

《数据结构课程实践》

**实验报告**

**实验名称：顺序栈和链栈**

**姓 名：杨天诏**

**学 号：20081831**

**专 业：通信工程**

**实验时间：10月24日**

**杭州电子科技大学**

**通信工程学院**

**（实验报告要求：填写三四五一共三部分内容，实验报告文件命名方式：*数据结构课程实践2021\_实验报告4-姓名-学号*，完成后上传电子版到校园网的网络教学平台，无需打印）**

1. **实验目的**

掌握顺序栈和链栈的基本算法设计。

1. **实验内容**

1.（实验指导书58页实验3.1）实现顺序栈基本运算的算法。1）其中进栈元素改为数字“1 2 3 4 5 6”一共6个元素；2）在参考程序代码之外，调用GetTop()方法查看结果

2.（实验指导书61页实验3.2）实现链栈基本运算的算法。

3. （附加题）判断表达式重点括号是否配对。输入表达式加以验证：（1）a\*{b\*[1/(2-c)+1/(1+d)]-3}

（2）a\*{b\*1/(2-c)+1/(1+d)]-3}（3）a\*{b\*1/(2-c]+1/[1+d)-3}

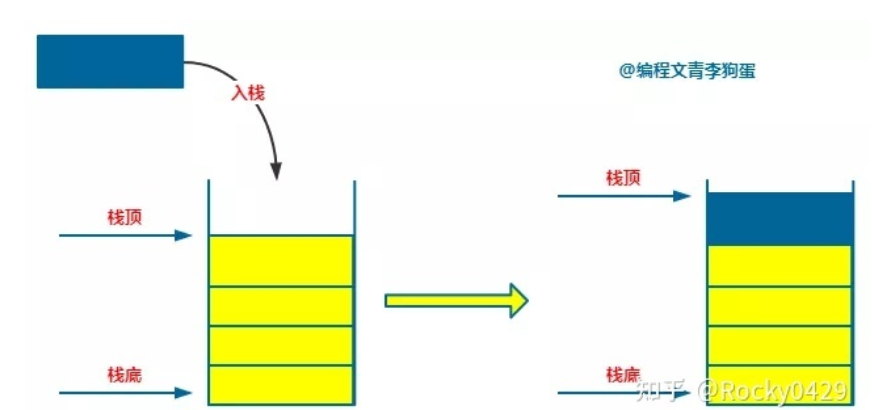
1. **算法设计**
2. 栈这种数据结构的最基本的特征是什么？顺序栈和链栈的基本操作有哪些不同？

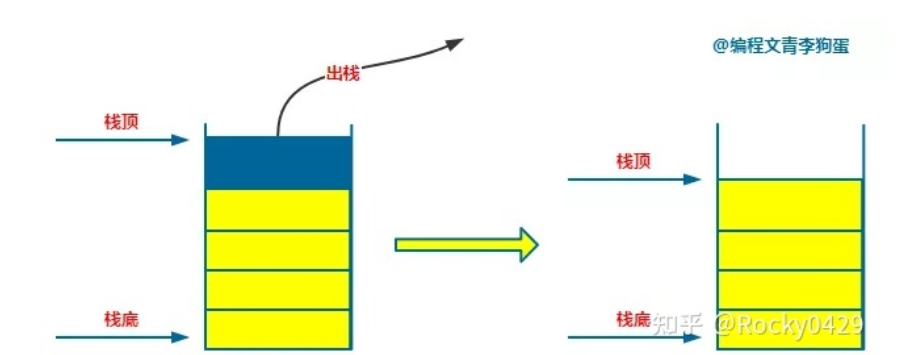
栈的最主要特点是后进先出。栈是一种运算受限的线性表，限定仅在表尾进行插入和删除操作的线性表。栈的所有插入和删除操作均在栈顶进行，而栈底不允许插入和删除。

顺序栈是用一组连续的存储空间来依次存放栈元素，在创建顺序栈时需指定存储空间的长度，元素从栈底到栈顶依次放置， 即每次入栈和出栈都在栈顶完成。入栈时在顺序表尾插入新的元素， 出站时取出表尾元素即可。

链栈是链式存储结构，如果选择从链尾作为栈顶，即每次入栈时往链表尾部添加一个节点元素， 每次出站时取出链尾元素。由于链表查询时需要从表头开始，导致每次出栈和出栈都要遍历整个链表后才能找到栈顶元素，会造成链栈的效率低。所以选择链表的第一个元素作为栈顶，入栈和出栈都只需要找到第一个元素，入栈时断开表头和链表的第一个元素之间的指针，将表头指针指向新的节点元素，再将新的节点元素的指针指向原来的第一个元素节点即可，出栈时通过表头指针可直接找到出栈元素，出栈完成后删除第一个节点。

