Yuan Ze University

C++程式初探 V

2015暑期

lobal Logistics Lab.

ver. 1.0.1

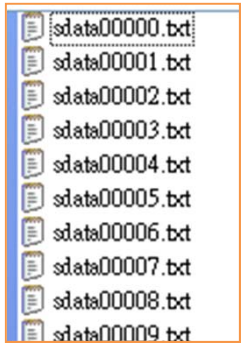
C++ 程式語言 – 大綱

1. 大量檔案讀取&計算
2. 指標
3. 動態記憶體 & **動態陣列**
4. 標準函式庫(STL) – vector, algorithm
5. 結構與類別

大量檔案讀取&計算

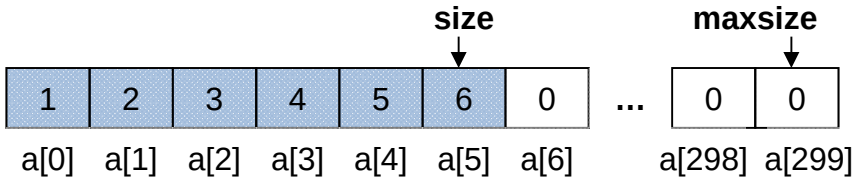
若目前有一個程式將讀取純文字文件(.txt) 中的整數，並將該文件中的整數有小到大排序後，儲存到另外一個新的純文字件中。假設有20個純文字文件，每個文件格式包含兩項資訊為：

1. 整數數量大小(size, 已知最大數量為300)
2. 欲排序之整數(data)



大量檔案讀取&計算 – 給定最大陣列

```
#include<iostream>
#include<fstream>
#include<algorithm>
using namespace std;
int main(void)
{
    const int maxsize = 300;
    int  size;
    int  data[maxsize];

    ifstream sinf("sdata\\sdata00000.txt");
    sinf>>size; //dynamic size is determined by reading file
    for(int i=0; i<size; i++) {
        sinf>>data[i];
    }
    //sorting
    sort(data, data+size);
    //output on the screen
    for(int i=0; i<size; i++) {
        cout<<data[i]<<" ";
    }
    system("PAUSE");
    return 0;
}
```



大量檔案讀取&計算 – 產生動態的檔名

```
#include<iostream>
#include<fstream>
#include<sstream>
#include<iomanip>
#include<algorithm>
using namespace std;
int main(void)
{
    const int maxsize = 300;
    int  size;
    int  data[maxsize];
    int  filenum = 20;

    for(int n=0; n<filenum; n++) {
        // set the name rule for reading file
        stringstream ifname;
        //give the path as "sdata\\sdata00000.txt"
        ifname<<"sdata\\sdata"<<setw(5)<<setfill('0')<<n<<".txt";

        //read file
        ifstream sinf(ifname.str().c_str());
    }

    system("PAUSE");
    return 0;
}
```

引入標頭檔

```
#include<sstream>
```

sstream物件

```
stringstream  sstr;
```

sstream物件使用

```
sstr<<"str"<<36<<var;
```

sstream物件轉換為c style字串

```
sstr.str().c_str();
```

大量檔案讀取&計算 – 產生動態的檔名

```
#include<iostream>
#include<fstream>
#include<sstream>
#include<iomanip>
#include<algorithm>
#include<direct.h>
using namespace std;
int main(void)
{
    const int maxsize = 300;
    int  size;
    int  data[maxsize];
    int  filenum = 20;

    mkdir("result"); //create new folder
    for(int n=0; n<filenum; n++) {
        // set the name rule for reading file
        stringstream ifname;
        //give the path as "sdata\\sdata00000.txt"
        ifname<<"sdata\\sdata"<<setw(5)<<setfill('0')<<n<<".txt";

        //read file
        ifstream sinf(ifname.str().c_str());

        //接續下一頁.....
```

引入標頭檔VC++

```
#include<direct.h>
```

mkdir函式

```
mkdir("folder_name");
```

大量檔案讀取&計算 – 產生動態的檔名

```
        //接續上一頁......
        sinf>>size; //dynamic size is determined by reading file
        for(int i=0; i<size; i++) {
            sinf>>data[i];
        }
        sort(data, data+size);

       // give the name for output file
       stringstream ofname;
       ofname<<"result\\rsdata"<<setw(5)<<setfill('0')<<n<<".txt";

       //output file for sorted data
       ofstream routf(ofname.str().c_str());
       for(int i=0; i<size; i++) {
           routf<<setw(5)<<data[i];
       }
       sinf.close();
       routf.close();
    }

    system("PAUSE");
    return 0;
}
```

