

105學年 資料結構TA課

類別(Class) x 樣板(Template) x 結構(Struct)

Outline

- 類別(Class)
 - ◆ 存取權限
 - ◆ 建構子與解構子
- 樣板(Template)
- 結構(Struct)

類別Class

- 類別要做什麼？
將資料與函數封裝，使用者只需知道方法，不需要知道如何實作。

```
int n;  
Stack S = Stack();  
while(cin >> n)  
{  
    if (n == 1) S.push();  
    if (n == 2) S.pop();  
    if (n == 3) S.list();  
    if (n == 4) S.clear();  
}
```

```
class Stack  
{  
private:  
    int *box;  
    int size, ptr;  
public:  
    Stack();  
    ~Stack(){};  
    void push();  
    void pop();  
    void list();  
    void clear();  
};
```

類別Class

- 建立一個新的類別：

存取權限

私有



保護



公開



```
class 類別名稱
```

```
{
```

```
private:
```

```
protected:
```

```
public:
```

```
    類別名稱();
```

```
    ~類別名稱();
```

```
};
```

建構子

解構子

記得要加分號

class_demo.cpp

存取權限

存取權限	意義	存取對象
Private	私有	該類別的成員函數 該類別的friend類別和friend函數
Protected	保護	該類別的成員函數 繼承該類別的衍生類別之成員函數
Public	公開	任何函數皆可存取

存取權限

當你想要存取類別中private的變數時
會出現類似以下的錯誤

```
class_stack.cpp:66:15: error: 'size' is a private member of 'Stack'
    cout << S.size << endl;
                  ^
class_stack.cpp:10:9:      declared private here
    int size, ptr;
    ^
1 error generated.
```

建構子與解構子

- 建構子(Constructor)
進行物件的初始化。
- 解構子(Destructor)
當此物件使用完畢後，進行善後。
例如：刪除動態陣列，進行檔案寫入等等。

樣板Template

- 什麼是樣板？
樣板又稱模板，可以適用於不同資料型態的函數或類別。
- 優點？
可以避免撰寫不同型態但功能相同的程式碼。

樣板Template

- 舉個例子：

我有兩個不同型態的陣列，然後要印出陣列的所有元素。

```
template <class T>
void list(T a[], int size)
{
    for (int i = 0; i < size; ++i)
        cout << a[i] << " ";
    cout << endl;
}
```

```
int a[5] = {1, 2, 3, 4, 5};
string s[3] = {"I", "Like", "Inori."};

list(a, 5);
list(s, 3);
```

```
1 2 3 4 5
I Like Inori.
```

template_demo.cpp

結構Struct

- 結構？

把相關的變數綁在同一個名稱底下，且可包含多種不同的資料型態。

```
struct 結構名稱  
{  
    資料型態1 變數名稱1;  
    資料型態2 變數名稱2;  
    ...  
};
```

結構Struct

- DEMO :
一個學生的屬性有名字、電話、學號。

```
struct student
{
    string name;
    string phone;
    int id;
};
```

```
student A = {"A", "0912345678", 40044123};

cout << A.name << endl;
cout << A.phone << endl;
cout << A.id << endl;
```

struct_demo.cpp

「總而言之，這個章節太深奧了，講個皮毛而已，
請大家多多Google，因為沒有問題是假的。」



TenTail