顺序栈与链栈实现与测试

1. 顺序栈的实现

SeqStack.h

#ifndef SEQSTACK\_H

#define SEQSTACK\_H

template **<**class T**,**int MaxSize**>**

class SeqStack

**{**

private**:**

T data**[**MaxSize**];** //存放栈元素数组

int top**;** //栈顶指针，指示栈元素在数组下标

public**:**

SeqStack**();** //构造函数

void Push**(**T x**);** //入栈

T Pop**();** //出栈

T Top**();** //取出栈顶元素（元素不出栈）

bool isEmpty**();** //判断栈是否为空

**~**SeqStack**();** //析构函数

**};**

#endif // SEQSTACK\_H

SeqStack.cpp

#include "SeqStack.h"

#include <iostream>

#include <stdlib.h>

**using** **namespace** std**;**

//顺序栈初始化

template **<**class T**,**int MaxSize**>**

SeqStack**<**T**,**MaxSize**>::**SeqStack**()**

**{**

top **=** **-**1**;**

**}**

//入栈操作

template **<**class T**,**int MaxSize**>**

void SeqStack**<**T**,**MaxSize**>::**Push**(**T x**)**

**{**

**if(**top**==**MaxSize**-**1**){**

cerr **<<** "Stack is full!"**;**

exit**(**1**);**

**}**

top**++;**

data**[**top**]** **=** x**;**

**}**

//出栈操作

template **<**class T**,**int MaxSize**>**

T SeqStack**<**T**,**MaxSize**>::**Pop**()**

**{**

**if(**top**==-**1**){**

cerr **<<** "Stack is empty!"**;**

exit**(**1**);**

**}**

T x **=** data**[**top**];**

top**--;**

**return** x**;**

**}**

//取栈顶元素

template **<**class T**,**int MaxSize**>**

T SeqStack**<**T**,**MaxSize**>::**Top**()**

**{**

**if(**top**==-**1**){**

cerr **<<** "Stack is empty!"**;**

exit**(**1**);**

**}**

**return** data**[**top**];**

**}**

//判断栈是否为空

template **<**class T**,**int MaxSize**>**

bool SeqStack**<**T**,**MaxSize**>::**isEmpty**()**

**{**

**return** **(**top**==-**1**);**

**}**

//析构函数

template **<**class T**,**int MaxSize**>**

SeqStack**<**T**,**MaxSize**>::~**SeqStack**()**

**{**

**}**

测试：main.cpp

#include <stdio.h>

#include "SeqStack.cpp"

int main**()**

**{**

int a**[**10**];**

SeqStack**<**int**,**20**>** S1**;**

//1~10压栈

**for(**int i**=**0**;** i**<**10**;** i**++)**

**{**

a**[**i**]** **=** i**;**

S1**.**Push**(**a**[**i**]);**

**}**

//9~5出栈

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

**{**

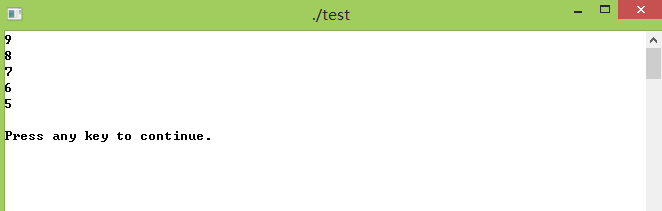
cout**<<**S1**.**Pop**()<<**endl**;**

**}**

**return** 0**;**

**}**

测试截图



1. 链栈的实现

LinkStack.h

#ifndef LINKSTACK\_H

#define LINKSTACK\_H

template**<**class T**>**

struct Node

**{**

T data**;**

Node**<**T**>\*** next**;**

**};**

template**<**class T**>**

class LinkStack

**{**

private**:**

Node**<**T**>\*** top**;** //栈顶指针

public**:**

LinkStack**();** //构造函数

**~**LinkStack**();** //析构函数

void Push**(**T x**);** //入栈

T Pop**();** //出栈

T Top**();** //取出栈顶元素（元素不出栈）

bool isEmpty**();** //判断链栈是否为空

**};**

#endif // LINKSTACK\_H

LinkStack.cpp

#include "LinkStack.h"

