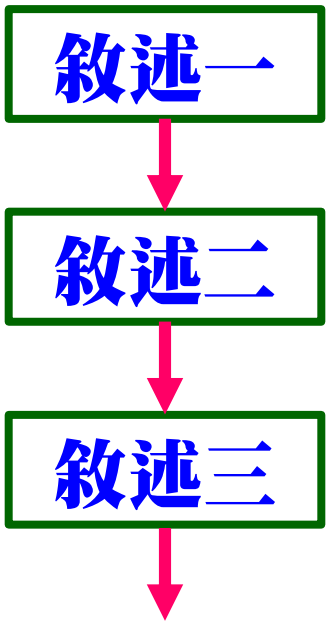
# C++程式設計

## 單元二：選擇結構

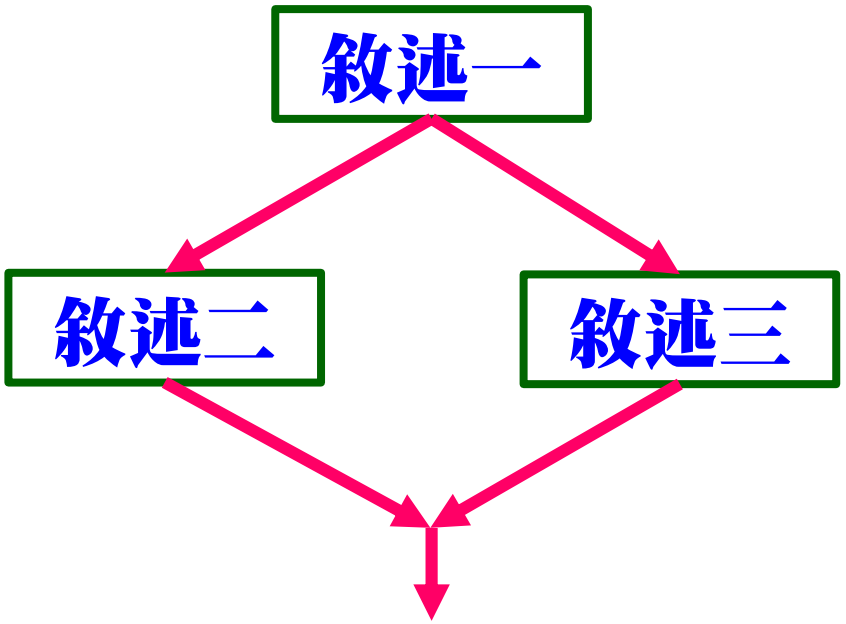
# 循序結構與選擇結構

## 循序結構




直直一條路走到底

## 選擇結構




有不同的路可走

# 選擇結構

選擇結構代表程式在執行時，會依條件適當的改變程式流程，使程式變得更聰明，C++的選擇結構可分為：

- 單一選擇
- 雙向選擇
- 多向選擇

# 單一條件控制－是非題

- 語法格式(一)

