

河南工业大学 数据结构 实验报告

课程名称 数据结构 实验项目 实验二 栈和队列的典型算法实现
院 系 信息科学与工程学院 专业班级 计科 xxxx 班
姓 名 COS 学 号 XXXXXXXXXXXX
指导老师 XXXXXXXX 日 期 2020.10.19
批改日期 _____ 成 绩 _____

一 实验目的

掌握顺序存储和链式存储上,操作受限的线性表的典型操作,掌握栈和队列的操作特点。

二 实验内容及要求

实验内容:

1. 编程实现栈的如下功能:

- (1) 建立一个长度为 n 的顺序栈,元素类型可自行定义,并输出栈中各元素值。
- (2) 将数据元素 e 入栈,并输出入栈后的顺序栈中各元素值。
- (3) 将顺序栈中的栈顶元素出栈,并输出出栈元素的值和出栈后顺序栈中各元素值。

2. 编程实现队列的如下功能:

- (1) 建立一个长度为 n 的循环队列,元素类型可自行定义,并输出队列中各元素值。
- (2) 将数据元素 e 入队,并输出入队后的队列中各元素值。
- (3) 将循环队列的队首元素出队,并输出出队元素的值和出队后队列中各元素值。

3. 编程实现链式栈的如下功能

建立长度为 n 的链式栈,元素类型可自行定义,实现栈的初始化、进栈、出栈等典型操作。

实验要求:

1. 键盘输入数据;
2. 屏幕输出运行结果。
3. 要求记录实验源代码及运行结果。
4. 运行环境: CodeBlocks/Dev c++/VC6.0 等 C 编译环境

三 实验过程及运行结果

1. 整体设计

整体上,一是实现顺序栈和链栈的基本操作:入栈操作(Push)、出栈操作(Pop)、取栈顶元素(Top)、判断栈空(IsEmpty)、判断栈满(IsFull)、获取栈中元素个数(Size)、展示栈中所有元素(PrintStack)、清空栈中所有元素(Clear)。二是实现队列的基本操作:入队操作(Push)、出队操作(Pop)、取队首元素(Front)、判断队空(IsEmpty)、判断队满(IsFull)、获取队中元素个数(Size)、展示队中所有元素(PrintQueue)、清空队中所有元素(Clear)。依旧是封装一个模板类。初始化和销毁操作在构造函数和析取函数中实现。

(1) 顺序栈设计

操作类

名称	类型	修饰符	摘要
方法			
~myStack1		public	析构函数 回收空间
IsEmpty	bool	public	判断栈空与否
IsFull	bool	public	判断栈满与否
myStack1		public	构造函数 分配空间
Pop	T	public	出栈操作,将元素弹出返回,Top值减一
PrintStack	void	public	展示顺序栈中所有元素
Push	void	public	入栈操作,top值加一,存储元素item在栈顶
Size	int	public	获取堆栈元素数量
Top	T	public	取栈顶元素,不弹出
字段			
Data	T*	private	数据
MaxSize	int	private	堆栈容量
top	int	private	记录栈顶元素 为栈顶元素下标后一个下标, 为空时top为0

(2) 链栈设计

结点类

名称	类型	修饰符	摘要
方法			
showdata	void	public	展示该结点数据
SNode		public	空结点
SNode		public	有数据的结点
字段			
Data	T	private	数据
next	SNode	private	指针

操作类定义

名称	类型	修饰符	摘要
方法			
~Stack		public	析构函数 回收空间
Clear	void	public	清空栈中所有元素
IsEmpty	bool	public	判断栈空与否
IsFull	bool	public	判断栈满与否
Pop	T	public	出栈操作 弹出栈顶元素并将其在栈中删除
PrintStack	void	public	展示栈中所有元素
Push	void	public	入栈操作 存储元素data在栈顶
Size	int	public	获取堆栈元素数量
Stack		public	构造函数 分配空间,空头结点
Top	T	public	取栈顶元素,不弹出
字段			
head	SNode<T>*	private	头指针 带空头结点, 所以栈顶元素一直为head->next
maxsize	int	private	堆栈容量

