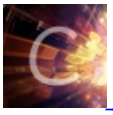
数据结构实验4、顺序栈的基本操作及应用

原创

henu_Scarlett 于 2020-11-06 23:21:22 发布 1487 收藏 25
分类专栏： 数据结构 文章标签： 数据结构 栈 c++

本文链接：https://blog.csdn.net/m0_52229815/article/details/109540728



实验四：

作者说：

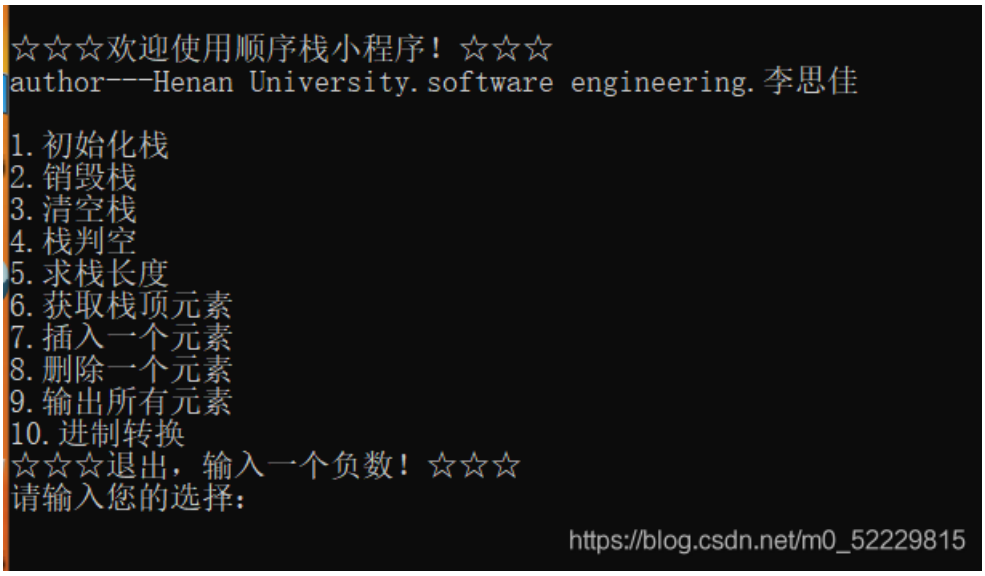
实验只包括栈的相关操作，没有用到队列，其实我觉得队列也挺复杂的。哈哈哈，刚刚看了一下，下一个实验就是关于队列的，慢慢学吧，加油！

定义：

栈和队列是两种特殊的线性表，即操作受限的线性表，只是对插入、删除运算加以限制。

栈是只允许在一端进行插入、删除运算，因而是后进先出表 LIFO，允许插入和删除运算的一端称为__ 栈顶 ___。不允许插入和删除运算的一端称为_ 栈底 ____ ；

一、运行效果截图



二、实验要求

（1）实验目的

通过该实验，让学生掌握栈的相关基本概念，认识栈是插入和删除集中在一端进行的线性结构，掌握栈的“先入后出”操作特点。栈在进行各类操作时，栈底指针固定不动，掌握栈空、栈满的判断条件。

（2）实验内容

用顺序存储结构，实现教材定义的栈的基本操作，提供数制转换功能，将输入的十进制整数转换成二进制、八进制或十六进制。

（3）参考界面

菜单中包括以下功能：

1.初始化栈，

2.销毁栈，

3.清空栈，

4.栈判空，

5.求栈长度，

6.获取栈顶元素，

7.插入一个 元素，

8.删除一个元素，

9.输出所有元素，

10.进制转换。

（4）验收/测试用例

通过菜单调用各个操作，测试点：

- 没有初始化前进行其他操作，程序是否能控制住；
- 初始化一个栈；
- 判栈空，屏幕显示栈为空；
- 3个数入栈， 2、4、6；
- 栈长度，屏幕输出3；
- 取栈顶元素，再判栈空，然后再判栈长度。让学生知道取栈顶元素不改变栈中的内容，栈顶指针不发生改变；
- 出栈，再判栈长度和输出栈中内容；（多次出栈，直到栈为空；再出栈，是否提示栈为空）
- 销毁栈，再做其他操作，判断程序是否能控制；
- 数制转换，（允许用户输入想把十进制转换成几进制），然后灵活的转换成对应的进制。

