**第三次实验**

**要求：**

利用两种不同的数据结构实现双端栈，即一个存储空间存放两个不同的栈（可参考P312-12,13）

1.利用数组实现双端栈（两个栈相向生长）

2.利用双端队列deque实现双端栈（两个栈反向生长）

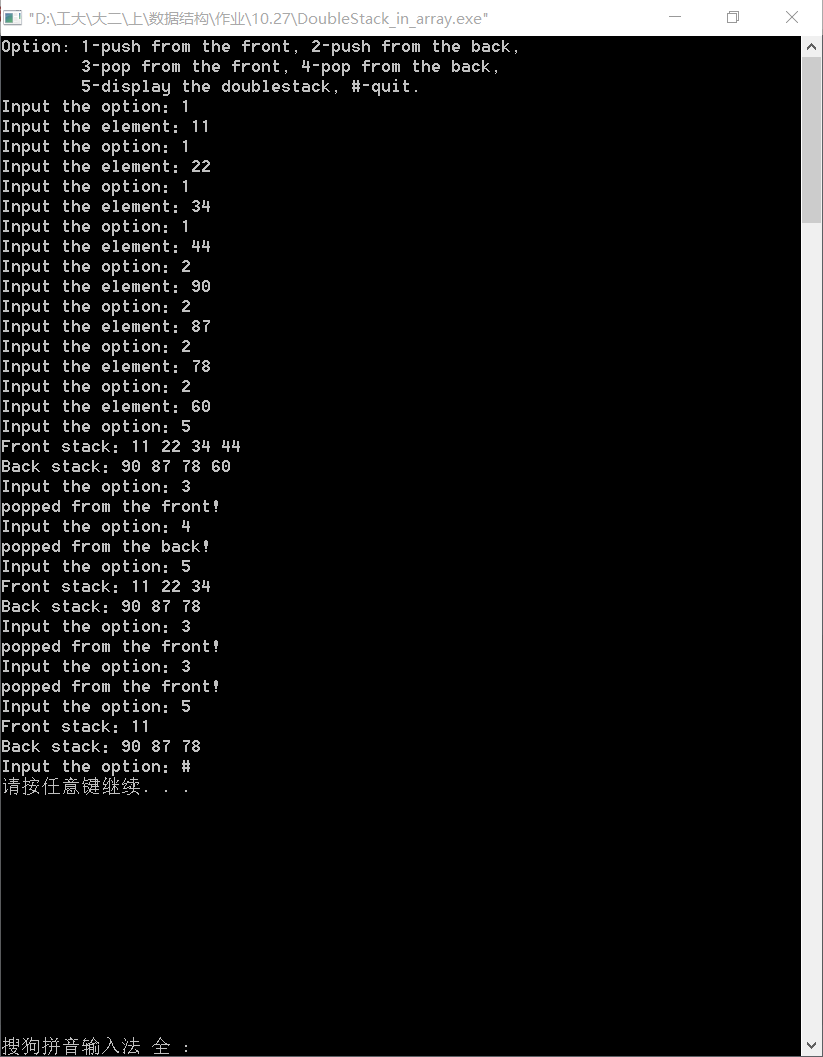
分析：

要求1是数组，开一个固定大小的数组，将数组的头部与尾部视为两个栈的空的栈顶，选择在前栈或是后栈输入，前栈栈顶向后移动，后栈栈顶向前移动，两个栈相向生长。若两个栈栈顶相遇，则无法继续输入。

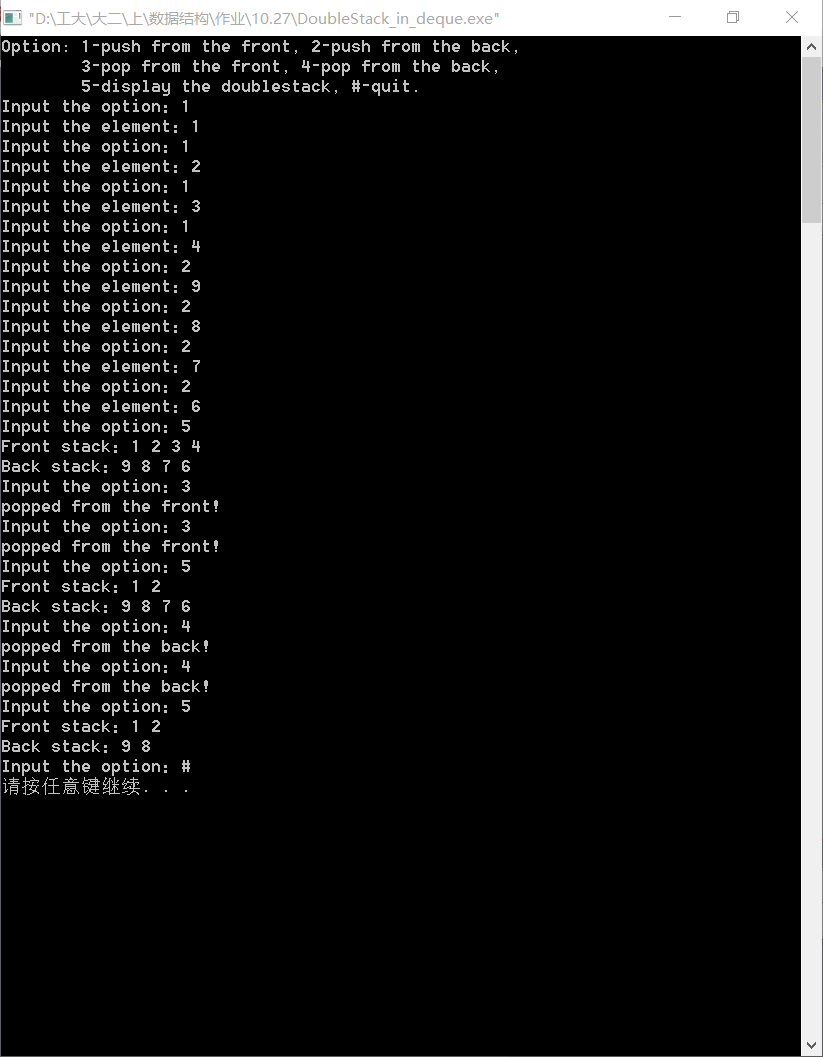
要求2是基于队列的双端栈，由于队列本身即允许在头尾两端输入输出，所以直接调用STL中的deque中的成员函数即可实现压栈与出栈的功能。但在deque中没有全部输出的成员函数，需要自己编写。

