

# Introduction to Programming (II) Project 2

## I. 功能要求

完成 **Vector**、**List** 與 **Set**。此外，以下是各個 container 的額外要求

- **Vector**

- 記憶體上的配置必須是連續的
- `erase`、`pop_back` 與 `pop_front` 不影響刪除位置前的 iterator 與 reference
- 使用 `insert`、`push_back` 與 `push_front` 後，若沒有 `reallocation`，則不影響插入位置前的 iterator 與 reference
- `reallocation` 是指分配一塊新的 memory，並把舊 memory 的 element 搬移到新的 memory

- **List**

- `erase`、`pop_back` 與 `pop_front` 只影響刪除位置的 iterator 與 reference
  - ◆ 傳入 `erase` parameter 的 iterator，在這些 function 執行完後，\*iterator 是不安全的
- `insert`、`push_back` 與 `push_front` 不影響 iterator 與 reference

- **Set**

- `erase` 與 `insert` 的要求如 **List**
- element 從 `begin()` 到 `end()`，key 必須從小到大（根據 `key_compare` 決定大小）
- 不影響 iterator 的意思為，使用 function 前後，該 iterator 仍指向同一個物件
- 不影響 reference 的意思為，使用 function 前後，該 reference 仍 bind 同個記憶體位置的物件
- 您可以選擇不實作 `capacity`、`reserve` 與 `shrink_to_fit`
  - 這三個 function 會在效能測試時發揮功能

如果對於 function 內容有任何疑問，可以上網查詢（<http://en.cppreference.com/w/cpp/container>）

## II. 設計要求

- **Vector**、**List** 與 **Set** 必須繼承 `container_base`

- 沒有規定一定要繼承 `randomaccess_container`、`ordered_container` 與 `sorted_container`

- **Vector**、**List** 與 **Set** 的 **public** member function 必須與 `Vector.h`、`List.h` 與 `Set.h` 一模一樣

- 是的，三者的 `begin()` 都回傳相同型態的 iterator 與 `const_iterator`
- 您可以多增加 public member function，但不可減少
  - ◆ 例如您為 **Vector** 增加 `resize`、**Set** 增加 default constructor 與 **List** 增加 `sort`
  - ◆ 但您不可以把 **Set** 的 `count` 刪掉

- **iterator** 與 `const_iterator` 完全依您設計，`I2P_iterator.h` 與 `I2P_iterator.cpp` 內全是範例

- **iterator** 與 `const_iterator` 要有 `difference_type`、`value_type`、`pointer`、`reference` 與 `iterator_category` 這 5 個 member type
  - ◆ member type 請參考 [http://en.cppreference.com/w/cpp/iterator/iterator\\_traits](http://en.cppreference.com/w/cpp/iterator/iterator_traits) 與 [http://en.cppreference.com/w/cpp/iterator/iterator\\_tags](http://en.cppreference.com/w/cpp/iterator/iterator_tags)，寫法可直接參考 `I2P_iterator.h`
- **iterator** 與 `const_iterator` 都是 **random access iterator**

- **iterator**、`const_iterator`、**Vector**、**List** 與 **Set** 必須在 `I2P` 的 namespace 裡面

- 宣告與定義的型態要使用 `I2P_def.h` 裡面的宣告，因為實際測試時，會改變 `key_type` 與 `value_type`

### III. 以下是必要條件

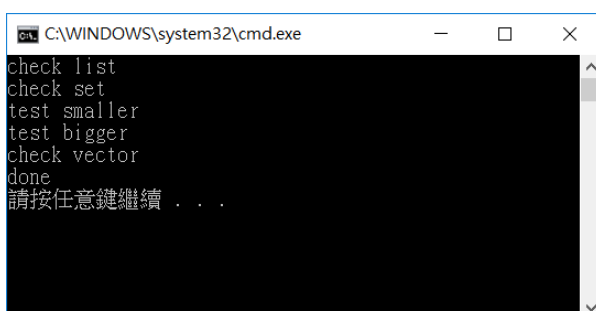
- 只能使用一個 thread 與一個 process
- 必須通過 visual c++ (19 以上)、clang++ (4 以上) 與 g++ (5 以上) 三者的編譯
  - 一個簡單的測試方式，是檢查是否有使用非 C++ standard library 的 header (例如 bits/stdc++.h，與非這個網頁裡面的 header <http://en.cppreference.com/w/cpp/header>)
  - 使用線上的 compiler 幫助編譯
    - ◆ <https://ideone.com/>
    - ◆ <http://codepad.org/>
    - ◆ <https://gcc.godbolt.org/>
  - 使用 Visual Studio
    - ◆ Visual Studio Community 2017 (免費)
      - <https://www.visualstudio.com/downloads/>
    - ◆ 系計中電腦教室、校園授權軟體有提供 Visual Studio Professional 2017

