก็บไว้ในตัวแปร num จากนั้นวนรอบอีก 12 ครั้ง เพื่ออ่านคะแนนเข้ามาเก็บไว้ในตัวแปร score เมื่ออ่านข้อมูลจากไฟล์ครบแล้ว ต้องปิดแฟ้มข้อมูลก่อนจบโปรแกรมเสมอ โดยใช้คำสั่ง

```
myfile.close ();
```

ตัวอย่าง ต่อไปนี้เป็นข้อมูลที่ได้จากการวัดอุณหภูมิเป็นองศาเซลเซียสที่เวลาต่าง ๆ กัน ข้อมูลจะถูกบันทึกไว้ในแฟ้มข้อมูล ชื่อ temp.dat

เวลา	อุณหภูมิ
05.20	23
05.50	23.2
06.30	23.0
07.10	23.5
07.40	23.8
08.20	24
09.10	24.3
09.45	24.8
10.00	25
10.35	26.5
12.05	28.2
13.00	28.4
13.50	28.3
14.40	27.1
15.30	26.8
16.20	26
17.00	26
17.55	25.3
18.00	24.8

ข้อมูลที่ได้มีจำนวนหลายรายการ (record) เช่นรายการที่ 1 เมื่อเวลา 5.20 น.วัดอุณหภูมิได้ 23 องศาเซลเซียส ข้อมูลเหล่านี้อาจได้จากหัววัดอุณหภูมิ หรือเครื่องตรวจวัดอุณหภูมิ จำนวนรายการที่วัดได้มีจำนวนเท่าใดนั้น จะใช้ฟังก์ชัน eof () ( end of file) ตรวจสอบหาจุดสิ้นสุดของไฟล์ จากนั้นจะหาค่าเฉลี่ย ค่าสูงสุด และค่าต่ำสุดของอุณหภูมิ ในช่วงเวลาที่เก็บข้อมูล

```
1: // Program 3.7- temp.cpp
2: // Read data from file , temp.dat
3: #include <iostream>
4: #include <fstream>
```

```

5:   using namespace std;
6:
7:   main() {
8:       int num=0;
9:       double time;
10:      double temp;
11:      double max, min, average, maxtime, mintime;
12:      double sum;
13:      fstream myfile;
14:
15:          sum = max =average = 0;
16:          min = 1000;
17:          myfile.open("temp.dat", ios::in);
18:          if (myfile.fail() ) {
19:              cout << "Error opening temp.dat\n";
20:              return 1;
21:          }
22:          myfile >> time >> temp;
23:          while (!myfile.eof()) {
24:
25:              sum = sum + temp;
26:              if (temp > max) {
27:                  maxtime = time;
28:                  max = temp;
29:              }
30:              if ( temp < min ) {
31:                  mintime = time;
32:                  min = temp;
33:              }
34:              num++;
35:
36:              //      cout << num << " " << time << " " << temp << endl;
37:              myfile >> time >> temp;
38:          }
39:          average = sum/num;
40:
41:          cout << "The number of record = " << num << endl;
42:          cout << "Max temperature is " << max
43:              << " at time = " << maxtime << endl;
44:          cout << "Min temperature is " << min
45:              << " at time = " << mintime << endl;
46:
47:          cout << "Average temperature = " << average << endl;
48:          myfile.close();
49:          return 0;
50:
51:      }

```

ผลของการรันโปรแกรม เป็นดังภาพ

```

C:\MyCpp\chap03>temp
The number of record = 19
Max temperature is 28.4 at time = 13
Min temperature is 23 at time = 5.2
Average temperature = 25.3684

```

ทดลองพลิกแพลงเพื่อตรวจสอบการอ่านไฟล์ของภาษา C++

จะเห็นว่าจะมีการอ่านค่าเวลาและอุณหภูมิมาเก็บไว้ที่ตัวแปร time , temp ในบรรทัดที่ 22 ก่อน แล้วจึงมีการวนรอบเพื่อตรวจสอบจุดสิ้นสุดของไฟล์ จากนั้นจึงเป็นการอ่านค่าเวลา และอุณหภูมิของรายการที่เหลือในบรรทัดที่ 37 บรรทัดที่ 36 จะไว้ใช้ตรวจสอบข้อมูลที่อ่านเข้ามา ว่าถูกต้องหรือไม่

ถ้าจะพลิกการอ่านข้อมูลจากไฟล์ให้เป็นแบบอื่น ๆ บ้างดังตัวอย่างต่อไปนี้

```
1: // Program 3_19. temp2.cpp
2: //      Read data from file , temp.dat
3: #include <iostream>
4: #include <fstream>
5: using namespace std;
6:
7: main() {
8:     int num=0;
9:     double time;
10:    double temp;
11:    double max, min, average, maxtime, mintime;
12:    double sum;
13:    fstream myfile;
14:
15:        sum = max =average = 0;
16:        min = 1000;
17:        myfile.open("temp.dat", ios::in);
18:        if (myfile.fail() ) {
19:            cout << "Error opening temp.dat\n";
20:            return 1;
21:        }
22:        while (!myfile.eof()) {
23:            myfile >> time >> temp;
24:
25:            sum = sum + temp;
26:            if (temp > max) {
27:                maxtime = time;
28:                max = temp;
29:            }
30:            if ( temp < min ) {
31:                mintime = time;
32:                min = temp;
33:            }
34:            num++;
35:
36:            cout << num << " " << time << " " << temp << endl;
37:        }
38:        average = sum/num;
39:
40:        cout << "The number of record = " << num << endl;
41:        cout << "Max temperature is " << max
42:            << " at time = " << maxtime << endl;
43:        cout << "Min temperature is " << min
44:            << " at time = " << mintime << endl;
45:
46:        cout << "Average temperature = " << average << endl;
47:        myfile.close();
48:        return 0;
49:
```

```
50: }  
51:
```

จากตัวอย่างนี้จะเห็นว่า จะทำการอ่านข้อมูลสุดท้าย 2 ครั้ง ทำให้จำนวนข้อมูลกลายเป็น 20 รายการ ผลที่ตามมาก็คือคำนวณค่าเฉลี่ยผิดพลาด

```
C:\MyCpp\chap03>temp2  
1 5.2 23  
2 5.5 23.2  
3 6.3 23  
4 7.1 23.5  
5 7.4 23.8  
6 8.2 24  
7 9.1 24.3  
8 9.45 24.8  
9 10 25  
10 10.35 26.5  
11 12.05 28.2  
12 13 28.4  
13 13.5 28.3  
14 14.4 27.1  
15 15.3 26.8  
16 16.2 26  
17 17 26  
18 17.55 25.3  
19 18 24.8  
20 18 24.8  
The number of record = 20  
Max temperature is 28.4 at time = 13  
Min temperature is 23 at time = 5.2  
Average temperature = 25.34
```

ทดสอบโดยการเปลี่ยนคำสั่งบางคำสั่ง ดังที่พิมพ์ไว้ด้วยตัวหนา ในโปรแกรม temp3.cpp

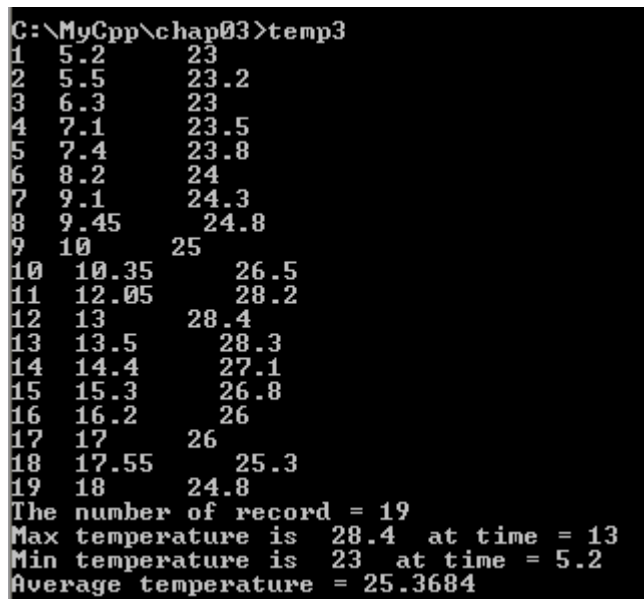
```
1: // Program 3_19. temp3.cpp  
2: // Read data from file , temp.dat  
3: #include <iostream>  
4: #include <fstream>  
5: using namespace std;  
6:  
7: main() {  
8: int num=0;  
9: double time;  
10: double temp;  
11: double max, min, average, maxtime, mintime;  
12: double sum;  
13: fstream myfile;  
14:  
15: sum = max =average = 0;  
16: min = 1000;  
17: myfile.open("temp.dat", ios::in);  
18: if (myfile.fail() ) {  
19: cout << "Error opening temp.dat\n";  
20: return 1;  
21: }  
22: while (myfile >> time >> temp) {  
23:  
24: sum = sum + temp;  
25: if (temp > max) {  
26: maxtime = time;
```

```

27:             max = temp;
28:         }
29:         if ( temp < min ) {
30:             mintime = time;
31:             min = temp;
32:         }
33:         num++;
34:
35:         cout << num << " " << time << " " << temp << endl;
36:     }
37:     average = sum/num;
38:
39:     cout << "The number of record = " << num << endl;
40:     cout << "Max temperature is " << max
41:         << " at time = " << maxtime << endl;
42:     cout << "Min temperature is " << min
43:         << " at time = " << mintime << endl;
44:
45:     cout << "Average temperature = " << average << endl;
46:     myfile.close();
47:     return 0;
48:
49: }
50:

```

ผลการรันโปรแกรม พบว่าเป็นไปอย่างถูกต้อง ดังรูป



```

C:\MyCpp\chap03>temp3
1 5.2 23
2 5.5 23.2
3 6.3 23
4 7.1 23.5
5 7.4 23.8
6 8.2 24
7 9.1 24.3
8 9.45 24.8
9 10 25
10 10.35 26.5
11 12.05 28.2
12 13 28.4
13 13.5 28.3
14 14.4 27.1
15 15.3 26.8
16 16.2 26
17 17 26
18 17.55 25.3
19 18 24.8
The number of record = 19
Max temperature is 28.4 at time = 13
Min temperature is 23 at time = 5.2
Average temperature = 25.3684

```

การบันทึกข้อมูลลงในแฟ้มข้อมูล

การบันทึกข้อมูลเก็บไว้ในแฟ้มข้อมูลก็เหมือนกับการแสดงผลที่จอภาพ ต่างกันตรงที่ข้อมูลถูกเขียนลงในแผ่นดิสก์ การบันทึกข้อมูลต่อไปนี้จะบันทึกเฉพาะตัวข้อมูลล้วนๆ ไม่มีส่วนหัวที่ใช้อธิบายจำนวนrecord หรือส่วนท้ายของข้อมูล



ตัวอย่าง การบันทึกข้อมูลต่อไปนี้จะบันทึกค่าความสูงและระยะไกลของวัตถุ ที่มีการเคลื่อนที่แบบวิถีโค้งภายใต้แรงโน้มถ่วงของโลกที่เวลาต่าง ๆ กัน โดยเพิ่มเวลาช่วงละ 0.5 วินาที กำหนดให้ความเร็วต้นของวัตถุเท่ากับ 24 m/s ทำมุม 60 องศา กับแนวราบ สมการความเร็วและการกระจัดที่เวลา t ใด ๆ เป็นไปตามสมการต่อไปนี้

$$v_x = u \cos \theta$$

$$v_y = u \sin \theta$$

$$x = u \cos \theta t$$

$$y = u \sin \theta t - \frac{1}{2} g t^2$$

```

1: // Program Projectile.cpp
2: // write data to file , prj.dat
3: #include <iostream>
4: #include <iomanip>
5: #include <fstream>
6: using namespace std;
7: const double PI = 3.1416;
8:
9: main() {
10: double u, delta_t, x,y, t,g;
11: fstream projectile;
12:     u = 24;
13:     delta_t = 0.5;
14:     g = 10;
15:     cout << "Writing the projectile motion information "
16:           << " \nTime (in sec) distance (m) and height (m) ";
17:     projectile.open ("prj.dat", ios::out);
18:     for (t=0; t <=5 ; t=t+delta_t ) {
19:         x = u* cos (60*PI/180)*t;
20:         y = u*sin( 60*PI/180)*t -0.5*g*t*t;
21:         cout << setw(4) << setprecision(4) << t << " "
22:               <<setw(8) << setprecision(4) << x << " "
23:               << setprecision(4) << y << " \n ";
24:         projectile << t << " " <<setprecision(4)<< x << " "
25:               <<setprecision(4)<< y << endl;
26:     }
27:     projectile.close();
28:     return 0;
29: }
30: }
```

เมื่อให้โปรแกรมทำงาน ผลลัพธ์ที่คำนวณได้และจะถูกเก็บไว้ในไฟล์ชื่อ prj.dat แสดงออกที่จอภาพดังรูป

```

C:\MyC++\chap03>projectile
Writing the projectile motion information
Time (in sec) distance (m) and height (m)
0 0 0
0.5 6 9.142
1 12 15.78
1.5 18 19.93
2 24 21.57
2.5 30 20.71
3 36 17.35
3.5 42 11.5
4 48 3.139
4.5 54 -7.719
5 60 -21.08

```

เมื่อเปิดไฟล์ prj.dat ด้วยโปรแกรม editor ในที่นี้ใช้ notepad จะเห็นข้อมูลที่เก็บไว้ดังภาพ

```

prj - Notepad
File Edit Format View Help
0 0 0
0.5 6 9.142
1 12 15.78
1.5 18 19.93
2 24 21.57
2.5 30 20.71
3 36 17.35
3.5 42 11.5
4 48 3.139
4.5 54 -7.719
5 60 -21.08

```

แบบฝึกหัด

1. จงหาค่าคำสั่งต่อไปนี้  
 floor (-2.6)      ceil (-2.6)  
 pow(2, -3)      sqrt(floor(10,7))
2. การหดสั้นของความยาว (Length contraction) ในทฤษฎีสัมพัทธภาพพิเศษของไอน์สไตน์ มีดังนี้

$$\text{length} = L_0 \sqrt{1 - \frac{v^2}{c^2}}$$

เมื่อ  $L_0$  คือความยาวของวัตถุเมื่ออยู่นิ่งเทียบกับผู้สังเกต

$v_0$  คือความเร็วของวัตถุ

c คือความเร็วของแสงในสุญญากาศ เท่ากับ  $3 \times 10^8$  m/s

จงเขียนสมการนี้ให้อยู่ในรูปนิพจน์คณิตศาสตร์ของคำสั่งในภาษา C++ จากนั้นเขียนเป็นโปรแกรม ป้อนค่า  $L_0, v$  ผ่านทางแป้นพิมพ์แล้วคำนวณหาค่า length

3. จงหาพื้นที่ของสามเหลี่ยมที่มีด้านแต่ละด้านยาว a, b, c โดยใช้สูตร

$$\text{area} = \sqrt{s(s-a)(s-b)(s-c)}$$

$$\text{เมื่อ } s = \frac{a+b+c}{2}$$

โดยให้โปรแกรมหยุดรอรับค่า a, b และ c จากแป้นพิมพ์

4. พิสัย(Range, R) ของมวลขึ้นหนึ่งที่เคลื่อนที่แบบวิถีโค้งภายใต้แรงดึงดูดของโลก หาได้จากสมการ

$$R = \frac{u^2 \sin 2\theta}{g}$$

เมื่อ u คือความเร็วต้นของมวล

$\theta$  คือมุมของ u กับพื้นดิน

$$g = 9.8 \text{ m/s}^2$$

ให้ป้อนค่าความเร็วต้น (u) และมุมที่ยิงสู่อากาศ  $\theta$

5. logarithm ของ x ฐาน b หาได้จาก

$$\log_b x = \frac{\ln x}{\ln b}$$

จงเขียนโปรแกรมพิมพ์ค่า log ฐาน 2 เช่น  $\log_2 8 = 3$

6. จงแปลงคำสั่งในภาษา C++ ต่อไปนี้ให้เป็นนิพจน์ทางคณิตศาสตร์

ก.  $dm = m * (\text{sqrt}(1+v/c)/\text{sqrt}(1 - v/c)-1);$

ข.  $\text{volume} = \text{PI} * r * r * h;$

ค.  $P = \text{atan2}(z, \text{sqrt}(x*x + y*y));$

7. ให้ n เป็นจำนวนเต็ม (integer) x เป็นจำนวนจริง (floating point number) คำสั่งต่อไปนี้ต่างกันอย่างไร

$$n = x;$$

และ  $n = \text{static\_cast}<\text{int}>(x + 0.5);$

x มีค่าเท่าใด จึงจะได้ผลลัพธ์จากทั้งสองคำสั่งเหมือนกัน และ x มีค่าเท่าใด ที่ทำให้ผลลัพธ์แตกต่างกัน

และจะเป็นอย่างไรถ้า x ติดลบ

8. จงอธิบายความแตกต่างระหว่าง

2, 20, "2", "2.0"

9. ผลลัพธ์จากคำสั่งต่อไปนี้ เป็นอย่างไร

$$x = 2;$$

$$y = x + x;$$

และ

```
s = "2";  
t = s + s;
```

10. จงหาคำที่ผิด

- ก. x และ "x" มีค่าเท่ากัน ถ้า x เป็นจำนวนเต็ม
- ข. static\_cast <int> (static\_cast<double>(x)) จะมีค่าเท่ากับ x ถ้า x เป็นจำนวนเต็ม
- ค. s.substr(0, s.length()) มีค่าเท่ากับ s

11. จงอธิบายความแตกต่างระหว่างการป้อนข้อมูลเป็นคำ (word oriented) และเป็นบรรทัด (line oriented) ในภาษา C++ ทำได้อย่างไร เมื่อใดจะใช้แบบไหน

12. จงหา syntax error จากโปรแกรมต่อไปนี้ ( 5 แห่ง)

```
1:      #include iostream  
2:      int main();  
3:      { cout << "Enter your number : "  
4:        cin >> a, b;  
5:        cout << "The sum of " << a << " and" << b  
6:        << " is " << a+b << endl;  
7:        return;  
8:      }  
9:  
10:
```

เฉลย บรรทัดที่ 1 ต้องมี <iostream> และมีคำสั่ง using namespace std;

บรรทัดที่ 2 หลังคำสั่ง main() ต้องไม่มี ;

บรรทัดที่ 4 ต้องประกาศตัวแปร a, b และต้องเปลี่ยนเป็น cin >> a >> b ;

บรรทัดที่ 7 แก้เป็น return 0;

13. จงหา logic error ของโปรแกรมต่อไปนี้ ( 3 แห่ง)

```
1:      #include <iostream>  
2:      using namespace std;  
3:  
4:      int main();  
5:      {  
6:          int total;  
7:          int x1, x2;  
8:          cout << "Enter your number : ";  
9:          cin >> x1;  
10:         total = total + x1;  
11:         cout << "Enter another number : ";  
12:         cin >> x2;  
13:         total = total + x1;  
14:         float average = total/2;  
15:         cout << "The average of the two number is " << average << endl;  
16:  
17:         return 0;  
18:  
19:     }
```

เฉลย บรรทัดที่ 10 ต้อง initialize ค่า total ก่อนนำไปใช้งาน

บรรทัดที่ 13 ควรเป็น total = total + x2

บรรทัดที่ 14 มีการคำนวณโดยใช้ข้อมูลต่างชนิดกัน โดยไม่มีการ casting

#### บทที่ 4 การเขียนฟังก์ชันใช้งาน

การทำงานของโปรแกรมในภาษา C++ จะเริ่มต้นที่ฟังก์ชัน `main()` โปรแกรมหนึ่ง ๆ อาจจะประกอบด้วยชุดคำสั่งที่ประมวลผลหรือทำงานต่างหน้าที่กัน ชุดคำสั่งชุดที่ 1 อาจรับข้อมูลเข้ามาเก็บไว้ในหน่วยความจำ ชุดคำสั่งชุดที่ 2 อาจเรียงลำดับข้อมูล ชุดคำสั่งที่ 3 คำนวณหาค่าเฉลี่ย ชุดคำสั่งที่ 4 อาจทำหน้าที่แสดงผลออกที่จอภาพ เป็นต้น การนำชุดคำสั่งซึ่งทำงานในหน้าที่ต่าง ๆ กัน ไปรวมกันอยู่ในฟังก์ชัน `main()` ทั้งหมด ทำให้โปรแกรมมีลักษณะซับซ้อน คำสั่งที่เรียงลดหลั่นกันมาทำให้ดูยุ่งยาก อ่านได้ยาก ขาดคุณสมบัติที่เรียกว่าเรียบง่าย สบายตา (Simplicity and readability) เราจึงนิยมนำชุดคำสั่งที่ทำหน้าที่ประมวลผลงานหนึ่ง ๆ ไปเขียนรวมกันเป็นฟังก์ชัน หรือ เรียกทั่ว ๆ ไป เป็น มอดูล (module)

ฟังก์ชันในภาษา C++ อาจนำมาจากฟังก์ชันสำเร็จรูป ที่มีอยู่ใน standard C++ Library เช่น `setw` (ใช้กำหนดความกว้างของฟิลด์ที่จะแสดงผลจำนวนเลข) หรือ `sqrt` ใช้หาค่ารากที่สองของจำนวนจริงใด ๆ ฟังก์ชันใดที่โปรแกรมเมอร์เขียนขึ้นมาเพื่อใช้ประมวลผลงานหนึ่ง ๆ เรียกว่า programmer defined function

แนวคิดเรื่องฟังก์ชัน เกิดจากแนวคิดการแก้ปัญหาแบบ Divide and conquer แบ่งปัญหาออกเป็น ส่วนย่อย ๆ (module) ส่วนย่อย ๆ นี้จะทำหน้าที่แก้ปัญหาในส่วนนั้น ๆ ให้สมบูรณ์ในตัวเอง มีลักษณะง่าย และเข้าใจขั้นตอนการทำงานได้ไม่ยากนัก

ประโยชน์การแบ่งโปรแกรมนาน ๆ ให้เป็นฟังก์ชัน คือ ฟังก์ชันมีการทำงานตามจุดประสงค์ที่แน่นอน เราอาจเขียนโปรแกรมตรวจสอบการทำงานของฟังก์ชัน โดยแยกออกจากโปรแกรมหลัก เพราะฟังก์ชันมีขนาดเล็ก การตรวจสอบจึงทำได้ง่ายและสะดวกกว่า ฟังก์ชันใดเมื่อผ่านการตรวจสอบมาดีแล้ว เมื่อนำไปใช้ร่วมกับโปรแกรมหลัก ก็ไม่จำเป็นต้องตรวจสอบซ้ำตรงฟังก์ชันนี้อีก เช่น เขียนฟังก์ชันหาค่าเฉลี่ยของจำนวนเลข  $n$  ชุด เมื่อตรวจสอบแล้วพบว่าการทำงานของฟังก์ชันนี้ไม่มีข้อบกพร่อง เราสามารถนำฟังก์ชันไปใช้ในโปรแกรมอื่น ๆ ที่ต้องการหาค่าเฉลี่ยได้อีกด้วย วิธีการเขียนฟังก์ชันเพียงครั้งเดียวแล้วนำไปใช้กับโปรแกรมอื่น ๆ ได้โดยไม่ต้องเขียนโปรแกรมซ้ำนี้ เรียกว่า การนำกลับมาใช้ใหม่ (reusability) เป็นการประหยัดเวลาในการพัฒนาโปรแกรมขนาดใหญ่ ๆ ได้เป็นอย่างมาก

