

班級：醫工二甲  
學號：11125114  
姓名：許仁駿

作業一、 【創建一個 Student class，再用 template 創一個 Queue class，  
並將 Student 用進 Queue】

執行流程：

【截圖 1】

```
1  #pragma once
2  #include <iostream>
3  #include <cstring>
4  #pragma warning(disable : 4996)
5  using namespace std;
6
7  //Student class 實作
8  class Student
9  {
10 public:
11     Student();
12     Student(int stud_id, const char* stud_name);
13     ~Student() {};
14
15     int SetID(int stud_id);
16     int SetName(const char* stud_name);
17     int GetID();
18     char* GetName();
19     void Print();
20
21 private:
22     int ID;
23     char* name;
24 };
25
```

說明：先在 Header 檔創建所需的類別 Student。

【截圖 2】

班級：醫工二甲  
學號：11125114  
姓名：許仁駿

```
1  #include "1019.h"
2  #include <iostream>
3  using namespace std;
4
5  Student::Student() //建構子
6  {
7      ID = 0;
8      name = NULL;
9  }
10
11 Student::Student(int stud_id, const char* stud_name) //建構子
12 {
13     ID = stud_id;
14     name = new char[10];
15     std::strcpy(name, stud_name);
16 }
17
18 int Student::SetID(int stud_id)
19 {
20     ID = stud_id;
21     return 1;
22 }
```

班級：醫工二甲  
學號：11125114  
姓名：許仁駿

```
24  int Student::SetName(const char* stud_name)
25  {
26      name = new char[10];
27      std::strcpy(name, stud_name);
28      return 1;
29  }
30
31  int Student::GetID()
32  {
33      return ID;
34  }
35
36  char* Student::GetName()
37  {
38      return name;
39  }
40
41
42  void Student::Print()
43  {
44      cout << "student's id is:" << ID << endl;
45      cout << "student's name is:" << name << endl;
46  }
```

說明：Source 檔定義函式。

【截圖 3】

班級：醫工二甲  
學號：11125114  
姓名：許仁駿

```
26
27 //Queue 實作
28 template<class T>
29 class Queue
30 {
31 public:
32     Queue();
33     ~Queue();
34     Queue(int MaxQueueSize);
35
36     bool IsEmpty();
37
38     T& Front();
39     T Rear();
40     void Push(T& element);
41     void Pop();
42
43 private:
44     T* queue_elements;
45     int front;
46     int rear;
47     int capacity;
48 };
49
```

說明：在 Header 檔創建所需的類別 Queue (用 template 創建)

【截圖 4】

班級：醫工二甲  
學號：11125114  
姓名：許仁駿

```
50     template <class T>
51     inline Queue<T>::Queue()
52     {
53         this->capacity = 3;
54         queue_elements = new T[capacity];
55         front = 0; //計算Pop次數，以得到front的座標
56         rear = -1; //初始化rear指標為-1，表示佇列是空的
57     }
58
59     template <class T>
60     Queue<T>::Queue(int capacity)
61     {
62         this->capacity = capacity;
63         queue_elements = new T[capacity];
64         front = 0; //計算Pop次數，以得到front的座標
65         rear = -1; //初始化rear指標為-1，表示佇列是空的
66     }
67
68     template <class T>
69     Queue<T>::~~Queue()
70     {
71         delete[] queue_elements; //delete堆疊內存
72     }
73
74     template <class T>
75     bool Queue<T>::IsEmpty()
76     {
77         if (rear - front == -1) //若成立，代表剛Pop出rear那一格的東西，即empty。
78         {
79             front = 0;
80             rear = -1;
81             return true;
82         }
83         else
84         {
85             return false;
86         }
87     }
```

