数据结构实验报告——实验 X

学号：_20201050331__姓名：_黄珀芝__得分：_____

一、实验目的

1、复习栈的逻辑结构、存储结构及基本操作；
2、掌握顺序栈、链栈。

二、实验内容

1、(课堂完成)假设栈中数据元素类型是字符型,请采用顺序栈实现栈的以下基本操作:
(1) Status InitStack(&s)构造空栈 S;
(2) Status Push(&s,e)∥元素 e 入栈 S;
(3)Status Pop(&s,&e)/栈 s 出栈,元素为 e；

2、(必做题)请实现:对于一个可能包括括号{} []的表达式,判定其中括号是否匹配。
3、(选做题)请实现:采用算符优先分析法分析输入的算术表达式语法是否正确,若表达式语法正确,请输出运算结果;否则输出提示“表达式错误!”

三、数据结构及算法描述

实验一：
数据结构：顺序存储（一组地址连续的存储空间）、入栈、出栈。
操作：S.top=S.base;

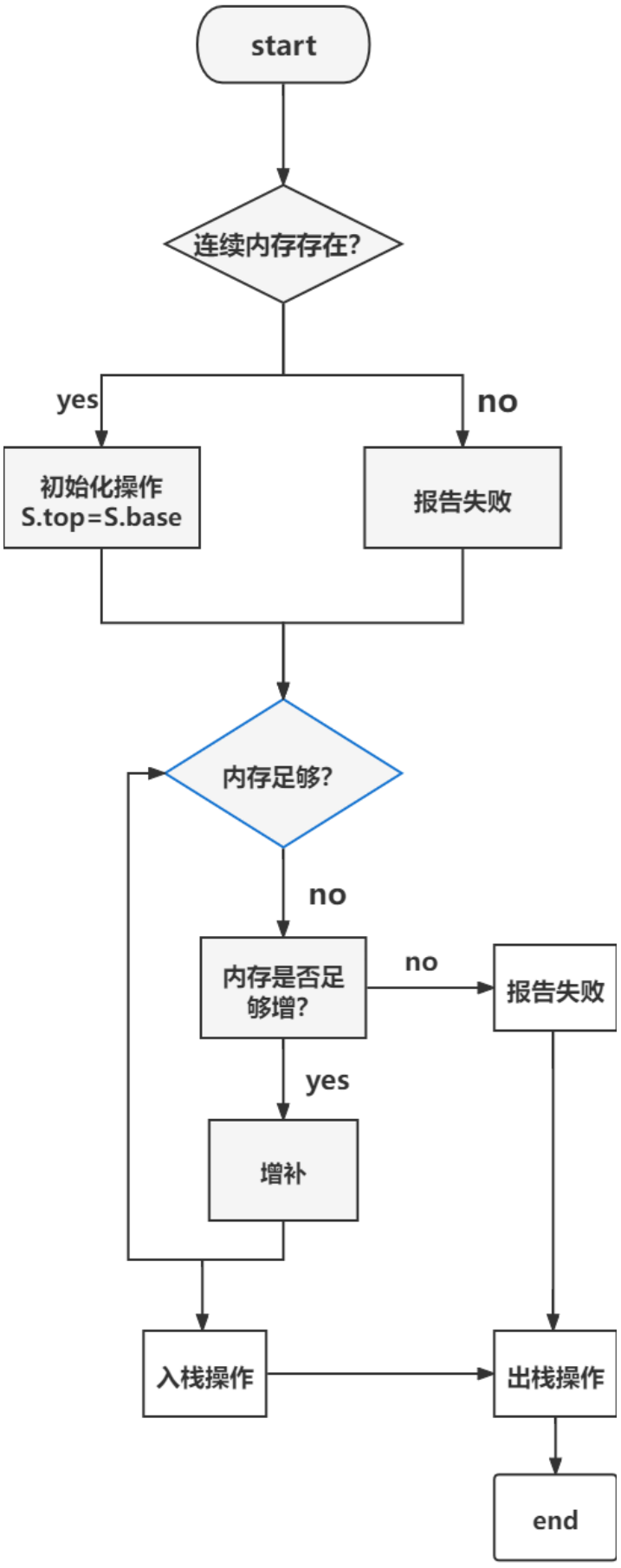
算法描述：（1）使用一组地址连续的存储单元依次存放自栈底到栈顶的数据元素。
（2）以指针 top 指示栈顶元素在顺序栈中的位置，以指针 top 指示栈底元素在顺序栈中的位置。
（3）初始化时令 top=base 得到空栈。入栈时首先把待插入元素写入这个位置，然后使栈顶指针上移一个位置。
（4）出栈时则是使栈顶指针下移一个位置。在主函数中调用并检验。

