

班級：醫工二甲
學號：11125107
姓名：李俞憲

作業一、

Home Works

- Create a Student class
- Using template, create a Queue class
 - in main(), to
 1. Push 3 students (student should be created by DMA)
 2. Show the front student (Print)
 3. Show the rear student (Print)
 4. Pop
 5. Show the front student (Print)
- **note: test by simple datatype first: e.g. int or float**
- Watch git lecture video

• [中原大學 - git&github 實戰與應用 - \(1\) - git commit --amend, 退版與救回commit點](#)

Student	Queue
+<<constructor>> Student() +<<destructor>> ~Student() +<<constructor>> Student(stud_id:int, stud_name:const char*) +SetID(stud_id:int):int +SetName(stud_name:const char*):int +GetID():int +GetName():char* +Print():void	+<<constructor>> Queue() +<<destructor>> ~Queue() +<<constructor>> Queue(MaxQueueSize:int) +IsEmpty():bool +Front():<template &T> +Rear():<template &T> +Push(<template T> &):void +Pop():void
-ID:int -name:char*	-queue_elements:<template T*> -front:int -rear:int -capacity:int

執行流程：

```
Show the front student:
ID: 1
Name:Alice

Show the rear student:
ID: 3
Name:Jack

Show the front student:
ID: 2
Name:Bob
PS D:\vscode> |
```

班級：醫工二甲
學號：11125107
姓名：李俞憲
說明：

```
int main()
{
    const int len=3;
    // 動態分配一個大小為5的Student陣列
    Student *studentArray = new Student[3];

    // 使用動態分配的陣列
    studentArray[0] = Student(1, "Alice");
    studentArray[1] = Student(2, "Bob");
    studentArray[2] = Student(3, "Jack");

    Queue <Student> qe(3);
    for(int i=0; i<3; i++)
    {
        qe.Push(studentArray[i]);
    }
}
```

先創建 Student 的 array，以及 Queue 的容量，隨後進行添加

```
Queue <Student> qe(3);
for(int i=0; i<3; i++)
{
    qe.Push(studentArray[i]);
}

cout<<"Show the front student:"<<endl;
qe.Front().Print();
cout<<"\nShow the rear student:"<<endl;
qe.Rear().Print();
qe.Pop();
cout<<"\nShow the front student:"<<endl;
qe.Front().Print();
```

隨後根據題意進行相應操作，接下來會說明 Pop 與 Push 以及排序

班級：醫工二甲
學號：11125107
姓名：李俞憲

```
template<class T>
void Queue<T>::Pop() {
    int temp_front = front;
    if (rear == -1)
    {
        throw std::runtime_error("Stack is empty."); //當內部無值時報錯
        return;
    }

    if (rear == front) //剩下最後一個時
    {
        rear = -1; //初始化
        front = -1; //初始化
        // cout<<"rear:"<<rear<<"\tf:"<<front<<endl;
        // cout<<"r="<<queue_elements[rear]<<"\tf="<<queue_elements[fr
        return;
    }

    front++; //物件取出時front值++
    // cout<<"rear:"<<rear<<"\tf:"<<front<<endl;
    // cout<<"r="<<queue_elements[rear]<<"\tf="<<queue_elements[fr
    return;
}
```

Pop 的部份我是拆成 3 種情況進行討論，當 rear 超出上限值時進行報錯，另一個是當陣列中的元素僅剩 1 個時，Pop 後需要進行初始化，若不是以上 2 種情形則為正常情形，front 直接+1

班級：醫工二甲
學號：11125107
姓名：李俞憲

```
template<class T>
void Queue<T>::Push(T &item) {
    int delta_empty; // 空格變數
    int counter = front;
    delta_empty = front; // 計算空格

    if (rear == (capacity - 1) && (front == 0))
    {
        throw std::runtime_error("Stack is full."); // 當空間滿時報錯 r=最末端, f前的空格數為0
        return;
    }
}
```

Push 的部份比較複雜，會逐一討論，首先遇到的情況是當序列滿時，需要進行報錯，上方的變數是接下來會使用到的

```
if (rear == capacity-1) // 末端
{
    cout<<"剩餘的空位還剩下"<<delta_empty<<"個\n";
    cout<<"sorting...\n";

    while(counter <= rear)
    {
        queue_elements[counter - delta_empty] = queue_elements[++counter];
        // cout<<"rear:"<<rear<<"\tf:"<<front<<endl;
        // cout<<"r="<<queue_elements[rear]<<"\tf="<<queue_elements[front]<<endl;
    }
    front = 0;
    rear -= delta_empty;
    queue_elements[++rear] = item; // 新物件加入時，rear值++
    // cout<<"rear:"<<rear<<"\tf:"<<front<<endl;
    // cout<<"r="<<queue_elements[rear]<<"\tf="<<queue_elements[front]<<endl;
    return;
}
```

這邊是考慮當序列已經排到末端，但是前面還有空格時，需要先排序，其實 counter 和 delta_empty 的值和 front 期時是一樣的，不過 counter 會變化，以及我怕 front 被誤植錯誤的數值，因此在此這 2 個變數做處理，排序好後，再將新的數值進行 Push

