西安财经学院 信息学院

姓名：程驰

学号：1931054041

班级：计本1902

指导教师：李薇

成绩：

数据结构 实验报告

实验名称 栈的基本操作

实验日期：2020年10月15日

**一、实验目的：**

1．复习C语言程序设计中的知识。

2．掌握栈的顺序和链式存储结构的表示和实现方法。

3．掌握栈的基本操作的算法实现。

**二、实验内容：**

1. 建立顺序栈。

2. 在顺序栈上实现入栈、出栈等操作。

3. 建立链栈。

4. 在链栈上实现入栈、出栈等操作。

5. 实现十进制数的数制转换。

6. 实现算术表达式(中缀式)的求值。（选做）

三、实验要求：

1.编写实现顺序栈的基本算法（初始化、入栈、出栈、取栈顶元素、遍历、置空等）的函数，并在此基础上设计一个主程序完成如下功能：

⑴初始化顺序栈S（数据类型任选）;

⑵入栈；

⑶出栈；

⑷求栈的元素个数；

⑸取栈顶元素；

⑹遍历输出顺序栈S；

⑺置空栈。

2. 编写实现顺序栈的基本算法（初始化、入栈、出栈、取栈顶元素、遍历、置空等）的函数，并在此基础上设计一个主程序完成如下功能：

⑴初始化链栈S（数据类型任选）;

⑵入栈；

⑶出栈；

⑷求栈的元素个数；

⑸取栈顶元素；

⑹遍历输出顺序栈S；

⑺置空栈。

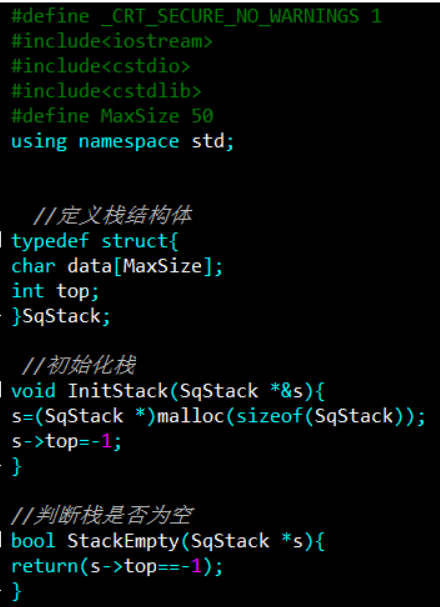
3. 用栈实现十进制数的数制转换。

4.任选一种栈的存储结构，实现算术表达式(中缀式)的求值。（选做）

四、实验步骤：

1.建立实验环境：中文Windows10、Visual Studio 2019、Dev—C++。

2.顺序栈的初始化



2.入栈、出栈、获取栈顶

//入栈

bool Push(SqStack \*&s,char e){

if(s->top==MaxSize-1)

return false;

s->top++;

s->data[s->top]=e;

return true;

}

//出栈

bool Pop(SqStack \*s){

int i;

i=s->top;

if(s->top==-1)

return false;

else{

for(;s->top!=-1;s->top--)

{

cout << s->data[s->top] << " " ;

}

s->top=i;

cout << endl;

return true;

}

}

3.求元素个数，获取栈顶元素，遍历，置空

//求栈的元素个数

int StackLength(SqStack \*s){

return (s->top+1);

}

// 获取栈顶元素，同时遍历输出顺序栈S

bool Get(SqStack \*&s){

if(s->top==-1)

return false;

else{

for(;s->top!=-1;s->top--)

{

cout << s->data[s->top] << " " ;

}

cout << endl;

return true;

}

}

//置空栈

void DestroyStack(SqStack\* &s){

free(s);

}

附程序清单：

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include<iostream>

#include<cstdio>

#include<cstdlib>

#define MaxSize 50

using namespace std;

//定义栈结构体

typedef struct{

char data[MaxSize];

int top;

}SqStack;

//初始化栈

void InitStack(SqStack \*&s){

s=(SqStack \*)malloc(sizeof(SqStack));

s->top=-1;

}

//判断栈是否为空

bool StackEmpty(SqStack \*s){

return(s->top==-1);

}

//入栈

bool Push(SqStack \*&s,char e){

if(s->top==MaxSize-1)

return false;

s->top++;

s->data[s->top]=e;

return true;

