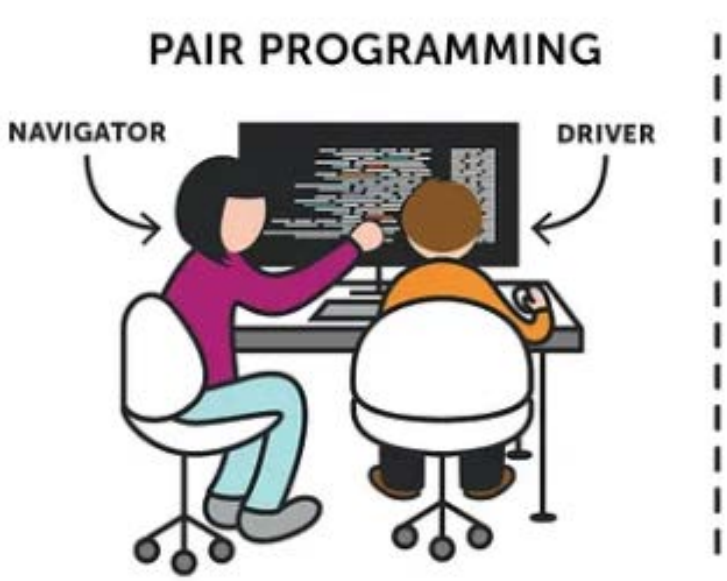
TAIPEI TECH 國立臺北科技大學 National Taipei University of Technology

Object Oriented Programming

Code Section: Class

Sun Chin-Yu（孫勤昱）
cysun@ntut.edu.tw
2023/10/02



Information Security LAB

目標

- 實作一個簡單的List類別
  - 創建一個指定大小的List
  - 使用List類別來存放一些整數，並把他列印出來
  - 設定List中每一位置的元素值
    - Ex. SetElement(index, value)
  - 取得List中某一位置的元素值
    - Ex. GetElement(index)
  - 取得List的Size
    - Ex. GetSize()

要求

- 實作list.hpp
- 實作list.cpp
- 實作main.cpp
- Output:

```
 6      List list(5);
 7      list.SetElement(0, 1);
 8      list.SetElement(1, 2);
 9      list.SetElement(2, 3);
10      list.SetElement(3, 4);
11      list.SetElement(4, 5);
```

```
C++ List
1 2 3 4 5
```

```
 6      List list(6);
 7      list.SetElement(0, 7);
 8      list.SetElement(1, 7);
 9      list.SetElement(2, 0);
10      list.SetElement(3, 7);
11      list.SetElement(4, 1);
12      list.SetElement(5, 5);
```

```
C++ List
7 7 0 7 1 5
```

List.hpp

- 建立保護機制
  - Include guards
- 定義List類別的
  - 成員
  - 建構子
  - 解構子
  - 方法

```
1  #ifndef LIST_HPP
2  #define LIST_HPP
3
4  class List {
5  public:
6      int m_Size;
7      int *m_Data;
8
9      List(int size);
10
11     ~List();
12
13     int GetSize();
14     int GetElement(int index) const;
15     void SetElement(int index, int value);
16 };
17 #endif
```

List.cpp

- 引入List類別的宣告
- 建構子
  - 它接受一個參數，表示列表的大小。在建構子內部，m_Size成員變數被賦值，並為m_Data分配了動態記憶體空間
- 解構子
  - 這是List類別的解構子。它釋放m_Data所指向的動態記憶體空間。

```cpp
#include "List.hpp"

List::List(int size) {
    m_Size = size;
    m_Data = new int[size];
}

List::~List() {
    delete[] m_Data;
}

int List::GetSize(){
    return m_Size;
}

int List::GetElement(int index) const {
    return m_Data[index]; // no error checking here
}

void List::SetElement(int index, int value) {
    m_Data[index] = value;
}
```

main.cpp

- 引入List類別的宣告
- 額外還需要一個iostream來處理cin/cout
- 主程式中
  - 我們要先設定List的大小
  - 然後用SetElement來給值
  - 處理Cout: C++ List
  - 處理Cout: List裡面的值
    - 用GetSize來處理for迴圈
    - 用GetElement來處理List裡面的值
    - 記得每個值後面應該有一個空格
    - 最後有個換行