指標 (pointer)

```
int  var = 4;
int* ptr = &var;
cout<<&var<<endl; //12FF88
cout<<ptr<<endl;  //12FF88
Cout<<*ptr<<endl; //4
Cout<< var<<endl; //4

int* ptr2;
ptr2 = &var;
```

指標

```
int*  ptr;
int*  ptr = &var;
```

name	var
address	0012FF88
value	4

name	ptr
address	0012FF80
value	0012FF88

取址運算子

&

指標

```
int  var = 4;
int* ptr = &var;
cout<<var<<endl; //4
*ptr = 5;
cout<<var<<endl; //5
Var = 6;
cout<<*ptr<<endl; //6
```

指標

```
int*  ptr;
int*  ptr = &var;
```

name	var
address	0012FF88
value	4

name	ptr
address	0012FF80
value	0012FF88

取址運算子

&

取值運算子

*****

問題 1

```
//*(&var) 等於 *(ptr)
cout<<*&var<<endl; //4
cout<<&*var<<endl; //invalid

cout<<*&ptr<<endl; //0012FF88
//&(*ptr) 等於 &(var)
cout<<&*ptr<<endl; //0012FF88
```

name	var
address	0012FF88
value	4

name	ptr
address	0012FF80
value	0012FF88

指標 – 指標做為函式引數(pointer as argument)

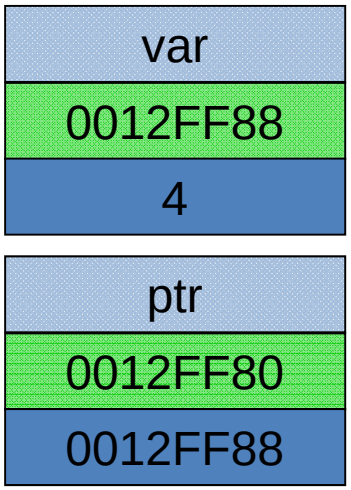
```
void swap_ref(int& a, int& b) {
    int temp = a;
    a = b;
    b= temp;
}

void swap_ptr(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b= temp;
}
```

指標 – 指標做為函式引數

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

int main(void)
{
    int a =3, b=5;
    cout<<a<<" "<<b<<endl;
    swap_ref(a,b);
    cout<<a<<" "<<b<<endl;
    swap_ptr(&a,&b);
    cout<<a<<" "<<b<<endl;

    system("PAUSE");
    return 0;
}
```

指標 – 指標運算

```
int  var = 4;
int* ptr = &var;
cout<<ptr<<endl;   //12FF88
cout<<--ptr<<endl; //12FF84
//(-4 因為int大小為4bytes)
cout<<++ptr<<endl; //12FF8c
//(+4 因為int大小為4bytes)
cout<<ptr-5<<endl; //12FF74 (-4)
cout<<ptr+5<<endl; //12FF9c (+20)
```

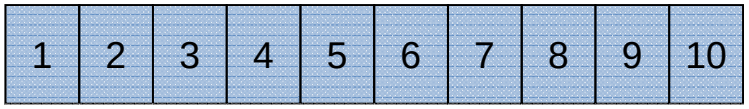


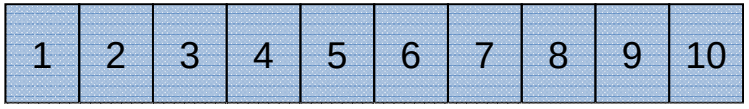
指標運算

```
ptr=ptr+5;
ptr=ptr-5;
ptr+=5;
Ptr-=5;
ptr++;
Ptr--;
```

指標 – 陣列與指標間的關係

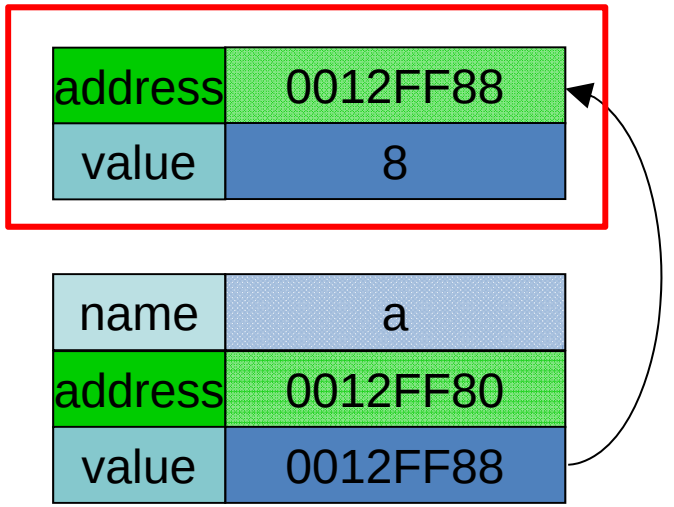
```
#include<iostream>
using namespace std;