(3) 循环队列设计

操作类

myQueue<T>

模板类

字段

Data

front

MaxSize

rear

方法

~myQueue

Clear

Front

IsEmpty

IsFull

myQueue

Pop

PrintQueue

Push

Size

类详细信息 - myQueue

名称	类型	修饰符	摘要
方法			
<div><div></div><div>~myQueue</div></div>		public	析构函数 回收空间
<div><div></div><div>Clear</div></div>	void	public	清空队中所有元素
<div><div></div><div>Front</div></div>	T	public	取队首元素,不出队
<div><div></div><div>IsEmpty</div></div>	bool	public	判断队空与否 front = rear时队空
<div><div></div><div>IsFull</div></div>	bool	public	判断队满与否 front = (rear+1) % MaxSize时队满
<div><div></div><div>myQueue</div></div>		public	构造函数 分配空间
<div><div></div><div>Pop</div></div>	T	public	出队操作,返回队首元素
<div><div></div><div>PrintQueue</div></div>	void	public	展示循环队列中所有元素
<div><div></div><div>Push</div></div>	void	public	入队操作,rear值加一,存储元素data在队尾rear
<div><div></div><div>Size</div></div>	int	public	获取队元素数量
字段			
<div><div></div><div>Data</div></div>	T*	private	存放数据的指针,申请MaxSize+1的空间只利用MaxSize的空间
<div><div></div><div>front</div></div>	int	private	记录队首元素下标
<div><div></div><div>MaxSize</div></div>	int	private	队列最大容量
<div><div></div><div>rear</div></div>	int	private	记录队尾元素下标 为实际队尾元素的后一个下标,初始时为0

2. 具体实现

顺序栈实现

(1) 构造函数 给Data指针分配空间

核心代码: `Data = new T[maxsize];`

(2) 析构函数 释放给Data指针分配的空间

核心代码: `if (Data) delete[] Data;`

(3) 判断栈空与否 (IsEmpty) 返回true/false

核心代码: `return top == 0;`

(4) 判断栈满与否 (IsFull) 返回true/false

核心代码: `return top == MaxSize;`

(5) 入栈操作: 存储元素item在栈顶(Push)

核心代码:

```
if (IsFull()) {
    cout << "error:Failed to Push,The Stack is Full!" << endl;
} else {
    Data[top++] = data;
}
```

(6) 取栈顶元素操作: 返回栈顶元素,但不将其弹出顺序栈(Top)

核心代码:

```
if (IsEmpty()) {
    cout << "error:The Top isn't existed, The Stack is Empty!" << endl;
    return error; // error为T类型中的特殊值
}
```

```

} else {
    return Data[top - 1];
}

```

(7) 出栈操作：弹出并返回栈顶元素] (Pop)

核心代码：

```

if (IsEmpty()) {
    cout << "error:Failed to Pop, The Stack is Empty!" << endl;
    return error; // error为T类型中的特殊值
} else {
    T temp = Data[--top];
    return temp;
}

```

(8) 展示顺序栈中所有元素(PrintStack)

核心代码：

```

if (IsEmpty())
    cout << "This Stack is empty!" << endl;
for (int i = 0; i < top; ++i) {
    cout << "The " << i + 1 << "th Data is:";
    cout << Data[i] << endl;
}
cout << "The Stack has " << top << " elements in total." << endl;

```

链栈实现

(1) 构造函数 分配空间, 带空头结点

核心代码：head = new SNode<T>;

head->next = nullptr;

(2) 析构函数

将栈中所有元素弹出并释放后释放头指针空间

核心代码：while (head->next) { Pop(); }

if (head) delete head;

(3) 判断栈空与否 (IsEmpty) 返回true/false

链栈这里是判断头结点的下一个结点是否为空

核心代码：if (head->next) return false;

else return true;

(4) 判断栈满与否 (IsFull) 返回true/false

类内封装有这个栈的最大容量maxsize，直接看当前栈是否有这么多元素

核心代码：if (Size() < maxsize) return false;

```
else return true;
```

(5) 入栈操作：存储元素data在栈顶(Push)