班級：醫工二甲
學號：11125107
姓名：李俞憲

```
if(front == -1)//如果是初值
{
    front++;
    queue_elements[++rear] = item;//新物件加入時，rear值++
    // cout<<"rear:"<<rear<<"\tf:"<<front<<endl;
    // cout<<"r="<<queue_elements[rear]<<"\tf="<<queue_elements[front]<<endl;
    return;
}
```

另一個情形是，如果 front、rear 皆為初值=-1 時，需要將 front 一起+1

```
queue_elements[++rear] = item;//新物件加入時，rear值++
// cout<<"rear:"<<rear<<"\tf:"<<front<<endl;
// cout<<"r="<<queue_elements[rear]<<"\tf="<<queue_elements[front]<<endl;
return;
```

這個就是一般狀況，直接添加數值，rear+1

程式碼：

//main

```
#include "1019_11125107.hpp"
#include "1019_11125107f.cpp"

using namespace std;

int main()
{
    const int len=3;
    // 動態分配一個大小為 5 的 Student 陣列
    Student *studentArray = new Student[3];

    // 使用動態分配的陣列
    studentArray[0] = Student(1, "Alice");
    studentArray[1] = Student(2, "Bob");
    studentArray[2] = Student(3, "Jack");

    Queue <Student> qe(3);
    for(int i=0;i<3;i++)
```

班級：醫工二甲
學號：11125107
姓名：李俞憲

```
{
    qe.Push(studentArray[i]);
}
cout<<"Show the front student:"<<endl;
qe.Front().Print();
cout<<"\nShow the rear student:"<<endl;
qe.Rear().Print();
qe.Pop();
cout<<"\nShow the front student:"<<endl;
qe.Front().Print();

//註解的部份僅為校驗用
// int int_list[] ={1,2,3};

// Queue <int> IntQueue(3);

// if (IntQueue.IsEmpty())
// {
//     cout << "序列(int)為空" << endl;
// }
// else
// {
//     cout << "序列(int)不為空" << endl;
// }

// for(int i=0;i<3;i++)
// {
//     IntQueue.Push(int_list[i]);
// }
//IntQueue.Push(int_list[0]);//嘗試加爆(狀況 1)Passed

// for(int i=0;i<3;i++)
// {
//     IntQueue.Pop();
// }
//IntQueue.Pop();//嘗試多扣除(狀況 2)Passed

// for(int i=0;i<3;i++)
```

班級：醫工二甲
學號：11125107
姓名：李俞憲

```
// {  
//     IntQueue.Push(int_list[i]);  
// }  
// IntQueue.Pop();  
// IntQueue.Push(int_list[0]); //當前面還有空格時，排序(狀況 3)Passed  
  
// if (IntQueue.IsEmpty())  
// {  
//     cout << "序列(int)為空" << endl;  
// }  
// else  
// {  
//     cout << "序列(int)不為空" << endl;  
// }  
  
return 0;  
}
```

//.h

```
#include <iostream>  
#include <cstdlib>  
#include <cstring>  
#include <typeinfo>  
  
using namespace std;  
#pragma once  
  
class Student {  
private:  
    int ID;  
    char* name;  
  
public:  
    Student();  
    ~Student();  
    Student(int stud_id, const char* stud_name);  
    int SetID(int stud_id);  
    int SetName(char* stud_name);  
};
```

班級：醫工二甲
學號：11125107
姓名：李俞憲

```
int GetID();
char* GetName();
void Print();
void operator=(Student src); //等號(class=class)
};

template<class T>
class Queue {
private:
    T* queue_elements; // 堆疊元素量
    int front; // 堆疊指標(尾端)
    int rear; //堆疊前標(首)
    int capacity; // 堆疊容量

public://add operator +
    Queue(); //動態記憶體自動配置
    Queue(int MaxQueueSise); //手動輸入空間值
    ~Queue(); //解構子 delete[]
    bool IsEmpty(); //檢查是否為空，T:為空；F:不為空
    T& Front(); //堆疊元素(尾端)
    T& Rear(); //堆疊元素(首)
    void Push(T &item); //放入元素
    void Pop(); //取出元素
};

template<class T>
Queue<T> :: Queue() : queue_elements(NULL), front(-1), rear(-1), capacity(0) {}

template<class T>
Queue<T> :: Queue(int MaxQueueSise) : front(-1), rear(-1), capacity(MaxQueueSise)
{
    queue_elements = new T[capacity](); // 將所有元素初始化為 0
}

template<class T>
Queue<T> :: ~Queue()
{
    delete[] queue_elements;
```