### IV. 以下是禁止事項

- 使用 C++ standard library 的 container 與 `basic_string`
  - 列在這個網頁的都不行 (<http://en.cppreference.com/w/cpp/container>)
- 使用非 C++ standard library 的 library
  - 如 boost
- 使用 asm
- 使用網路
  - 如 MPI
- 使用非 CPU 的計算資源
  - 如 GPU
- 使用特定 CPU 的指令
  - 如 SSE
- 抄襲
  - 如 Visual C++ 與 libstdc++ 的實作
    - ◆ 判斷方式：為了避免抄襲，會檢查程式碼。如果無法解釋程式碼，會予以該 container 項目 0 分

### V. (您) 如何檢驗 container 是否正確

- 使用 I2P\_main.cpp (請利用 I2P\_test.cpp 裡的 op\_test\_cnt 調整操作數量)
- 自行測試正確性時，請將 I2P\_def.h 裡面的 `value_type` 改成不同型態 (如果是測試 `Set`，那請把 `key_type` 改成與 `value_type` 相同)
- 只要執行結果如下圖，就表示您的 container 應該是正確的



```
C:\WINDOWS\system32\cmd.exe
check list
check set
test smaller
test bigger
check vector
done
請按任意鍵繼續...
```

## VI. 評分方式

### ● 評分流程

測試**各個** container 是否能通過編譯

是，檢查 I2P\_main.cpp 輸出是否正確（會更改 **key\_type** 與 **value\_type**）

正確，則 **vector** 2 分、**list** 2 分與 **set** 3 分，並進行效能測試

檢查**各個**效能測試的輸出是否正確

正確，則列入效能測試的評分名單

### ● 報告（3 分）

報告必須涵蓋以下幾個測試，**並分析可能原因（要合乎常理）**，**或是何時該使用何種 container**

#### ■ **Vector** 與 **List** 的比較

◆ 插入 element 在最前面（0.3）

◆ 插入 element 在最後面（0.3）

◆ 隨機在任意位置插入 element（已經知道需要在何處插入）（0.4）

◆ 隨機在任意位置刪除 element（已經知道需要在何處刪除）（0.4）

#### ■ **Set**

◆ input 的插入順序，是否會影響效能？（0.4）

（例如：由小至大或由大至小或是忽小忽大）

◆ 隨機在任意位置刪除 element（已經知道需要在何處刪除）（0.4）

#### ■ 三者皆須測試

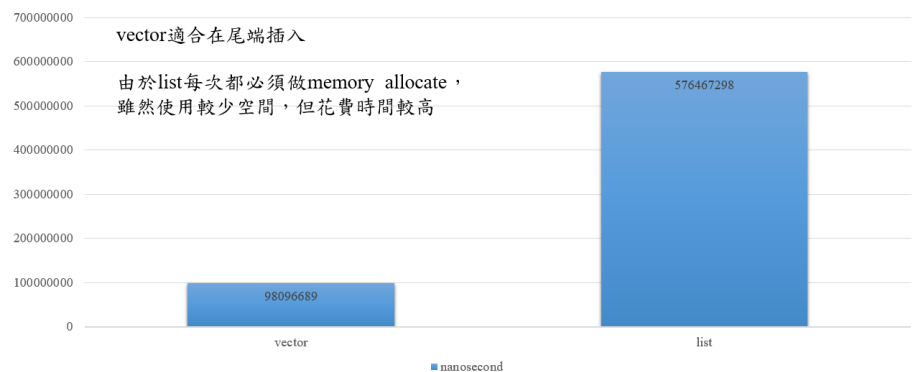
◆ 遍歷全部 element（0.3）

◆ 同樣數量的 input，記憶體的使用狀況（0.5）

#### ■ 範例

（數字僅供參考）

插入element在最後面（10000000個**long**）



◆ 呈現方式不限，用表格、文字敘述或圖檔皆可（不可以拍影片）

◆ 只能使用中文或英文描述

◆ 檔案格式必須為 pdf

#### ■ 即使 container 的輸出錯誤，仍不影響報告分數（兩者互相獨立）

### ● 效能測試（額外 2 分）

#### ■ 測試程式不是 I2P\_main.cpp

■ 測資會非常龐大（例如超過 2 GiB）

■ **value\_type** 有四種型態（**int8\_t**、**int64\_t**、**long double** 與 **student**）

■ 根據效能測試分布及結果（執行時間與記憶體使用狀況），再予以評分

## VII. 編譯方式與檔案擺放格式

- 以下是我們測試時會提供的檔案：

I2P\_container.h、I2P\_def.h、I2P\_student.h、I2P\_test.h、I2P\_test\_unit.h、I2P\_container.cpp、I2P\_main.cpp、I2P\_student.cpp、I2P\_test.cpp 與 I2P\_test\_unit.cpp（不會提供 I2P\_iterator.h、I2P\_iterator.cpp、List.h、Set.h 與 Vector.h）