实验二：
数据结构：入栈、出栈、函数调用

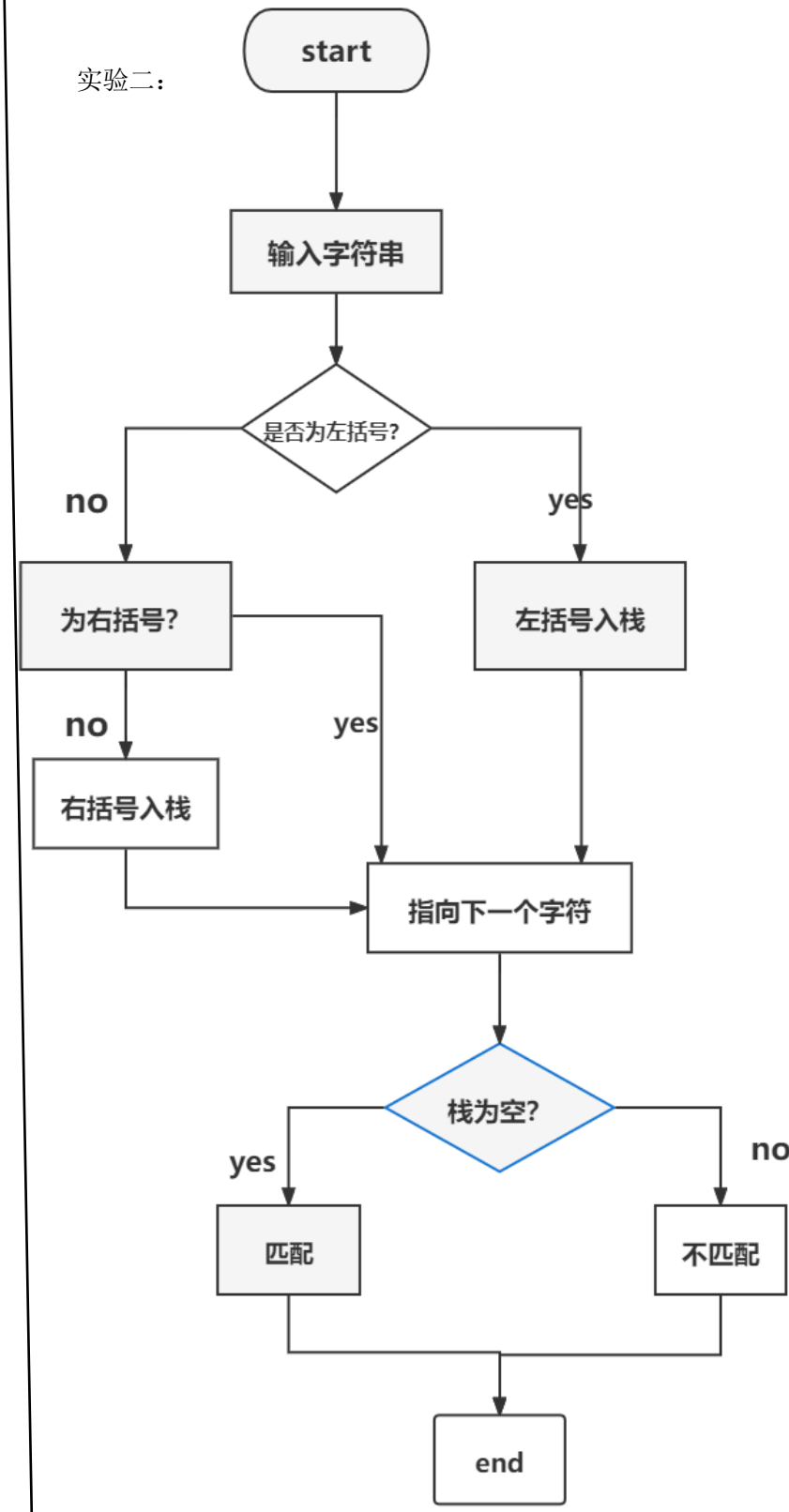
算法描述：（1）从左到右判断，遇上左括号就入栈，遇上右括号就弹出一个左括号.
（2）如果栈空依旧需要弹出左括号，则右括号不配对.
（3）如果表达式完成之后栈不空，则左括号不配对。
（4）表达式完成后栈空则括号配对，在主函数中调用并检验。