การเขียนโปรแกรมในลักษณะ มอดูล หรือฟังก์ชันนี้ช่วยลดความยาวและความซับซ้อนของโปรแกรม ชุดคำสั่งบางคำสั่งที่อาจเกิดขึ้นซ้ำ ๆ ตรงส่วนต่าง ๆ ของโปรแกรม เมื่อถูกนำไปทำเป็นมอดูล หรือฟังก์ชัน ชุดคำสั่งที่ปรากฏซ้ำ ๆ นั้น จะถูกเขียนแทนด้วยคำสั่งที่เรียกใช้ฟังก์ชันนั้นเพียงบรรทัดเดียว

การแบ่งโปรแกรมขนาดใหญ่ออกเป็นมอดูล ทำให้โปรแกรมเมอร์หลาย ๆ คนสามารถทำงานคู่ขนานกันไปในโครงการเดียวกัน โดยต่างคนต่างเขียนฟังก์ชันที่ได้รับมอบหมาย จากนั้นนำฟังก์ชันทั้งหมดมารวมกันเป็นโครงการ ทำให้ระยะเวลาในการเขียนโปรแกรมในโครงการหนึ่ง ๆ สั้นลง

ฟังก์ชันหรือมอดูลหนึ่ง ๆ ซึ่งถูกเขียนให้ประมวลผลงานหนึ่งสำเร็จตามจุดประสงค์นั้นเป็นการสนับสนุนแนวคิดที่เรียกว่า ซ่อนรายละเอียดที่ไม่จำเป็นต้องรู้ (abstraction) ฟังก์ชันประกอบด้วยรายละเอียดของขั้นตอนในการทำงานขั้นนั้น โปรแกรมเมอร์สามารถอ้างถึงหรือเรียกใช้ฟังก์ชันนั้นโดยไม่ต้องสนใจในรายละเอียดของฟังก์ชันนั้นว่ามันทำงานอย่างไร อาจมองฟังก์ชันหรือมอดูล เป็น “กล่องดำ” (black box) เพียงป้อนข้อมูลอินพุตให้ถูกต้อง ก็จะได้ผลลัพธ์จากกล่องดำนั้น ใน standard C++ library มีฟังก์ชันใช้คำนวณหาค่าลอการิทึม เราสามารถนำฟังก์ชันนั้นมาหาค่า  $\log$  ของจำนวนใด ๆ ได้โดยไม่ต้องไปรู้ในรายละเอียดว่าฟังก์ชันนั้นใช้วิธีเปิด

ตารางหาค่า log หรือคำนวณโดยใช้การประมาณค่าจากอนุกรม การซ่อนรายละเอียดจะช่วยลดเวลาการพัฒนาโปรแกรม และเพิ่มความมั่นใจว่าโปรแกรมไม่มีข้อผิดพลาด (increase its quality)

ฟังก์ชันหนึ่ง ๆ ควรมีความยาวเท่าใด ถึงแม้จะไม่มีข้อกำหนดความยาวของชุดคำสั่ง โปรแกรมเมอร์หลายคนเสนอแนะว่า ฟังก์ชันควรมีขนาดยาวพอเหมาะกับ 1 จอภาพ เพื่อที่สามารถอ่านคำสั่งในฟังก์ชันได้ตลอด ฟังก์ชันที่มีขนาดเล็กนั้นง่ายต่อทำความเข้าใจและการแก้ไข ฟังก์ชันหนึ่งควรทำงานหรือเกิดผลลัพธ์เพียง 1 งาน ถ้าเขียนฟังก์ชันใดแล้วพบว่ามันยาวเกินไป แสดงว่าควรแตกฟังก์ชันนั้นออกเป็นฟังก์ชันย่อย ๆ อีกได้แล้ว จะแสดงตัวอย่างการเขียนโปรแกรมที่ประกอบด้วยฟังก์ชันที่เราเขียนขึ้นใช้เองดังนี้

ตัวอย่าง 4.1 เมื่อป้อนค่ารัศมีของวงกลม โปรแกรมจะคำนวณหาพื้นที่ของวงกลมและเส้นรอบวง

```
1: // Program 4.1 Circle.cpp
2: // Feb16, 2004
3: #include <iostream>
4: using namespace std;
5:
6: float find_area(float);
7: float compute_perimeter(float);
8: float input_radius(void);
9: void display_result(float);
10:
11: int main() {
12:     float area, circumference;
13:     float r;
14:     r = input_radius();
15:     cout << "Calculating area of a circle \n";
16:     area = find_area(r);
17:     display_result( area);
18:     cout << "Compute circumference of this circle \n";
19:     circumference = compute_perimeter(r);
20:     display_result(circumference);
21:     return 0;
22: }
23:
24: float input_radius() {
25:     float radius;
26:     cout << "Input a radius of circle : ";
27:     cin >> radius;
28:     return radius;
29: }
30:
31: float find_area(float radius) {
32:     float a;
33:     a = 3.14159 * radius * radius;
34:     return a;
35: }
36:
37: float compute_perimeter(float radius) {
38:     return (3.14159*2*radius);
39: }
40:
41: void display_result(float result) {
42:     cout << "The result : " << result << endl;
43: }
```

จากตัวอย่าง 4.1 จะแบ่งตัวงานออกเป็น 4 งานย่อย ๆ คือ ส่วนที่ทำหน้าที่รับข้อมูลได้แก่ฟังก์ชัน `input_radius ( )` ส่วนที่ 2 เป็นส่วนที่คำนวณหาพื้นที่ของวงกลม ฟังก์ชัน `find_area` ส่วนที่ 3 ทำหน้าที่คำนวณความยาวเส้นรอบวงกลม `compute_perimeter ( )` ส่วนที่ 4 เป็นส่วนที่แสดงผลการคำนวณ คือฟังก์ชัน `display ( )` งานแต่ละส่วนถูก แยกเขียนเป็นฟังก์ชัน 4 ฟังก์ชัน จะเห็นว่าจำนวนบรรทัดในฟังก์ชัน `main ( )` ซึ่งเป็นส่วนหลักของโปรแกรมจะสั้นลง ในแต่ละบรรทัดจะสื่อความหมายชัดเจนว่ากำลังทำอะไร รายละเอียดของแต่ละส่วน เช่นคำนวณหาพื้นที่อย่างไร จะถูกซ่อนไว้ในฟังก์ชัน

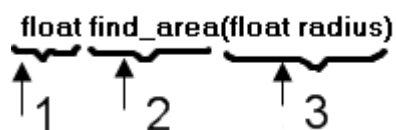
ฟังก์ชันที่เขียนขึ้นใช้เองจะประกอบด้วย 2 ส่วนใหญ่ คือ ส่วนหัว (header) และส่วนที่เป็นร่างของฟังก์ชัน (body)

ส่วนหัวของฟังก์ชัน มีรูปแบบการเขียนดังนี้

```
return_type function_name ( [ type [ parameter_name] ] ,...)
```

เช่น ฟังก์ชัน `find_area` จะมีส่วนหัวของฟังก์ชัน ในบรรทัดที่ 31

```
float find_area(float radius)
```



จะเห็นว่าไม่ต้องมี เซมิโคลอน ต่อท้าย

หมายเลข 1 คือ return type คือชนิดของข้อมูลที่ฟังก์ชันส่งค่ากลับคืนไปยังโปรแกรมที่เรียกใช้งาน

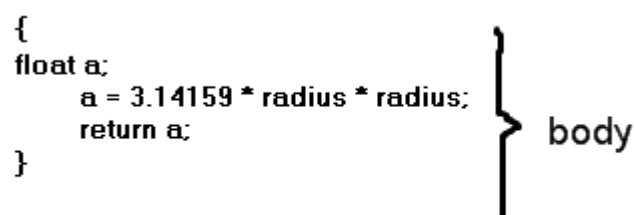
หมายเลข 2 คือ function name เป็นชื่อของฟังก์ชัน การตั้งชื่อฟังก์ชันมีกฎเกณฑ์เช่นเดียวกับการตั้งชื่อตัวแปร ต้องขึ้นต้นด้วยตัวอักษรเท่านั้น อักษรถัดไปจะเป็นตัวเลขหรือตัวอักษรหรือเครื่องหมาย under score ผสมปนกันก็ได้ ชื่อของฟังก์ชันจะยาวเท่าใดก็ได้ แต่คอมไพเลอร์จะเก็บชื่อฟังก์ชันไว้เพียง 32 ตัวอักษรแรกเท่านั้น ชื่อฟังก์ชันควรตั้งให้สอดคล้องและสื่อความหมายกับงานที่ฟังก์ชันทำ เพื่อให้การอ่านโปรแกรมแล้วสามารถเข้าใจได้ง่าย

ตัวอักษรตัวพิมพ์ใหญ่และตัวพิมพ์เล็ก จะทำให้ฟังก์ชันกลายเป็นฟังก์ชันคนละฟังก์ชันกัน เช่น `find_area` และ `Find_Area` ถือว่าเป็นคนละฟังก์ชันกัน

หมายเลข 3 จะอยู่ในวงเล็บเรียกว่า parameter list ฟังก์ชัน `find_area` มีพารามิเตอร์ที่ใช้ในฟังก์ชันเพียง 1 ตัว คือ `float radius` ส่วนที่เป็น body ของฟังก์ชันจะนำพารามิเตอร์เหล่านี้ไปใช้ในการทำงาน

ส่วนที่จัดว่าเป็น body ของฟังก์ชันจะอยู่ระหว่างวงเล็บปีกกา ประกอบด้วยส่วนที่เป็นประกาศของตัวแปรที่ใช้งานในฟังก์ชัน ตามด้วยประโยคคำสั่งต่าง ๆ ที่ฟังก์ชันนั้นทำงาน และจะปิดท้ายด้วยคำสั่ง `return` ดังปรากฏในบรรทัดที่ 34

```
{
    float a;
    a = 3.14159 * radius * radius;
    return a;
}
```



คำสั่ง `return` เป็นการจบการทำงานของฟังก์ชันนั้นและส่งค่าที่ฟังก์ชันนั้นทำงานได้กลับไปให้กับโปรแกรมที่เรียกใช้ฟังก์ชันนั้นใช้งาน ฟังก์ชันหนึ่ง ๆ ส่งค่าคืนกลับได้เพียงค่าเดียว ฟังก์ชันอาจทำงานโดยไม่มี การส่งค่าคืนกลับก็ได้ ดังเช่น ฟังก์ชัน `display ( )` ในบรรทัดที่ 41



```
void display_result(float result) {
    cout << "The result : " << result << endl;
}
```

← ส่วนหัวของฟังก์ชัน

คำสั่ง void หมายถึง ฟังก์ชันนี้ไม่มีการส่งค่าคืนกลับไปยังโปรแกรมที่เรียกใช้งาน

ย้อนกลับไปดูโปรแกรม 4.1 อีกครั้ง จะเห็นโปรแกรมที่มีการแบ่งเป็นส่วนย่อย ๆ แต่เป็นฟังก์ชันนั้นจะเกี่ยวข้องกับส่วนต่าง ๆ ดังนี้

```
1: // Program 4.1 Circle.cpp
2: // Feb16, 2004
3: #include <iostream>
4: using namespace std;
5:
6: float find_area(float);
7: float compute_perimeter(float);
8: float input_radius(void);
9: void display_result(float);
10:
11: int main() {
12:     float area, circumference;
13:     float r;
14:     r = input_radius();
15:     cout << "Calculating area of a circle \n";
16:     area = find_area(r);
17:     display_result( area);
18:     cout << "Compute circumference of this circle \n";
19:     circumference = compute_perimeter(r);
20:     display_result(circumference);
21:     return 0;
22: }
23:
24: float input_radius() {
25:     float radius;
26:     cout << "Input a radius of circle : ";
27:     cin >> radius;
28:     return radius;
29: }
30:
31: float find_area(float radius) {
32:     float a;
33:     a = 3.14159 * radius * radius;
34:     return a;
35: }
36:
37: float compute_perimeter(float radius) {
38:     return (3.14159*2*radius);
39: }
40:
41: void display_result(float result) {
42:     cout << "The result : " << result << endl;
43: }
```

ส่วน A เรียกว่า การกำหนดรูปแบบฟังก์ชัน (function prototype) ก่อนที่จะนำไปใช้งาน (ในภาษา C ไม่จำเป็นต้องมีส่วนนี้)

ส่วน B เรียกว่า function definition ประกอบด้วยส่วนหัวและส่วนที่เป็นร่างของฟังก์ชัน ส่วนนี้จะอยู่ก่อน main() หรือหลังก็ได้

ส่วน C คือส่วนที่มีการเรียกใช้ฟังก์ชัน (function call)

ฟังก์ชันที่เป็นมาตรฐานใน Library การกำหนดรูปแบบฟังก์ชัน จะมีการกำหนดไว้เรียบร้อยแล้วใน include file

ต่อไปนี้เป็นตัวอย่างการเขียน function prototype เพิ่มเติม

long calculate\_rectangle (long length, long width);

คำนวณพื้นที่สี่เหลี่ยมผืนผ้าส่งค่ากลับเป็นจำนวนเต็มชนิด long มีพารามิเตอร์ 2 ตัวคือ length และ width ซึ่งเป็นตัวแปรชนิด long ด้วยกันทั้งคู่

void print\_result (int number);

ฟังก์ชัน print\_result ไม่มีการส่งค่ากลับ มีพารามิเตอร์เป็นข้อมูลชนิดจำนวนเต็ม 1 ตัว

int get\_your\_selection ( );

ส่งค่ากลับเป็นข้อมูลชนิดจำนวนเต็ม ไม่มีพารามิเตอร์

myfunction ( );

ส่งค่ากลับเป็นข้อมูลประเภท int ไม่มีพารามิเตอร์

ขอบเขตของตัวแปรและฟังก์ชัน

ตัวแปรที่ถูกประกาศไว้ภายในบล็อกหรือฟังก์ชันหนึ่ง จะถูกจำกัดการใช้งานได้เพียงภายในบล็อกนั้น หรือภายในฟังก์ชันนั้นเท่านั้น เรียกตัวแปรประเภทนี้ว่าเป็นตัวแปรท้องถิ่น (local variable) คำสั่งอื่น ๆ นอกบล็อกนั้นหรือฟังก์ชันอื่น ไม่สามารถนำตัวแปรนั้นมาใช้งานได้

ในโปรแกรม 4.1 ตัวแปรที่ประกาศไว้ในฟังก์ชัน main คือ area, circumference, r ตัวแปรเหล่านี้จะถูกใช้งานภายในฟังก์ชัน main ฟังก์ชันอื่น ๆ ได้แก่ find\_area, input\_radius, compute\_perimeter ไม่สามารถนำค่าที่เก็บไว้ในตัวแปรมาใช้ในการทำงานได้ ในทำนองเดียวกัน ในฟังก์ชัน input\_radius จะมีตัวแปร radius ชนิด float ใช้งานได้เฉพาะในฟังก์ชันนี้เท่านั้น ฟังก์ชัน main หรือ ฟังก์ชันอื่น ไม่สามารถนำตัวแปรนี้ไปใช้งานได้ ในฟังก์ชัน find\_area มีตัวแปรชื่อ area ซึ่งเป็นตัวแปรชนิด float มีชื่อซ้ำกับตัวแปร area ที่ปรากฏในฟังก์ชัน main ถือว่าเป็นตัวแปรคนละตัวกัน ตัวแปรจะยึดครองหน่วยความจำ ขณะที่ฟังก์ชันนั้น หรือบล็อกคำสั่งนั้น ทำงาน เมื่อบล็อกหรือฟังก์ชันนั้นสิ้นสุดการทำงาน ตัวแปรเหล่านั้นจะไม่ถูกลบจากหน่วยความจำทันทีทันใด ตัวแปรเหล่านั้นจะคงอยู่ในหน่วยความจำอีกถ้ามีการทำงานในบล็อกนั้นหรือในฟังก์ชันนั้นซ้ำอีกครั้ง

ตัวอย่างต่อไปนี้จะแสดงให้เห็นถึงขอบเขตของตัวแปรในบล็อกและฟังก์ชัน

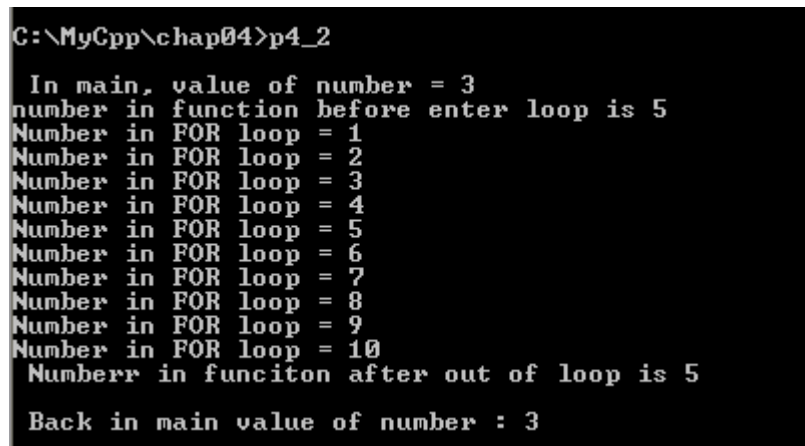
```

1: // Program 4.2 Demonstrate variable scope
2: // Feb 16, 2004
3: #include <iostream>
4: using namespace std;
5:
6: void displaynumber(void);
7:
8: main() {
9:     int number = 3;
10:    cout << "\n In main, value of number = " << number << endl;
11:    // now call displaynumber for displaying a nubmer
12:    displaynumber();
13:    //back to main program
14:    cout << "\n Back in main value of number : " << number << endl;
15:    return 0;
16: }
```

```

17:
18: void displaynumber() {
19:     int number = 5;
20:     cout << "number in function before enter loop is " << number << endl;
21:     for (int number = 1 ; number <= 10 ; number++) {
22:         cout << "Number in FOR loop = " << number << endl;
23:     }
24:     cout << " Numberr in funciton after out of loop is " << number << endl;
25: }
26:

```



```

C:\MyCpp\chap04>p4_2
In main, value of number = 3
number in function before enter loop is 5
Number in FOR loop = 1
Number in FOR loop = 2
Number in FOR loop = 3
Number in FOR loop = 4
Number in FOR loop = 5
Number in FOR loop = 6
Number in FOR loop = 7
Number in FOR loop = 8
Number in FOR loop = 9
Number in FOR loop = 10
Numberr in funciton after out of loop is 5
Back in main value of number : 3

```

โปรแกรม 4.2 เริ่มต้นที่ main ( ) ภายในฟังก์ชัน main ( ) มีการกำหนดตัวแปรชนิด int ชื่อ number ให้มีค่าเป็น 3

บรรทัดที่ 10 เมื่อพิมพ์ค่า number จะได้ผลลัพธ์เท่ากับ 3 บรรทัดที่ 12 เรียกใช้ฟังก์ชัน displaynumber() ในฟังก์ชันนี้มีตัวแปรท้องถิ่น ชื่อ number เหมือนกัน แต่กำหนดให้มีค่าเป็น 5 และจะแสดงผล number มีค่าเท่ากับ 5

ในการวนรอบด้วยคำสั่ง for ในบรรทัดที่ 21 มีการประกาศค่าตัวแปร ชื่อ number (เป็นตัวแปรตัวที่ 3 ที่มีชื่อ number เหมือนกัน) โดยให้วนรอบ เริ่มตั้งแต่ number = 1 วนรอบจำนวน 10 ครั้ง ในแต่ละครั้งจะแสดงค่า number บนจอภาพ จะเห็นว่าค่า number จะมีค่าตั้งแต่ 1 ถึง 10 เมื่อวนรอบจบแล้ว ให้แสดงผล number อีกครั้ง ดังปรากฏในคำสั่งบรรทัดที่ 24 ขณะนี้ การทำงานของโปรแกรมออกมานอกขอบเขตของการวนรอบของคำสั่ง for แล้ว ดังนั้นค่า number ที่ได้ในบรรทัดที่ 24 นี้จึงเป็นค่าของตัวแปร number = 5 ที่กำหนดไว้ในฟังก์ชัน displaynumber ( )

เมื่อสิ้นสุดการทำงานของฟังก์ชันในบรรทัดที่ 13 ให้แสดงผล number อีกครั้ง บรรทัดที่ 14 จะนำค่า number = 3 แสดงผลบนจอภาพ จะเห็นว่าค่า number ที่กำหนดไว้ในฟังก์ชัน displaynumber ( ) ไม่มีผลต่อตัวแปรในฟังก์ชัน main ( )

โปรแกรมนี้นี้แสดงให้เห็นขอบเขตการทำงานของตัวแปรที่ถูกกำหนดค่าไว้คนละที่เท่านั้น ในทางปฏิบัติควรกำหนดชื่อตัวแปรทั้งสามตัวให้มีชื่อต่างกันไปจะดีกว่า เพื่อลดความสับสน ขณะไล่ดูการทำงานของโปรแกรมตัวแปรแบบโกลบบอล (Global variable)

ตัวแปรที่นิยามไว้นอกฟังก์ชันใด ๆ รวมทั้ง main ด้วย เป็นตัวแปรที่ทุกฟังก์ชันมองเห็นและสามารถนำไปใช้งานได้ ในบางกรณีที่มีโปรแกรมเมอร์ ต้องการให้ฟังก์ชันทุกฟังก์ชันสามารถค่าจากตัวแปรนี้ไปใช้หรือเปลี่ยนค่าตัวแปรได้โดยไม่ต้องผ่านข้อมูลในตัวแปรผ่านทางตัวพารามิเตอร์ของฟังก์ชัน โดยปกติเราไม่นิยมใช้ตัวแปรแบบโกลบบอลในภาษา C++ เพราะค่อนข้างเสี่ยงต่ออันตราย ทั้งนี้เพราะการใช้ข้อมูลร่วมกันในตัวแปรแบบโกลบบอล ฟังก์ชันอื่น ๆ ต่างก็สามารถแก้ไขข้อมูลในตัวแปร โดยมีได้ตั้งใจ อาจทำให้โปรแกรมเกิดข้อผิดพลาดและการแก้ไขทำได้ลำบาก

