C++

物件導向程式語言---
程式語言可以讓我做些啥事情?

推薦參考書目:

C++ Primer 中文版 作者-Stanley B. Lippman & Josee Lajoie, 譯者-侯捷
Programming Language 中文版 作者-Bjarne, 譯者-孟岩

C++ 的議題

- 1.以程序爲基礎的程式設計(Procedural-Based Programming)
- 2.以物件爲基礎的程式設計(Object-Based Programming)
- 3.物件導向程式設計(Object-Oriented Programming)
- 4.例外處裡(Exception handling)
- 5.泛型程式設計(Generic Programming)
- 6.與視窗接口的圖形化介面 (Graphic User Interface-GUI)

物理研究工作者該如何學習程式語言?

- 1.以有效率的時間學習程式語言的基礎 (多寫, 多看, 多想)
- 2.了解創造輪子的基本方法
- 3.不要重新發明輪子
- 4.學習加快擴充知識的速度
- 5.需要把全部的 C++ 都學全嗎 ?
- 6.資料結構? 演算法? …….請善用-STL

我所要教的 C++ 有哪些部份

- 1.C++的基本元素
- 2.檔案輸入輸出
- 3.學習使用 STL (Standard Template Library)
- 4.設計可使用的物件 (以物件爲基礎的程式設計)
- 5.多載化函式，多載化運算子
- 6.泛型程式概要

1.C++語言的基本元素

- A. Consol 程式的基本架構，資料型別
  (常數,變數, 指標, 參考, 陣列)
- B.算式
  (Equality、Relational、Logical運算子)
- C.述句
  (if, switch, for, while, do while, break, continue, goto)
- D.函式
  (變數生存空間與生命週期, 函式指標)

A.Consol 程式的基本架構，資料型別：

```
#include <cstdlib>
#include <iostream>
using namespace std;

/* 這是一段註解*/

/*這也是一段註解……
    …..*/

//這依然是一段註解~~~

int main(int argc, char *argv[])
{
     cout<<“C++ is Wonderful”<<endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

基本的常數型態
123 ， ‘a’ ， ‘B’ ， 3.1415926 , “這是一段字串常數”

基本的變數型態 ---
電腦儲存資訊的方法 0,1 -> bit ，byte = 8 bites , word = 4 bytes , word 的大小取決於機器

bool -------只儲存 0 1 的變數
int , short , long ------- 整數
float , double , long double ------浮點數 (有理數)

變數加上 const 變成常數 const int = 0; , const double = 3.14159;
變數加上 unsigned 變成不帶有負號的變數 (就是正數啦)

述句結束都以 “ **;** ” 結尾
同一個述句可以折行例如, 你可以使用下面兩種方法宣告變數:

```
double = 3.14159;
double
= 3.14159;
```

宣告一個變數可以出始化其值, 或是不要初使話之:

```
int  variable_without_constructed; //一個沒有初始化的變數
int  firstly_constructed_variable = 0; //一個一開始以常數 0 的數值初始化的變數
```

Pointer(指標):

```
int a=0;
int *pb;          //一個未初始化的指標
pb=&a;            //將 pb 指向變數 a 的位址

int *pc=&a;   //以變數 a 的位址初始化指標 pc

int *pd=pc;   //以指標 pc 初始化 pd

pd=pb;            //pd 指標轉指向 pb 指標所指的位址
*pc = *pc+1;  //讓 pc 指標所指的位址的變數值 +1

 const   int   *ptr; // →這是啥??????????????
 int     *const ptr; // → 這又是啥??????????????
```

Reference(參考):

```
int  variable=0;
Int  &ref_variable=variable; //reference 變數必定要使用一個變數來初始化
 const   int    &c_ref_variable = variable; //→啥 ???
```

----------分隔線----------

問題: 有變數就夠我玩了, 我要 Pointer 或 Reference 作啥呢 ?

陣列: C++ 陣列的元素量必須在一開始決定, 沒有辦法更改

```
int  array_A[10];
const   int   N=10; //宣告一個常數整數
int  array_B[N];
int  Length=5;

array_B[0]=0;
array_B[1]=1;
array_B[2]=array_B[0]+array_B[1];

int  array_C[Length];    ←錯誤的語法, 不能夠以變數出始化陣列的長度
        //補充, 雖然最新的 C99 標準已經允許 C 語言可以使用這種語法,
        //但是還沒有普遍的  編譯器支援
int  matrix_A[10][20];
int  matrix_A[0][0] = 0;
int  matrix_A[1][1] = martix_A[0][0];
```

字串變數:

```
char *string_A= “I love you so much.”;
string_A = “How can you abandon me for that ugly girl?”;
char  ch = string_A[3];
```

爲變數命名該注意的事項

- 1.變數的名稱最好具有實質上的意義
- 2.不要害怕變數的名稱過長,因爲隱誨的變數名稱將導致你日後無法判別該變數的功能
- 3.可以適當的使用底線 “_” 用以命名, 但是不要過分使用, ex: int _____ = 0; 這種變數雖然合法, 但是很糟糕
- 4.變數名稱須區分大小寫, 最好相同的變數名稱不要同時擁有大寫與小寫版本, 以免日後搞不清楚
- 5.清楚的判別 一般變數、指標(pointer)、參考(reference)，之間功能的區別
- 6.可以的話, 在該變數的旁邊加上些許的注解
- 7.變數名稱不可與 C++ 的關鍵字重複

運算子:

算術運算子:

```
int a=10;
int b=a; // 使用指派運算子將 a 的數值傳給 b
int c=b+a; //家法運算
c++;     //讓 c 加 1
b--;      //讓 b 減 1
a = b*c; //乘法運算
c = b/a; //除法運算
c = 10 % 7 ; //取餘數
```

關係運算:

```
bool  the_result = c < a; //指派 c 是否小於 a的結果給 the_result
the_result = c > a;
the_result = a == b; //相等比較
the_result = c >= a; // 大於等於
the_result = c <= a; // 小於等於
the_result = c != a;  // 不等於
```

邏輯運算:

```
the_result = ( a==b && c<=a) //AND
the_result = ( a==b || c>=a )   //OR
the_result = !(a!=b && c>a )   //NOT → !
```

述句(Statements):

條件判斷式:

```
if ( 一些關係運算與邏輯運算 )
{
    //如果上述 () 內爲 true, 則執行這裡的動作
} //這裡不需加分號 ";"

if (一些關係運算與邏輯運算 )
{
    //如果上述 () 內爲 true, 則執行這裡的動作
}
else
{
    //如果上述 () 內爲 false, 則執行這邊的動作
}
```

while 迴圈:

```
while(一些關係運算與邏輯運算 )
{
   //如果 () 爲真, 則重複執行直到 () 內爲 false
}

do{   //先上車後補票的迴圈

  //先行執行一次, 再進行判斷, 重複執行直到 () 內爲 false

}while(一些關係運算與邏輯運算 );
```

For 迴圈 (您最好的朋友):

```
for( statement 1 ; statement 2 ; statement 3 )
{
    //statement 1 的地方宣告某些變數
    //statement 2 測驗此地的述句是否爲 true
    //statement 3 對某些變數進行修改
}
//ex: 下列程式將 Array 裡所有的元素初始化爲 0
const   int   N = 10;
 int  Array[N];
for (int index=0 ; i<N ; i++)
{
    Array[ index ] = 0;
}
```

Switch 述句 :

```
switch ( 經由外部變數給定 ) //整數變數或是字元變數
{
    case  變數1 :  //如果外部給定的變數 與 變數1 相等, 則由此開始執行
    case  變數2 :
    ….
    default:   //都沒有符合上述例子, 則執行這邊的敘述
}
```

Switch 還需要一個好朋友 → break ;

//我們可以將上述 switch 述句修改如下

```
switch ( 外部給定的變數 )
{
    case 變數 1: // 符合的話就執行該敘述
                    break;
      case 變數 2: //….
                    break;
    ……
    default:
                    break;  //← 這個break有必要嗎 ?
}
```

雖然不一定要再 default 之後加上一個 break, 因爲它就擺在最後面, 但是最好還是加上一個, 以免在日後有更多的敘述, 有可能讓你忘記該敘述並沒有 break 那麼就會隱藏著邏輯上的錯誤

列舉型別 (enum) :

```
enum  My_Book { Harry_Porter,  Little_Prince, PlayBoy };

My_Book   aBook = PlayBoy;

switch ( aBook )
{
    case  Harry_Porter:   cout<<“Good”;     break;
    case  Little_Prince:   cout<<“Push”;    break;
    case  PlayBoy:         cout<<“Ouch”;   break;
}
```

Break述句:

```
//break 述句可以出現於迴圈與 switch 述句中  ex:
for( int index=0 ; index<10 ; index++)
{
    if( index == 5 ) break;
}
//break 一旦被執行, 整個回圈將會終止
```

Continue述句:

```
//continue 只能出現於迴圈當中  ex:
int var;
while ( cin>>var )
{
    if( var == 0 ) continue;
    // 做點事情
    // 如果 continue 被執行, 這邊就不會有動作
    if( var == 1 ) break;
}
//continue 一旦被執行, 只會終止該次回圈以後的敘述
//但是之後還是會繼續完成該回圈
```

Goto述句:

```
Label1:
// 做了某些事情
 if(…)goto Label1;
If(…)goto Label2;

Label2:
```

撰寫述句的時候需要注意的:

- 1.注意大括弧前後對應的關係
- 2.創造良好的縮排風格, 並且養成習慣

- 3.除非不得已, 千萬不要使用 goto
- 4. goto 的使用時機在於跳出巢狀迴圈

函式 (Function)

- 函式的架構
- 函式的引入值
- 函式回傳的數值
- 函式宣告的順序
- 區域變數 v.s 全域變數