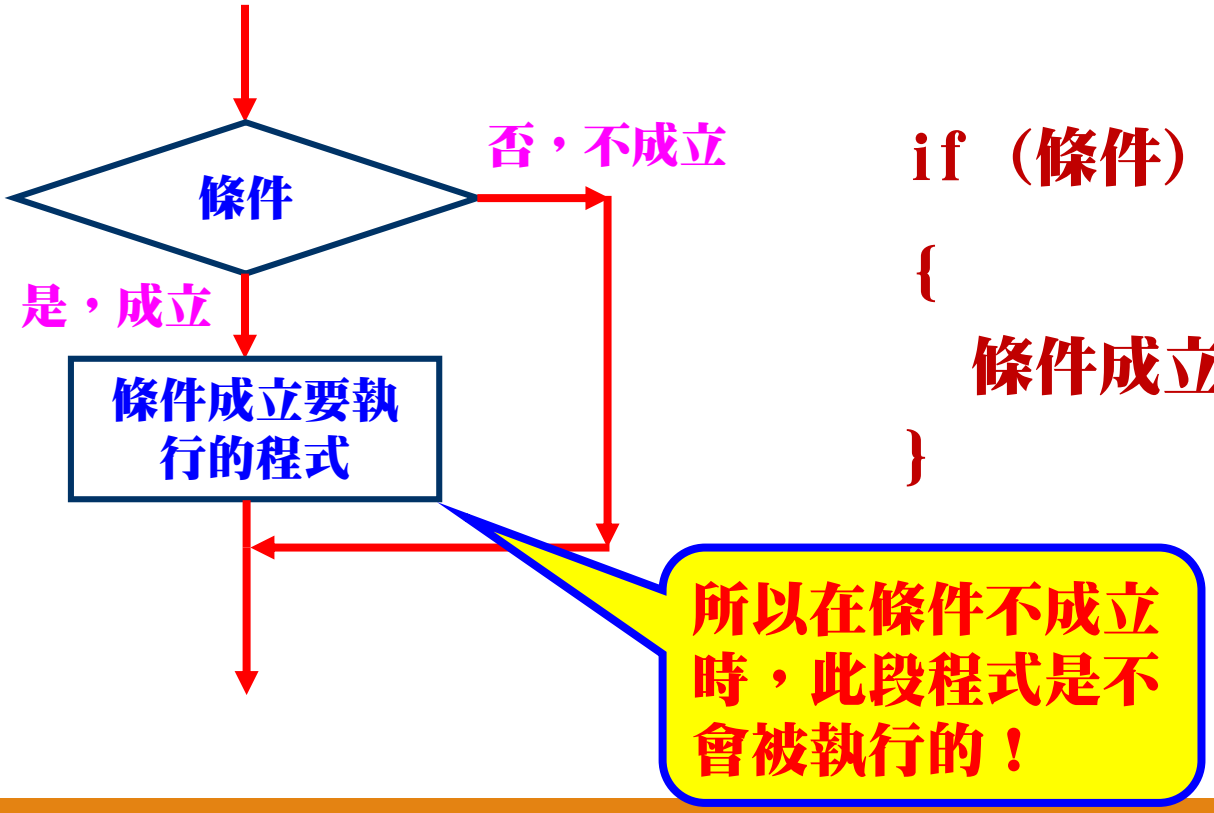
```
if(條件式) 符合條件時要執行的指令;
    (如果．．．．然後．．．．)
```

- 語法格式(二)

```
If(條件式) {
    符合條件時要執行的指令;
}
```

# 單一條件控制流程圖

- 流程圖－

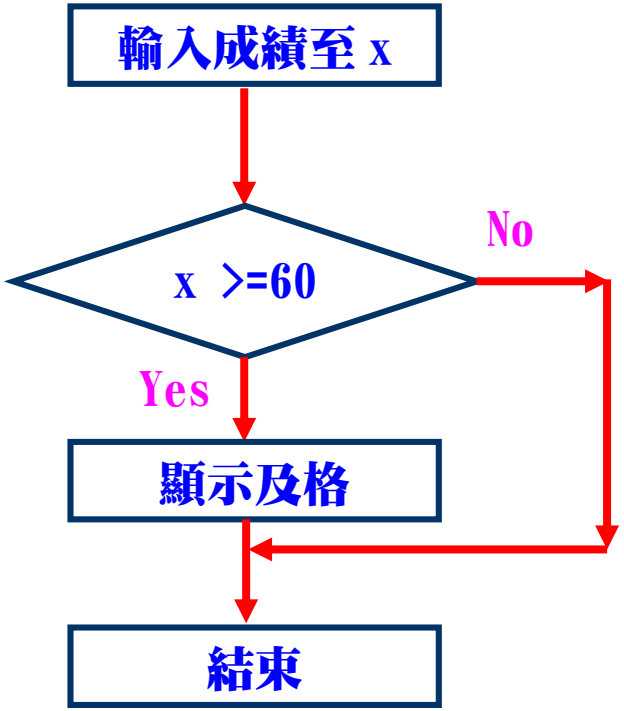



- C++程式－

```
if (條件)
{
    條件成立要執行的程式
}
```

# 單一選擇結構

## 由使用者輸入成績，若大於等於60分則顯示及格




```
main(){
   int   x;

   cout<<"輸入成績:";cin>>x;
   if ( x>=60) {
      cout<<"成績及格";
   }
   return 0;
}
```

# 雙條件控制－二選一

- 語法格式(一)