#include <iostream>

#include <stdlib.h>

**using** **namespace** std**;**

//链栈初始化

template**<**class T**>**

LinkStack**<**T**>::**LinkStack**()**

**{**

top **=** **NULL;**

**}**

//入栈操作

template**<**class T**>**

void LinkStack**<**T**>::**Push**(**T x**)**

**{**

Node**<**T**>\*** s **=** **new** Node**<**T**>;** //申请一个数据域为x的结点s

s**->**data **=** x**;**

s**->**next **=** top**;**

top **=** s**;** //将结点s插入在栈顶

**}**

//出栈操作

template**<**class T**>**

T LinkStack**<**T**>::**Pop**()**

**{**

**if(**top**==NULL){**

cerr **<<** "Stack is empty!"**;**

exit**(**1**);**

**}**

T x **=** top**->**data**;** //x储存栈顶元素

Node**<**T**>\*** p **=** top**;** //p指向top

top **=** top**->**next**;** //栈顶指针后移

**delete** p**;** //删除栈顶结点

**return** x**;**

**}**

//取出栈顶元素

template**<**class T**>**

T LinkStack**<**T**>::**Top**()**

**{**

**if(**top**==NULL){**

cerr **<<** "Stack is empty!"**;**

exit**(**1**);**

**}**

**return** top**->**data**;**

**}**

//判断链栈是否为空

template**<**class T**>**

bool LinkStack**<**T**>::**isEmpty**()**

**{**

**return** **(**top**==NULL);**

**}**

//析构函数

template**<**class T**>**

LinkStack**<**T**>::~**LinkStack**()**

**{**

Node**<**T**>\*** p **=** top**;**

**while(**p**)**

**{**

Node**<**T**>\*** q **=** p**;**

p **=** p**->**next**;**

**delete** q**;**

**}**

top **=** **NULL;**

**}**

测试：main.cpp

#include <stdio.h>

#include "LinkStack.cpp"

int main**()**

**{**

int a**[**10**];**

LinkStack**<**int**>** L1**;**

//1~10压栈

**for(**int i**=**0**;** i**<**10**;** i**++)**

**{**

a**[**i**]** **=** i**;**

L1**.**Push**(**a**[**i**]);**

**}**

//9~5出栈

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

**{**

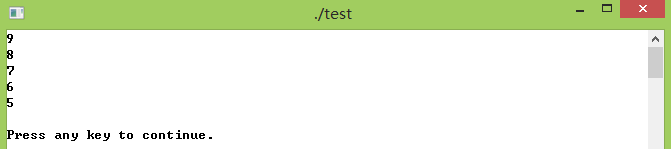
cout**<<**L1**.**Pop**()<<**endl**;**

**}**

**return** 0**;**

**}**

测试截图



1. 栈的共享

BothStack.h

#ifndef BOTHSTACK\_H

#define BOTHSTACK\_H

template **<**class T**,** int MaxSize**>**

class BothStack

**{**

private**:**

T data**[**MaxSize**];**

int top1**,** top2**;**

public**:**

BothStack**();**

**~**BothStack**();**

void Push**(**int i**,**T x**);**

T Pop**(**int i**);**

T GetTop**(**int i**);**

bool isEmpty**(**int i**);**

**};**

#endif // BOTHSTACK\_H

BothStack.cpp

#include "BothStack.h"