班級：醫工二甲  
學號：11125114  
姓名：許仁駿

```
89     template <class T>
90     void Queue<T>::Push(T& element)
91     {
92         if (rear == capacity - 1 && front==0) //若成立，表示rear在最後一格，且front在第一格，佇列已滿
93         {
94             throw std::runtime_error("佇列已滿，無法推進元素");
95         }
96         else if (rear == capacity - 1) //若成立，表示rear到底，但front前面還有空。
97         {
98             for (int i = front; i <= capacity-1; i++)
99             {
100                 queue_elements[i - front] = queue_elements[i]; //所以要把整個佇列往前移front格
101             }
102         }
103         else
104         {
105             queue_elements[++rear] = element; //推進元素並更新rear指標
106         }
107     }

109     template <class T>
110     void Queue<T>::Pop()
111     {
112         if (IsEmpty()) //若為empty，則在IsEmpty()把front跟rear重製
113         {
114             throw std::runtime_error("堆疊是空的，無法彈出元素");
115         }
116
117         front += 1; //計算Pop次數，同時表示front座標
118         cout << "已 Pop 出 Front 之成員!" << endl;
119     }

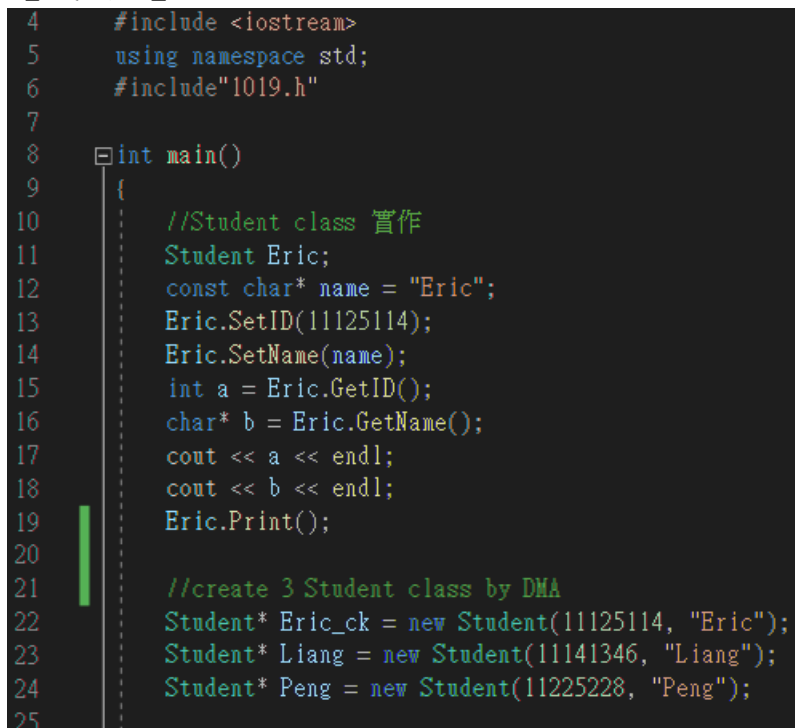
120
121     template <class T>
122     T& Queue<T>::Front()
123     {
124         return queue_elements[front];
125     }

126
127     template <class T>
128     T Queue<T>::Rear()
129     {
130         return queue_elements[rear];
131     }
```

班級：醫工二甲  
學號：11125114  
姓名：許仁駿

說明：在 Header 檔 定義函式。

【截圖 5】



```
4 #include <iostream>
5 using namespace std;
6 #include "1019.h"
7
8 int main()
9 {
10     //Student class 實作
11     Student Eric;
12     const char* name = "Eric";
13     Eric.SetID(11125114);
14     Eric.SetName(name);
15     int a = Eric.GetID();
16     char* b = Eric.GetName();
17     cout << a << endl;
18     cout << b << endl;
19     Eric.Print();
20
21     //create 3 Student class by DMA
22     Student* Eric_ck = new Student(11125114, "Eric");
23     Student* Liang = new Student(11141346, "Liang");
24     Student* Peng = new Student(11225228, "Peng");
25 }
```