ตัวแปรท้องถิ่นถึงแม้จะมีชื่อเดียวกันกับตัวแปรแบบโกลบบอล ก็จะไม่กระทบกระเทือนถึงข้อมูลที่เก็บไว้ในตัวแปรแบบโกลบบอล การอ้างถึงตัวแปรในฟังก์ชันใดที่มีชื่อซ้ำกัน จะหมายถึงตัวแปรประจำถิ่นในฟังก์ชันนั้น

```
1: // Program 4.3 global variable demonstration
2: // Feb 16, 2004
3: #include <iostream>
4: using namespace std;
5:
6: void displayvar(); //function prototype
7: int x = 10; // Global variables
8: int y = 20;
9:
10: main() {
11:     int x = 5;
12:     cout << "\n In main, x = " << x << endl;
13:     cout << "\n In main, y = " << y << endl;
14:     cout << " -----" << endl;
15:     // now call displayvar
16:     displayvar();
17:     //back to main program
18:     cout << " -----" << endl;
19:     cout << "\n In main again, x = " << x << endl;
20:     cout << "\n In main again, y = " << y << endl;
21:     cout << "\n Global x = " << ::x << endl;
22:     return 0;
23: }
24:
25: void displayvar() {
26:     int x = 2;
27:     cout << "\n In function, x = " << x << endl;
28:     cout << "\n In function, y = " << y << endl;
29: }
30:
```

```

C:\MyCpp\chap04>p4_3
In main, x = 5
In main, y = 20
-----
In function, x = 2
In function, y = 20
-----
In main again, x = 5
In main again, y = 20
Global x = 10

```

จากโปรแกรม 4.3 บรรทัดที่ 7 และ 8 เป็นการประกาศตัวแปรชนิด int ชื่อ x และ y ซึ่งเป็นตัวแปรแบบโกลบบอล กำหนดให้ x = 10, y = 20 ตามลำดับ

บรรทัดที่ 12 และ 13 แสดงผลค่า x และ y ในฟังก์ชันเนื่องจากการนิยามตัวแปร x ในฟังก์ชัน main() เข้ากับตัวแปรโกลบบอล ผลลัพธ์ที่ได้จึงเป็นค่า x ของตัวแปรท้องถิ่น และค่า y ของตัวแปรโกลบบอล

เมื่อเรียกใช้งานฟังก์ชัน displayvar() ในบรรทัดที่ 16 ฟังก์ชันนี้จะมีการนิยามและประกาศค่าตัวแปร 1 ตัวคือ int x = 2 การแสดงค่า x จึงใช้ค่าตัวแปรท้องถิ่น

เมื่อฟังก์ชันทำงานจบกลับมาที่ main() แสดงผลค่า x และ y อีกครั้ง จะเป็นค่า x ในตัวแปรท้องถิ่น และค่า y ที่เก็บไว้ในตัวแปรแบบโกลบบอล

ในบรรทัดที่ 21 แสดงให้เห็นว่าถ้าต้องการเข้าถึงข้อมูลที่เก็บไว้ในตัวแปรโกลบบอล x ในฟังก์ชัน main() ทำได้โดยใช้เครื่องหมาย :: (scope resolution operator) กำกับหน้าตัวแปร x

```
cout << "Global x << ::x << endl;
```

เครื่องหมายนี้บอกให้คอมไพเลอร์รู้ว่าต้องการแสดงค่า x ที่เก็บไว้ในตัวแปรแบบโกลบบอล ไม่ใช่ค่าที่เก็บไว้ในตัวแปรท้องถิ่น

การผ่านค่า parameter ไปยังฟังก์ชัน (Passing parameter to function)

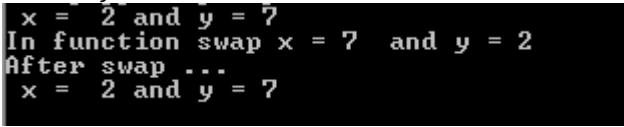
เมื่อมีการเรียกใช้ฟังก์ชันในโปรแกรมใด จะต้องมีการผ่านค่าพารามิเตอร์ ไปยังฟังก์ชัน ตามที่ประกาศไว้ในการกำหนดรูปแบบฟังก์ชัน (Function prototype) การผ่านค่าใน C++ มี 2 แบบดังนี้

1. การผ่านเฉพาะค่าของตัวแปรนั้น (passing by value) ฟังก์ชันจะสำเนาเฉพาะข้อมูลของตัวแปรที่ถูกส่งผ่านไปประมวลผล ฟังก์ชันจะสร้างตัวแปรชนิดเดียวกัน โดยข้อมูลที่เก็บไว้ในตัวแปรที่ถูกส่งผ่านนั้นยังคงเดิม นั่นคือฟังก์ชันจะคัดลอกเฉพาะข้อมูลจากตัวแปรที่ส่งผ่านและสร้างตัวแปรที่มีชนิดข้อมูลแบบเดียวกันในหน่วยความจำคนละแห่ง

ในโปรแกรม 4.1 คำสั่งเรียกใช้ฟังก์ชัน find\_area (float r) เป็นการผ่านค่า r (รัศมีของวงกลม) ไปให้ฟังก์ชัน find\_area หาพื้นที่ของวงกลม ฟังก์ชัน find\_area จะก๊อปปี้เฉพาะค่า r ไปใช้ในการคำนวณเท่านั้น ไม่ทำให้ค่าที่เก็บไว้ใน r มีการเปลี่ยนแปลงแต่อย่างใด

การผ่าน r ในการเรียกใช้ฟังก์ชัน computer\_preimeter ก็เช่นเดียวกัน เป็นการผ่านค่าแบบ passing by value เพราะข้อมูลในตัวแปรที่ใช้ในการผ่านค่าไม่มีการเปลี่ยนแปลง การเขียนโปรแกรมจึงต้องพึงระวัง ในกรณีที่ต้องการให้ฟังก์ชันนั้นมีการปรับปรุงข้อมูล ตัวแปรที่ใช้ผ่านค่า ดังตัวอย่างต่อไปนี้

```
1: // Program 4.4 Demonstrate passing by value
2: // Feb 16, 2004
3: #include <iostream>
4: using namespace std;
5:
6: void swap(int, int);
7:
8: main() {
9:     int x, y;
10:    x = 2;
11:    y = 7;
12:    cout << " x = " << x << " and y = " << y << endl;
13:    swap(x,y);
14:    cout << "After swap ..." << endl;
15:    cout << " x = " << x << " and y = " << y << endl;
16:    return 0;
17: }
18: void swap ( int a, int b) {
19:     int temp;
20:     temp = a;
21:     a = b;
22:     b = temp;
23:     cout << "In function swap x = " << a << " and y = " << b << endl;
24: }
```



```
x = 2 and y = 7
In function swap x = 7 and y = 2
After swap ...
x = 2 and y = 7
```

โปรแกรมนี้ต้องการสลับค่าระหว่าง x กับ y โดยสร้างฟังก์ชันชื่อ swap ให้ทำการสลับค่า จะเห็นว่าเมื่อผ่านค่า x = 2, y = 7 ไปให้ฟังก์ชัน swap ฟังก์ชันจะรับค่าไปสลับได้อย่างถูกต้อง สังเกตจากการแสดงผลภายในฟังก์ชัน จะได้ x = 7, y = 2

เมื่อฟังก์ชัน swap ทำงานเสร็จแล้ว กระโดดกลับมายังฟังก์ชัน main เมื่อแสดงผลค่า x, y พบว่ายังคงมีค่าเท่าเดิม ค่าที่เก็บไว้ในตัวแปร x, y ไม่มีการสลับแต่อย่างใด

การผ่านค่าแบบ passing by value นอกจากสามารถใช้ตัวแปรเป็นตัวพารามิเตอร์แล้ว ยังสามารถใช้ค่าคงที่ หรือ นิพจน์ ในการผ่านค่าได้ด้วย เช่น

```
find_area ( 5 ) ; // คำนวณพื้นที่วงกลมรัศมี 5 หน่วย
```

```
find_area ( 3*x + 7 ) ; // คำนวณพื้นที่วงกลมรัศมีเกิดจาก 3x + 7
```

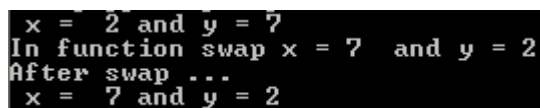
```
display( find_area( 5 ) ) // แสดงผลลัพธ์ที่ได้จากการคำนวณหาพื้นที่วงกลม เป็นการให้ฟังก์ชันเป็นพารามิเตอร์
```

2. การผ่านโดยการอ้างอิงตำแหน่งหน่วยความจำของตัวแปรนั้น (passing by reference) ประโยชน์ของการผ่านค่าโดยวิธีนี้คือฟังก์ชันสามารถเข้าถึงตัวแปรใน calling program และใช้กรณีที่ต้องการให้ฟังก์ชัน

ส่งค่ากลับคืนมากกว่า 1 ค่า หรือต้องการเปลี่ยนแปลงค่าตัวแปรหรือตัวแปรที่ใช้ผ่านค่าใช้เนื้อที่หน่วยความจำมาก เสียเวลาในการคัดลอก เช่นข้อมูลรูปภาพขนาด 5 Mb เป็นต้น กลไกการผ่านค่าแบบนี้จะต่างจากการผ่านค่าแบบ passing value คือ แทนที่จะสำเนาข้อมูลในตัวแปรที่ใช้เป็นตัวพารามิเตอร์ กลับถือว่าพารามิเตอร์นั้นเป็นตัวแปรเดียวกับตัวแปรพารามิเตอร์ที่ใช้ผ่านค่า เพียงแต่เป็นคนละชื่อกันเท่านั้น ไม่ได้มีการสร้างตัวแปรขึ้นมาใหม่

โปรแกรม 4.5 จะนำโปรแกรมที่ 4.4 มาปรับปรุงโดยให้การผ่านค่าไปยังฟังก์ชัน swap เป็นแบบอ้างอิง

```
1: // Program 4.5 Demonstrate passing by reference
2: // Feb 16, 2004
3: #include <iostream>
4: using namespace std;
5:
6: void swap(int&, int&);
7:
8: main() {
9:     int x ,y;
10:        x = 2;
11:        y = 7;
12:        cout << " x = " << x << " and y = " << y << endl;
13:        swap(x,y);
14:        cout << "After swap ..." << endl;
15:        cout << " x = " << x << " and y = " << y << endl;
16:        return 0;
17: }
18: void swap ( int& a, int& b) {
19:     int temp;
20:     temp = a;
21:     a = b;
22:     b = temp;
23:     cout << "In function swap x = " << a << " and y = " << b << endl;
24: }
25:
```



```
x = 2 and y = 7
In function swap x = 7 and y = 2
After swap ...
x = 7 and y = 2
```

ในการประกาศรูปแบบฟังก์ชัน swap มีสิ่งที่เพิ่มเติมจากเดิมดังนี้

`void swap (int &, int & ) ;`

เครื่องหมาย & (reference operator) จะอยู่ต่อท้ายชนิดของข้อมูล ในที่นี้คือ int เป็นสัญลักษณ์บอกว่าเป็นการผ่านค่าตัวแปรแบบอ้างอิง

int &a หมายถึง a จะเป็นตัวใช้อ้างอิงตัวแปรชนิด int ที่ถูกผ่านค่ามา (a is a reference to the int variable passed to it) ในทำนองเดียวกัน int &b b จะเป็นชื่อที่ใช้เป็นตัวอ้างอิงตัวแปรชนิด int ตัวที่ 2 ที่ถูกผ่านค่ามา a และ b จะมีตำแหน่งในหน่วยความจำเดียวกันกับ x, y ที่ผ่านค่ามา อาจกล่าวได้ว่า a, b คือตัวแปร x, y ที่ถูกเปลี่ยนชื่อ

สังเกตเห็นได้อย่างหนึ่งว่า เราไม่ได้ใช้เครื่องหมาย & ในการเรียกใช้ฟังก์ชัน swap (x, y) ในบรรทัดที่ 13 ดังนั้นการดูประโยคคำสั่งที่เรียกใช้ฟังก์ชัน จึงไม่สามารถตัดสินได้เลยว่า การผ่านค่าพารามิเตอร์ เป็นแบบส่งค่าหรืออ้างอิงตำแหน่ง

การผ่านค่าแบบอ้างอิงตำแหน่ง สามารถเขียนเป็น

int &x, int &y; หรือ

int &x, int &y; ก็ได้

พารามิเตอร์ที่ใช้ในการผ่านค่า ต้องเป็นตัวแปรเท่านั้น ไม่สามารถใช้ค่าคงที่หรือสมการ หรือฟังก์ชันในการผ่านค่าได้

ตัวอย่างต่อไปนี้จะอาศัยประโยชน์จากการผ่านค่าแบบอ้างอิง โดยไม่ต้องอาศัยคำสั่ง return

```
1: // Program 4.6 Convert inch to centimeter
2: // Feb 16, 2004
3: #include <iostream>
4: using namespace std;
5:
6: void convert_inch2centimeter (float &);
7:
8: main() {
9:     float length;
10:    cout << " Input length in inch : ";
11:    cin >> length;
12:    convert_inch2centimeter(length);
13:    cout << "In centimeter = " << length << endl;
14:
15: }
16:
17:
18: void convert_inch2centimeter (float & length) {
19:     length = length *2.54;
20:
21: }
```

```
Input length in inch : 12
In centimeter = 30.48
```

เปรียบเทียบระหว่างการผ่านค่าแบบ passing by value กับ passing by reference จะได้ดังนี้

Passing by value	passing by reference
การประกาศฟังก์ชัน float x; myfunction (float x);	float &x; myfunction (float &x);
x เป็นตัวแปรท้องถิ่น	x เป็นตัวแปรท้องถิ่นเช่นกัน
ฟังก์ชันที่ถูกเรียกจะคัดลอกค่าจากพารามิเตอร์ที่ใช้ผ่านค่า	ฟังก์ชันที่ถูกเรียกจะมีตัวแปรเป็นตัวเดียวกับพารามิเตอร์ที่ใช้ในการผ่านค่า เพียงแต่ชื่อตัวแปรต่างกัน
พารามิเตอร์ที่ใช้ในการผ่านค่าไม่มีการเปลี่ยนแปลงค่า (read only)	สามารถเปลี่ยนแปลงค่าที่เก็บไว้ในพารามิเตอร์ที่ใช้ผ่านค่า (read write)
พารามิเตอร์ที่ใช้ผ่านค่าเป็นได้ทั้งค่าคงที่ ตัวแปร สมการ	ใช้ตัวแปรเป็นพารามิเตอร์ในการผ่านค่าเท่านั้น



โปรแกรม 4.7 เป็นการนำโปรแกรม 4.1 มาปรับปรุงโดยใช้การผ่านค่าแบบอ้างอิง จำนวนบรรทัดของโปรแกรมจะสั้นลง

```
1: // Program 4.7 Circle2.cpp
2: // Feb16, 2004
3: #include <iostream>
4: using namespace std;
5:
6: void calculate_circle( float &, float &, float );
7:
8:
9: int main() {
10: float radius, area, circumference;
11: float r;
12:     cout << "Input radius : ";
13:     cin >> radius;
14:     calculate_circle(area, circumference, radius);
15:     cout << "Area = " << area << endl ;
16:     cout << "Circumference = " << circumference << endl;
17:
18:     return 0;
19: }
20:
21: void calculate_circle(float &a, float &c, float r) {
22:     const float PI = 3.141592653;
23:     a = PI*r*r;
24:     c = 2*PI*r;
25:
26: }
```

ตัวอย่างต่อไปนี้เป็น การตรวจสอบการผ่านค่าแบบอ้างอิง ทดลองพิมพ์ ค่าก่อนส่งผ่านค่า และระหว่างที่อยู่ใน

ฟังก์ชัน และหลังจากออกจากฟังก์ชัน

```
1: // Program 4.8 testRef.cpp
2: // Feb16, 2004
3: #include <iostream>
4: using namespace std;
5:
6: void change_value(float &);
7:
8:
9: int main() {
10: float x=50.8;
11: cout << "Before calling function : " << endl;
12: cout << " x = " << x << endl;
13: cout << "&x = " << &x << endl;
14: change_value( x ) ; // passing by reference
15: cout << "After calling function : " << endl;
16: cout << " x = " << x << endl;
17: return 0;
18: }
19:
20: void change_value( float &a) {
21:     cout << "in called function .... " << endl;
22:     cout << " a = " << a << endl;
23:     cout << "&a = " << &a << endl;
24:     a = 21.2;
25:     cout << "Now a = " << a << endl;
```

```
26:  }
27:
```

```
Before calling function :
x = 50.8
&x = 0012FF88
in called function ....
a = 50.8
&a = 0012FF88
Now a = 21.2
After calling function :
x = 21.2
```

ตำแหน่ง address เดียวกัน

การผ่านค่าแบบอ้างอิง มีความเสี่ยงตรงที่ตัวแปรที่ใช้เป็นตัวพารามิเตอร์ถูกเปลี่ยนแปลงค่าได้ ในภาษา C++ จึงมีทางเลือกที่ 3 ให้ คือ ผ่าน ค่า by constant reference เป็นการผ่านค่าเหมือนกับแบบอ้างอิงปกติทั่วไป แต่ฟังก์ชันจะมีการป้องกันไม่ให้ตัวแปรเปลี่ยนค่าระหว่างที่ฟังก์ชันทำงาน

โปรแกรม 4.9 เป็นตัวอย่างการผ่านค่าแบบ constant reference

```
1:          // Program 4.9 test constant reference
2:          // Feb16, 2004
3:          #include <iostream>
4:          using namespace std;
5:
6:          void test_const_ref( int, int &, const int &);
7:
8:          int main() {
9:              int x,y,z;
10:             x = 1; y =2; z = 3;
11:             cout << "x = " << x << "   y = " << y << "   z = " << z <<
endl;
12:             test_const_ref(x,y, z);
13:             cout << "x = " << x << "   y = " << y << "   z = " << z <<
endl;
14:
15:             return 0;
16:         }
17:
18:         void test_const_ref( int p, int &q, const int &r)
19:         {
20:             p=2*r;
21:             q = r+5;
22:             //      r= q-3;
23:             cout << "p = " << p << "   q = " << q << "   r = " << r <<
endl;
24:
25:         }
26:
```

```
x = 1      y = 2      z = 3
p = 6      q = 8      r = 3
x = 1      y = 8      z = 3
```

ถ้าทดลองเปลี่ยนค่า z โดยลบ comment ในบรรทัดที่ 22 ออก จะคอมไพล์ไม่ผ่าน มีการแจ้งข้อความผิดพลาด การผ่านค่าโดยวิธี constant reference นิยมใช้ผ่านค่า object ที่มีขนาดใหญ่ เช่น array

inline function

ฟังก์ชันช่วยให้การเขียนโปรแกรมมีขนาดกะทัดรัดและอ่านง่าย เมื่อผ่านการคอมไพล์และถูกโหลดเข้าไปในเครื่องแล้ว ไม่สิ้นเปลืองหน่วยความจำ เพราะส่วน binary code ของฟังก์ชันจะอยู่ในหน่วยความจำเพียงที่เดียว ส่วนอื่น ๆ ของโปรแกรมที่มีการเรียกใช้ฟังก์ชันนี้ ไม่ว่าจะกี่ครั้งก็ตาม คอมไพเลอร์จะกำหนดให้กระโดดไปทำงานยังตำแหน่งของหน่วยความจำที่ฟังก์ชันนั้นอยู่ทันที เมื่อทำงานเสร็จจะกระโดดกลับไปยังตำแหน่งที่อยู่หลังคำสั่งที่เรียกใช้ฟังก์ชันนี้

ขณะที่ได้ประโยชน์จากการประหยัดหน่วยความจำในการเก็บชุดคำสั่งของฟังก์ชันนี้ บางครั้งต้องสูญเสียเวลาและหน่วยความจำบางส่วนในขณะที่มีการเรียกใช้ฟังก์ชัน ถ้าผ่านค่าแบบ by value จะต้องคัดลอกค่าของพารามิเตอร์ต่าง ๆ เก็บไว้ในตัวแปรท้องถิ่น ต้องเก็บตำแหน่งหน่วยความจำของคำสั่งที่เรียกใช้ฟังก์ชันเพื่อที่จะได้กระโดดกลับมาได้ถูก เมื่อฟังก์ชันทำงานเสร็จแล้ว จะต้องใช้เวลาลบตัวแปรต่าง ๆ ที่สร้างไว้ และส่งค่าคืนกลับไปยังตัวแปรของโปรแกรมหลัก (ถ้าฟังก์ชันมีการส่งค่ากลับคืน)

เพื่อลดเวลาการทำงานของโปรแกรม ถ้าฟังก์ชันนั้นมีขนาดสั้น และถูกเรียกใช้บ่อย ๆ จึงเลือกที่จะให้ฟังก์ชันนั้นฝังตัวอยู่ในโปรแกรมตรงส่วนที่มีการเรียกใช้ทันที ยอมเสียเนื้อที่หน่วยความจำเพื่อแลกกับเวลาการทำงานที่เร็วขึ้น

การบอกให้คอมไพเลอร์นำชุดคำสั่งของฟังก์ชันไปใส่แทนตรงตำแหน่งที่มีการเรียกใช้ฟังก์ชัน ทำได้โดยเพิ่มคำว่า inline ไว้ข้างหน้าฟังก์ชันนั้น

```
1: // Program 4.10 test inline function
2: // Feb16, 2004
3: #include <iostream>
4: using namespace std;
5:
6: inline float mile2km (float mile) {
7:     return (8.0*mile/5.0);
8: }
9:
10:
11: int main() {
12:     float m;
13:     cout << "Enter Distance in mile : " ;
14:     cin >> m;
15:     cout << "Distance in kilometer = " << mile2km(m) ;
16: }
17:
```

มีข้อปด็กย่อยเกี่ยวกับการใช้ inline function บางประการ คือ คอมไพเลอร์จะต้องมองเห็นตัวฟังก์ชันทั้งหมด ก่อนที่จะมีการเรียกใช้ฟังก์ชัน (ไม่ใช่เฉพาะการประกาศฟังก์ชัน) เพื่อที่จะได้แทรกชุดคำสั่งลงในโปรแกรมได้ถูกต้องตรงตำแหน่งที่เรียกใช้ จากตัวอย่างจะเห็นว่า ฟังก์ชัน mile2km จึงถูกเขียนไว้ก่อน main () ดังนั้นการประกาศฟังก์ชันก่อนเรียกใช้จึงไม่จำเป็น

เมื่อผ่านการคอมไพล์แล้ว ภายในโปรแกรม main () จะถูกแทนที่ด้วยชุดคำสั่งที่อยู่ในฟังก์ชัน mile2km ดังนี้

```

int main() {
    float m;
    cout << "Enter Distance in mile : " ;
    cin >> m;
    cout << "Distance in kilometer = " << 8.0*mile/5.0;
}

```

ฟังก์ชันเรียกใช้ตัวมันเอง (Recursion)

คือการที่ฟังก์ชันหนึ่ง ๆ เรียกใช้ตัวมันเอง มีอยู่ 2 แบบ คือ เรียกใช้ตัวมันเองโดยตรง (direct) และโดยอ้อม คือการที่ฟังก์ชัน A เรียกใช้ฟังก์ชัน B และ ในฟังก์ชัน B นั้นก็มีการเรียกใช้ฟังก์ชัน A

การใช้วิธีเรียกใช้ตัวมันเองแก้ปัญหาบางปัญหา ทำให้โปรแกรมง่ายขึ้นและมีขนาดกะทัดรัด มักใช้กับปัญหาที่กระทำกับข้อมูลแต่ละตัว ในลักษณะที่มีวิธีการซ้ำ ๆ กัน การออกแบบฟังก์ชันแบบรีเคอร์ชัน จะได้ผลลัพธ์จากการประมวลผลตามความต้องการ แต่ก็เสี่ยงกับการวนรอบแบบไม่รู้จบและเกิด runtime error

เมื่อฟังก์ชันเรียกใช้ตัวมันเอง โปรแกรมจะสร้างฟังก์ชันนั้นขึ้นในหน่วยความจำอีกชุด ตัวแปรท้องถิ่นของฟังก์ชันอันหลังจะเป็นคนละตัวกับตัวแปรของฟังก์ชันอันแรก ดังนั้นค่าที่เก็บไว้ในตัวแปรแต่ละชุดของฟังก์ชันจึงไม่กระทบกระเทือนต่อกัน

ตัวอย่างปัญหาที่ใช้วิธี recursion ในการแก้ปัญหาได้แก่เลขอนุกรมของฟีโบนัชชี (Fibonacci series) ประกอบด้วยเลขจำนวนเต็มต่อไปนี้

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, .....