三、代码示例

```
#include<iostream>
#include<stdlib.h>
#define STACK_INIT_SIZE  100//存储空间初始分配量。
#define STACKINCREMENT 10//存储空间分配增量。
using namespace std;

typedef int elemtype;
typedef struct{
 elemtype *base;//在栈构造之前和销毁之后，base的值为NULL。
 elemtype *top;//栈顶指针。
 int stacksize;//当前已分配的存储空间，以元素为单位。
}Stack;
```

```
void Initstack(Stack &s);//1.初始化栈
void Destroystack(Stack &s);//2.销毁栈
void Clearstack(Stack &s);//3.清空栈
int StackEmpty(Stack s);//4.栈判空
int Stacklength(Stack s);//5.求栈长度
elemtype GetTop(Stack s);//6.获取栈顶元素
void Push(Stack &s,elemtype e);//7.插入一个元素
elemtype Pop(Stack &s);//8.删除一个元素
elemtype Stackvisit(Stack &s,int i);//9.输出所有元素
void conversion(Stack &jinzhi,int choose,int num);//10.进制转换

int main()
{
 Stack zhan;
 zhan.base=NULL;
 zhan.top=NULL;
 elemtype yuansu;

 int in=1;
 cout<<endl;
 cout<<"☆☆☆欢迎使用顺序栈小程序！☆☆☆"<<endl;
 cout<<"author---Henan University.software engineering.李思佳"<<endl<<endl;
 while(in==1)
 {
 cout<<"1.初始化栈"<<endl;
 cout<<"2.销毁栈"<<endl;
 cout<<"3.清空栈"<<endl;
 cout<<"4.栈判空"<<endl;
 cout<<"5.求栈长度"<<endl;
 cout<<"6.获取栈顶元素"<<endl;
 cout<<"7.插入一个元素"<<endl;
 cout<<"8.删除一个元素"<<endl;
 cout<<"9.输出所有元素"<<endl;
 cout<<"10.进制转换"<<endl;
 cout<<"☆☆☆退出，输入一个负数！☆☆☆"<<endl;
 int select;
 cout<<"请输入您的选择：";
 cin>>select;
 switch(select)
 {
  case 1://1.初始化栈.
   system("cls");
   Initstack(zhan);
   if(!zhan.base)
   {
    cout<<"存储空间分配失败，请重新操作！"<<endl;
   }
   else
   {
    cout<<"您已经成功初始化一个栈!"<<endl;
   }
   cout<<endl;
   break;
  case 2://2.销毁栈
   system("cls");
   if(zhan.base==NULL)
   {
    cout<<"还未初始化栈，请您先初始化！"<<endl;
```

```
  cout<<"还未初始化栈，请您先初始化！"<<endl;
 }
 else
 {
  Destroystack(zhan);
  cout<<"您已成功销毁栈！"<<endl;
 }
 cout<<endl;
 break;
case 3://3.清空栈
 system("cls");
 if(zhan.base==NULL)
 {
  cout<<"还未初始化栈，请您先初始化！"<<endl;
 }
 else
 {
  if(zhan.base==zhan.top)
  {
   cout<<"栈为空，不用清空！"<<endl;
  }
  else
  {
   Clearstack(zhan);
   cout<<"已经成功清空栈！"<<endl;
  }
 }
 cout<<endl;
 break;
case 4://4.栈判空
 system("cls");
 if(zhan.base==NULL)
 {
  cout<<"还未初始化栈，请您先初始化！"<<endl;
 }
 else
 {
  if(StackEmpty(zhan)==0)
  {
   cout<<"栈为空！"<<endl;
  }
  else
  {
   cout<<"栈不为空！"<<endl;
  }
 }
 cout<<endl;
 break;
case 5://5.求栈长度
 system("cls");
 if(zhan.base==NULL)
 {
  cout<<"还未初始化栈，请您先初始化！"<<endl;
 }
 else
 {
  cout<<"栈的长度为："<<Stacklength(zhan)<<endl;
 }
 cout<<endl;
 break;
```

```
case 6://6.获取栈顶元素
 system("cls");
 if(zhan.base==NULL)
 {
  cout<<"还未初始化栈，请您先初始化！"<<endl;
 }
 else
 {
  if(zhan.base==zhan.top)
  {
   cout<<"这是一个空栈，不存在元素！"<<endl;
  }
  else
  {
   cout<<"栈顶元素为："<<GetTop(zhan)<<endl;
  }
 }
 cout<<endl;
 break;
case 7://7.插入一个元素
 system("cls");
 if(zhan.base==NULL)
 {
  cout<<"还未初始化栈，请您先初始化！"<<endl;
 }
 else
 {
  cout<<"请输入您要插入的元素：";
  cin>>yuansu;
  if(zhan.top-zhan.base>=zhan.stacksize)//栈满，追加存储空间。
  {
   zhan.base=(elemtype *)realloc(zhan.base,(zhan.stacksize+STACKINCREMENT)*sizeof(elemtype));
   if(!zhan.base){
    cout<<"存储空间分配失败！请重新操作。"<<endl;
   }
   else
   {
    zhan.top=zhan.base+zhan.stacksize;
    zhan.stacksize+=STACKINCREMENT;
    cout<<"内存空间分配成功！"<<endl;
   }
  }
  Push(zhan,yuansu);
  cout<<"入栈成功！"<<endl;
 }
 cout<<endl;
 break;
case 8://8.删除一个元素
 system("cls");
 if(zhan.base==NULL)
 {
  cout<<"还未初始化栈，请您先初始化！"<<endl;
 }
 else
 {
  if(zhan.base==zhan.top)
  {
   cout<<"栈为空，没有元素可以出栈！"<<endl;
  }
  else
```

```
   else
   {