四、详细设计

实验一：



实验二：



五、程序代码

实验一：

```
#include <stdio.h>
#include <stdlib.h>

#define OK
#define ERROR
#define OVERFLOW
#define STACK_INIT_SIZE
#define STACKINCREMENT

typedef int Status;
typedef char Selemtype;

typedef struct //栈的结构体定义
{
    Selemtype *base;
    Selemtype *top;
    int stacksize;
}Sqstack;

Status InitStack(Sqstack &S)   //初始化
{
    S.base=(Selemtype*)malloc(STACK_INIT_SIZE*sizeof(Selemtype));
    if(!S.base)exit(OVERFLOW);
    S.top=S.base;
    S.stacksize=STACK_INIT_SIZE;
    return OK;
}

Status Push(Sqstack &S,Selemtype e)//入栈
{
    if(S.top-S.base>=S.stacksize)
    {

S.base=(Selemtype*)realloc(S.base,(S.stacksize+STACKINCREMENT)*sizeof(Selemtype));
        if(!S.base)
            exit(OVERFLOW);
        S.top=S.base+S.stacksize;
        S.stacksize+=STACKINCREMENT;
    }
```

```
    *S.top++=e;
    return OK;
}

Status Pop(Sqstack &S,Selemtype &e)//出栈
{
    if(S.top==S.base)
        return ERROR;
    e=*--S.top;
    return OK;
}

int main()
{
    int i,n;
    Selemtype e;
    Sqstack S;
    if(InitStack(S))
        printf("初始化栈已经成功！\n");
    else
        printf("初始化栈未成功！\n");
    printf("\n 请输入要入栈的元素个数（大于 0 的数字）：\n");
    scanf("%d",&n);
    if(n<=0)
    {
        printf("\n 输入错误导致程序结束运行！");
        return 0;
    }
    else
    {
        printf("\n 请输入要入栈的元素：\n");
        getchar();
        for(i=0;i<n;i++)
        {
            scanf("%c",&e);
            if(Push(S,e))
                printf("第%d 个元素已经入栈成功！\n",i+1);
        }
    }
    printf("\n 请输入要出栈的元素个数（小于等于已入栈的个数）：\n");
    scanf("%d",&n);
    if(n<=0)
    {
        printf("\n 输入错误导致程序结束运行！");
```

```
        return 0;
    }
    else
    {
        for(i=0;i<n;i++)
        {
            Pop(S,e);
            printf("第%d 个出栈元素为：\t%c\n",i+1,e);
        }
    }
    return 0;
}
```

实验二；

```
#include <stdio.h>
#include <stdlib.h>
#define MaxSize 100

typedef char Elemtype;

typedef struct
{
    Elemtype data[MaxSize];
    int top;
} SqStack;

//初始化
void InitStack(SqStack *S)
{
    S->top = -1;
}

//判栈空
int IsEmpty(SqStack S)
{
    if(S.top == -1)
        return 1;
    else
        return 0;
}

//进栈
void Push(SqStack *S, Elemtype x)
{
```