เลขสองตัวค่าแรกจะเป็น 0 และ 1 เลขถัดจากนี้ไปจะเกิดจากการนำผลบวกของเลขสองจำนวนที่อยู่ก่อนหน้ารวมกัน หรือเลขตัวที่ n เกิดจากผลรวมของเลขเทอมที่ n-2 และ n-1 โดยที่ n มากกว่า 2

ฟังก์ชันเรียกตัวมันเองต้องมีเงื่อนไขในการให้การเรียกตัวมันเองหยุดลง เช่นในเรื่องอนุกรมฟีโบนัชชี ฟังก์ชันจะหยุดการเรียกตัวมันเองเมื่อ  $n > 2$

ขั้นตอนการทำงานจะเป็นดังนี้

- ต้องการเลขฟีโบนัชชีที่ตำแหน่ง n
- เรียกใช้ฟังก์ชัน fibonacci (n)
- ฟังก์ชัน fibonacci (n) จะตรวจสอบอาทิวเม้นต์ n

ถ้า  $n < 3$  จะคืนค่ากลับเท่ากับ 1

ถ้า  $n > 3$  จะเรียกใช้ฟังก์ชัน fibonacci โดยผ่านค่า n-2 และเรียกตัวมันเองอีกครั้ง โดยผ่านค่า n-1 แล้วคืนค่าที่เป็นผลบวก

- ถ้าเรียกใช้ฟังก์ชัน fibonacci (0) จะส่งค่า 0 คืนกลับ

ถ้าเรียกใช้ฟังก์ชัน fibonacci (1) จะส่งค่า 1 คืนกลับ

ถ้าเรียกใช้ฟังก์ชัน fibonacci (2) จะส่งค่า 1 คืนกลับ

ถ้าเรียกใช้ฟังก์ชัน fibonacci (3)จะส่งค่า fibonacci (2) และ fibonacci (1) คืนกลับ เพราะ fibonacci (2) ส่งค่ากลับมาเป็น 1 และ fibonacci (2) ส่งค่าคืนกลับมาเป็น 1 ดังนั้น fibonacci (3)จะส่งค่ากลับมาเป็น 2

- ถ้าเรียกใช้ฟังก์ชัน fibonacci (4) จะส่งค่าผลบวกจากการเรียกใช้ fibonacci (3) และ fibonacci (2) เพราะ fibonacci (3) ส่งค่าคืนกลับมาเป็น 2 โดยการเรียกใช้ fibonacci (2) และ fibonacci (1) fibonacci (2) ส่งค่าคืนกลับมาเป็น 1 ดังนั้น fibonacci (4) จะได้ผลลัพธ์เป็น 3

- ถ้าเรียกใช้ฟังก์ชัน fibonacci (5) ฟังก์ชันจะส่งค่าผลบวกระหว่าง fibonacci (4) และ fibonacci (3) เพราะ fibonacci (4) ส่งค่าคืนกลับมาเป็น 3 และ fibonacci (3) ส่งค่าคืนกลับมาเป็น 2 ผลรวมที่ fibonacci (5) ส่งกลับมาคือ 5

ขั้นตอนการหาเลขฟีโบนัชชีดังกล่าวไม่มีประสิทธิภาพนัก เพราะเมื่อเราหา fibonacci (20) จะต้องมีการเรียกใช้ตัวมันเองถึง 13,529 ครั้ง หรือ fibonacci (n) จำนวนครั้งที่เรียกใช้ตัวมันเองคือ  $2^n$  ในการเรียกใช้ตัวมันเองแต่ละครั้งจะต้องสูญเสียหน่วยความจำในการคัดลอกฟังก์ชัน เมื่อมีคำสั่ง return จึงจะปลดปล่อยหน่วยความจำส่วนนั้น

```
1: // Program 4.11 - demonstrates recursion
2: // Calculates the nth Fibonacci number
3: // Uses this algorithm: fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)
4: // Stop conditions: n < 3
5:
6: #include <iostream>
7: using namespace std;
8:
9: long fibonacci(long n);
10:
11: int main()
12: {
13:
14:     long n, result;
15:     cout << "Enter number to find: ";
16:     cin >> n;
17:
18:     cout << "\n\n";
19:     result = fibonacci(n);
20:
21:     cout << result << " is the " << n << "th Fibonacci number\n";
22:     return 0;
23: }
24:
25: long fibonacci (long n)
26: {
27:     cout << "Processing fibonacci(" << n << ")... ";
28:
29:     if (n < 3 )
30:     {
31:         cout << "Return 1!\n";
32:         return (1);
33:     }
34:     else
35:     {
36:         cout << "Call fibonacci(" << n-2 << ") and fibonacci(" << n-1 << ").\n";
37:         return( fibonacci(n-2) + fibonacci(n-1));
38:     }
39: }
40:
41:
```

เช่นต้องการหาเลขฟีโบนัชชี เทอมที่ 35 หาได้ดังนี้

```
Enter number to find: 35
9227465 is the 35th Fibonacci number
```

การหาแฟคทอเรียลของจำนวนเต็มบวก  $n$  ใด ๆ เป็นตัวอย่างการแก้ปัญหาแบบเรียกใช้ตัวมันเองที่ดีอีกตัวอย่างหนึ่ง เขียนเป็นสัญลักษณ์ทางคณิตศาสตร์ได้เป็น  $n!$  อ่านว่า  $n$  factorial หาได้จาก

$$n! = n(n-1)(n-2)(n-3)\dots 1$$

โดยที่  $1! = 1$  และ  $0! = 1$

$$6! = (6)(5)(4)(3)(2)(1)$$

เราสามารถใช่วิธีการวนรอบแบบธรรมดาคำนวณหาค่า  $n!$  ได้ดังนี้

```
factorial = 1;
for (int i = n; i >= 1; i--)
    factorial = factorial * i;
return factorial;
```

เมื่อใช้วิธีแก้ปัญหาโดยใช้ฟังก์ชันเรียกตัวมันเอง โปรแกรมที่สมบูรณ์จะเป็นดังนี้

```
1: // Program 4.12 - demonstrates recursion
2: // Calculates n factorial
3:
4: #include <iostream>
5: using namespace std;
6:
7: unsigned long factorial (unsigned long n);
8:
9: int main()
10: {
11:
12:     unsigned long number;
13:
14:     cout << "Enter positive integer : ";
15:     cin >> number;
16:
17:     cout << "\n\n" << number << "! = " << factorial (number) << endl;
18:
19:     return 0;
20: }
21:
22: unsigned long factorial (unsigned long n)
23: {
24:
25:     if (n <= 1 ) return 1;
26:     else
27:         return n*factorial(n-1);
28: }
29:
```

ทดลองหา 33! จะได้ผลลัพธ์ดังภาพ

```
Enter positive integer : 33
```

```
33! = 2147483648
```

ฟังก์ชัน factorial รับค่าพารามิเตอร์ชนิด unsigned long และคืนค่ากลับเป็นข้อมูลชนิด unsigned long เหมือนกัน ข้อมูลชนิด unsigned long จะเก็บข้อมูลโดยใช้หน่วยความจำ 4 byte สามารถรับค่าที่ได้จากการคำนวณแฟคทอเรียลได้ถึง 4,294,967,295 แต่ค่าแฟคทอเรียลที่ได้จะเพิ่มค่าอย่างรวดเร็ว n มีค่าเท่าใดจึงจะไม่เกินค่าที่เก็บได้ในหน่วยความจำนี้

การเขียนฟังก์ชันเรียกตัวมันเอง โดยไม่มีการคืนค่ากลับ หรือไม่มีคำสั่ง return จะถูกคอมไพเลอร์แจ้งเตือนขณะที่ทำการคอมไพล์ ถ้าในฟังก์ชันนั้นขาดเงื่อนไขที่จะให้ฟังก์ชันหยุดการเรียกตนเอง จะทำให้เกิดกรณีเรียกตนเองไม่สิ้นสุด สิ้นเปลืองหน่วยความจำ และให้ผลลัพธ์จากการทำงานที่มีอากาตเดาได้

ฟังก์ชันที่มีการตั้งค่าปริยายให้อากิวเมนต์

(Function with default argument)

การเรียกใช้ฟังก์ชัน จะต้องผ่านค่าที่ต้องการให้กับพารามิเตอร์แก่ฟังก์ชันที่ถูกเรียก โปรแกรมเมอร์สามารถจะกำหนดค่าอากิวเมนต์ให้มีค่าตามที่ตั้งไว้ได้ เมื่อเรียกใช้ฟังก์ชันโดยไม่มีการผ่านค่าพารามิเตอร์ใด ๆ ค่าที่กำหนดไว้จะถูกนำมาใช้งานทันที

ค่าอากิวเมนต์ที่มีการตั้งค่าที่กำหนดไว้ จะต้องเป็นพารามิเตอร์ที่อยู่ขวามือสุดของฟังก์ชันนั้น ถ้ามีหลายตัว ต้องเรียงลำดับไว้ขวามือสุด ค่าที่ถูกตั้งไว้โดยปริยายนั้นสามารถเป็นได้ทั้งตัวคงที่ ตัวแปรแบบโกลบ บอลหรือฟังก์ชัน การตั้งค่า default นี้ สามารถทำได้ใน inline function ด้วย

```
1: // Program 4.13 How to use default argument
2: // Feb16, 2004
3: #include <iostream>
4: using namespace std;
5:
6: float volumeOfBox ( float length=1.0, float width=1.0, float height=1.0);
7:
8: int main() {
9:     // ommiting all parameters
10:    cout << "The volume of this box (in default size) = " << volumeOfBox() << endl ;
11:    // ommiting the 2nd and 3rd parameter
12:    cout << "The volume of this box with length =5.0 = " << volumeOfBox(5) << endl;
13:    // ommiting the 3rd parameter
14:    cout << "The volume of this box with length=5, width = 3.0 = " <<
        volumeOfBox(5, 3) << endl;
15:    // Overriding all the default parameter
16:    cout << "The volume of this box with length=5.0, width=3.0, height=2.0 = " <<
        volumeOfBox(5,3,2) << endl;
17:    return 0;
18:
19: }
20: float volumeOfBox (float l, float w, float h)
21: {
22:     return (l*w*h);
23: }
```

24:

```
The volume of this box <in default size> = 1
The volume of this box with length =5.0 = 5
The volume of this box with length=5, width = 3.0 = 15
The volume of this box with length=5.0, width=3.0, height=2.0 = 30
```

การส่งผ่านค่าพารามิเตอร์ต่อไปนี้ ไม่ถูกต้อง

volumeOfBox ( , 3, 2) เมื่อจะละเว้นพารามิเตอร์ตัวแรก จะต้องละเว้นตัวที่ 2 และ 3 ด้วย

volumeOfBox ( 5, , 2) เมื่อละเว้นพารามิเตอร์ตัวที่ 2 จะต้องละเว้นค่าพารามิเตอร์ทุกตัวที่ตามหลังตัวที่ 2

การประกาศฟังก์ชันที่มีค่า default ต่อไปนี้ ไม่ถูกต้อง

ตัวอย่าง void myfunc ( int a=2, int b , int c =3);

แก้ไขโดยตำพารามิเตอร์ที่มีค่าตั้งไว้ ไปไว้ทางขวามือ

void myfunc (int b , int a=2, int c =3);

ตัวอย่าง int func(int a, int b = 0, int c);

แก้ไขโดยเปลี่ยนลำดับพารามิเตอร์ใหม่ดังนี้

int func(int a, int c , int b = 0);

ตัวอย่าง void anotherFunc( int , int =2 , int = 3);

การประกาศในตัวอย่างนี้ ทำได้ถูกต้อง

### Function Overloading

ในภาษา C++ ยอมให้เราเขียนฟังก์ชันที่มีชื่อเหมือนกันแต่ชนิดข้อมูลและจำนวนพารามิเตอร์แตกต่างกัน ความสามารถที่ยอมให้ฟังก์ชันมีชื่อซ้ำกันได้นี้ เรียกว่า overloading เมื่อมีการเรียกใช้ overloaded function คอมไพเลอร์จะตรวจสอบจำนวน ชนิด และลำดับของอาร์กิวเมนต์ที่ถูกส่งผ่านในพารามิเตอร์ และจะเลือกฟังก์ชันที่มีชื่อเดียวกันนั้นให้เหมาะสมกับอาร์กิวเมนต์นั้น ฟังก์ชันโอเวอร์โหลดติง จะใช้ในการสร้างฟังก์ชันที่ทำหน้าที่เดียวกัน แต่มีชนิดข้อมูลต่างกัน ดังตัวอย่างต่อไปนี้

```
1: // Program 4.14 Overloading Function
2: #include <iostream>
3: using namespace std;
4:
5: int multiply ( int num1, int num2) { return (num1*num2); }
6: double multiply ( double num1, double num2) { return ( num1*num2); }
7: short multiply ( short num1, short num2) { return (num1*num2); }
8:
9: int main() {
10: int a,b;
11: short c, d;
12: double x,y;
13: a = 6; b=8;
14: cout << "Integer : " << a << " x " << b << " = " << multiply(a,b) << endl;
15: c = 4; d = 5;
16: cout << "Short : " << c << " x " << d << " = " << multiply(c,d) << endl;
17: x = 3.2 ; y = 5.4;
```



```

18: cout << "double : " << x << " x " << y << " = " << multiply(x,y) << endl;
19:
20: return 0;
21:
22: }

```

การใช้ overloaded function ทำให้เราประหยัดการตั้งชื่อฟังก์ชัน ในภาษา C ถ้าชนิดข้อมูลต่างกัน ถึงแม้ฟังก์ชันจะทำงานแบบเดียวกัน ต้องตั้งชื่อต่างกัน เช่น

```

int imultiply ( int num1, int num2);
double dmultiply (double num1, double num2);
short smultiply (short num1, short num2);

```

การสร้างฟังก์ชัน overloading มีข้อพึงระวังดังนี้

1. ไม่สามารถทำ overload ฟังก์ชันที่มีชนิดและจำนวนของตัวพารามิเตอร์เท่ากัน แต่คืนค่าชนิดข้อมูลต่างกัน

```

int max (int a, int b);
unsigned max ( int a , int b);

```

ฟังก์ชัน max มีพารามิเตอร์เหมือนกันทุกประการ แต่ส่งคืนค่าข้อมูลต่างชนิดกัน C++ จะจำแนก overloading function โดยดูจาก argument list ไม่สามารถจำแนกจากชนิดข้อมูลที่คืนค่า

2. ถ้าค่า argument ที่ส่งผ่านมีชนิดข้อมูลไม่สอดคล้องกับที่ได้ประกาศไว้ จะต้อง cast ชนิดข้อมูลให้ตรงกันก่อน เช่น

```

float float_var = 2.4;
int tint_var = 4;
int result;
result = multiply (float_var , int_var);

```

จะขึ้นข้อความผิดพลาดว่าสับสนชนิดข้อมูล ใน argument list ต้องแก้เป็น

```

result = multiply ( (int) float_var , int_var);

```

แต่ถ้าข้อมูลที่ส่งผ่านค่าเป็นชนิดข้อมูลที่ใช้นี้ที่หน่วยความจำในการเก็บน้อยกว่าชนิดข้อมูลของ argument list C++ จะเปลี่ยนชนิดข้อมูลนั้น ให้เป็นชนิดข้อมูลตัวเลข (numeric type) ตามที่ปรากฏใน argument list เช่น

```

int multiply ( int i1, int i2);
double multiply (double d1, double d2);
float f1, f2;
cout << multiply (f1, f2);

```

จะเปลี่ยน ข้อมูลชนิด float ให้เป็นข้อมูลชนิด double

```

long L1, L2;
cout << multiply (L1, L2);

```

ไม่สามารถเปลี่ยนข้อมูลชนิด long ให้เป็นข้อมูลชนิด int ได้ เพราะข้อมูลชนิด long ใช้น้ำที่หน่วยความจำมากกว่าข้อมูลชนิด int

3. Overloaded function ที่มีการตั้งค่าตัวแปรโดยปริยายไว้ อาจจะสับสนกับ overloaded function ที่ผ่านค่าตัวแปรเป็น void ได้เช่น

```
int multiply ( int i1=5, int i2=7);  
  
int multiply (void) {  
    int i1=3 ;  
    int i2=4;  
    return i1*i2;  
}
```

เมื่อเรียกใช้ฟังก์ชัน

```
int result = multiply ( );
```

คอมไพเลอร์จะสับสนไม่สามารถเลือกได้ว่าจะใช้ฟังก์ชันชุดใด

ฟังก์ชันต้นแบบ ( Function template)

ฟังก์ชันโอเวอร์โหลดใช้เมื่อการผ่านค่า argument ที่มีจำนวนและชนิดข้อมูลต่างกัน งานที่ประมวลผลมีลักษณะเหมือนกันหรือมีลักษณะซ้ำกันทุกประการ การใช้ template จะทำให้เขียนโค้ดของฟังก์ชันนั้นเพียงครั้งเดียว แต่ใช้กับตัวแปรที่มีชนิดข้อมูลต่างกันได้ เราสามารถสร้างฟังก์ชันต้นแบบ เพียง 1 ชุด นำไปแก้ปัญหาเดียวกันได้ทั้งหมด ถึงแม้จะเป็นข้อมูลต่างชนิดกัน เป็นการสร้าง generic function ที่ใช้ได้กับข้อมูลครอบคลุมจักรวาล

ฟังก์ชันต้นแบบ จะนำหน้าด้วยคำว่า template ตามด้วยชนิดข้อมูล รูปแบบทั่วไปจะเป็นดังนี้

```
template <class Ttype>  
  
return_type FunctionName ( parameter list) {  
  
    // body function  
  
}
```

เมื่อนำฟังก์ชัน multiply เขียนเป็นฟังก์ชันต้นแบบจะได้ดังนี้

```
1: // Program 4.15 templated function demo  
2: #include <iostream>  
3: using namespace std;  
4:  
5: template <class T> // or template <TypeName>  
6: T multiply ( T num1, T num2)  
7: { return num1*num2;  
8: }  
9:  
10: int main() {  
11:     int a,b;  
12:     short c, d;  
13:     double x,y;  
14:     a = 6; b=8;  
15:     cout <<"Integer : " << a << " x " << b << " = " << multiply(a,b) << endl;  
16:     c = 4; d = 5;  
17:     cout << "Short : " << c << " x " << d << " = " << multiply(c,d) << endl;  
18:     x = 3.2 ; y = 5.4;  
19:     cout << "double : " << x << " x " << y << " = " << multiply(x,y) << endl;
```

```

20:
21:     return 0;
22:
23:}

```

บรรทัดที่ 5 ถึง 8 เป็นการเขียนฟังก์ชันต้นแบบ ชื่อ multiply มีการประกาศข้อมูลชนิด T ซึ่งจะถูกใช้ในฟังก์ชัน และข้อมูลที่ส่งค่าคืนกลับก็เป็นข้อมูลชนิด T เช่นเดียวกัน จะเห็นว่าการใช้ฟังก์ชันต้นแบบจะลดจำนวนบรรทัดการเขียนคำสั่งลงไปได้มาก

ตัวอย่างต่อไปนี้เป็นฟังก์ชันต้นแบบที่ใช้เปรียบเทียบข้อมูล 3 ชุด แล้วหาว่าข้อมูลชุดใดมีค่ามากที่สุด

```

1: // Program 4.16 find them maximum value of 3 data sets
2: #include <iostream>
3: using namespace std;
4:
5: template <class anytype>
6: anytype maximum (anytype a, anytype b, anytype c)
7: {anytype max = a;
8:     if (b > max) max = b;
9:     if ( c > max ) max = c;
10:    return max;
11:}
12:
13:int main() {
14:    int i1, i2, i3;
15:    cout << "Input three Integer number ";
16:    cin >> i1 >> i2 >> i3;
17:    cout << "The Maximum number is " << maximum(i1,i2, i3)<< endl;
18:    double d1, d2, d3;
19:    cout << "Input three real number ";
20:    cin >> d1 >> d2 >> d3;
21:    cout<<"The Maximum real number is " <<maximum(d1,d2, d3)<< endl;
22:
23:    char c1, c2, c3;
24:    cout << "Input three character ";
25:    cin >> c1 >> c2 >> c3;
26:    cout <<"The Maximum character is " <<maximum(c1,c2, c3)<< endl;
27:
28:
29:    return 0;
30:
31:}

```

```

Input three Integer number 90 32 238
The Maximum number is 238
Input three real number 4.78 32.23 0.076
The Maximum real number is 32.23
Input three character x u w
The Maximum character is x

```

## แบบฝึกหัด

1. จงคอมไพล์โปรแกรมที่ใช้งาน overloaded function ต่อไปนี้ แล้วดูผลที่ได้ ถ้ามีข้อผิดพลาดจะต้องแก้ไขอย่างไร

```
1:      // ex01.cpp
2: // from Borland C++ 3.1 -- Object oriented programming page 186
3: #include <iostream>
4: using namespace std;
5:
6: void display (long val) { cout << val << " in function 1 " << endl; } // .....(1)
7: void display (double val) { cout << val << " in function 2" << endl; } //.....(2)
8: void display (float val) { cout << val << " in function 3 " << endl; } //.....(3)
9:
10:
11: main() {
12:     display (123456789);
13:     display (3.141592653);
14:     display (23);
15:     return 0;
16: }
17: }
```

(เมื่อ คอมไพล์ด้วย BCC 5.1 จะแจ้งข้อผิดพลาด ในบรรทัดที่ 12 และ 14 ว่าเป็นการผ่านค่าแบบกำกวม ทำให้ไม่สามารถเลือกใช้ display ที่เป็น long หรือ float

บรรทัดที่ 13 น่าจะเรียกใช้ฟังก์ชัน 3 แต่ค่าคงที่ 3.141592653 จะถูกคอมไพเลอร์มองว่าเป็น double เพราะไม่มีการประกาศตัวแปรว่าเป็น float ดังนั้นจึงเรียกใช้ฟังก์ชันที่ 2)

2. จงเขียนฟังก์ชันที่รับตัวเลขจำนวนเต็ม 3 ค่า ฟังก์ชันจะส่งกลับจำนวนที่มีค่ามากที่สุด

int max\_int ( int i, int j, int k);