int main(void)
{
    int a[]={1,2,3,4,5,6,7,8,9,10};
    cout<<*a<<endl;       // 1
    cout<<*(a+1)<<endl;   // 2
    cout<<*(a+2)<<endl;   // 3
    cout<<*a+1<<endl;     //

    for(int i=0, i<9; i++){
        cout<<*(a+i);
    } //

    system("PAUSE");
    return 0;
}
```





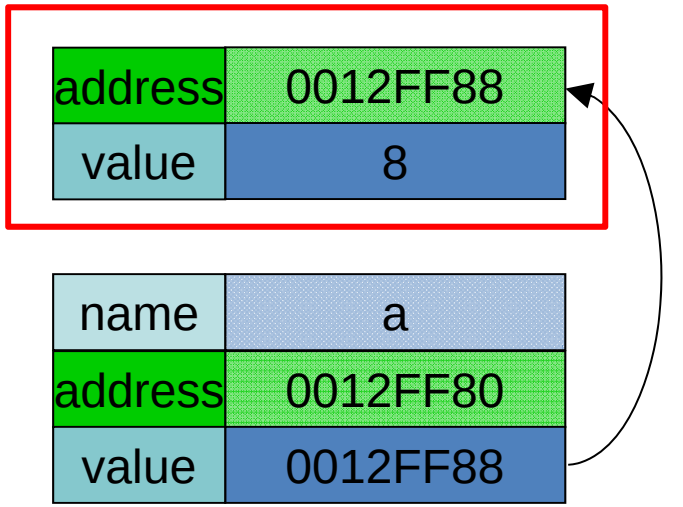
動態記憶體 & 動態陣列

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

int main(void)
{
    // 動態記憶體(new & delete)
    int *a = new int;     // 配置記憶體
    *a = 8;
    cout<<*a<<endl;       // 8
    delete a;           // 釋放記憶體

    system("PAUSE");
    return 0;
}
```



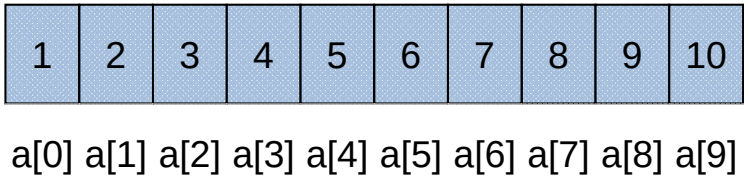
動態記憶體

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

void fun(){
     int *a = new int;      // 配置記憶體
     *a = 8;
     cout<<*a<<endl;
}  // 未釋放記憶體！！！

int main(void)
{
     fun();

     system("PAUSE");
     return 0;
}
```



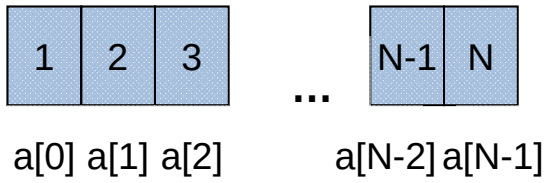
動態陣列 – 一維陣列

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

int main(void)
{
    // 一維動態陣列
    int N=10;       // 不需為const！！！
    int *a = new int [N];     // 配置記憶體
    for(int i=0, i<9; i++){
        a[i]=i+1;
    }
    for(int i=0, i<9; i++){
        cout<<a[i]<<endl;
    }
    delete [] a;              // 釋放記憶體

    system("PAUSE");
    return 0;
}
```