}

//求栈的元素个数

int StackLength(SqStack \*s){

return (s->top+1);

}

//出栈

bool Pop(SqStack \*s){

int i;

i=s->top;

if(s->top==-1)

return false;

else{

for(;s->top!=-1;s->top--)

{

cout << s->data[s->top] << " " ;

}

s->top=i;

cout << endl;

return true;

}

}

// 获取栈顶元素，同时遍历输出顺序栈S

bool Get(SqStack \*&s){

if(s->top==-1)

return false;

else{

for(;s->top!=-1;s->top--)

{

cout << s->data[s->top] << " " ;

}

cout << endl;

return true;

}

}

//置空栈

void DestroyStack(SqStack\* &s){

free(s);

}

int main()

{

SqStack\* s;

InitStack(s);

cout << StackEmpty(s) << endl;

Push(s,'a');

Push(s,'b');

Push(s,'c');

Push(s,'d');

Push(s,'e');

cout << StackEmpty(s) << endl;

cout << StackLength(s) << endl;

Pop(s);

Get(s);

cout << StackEmpty(s) << endl;

DestroyStack(s);

}

对链栈的操作;

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include<iostream>

#include<cstdio>

#include<cstdlib>

using namespace std;

typedef int elemtype;

typedef struct LinkedStackNode

{

elemtype data;

struct LinkedStackNode \* next;

} LinkedStackNode, \* LinkedStack;

LinkedStack top;

//初始化

LinkedStack Init\_LinkedStack()

{

LinkedStack top=(LinkedStackNode \* )malloc (sizeof( LinkedStackNode));

if(top!=NULL)//申请空间成功

top->next=NULL;//设置栈顶指针为空

return top;

}

//判栈空

int LinkedStack\_Empty(LinkedStack top)

{

if(top->next==NULL)//检查栈顶指针的值

{

return 1;//栈S为空，函数返回1

}

else

{

return 0;

}

}

//入栈

int Push\_LinkedStack(LinkedStack top,elemtype x)

//插入数据元素x为新的栈顶元素

{

LinkedStackNode \* node;

node=(LinkedStackNode \* )malloc(sizeof(LinkedStackNode));

if(node==NULL)

{

return 0;//申请结点空间失败,插入失败，函数返回0

}

else

{

node->data=x;//设置新结点的数据域

node->next=top->next;//设置新结点的指针城

top->next=node;//设置头结点指针城指向新的栈顶元素

return 1;//插入成功，函数返回1

}

}

//求栈长

int Length\_LinkedStack(LinkedStack top)

{

int count = 0;

while(top->next!=NULL)

{

++count;

top=top->next;

}

return count;

}

//出栈

int Pop\_LinkedStack(LinkedStack top, elemtype \*x)

{ LinkedStackNode \* node;

if(top->next==NULL)

{

return 0;

}

else

{

node=top->next;//将原栈顶数据元素弹出并赋给node

\*x=node->data;//将原栈顶数据元素的数据赋值给x

top->next=node->next;//top指向链栈中的下一个数据元素

free (node);//释放原栈顶数据元素所占的空间

return 1;

}

}

//取栈顶元素

int GetTop\_LinkedStack(LinkedStack top)

{

if(top->next)

{

return top->next->data;

}

return -1;

}

//主函数

Int main()

{

int i,t,x,a[20];

LinkedStack top=Init\_LinkedStack();//初始化栈

x=LinkedStack\_Empty(top);//判栈空结果赋值给X

if(x=0)

{

printf("栈为空\n");

}

printf("请依次输入5个数,开始入栈：\n");

for(i=0;i<5;i++)

{

scanf("%d",&a[i]);

Push\_LinkedStack(top,a[i]);

x=GetTop\_LinkedStack(top);

if(x!=-1)

{

printf("当前栈顶元素为%d\n",x);

}

}

printf("入栈结束\n");

printf("栈长为%d\n",Length\_LinkedStack(top));

printf("开始出栈:\n");

for (i=0;i<5;i++)

{

Pop\_LinkedStack(top,&t);

printf("%d",t);

}

printf("\n");

printf("出栈后栈长为%d\n",Length\_LinkedStack(top));

}

3. 用栈实现十进制数的数制转换

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include<iostream>

#include<cstdio>

#include<cstdlib>

#define ERROR 0

#define OK 1

#define INITSIZE 10

#define INCREMENT 2

using namespace std;