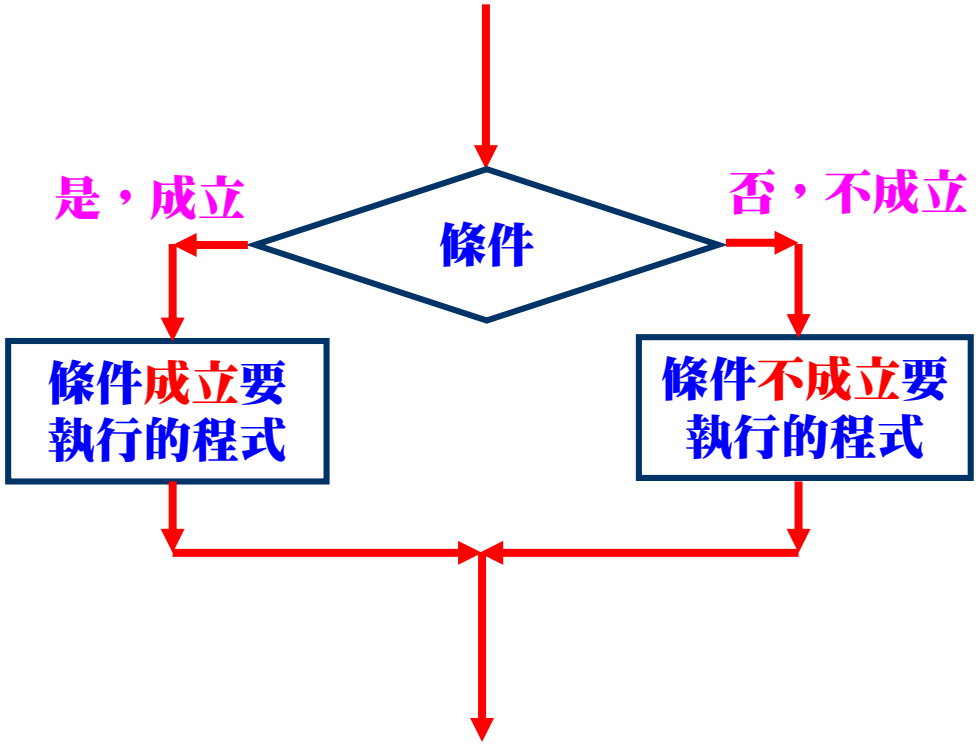
```
if(條件式) 符合條件時執行; else 不符合條件時執行;
(如果・・・・然後・・・否則・・・)
```

- 語法格式(二)

```
if(條件式){
    符合條件時執行;
}else {
    不符合條件時執行;
}
```

# 雙條件控制流程圖

- 流程圖－

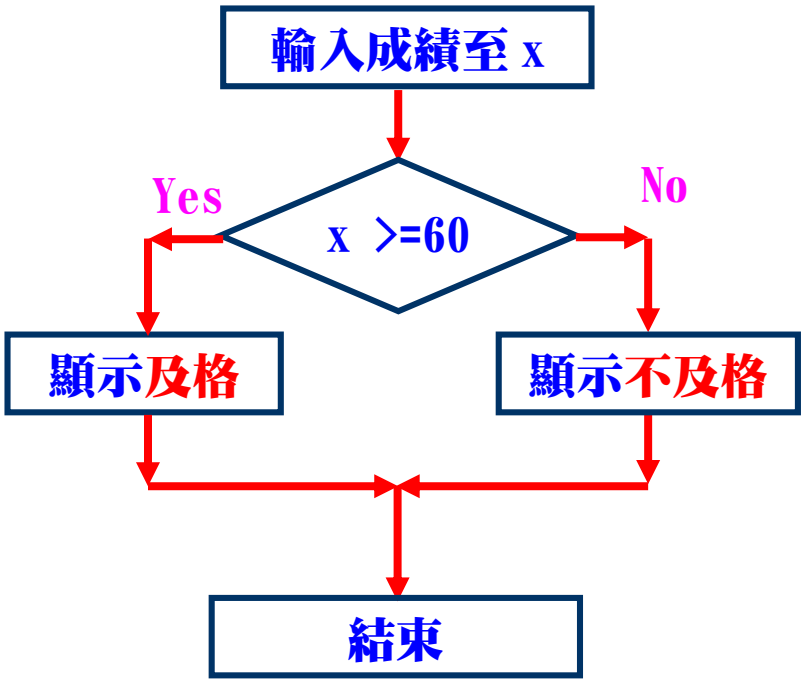



- C++程式－

```
if（條件）
{
   條件成立要執行的程式;
} else
   條件不成立要執行的程式;
}
```

# 雙條件控制

**由使用者輸入成績，若大於等於60分則顯示及格，反之顯示不及格**




```
main(){
   int   x;

   cout<<"輸入成績:";cin>>x;
   if ( x>=60) {
      cout<<"成績及格";
   }
   else {
     cout<<"成績不及格";
  }
  return 0;
}
```

