C++基礎語法3

鄭詠堯AKA沒有公關力只有一點點學術力的公關

Today力課程

函式、結構及一些題目，你們會寫力。

遞迴，希望我講的到這邊QQ

\學弟你們好電/

結構(struct)

在寫程式時，很常會遇到一份資料是不只一個數值的(例如坐標系)，此時如果要開兩個不同的變數去儲存總會有點麻煩，因此便有了結構這個概念，能讓你把多種資料放在一起使用。

struct的建構

比如說我想記錄一個班的英文成績，那我需要同時儲存名字跟成績，此時就可以利用struct。另外struct通常會在全域建構，也就是`int main(){}`以外的區域。而語法是`struct 結構名稱 {`裡頭會有的變數`};`

```
struct scores{
    string name;
    int english;
};
```

struct的使用

在建構完一個結構後，就可以直接將結構名字當作一個變數型態來使用了，也就是跟int,char,float那些相同的用法。

而當要存取那些子資料時，便是使用 變數.子結構名稱 來像一個普通變數一樣存取。

//這一頁放不下code力

前一頁的實作

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

struct scores{
    string name;
    int english;
}; //建構scores結構，包含字串name及整數變數english

int main(){
    scores pianxiang; //宣告scores變數pianxiang

    pianxiang.name = "yungyaocheng"; //將pianxiang的name指派為"yungyaocheng"
    pianxiang.english = 59;  //將pianxiang的english指派為59

    return 0;
}
```

搭配陣列使用

自己建構出來的變數型態，運作起來就跟其他變數一樣，而像要記錄不只一筆紀錄時，自然就會搭配陣列去使用。

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

struct scores{
    string name;
    int inforScore;
}; //建構scores結構，包含字串name及整數變數inforScore

int main(){
    int n;
    scores ckefgisc[1000]; //宣告大小為1000的scores陣列ckefgisc

    cin >> n; //輸入n
    for (int i=0;i<n;++i){ //代表這個迴圈會跑n次，且i的值依序會是0到n-1
        cin >> ckefgisc[i].name >> ckefgisc[i].inforScore; //輸入ckefgisc第i項的name及inforScore
        ckefgisc[i].inforScore += 10; //把第i項的inforScore加10
    }
    for (int i=0;i<n;++i){ //代表這個迴圈會跑n次，且i的值依序會是0到n-1
        cout << ckefgisc[i].name << ' ' << ckefgisc[i].inforScore << '\n';
        //輸出ckefgisc第i項的name及inforScore
    }

    return 0;
}
```

函式

函式有點類似數學學過的函數，也就是當給予一個或多個值時，就會執行某些事情並回傳一個值。

在C++中有一些已經定義好給我們使用的函式，當然也可以按自己的需求去定義一個函式。

自定義的函式

自定義一個函式跟宣告變數相似，都要先定義型態，在函式中代表的則是回傳值的型態，如果沒有回傳值則用`void`替代。而在變數後面則會用`()`包住需要傳入的值(這裡可以視同在函式裡宣告一個變數，並將初始值設為傳入值)，最後在`{}`中放入執行內容。而除非是`void`型態，否則依定要用 `return 回傳值;` 來回傳並表示函式結束。

示範code

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

int demoFunction(int x){ //宣告demoFunction，且傳入值是x
    if (x < 0){
        return 0; //若x < 0，則回傳0
    }
    else{
        return x; //否則回傳x
    }
}

int main(){
    int n;
    cin >> n;
    cout << demoFunction(n); //輸出經過demoFunction的n

    return 0;
}
```

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

int bigger (int x,int y){//宣告整數函式bigger，傳入值為int x,int y
    if (x > y){
        return x;
    }
    else if(y > x){
        return y;
    }
    else{
        return 0;
    }
}

int main(){
    int n1,n2;  //宣告整數變數n1,n2
    cin >> n1 >> n2;//輸入n1,n2

    cout << bigger(n1,n2); //輸出經過bigger的n1,n2

    return 0;
}
```

如何透過函式來更改變數

如果用前一頁的方式來定義函式的話，就會發現你對函式中的值修改，都不會影響到函式外的值。而當想要定義一個函式去交換兩個變數的值，就不能用前一頁中的方式(傳值,pass-by-value)，可以使用傳址,pass-by-address 或 傳參考,pass-by-reference。

傳參考

由於傳址會利用到下禮拜才會講的指標概念，因此這周只會講到傳參考。

若要傳參考的話，便是在定義函式時，在()內宣告傳入值時，在變數名稱前加入&，以int型態為例便是 `int &x` ，這樣就會變成傳入變數的參考，而在函式內對變數修改就能夠連帶影響函式外傳入的變數。

//很抽象ㄉ話我很抱歉，這邊比較難懂啦><

Try try看

要怎麼將兩個變數交換？

Hint:當你第一個變數指定為第二個變數時，第一個變數原有的值就消失ㄌ

答案

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

void swap(int &x,int &y){ //宣告無回傳值函式swap，且傳入的是int x,int y的參考
    int t;
    t = x;
    x = y;
    y = t;
}

int main(){
    int n1,n2;  //宣告整數變數n1,n2
    cin >> n1 >> n2; //輸入n1,n2
    swap(n1,n2);   //執行swap n1及n2
    cout << n1 << ' ' << n2; //輸出n1,n2

    return 0;
}
```

遞迴函式

遞迴函式就是指在函式中再呼叫一次函式本身，而用這種做法可以省下許多麻煩的coding，譬如若要算出x的階乘，其實x! = x * (x-1)!，再以此往下類推到1!為止，此時用遞迴去寫就很方便。

Demo code

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

int factorial (int x){
    if (x == 1){
        return 1; //若x為!，回傳1
    }
    else{
        return x * factorial(x-1);
    } //否則回傳x乘以x-1的階乘
}

int main(){
    int n;  //宣告整數變數n
    cin >> n;//輸入n

    cout << factorial(n); //輸出n的階乘

    return 0;
}
```

排版很醜我很抱歉OAO

Try try看

Ckeisc oj 009(X)，這題用遞迴在oj會超時，不要丟上去QAQ

以下是補充ㄛ

上不到就算ㄌ，回家自己讀辣(上的到的話有點可怕)

pair

pair是一個在utility標頭檔內的資料型態，使用前需要打上一行#include <utiliy>，而pair則是一種資料型態，他的特色是他有點像struct，但不需要寫整段struct語法，只要

pair <資料型態1,資料型態2> 變數名稱;

另外存取資料的方式也跟struct類似，左邊的資料是.first，右邊的是.second。同時也可以用make_pair(左邊資料,右邊資料)去快速賦值。

常用函式

`max(int,int),min(int,int)` 回傳比較大or小的值

`pow(int,int)` 回傳左邊int的右邊int次方

`abs(int)` 回傳int的絕對值