3. เขียนโปรแกรมที่ใช้ฟังก์ชันและตรวจสอบ

$$f(x) = \frac{2x^2 - 3x - 2}{x^2 + 4x + 4}$$

โดยกำหนด function prototype ดังนี้

double fx( double x);

4. จงเขียนและตรวจสอบฟังก์ชันต่อไปนี้

$$g(x) = \begin{cases} 0 & x < 0 \\ 3e^{(x-1)} & x \geq 0 \end{cases}$$

โดยกำหนด function prototype ดังนี้

double gx( double x);

5. เขียนและตรวจสอบฟังก์ชันที่หาความยาวเส้นตรงระหว่างจุด  $(x_1, y_1)$  และ จุด  $(x_2, y_2)$

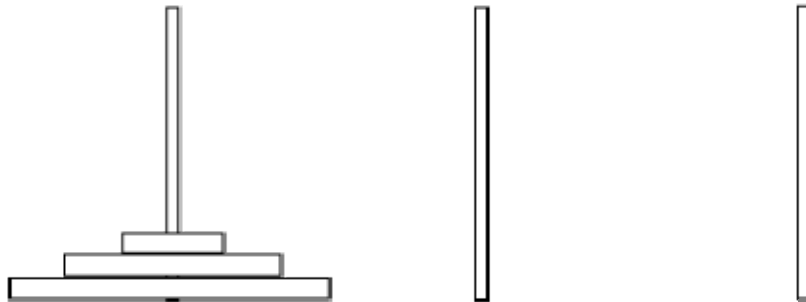
$$\text{length} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

6. จงเขียนและตรวจสอบว่า จุด  $(x,y)$  อยู่ใน quadrant ไດ

int quadrant ( double x, double y);

7. หอคอยแห่งฮานอย เกมนี้ประกอบด้วยเสาหลักจำนวน 3 หลัก ให้เป็นเสา A, B และ C ตามลำดับ ที่เสา A มีแผ่นจานขนาดต่าง ๆ เรียงซ้อนกันโดยให้จานแผ่นใหญ่อยู่ล่างสุดไล่ไปตามลำดับแผ่นเล็กอยู่บนสุดจำนวน N แผ่น (จากรูป  $N = 3$  แผ่น) กติกาการเล่นคือให้เคลื่อนย้ายแผ่นจากจากเสา A ทั้งหมดไปยังเสา C มีข้อบังคับว่า

- การหยิบแต่ละครั้ง จะหยิบแผ่นจานได้เพียง 1 แผ่น
- ห้ามนำจานแผ่นใหญ่วางซ้อนบนจานแผ่นเล็ก
- สามารถใช้เสา B เป็นที่พักจานชั่วคราวได้ แต่ห้ามนำจานแผ่นใหญ่วางซ้อนบนจานแผ่นเล็กเช่นกัน
- 



ถ้าจานมีเพียง 1 แผ่น ( $N = 1$ ) สามารถย้ายจานจากเสา A ไปยังเสา C ได้เลยโดยไม่ต้องใช้เสา B เป็นที่พักจาน (จะใช้เป็นเงื่อนไขในการหยุดการทำงาน และส่งค่าคืนกลับของฟังก์ชัน)

ถ้ามีจานอยู่ 3 แผ่น จะย้ายจานจากเสา A มาไว้ที่เสา C ลำดับการย้ายจะเป็นดังนี้

```

A → C
A → B
C → B
A → C
B → A
B → C
A → C

```

8. ชุดคำสั่งต่อไปนี้ ผิดตรงส่วนใด

```

void main ( ) {
    discount_price ( 500.0 , 10 );
    ....
}

void discount_price (float & price, float & percent_discount) {
    price = price - price*percent_discount/100.0;
}

```

ผิดตรงที่ผ่านค่าคงที่ให้กับการผ่านค่า by reference

9. ต้องการประกาศตัวแปร 3 ตัว เป็นแบบ passing by reference

```

int& a, b, c;

int &a, &b, &c;

int &a, b ,c;

```

บรรทัดแรก ถูกเพียงบรรทัดเดียว บรรทัดที่ 2, 3 ไม่ถูกต้อง

10. (Going from C to C++) พิจารณาคำสั่งต่อไปนี้ มีข้อผิดพลาดตรงไหนบ้าง

```
int minus (int i ) {  
    return -i;  
}  
  
long minus (long r) {  
    return -r;  
}  
  
int main ( ) {  
    int a = minus ('A');  
    cout << a;  
  
    long b = minus (271.83);  
}
```

11. การประกาศ Overloaded function ถูกต้องหรือไม่

```
int add ( int i, int j);  
int add ( int i, int j, int k);
```

(ถูกต้อง)

12. การประกาศ Overloaded function ต่อไปนี้มีข้อบกพร่องหรือไม่

```
int calculate ( int x);  
int calculate ( int x, int y = 50);
```

ปัญหาคือ การเรียกใช้โดยผ่านค่าเพียงตัวเดียวอาจจะเป็นฟังก์ชันที่ 2 โดยละเว้นค่า default ก็ได้

## บทที่ 5 Array, structure และ pointer

### อะเรย์ 1 มิติ

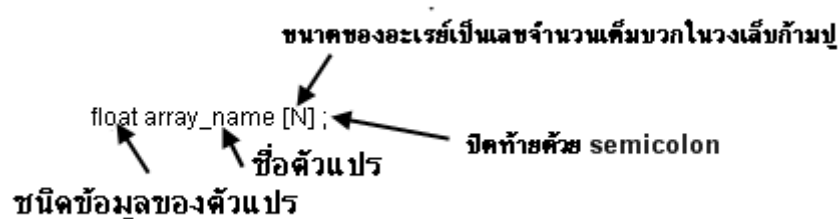
การแก้ปัญหาในทางวิทยาศาสตร์และวิศวกรรม บางครั้งข้อมูลมีเพียง 1 ค่า เช่นการหาพื้นที่วงกลม เราใช้รัศมีของวงกลมในการหาพื้นที่นั้น จุดต่าง ๆ ในระนาบ x, y สามารถใช้ตัวแปรเพียง 2 ตัว คือ x-coordinate และ y-coordinate สำหรับหาจุดพิกัด ปัญหาบางปัญหาต้องใช้ข้อมูลแบบเดียวกันเป็นจำนวนมาก และเราก็ไม่ต้องการตั้งชื่อข้อมูลเหล่านั้นให้แตกต่างกัน เช่น เราวัดอุณหภูมิของน้ำ 10 ครั้งได้ข้อมูล 10 ค่า เราไม่ต้องการที่จะตั้งชื่อข้อมูลเหล่านี้ถึง 10 ชื่อ แต่จะใช้ข้อมูลเพียงชื่อเดียวนี้จัดการข้อมูลทั้ง 10 ค่านี้ได้ โดยจัดเก็บข้อมูลเหล่านี้ในรูปอะเรย์

อุณหภูมิที่เก็บได้จะจัดเก็บในรูปอะเรย์ดังแผนภาพต่อไปนี้

27	27.7	28.1	28.3	28.5	29.4	30	32.2	33.8	34.1
T[0]	T[1]	T[2]	T[3]	T[4]	T[5]	T[6]	T[7]	T[8]	T[9]

ให้ T เป็น identifier ของอะเรย์ จำแนกค่าต่าง ๆ ของอุณหภูมิทั้ง 10 ค่าโดยใช้ subscript หรือ index value ในภาษา C++ subscript จะเริ่มต้นที่ 0 และจะเพิ่มค่าทีละ 1 ข้อมูลอุณหภูมิค่าที่ 1 จะถูกเก็บไว้ใน T [0] อุณหภูมิค่าที่ 2 จะถูกเก็บไว้ใน T [1] ไปเรื่อย ๆ จนถึงอุณหภูมิค่าที่ 10 จะถูกเก็บไว้ใน T[9]

การสร้างตัวแปรแบบอะเรย์ทำได้ดังนี้



ข้อมูลที่เก็บไว้ในตัวแปรอะเรย์ เรียกว่า สมาชิก(elements)ของอะเรย์ ตัวอย่างการนิยามตัวแปรแบบอะเรย์

```
int NumberOfStudentEachYear[12];
```

อะเรย์เก็บข้อมูลชนิดจำนวนเต็มเก็บข้อมูล 12 ค่า

```
double distance[100];
```

 เก็บข้อมูลชนิดจำนวนจริงแบบ double ได้ 100 ค่า

```
char ch[10];
```

 เก็บข้อมูลชนิดตัวอักษร 10 ตัว

เราสามารถกำหนดค่าเริ่มต้นให้อะเรย์ขณะที่นิยามอะเรย์นั้น ทำได้โดยกำหนดค่าสมาชิกแต่ละตัวคั่นด้วยเครื่องหมายจุลภาค ปิดหัวท้ายด้วยวงเล็บปีกกา เช่น

```
int a[5] = {3,5,7,9, 11} ;
```

กำหนดให้ a เป็นตัวแปรแบบอะเรย์ชนิดจำนวนเต็ม ให้ a[0]=3, a[1]=5, a[2]=7, a[3]=9, a[4]=11,

```
float f[ ] = {4.2, 6.8, 3.7 };
```

กรณีที่ไม่ได้ขนาดของอะเรย์ในวงเล็บ ขนาดของอะเรย์จะถูกจองไว้ในหน่วยความจำเท่ากับจำนวนสมาชิกที่กำหนดไว้ตอนเริ่มต้น ในที่นี้คือ 3

```
float ff[10]= { 0};
```

กำหนดให้อะเรย์ ff ทั้ง 10 ตัวมีค่าเป็นศูนย์

การกำหนดค่าให้กับสมาชิกอะเรย์ไม่ครบทุกตัว ตัวที่ไม่มีการกำหนดค่าจะถูกกำหนดโดยคอมพิวเตอร์ให้มีค่าเป็นศูนย์

```
double b[5] = {3.2, 6.8, 4.1 };
```

จะได้ b[0]=3.2, b[1]=6.8, b[2]=4.1, b[3] = 0, b[4] =0

ถ้าประกาศตัวแปรแบบอะเรย์ แต่ไม่มีการกำหนดค่าเริ่มต้น จะได้ข้อมูลที่เป็นขยะเก็บไว้ในอะเรย์นั้น ดังตัวอย่างโปรแกรมต่อไปนี้

```
1: //Array uninitialized
2: #include <iostream>
3: using namespace std;
4:
5: int main() {
6:     double a[5] ;
7:     cout << "Show gabage data in array \n" ;
8:     for(int i = 0 ; i < 5; i++)
9:         cout << "a[" << i << " ] = " << a[i] << endl;
10:    return 0;
11: }
12:
```

```
char vowels[5] = { 'a', 'e', 'i', 'o', 'u'};
```

การกำหนดชนิดข้อมูลแบบตัวอักษร จะได้ vowels[0] = 'a', vowels[1]='e', vowels[2]= 'i', vowels[3] = 'o', vowels[4]='u' หรือกำหนดเป็น

```
char vowels[5] = { "aeiou" };
```

```
char str[ ] = "This is a sentence";
```

ไม่สามารถใช้ตัวแปรในการกำหนดขนาดของอะเรย์ได้ ตัวอย่างต่อไปนี้จึงไม่ถูกต้อง

```
int n = 20;
```

```
int myArray[n];
```

แต่เราสามารถใส่ค่าคงที่ไปกำหนดขนาดของอะเรย์ได้ เมื่อต้องการเปลี่ยนขนาดของอะเรย์ สามารถเปลี่ยนตัวเลขที่ ArraySize เพียงแห่งเดียว ไม่ต้องไปตามแก้หลายแห่ง ถ้ามีการสร้างอะเรย์ไว้หลายที่

```
#include <iostream>
using namespace std;
const int ArraySize=100;
int main ( ) {
    int testArray[ArraySize];
    int i;
    for ( i=0; i < ArraySize; i++) {
        testArray[i] = i;
        cout << "testArray[ " << i << " ] = " << testArray[i] << endl;
    }
}
```



จะต้องพึงระวังว่าเมื่อสร้างตัวแปรอะเรย์ a มีขนาด n อะเรย์ a จะมีค่าตั้งแต่ a[0], a[1], .., a[n-1] เท่านั้น ไม่มี a[n] ถ้าใช้ a[n] จะพบข้อผิดพลาดขณะคอมไพล์โปรแกรม ว่าไม่มีสมาชิกตัวที่ n หรือ out of bound error. คอมไพเลอร์บางตัวไม่มีการตรวจสอบขนาดของอะเรย์และจะนำตัวเลขที่เป็นขยะมาแสดงผล โปรแกรมเมอร์ควรตรวจสอบขนาดของอะเรย์ เพื่อป้องกันการนำข้อมูลขยะที่อยู่นอกขอบเขตของอะเรย์มาใช้งาน

โปรแกรมต่อไปนี้เป็นารแสดงผลข้อมูลที่เก็บไว้ในอะเรย์

```
1: // program 5.1
2: #include <iostream>
3: using namespace std;
4:
5: int main() {
6:
7:     int number[6] = { 2,3,5,7,11,13};
8:     // display number on screen
9:     for (int i = 0; i < 6 ; i++ )
10:         cout << " Number [ " << i << " ] = " << number[i] << endl;
11:     // assign new number to number [i]
12:     for (int i = 0; i < 6 ; i++ )
13:         number[i] = (i+1)*2;
14:     cout << "Now, data in array number[] has been changed...\n";
15:     for (int i = 0; i < 6 ; i++ )
16:         cout << " Number [ " << i << " ] = " << number[i] << endl;
17:     return 0;
18: }
19:
20:
```

อะเรย์ประเภท char จะถูกปิดท้ายด้วย null character ( \0 ) ไว้ท้ายอะเรย์เสมอ เพื่อแสดงว่าสิ้นสุดขอบเขตของอะเรย์เพียงเท่านี้ และจะแสดงผลเพียงเท่านี้ ดังตัวอย่างในโปรแกรม 5.2

mystr เป็นอะเรย์ประเภท char เก็บค่า Test character arrays. ไว้ในหน่วยความจำปิดท้ายด้วย null character ดังรูป

T	e	s	t		c	h	a	r	a	c	t	e	r		a	r	r	a	y	s	.	\0
---	---	---	---	--	---	---	---	---	---	---	---	---	---	--	---	---	---	---	---	---	---	----

เมื่อกำหนด mystr[4] = '\0'

T	e	s	t	\0	c	h	a	r	a	c	t	e	r		a	r	r	a	y	s	.	\0
---	---	---	---	----	---	---	---	---	---	---	---	---	---	--	---	---	---	---	---	---	---	----

เมื่อให้แสดงผล mystr พบว่าข้อความที่เหลือ จะได้ Test เท่านั้น ข้อความหลัง '\0' จะถูกละทิ้งไม่แสดงผลทั้งหมด

```
1: //Program 5.2 Char array
2: #include <iostream>
3: using namespace std;
4:
5: int main() {
```

```

6:   char mystr[] = "Test character arrays.";
7:       cout << mystr << endl; //
8:       mystr[4] = '\0';
9:       cout << mystr<< endl;
10:      return 0;
11:  }
12:

```

ผลที่ได้จากการรันโปรแกรม

```

Test character arrays.
Test

```

เราสามารถใช้คำสั่ง `mystr[4] = 0;` แทน `mystr[4] = '\0';` ในบรรทัดที่ 8 ได้ การใช้เครื่องหมาย \ (backslash)

บอกให้คอมพิวเตอร์รู้ว่ามีตัวอักษรพิเศษตามหลัง \

ถ้าต้องการกำหนดค่า เลขศูนย์เก็บไว้ในอะเรย์จะต้องใช้คำสั่งดังนี้

```
mystr[4] = '0';
```

หรือ `mystr[4] = 48;` ซึ่งเป็นรหัสอักขระของเลขศูนย์

โปรแกรม 5.3 เป็นการนำตัวแปรแบบอะเรย์มาใช้งาน โปรแกรมนี้จะให้ป้อนเลขจำนวนจริงหรือจำนวนเต็ม ไม่เกิน 100 ค่า แล้วแสดงผลตัวเลขที่ป้อนเข้าไป พร้อมค่าเฉลี่ย

```

1:   // Program5.3
2:   // Terminating input by press Control -z
3:
4:   #include <iostream>
5:   using namespace std;
6:
7:   const int MAXSIZE = 100;
8:
9:   int main()
10:  {
11:      float data[MAXSIZE];
12:      int size = 0;
13:      float currentnumber, average, sum;
14:      cout << "Enter your first number - > (CTR-z to stop Entering) ";
15:      cin >> currentnumber;
16:      while (!cin.eof()) {
17:          data[size] = currentnumber;
18:          size++;
19:          cout << "Enter your next number : ";
20:          cin >> currentnumber;
21:      }
22:      cout << "You've enter " << size << " data " << endl;
23:      sum = 0.0;
24:      for ( int i = 0; i < size ; i++ ) {
25:          cout << data[i] << " " ;
26:          sum = sum + data[i];
27:      }
28:      average = sum / size;
29:      cout << "Average value of these number is " << average << endl;
30:      return 0;

```

```
31:  }
32:
```

```
Enter your first number - > <CTR-z to stop Entering> 2
Enter your next number : 4.5
Enter your next number : 7
Enter your next number : 3
Enter your next number : ^Z
^Z
You've enter 4 data
2 4.5 7 3 Average value of these number is 4.125
```

เพื่อความปลอดภัยอาจตรวจสอบขอบเขตของอะเรย์ โดยแทรกคำสั่งต่อไปนี้หลังบรรทัดที่ 22

```
if ( size*sizeof(float) > sizeof(data) return 0;

sum =0.0;

for ( int i=0; int i < size; i++ )

.....
```

การใช้ตัวแปรอะเรย์กับ enumeration type

เราสามารถนำข้อมูลแบบ enumeration type มาใช้ร่วมกับตัวแปรชนิดอะเรย์ได้อย่างกลมกลืน ทำให้โปรแกรมมีลักษณะอ่านง่ายและชัดเจนในตัวเอง

ข้อมูลแบบ enum type จะคล้ายกับข้อมูลประเภท int หรือ char ต่างกันตรงที่สามารถแสดงเป็นชื่อที่บ่งบอกถึงความหมาย ไม่จำเป็นต้องเรียงลำดับ ขึ้นอยู่กับโปรแกรมเมอร์จะกำหนด

โปรแกรมต่อไปนี้แสดงความยาวคลื่นของแสงสีต่าง ๆ ในหน่วย นาโนเมตร

```
1:  #include <iostream>
2:  using namespace std;
3:
4:  enum LightColor { violet, blue, green, yellow, orange, red };
5:  float wavelength[red+1]= {300, 420, 480, 530,600,620 };
6:
7:  int main ()
8:  {
9:      cout << "The wavelength from violet to red --> " << endl;
10:     for (LightColor light = violet; light <= red; light++ )
11:         cout << wavelength[light] << endl;
12:
13:     return 0;
14: }
15:
```

```
The wavelength from violet to red -->
300
420
480
530
600
620
```

ในบรรทัดที่ 10, 11, 12 จะเห็นว่าการใช้ enum type จะช่วยให้การเขียนคำสั่งมีลักษณะใกล้เคียงกับภาษาพูด เป็นการบอกให้แสดงค่าความยาวคลื่นของแสงสีต่าง ๆ ตั้งแต่ violet, blue, green, yellow, orange, และ red

อะเรย์หลายมิติ

อะเรย์ที่กล่าวมาจะเป็นอะเรย์เพียงแถวเดียว ถ้าอะเรย์นั้นมีหลายแถวจัดว่าเป็นอะเรย์หลายมิติ การประกาศตัวแปรที่เป็นอะเรย์หลายมิติ ทำได้โดยกำหนดขนาดให้เท่ากับจำนวนของมิตินั้น เช่น

```
int a[3][4];
```

a เป็นตัวแปรอะเรย์เก็บข้อมูลชนิดจำนวนเต็มขนาด 3 แถว 4 สดมภ์ เป็นอะเรย์ 2 มิติ

	a[ ][0]	a[ ][1]	a[ ][2]	a[ ][3]
a[0][ ]				
a[1][ ]				
a[2][ ]				

ในหน่วยความจำจะถูกจัดวางตำแหน่งดังนี้

a[0]		a[0][0]
		a[0][1]
		a[0][2]
		a[0][3]
a[1]		a[1][0]
		a[1][1]
		a[1][2]
		a[1][3]
a[2]		a[2][0]
		a[2][1]
		a[2][2]
		a[2][3]

ตัวอย่างการประกาศค่าตัวแปรที่เป็นอะเรย์ 3 มิติ

```
float farray[6][2][4];
```

เป็นอะเรย์ที่เก็บข้อมูลชนิดจำนวนจริง (float) มีขนาด 6 x 2 x 4

รูปแบบทั่วไปของการประกาศตัวแปรแบบอะเรย์มีดังนี้

```
Data_type ArrayName [size1][size2][...][...];
```

การกำหนดค่าในอะเรย์หลายมิติ มีลักษณะคล้ายกับเมตริกซ์ (Matrix) ในคณิตศาสตร์ เมตริกซ์ b มีขนาด 2 x 3 มีค่าดังนี้

$$b = \begin{bmatrix} 3 & -1 & 0 \\ 2 & 5 & 7 \end{bmatrix}$$

เขียนเป็นอะเรย์ 2 มิติ ในภาษา C++ ได้ดังนี้

```
int b[2][3] = { { 3, -1, 0 },
               { 2, 5, 7 }
             };
```

หรือจะเขียนให้อยู่ในแถวเดียว

```
int b[2][3] = { 3, -1, 0, 2, 5, 7};
```

```
int calendar[12][31];
```

calendar เป็นอะเรย์เก็บข้อมูลชนิดจำนวนเต็มจำนวน 11 แถวแต่ละแถวมีสมาชิก 31 ตัว (ไม่ใช่ 31 แถว แต่ละแถวมีสมาชิก 12 ตัว)

```
sizeof (calendar) = (31)(12)(sizeof (int)) = 372
```

ให้ c เป็นอะเรย์ 3 มิติ มีขนาดเป็น 2 x 4 x 3 มีจำนวนสมาชิก 24 ตัว

การกำหนดค่าเริ่มต้นของอะเรย์ 3 มิติ ทำได้ดังนี้

```
int c [2][4][3] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 };
```

หรือ

```
int c [2][4][3] = { { {0, 1, 2} , {3, 4, 5}, {6, 7, 8}, {9, 10, 11} },
                   { {12, 13, 14}, {15, 16, 17}, {18, 19, 20}, {21, 22, 23} };
```