    cout<<"元素"<<Pop(zhan)<<"出栈成功！"<<endl;
   }
  }
  cout<<endl;
  break;
 case 9://9.输出所有元素
  system("cls");
  if(zhan.base==NULL)
  {
   cout<<"还未初始化栈，请您先初始化！"<<endl;
  }
  else
  {
   if(zhan.base==zhan.top)
   {
    cout<<"栈为空，没有元素可以输出！"<<endl;
   }
   else
   {
   cout<<"栈中的元素（从栈底到栈顶）为：";
   for(int i=zhan.top-zhan.base;i>0;i--)
   {
    cout<<Stackvisit(zhan,i)<<" ";
   }

   }
  }
  cout<<endl<<endl;
  break;
 case 10://10.进制转换
  system("cls");
  int  num;
  cout<<"请您输入一个十进制整数：";
  cin>>num;
  if(num<=0)
  {
   cout<<"您输入的数据不合法！"<<endl;
  }
  else
  {
   int choose;
   Stack jinzhi;//构造进制转换的栈。
   jinzhi.base=NULL;
   jinzhi.top=NULL;
   cout<<endl;
   cout<<"1.转化成二进制"<<endl;
   cout<<"2.转化成八进制"<<endl;
   cout<<"3.转化成十六进制"<<endl<<endl;
   cout<<"请输入您的选择：";
   cin>>choose;
   switch(choose)
   {
    case 1:
     conversion(jinzhi,choose,num);
     cout<<endl;
     cout<<"转化结果为：";
     while(StackEmpty(jinzhi)==1)
     {
```

```
     cout<<Pop(jinzhi);
    }
    cout<<endl;
    break;
   case 2:
    conversion(jinzhi,choose,num);
    cout<<"转化结果为：";
    while(StackEmpty(jinzhi)==1)
    {
     cout<<Pop(jinzhi);
    }
    cout<<endl;
    break;
   case 3:
    conversion(jinzhi,choose,num);
    cout<<"转化结果为：";
    while(StackEmpty(jinzhi)==1)
    {
     num=Pop(jinzhi);
     if(num>=10)
     {
      switch(num)
      {
       case 10:
        cout<<"A";
        break;
       case 11:
        cout<<"B";
        break;
       case 12:
        cout<<"C";
        break;
       case 13:
        cout<<"D";
        break;
       case 14:
        cout<<"E";
        break;
       case 15:
        cout<<"F";
        break;
      }
     }
     else
     {
      cout<<num;
     }
    }
    cout<<endl;
    break;

   default:
    cout<<"您输入的选择不正确哦！"<<endl;
    break;

  }
 }
 cout<<endl;
 break;
```

```
  default:
   system("cls");
   if(select<0)//退出程序
   {
    in=-999;
    cout<<"☆☆☆您已经退出程序，欢迎下次使用！☆☆☆"<<endl;
    break;
   }
   else
   {
    cout<<"您的选择有误，请重新输入！"<<endl<<endl;
   }
   break;
 }
 }

}


//1.初始化栈
void Initstack(Stack &s)
{
 s.base=(elemtype *)malloc(STACK_INIT_SIZE *sizeof(elemtype));
 s.top=s.base;
 s.stacksize=STACK_INIT_SIZE;
}

//2.销毁栈
void Destroystack(Stack &s)
{
 s.stacksize=0;
 s.base=NULL;
 s.top=NULL;
 free(s.base);
 free(s.top);
}

//3.清空栈
void Clearstack(Stack &s)
{
 s.stacksize=0;
 s.base=s.top;
}

//4.栈判空
int StackEmpty(Stack s)
{
 if(s.base==s.top)
 {
  return 0;//栈为空。
 }
 else
 {
  return 1;//栈不为空。
 }
}

//5.求栈长度
int Stacklength(Stack s)
```

```
{
 elemtype *p=s.top;
 int i=0;//计数。
 while(p!=s.base)
 {
  p--;
  i++;
 }
 return i;
}

//6.获取栈顶元素
elemtype GetTop(Stack s)
{
 return *(s.top-1);
}

//7.插入一个元素
void Push(Stack &s,elemtype e)
{
 *s.top=e;
 s.top++;
}

//8.删除一个元素
elemtype Pop(Stack &s)
{
 s.top--;
 return *s.top;
}

//9.输出所有元素
elemtype Stackvisit(Stack &s,int i)
{
 elemtype *p=s.top;
 for(int j=1;j<=i;j++){
  p--;
 }
 return *p;
}

//10.进制转换
void conversion(Stack &jinzhi,int choose,int num)
{
 Initstack(jinzhi);
 int njinzhi;
 switch(choose)
 {
  case 1:
   njinzhi=2;
   break;
  case 2:
   njinzhi=8;
   break;
  case 3:
   njinzhi=16;
   break;
 }
 while(num)
```

```
 {
     Push(jinzhi,num%njinzhi);
        num=num/njinzhi;
     //n=(n div d)*d + n mod d
        //十进制n和其他进制d的转换机制。div为整除运算，mod为求余运算。
 }
}
```