// 定义栈结构体

typedef int Status;

typedef int Elemtype;

typedef struct{

Elemtype \*base;

Elemtype \*top;

int StackSize;

}SqStack;

//初始化

Status InitStack(SqStack \*S)

{

S->base=(Elemtype\*)malloc(sizeof(Elemtype)\*INITSIZE);

if(!S->base)

{

return ERROR;

}

S->top=S->base;

S->StackSize=INITSIZE;

return OK;

}

//入栈

Status PushStack(SqStack \*S,Elemtype e)

{

if( S->top - S->base >= S->StackSize)

{

S->base=(Elemtype\*)realloc(S->base,(S->StackSize+INCREMENT)\*sizeof(Elemtype));

if(!S->base)

{

return ERROR;

}

S->top = S->base+S->StackSize;

S->StackSize+=INCREMENT;

}

\*S->top=e;

S->top++;

return OK;

}

//判断栈空

Status StackEmpty(SqStack \*S)

{

if(S->top == S->base)

{

return OK;

}

return ERROR;

}

//出栈

Status PopStack(SqStack \*S,Elemtype \*e)

{

if(S->top == S->base)

{

return ERROR;

}

\*e=\*--S->top;

return OK;

}

//进制转换 将十进制转换为二进制、八进制、十六进制

void Fun\_10\_2\_8\_16(int number,int cet)

{

SqStack S;

Elemtype e;

InitStack(&S);

if(number==0)

{

printf("转换后的%d进制数为:0",cet);

return ;

}

while(number)

{

PushStack(&S,number%cet);

number=number/cet;

}

printf("转换后的%d进制数为：",cet);

while(!StackEmpty(&S))

{

PopStack(&S,&e);

switch(e) //整型数转成字符输出

{

case 10:

case 11:

case 12:

case 13:

case 14:

case 15: e+=55;break;

default :e+=48;

}

printf("%c",e);

}

}

int main()

{

int number,cet;

printf("请输入要转换的十进制数：");

scanf("%d",&number);

printf("请输入要转换的数制：");

scanf("%d",&cet);

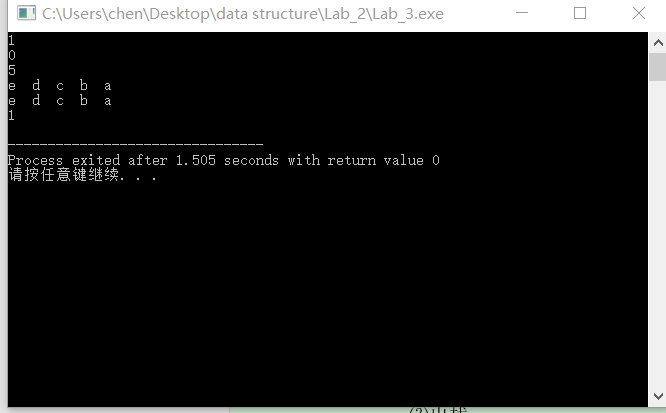
Fun\_10\_2\_8\_16(number,cet);

return 0;

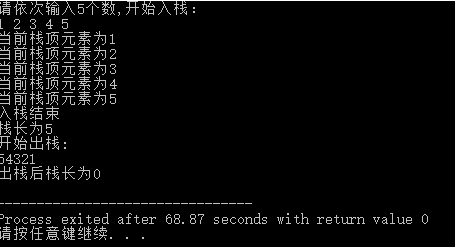
}

五、实验结果：

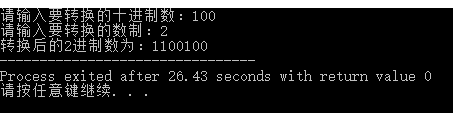
1.顺序栈操作

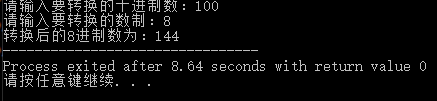


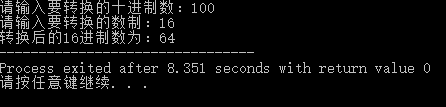
2.链栈操作



3.用栈实现十进制数的数制转换。







六、实验总结：

通过这次实验，我对栈的操作更加熟悉，对它的利用也更加熟练。