# 條件控制寫四捨五入程式

```
main(){
   float   n;
   int x;
   cout<<“輸入一實數:";cin>>n;

   x=n;
   if ( n-x>=0.5)
      x++;
    cout<<n<<"四捨五入後為："<<x;
   return 0;
}
```

先宣告變數，以存放輸入之數值及轉換後的結果

輸入數值

先去尾

‘判斷如果大於等於0.5則進位

將結果顯示出來

# 字元判斷程式

```
main(){
   char   k;

   cout<<“輸入一字元:";cin>>k;

   if ( k==‘+’)
        cout<<k<<“是加號\n";
   else
        cout<<k<<“不是加號\n";
     return 0;
}
```

# 多條件控制A(if語法)－多選一

- 原if(條件) else 語法

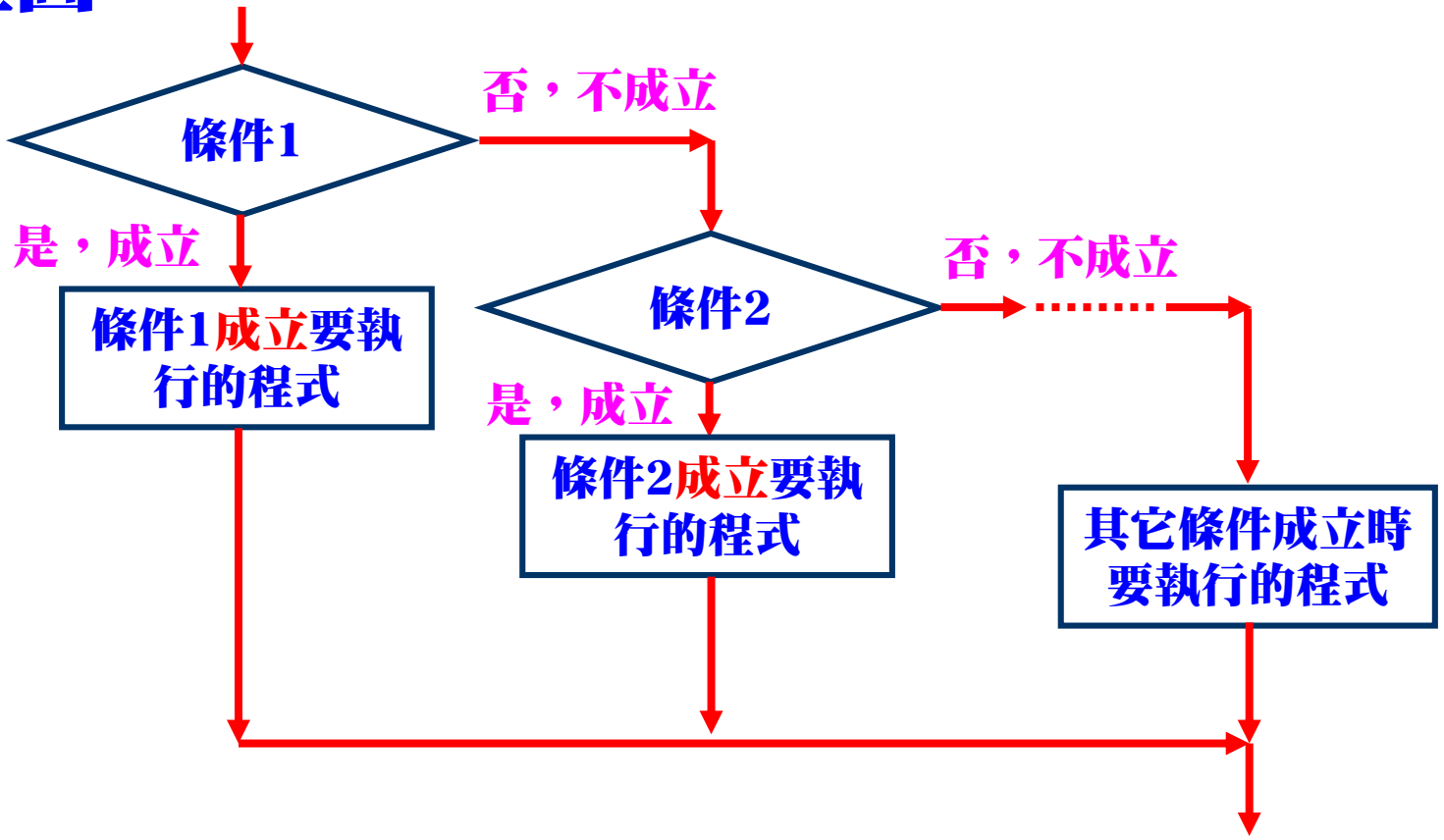
```
if(條件式){
    符合條件時執行;
}else{
    不符合條件時執行;
}
```