說明：在 main 檔，先做 Student 的測試，再用動態記憶體創建三個。

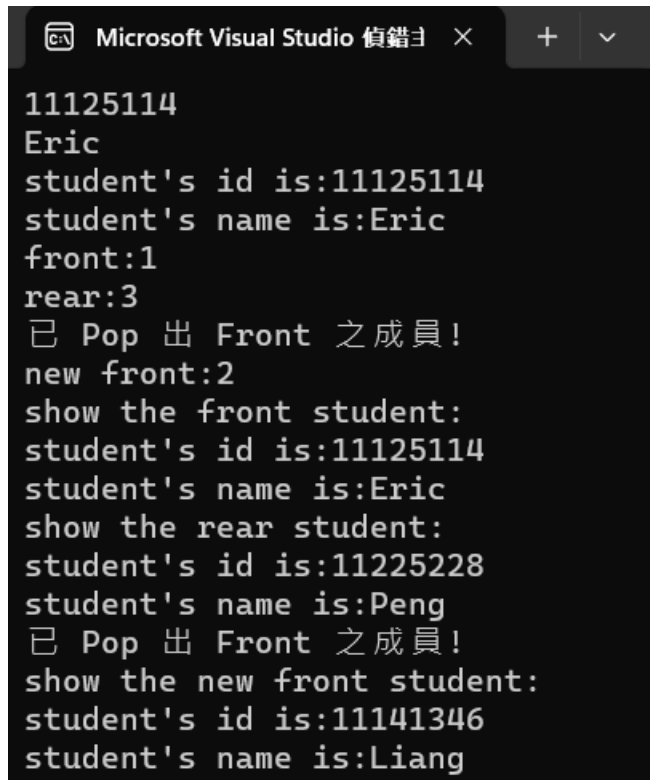
【截圖 6】

班級：醫工二甲  
學號：11125114  
姓名：許仁駿

```
26 //Queue class 實作 (T = int)
27 Queue<int> Number;
28 int aa = 1, bb = 2, cc = 3;
29 Number.Push(aa);
30 Number.Push(bb);
31 Number.Push(cc);
32 int front = Number.Front();
33 int rear = Number.Rear();
34 cout << "front:" << front << endl;
35 cout << "rear:" << rear << endl;
36 Number.Pop();
37 int new_front = Number.Front();
38 cout << "new front:" << new_front << endl;
39
40 //Queue class 實作 (T = Student, Student 為 class)
41 Queue<Student> Student_Queue;
42 Student_Queue.Push(*Eric_ck); //Push 3 students
43 Student_Queue.Push(*Liang);
44 Student_Queue.Push(*Peng);
45 cout << "show the front student:" << endl;
46 Student_Queue.Front().Print(); //Show the front student
47 cout << "show the rear student:" << endl;
48 Student_Queue.Rear().Print(); //Show the rear student
49 Student_Queue.Pop(); //Pop
50 cout << "show the new front student:" << endl;
51 Student_Queue.Front().Print(); //Show the front student
52
```



班級：醫工二甲  
學號：11125114  
姓名：許仁駿



```
Microsoft Visual Studio 偵錯 1  x + v
11125114
Eric
student's id is:11125114
student's name is:Eric
front:1
rear:3
已 Pop 出 Front 之成員!
new front:2
show the front student:
student's id is:11125114
student's name is:Eric
show the rear student:
student's id is:11225228
student's name is:Peng
已 Pop 出 Front 之成員!
show the new front student:
student's id is:11141346
student's name is:Liang
```

說明：之後在 main 檔測試 Queue，再把剛剛的 3 個 Student 丟進去。

---

程式碼：

Main 檔：

```
#include <iostream>
using namespace std;
#include "1019.h"

int main()
{
    //Student class 實作
    Student Eric;
    const char* name = "Eric";
    Eric.SetID(11125114);
    Eric.SetName(name);
    int a = Eric.GetID();
    char* b = Eric.GetName();
```

班級：醫工二甲  
學號：11125114  
姓名：許仁駿

```
cout << a << endl;
cout << b << endl;
Eric.Print();

//create 3 Student class by DMA
Student* Eric_ck = new Student(11125114, "Eric");
Student* Liang = new Student(11141346, "Liang");
Student* Peng = new Student(11225228, "Peng");

//Queue class 實作 (T = int)
Queue<int> Number;
int aa = 1, bb = 2, cc = 3;
Number.Push(aa);
Number.Push(bb);
Number.Push(cc);
int front = Number.Front();
int rear = Number.Rear();
cout << "front:" << front << endl;
cout << "rear:" << rear << endl;
Number.Pop();
int new_front= Number.Front();
cout << "new front:" << new_front << endl;

//Queue class 實作 (T = Student, Student 為 class)
Queue<Student> Student_Queue;
Student_Queue.Push(*Eric_ck); //Push 3 students
Student_Queue.Push(*Liang);
Student_Queue.Push(*Peng);
cout << "show the front student:" << endl;
Student_Queue.Front().Print(); //Show the front student
cout << "show the rear student:" << endl;
Student_Queue.Rear().Print(); //Show the rear student
Student_Queue.Pop();//Pop
cout << "show the new front student:" << endl;
Student_Queue.Front().Print(); //Show the front student

return 0;
}
```