- 以下是您必須提供的檔案（您可以提供額外的檔案，但至少要有以下 3 個）：

List.h、Set.h 與 Vector.h

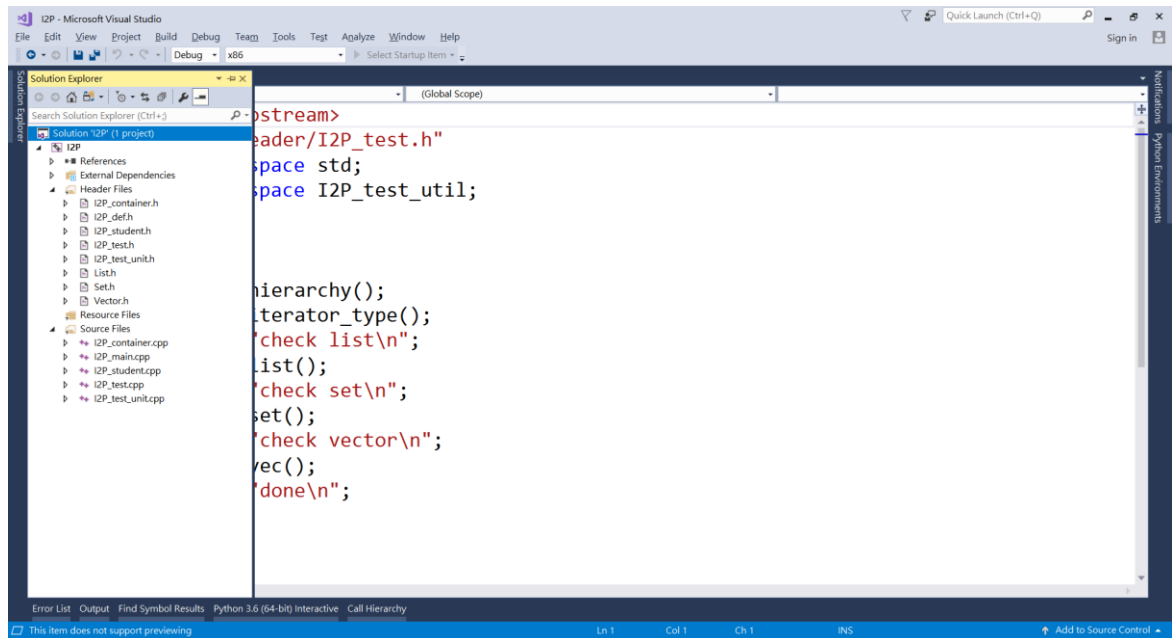
- 檔案擺放方式：

106123456.zip（您的學號.zip）

```
|—— header
|   |—— I2P_container.h
|   |—— I2P_def.h
|   |—— I2P_student.h
|   |—— I2P_test.h
|   |—— I2P_test_unit.h
|   |—— List.h
|   |—— Set.h
|   |—— something_you_need.h
|   |—— Vector.h
|—— I2P_main.cpp
|—— report
|   |—— 106123456.pdf（您的學號.pdf）
|—— src
|   |—— I2P_container.cpp
|   |—— I2P_student.cpp
|   |—— I2P_test.cpp
|   |—— I2P_test_unit.cpp
|   |—— something_you_need.cpp
```

- something\_you\_need.h 是指其他您可能會用到的.h 檔案（例如您額外提供 iterator.h、const\_iterator.h、...）。something\_you\_need 的名字只是範例，您可以取自己偏好的名字
- something\_you\_need.cpp 是指其他您可能會用到的.cpp 檔案（例如您額外提供 List.cpp、Set.cpp、...）。something\_you\_need 的名字只是範例，您可以取自己偏好的名字
- 所有在 src 裡面，副檔名是.cpp 的檔案都會被編譯。以下是編譯指令：

```
g++ -std=c++11 I2P_main.cpp `find src -name '*.cpp'`
clang++ -std=c++11 I2P_main.cpp `find src -name '*.cpp'`
Visual Studio
```



### VIII. 其他注意事項

- 不用考慮有 exception 時，需要還原 container 狀態的情況
- `student` 沒有 default constructor
- `Set` 要利用 `key_compare` 比較大小
- `Set` 有非常多的優化手段
- 編譯會統一使用 C++11 的標準。效能測試時會開優化選項
- 以下是 `I2P_test_util.h` 的程式碼的 tag

`CheckAfter`：檢查 iterator 插入位置後的 iterator 與 pointer，當測試的為 list 與 set 時，設成 `yes_tag`

`RandomIter`：如果 iterator 是 random access iterator，設成 `yes_tag`。基本上這次的 iterator 都要設成 `yes_tag`

`Reallocation`：是否有 reallocation，當 `test_type` 為 vector 時，設成 `yes_tag`

`StdEraseBegin`：是否要用 erase 與 begin 取代 pop\_back，如果 `std_type` 為 `std::vector`，設成 `yes_tag`

`UsrEraseBegin`：是否要用 erase 與 begin 取代 pop\_back，基本上這次的 container 都要設成 `no_tag`

`WithAlgo`：`std_type` 是否有 sort 與 unique，若為 `std::list`，設成 `yes_tag`

`WithCapacity`：是否有 capacity，基本上這次的 container 都要設成 `yes_tag`

`WithPos`：insert 是否需要 pos 做為第一個參數，當 `test_type` 為 list 或 vector 時，設成 `yes_tag`