判断是否已满，否则先创建一个新结点p并分配空间，再将这个新结点插入在链栈顶

核心代码：

```
if (IsFull()) {  
    cout << "error:Failed to Push,The Stack is Full!" << endl;  
} else {  
    SNode<T>* p = new SNode<T>;  
    p->Data = data;  
    p->next = head->next;  
    head->next = p;  
}
```

(6) 取栈顶元素操作：返回栈顶元素，但不将其弹出顺序栈(Top)

判断是否为空，否则返回栈顶元素（返回值为T）

核心代码：

```
if (IsEmpty()) {  
    cout << "error:The Top isn't existed, The Stack is Empty!" << endl;  
    return Error;  
} else {  
    T temp = head->next->Data;  
    return temp;  
}
```

(7) 出栈操作：弹出并返回栈顶元素(Pop)

核心代码：

```
if (IsEmpty()) {  
    cout << "error:Failed to Pop, The Stack is Empty!" << endl;  
    return Error;  
}  
else {  
    SNode<T>* temp = head->next;  
    T TopData = temp->Data;  
    head->next = temp->next;  
    delete temp;  
    return TopData;  
}
```

(8) 展示栈中所有元素(PrintStack)

核心代码：

```
if (IsEmpty())  
    cout << "This Stack is empty!" << endl;  
SNode<T>* p = head;
```

```

int cnt = 0;
while (p->next) {
    ++cnt;
    p = p->next;
    cout << "The " << cnt << "th Data is:";
    cout << p->Data << endl;
}
cout << "The Stack has " << Size() << " elements in total." << endl;

```

循环队列实现

(1) 构造函数 分配空间, 带空头结点

最大能存储MaxSize个元素, 申请MaxSize+1的空间便于判断循环队列的空与满
核心代码: `Data = new T[MaxSize+1];`

(2) 析构函数 释放空间

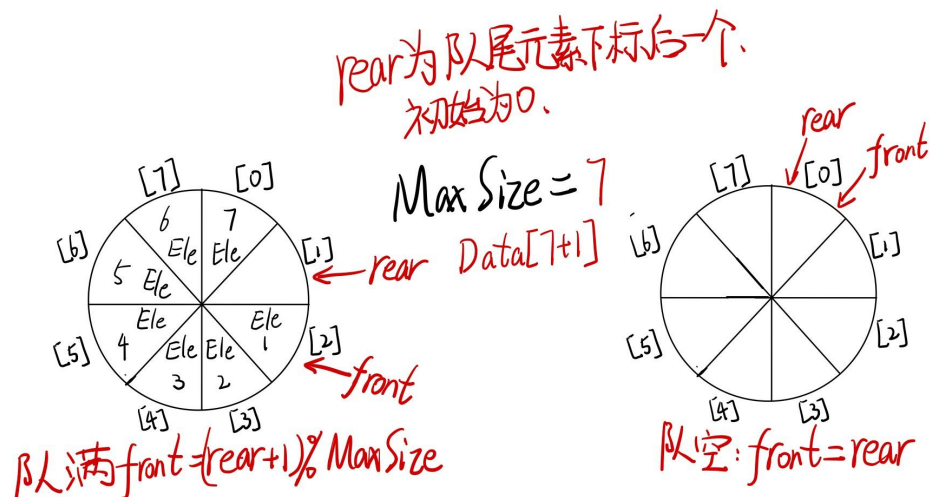
核心代码: `if (Data) delete[] Data;`

(3) 判断队空与否 (IsEmpty) 返回true/false

核心代码: `return front == rear;`

(4) 判断队满与否 (IsFull) 返回true/false

类内封装有这个栈的最大容量MaxSize, 判断思想如图
核心代码: `return front == (rear + 1) % (MaxSize+1);`



(5) 入队操作: 存储元素data在队尾 (Push)

判断是否已满, 否则存储元素data在队尾rear, rear值加一。
核心代码:

```

if (IsFull()) {
    cout << "error:Failed to Push, The Queue is Full!" << endl;
} else {
    Data[rear] = data;
    rear++;
    rear %= MaxSize + 1; //保证不会数组越界
}

```

(6) 取队首元素操作：返回队首元素,但不将其出队(Front)

判断是否为空，否则返回队首元素（返回值为T）

核心代码：

```

if (IsEmpty()) {
    cout << "error:The Top isn't existed, The Queue is Empty!" << endl;
    return error;
} else return Data[front];

