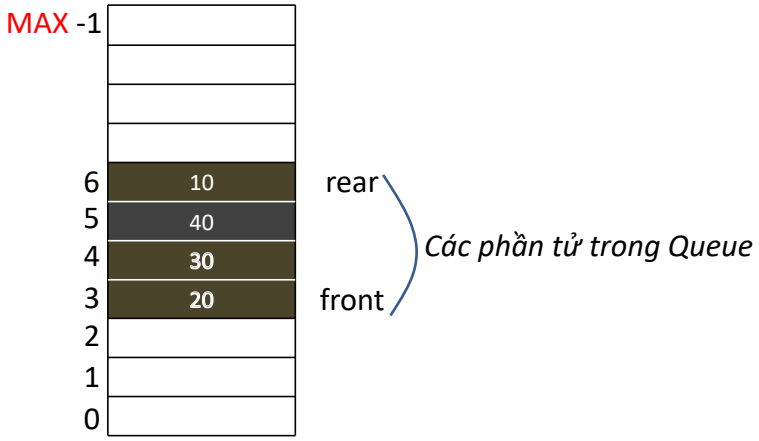
BÀI THỰC HÀNH SỐ 4

1. MỤC TIÊU:

- Làm quen với cấu trúc dữ liệu: Queue
- Thực hiện các thao tác cơ bản trên danh sách Queue:
  - Tạo Queue
  - Kiểm tra Queue rỗng
  - Kiểm tra Queue đầy
  - Push
  - Pop

2. LÝ THUYẾT CẦN GHI NHỚ

- Hàng đợi (Queue) là danh sách chứa các phần tử được quản lý theo thứ tự sau: Phần tử được thêm vào trước, sẽ được lấy ra (xóa) trước (FIFO)
- Có 2 cách để hiện thực Queue:
  - Dùng danh sách đặc
  - Dùng danh sách liên kết đơn



3. BÀI TẬP THỰC HÀNH CƠ BẢN

SV tạo project "Win32 Console Application" trên VS 2015, với tên là "**Lab3**".

Tạo file source .cpp có tên: **hovaten_mssv_lab4.cpp**

*** Cách tạo project và cấu trúc chương trình C++: Xem lại file hướng dẫn thực hành lab 1

Bài 1:

Quản lý một queue có tối đa 100 phần tử, mỗi phần tử trong queue có kiểu int (queue danh sách đặc)

a. Khai báo cấu trúc queue.

```
# define MAX 100
int a[MAX];
int front, rear; // front vị trí lấy ra, rear vị trí thêm vào
```

b. Viết thủ tục khởi tạo queue rỗng

```
void init(int a[], int& front, int& rear)
{
    front = -1;
    rear = -1;
}
```

c. Viết thủ tục kiểm tra queue rỗng.
Queue rỗng khi front =-1
SV tự viết code

d. Viết thủ tục kiểm tra queue đầy
Queue đầy thi rear = MAX -1
SV tự viết code

e. Viết thủ tục thêm một phần tử vào queue

- SV viết code thêm một phần tử vào queue bằng 2 phương pháp:
  - Phương pháp tịnh tiến
  - Phương pháp vòng
- Viết thủ tục xóa một phần tử trong queue
  SV viết code xóa một phần tử ra khỏi queue bằng 2 phương pháp:
  - Phương pháp tịnh tiến
  - Phương pháp vòng

Bài 2:

Quản lý một Queue có số phần tử khá lớn, biến động. Mỗi phần tử có kiểu int (danh sách liên kết đơn)

a. Khai báo cấu trúc queue.

```
struct Node
{
    int info;
    Node* link;
};
Node* front, * rear;
```

b. Viết thủ tục khởi tạo queue rỗng.

```
void init()
{
    front = NULL;
    rear = NULL;
}
```

c. Viết thủ tục kiểm tra queue rỗng.
Queue rỗng nếu như font =NULL.
SV tự viết code (lưu ý: khi hiện thực queue bằng danh sách liên kết đơn thì không có hàm kiểm tra queue đầy

d. Viết thủ tục thêm một phần tử vào queue.
Tương tự insert_last (thêm một phần tử vào cuối danh sách liên kết), SV tự viết code
Viết thủ tục xóa một phần tử trong queue.
Tương tự delete_first (xóa một phần tử ở đầu danh sách liên kết), SV tự viết code

Bài 3:

Dùng cấu trúc danh sách đặc quản lý một đa thức

a. Khai báo cấu trúc danh sách
Một đa thức có dạng: $ax^n + bx^m + \ldots + c$
Khai báo một struct để chứa thông tin 1 thành phần của đa thức gồm:
   - Hệ số (a, b,c…)
   - Số mũ (m, n, ….) với m, n là số nguyên >=0

```
struct dathuc
{
    int heso;
    int somu;
};
```

- Khai báo danh sách đặc có chứa tối đa MAX phần tử

```
dathuc D[MAX]
```

b. Viết thủ tục nhập vào một đa thức

c. Viết thủ tục xuất đa thức

- Dùng vòng lặp do/while hoặc for (nếu xác định trước số lượng phần tử)

```
dathuc dt;
cout << "\n nhap he so\t";
cin >> dt.heso;
cout << "\n nhap so mu~ \t";
cin >> dt.somu;
D[i] = dt;
```

d. Viết thủ tục cộng hai đa thức

- Cộng các hệ số của các phần phần có cùng số mũ
- Sinh viên tự làm

e. Viết thủ tục trừ hai đa thức

- Tương tự thủ tục cộng 2 đa thức

4. BÀI TẬP NÂNG CAO

Bài 4:

Viết thủ tục nhân hai đa thức

Bài 5:

Viết thủ tục chia hai đa thức

5. BÀI TẬP VỀ NHÀ

Bài 1:

Quản lý một danh sách có số phần tử khá lớn, biến động. Mỗi phần tử có kiểu int. Thường có nhu cầu truy xuất phần tử đứng trước và phần tử đứng sau phần tử đang truy xuất. (Dùng cấu trúc danh sách liên kết đôi)

a. Khai báo cấu trúc danh sách.
b. Viết thủ tục khởi tạo danh sách rỗng.
c. Xuất các phần tử trong danh sách
d. Viết thủ tục thêm một phần tử vào đầu danh sách.
e. Viết thủ tục thêm một phần tử vào cuối danh sách.
f. Viết thủ tục xóa phần tử đầu danh sách.
g. Viết thủ tục xóa phần tử cuối dang sách.
h. Viết thủ tục tìm một phần tử trong danh sách. Nếu tìm thấy, xóa phần tử này.
i. Viết thủ tục tìm một phần tử có giá trị bằng với giá trị X hoặc gần nhất và lớn hơn phần tữ nhập vào; Thêm một phần tử đứng trước phần tử tìm thấy.