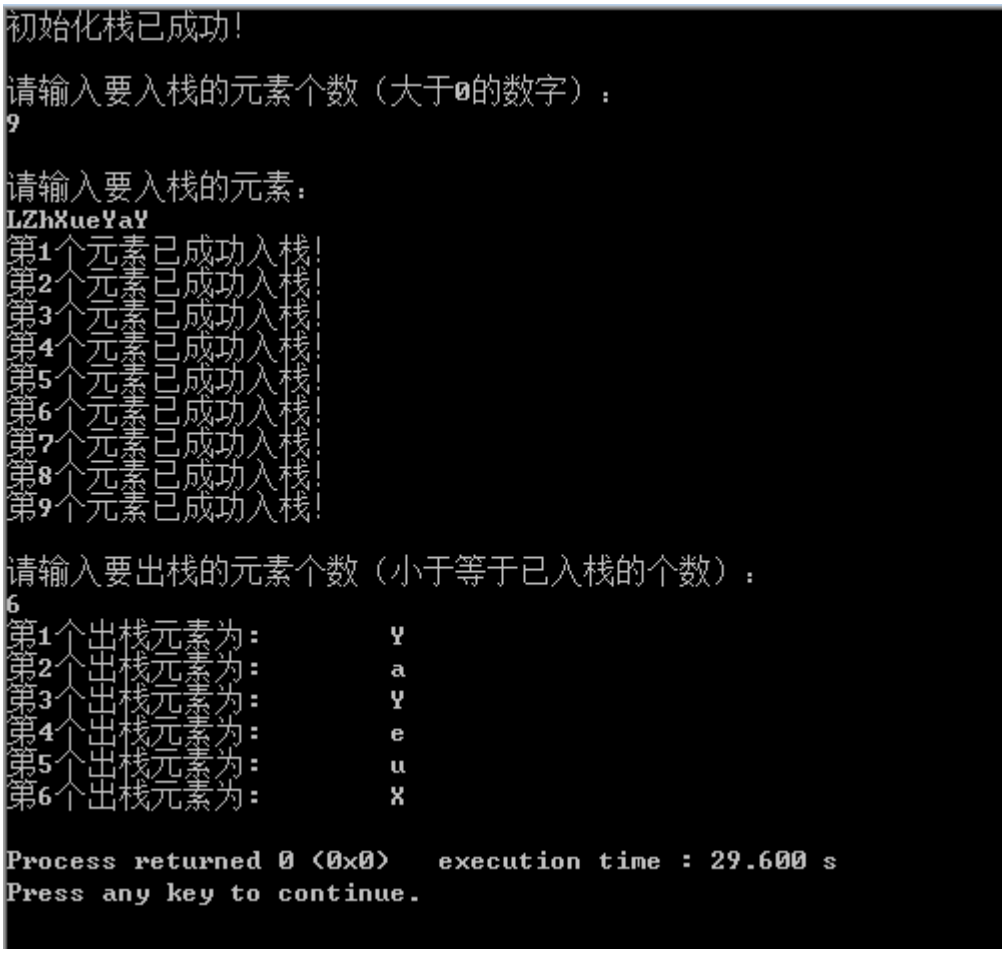
```
    if(S->top != MaxSize-1)     //栈满:S->top == MaxSize-1
        S->data[++S->top] = x; //指针先加 1，再入栈
}

//出栈
Elemtype Pop(SqStack *S)
{
    if(S->top != -1) //栈空:S->top == -1
        return S->data[S->top--]; //先出栈，指针再减 1
}


//判断{},[],()是否匹配
int BracketsCheck(Elemtype str[])
{
    SqStack S;
    InitStack(&S);
    int i = 0;
    char e;
    while(str[i] != '\0')
    {

        switch(str[i])
        {
            // 凡是左括号，入栈
        case '{':
            Push(&S,'{');
            break;
        case '[':
            Push(&S,'[');
            break;
        case '(':
            Push(&S,'(');
            break;
            //凡是右括号，出栈，然后与栈顶匹配
        case '}':
            e = Pop(&S);
            if(e != '{') //不匹配
                return 0;
            break;
        case ']':
            e = Pop(&S);
            if(e != '[') //不匹配
                return 0;
```

```
            break;
        case ')':
            e = Pop(&S);
            if(e != '(') //不匹配
                return 0;
            break;
        default:
            break;
        }

        i++;
    }

    if(IsEmpty(S)==1)
        return 1; //栈空，括号匹配
    else
        return 0;//栈不空，括号不匹配
}


int main()
{
    int i,n;
    Elemtype str[100];
    SqStack *S;
    InitStack(S);
    printf("请输入需要检验的表达式数量：\n");
    scanf("%d",&n);
    if(n<=0)
    {
        printf("输入错误导致程序结束！");
        return 0;
    }
    else
    {
        for(i=0; i<n; i++)
        {

            printf("\n 请输入第%d 个表达式：\n",i+1);
            fflush(stdin);
            gets(str);
            BracketsCheck(str);
            if(BracketsCheck(str)==1)
                printf("第%d 个表达式匹配\n",i+1);
```