จะทดลองนำอะเรย์ c ไปแสดงผล

```
1: // Print 3 D arrayProgram5_2
2: #include <iostream.h>
3:
4: int main()
5: {
6:   int c[2][4][3] = { { {0,1,2}, {3,4,5}, {6,7,8}, {9,10,11} },
7:                     { {12,13,14}, {15,16,17}, {18,19,20}, {21,22,23} } };
8:   for (int i = 0; i < 2 ; i++) {
9:     cout << endl;
10:    for(int j = 0; j < 4 ; j++) {
11:      cout << endl;
12:      for(int k = 0; k < 3; k++ ) {
13:        cout << "C[" << i << "]"[" << j << "]"[" << k << "] = "
14:        << c[i][j][k] << " ";
15:      }
16:    }
17:  }
18:  return 0;
19: }
20:
21:
```

```

C[0][0][0] = 0   C[0][0][1] = 1   C[0][0][2] = 2
C[0][1][0] = 3   C[0][1][1] = 4   C[0][1][2] = 5
C[0][2][0] = 6   C[0][2][1] = 7   C[0][2][2] = 8
C[0][3][0] = 9   C[0][3][1] = 10  C[0][3][2] = 11

C[1][0][0] = 12  C[1][0][1] = 13  C[1][0][2] = 14
C[1][1][0] = 15  C[1][1][1] = 16  C[1][1][2] = 17
C[1][2][0] = 18  C[1][2][1] = 19  C[1][2][2] = 20
C[1][3][0] = 21  C[1][3][1] = 22  C[1][3][2] = 23

```

การผ่านค่าตัวแปรอะเรย์ไปยังฟังก์ชัน

การผ่านค่าอะเรย์ไปยังฟังก์ชันอื่นเพื่อประมวลผลเป็นการผ่านค่าแบบ reference ตัวอย่างเช่น

`int myarray[ ]` ; เป็นการประกาศตัวแปรอะเรย์ชื่อ `myarray` สมาชิกของอะเรย์แต่ละตัวจะเป็นแบบ `integer` ชื่อ `myarray` จะเก็บตำแหน่งของหน่วยความจำที่เก็บอะเรย์ คอมไพเลอร์จะรับรู้เพียงตำแหน่งของหน่วยความจำ และชนิดข้อมูลที่เก็บ ไม่จำเป็นต้องระบุจำนวนสมาชิกอะเรย์

เมื่อผ่านค่าอะเรย์ไปยังฟังก์ชัน จะเป็นการผ่านตำแหน่งของหน่วยความจำที่เก็บตัวแปรอะเรย์ตัวแรกเท่านั้น ฟังก์ชันสามารถเปลี่ยนแปลงค่าที่เก็บไว้ในอะเรย์โดยการระบุตำแหน่งหน่วยความจำที่เหมาะสม การผ่านค่าตัวแปรอะเรย์ไปยังฟังก์ชัน ควรระบุขนาดของอะเรย์ไปยังฟังก์ชันที่ถูกเรียก เพื่อป้องกันการประมวลผลอะเรย์ที่อยู่นอกเหนือขอบเขต

ตัวอย่างต่อไปนี้เป็นการผ่านค่าอะเรย์แบบ 1 มิติ ไปยังฟังก์ชัน เป็นการขยายโปรแกรมการหาค่าเฉลี่ย โดยตกแต่งงานออกเป็น 3 ฟังก์ชันนั่นเอง

```

1:  // program - ArrayAverage.cpp
2:  #include <iostream>
3:  using namespace std;
4:
5:  const int MAXSIZE = 100;
6:  void ArrayInput( double [], int & );
7:  void ArrayDisplay( double[], int );
8:  double Average (double[], int);
9:
10: int main()
11: {
12:     double data[MAXSIZE];
13:     int n;
14:     double mean;
15:     ArrayInput(data,n);
16:     cout << "Array Name \"DATA\" has " << n << " elements." << endl;
17:     ArrayDisplay(data,n);
18:     mean = Average(data,n);
19:     cout << "Average of these value = " << mean;
20:     return 0;
21: }
22: void ArrayInput( double a[], int &num)
23: {
24:     num = 0;
25:     cout << "Input data (type -99 for stop)-->";
26:     for( num = 0; num < MAXSIZE ; num++ ) {
27:         cout << "a[" << num << "] = " ;
28:         cin >> a[num];
29:         if (a[num] == -99) break;

```

```

29:     }
30: }
31: void ArrayDisplay (double a[], int num) {
32:     for (int i = 0; i < num ; i++) {
33:         cout << "dat[" << i <<"] = " << a[i] << endl;
34:     }
35: }
36: double Average(double array[], int n)
37: {
38:     double sum, mean;
39:     sum = 0;
40:     for (int i = 0; i < n; i++ )
41:         sum = sum + array[i];
42:     mean = sum/(double)n;
43:     return mean;
44: }

```

ฟังก์ชันทั้ง 3 ฟังก์ชันในโปรแกรม ArrayInput (data, n), ArrayDisplay (data, n) และ Average (data ,n) เป็นการผ่านเฉพาะชื่อตัวแปรอะเรย์ (คือ data) ไปยังฟังก์ชันนั้น ๆ และผ่านค่าจำนวนของสมาชิกอะเรย์โดยใช้ตัวแปร n

ฟังก์ชัน ArrayInput จะผ่านค่า n แบบตัวแปรอ้างอิงนั้นหมายถึงฟังก์ชัน ArrayInput จะเป็นตัวกำหนดจำนวนสมาชิกของอะเรย์ data ว่าผู้ใช้จะป้อนข้อมูลเข้ามากี่ชุด n จะมีค่าตามจำนวนชุดที่ป้อน การวนรอบ for จะกำหนดไว้ให้ผู้ใช้ป้อนข้อมูลได้ไม่เกิน MAXSIZE = 200 ชุด

ฟังก์ชัน ArrayDisplay จะนำข้อมูลในตัวแปรอะเรย์ data มาแสดงผลเท่ากับจำนวนสมาชิก n

ฟังก์ชัน Average จะหาค่าเฉลี่ยของข้อมูลที่ใช้ป้อนเข้ามา n ชุด ส่งค่าเฉลี่ยคืนกลับไปยังโปรแกรมหลัก

การผ่านค่าอะเรย์ 2 มิติ ไปยังฟังก์ชัน

ตัวอย่าง โปรแกรมต่อไปนี้เป็นการเก็บคะแนนของนักศึกษา 3 คน แต่ละคนทำแบบทดสอบ 5 ชุด ได้คะแนนแตกต่างกัน การเก็บคะแนนจะเก็บไว้ในตัวแปรอะเรย์ขนาด 3 x 5 ( 3 แถว 5 คอลัมน์) จากนั้นคำนวณหาคะแนนเฉลี่ยของนักศึกษาแต่ละคน และคะแนนเฉลี่ยของแบบทดสอบแต่ละชุดที่นักศึกษาทำได้

```

1: // program – Array2Av.cpp
2: #include <iostream>
3: using namespace std;
4:
5: const int NumberOfStudents = 3;
6: const int NumberOfQuiz = 5;
7:
8: void DisplayTable( float table[][NumberOfQuiz], int row);
9: void ShowQuizAverage(float table[][NumberOfQuiz],int row);
10: void ShowClassAverage(float table[][NumberOfQuiz],int row);
11:
12: int main()
13: {
14:     float data[NumberOfStudents][NumberOfQuiz]= { {8.0, 8.5, 5.0,7.0,5.0},
15:                                                     { 4.0,2.0,1.5,7.0,5.5},
16:                                                     {9.0, 7.0,6.0, 7.0, 8.0}};
17:     int n = NumberOfStudents;

```

```

18:         cout << "Display scores of 3 students and 5 quizzes : \n";
19:         DisplayTable(data, n);
20:         cout << "\nNow calculate the class average and quiz average....\n";
21:         ShowQuizAverage(data, n);
22:         cout << endl;
23:         ShowClassAverage(data,n);
24:     return 0;
25: }
26: void DisplayTable (float a[][NumberOfQuiz], int num) {
27:     for (int i = 0; i < num ; i++) {
28:         cout << "\nStudent # " << i+1 << " : ";
29:         for(int j=0; j < NumberOfQuiz ; j++)
30:             cout << a[i][j] << " ";
31:     }
32: }
33: // finding quiz average of each student
34: void ShowQuizAverage (float a[][NumberOfQuiz], int num) {
35:     float sum;
36:     for (int i = 0; i < num ; i++) {
37:         sum = 0.0;
38:         for(int j=0; j < NumberOfQuiz ; j++)
39:             sum = sum+ a[i][j] ;
40:         cout << "Student's quizzes average # " << i+1 << " = "
41:             << sum/(float)NumberOfQuiz << endl;
42:     }
43: }
44: }
45: void ShowClassAverage (float a[][NumberOfQuiz], int num) {
46:     float sum;
47:     for (int i = 0; i < NumberOfQuiz ; i++) {
48:         sum = 0.0;
49:         for(int j=0; j < num ; j++)
50:             sum = sum+ a[j][i] ;
51:         cout << "Class average of quiz # " << i+1 << " = "
52:             << sum/(float)num << endl;
53:     }
54: }
55: }
56:

```

```

Display scores of 3 students and 5 quizzes :
Student # 1 : 8    8.5    5    7    5
Student # 2 : 4    2    1.5    7    5.5
Student # 3 : 9    7    6    7    8
Now calculate the class average and quiz average....
Student's quizzes average # 1 = 6.7
Student's quizzes average # 2 = 4
Student's quizzes average # 3 = 7.4

Class average of quiz # 1 = 7
Class average of quiz # 2 = 5.83333
Class average of quiz # 3 = 4.16667
Class average of quiz # 4 = 7
Class average of quiz # 5 = 6.16667

```



การเรียงลำดับข้อมูลโดยใช้ตัวแปรอะเรย์

การเรียงลำดับข้อมูล(จากมากไปหาน้อย (descending) หรือจากน้อยไปหามาก (Ascending)) เป็นสิ่งที่หลีกเลี่ยงไม่พ้นในการแก้ปัญหาทางวิทยาศาสตร์ ตัวแปรแบบอะเรย์จะช่วยแก้ปัญหการเรียงลำดับข้อมูลให้ง่ายขึ้น

การเรียงลำดับข้อมูลที่ง่ายที่สุดคือ การเรียงลำดับแบบ bubble sort ในการเรียงลำดับ(ในที่นี้จะเรียงจากน้อยไปมาก) จะมีการวนรอบแบบซ้อนกัน 2 ชั้น ชั้นในสุดจะเปรียบเทียบสมาชิกกับสมาชิกตัวที่อยู่ถัดไป ถ้ามีค่ามากกว่าจะสลับค่ากัน ลักษณะจะคล้ายกับสมาชิกตัวที่น้อยเป็นฟองอากาศที่ลอยตัวขึ้นมาอยู่บนสุดของผิวน้ำ

```
1:  #include <iostream>
2:  using namespace std;
3:
4:  const int ascending =0;
5:  const int decending =1;
6:  void BubbleSort ( double a[], int , int);
7:  void printdata (double a[], int );
8:
9:  int main()
10: {
11:     double data[10] = {9.5, 6.2, 7.6, 4.1, 5.3, 2.1, 8.7, 9.4, 1.6, 3.2 };
12:
13:     cout << "Before bubble sorting ... \n";
14:     printdata (data,10);
15:     cout << "\nAfter sorting by ascending .. \n ";
16:     BubbleSort(data,10,ascending);
17:     printdata (data,10);
18:
19:     cout << "\nAfter sorting by decending .. \n ";
20:     BubbleSort(data,10,decending);
21:
22:     printdata (data,10);
23:
24:     return 0;
25: }
26: void BubbleSort ( double a[], int n, int SortType) {
27:     double temp;
28:     for (int i = n-1; i > 0 ;i-- )
29:         for( int j =0; j < i; j++ )
30:         {
31:             if (SortType==0)          // ascending sort
32:                 { if (a[j] > a[j+1])
33:                     { temp = a[j];
34:                       a[j] = a[j+1];
35:                       a[j+1] = temp;
36:                     }
37:                 }
38:             else {
39:                 if (a[j] < a[j+1])
40:                     { temp = a[j];
41:                       a[j] = a[j+1];
42:                       a[j+1] = temp;
43:                     }
44:             } //else
```

```

45:          } //for loop
46:      }
47:      void printdata (double a[], int n ) {
48:          for (int i = 0; i < n ; i++ )
49:          {
50:              cout << a[i] << " , " ;
51:          }
52:      }
53:

```

```

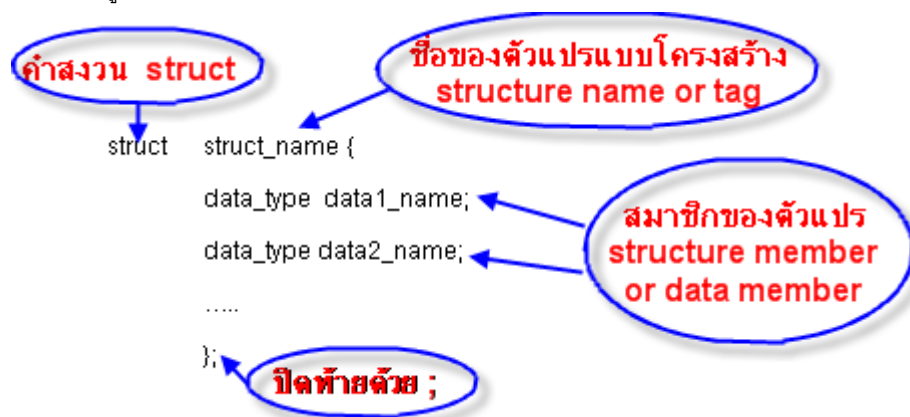
Before bubble sorting ...
9.5, 6.2, 7.6, 4.1, 5.3, 2.1, 8.7, 9.4, 1.6, 3.2,
After sorting by ascending ..
1.6, 2.1, 3.2, 4.1, 5.3, 6.2, 7.6, 8.7, 9.4, 9.5,
After sorting by descending ..
9.5, 9.4, 8.7, 7.6, 6.2, 5.3, 4.1, 3.2, 2.1, 1.6,

```

ข้อมูลแบบโครงสร้าง (structure)

สตรักเจอร์เป็นการนำข้อมูลที่เกี่ยวข้องกันแต่มีชนิดข้อมูลแตกต่างกันมารวมไว้เป็นกลุ่มเดียวกัน ต่างจากข้อมูลแบบอะเรย์ซึ่งต้องเป็นข้อมูลชนิดเดียวกันเท่านั้น

รูปแบบการใช้งาน



ต้องการเก็บรหัสนักศึกษา (Id) ชื่อ-สกุล (name) กลุ่มนักศึกษา(section) ระดับคะแนนเฉลี่ย(GPA) ของนักศึกษาแต่ละคน จะเป็นการสะดวกถ้านำข้อมูลของนักศึกษาแต่ละคนรวบรวมไว้เป็นชุดเดียวกัน ซึ่งทำได้โดยเก็บข้อมูลเป็นแบบ structure ดังนี้

```

struct StudentRecord {
    long id;
    char name [40];
    int section ;
    float gpa;
};

```

StudentRecord เป็นตัวแปรแบบโครงสร้าง ที่มีสมาชิกที่เป็นข้อมูลประเภท char int และ float การนำ StudentRecord ไปใช้งานในการเก็บข้อมูลนักศึกษา 5 คน ทำได้โดยประกาศตัวแปรดังต่อไปนี้

```
StudentRecord student01, student02, student03, student04, student05;
```

หรือจะใช้ตัวแปรแบบอะเรย์ในการประกาศค่าก็ได้

```
StudentRecord students[5];
```

การกำหนดค่าเริ่มต้นของตัวแปรโครงสร้าง ทำได้ดังนี้

```
StudentRecord students[5] = { { 100001, "Dang", 1, 3.2},  
                                {100002, "Sompong", 2, 2.4},  
                                {100003, "Naree", 1, 2.75},  
                                {100004, "Supan", 2, 1.89},  
                                {100005, "Wipa", 2, 3.00} };
```

ตัวอย่างต่อไปนี้เป็นกรป้อนข้อมูลนักศึกษา จำนวน 3 ราย ลงในตัวแปรแบบโครงสร้าง StudentRecord

จากนั้น แสดงผลข้อมูลที่ป้อนเข้าไปบนจอภาพ

```
1: // structure data type  
2: #include <iostream>  
3: using namespace std;  
4:  
5: struct StudentRecord {  
6:     long id;  
7:     char name [40];  
8:     int section ;  
9:     float gpa;  
10: };  
11:  
12:  
13: void ShowRecord(int , StudentRecord);  
14:  
15: int main() {  
16:     int index = 0;  
17:     char temp[20];  
18:     StudentRecord students[5];  
19:     do {  
20:         cout << "Id no.: ";  
21:         cin.getline(temp, sizeof(temp)-1);  
22:         students[index].id= atol(temp);  
23:         cout << "Name : ";  
24:         cin.getline(students[index].name, sizeof(students[index].name)-1);  
25:         cout << "Section : " ;  
26:         cin.getline(temp, sizeof(temp)-1);  
27:         cout << "Grade Point Average :";  
28:         students[index].section= atoi(temp);  
29:         cin.getline(temp, sizeof(temp)-1);  
30:         students[index].gpa= atof(temp);  
31:         index++;  
32:     }  
33:     while ( index < 3) ;  
34:     for (int i = 0; i < 3; i++ ) {  
35:         ShowRecord(i, students[i]);  
36:     }  
37: }  
38: void ShowRecord (int n, StudentRecord s) {  
39:     cout << "Record # " << (n+1) << " ==> " << endl;  
40:     cout << "Identification : " << s.id << " Name : " << s.name  
41:         << " Section : " << s.section << " Grade : " << s.gpa
```

```

42:                 << endl ;
43:     }
44:

```

```

Id no.: 1001
Name : Sompong
Section : 1
Grade Point Average :3.2
Id no.: 1003
Name : Wipapan
Section : 2
Grade Point Average :2.4
Id no.: 1004
Name : Tawee
Section : 1
Grade Point Average :1.8
Record # 1 ==>
Identification : 1001  Name : Sompong  Section : 1 Grade : 3.2
Record # 2 ==>
Identification : 1003  Name : Wipapan  Section : 2 Grade : 2.4
Record # 3 ==>
Identification : 1004  Name : Tawee  Section : 1 Grade : 1.8

```

ทดลองนำข้อมูลจากตัวแปรโครงสร้างซึ่งมีการกำหนดค่าไว้แล้วในตอนต้นโปรแกรม มาแสดงผลบนจอภาพ

```

1:    // structure data type
2:    #include <iostream>
3:    using namespace std;
4:
5:    struct StudentRecord {
6:        long id;
7:        char name [40];
8:        int section ;
9:        float gpa;
10:    };
11:
12:
13:    void ShowRecord(int , StudentRecord);
14:
15:    int main() {
16:        StudentRecord students[5] = { { 100001, "Dang", 1, 3.2},
17:                                         {100002, "Sompong", 2, 2.4},
18:                                         {100003, "Naree", 1, 2.75},
19:                                         {100004, "Supan", 2, 1.89},
20:                                         {100005, "Wipa", 2, 3.00} };
21:
22:        int index = 0;
23:        for (int i = 0; i < 5; i++ ) {
24:            ShowRecord(i, students[i]);
25:        }
26:    }
27:    void ShowRecord (int n, StudentRecord s) {
28:        cout << "Record # " << (n+1) << " ==> " << endl;
29:        cout << "Identification : " << s.id << " Name : " << s.name
30:             << " Section : " << s.section << " Grade : " << s.gpa
31:             << endl ;
32:    }
33:

```

ถ้าเรากำหนดค่าเริ่มต้นให้แก่ students[0], students[1], students[2], students[3] แต่ students[4] ไม่กำหนดค่าเริ่มต้นให้ จะเกิดอะไรขึ้น พบว่าคอมไพเลอร์จะใส่ค่า 0 ให้แก่ข้อมูลชนิดตัวเลข และใส่ค่า null ให้แก่ข้อมูลชนิดตัวอักษร

## Pointers

เมื่อประกาศตัวแปรในโปรแกรม เช่น `int i = 3;` หรือ `float x = 5.6;` เมื่อให้โปรแกรมทำงานจะเริ่มด้วยการจัดสรรหน่วยความจำแรม เพื่อเก็บตัวแปรเหล่านี้ ตัวแปรแต่ละตัวจะถูกเก็บในหน่วยความจำซึ่งมี address เป็นตัวเลขที่แน่นอน ขนาดของหน่วยความจำที่ใช้เก็บขึ้นอยู่กับชนิดข้อมูล เช่นตัวแปรชนิด `int` จะใช้เนื้อที่เก็บ 2 ไบต์ ตัวแปรชนิด `float` จะใช้เนื้อที่เก็บ 4 ไบต์

address ของหน่วยความจำจะเป็นเสมือนบ้านเลขที่ จะเริ่มต้นที่ 0, 1, 2, ... จนถึงค่าสูงสุดซึ่งขึ้นอยู่กับว่าคอมพิวเตอร์เครื่องนั้นมีหน่วยความจำเท่าใด ถ้าเครื่องคอมพิวเตอร์นั้นมีหน่วยความจำ 640 kB address ของหน่วยความจำจะมีค่าไปถึง 655,359 ถ้าเครื่องคอมพิวเตอร์นั้นมีหน่วยความจำ 1 MB จะมี address ได้ถึง 1,048,575 เมื่อโปรแกรมถูกโหลดเข้าสู่หน่วยความจำ ถ้าอยากรู้ว่าตัวแปรต่าง ๆ ในหน่วยความจำอยู่ที่ตำแหน่งเลขที่เท่าใด สามารถใช้ `&` (address of operator) หาตำแหน่งของตัวแปรนั้นได้ดังนี้

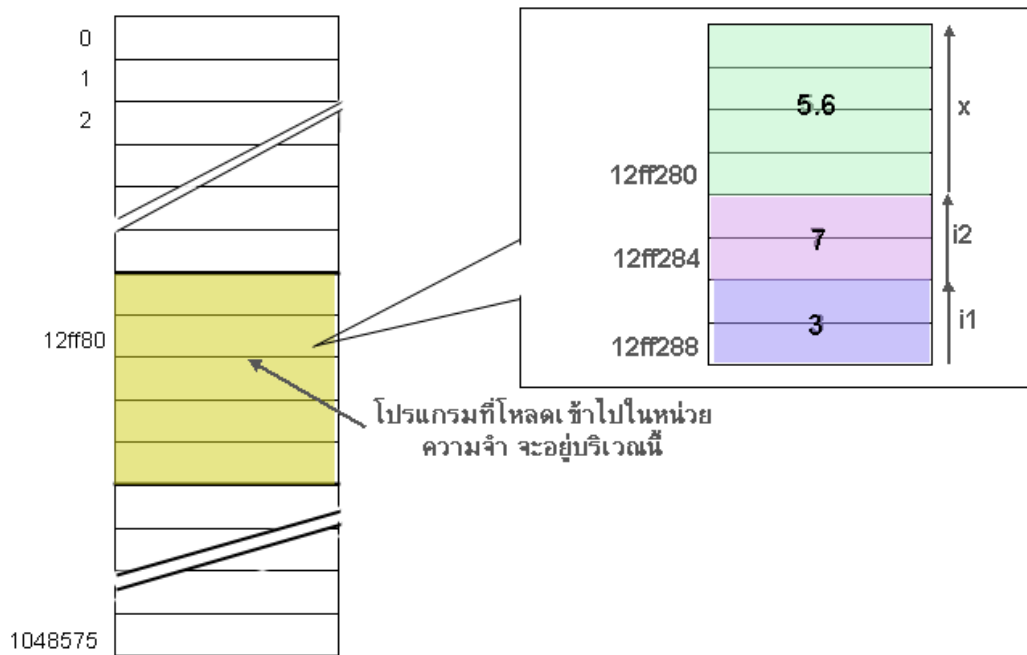
```
1: // pointer1.cpp
2: #include <iostream>
3: using namespace std;
4: int main() {
5:     int i1=3;
6:     int i2=7;
7:     float x = 5.6;
8:     cout << "Address of i1 =" << &i1 << endl
9:           << "Address of i2 =" << &i2 << endl
10:          << "Address of x =" << &x << endl;
11:
12: }
13:
```