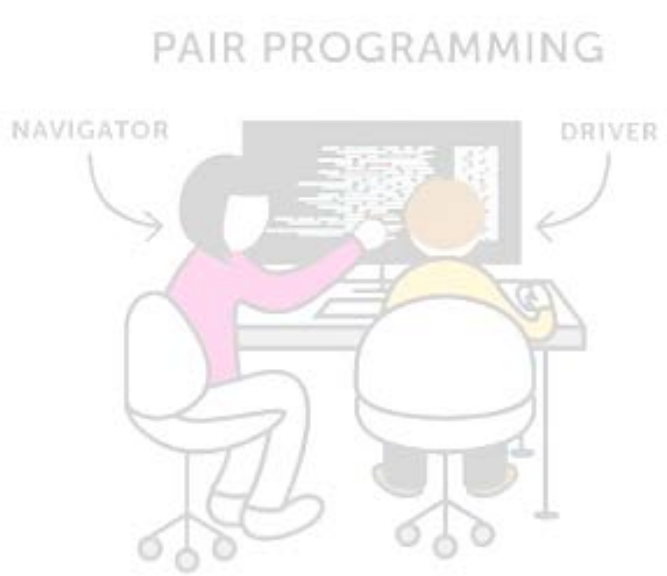
```cpp
#include <iostream>
#include "List.hpp"

int main() {
    List list(5);
    list.SetElement(0, 1);
    list.SetElement(1, 2);
    list.SetElement(2, 3);
    list.SetElement(3, 4);
    list.SetElement(4, 5);

    std::cout << "C++ List\n";
    for (int i = 0; i < list.GetSize(); i++) {
        std::cout << list.GetElement(i) << " ";
    }
    std::cout << "\n";
}
```

回顧與檢查

Code Section: Class



Public vs. Private

- 類別成員的可訪問性
  - Public：被標記為public的成員可以從類別產生的任何物件訪問，還可以被該類別之外的code訪問
  - Private：被標記為private的成員只能在其所屬的類別內部被訪問

- 「封裝」單元介紹

```
#ifndef LIST_HPP
#define LIST_HPP

class List {
public:
    List(int size);

    ~List();

    int GetSize() const { return m_Size; }
    int GetElement(int index) const;

    void SetElement(int index, int value);

private:
    int m_Size;
    int *m_Data;
};

#endif
```

Const

- 在成員函數的宣告或定義的末尾加上 const，表示這個函數不會修改這個物件的任何資料成員
  - **提供更好的保證：**通知其他程式設計師，這個函數不會更改物件的狀態
  - **增加程式的健壯性：**當一個函數被標記為 const，編譯器會幫忙確保你不會在這個函數裡更改任何成員資料

```
#ifndef LIST_HPP
#define LIST_HPP

class List {
public:
    List(int size);

    ~List();

    int GetSize() const { return m_Size; }
    int GetElement(int index) const;

    void SetElement(int index, int value);

private:
    int m_Size;
    int *m_Data;
};

#endif
```

Copy constructor

```cpp
#include <iostream>

#include "List.hpp"

void func(List list) {
    std::cout << "C++ List\n";
    for (int i = 0; i < list.GetSize(); i++) {
        std::cout << list.GetElement(i) << " ";
    }
    std::cout << "\n";
}

int main() {
    List list(5);
    list.SetElement(0, 1);
    list.SetElement(1, 2);
    list.SetElement(2, 3);
    list.SetElement(3, 4);
    list.SetElement(4, 5);

    func(list);
}
```

```cpp
#ifndef LIST_HPP
#define LIST_HPP

class List {
public:
    List(int size);

    ~List();

    int GetSize() const { return m_Size; }
    int GetElement(int index) const;

    void SetElement(int index, int value);

private:
    int m_Size;
    int *m_Data;
};

#endif
```