```

(7) 出队操作：将队首元素出队(Pop)

判断是否已空，否则将队首元素出队, front值加一, 取余防止越界。

核心代码：

```

if (IsEmpty()) {
    cout << "error:Failed to Pop, The Queue is Empty!" << endl;
    return error;
} else {
    T temp = Data[front];
    front++;
    front %= MaxSize + 1; //保证不会数组越界
    return temp;
}

```

(8) 获取队中元素数量(PrintStack)

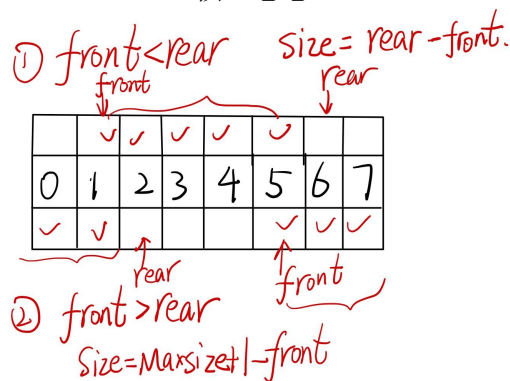
核心代码：

```

if (IsFull())
    return MaxSize;
if (IsEmpty())
    return 0;
if (front < rear) //正常情况 front在前
    return rear - front;
else {
    //front在rear后，先到数组尾部，再从头开始到rear
    int size = MaxSize + 1 - front;
    size += rear;
}

```


核心思想



3. 运行结果

顺序栈运行结果：

- (1) 建立一个长度为 n 的顺序栈，元素类型可自行定义，并输出栈中各元素值。

```
E:\Programming\数据结构\class\实验\计科201916010728余思娴实验2\...
输入n, 建立长度为n的顺序栈:5
输入n个元素:
6 7 8 9 10
1 入栈操作
2 出栈操作
3 取栈顶元素
4 取堆栈元素个数
5 输出顺序栈中所有元素
6 结束
菜单选择:5
----- Stack S -----
The 1th Data is:10
The 2th Data is:9
The 3th Data is:8
The 4th Data is:7
The 5th Data is:6
The Stack has 5 elements in total.
按回车键继续...
```

- (2) 将数据元素 55 入栈，此时已满入栈失败，弹出一个元素再入栈。

```
菜单选择:1
请输入要入栈的元素:55
error:Failed to Push, The Stack is Full!
----- After Push, Stack S -----
The 1th Data is:10
The 2th Data is:9
The 3th Data is:8
The 4th Data is:7
The 5th Data is:6
The Stack has 5 elements in total.
按回车键继续...
菜单选择:2
出栈元素为:10
----- After Pop, Stack S -----
The 1th Data is:9
The 2th Data is:8
The 3th Data is:7
The 4th Data is:6
The Stack has 4 elements in total.
按回车键继续...
菜单选择:1
请输入要入栈的元素:55
----- After Push, Stack S -----
The 1th Data is:55
The 2th Data is:9
The 3th Data is:8
The 4th Data is:7
The 5th Data is:6
The Stack has 5 elements in total.
按回车键继续...
菜单选择:3
栈顶元素为:55
按回车键继续...
```


链栈运行结果:

(1) 建立一个长度为 n 的顺序栈 输出栈中各元素, 此时入判断栈已满, 则入栈失败

```
选择E:\Programming\数据结构\class\实验\计科201916010728余思鹏
输入n, 建立长度为n的顺序栈:4
输入n个元素:
1 2 3 4
1 入栈操作
2 出栈操作
3 取栈顶元素
4 取堆栈元素个数
5 输出栈中所有元素
6 清空栈中所有元素
7 结束
菜单选择:5
----- Stack S -----
The 1th Data is:4
The 2th Data is:3
The 3th Data is:2
The 4th Data is:1
The Stack has 4 elements in total.
按回车键继续...

菜单选择:1
请输入要入栈的元素:4
error:Failed to Push, The Stack is Full!
----- After Push, Stack S -----
The 1th Data is:4
The 2th Data is:3
The 3th Data is:2
The 4th Data is:1
The Stack has 4 elements in total.
按回车键继续...

菜单选择:3
栈顶元素为:4
按回车键继续...

菜单选择:4
该堆栈中元素个数为:4
按回车键继续...
```