ผลการทำงานของโปรแกรมจะได้ดังนี้

```
Address of i1 =0012FF88
Address of i2 = 0012FF84
Address of x  = 0012FF80
```

ในการให้โปรแกรมทำงานแต่ละครั้งหรือในแต่ละเครื่องจะได้ตำแหน่งของหน่วยความจำที่ต่างกันไป ขึ้นอยู่กับขนาดของระบบปฏิบัติการของเครื่องนั้น และขึ้นอยู่กับว่าขณะนั้นมีโปรแกรมอื่นกำลังทำงานอยู่ในเครื่องหรือไม่

จากรูปจะเห็นว่าตำแหน่งของหน่วยความจำจะเรียงจากมากไปน้อย ขณะรันโปรแกรม ตัวแปรเหล่านี้จะถูกเก็บไว้ในหน่วยความจำที่เรียกว่า stack โดยเริ่มต้นจาก address ที่มีค่ามากไปสู่ address ที่มีค่าน้อย ถ้าตัวแปรนั้นเป็นแบบ external จะถูกเก็บไว้ในหน่วยความจำที่เรียกว่า heap จะเก็บตัวแปรเรียงซ้อนกันในลักษณะตรงข้ามกับ stack เลขที่ตำแหน่งจะเรียงจากน้อยไปมาก รายละเอียดตรงส่วนนี้ โปรแกรมเมอร์ไม่ต้องใส่ใจก็ได้ คอมไพเลอร์จะเป็นผู้จัดการรายละเอียดปลีกย่อยเหล่านี้ให้เรา



ระวังอย่าสับสน address of operator & ซึ่งนำหน้าชื่อตัวแปร กับ reference operator & ซึ่งตามหลังชนิดข้อมูล

ตัวแปรพอยเตอร์ ( Pointer variable)

ตัวแปรใดก็ตามที่เก็บตำแหน่งเลขที่ของตัวแปรอื่น หรือออปเจกต์อื่น เรียกตัวแปรนั้นว่าเป็นตัวแปรพอยเตอร์ คอมไพเลอร์ต้องรู้ด้วยว่าตัวแปรพอยน์เตอร์เก็บ address ของตัวแปรเป็นข้อมูลชนิดใด เพราะการเพิ่มค่า pointer ไปยัง address ต่าง ๆ จะเพิ่มตามขนาดของข้อมูล การประกาศตัวแปรพอยน์เตอร์ จึงต้องประกาศให้ตรงกับชนิดของตัวแปรนั้น ๆ ด้วย

การประกาศตัวแปรพอยน์เตอร์ ใน C++ มีรูปแบบดังนี้

```
data_type *pointer_var ; // define a ' pointer to type'
```

เช่น

```
int *pint; // define a pointer to int
```

```
float *ptrx; // define a pointer to float
```

\*pint และ \*ptrx เป็นตัวแปรพอยน์เตอร์ที่เก็บตำแหน่งเลขที่หน่วยความจำของตัวแปรที่มีชนิดข้อมูลเป็น int และ float ตามลำดับ เครื่องหมาย \* มีความหมายว่าเป็น pointer to type

ตัวอย่างต่อไปนี้เป็นกำหนัดค่า address ให้กับตัวแปรพอยน์เตอร์และใช้ประโยชน์จากตัวแปรพอยน์เตอร์ในการแสดงค่า

```
1: // pointer2.cpp
2: #include <iostream>
3: using namespace std;
4: int main() {
```

```

5:   int i1=3;
6:   int i2=7;
7:   int *ptr_int;
8:   float x = 5.6;
9:   float *ptr_x;
10:  ptr_int = &i1; // pointer point to i1
11:  cout << "content in ptr_int = *ptr_int = " << *ptr_int << endl;
12:  ptr_int = &i2;
13:  cout << "content in ptr_int = *ptr_int = " << *ptr_int << endl;
14:  ptr_x = &x;
15:  cout << "content in ptr_x = *ptr_x = " << *ptr_x << endl;
16:
17: }
18:

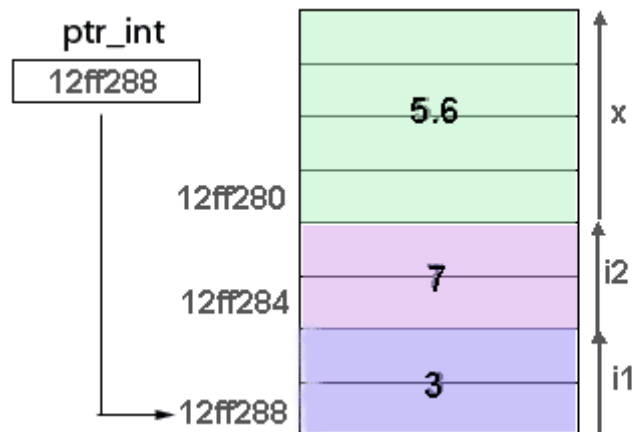
```

```

content in ptr_int = *ptr_int = 3
content in ptr_int = *ptr_int = 7
content in ptr_x = *ptr_x = 5.6

```

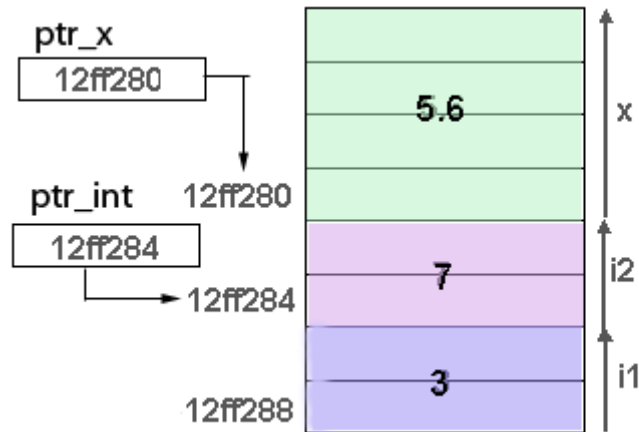
บรรทัดที่ 10 เป็นการกำหนดค่าให้ตัวแปรพอยน์เตอร์ ptr\_int เก็บตำแหน่ง address ของ i1



เมื่อพิมพ์ค่า ptr\_int จะได้ค่าหน่วยความจำของตัวแปร i1 เมื่อให้พิมพ์ค่า \*ptr\_int จะแสดงข้อมูลที่เก็บไว้ในหน่วยความจำที่ address นี้ จะเห็นว่าในบางครั้ง เราอาจไม่ทราบชื่อตัวแปร แต่ถ้ารู้ address ของตัวแปร ก็สามารถรู้ค่าข้อมูลของตัวแปรนั้นมีค่าเท่าใดได้ โดยอาศัยตัวแปรแบบพอยน์เตอร์

บรรทัดที่ 12 เป็นการให้ตัวแปรพอยน์เตอร์เปลี่ยนค่า address ของหน่วยความจำเป็นของ i2 ข้อมูลใน ptr\_int จะกลายเป็นตำแหน่งเลขที่ของหน่วยความจำของตัวแปร i2 ดังรูป

และตัวแปร ptr\_x จะเก็บ address ของตัวแปร x



เมื่อประกาศตัวแปรพอยน์เตอร์โดยไม่กำหนดค่าเริ่มต้น ค่าที่เก็บอยู่ในตัวแปรพอยน์เตอร์อาจเป็นค่าอะไรก็ได้ อาจชี้ไปยังหน่วยความจำที่ address ใด ๆ ซึ่งอาจเป็นที่เก็บข้อมูลสำคัญของระบบปฏิบัติการ ถ้ามีการเปลี่ยนค่าอาจทำให้เครื่องหยุดทำงาน การกำหนดค่าเริ่มต้นให้แก่ตัวแปรพอยน์เตอร์จึงเป็นสิ่งสำคัญ

\* เป็น operator เข้าถึงข้อมูลของตัวแปรที่มี address ตรงกับ พอยน์เตอร์นั้นชี้อยู่ อาจกล่าวได้ว่า เครื่องหมาย \* คือ 'at address'

& จะให้ค่าเลขที่ตำแหน่งของหน่วยความจำของตัวแปรนั้น จำง่าย ๆ คือ 'The address of ....'

ให้ทดลองเปลี่ยนคำสั่งให้เป็นอย่างต่อไปนี้ ทดลองดูว่าเมื่อคอมไพล์จะเกิดอะไรขึ้น

```
ptr_int = &x;
```

```
ptr_x = &i1;
```

หรือประกาศตัวแปรดังนี้

```
int *ptr_int = &i1;
```

```
int i1;
```

พิจารณาส่วนหนึ่งของโปรแกรมต่อไปนี้ แล้วคาดคะเนว่าผลลัพธ์ที่ได้

```
int *ptr_int;
int n;
ptr_int = &n;
*ptr_int = 15;
cout << "n = " << n << endl;
cout << " *ptr_int = " << *ptr_int << endl;
```

.....

ตัวแปรพอยน์เตอร์สามารถใช้ในการคำนวณคณิตศาสตร์

```
int *ptri;
```

```
int i, n =5;
```

```
ptri = &n;
```

```
i = 8*(*ptri); // 8*5 = 40
```

.....

.....

หรือจะเขียนเป็น `8 * *ptri` ก็ได้ เพราะ `*ptri` เป็น indirection operator มีลำดับความสำคัญก่อนเครื่องหมายคูณ



คำสั่งทั้งสองบรรทัดต่อไปนี้จะให้ผลลัพธ์เช่นเดียวกัน

```
j = i;
```

```
j = *i;
```

ประโยคต่อไปนี้จะระวัง

```
i = 8/*ptri // divide 8 by object pointer to, assign result to i
```

เพราะเครื่องหมาย /\* คอมไพเลอร์ (บางบริษัท) จะมองว่าเป็นเริ่มต้นของการ comment

การเพิ่มค่า pointer ดังตัวอย่างต่อไปนี้

```
int i = 10;
```

```
int *pi;
```

```
*pi = i;
```

```
*pi += 1; // add 1 to *pi
```

หรือจะใช้คำสั่ง (\*pi)++; การใส่วงเล็บ จะประมวลผลเริ่มจากขวาไปซ้าย (กรณี ++ และ \* )

ถ้าเขียน \*pi ++ จะเป็น \*(pi++)

### ความสัมพันธ์ระหว่างอะเรย์และพอยน์เตอร์

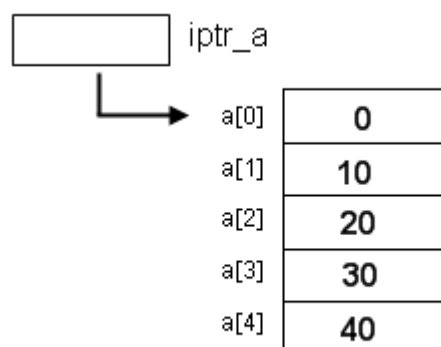
อะเรย์และพอยน์เตอร์มีความสัมพันธ์ใกล้ชิดกันมาก บางครั้งการจัดการกับสมาชิกของอะเรย์โดยใช้ตัวแปรพอยน์เตอร์ จะมีประสิทธิภาพมากกว่า การกล่าวถึงชื่ออะเรย์และดรรชนีหรือ subscript โดยตรง

เมื่อประกาศตัวแปร `int a [ 5 ] = { 0, 10, 20, 30, 40};` หมายถึง a เป็นตัวแปรชนิดอะเรย์เก็บจำนวนเต็มได้ 5 ค่า

`int *iptr_a;` ตัวแปรชื่อ `ptr_a` เป็นตัวแปรพอยน์เตอร์ชี้ไปยัง address ของอะเรย์ที่เก็บจำนวนเต็ม

คำสั่ง `iptr = &a[0];`

กำหนดให้ `iptr_a` ชี้ไปที่ตำแหน่ง `a[0]` หรือจะเขียนเป็น `iptr_a = a;` จะให้ผลเหมือนกัน



ตัวอย่างต่อไปนี้เป็นารอ้างถึงหรือแสดงข้อมูลในตัวแปรอะเรย์โดยใช้พอยน์เตอร์ ดังนี้

```
1: // array pointer.cpp
2: #include <iostream>
3: using namespace std;
4: int main() {
```

```

5:   int a[5] = {0, 10,20,30,40};
6:   int *iptr, index;
7:   iptr = a;
8:   cout << " \n1) Using a[index] " << endl;
9:   for (index = 0; index < 5; index++)
10:      cout << "a[" << index << "] = " << a[index] << " " ;
11:   cout << endl;
12:
13:   cout << " \n2) Using iptr[index] " << endl;
14:   for (index = 0; index < 5; index++)
15:      cout << "iptr[" << index << "] = " << iptr[index] << " " ;
16:   cout << endl;
17:
18:   cout << " \n 3) Using array name as a pointer *(a+index) " << endl;
19:   for (index = 0; index < 5; index++)
20:      cout << "**(a+ " << index << ") = " << *(a+index) << " " ;
21:   cout << endl;
22:
23:   cout << " \n 4) Using a pointer name as a pointer *(iptr+index) " << endl;
24:   for (index = 0; index < 5; index++)
25:      cout << "**(iptr + " << index << ") = " << *(iptr+index) << " " ;
26:   cout << endl;
27:
28:   cout << " \n 5) Using increasing pointer Arithmetics iptr++ " << endl;
29:   for (iptr = &a[0]; iptr <= &a[4]; iptr++)
30:      cout << "**(iptr) at address " << iptr << ") = " << *iptr << " " ;
31:   cout << endl;
32:
33:   return 0;
34:   }
35:

```

ผลลัพธ์จากการทำงานของโปรแกรมดังกล่าวจะได้ดังนี้

1) Using a[index]

a[0] = 0 a[1] = 10 a[2] = 20 a[3] = 30 a[4] = 40

2) Using iptr[index]

iptr[0] = 0 iptr[1] = 10 iptr[2] = 20 iptr[3] = 30 iptr[4] = 40

3) Using array name as a pointer \*(a+index)

\*(a+ 0) = 0 \*(a+ 1) = 10 \*(a+ 2) = 20 \*(a+ 3) = 30 \*(a+ 4) = 40

4) Using a pointer name as a pointer \*(iptr+index)

\*(iptr + 0) = 0 \*(iptr + 1) = 10 \*(iptr + 2) = 20 \*(iptr + 3) = 30 \*(iptr + 4) = 40

5) Using increasing pointer Arithmetics iptr++

\*(iptr) at address 0012FF78) = 0 \*(iptr) at address 0012FF7C) = 10

\*(iptr) at address 0012FF80) = 20 \*(iptr) at address 0012FF84) = 30

\*(iptr) at address 0012FF88) = 40

การใช้ indirection operator กับการเพิ่มค่าและลดค่า

กำหนดให้ int data[ ] = { 10, 20, 30, 40, 50};

และ int \*ip, temp;

ip = & data[1];

temp = \*ip;

จะเห็นว่า ip เป็นตัวแปรพอยน์เตอร์ชี้ไปยังสมาชิกอะเรย์ data ตัวที่ 2 ซึ่งมีค่าเท่ากับ 20 ค่าที่เก็บไว้ในตัวแปร temp จึงเท่ากับ 20 ด้วย

ถ้าเพิ่มประโยคคำสั่งต่อไปนี้

++ip; // \*(++ip)

temp = \*ip;

เป็นการเพิ่มค่าพอยน์เตอร์ ip ไปอีก 1 ตำแหน่ง พอยน์เตอร์จะชี้ไปที่สมาชิกอะเรย์ตัวที่ 3 ค่าที่เก็บไว้ใน temp จึงมีค่าเท่ากับ 30

++ \*ip; // ++ (\*ip)

temp = \*ip;

ขณะนี้ ip ชี้อยู่ที่สมาชิกตัวที่ 3 ของ data คำสั่ง ++(\*ip) เป็นการเพิ่มค่าข้อมูลที่เก็บไว้ใน address ที่ ip ชี้อยู่ จาก 30 เป็น 31 ค่าที่เก็บไว้ใน temp จึงมีค่า 31

temp = \*ip++; // y = \*(ip++)

เครื่องหมาย++ จะกระทำกับ ip เพราะใส่ไว้ข้างหลัง ip หรือมีลักษณะเป็น postfix จึงมีผลเมื่อประโยคนี้นี้ได้ถูกประมวลผลแล้ว นั่นคือ temp ยังคงมีค่าเท่ากับ 31 เมื่อ temp ได้รับค่าแล้ว ip จะเพิ่มค่าขึ้นอีก 1 จะชี้ไปยังสมาชิกตัวที่ 4 ซึ่งมีค่าเท่ากับ 40

temp = (\*ip)++;

เป็นการเพิ่มค่าข้อมูลที่เก็บไว้ใน address ที่ ip ชี้อยู่ จาก 40 จะมีค่าเป็น 41

## ความสัมพันธ์ระหว่าง Structure และ Pointer

จากตัวแปรแบบโครงสร้าง

```
struct StudentRecord {  
    long id;  
    char name [40];  
    int section ;  
    float gpa;  
};
```

ประกาศตัวแปรที่มีชนิดข้อมูลแบบ StudentRecord ได้ดังนี้

StudentRecord mystudent, \*p\_student;

p\_student = &mystudent;

p\_student เป็นตัวแปรพอยน์เตอร์ชี้ไปยังข้อมูลชนิด StudentRecord ซึ่งเป็นตัวแปรแบบโครงสร้าง การใช้ตัวแปรพอยน์เตอร์เข้าถึงข้อมูลแบบโครงสร้างทำได้ดังนี้

```
(*p_student).id = 1001;
strcpy(*p_student.name = "Somsak");
(*p_student).section = 1;
(*p_student).gpa = 2.75;
```

เพราะเครื่องหมายจุด . มีลำดับความสำคัญมากกว่าเครื่องหมาย \* จึงต้องใส่วงเล็บที่ตัวแปรพอยน์เตอร์ เพื่อบังคับให้คอมไพเลอร์มองเห็น p\_student เป็นพอยน์เตอร์ก่อน การใช้พอยน์เตอร์กับตัวแปรแบบโครงสร้าง จะนิยมใช้เครื่องหมาย → คำสั่งชุดข้างบนจึงเขียนใหม่ได้เป็น

```
p_student→id = 1001;
strcpy(p_student→name = "Somsak");
p_student→section = 1;
p_student→gpa = 2.75;
```

การจัดสรรหน่วยความจำขณะ run time โดยใช้พอยน์เตอร์

การใช้โปรแกรมที่เขียนด้วยภาษา C++ จะเกี่ยวข้องกับหน่วยความจำใน 3 ลักษณะ คือ จัดเก็บแบบอัตโนมัติ (automatic storage) เป็นบริเวณหน่วยความจำที่ใช้เก็บตัวแปรของฟังก์ชัน เมื่อมีการเรียกใช้ฟังก์ชันแบบที่ 2 คือจัดเก็บแบบสถิต (static storage) เมื่อข้อมูลถูกประกาศให้เป็น static ซึ่งจะคงอยู่ตลอดเวลาเมื่อโปรแกรมทำงาน และส่วนที่จัดเก็บอย่างอิสระ (free storage) ในภาษา C เรียกส่วนนี้ว่า heap หน่วยความจำส่วนนี้จะถูกจัดสรรสำหรับคำสั่งของโปรแกรมขณะที่ประมวลผล เราสามารถจัดสรรและจัดการหน่วยความจำบริเวณนี้โดยใช้คำสั่ง new และ delete

การที่เราสามารถจัดสรรควบคุมหน่วยความจำได้ขณะที่โปรแกรมทำงานจะเป็นการใช้ทรัพยากร (หน่วยความจำ) ได้อย่างมีประสิทธิภาพ ณ ขณะนี้เราสามารถจับจองและใช้หน่วยความจำโดยวิธีสร้างตัวแปร นั่นคือบอกให้คอมไพเลอร์รู้ว่าเราจะต้องใช้หน่วยความจำจำนวนเท่าใดในการรันโปรแกรม ในบางกรณีทำให้เกิดการใช้พื้นที่หน่วยความจำไม่คุ้มค่า

ตัวอย่างเช่น ต้องการหาคะแนนเฉลี่ยของนักศึกษาแต่ละกลุ่ม กลุ่มนักศึกษาอาจมีมากถึง 200 คน บางกลุ่มมี 20 คน บางกลุ่มมี 10 คน การประกาศตัวแปร float score[200]; ต้องจองเนื้อที่ไว้ถึง 200 ที่ ถึงแม้ว่าส่วนใหญ่กลุ่มนักศึกษาจะมีเพียง 20 คน หรือ 30 คน ก็ตาม

การจัดสรรหน่วยความจำแบบพลวัต หมายถึงการจองเนื้อที่หน่วยความจำเท่าที่จำเป็นในขณะโปรแกรมประมวลผล การจัดสรรหน่วยความจำ ทำได้โดยใช้คำสั่ง new และปล่อยคืนหน่วยความจำด้วยคำสั่ง delete ดังตัวอย่างต่อไปนี้

```
1: // Program -- NewDel1.cpp
2: #include <iostream>
3: using namespace std;
4:
5: int main( )
6: {
7:     double *e = new double;
8:     *e=2.71828;
9:     cout << "The values of e = " << *e << endl;
10:    delete e;
11:
12:
```

```

13:         return 0;
14:     }

```

ในการจองเนื้อที่หน่วยความจำ ถ้าจองไม่สำเร็จ คำสั่ง new จะคืนค่ากลับเป็น null pointer ซึ่งปกติมีค่าเป็น 0 แทนด้วยสัญลักษณ์ NULL การใช้คำสั่ง new ควรมีการตรวจสอบทุกครั้งว่าสามารถจองหน่วยความจำได้สำเร็จหรือไม่ ดังนี้

```

.....
double *e = new double;
    if ( e == NULL) exit(1);
    *e =2.71828;
.....

