《数据结构与算法》实验报告

实验三：栈和队列

**教 师：潘晔**

**学 生：梁书恺**

**学 号：2022040906023**

**时 间：10.9**

**地 点：科B119**

**一、ex3-1**

1. **问题分析：**
2. **题意理解**

从键盘输入元素，逐个进栈，输入0结束，显示进栈后的数据元素，调用若干次出栈函数，显示出栈后的数据元素，统计需要调用多少次使栈为空。

1. **数据结构设计**

typedef unsigned int DataType;

typedef struct Node

{

DataType data;

struct Node \*next;

} Node;

typedef struct

{

Node \*top;

int length;

} Lstack;

1. **关键算法思路**

int pop(Lstack \*LS)

{

Node \*p;

DataType ptr;

p = LS->top;

if (LS->top == NULL)

errorHandler(2);

ptr = LS->top->data;

LS->top = LS->top->next;

LS->length--;

free(p);

return ptr;

}

1. **健壮性设计**

通过调用统一出错处理函数errorHandler打印出错信息并跳出程序。

1. **性能分析**

时间复杂度：O(n);

空间复杂度：O(n);

1. **测试数据与运行结果截图：**

请输入数据元素：

1

2

3

4

5

0

进栈后的数据元素：

5 4 3 2 1

出栈的数据元素：

5 4 3 2 1

出栈次数：5

请输入数据元素：

1

1

4

5

1

4

1

9

1

9

8

1

0

进栈后的数据元素：

1 8 9 1 9 1 4 1 5 4 1 1

出栈的数据元素：

1 8 9 1 9 1 4 1 5 4 1 1

出栈次数：12

请输入数据元素：

1

2

-1

输入错误！

1. **上机时遇到的问题**
2. **问题现象：** pop弹出异常 **原因：**想传入一个指针，储存弹出值的地址，但节点被释放 **解决办法：**直接使用弹出值作为返回值
3. **程序代码**

ex1.c

ex1.h

**二、ex1-2**

1. **问题分析：**
2. **题意理解**

定义循环队列CQueue，使用两种方法判别队空队满，并编写入队、出队等函数。

1. **数据结构设计**

typedef struct

{

DataType data[QueueSize];

int front, rear;

} CQueue;

1. **关键算法思路**

// 循环队列出队

int deQueue(CQueue \*cq, DataType \*ptr)

{

if ((getState1() == 1) && (getState2(cq) == 1))

{

errorHandler(1);

}

else

{

\*ptr = cq->data[cq->front];

cq->front = (cq->front + 1) % MAXNUM;

counter--;

flag = 0;

}

return 1;

}

int enQueue(CQueue \*cq, DataType x)

{

if ((getState1() == 2) && (getState2(cq) == 2))

{

errorHandler(1);

}

else

{

cq->data[cq->rear] = x;

cq->rear = (cq->rear + 1) % MAXNUM;

counter++;

if (cq->front == cq->rear)

{

flag = 1;

}

return 1;

}

}

1. **健壮性设计**

通过调用统一出错处理函数errorHandler打印出错信息并跳出程序。

1. **性能分析**

时间复杂度：O(n);

空间复杂度：O(n);

1. **测试数据与运行结果截图：**

请输入数据元素：

1

-2

3

-4

5

调用aa前的数据元素：

1 -2 3 -4 5

调用aa后的数据元素：

1 3 5

请输入数据元素：

1

-1.1

输入错误！

1. **上机时遇到的问题**
2. **问题现象：** 判断队空队满出现问题 **原因：**front和rear的判断条件有误； **解决办法：**修改判断条件
3. **程序代码**

ex2.c

ex2.h

**三、ex1-3**

1. **问题分析：**
2. **题意理解**

建立一个数据结构，两个栈公用一个存储数组，拥有两对不同的栈顶和栈底，编写该数据结构的压栈、弹出等操作。

1. **数据结构设计**

typedef struct

{

int data[StackSize];

int L\_top, R\_top;

} Stack;

1. **关键算法思路**

int get(Stack \*stack, int \*ptr, int id)

{

if (id == 0)

{

\*ptr = stack->data[stack->L\_top - 1];

printf("左栈顶元素为:%d\n", \*ptr);

}

else if (id == 1)

{

\*ptr = stack->data[stack->R\_top + 1];

printf("右栈顶元素为:%d\n", \*ptr);

}

return 1;

}

1. **健壮性设计**

通过调用统一出错处理函数errorHandler打印出错信息并跳出程序。

1. **性能分析**

时间复杂度：O(1);

空间复杂度：O(1);

1. **测试数据与运行结果截图：**

左栈顶元素为:8

右栈顶元素为:9

左栈压入元素为:10

左栈顶元素为:10

右栈弹出元素为:9

右栈顶元素为:7

左栈弹出元素为:10

左栈顶元素为:8

右栈压入元素为:11

右栈顶元素为:11

左栈压入元素为:12

左栈顶元素为:12

右栈压入元素为:13

右栈顶元素为:13

1. **上机时遇到的问题**

无

1. **程序代码**

ex3.c

ex3.h

**四、ex1-4**

1. **问题分析：**
2. **题意理解**

通过键盘输入表达式，检测表达式的括号是否匹配

1. **数据结构设计**

typedef struct

{

char data[STACK\_SIZE];

int size;

} Stack;

1. **关键算法思路**

int check(char \*str)

{

Stack stack;

initStack(&stack);

char \*ptr = str;

while (\*ptr != '\0')

{

if (\*ptr == '(')

{

push(&stack, \*ptr);

}

else if (\*ptr == ')')

{

char x;

pop(&stack, &x);

}

ptr++;

}

if (stack.size == 0)

{

return 1;

}

else

{

return 0;

}

}

1. **健壮性设计**

通过调用统一出错处理函数errorHandler打印出错信息并跳出程序。

1. **性能分析**

时间复杂度：O(n);

空间复杂度：O(1);

1. **测试数据与运行结果截图：**

请输入一个表达式：

(a+b)\*c

括号匹配！

请输入一个表达式：

(a\*(c+d)/(e-f)+g)\*h\*(i\*j(k+l)-m)/n

括号匹配！

请输入一个表达式：

)a+b(

括号不匹配！

请输入一个表达式：

(a\*(c+d)/(e-f)+g\*h\*(i\*j(k+l)-m)/n

括号不匹配！

1. **上机时遇到的问题**
2. **问题现象：** 在先出现右括号时出现问题 **原因：**出现下溢错误，进入错误处理函数 **解决办法：**不进入错误处理，pop函数返回是否成功
3. **程序代码**

ex4.c

ex4.h

**小结**

本章主要学习栈和队列。对于链栈，在弹出数据的时候，由于要释放其内存，不能使用指针参数来得到弹出值。对于循环队列这一数据结构，最大的难点是判断它的队空和队满，对于rear和head的状态要特别注意。