Stack & Queue

MSSV	20280083
Tên	Lại Toàn Thắng

1. Biên dịch đoạn chương trình trên.
2. Thay giá trị n=10 thành n=1 trong main, đọc giá trị Queue và Stack in ra màn hình.

```
Gia tri phan tu (Queue) : 1

Gia tri phan tu (Stack) : 1
```

3. Giải thích <4> khi nào giá trị pNode khác NULL, khi nào pNode bằng NULL, ý nghĩa của mỗi trường hợp trên.

```
NODE* pNode = DeQueue(pQueue);
if(pNode != NULL) //<4>
    printf("\nGia tri phan tu (Queue) : %d\n", pNode->Key);
else
    printf("\nNULL\n");
```

- Khi pNode khác NULL thì in giá trị pNode đấy ra console. Giá trị đấy là giá trị đã được DeQueue hay có thể hiểu là RemoveTail của một linked list
- Khi pNode bằng NULL thì in "NULL" ra console. Trường hợp này xảy ra khi Queue rỗng và không có gì để Dequeue

4. Giải thích hàm RemoveTail ở các điểm <1>, <2>, <3> cho biết ý nghĩa của chúng.

- <1> Xảy ra với trường hợp khi List rỗng và không có gì để remove.
- <2> Xảy ra khi List chỉ có một Node. Sau khi remove thì list rỗng và trả về kết quả là pHead.
- <3> xảy ra khi List có nhiều hơn một Node. Dòng <3> là dòng lặp while duyệt List đến Node kế cuối.

```
NODE *pNode;
if(pHead == NULL) //<1>
{
    return NULL;
}
else if(pHead->pNext == NULL) //<2>
{
    pNode = pHead;
    pHead = NULL;
    return pNode;
}

pNode = pHead;
while(pNode->pNext->pNext != NULL) //<3>
{
    pNode = pNode->pNext;
}
```

5. Sử dụng các hàm PushStack, PopStack, EnQueue, DeQueue để cài đặt.
    a. Về Stack: Trong hàm main, thực hiện việc thêm vào 3 giá trị do người dùng nhập vào (thực hiện 3 lệnh thêm phần tử vào stack), sau đó thực hiện 4 lần lệnh lấy giá trị phần tử ra khỏi stack, nếu có, in giá trị phần tử ra màn hình, nếu không có (stack rỗng), in ra màn hình "STACK RONG, KHONG LAY DUOC PHAN TU".
    b. Về Queue: Trong hàm main, thực hiện việc thêm vào 3 giá trị do người dùng nhập vào (thực hiện 3 lần lệnh thêm phần tử vào queue), sau đó thực hiện 4 lần lệnh lấy giá trị phần tử ra khỏi queue, nếu có, in giá trị phần tử ra màn hình, nếu không có (queue rỗng), in ra màn hình "QUEUE RONG, KHONG LAY DUOC PHAN TU".

```cpp
int temp = 0;
cin >> temp;
PushStack(pStack, temp);
cin >> temp;
PushStack(pStack, temp);
cin >> temp;
PushStack(pStack, temp);
for (int i = 0; i < 4; i++)
{
    NODE* t = PopStack(pStack);
    if (t != nullptr)
        cout << t->Key << "\t";
    else
        cout << "STACK RONG, KHONG LAY DUOC PHAN TU\n";
}
```

```cpp
int tem = 0;
cin >> tem;
EnQueue(pQueue, tem);
cin >> tem;
EnQueue(pQueue, tem);
cin >> tem;
EnQueue(pQueue, tem);
for (int i = 0; i < 4; i++)
{
    NODE* t = DeQueue(pQueue);
    if (t != nullptr)
        cout << t->Key << "\t";
    else
        cout << "QUEUE RONG, KHONG LAY DUOC PHAN TU\n";
}
return 0;
```

Áp dụng – Nâng cao

1. Cài đặt hàm AddTail để có phiên bản cài đặt Stack (thêm phần tử vào cuối danh sách và lấy phần tử ở cuối danh sách liên kết) cũng như áp dụng 1 phiên bản khác khi cài đặt Queue (thêm phần tử vào cuối danh sách liên kết và lấy phần tử ở đầu danh sách liên kết).
2. Nhận xét cách cài đặt trên ở phần 1 (áp dụng – nâng cao) so với chương trình mẫu đối với trường hợp stack cũng như queue.
3. Sử dụng cấu trúc Stack để chuyển giá trị từ cơ số 10 sang cơ số 2.
   Gợi ý : thực hiện việc chia liên tiếp giá trị trong cơ số 10 cho 2, lấy phần dư đưa vào stack, cho đến khi giá trị đem đi chia là 0. In giá trị trong stack ra (đó chính là kết quả khi chuyển số từ hệ cơ số 10 sang hệ cơ số 2).

```cpp
NODE* AddTail(NODE* &pHead, int data)
{
    NODE* temp = CreateNode(data);
    NODE* pCurr = pHead;
    if (pCurr == nullptr)
        pCurr->pNext = temp;
    else {
        while (pCurr->pNext != nullptr)
            pCurr = pCurr->pNext;
        pCurr->pNext = temp;
    }
    return temp;
}
```

2. Cách cài đặt ở trên, em mô phỏng theo đoạn code có sẵn. Nên em thấy chúng như nhau... <Đoạn này em nói tào lao ạ, em hông hiểu đề bài nói gì🙂>
3. Chuyển cơ số 10 sang 2

```
int n = 0;
cout << "Input n = ";
cin >> n;
while (n != 0) {
    PushStack(pStack, n%2);
    n = n / 2;
}
NODE* pCur = pStack;
while (pCur != nullptr)
{
    cout << pCur->Key;
    pCur = pCur->pNext;
}
```

BÀI TẬP THÊM

Tìm đường trong mê cung (thực hiện loang theo chiều rộng <sử dụng queue> hoặc loang theo chiều sâu <sử dụng stack>).

Bài toán: cho ma trận mxn, mỗi phần tử là số 0 hoặc 1.

Giá trị 1 : có thể đi tới và giá trị 0 : không thể đi tới được.

Câu hỏi:

Từ ô ban đầu có tọa độ (x1, y1) có thể đi tới ô (x2, y2) không?

Biết rằng từ 1 ô (x,y) chỉ có thể đi qua ô có chung cạnh với ô đang đứng và mang giá trị là 1, ngược lại không có đường đi.