20190408-左士海-作业 4

2. 假设以带头结点的循环链表表示队列，并且只设一个表尾指针，试编写相应的置队列空、入队和出队操作。

```cpp
template<class T>
struct Node {
    T data;
    Node<T> * next;
};

template<class T>
class LinkQueue {
    Node<T> *rear;
public:
    LinkQueue();
    ~LinkQueue();
    void push(T x);
    T pop();
};

template<class T>
LinkQueue<T>::LinkQueue() {
    auto * s =new Node<T>;
    rear = s;
    rear->next = rear;
}

template<class T>
LinkQueue<T>::~LinkQueue() {
    auto * p = rear;
    auto *q = p;
    while(p != rear) {
        q = p;
        p = p->next;
        delete q;
    }
}

template<class T>
void LinkQueue<T>::push(T x) {
    auto * s = new Node<T>;
    s->data = x;
    s->next = rear->next;
    rear->next = s;
    rear = s;
}

template<class T>
T LinkQueue<T>::pop() {
    if(rear == rear->next) {
        cerr << "下溢";
        exit(1);
    }
    auto * p = rear->next->next;
    T x = p->data;
    rear->next->next = p->next;
    if(p->next == rear->next)
        rear = rear->next;
    delete p;
    return x;
}
```

带有头结点的循环链表表示队列，只设表尾指针 rear，单链表中 rear 指向 nullptr，循环链表中令 rear 指向头结点来表示队列的 front，判断队列为空或者为满即为 rear == rear->next。

3. 假设以一维数组 data[m]存储循环队列的元素，若要使这 m 个分量都得到应用，则另设一辅助标志变量 flag 判断队列的状态为“空”还是“满”。编写入队和出队算法。

```cpp
template<class T, int MAXSIZE>
class SeqQueue {
    T data[MAXSIZE];
    int front;      // 队头前一个元素的下标
    int rear;       // 队尾下标
    bool flag;      // false 表示队空 true表示队满
    //队空的条件为：front == rear && flag == false
    //队满的条件为：front == rear && flag == true

public:
    SeqQueue();
    void push(T x);     // 将 x 入队
    T pop();        // 将对头出队
};

template<class T, int MAXSIZE>
SeqQueue<T, MAXSIZE>::SeqQueue() {
    front = rear = 0;
    flag = false;
}

template<class T, int MAXSIZE>
void SeqQueue<T, MAXSIZE>::push(T x) {
    if (front == rear && flag) {
        cerr << "上溢";
        exit(1);
    }
    rear = (rear + 1) % MAXSIZE;
    if ((rear - front + MAXSIZE) % MAXSIZE == MAXSIZE) {
        flag = true;
    }
    data[rear] = x;
}

template<class T, int MAXSIZE>
T SeqQueue<T, MAXSIZE>::pop() {
    if(rear == front && !flag) {
        cerr << "下溢" << endl;
        exit(1);
    }
    front = (front + 1) % MAXSIZE;
    if ((rear - front + MAXSIZE) % MAXSIZE == 0) {
        flag = false;
    }
    return data[front];
}
```

使用 flag 标志变量判断队列状态，由于循环数组表示队列是，空或者满的状态都有：front == rear。添加 flag 标志当为 true 时表示队列满，于是有：
队列为空：front == rear && flag == true。
队列为满：front == rear && flag == false。
在每次出入队时对队列长度进行判断是否等于 MaxSize，对 flag 标志变量进行更新。

4. 假设以一维数组 data[m]存放循环队列的元素，同时设变量 num 表示当前队列中元素的个数，以判断队列的状态为“空”还是“满”。试给出此循环队列满的条件，并编写入队和出队算法。

```cpp
template<class T, int MAXSIZE>
class SeqQueue {
    T data[MAXSIZE];
    int front;      // 队头前一个元素的下标
    int rear;       // 队尾下标
    int num;      // 队满时 num == MAXSIZE 队空时 num == 0
public:
    SeqQueue();
    void push(T x);     // 将 x 入队
    T pop();        // 将队头出队
};

template<class T, int MAXSIZE>
SeqQueue<T, MAXSIZE>::SeqQueue() {
    front = rear = num = 0;
}

template<class T, int MAXSIZE>
void SeqQueue<T, MAXSIZE>::push(T x) {
    if (front == rear && num == MAXSIZE) {
        cerr << "上溢";
        exit(1);
    }
    rear = (rear + 1) % MAXSIZE;
    num++;
    data[rear] = x;
}

template<class T, int MAXSIZE>
T SeqQueue<T, MAXSIZE>::pop() {
    if(rear == front && num == 0) {
        cerr << "下溢" << endl;
        exit(1);
    }
    front = (front + 1) % MAXSIZE;
    num--;
    return data[front];
}
```

使用 num 记录当前队列的元素，则队满时 num == MAXSIZE，队空时：num == 0。
每次出入队时对 num 进行更新。

5. 如何用两个栈来实现队列？并写出队列基本操作的算法。

```cpp
template <typename T>
class Queue {
    LinkStack<T> stk1;
    LinkStack<T> stk2;
public:
    // 入队
    void push(T x) {
        stk1.Push(x);
    }
    // 出队
    T pop() {
        if ( ! stk2.Empty() ) {
            return stk2.Pop();
        } else {
            while ( !stk1.Empty() ) {
                stk2.Push(stk1.Pop());
            }
            return stk2.Pop();
        }
    }
};
```

使用两个栈实现队列，栈 1 用于入队，栈 2 用于出队。由于栈的先进后出性质，而队列的先进先出性质，将栈 1 元素出栈再入栈至 2，即为实际的出队顺序。