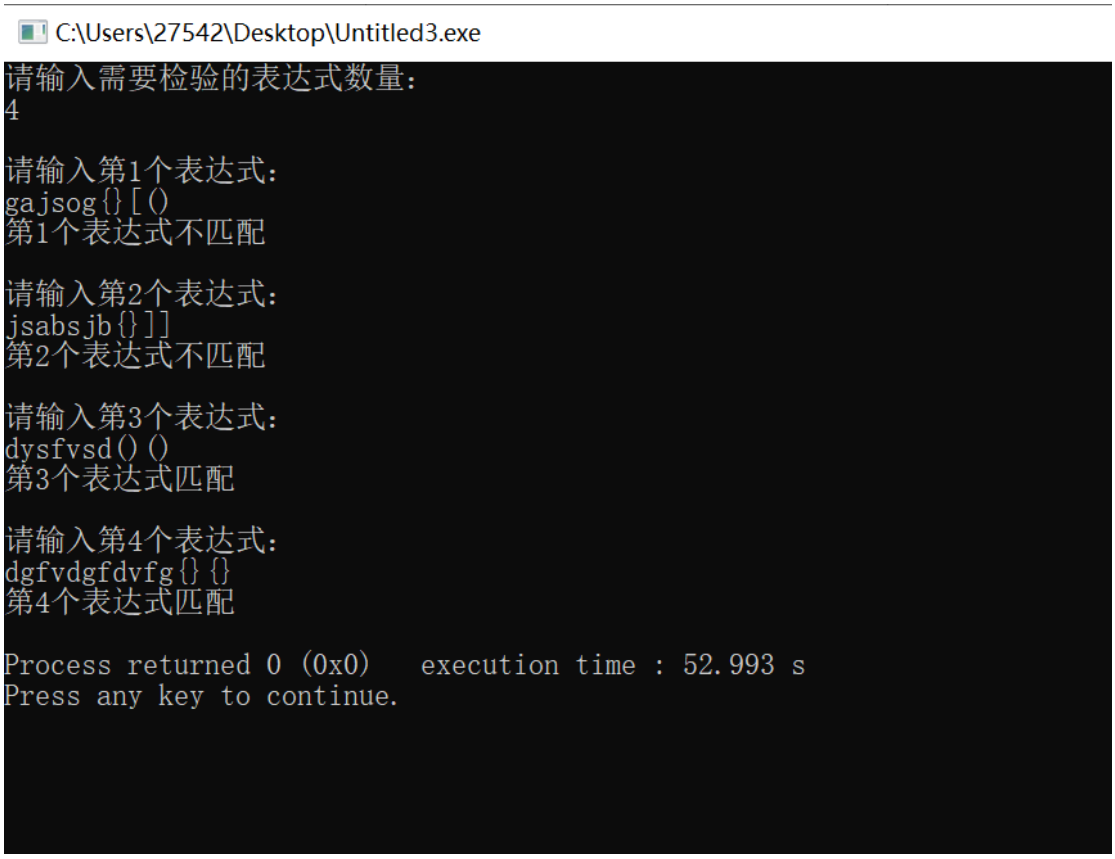
```
                else
                    printf("第%d 个表达式不匹配\n",i+1);
            }
        }

        return 0;
}
```

六、测试和结果

实验一：



实验二：



七、用户手册

1、打开程序并运行，具体步骤按文字提示进行。

2、在输入各个字符和表达式时，按回车键进入下一步操作，未按回车键时可以删除已输入的字符和表达式，按回车键后则不可以删除。