### Source 檔:

```
#include "1019.h"
#include <iostream>
using namespace std;
```

班級：醫工二甲  
學號：11125114  
姓名：許仁駿

```
Student::Student() //建構子
{
    ID = 0;
    name = NULL;
}

Student::Student(int stud_id, const char* stud_name) //建構子
{
    ID = stud_id;
    name = new char[10];
    std::strcpy(name, stud_name);
}

int Student::SetID(int stud_id)
{
    ID = stud_id;
    return 1;
}

int Student::SetName(const char* stud_name)
{
    name = new char[10];
    std::strcpy(name, stud_name);
    return 1;
}

int Student::GetID()
{
    return ID;
}

char* Student::GetName()
{
    return name;
}

void Student::Print()
{
    cout << "student's id is:" << ID << endl;
    cout << "student's name is:" << name << endl;
}
```

---

班級：醫工二甲  
學號：11125114  
姓名：許仁駿

Header 檔:

```
#pragma once
#include <iostream>
#include <cstring>
#pragma warning(disable : 4996)
using namespace std;

//Student class 實作
class Student
{
public:
    Student();
    Student(int stud_id, const char* stud_name);
    ~Student() {};

    int SetID(int stud_id);
    int SetName(const char* stud_name);
    int GetID();
    char* GetName();
    void Print();

private:
    int ID;
    char* name;
};

//Queue 實作
template<class T>
class Queue
{
public:
    Queue();
    ~Queue();
    Queue(int MaxQueueSize);

    bool IsEmpty();

    T& Front();
    T Rear();
    void Push(T& element);
    void Pop();
};
```

班級：醫工二甲  
學號：11125114  
姓名：許仁駿

```
private:
    T* queue_elements;
    int front;
    int rear;
    int capacity;
};

template <class T>
inline Queue<T>::Queue()
{
    this->capacity = 3;
    queue_elements = new T[capacity];
    front = 0; //計算Pop次數，以得到front的座標
    rear = -1; //初始化rear指標為-1，表示佇列是空的
}

template <class T>
Queue<T>::Queue(int capacity)
{
    this->capacity = capacity;
    queue_elements = new T[capacity];
    front = 0; //計算Pop次數，以得到front的座標
    rear = -1; //初始化rear指標為-1，表示佇列是空的
}

template <class T>
Queue<T>::~~Queue()
{
    delete[] queue_elements; //delete堆疊內存
}

template <class T>
bool Queue<T>::IsEmpty()
{
    if (rear - front == -1) //若成立，代表剛Pop出rear那一格的東西，即empty。
    {
        front = 0;
        rear = -1;
        return true;
    }
    else
    {
        return false;
    }
}
```

班級：醫工二甲  
學號：11125114  
姓名：許仁駿

```
    }  
}  
  
template <class T>  
void Queue<T>::Push(T& element)  
{  
    if (rear == capacity - 1 && front==0) //若成立，表示rear在最後一格，且front在第一格，佇列已滿  
    {  
        throw std::runtime_error("佇列已滿，無法推進元素");  
    }  
    else if (rear == capacity - 1) //若成立，表示rear到底，但front前面還有空。  
    {  
        for (int i = front; i <= capacity-1; i++)  
        {  
            queue_elements[i - front] = queue_elements[i]; //所以要把整個佇列往前移front格  
        }  
    }  
    else  
    {  
        queue_elements[++rear] = element; //推進元素並更新rear指標  
    }  
}  
  
template <class T>  
void Queue<T>::Pop()  
{  
    if (IsEmpty()) //若為empty，則在IsEmpty()把front跟rear重製  
    {  
        throw std::runtime_error("堆疊是空的，無法彈出元素");  
    }  
  
    front += 1; //計算Pop次數，同時表示front座標  
    cout << "已 Pop 出 Front 之成員!" << endl;  
}  
  
template <class T>  
T& Queue<T>::Front()  
{  
    return queue_elements[front];  
}  
  
template <class T>  
T Queue<T>::Rear()
```

班級：醫工二甲

學號：11125114

姓名：許仁駿

```
{  
    return queue_elements[rear];  
}
```

---

---

**補充說明（遇到的困難或心得，選填）：**

在上一次作業中了解 template 以及 Stack 後，這次的 Queue 概念其實相似，改變的大概是把 top 去掉，多了 front 以及 rear 的概念，並且 Pop 是把最前 Push 的移出，就好比 Stack 是堆疊，Queue 是排隊一樣，了解完後，其實我在上課就順利完成作業了！

---

---