```

การใช้ new และ delete กับตัวแปรชนิดอะเรย์

เมื่อใช้คำสั่ง new หรือ delete เพื่อจองหรือปลดปล่อยหน่วยความจำกับตัวแปรประเภทอะเรย์ จะต้องเพิ่มวงเล็บต่อท้ายดังนี้ new[ ] และ delete [ ] อะเรย์ที่ถูกสร้างโดยคำสั่ง new เรียกว่า dynamic array

ตัวอย่างต่อไปนี้เป็น การจองหน่วยความจำเพื่อเก็บค่า double จำนวน 10 ค่า

```

1:  // Program -- NewDel2.cpp
2:  #include <iostream>
3:  using namespace std;
4:
5:  int main( )
6:  {
7:      double *number = new double[10];
8:      if (number == NULL) exit(1);
9:      number[2] = 2.345;
10:     cout << "The value stored in number[2] = " << number[2] << endl;
11:     delete[] number;
12:
13:     return 0;
14: }

```

ตัวอย่างการใช้ new และ delete อีกตัวอย่างหนึ่ง

```

1:  // Program -- NewDel3.cpp
2:  #include <iostream>
3:  using namespace std;
4:
5:  int main( )
6:  {
7:      int *i;
8:      char *c;
9:      i = new int; // return pointer to int-size , 2 byte
10:     *i = 15273;
11:     c = new char [125]; // allocate block of 125 * char-size byte
12:     cout << "Type any letters....";
13:     cin >> c;
14:     cout << c << endl;
15:     delete i;
16:     delete c;
17:
18:     return 0;
19: }

```

ในภาษา C มีคำสั่ง malloc () และ free () ซึ่งเทียบเท่ากับคำสั่ง new และ delete สามารถใช้ได้ ใน C++ เช่นเดียวกัน แต่ new และ delete จะดูเข้าใจง่ายกว่า

```
main () {  
    int *i;  
    i = (int *) malloc( sizeof(int));  
    *i = 15273;  
    printf(" %d \n ", *i);  
    free(i);  
}
```

ข้อควรระวังการใช้คำสั่ง new

1. ไม่สามารถใช้กับค่าคงที่ได้โดยตรง เช่น

```
i = new 125; // จองพื้นที่หน่วยความจำ 125 byte
```

2. การจองหน่วยความจำสำหรับอะเรย์หลายมิติมีรูปแบบที่พึงระวางดังนี้

```
int *k;
```

```
k = new int[4][10]; // เป็นการจองอะเรย์ k ชนิดจำนวนเต็มใช้เนื้อที่ 80 byte
```

หรือ

```
int *k;
```

```
int i;
```

```
i =4;
```

```
k = new int [i][10]; // จัดว่าเป็นคำสั่งที่ถูกต้อง
```

```
i = 10;
```

```
k = new int[4][i] ; // ไม่ถูกต้องในวงเล็บหลังจะต้องเป็นตัวคงที่เสมอ
```

แบบฝึกหัด

1. ข้อใดประกาศตัวแปรแบบอะเรย์ได้ถูกต้อง

ก. int a[10];

ข. int a;

ค. a { 10};

ง. array a[10];

2. เลขตรรกของอะเรย์ที่มีสมาชิก 29 ตัว คือข้อใด

ก. 29

ข. 28

ค. 0

ง. ขึ้นอยู่กับโปรแกรมเมอร์กำหนด

เฉลย ข

3. ข้อใดเป็นอะเรย์แบบ 2 มิติ

- ก. array a[20][20];
- ข. int a[20][20];
- ค. char a[20];
- ง. int a[20,20];

4. ต้องการเข้าถึงข้อมูลของสมาชิกอะเรย์ตัวที่ 7 คือข้อใด

- ก. f[6];
- ข. f[7];
- ค. f(7);
- ง. f

เฉลย ก.

5. ข้อใดต่อไปนี้บอกถึงตำแหน่งของหน่วยความจำของสมาชิกตัวแรกของอะเรย์ foo ซึ่งมีจำนวนสมาชิก 100 ตัว

- ก. foo[0];
- ข. foo;
- ค. &foo;
- ง. foo[1];

เฉลย ข.

- 6. จงเขียนโปรแกรมหาค่าตัวเลขที่มากที่สุดในอะเรย์พร้อมกับระบุตำแหน่งของสมาชิกอะเรย์ที่เก็บตัวเลขค่ามากที่สุด
- 7. จงประกาศตัวแปรอะเรย์ชื่อ rainfall ซึ่งใช้เก็บปริมาณน้ำฝนที่ตกลงมาใน 1 สัปดาห์ให้ค่าเริ่มต้นของสมาชิกแต่ละตัวมีค่าเป็นศูนย์

เฉลย const int days = 7;

float rainfall [days] = {0};

8. จงประกาศตัวแปรอะเรย์เก็บข้อมูลชนิดตัวอักษรจำนวน 100 ตัว โดยเริ่มต้นให้สมาชิกทุกตัวมีค่าเป็น a

เฉลย char arrayOfChar[100] = {'a'};

9. จงประกาศตัวแปรอะเรย์เพื่อใช้ระบุตำแหน่งตารางหมากรุกขนาด 8 x 8

เฉลย int chessboard [8][8];

10. จงประกาศตัวแปรอะเรย์ 2 มิติ ขนาด 10 x 10 ให้แต่ละช่องเก็บอักษร x เมื่อตอนเริ่มต้น

เฉลย char table[10][10] = { 'x' };

11. จงเขียนประกาศตัวแปรอะเรย์ต่อไปนี้

11.1 อะเรย์เก็บจำนวนเต็ม 10 ค่า

11.2 อะเรย์เก็บจำนวนจริง 4 ค่า กำหนดค่าเริ่มต้นดังนี้ 3.2, 1.8, 7.7 และ 5.1

11.3 อะเรย์เก็บข้อความ "Hello"

12. การประกาศตัวแปรต่อไปนี้ต่างกันอย่างไร

```
char word[8] = "Welcome";
```

```
char word[8] = { 'W', 'e', 'l', 'c', 'o', 'm', 'e' };
```

12. จงนำข้อมูลบอกความสูงในตารางต่อไปนี้เขียนเป็นอะเรย์

เพศ	น้อยกว่า 150 cm	150 – 170 cm	มากกว่า 170 cm
ชาย	13 %	55 %	22 %
หญิง	15%	75%	10%



## บทเบ็ดเตล็ด

โปรแกรมเล็ก ๆ ต่อไปนี้ เป็นโปรแกรมที่ผู้เขียนเขียนไว้ใช้งานส่วนตัว อาจจะมีประโยชน์บ้างสำหรับผู้ที่สนใจ

### โปรแกรมใส่เลขบรรทัดของ Source code ของโปรแกรมคอมพิวเตอร์ (LineNo.cpp)

โดยปกติ โปรแกรมต้นฉบับภาษา C++ จะไม่มีเลขบรรทัดกำกับ ในการอธิบายโปรแกรมว่าแต่ละบรรทัดทำหน้าที่อะไรนั้น การมีตัวเลขกำกับหน้าบรรทัดจะทำให้การอ้างอิงถึงบรรทัดที่จะอธิบายทำได้สะดวกขึ้น เป็นประโยชน์ต่อผู้อ่านที่จะสามารถติดตามได้ง่าย โดยไม่ต้องมาคอยนับบรรทัด

ในการใส่ตัวเลขหน้าบรรทัดโดยวิธีเติมตัวเลขที่ละบรรทัดนั้นเป็นการเสียเวลาไม่น้อย ในที่นี้จึงเขียนโปรแกรมให้ทำหน้าที่ใส่ตัวเลขหน้าบรรทัดของแต่ละบรรทัดให้อย่างอัตโนมัติ โปรแกรมจะ save ไฟล์ที่ใส่เลขประจำบรรทัดนั้นเป็นชื่อเดิม แต่เปลี่ยนนามสกุลของไฟล์ใหม่จาก cpp เป็น lin

Source code ของโปรแกรมมีดังนี้

```
1:      // LineNumber.cpp
2:      // Jan 3.2004
3:      #include <iostream.h>
4:      #include <fstream.h>
5:      #include <string.h>
6:      #include <stdlib.h>
7:      #include <iomanip.h>
8:
9:      main( int argc, char *argv[])
10:     {
11:         char *infilename;
12:         char outfilename[25] ;
13:         ofstream outfile;
14:         ifstream infile;
15:         int line = 1;
16:         char c;
17:         if (argc < 2 ) {
18:             cout << "Enter filename : ";
19:             cin >> infilename;
20:         }
21:         else {
```

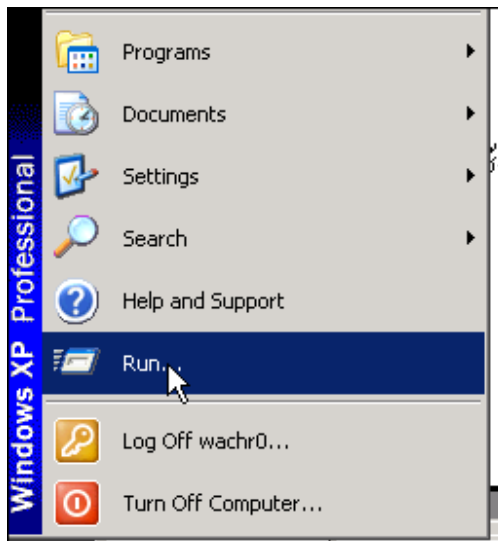
```

22:             strcpy(infilename,argv[1]);
23:         }
24:         cout << "Input your filename : " << infilename << endl;
25:         infile.open( infilename, ios::in);
26:         if (!infile) {
27:             cerr << " Could not open file \n";
28:             exit(1);
29:         }
30:
31:         strncpy(outfilename, infilename,strlen(infilename) - 5);
32:         outfilename[strlen(infilename)-5]= '\0';
33:         strcat(outfilename, ".lin");
34:         cout << "has been inserted line number.\n"
35:             << "And its name 's changed to " << outfilename << endl;
36:         outfile.open(outfilename, ios::out);
37:         if (!outfile) { cerr << "Could not create output file\n";}
38:         outfile << setw(3) << line << ":\t";
39:         while((c = infile.get()) != EOF) {
40:             if (c == '\n') {
41:                 line++;
42:                 outfile.put(c);
43:                 outfile << setw(3) << line << ":\t";
44:             }
45:             else outfile.put(c);
46:         }
47:
48:         infile.close();
49:         outfile.close();
50:         return 0;
51:     }

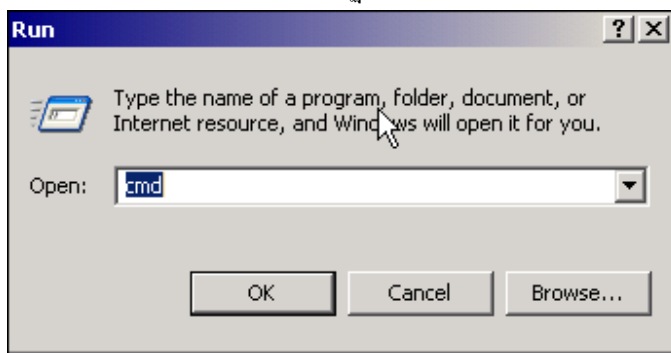
```

การใช้งานโปรแกรม สามารถพิมพ์คำสั่งได้โดยตรง ใน DOS mode ตามลำดับดังนี้

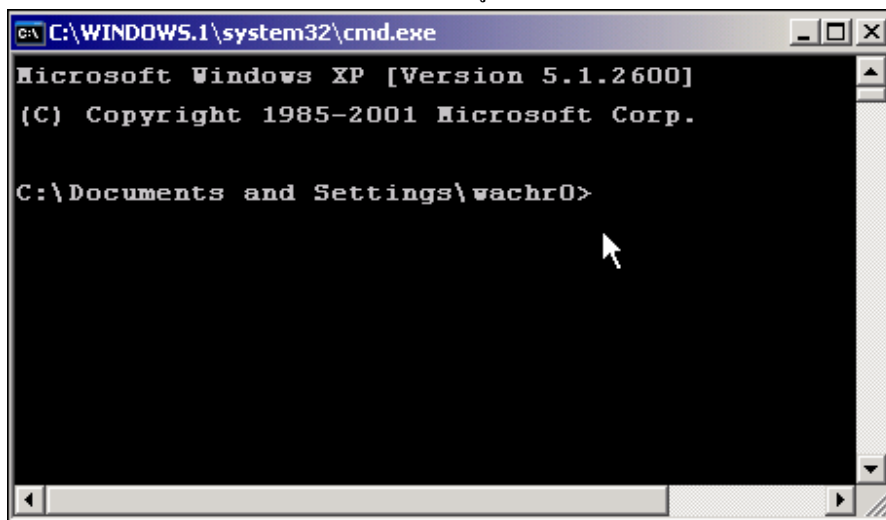
- คลิกที่ปุ่ม start และคลิกที่เมนู Run



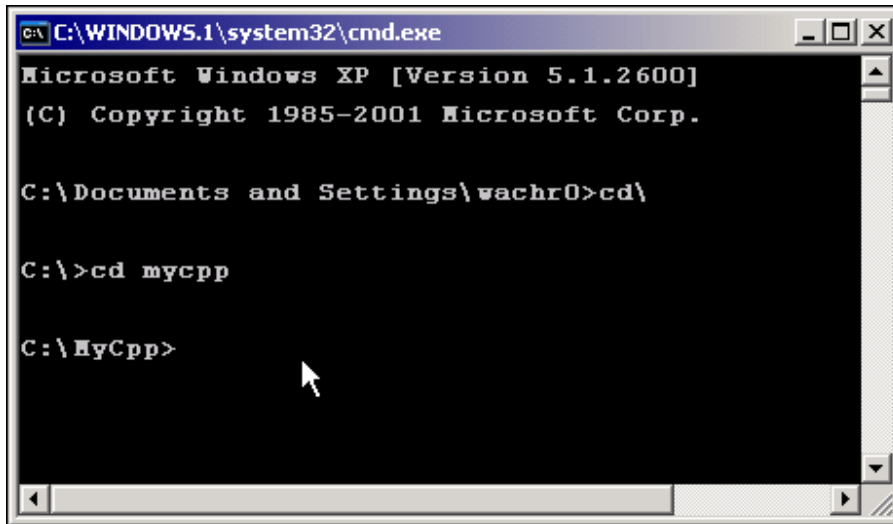
- จะมีหน้าต่าง Run ปรากฏขึ้น ให้พิมพ์คำว่า cmd พร้อมกด Enter



- จะได้หน้าต่าง Command window ดังรูป



- ไปที่ โฟลเดอร์ ที่เราเก็บ Source code ในที่นี้ คือ MyCpp ให้นำโปรแกรม LineNo.exe อยู่ในโฟลเดอร์เดียวกันนี้ด้วย



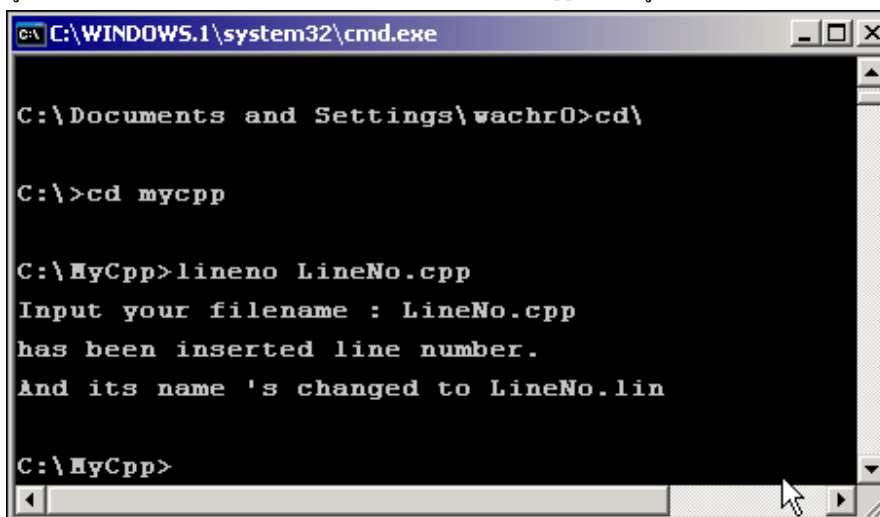
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\wachr0>cd\

C:\>cd mycpp

C:\MyCpp>
```

- จะทดลองใส่เลขหน้าบรรทัดของไฟล์ LineNo.cpp ให้พิมพ์คำสั่งดังรูป โปรแกรมจะแสดงข้อความให้ทราบว่าได้ใส่หมายเลขที่หน้าบรรทัดแต่ละบรรทัดของไฟล์ LineNo.cpp แล้ว และไฟล์ที่มีการใส่เลขหน้าบรรทัดนี้จะถูกเปลี่ยนชื่อใหม่เป็น LineNo.lin ไฟล์ต้นฉบับ LineNo.cpp ไม่ได้ถูกเปลี่ยนแปลงหรือแก้ไขแต่อย่างใด



```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\wachr0>cd\

C:\>cd mycpp

C:\MyCpp>lineno LineNo.cpp
Input your filename : LineNo.cpp
has been inserted line number.
And its name 's changed to LineNo.lin

C:\MyCpp>
```

จากนั้นเราสามารถนำไฟล์ LineNo.lin ไปใช้งานประกอบการอธิบายโปรแกรม โปรแกรมนี้สามารถใส่หมายเลขบรรทัด source code ของโปรแกรมที่เขียนโดยภาษาคอมพิวเตอร์อื่น ได้เช่นกัน โดยไฟล์ต้นฉบับนี้จะต้องถูกจัดเก็บในรูปแบบ text file

### โปรแกรมค้นหาคำใน text file

เป็นโปรแกรมที่ใช้ค้นหาคำที่ต้องการ ผู้ใช้จะป้อนคำที่ต้องการค้นหา โปรแกรมจะแจ้งให้ทราบว่าคำที่ค้นหานั้นพบเป็นจำนวนกี่ครั้ง อยู่ที่บรรทัดที่เท่าใดบ้าง

โปรแกรมนี้ต้องระบุชื่อไฟล์ที่ต้องการค้นหาลงในโปรแกรม ซึ่งในโปรแกรมนี้จะใช้ค้นหาคำที่ต้องการในไฟล์ชื่อ somewhere.txt อาจไม่สะดวกนัก ถ้าไฟล์ที่ต้องการค้นหานั้นมีอยู่หลายไฟล์ ซึ่งควรจะแก้ไขให้ผู้ใช้สามารถระบุชื่อไฟล์ที่ต้องการค้นหาได้เมื่อเริ่มต้นใช้คำสั่งให้โปรแกรมทำงาน

รายละเอียดของโปรแกรมนี้นี้

```
1: // Findtext.cpp --- search your word or phrase in text file
2: // Last update : Sept 22, 2003
3:
4: #include <stdio.h>
5: #include <iostream.h>
6: #include <string.h>
7:
8: int main( ) {
9:
10: //char *filename;
11: FILE *fp;
12: int i,c,findFlag, wlength,buflength,position,foundtime;
13: int foundpos[64];
14: char buffer[10240];
15: char yourword[255];
16:
17: char ch;
18:     cout << "Input what you want to search for : ";
19:     cin.get(yourword,255);
20: //     cout << yourword << "\n";
21:     wlength = strlen(yourword);
22: //     printf("length of buffer = %d \n",wlength);
23:     findFlag = 0;
24:     if ((fp=fopen("somewhere.txt","r"))== NULL) {
25:         cout << "File opening failed \n";
26:         return 1;
27:     }
28:     // store all texts into memory
29:     c = fgetc(fp);
30:     for( i=0; (i < 10240) && (feof(fp)==0) ; i++) {
31:         buffer[i] = c;
32:         c = fgetc(fp);
33:     }
34:     buffer[i] = '\0' ; //make a c -style string
35:     buflength = strlen(buffer);
36:     cout << "\nThe content in this file ..... : \n\n";
37:     cout << buffer << "\n\n";
38:     position = 0;
39:     foundtime=0;
40:     for (i =0; i < (buflength - wlength) ;i++) {
41:         ch=buffer[i];
42:         if (ch != '\n' ) position +=1;
```

```

43:         if (ch == yourword[findFlag]) {
44:             findFlag += 1;
45:             if (findFlag == wlength) {
46:                 foundtime += 1;
47:                 foundpos[foundtime] = position - wlength + 1;
48:             }
49:         } else { findFlag = 0; }
50:     } //end for loop
51:     cout << "" << yourword << "" << " was found " << foundtime << " time(s) at
    postion as following \n";
52:     for (i= 1; i <= foundtime ; i++) {
53:         cout << " position ---> " << foundpos[i] << " \n";
54:     }
55:     cout << " \n\n\n.....All job has been done successfully.....\n\n\n";
56: }

```

ตัวอย่างการใช้งาน

หลังจาก compile โปรแกรมแล้ว เรียกใช้งาน ผ่านหน้าต่าง Dos command โปรแกรมจะให้ใส่คำที่ต้องการค้นหา ในที่นี้ต้องการค้นคำว่า love

```

C:\MyCpp\FindText>findtext
Input what you want to search for : love

```

โปรแกรมจะค้นหาคำที่เราต้องการ พร้อมแจ้งให้ทราบว่าพบคำนี้กี่ครั้งในไฟล์นี้ พบที่ตำแหน่งใดบ้าง

```

'love' was found 5 time(s) at postion as following
position ---> 55
position ---> 254
position ---> 437
position ---> 495
position ---> 695

.....All job has been done successfully.....

```

หมายเหตุ : ไฟล์ ชื่อ somewhere.txt เป็นไฟล์ของเนื้อเพลงที่ชื่อว่า Somewhere my love ไฟล์นี้จะต้องอยู่ในโฟลเดอร์เดียวกันกับ โปรแกรม Findtext.exe อยู่

## reference

1. Reisdorph, Kent. Teach Yourself Borland C++ Builder in 24 Hours., Sams Publishing, New York, 1999, 451p.
2. Breamer, Brian and Bramer, Susan. C++ for Engineers. Hodder Headline Group, London, 1996, 468p.
3. Faison, Ted. Borland C++ 3.1 Object-Oriented Programming. 2<sup>nd</sup> Sams Publishing, Indiana, 1992, 1107p.
4. Etter, Delores M. Introduction to C++ for Engineers and Scientists., Prentice-Hall, Inc., 1997, New Jersey, 162p.
5. Bell, Douglas, The Essence of Programming Using C++, Prentice-Hall, Hertfordshire, 1997, 220p.
6. Lafore, Robert. Object-Oriented Programming in Turbo C++, Waite Group, Inc. , 1991, California 739p.
7. Cannon Scott R., Understanding Programming An Introduction Using C++, West Publishing Company, Minneapolis, 1997, 479p.
8. Hubbard John R. Programming with C++, Schaum's Outline Series, McGraw Hill, Singapore, 1996, 436p.
9. Conte Paul, Commonsense C. , Duke Communication. Colorado, 1983, 100p.
10. Traister, Robert J., Going from C to C++, Academic Press, Messachusetta, 1993, 185p.
11. Schildt, Herbert. Schildt's Expert C++, Osborn McGraw Hill, 1996, California, 402p.
12. Reverchon, Alain and Ducamp Marc. Mathematical Software Tools in C++, John Wiley & Sons, 1993, England, 507p.
13. Eckel, Bruce. Thinking in C++. Prentice Hall, 2000, New Jersey, 842p.
14. Deitel H.M. and Deitel P. J. C++ How to Program., Prentice Hall, 2001, New Jersey, 1168p.
15. Shamma, Namir. C/C++ Mathematical Algorithms for Scientists and Engineer., McGraw-Hill, Inc., 1996, Singapore, 304p.
16. Gurewicz Nathan and Gurewicz Ori., Mastering C++ (from C to C++ in 2 weeks), Sybex, Inc., 1994, Singapore, 503p.
17. Horstmann Cay S., Computing Concepts with C++ Essential., 2<sup>nd</sup> edition, John Wiley & Sons, Inc., 1999, New York, 685p.