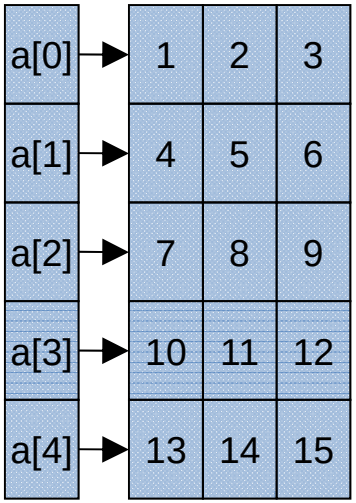
動態陣列 – 一維陣列

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

int main(void)
{
    // 一維動態陣列
    int N;
    cin>>N;                     // 動態決定

    int *a = new int [N];       // 配置記憶體
    for(int i=0, i<N; i++){
        a[i]=i+1;
    }
    for(int i=0, i<N; i++){
        cout<<a[i]<<endl;
    }
    delete [] a;                // 釋放記憶體
    system("PAUSE");
    return 0;
}
```



動態陣列 –二維陣列

```
int main(void)
{
    // 二維動態陣列
    int N=5, M=3;
    int **a = new int* [N];     // 配置記憶體
    for(int i=0, i<N; i++){
        a[i]= new int [M];
    }
    for(int i=0, i<N; i++){
        for(int i=0, i<M; i++){
            a[i][j] = i*M+j+1;
        }
    }
    for(int i=0, i<N; i++){// 釋放記憶體
        delete [] a[i];
    }
    delete [] a;
    system("PAUSE");
    return 0;
}
```



標準程式庫(standard template library; STL)

```
#include<iostream>
#include<fstream>
#include<string>

using namespace std;

int main(void)
{
    cout<<"STL"<<endl;

    system("PAUSE");
    return 0;
}
```

STL命名空間

```
using namespace std;
```

標準程式庫(STL) – vector 動態陣列

```
#include<vector>
using namespace std;

int main(void)
{
    vector<int> vct1(10);
    vector<int> vct2(10,2);
    vector<int> vct3;
    vct3.resize(10);

    int size=10;
    vector<int> vct4(size);
    vct4.resize(12);

    system("PAUSE");
    return 0;
}
```

vector 宣告

```
vector<type> var
```

標準程式庫(STL) – vector 動態陣列

```
#include<vector>
#include<iostream>
using namespace std;
int main(void)
{
    vector<int> vct(10);
    int size = vct.size(); //陣列大小
    vct[0]=100; //存入陣列
    cout<<vct[0]<<endl; //取得陣列數值

    system("PAUSE");
    return 0;
}
```

標準程式庫(STL) – vector 動態陣列

```
#include<vector>
#include<iostream>
using namespace std;
int main(void)
{
    vector<int> vct(3,2);
    vct.push_back(6); vct.insert(vct.begin()+3, 5);
    vct.erase(vct.begin()+4);

    system("PAUSE");
    return 0;
}
```

標準程式庫(STL) – 多維vector 動態陣列

```
#include<vector>
#include<iostream>
using namespace std;
int main(void)
{
    //宣告3*4的vector陣列 – 方法1
    vector<vector<int> > vec2D_1(3);
    for(int i=0; i<3; i++) {
        vec2D_1[i] = vector<int>(4);
    }

    //宣告3*4的vector陣列 – 方法2
    vector<vector<int> > vec2D_2(3,vector<int>(4));

    system("PAUSE");
    return 0;
}
```

標準程式庫(STL) – algorithm

```
#include<algorithm>
```

方法	說明
sort	**排序**
random_shuffle	亂數排序
next_enumeration	下一個列舉數
prev_enumeration	上一個列舉數
find	搜尋

結構(structure)

結構宣告

```
struct name
{
      ......
};
```

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

struct customer
{
    double x;
    double y;
    int demand;
};
void main(void){
    customer client;
    client.x = 10.5;
    client.y = 4.3;
    client.demand = 40;
    cout<<"the customer at ("<< client.x
        <<","<< client.y <<") "
        <<"has demand of "<< client.demand <<endl;

    system("PAUSE");
}
```

結構 – 結構陣列

```
struct customer
{
    double x;
    double y;
    int demand;
};

void main(void){
    customer client[5];
    system("PAUSE");
}
```

結構與類別(class)

```
class customer
{
public:
    double x;
    double y;
    int demand;
};
//兩者的宣告意義相同
struct customer
{
    double x;
    double y;
    int demand;
};
```

結構宣告

```
class name
{
public:
    ......
protected:
    ......
private:
    ......
};
```

練習 1

練習建構一個學生資料的類別(結構)，包含：

1.學生學號

2.學生姓名

3.國文成績

4.數學成績