#include <iostream>

#include <stdlib.h>

**using** **namespace** std**;**

//构造函数

template **<**class T**,** int MaxSize**>**

BothStack**<**T**,**MaxSize**>::**BothStack**()**

**{**

top1 **=** **-**1**;**

top2 **=** MaxSize**;**

**}**

//析构函数

template **<**class T**,** int MaxSize**>**

BothStack**<**T**,**MaxSize**>::~**BothStack**()**

**{**

**}**

//入栈操作

template **<**class T**,** int MaxSize**>**

void BothStack**<**T**,**MaxSize**>::**Push**(**int i**,**T x**)**

**{**

**if(**i**==**1**){** //向Stack1 push

**if(**top1**+**1**==**top2**){**

cerr **<<** "Stack1 is full!"**;**

exit**(**1**);**

**}**

data**[++**top1**]** **=** x**;**

**}**

**else** **if(**i**==**2**){** //向Stack2 push

**if(**top1**+**1**==**top2**){**

cerr **<<** "Stack2 is full!"**;**

exit**(**1**);**

**}**

data**[--**top2**]** **=** x**;**

**}** **else** **{**

cerr **<<** "Error input!"**;**

exit**(**1**);**

**}**

**}**

//出栈操作

template **<**class T**,** int MaxSize**>**

T BothStack**<**T**,**MaxSize**>::**Pop**(**int i**)**

**{**

**if(**i**==**1**){**

**if(**top1**==-**1**)** **{**

cerr **<<** "Stack1 is empty!"**;**

exit**(**1**);**

**}**

T x **=** data**[**top1**--];**

**return** x**;**

**}**

**else** **if(**i**==**2**)** **{**

**if(**top2**==**MaxSize**){**

cerr **<<** "Stack2 is empty!"**;**

exit**(**1**);**

**}**

T x **=** data**[**top2**++];**

**return** x**;**

**}** **else** **{**

cerr **<<** "Error input!"**;**

exit**(**1**);**

**}**

**}**

//取栈顶元素

template **<**class T**,** int MaxSize**>**

T BothStack**<**T**,**MaxSize**>::**GetTop**(**int i**)**

**{**

**if(**i**==**1**)**

**{**

**if(**top1**==-**1**){**

cerr **<<** "Stack1 is empty!"**;**

exit**(**1**);**

**}**

**return** data**[**top1**];**

**}**

**else** **if(**i**==**2**)**

**{**

**if(**top2**==**MaxSize**){**

cerr **<<** "Stack2 is empty!"**;**

exit**(**1**);**

**}**

**return** data**[**top2**];**

**}**

**else**

**{**

cerr **<<** "Error input!"**;**

exit**(**1**);**

**}**

**}**

//判断两栈是否为空

template **<**class T**,** int MaxSize**>**

bool BothStack**<**T**,**MaxSize**>::**isEmpty**(**int i**)**

**{**

**if(**i**==**1**)**

**return** **(**top1**==-**1**);**

**else** **if(**i**==**2**)**

**return** **(**top2**==**MaxSize**);**

**}**

测试：main.cpp

#include <stdio.h>

#include "BothStack.cpp"

int main**()**

**{**

int a**[**10**],**b**[**10**];**

BothStack**<**int**,**30**>** B**;**

//0~18压栈1

**for(**int k**=**0**;** k**<**10**;** k**++)**

**{**

a**[**k**]** **=** k**;**

B**.**Push**(**1**,**a**[**k**]);**

**}**

//1~19压栈2

**for(**int k**=**0**;** k**<**10**;** k**++)**

**{**

b**[**k**]** **=** 2**\***k**;**

B**.**Push**(**2**,**b**[**k**]);**

**}**

//9~5出栈1

cout **<<** "Pop 5 from Stack2:" **<<** endl**;**

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

**{**

cout**<<**B**.**Pop**(**1**)<<**endl**;**

**}**

//18~14出栈1

cout **<<** "Pop 3 from Stack2:" **<<** endl**;**

**for(**int i**=**0**;** i**<**3**;** i**++)**

**{**

cout**<<**B**.**Pop**(**2**)<<**endl**;**

**}**

cout**<<**"Get top:"**<<**endl**;**

cout**<<**B**.**GetTop**(**1**)<<**endl**;**

cout**<<**B**.**GetTop**(**2**)<<**endl**;**

**return** 0**;**

**}**

测试截图