1. 对于实验中链栈的进栈和出栈操作，分别画图表示其过程步骤。





1. 给出附加题的算法代码，并画示意图说明算法思想。

**bool Match(char exp[], int n)**

**{**

**int i = 0; char e;**

**bool match = true;**

**LinkStNode\* st;**

**InitStack(st);**

**while (i < n && match)**

**{**

**if (exp[i] == '(')**

**Push(st,exp[i]);**

**else if (exp[i] == ')')**

**{**

**if (GetTop(st, e) == true)**

**{**

**if (e != '(')**

**match = false;**

**else**

**Pop(st, e);**

**}**

**else match = false;**

**}**

**else if (exp[i] == '[')**

**Push(st,exp[i]);**

**else if (exp[i] == ']')**

**{**

**if (GetTop(st,e) == true)**

**{**

**if (e != '[')**

**match = false;**

**else**

**Pop(st,e);**

**}**

**else match = false;**

**}**

**else if (exp[i] == '{')**

**Push(st,exp[i]);**

**else if (exp[i] == '}')**

**{**

**if (GetTop(st,e) == true)**

**{**

**if (e != '{')**

**match = false;**

**else**

**Pop(st,e);**

**}**

**else match = false;**

**}**

**++i;**

**}**

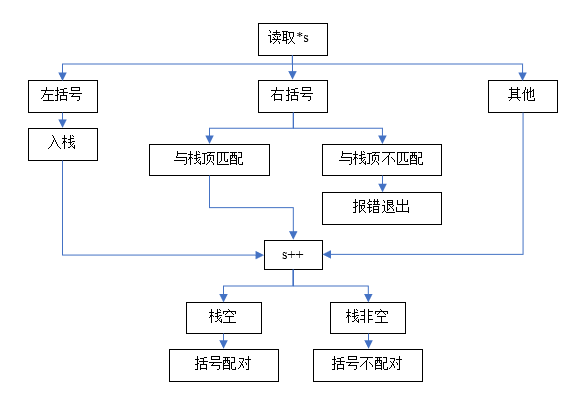
**if (!StackEmpty(st))**

**match = false;**

**DestroyStack(st);**

**return match;**

**}**

****

1. **运行结果与分析（给出运行结果截图，如有必要，请加以简要说明）**
2. **实验3-1**

#include "x.h"

int main()

{

//定义顺序栈st

SqStack\* st;

ElemType e;

printf("(1)初始化栈st\n");

InitStack(st);

printf("(2)栈为%s\n", (StackEmpty(st) == 1 ? "空" : "不空"));

Push(st, '1');

Push(st, '2');

Push(st, '3');

Push(st, '4');

Push(st, '5');

printf("(3)依次进栈元素1,2,3,4,5\n");

printf("(4)栈为%s\n", (StackEmpty(st) == 1 ? "空" : "不空"));

GetTop(st, e);

printf("(5)栈顶元素:%c\n", e);

printf("(6)出栈顺序：");

//栈不空循环

while (!StackEmpty(st))

{

//出栈元素e并输出

Pop(st, e);

printf("%c ", e);

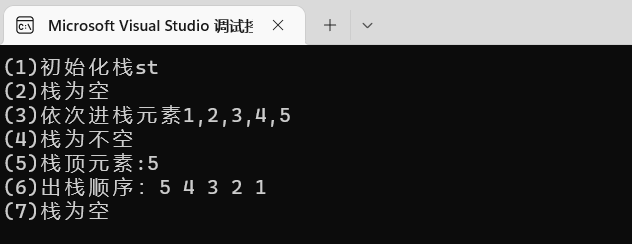
}

printf("\n");

printf("(7)栈为%s\n", (StackEmpty(st) == 1 ? "空" : "不空"));

DestroyStack(st);

}

****

**(2)实验3-2**

#include "y.h"

int main()

{

ElemType e;

linknode\* s;

printf("(1)初始化链栈s\n");

InitStack(s);

printf("(2)链栈为%s\n", (StackEmpty(s) ? "空" : "非空"));

printf("(3)依次进链栈元素a,b,c,d,e\n");

Push(s, 'a');

Push(s, 'b');

Push(s, 'c');

Push(s, 'd');

Push(s, 'e');

printf("(4)链栈为%s\n", (StackEmpty(s) ? "空" : "非空"));

printf("(5)出链栈序列:");

while (!StackEmpty(s))

{

Pop(s, e);

printf("%c ", e);

}

printf("\n");

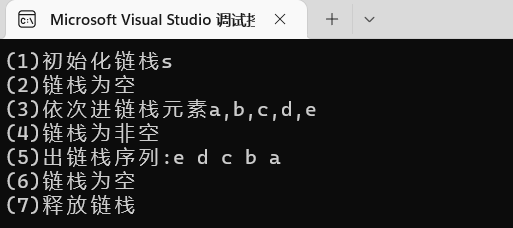
printf("(6)链栈为%s\n", (StackEmpty(s) ? "空" : "非空"));

printf("(7)释放链栈\n");

DestroyStack(s);

return 0;

}

****

**附加题**

**（1）a\*{b\*[1/(2-c)+1/(1+d)]-3}**

**（2）a\*{b\*1/(2-c)+1/(1+d)]-3}**

**（3）a\*{b\*1/(2-c]+1/[1+d)-3}**

#include"y.h"

int main()

{

char a[] = "a\*{b\*[1/(2-c)+1/(1+d)]-3}";

char b[] = "a\*{b\*1/(2-c)+1/(1+d)]-3}";

char c[] = "a\*{b\*1/(2-c]+1/[1+d)-3}";

if (Match(a, strlen(a)) == 1)

printf("表达式%s括号配对\n", a);

else

printf("表达式%s括号不配对\n", a);

if (Match(b, strlen(b)) == 1)

printf("表达式%s括号配对\n", b);

else

printf("表达式%s括号不配对\n", b);

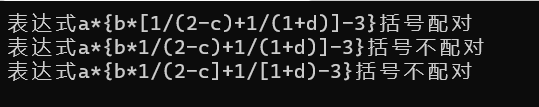
if (Match(c, strlen(c)) == 1)

printf("表达式%s括号配对\n", c);

else

printf("表达式%s括号不配对\n", c);

}

****

1. **实验小结（记录实验过程中遇到的主要问题和心得）**

通过本次实验，掌握顺序栈和链栈的基本算法设计，加深了对顺序栈和链栈的理解。