C++笔记2

模版栈

```cpp
using namespace std;
template <typename T>
class Stack {


public:
    struct ListNode {
        T Val;
        ListNode * Next;

        ListNode(T&& val, ListNode * next = nullptr)
            : Val(val), Next(next)
        {
        }

        ListNode(const T& val, ListNode * next = nullptr)
            : Val(val), Next(next)
        {
        }
};

public:
    explicit Stack()
        : Head(nullptr), Size(0)
    {
    }

    explicit Stack(const Stack<T>& stk)
    {
        ListNode * node = stk.Head;
        while (node)
        {
            push(node->Val);
            node = node->next;
        }
    }

    explicit Stack(Stack<T>&& stk)
        : Head(std::move(stk.Head)), Size(std::move(stk.Size))
    {
    }

    ~Stack()
    {
        while (Size)
            pop();
    }
    void push(T&& val)
    {
        Head = new ListNode(std::move(val), Head);
        Size++;
    }

    void push(const T& val)
    {
        Head = new ListNode(val, Head);
        Size++;
    }

    void pop()
    {
        assert(Size);
        ListNode * delete_node = Head;
        Head = Head->Next;
        delete delete_node;
        Size--;
    }

    bool empty() const
    {
        return !Size;
    }

    size_t size() const
    {
        return Size;
    }

    T& top()
    {
        assert(Size);
        return Head->Val;
    }

    void swap(Stack<T>& stk)
    {
        std::swap(*this, stk);
    }
private:
    ListNode * Head;
    size_t Size;
};
```

模版队列

```
using namespace std;
template <typename T>class Queue
{
public:
    struct QNode
    {
        T Data;
        QNode * Next, *Pre;

        QNode(T&& data, QNode * next = nullptr,QNode * pre = nullpt
r) :
        Data(data), Next(next), Pre(pre)
        {
        }

        QNode(const T&data, QNode * next = nullptr,QNode * pre = nu
llptr) :
        Data(data), Next(next), Pre(pre)
        {
        }
    };
private:
    QNode * Head, *Tail;
    size_t Size;
public:
    explicit Queue() :Head(nullptr), Size(0)
    {
    }

    explicit Queue(const Queue<T>& qn)
    {
        QNode * node = qn.Head;
        while (node)
        {
            push(node->Data);
            node = node->next;
        }
    }

    void push(T&&data)
    {
        Head = new QNode(std::move(data), Head, nullptr);
        Head->Next->Pre = Head;
        Size++;
        if (Size == 1)
        {
            Tail = Head;
        }
```

```cpp
    }

    void push(const T&data)
    {
        Head = new QNode(data, Head, nullptr);
        if (Size == 0)
        {
            Tail = Head;
        }
        else
        {
            Head->Next->Pre = Head;
        }
        Size++;
    }

    void pop()
    {
        assert(Size);
        QNode *Delete_Note = Tail;
        Tail = Tail->Pre;
        delete Delete_Note;
        Size--;
    }

    bool empty() const
    {
        return !Size;
    }

    T& GetData()
    {
        return Tail->Data;
    }

    void swap(Stack<T>& stk)
    {
        std::swap(*this, stk);
    }
};

int main()
{
    Queue<int> que;
    for(int a = 1;a<10;a++)
        que.push(a);
    que.pop();
    que.pop();
    for (int a = 1; a < 8; a++)
    {
        cout << que.GetData();
        que.pop();
    }
    return 0;
}
```