(2) 清空链栈, 此时栈中无元素, 提示出栈失败, 测试各功能正常

```
选择E:\Programming\数据结构\class\实验\计科201916010728余
菜单选择:6
成功清空!
按回车键继续...

菜单选择:2
error:Failed to Pop, The Stack is Empty!
----- After Pop, Stack S -----
This Stack is empty!
The Stack has 0 elements in total.
按回车键继续...

菜单选择:3
error:The Top isn't existed, The Stack is Empty!
按回车键继续...

菜单选择:4
该堆栈中元素个数为:0
按回车键继续...

菜单选择:5
----- Stack S -----
This Stack is empty!
The Stack has 0 elements in total.
按回车键继续...

菜单选择:1
请输入要入栈的元素:7
----- After Push, Stack S -----
The 1th Data is:7
The Stack has 1 elements in total.
按回车键继续...
```

循环队列运行结果:

(1) 建立一个长度为 n 的循环队列, 元素类型可自行定义, 并输出队中各元素值。队满时入队失败, 有提示

```
D:\caomeilin\Queue\bin\Debug\Queue.exe
Please input n:4
Please input n element:
1 2 3 4
1 Queue_Push
2 Queue_Pop
3 Queue_Front
4 Queue_Clear
5 Queue_Size
6 Queue_PrintQueue
7 Exit
Please chose:6
----- Queue Q -----
The 1th Data is:1
The 2th Data is:2
The 3th Data is:3
The 4th Data is:4
The Queue has 4 elements in total.
Press Enter to continue...

Please chose:3
Queue_Front:1
Press Enter to continue...

Please chose:5
Queue_Size:4
Press Enter to continue...

Please chose:1
Queue_Push:5
error:Failed to Push, The Queue is Full!
----- After Push, Queue Q -----
The 1th Data is:1
The 2th Data is:2
The 3th Data is:3
The 4th Data is:4
The Queue has 4 elements in total.
Press Enter to continue...
```

(2) 将队列清空, 测试各功能正常

```

选择D:\caomeilin\Queue\bin\Debug\Queue.exe
Please chose:
4
Cleared successfully!
----- After Clear, Queue Q -----
This Queue is empty!
Press Enter to continue...

Please chose:3
error:The Front isn't existed, The Queue is Empty!
Press Enter to continue...

Please chose:5
Queue_Size:0
Press Enter to continue...

Please chose:6
----- Queue Q -----
This Queue is empty!
Press Enter to continue...

Please chose:2
error:Failed to Pop, The Queue is Empty!
----- After Pop, Queue Q -----
This Queue is empty!
Press Enter to continue...

Please chose:1
Queue_Push:6
----- After Push, Queue Q -----
The 1th Data is:6
The Queue has 1 elements in total.
Press Enter to continue...
Please chose:3
Queue_Front:6
Press Enter to continue...

Please chose:6
----- Queue Q -----
The 1th Data is:6
The Queue has 1 elements in total.
Press Enter to continue...

Please chose:7

Process returned 0 (0x0)   execution time : 100.7
Press any key to continue.
```

四 调试情况、设计技巧及体会

1. 调试情况

问题 1：输出时频频出现中文乱码

解决办法：使用 VS2019 输出就没乱码了，多半是编码问题

问题 2：循环队列判断空满、获取元素数量以及遍历时越界

解决办法：取余，使 front 与 rear 不会越界。

2. 设计技巧

- ① 每次插入或删除时都要考虑特殊情况，如栈空/栈满/队空/队满等
- ② 封装模板类，调用的时候想调用什么类型就调用什么类型，如 `Queue<int>`、`Queue<double>`、`Queue<float>`、乃至自定义类型
- ③ 将 STL 中 `Stack`、`Queue` 的操作都简单实现一遍也就是基本操作了

3. 体会

栈、队列的代码还是比较容易的，写好些，就是要注意一些细枝末节的地方，由于时间关系还是写的太仓促了，不过基本功能的实现暂时没有问题