班級：醫工二甲
學號：11125107
姓名：李俞憲

```
}

template<class T>
bool Queue<T> :: IsEmpty()
{
    if (front == -1)
    {
        return true;
    }
    else
    {
        return false;
    }
}

template<class T>
T& Queue<T>::Front()
{
    if (front == -1) {
        throw std::runtime_error("Front is empty."); //出現空時報錯
    }
    //cout<<front<<endl;
    return queue_elements[front]; //輸出 front 指標所指的位置
}

template<class T>
T& Queue<T>::Rear()
{
    if (rear == -1) {
        throw std::runtime_error("Rear is empty."); //出現空時報錯
    }
    return queue_elements[rear]; //輸出 front 指標所指的位置
}

template<class T>
void Queue<T>::Push(T &item) {
    int delta_empty; //空格變數
    int counter = front;
```

班級：醫工二甲
學號：11125107
姓名：李俞憲

```
delta_empty = front; //計算空格

if (rear == (capacity - 1) && (front == 0))
{
    throw std::runtime_error("Stack is full."); //當空間滿時報錯 r=最末端, f 前的
// 空格數為 0
    return;
}

if (rear == capacity-1) //末端
{
    cout<<"剩餘的空位還剩下"<<delta_empty<<"個\n";
    cout<<"sorting...\n";

    while(counter <= rear)
    {
        queue_elements[counter - delta_empty] = queue_elements[++counter];
        //cout<<"rear:"<<rear<<"\tf:"<<front<<endl;
        //cout<<"r="<<queue_elements[rear]<<"\tf="<<queue_elements[front]<<endl;
//endl;

    }
    front = 0;
    rear -= delta_empty;
    queue_elements[++rear] = item; //新物件加入時, rear 值++
    // cout<<"rear:"<<rear<<"\tf:"<<front<<endl;
    // cout<<"r="<<queue_elements[rear]<<"\tf="<<queue_elements[front]<<endl;
    return;
}

if(front == -1) //如果是初值
{
    front++;
    queue_elements[++rear] = item; //新物件加入時, rear 值++
    // cout<<"rear:"<<rear<<"\tf:"<<front<<endl;
    // cout<<"r="<<queue_elements[rear]<<"\tf="<<queue_elements[front]<<endl;
    return;
}
```

班級：醫工二甲
學號：11125107
姓名：李俞憲

```
        queue_elements[++rear] = item; //新物件加入時，rear 值++
        // cout<<"rear:"<<rear<<"\tfront:"<<front<<endl;
        // cout<<"r="<<queue_elements[rear]<<"\tf="<<queue_elements[front]<<endl;
        return;
    }

template<class T>
void Queue<T>::Pop() {
    int temp_front = front;
    if (rear == -1)
    {
        throw std::runtime_error("Stack is empty."); //當內部無值時報錯
        return;
    }

    if (rear == front) //剩下最後一個時
    {
        rear = -1; //初始化
        front = -1; //初始化
        // cout<<"rear:"<<rear<<"\tfront:"<<front<<endl;
        // cout<<"r="<<queue_elements[rear]<<"\tf="<<queue_elements[front]<<endl;
        return;
    }

    front++; //物件取出時 front 值++
    // cout<<"rear:"<<rear<<"\tfront:"<<front<<endl;
    // cout<<"r="<<queue_elements[rear]<<"\tf="<<queue_elements[front]<<endl;
    return;
}

//.cpp
#include "1019_11125107.hpp"

Student::Student() : ID(0), name(nullptr) {}
```

班級：醫工二甲
學號：11125107
姓名：李俞憲

```
Student :: ~Student()
{
    // if (name != NULL)
    // {
    //     delete[] name;
    // }
}

Student::Student(int stud_id, const char* stud_name) {
    ID = stud_id;
    name = new char[strlen(stud_name) + 1];
    strcpy(name, stud_name);
}

int Student :: SetID(int stud_id)
{
    ID = stud_id;
    return 0;
}

int Student :: SetName(char* stud_name)
{
    if (name != stud_name) {
        if (name != nullptr) {
            delete[] name;
        }
        name = new char[strlen(stud_name) + 1];
        strcpy(name, stud_name);
    }
    return 0;
}

int Student :: GetID()
{
    int id_temp = ID;
    return id_temp;
}
```

班級：醫工二甲
學號：11125107
姓名：李俞憲

```
char* Student :: GetName()
{
    char* data2pub = new char[strlen(name)+1];
    strcpy(data2pub,name);
    return data2pub;
}

void Student :: Print()
{
    cout << "ID: " << GetID()<<endl<<"Name:";
    for (int i=0;i<strlen(name);i++)
    {
        char* ptr;
        ptr = GetName();
        cout<<ptr[i];
    }
    cout<<endl;
}

void Student::operator=(Student src)//A=B
{
    if (name != NULL)
    {
        delete[] name;//刪除舊資料
        name = NULL;
    }

    ID = src.ID;
    name = new char[strlen(src.name)];
    strcpy(name,src.name);
    return;
}
```

班級：醫工二甲

學號：11125107

姓名：李俞憲

補充說明（遇到的困難或心得，選填）：

這次的作業期時只是將一點邏輯做修改而已，但是要考慮的狀況有比較多，反而是思考的成分比較高，但解開了，就勢如破竹，很快就能夠解決了。