- 語法格式－

```
if(條件式1){
    符合條件1時執行;
}else if(條件式2){
    符合條件2時執行;
}else if(條件式3){
    符合條件3時執行;
}[Else{
    條件都不成立時執行;
}]
```

# 多條件控制流程圖

- 流程圖一



# 多條件控制範例

## 由使用者輸入成績，並將成績依不同標準給予評語

| 成績 | 評語 |
| --- | --- |
| 91～100 | 厲害 |
| 81～90 | 很棒 |
| 71～80 | 不錯哦 |
| 60～70 | 還可以 |
| 60以下 | 你被當了 |

```
main(){
   int   x;
   cout<<"輸入成績:";cin>>x;
   if ( x>=91)
      cout<<"厲害";
   else if(x>=81)
      cout<<"很棒";
   else if (x>=71)
      cout<<"不錯";
   else if(x>=60)
      cout<<"還可以";
   else
      cout<<"你被當了";
   return 0;
}
```

# 多條件控制B(switch語法)－多選一

```
switch(變數或運算式)
{
   case 選擇值1:
      符合選擇值1時執行的敘述;
      break;
   case 選擇值2:
      符合選擇值2時執行的敘述;
      break;
   case 選擇值3:
      符合條件值3時執行的敘述;
      break;
   default:
      當所有的選擇值都不合時所執行的敘述;
}
```

# switch 使用說明

1. switch指令後的括號中可放變數或運算式
   例如：switch(x) 或 switch(x+5)
2. 在每一個case中，最後一行一定要放 break指令，以便跳出該程式區塊。
3. case後的選擇值可為下列兩種表示方式：
   A. 單一值，例如：1、2、3等，需符合變數的資料型態，撰寫方式為 case 1:。
   B. 多個值，例如：1、2、3等，需符合變數的資料型態，撰寫方式為 case 1: case 2: …。
   C. 範圍值，例如：a≧變數≧b，程式撰寫時的表示方式為：case a ... b:。

# 多條件控制範例一

**讓使用者輸入一個星期的第幾天，由程式顯示對應的星期名稱（1代表星期一、其它以此類推……）**

```
main(){
   int   x;
   cout<<"輸入一個星期的第幾天:";cin>>x;
   switch(x)
   {
        case 1:
           cout<<"星期一";
           break;
        case 2:
           cout<<"星期二";
           break;
        case 3:
           cout<<"星期三";
           break;
        case 4:
           cout<<"星期四";
           break;
        case 5:
           cout<<"星期五";
           break;
        case 6:
           cout<<"星期六";
           break;
        case 7:
           cout<<"星期日";
           break;
        default:
           cout<<"輸入錯誤";
    }
}
```

# 多條件控制範例二

**讓使用者輸入月份，由程式顯示對應的季節（1代表星期一、其它以此類推……）**

```
main(){
   int   x;
   cout<<"輸入月份:";cin>>x;
   switch(x)
   {
         case 2 … 4:
            cout<<"春天\n";
            break;
         case 5 … 7:
            cout<<"夏天\n";
            break;
         case 8 … 10:
            cout<<"秋天\n";
            break;
         case 11 … 12:  case 1:
            cout<<"冬天\n";
            break;
         default:
            cout<<"輸入錯誤";
      }
      return 0;
}
```

# 雙限制輸出小數位數

- 引用標檔

```
#include <iomanip>
```

- 語法

```
cout<<fixed<<setprecision(小數位數)<<浮點數;
```

# 雙限制輸出小數位數範例

```
#include <iostream>
#include <iomanip>
using namespace std;
main(){

   float   x=3.6789;

   cout<<“小數下兩位:“<<fixed<<precision(2)<<x;

  return 0;
}
```