**结果截图：**

要求1：



要求2：



**源代码：**

要求1:数组形式

//Double\_direction\_Stack\_in\_array project\_Experiment3.

//Created by Yule, Nov\_1\_2015

//==============================================================================

#include<iostream>

using namespace std;

const int SIZE=100;

// Increate the structure of stack in array ways.

class DoubleStack

{

public:

DoubleStack() {frontTop=-1; backTop=SIZE;}

bool full() const;

void frontPush (int item); // push from the front

void backPush (int item); // push form the back

void frontPop(); // pop from the front

void backPop(); // pop from the back

void display();

private:

int StackArray[SIZE];

int frontTop;

int backTop;

};

// Judge wether the array is full or not.

bool DoubleStack::full() const

{

if (frontTop >= backTop-1){

cout<<"Stack full!\n";

return true;

}

return false;

}

// push from the front

void DoubleStack::frontPush(int item)

{

if (!full()){

frontTop += 1;

StackArray[frontTop] = item;

}

}

// push from the back

void DoubleStack::backPush(int item)

{

if (!full()){

backTop -= 1;

StackArray[backTop] = item;

}

}

// pop from the front

void DoubleStack::frontPop()

{

if (frontTop>-1){

frontTop -= 1;

cout<<"popped from the front!\n";

}

else

cout<<"Front Stack empty!\n";

}

// pop from the back

void DoubleStack::backPop()

{

if (backTop<SIZE){

backTop += 1;

cout<<"popped from the back!\n";

}

else

cout<<"Back Stack empty!\n";

}

// display all the elements in the doublestack

void DoubleStack::display()

{

// output the front stack

cout<<"Front stack：";

int i=0;

while (i <= frontTop){

cout<<StackArray[i]<<" ";

i++;

}

cout<<endl;

// output the back stack

cout<<"Back stack：";

i = SIZE-1;

while (i >= backTop){

cout<<StackArray[i]<<" ";

i--;

}

cout<<endl;

}

int choose(DoubleStack s)

{

int temp;

char c;

cout<<"Option: 1-push from the front, 2-push from the back,\n"

<<" 3-pop from the front, 4-pop from the back,\n"

<<" 5-display the doublestack, #-quit.\n";

while(1){

cout<<"Input the option：";

cin>>c;

switch(c)

{

// push from the front

case'1':cout<<"Input the element：";

cin>>temp;

s.frontPush(temp);

break;

// push from the back

case'2':cout<<"Input the element：";

cin>>temp;

s.backPush(temp);

break;

// pop from the front

case'3':s.frontPop();

break;

// pop from the back

case'4':s.backPop();

break;

// display all the elements

case'5':s.display();

break;

case'#':return 0;

default:cout<<"Error input! Please try again."<<endl;

}

}

}

int main()

{

int choose(DoubleStack );

DoubleStack S;

choose(S);

return 0;

}

要求2：队列形式

//Double\_direction\_Stack\_in\_deque project\_Experiment3.

//Created by Yule, Nov\_1\_2015

//==============================================================================

#include<iostream>

#include<deque>

using namespace std;

int choose(deque<int> de)

{

void display(deque<int>, int);

int temp;

char c;

int cal\_num = 0; // To calculate the size of the front stack.

cout<<"Option: 1-push from the front, 2-push from the back,\n"

<<" 3-pop from the front, 4-pop from the back,\n"

<<" 5-display the doublestack, #-quit.\n";

while(1){

cout<<"Input the option：";

cin>>c;

switch(c)

{

// push from the front

case'1':cout<<"Input the element：";

cin>>temp;

de.push\_front(temp);

cal\_num++;

break;

// push from the back

case'2':cout<<"Input the element：";

cin>>temp;

de.push\_back(temp);

break;

// pop from the front

case'3':if (cal\_num > 0){

cal\_num--;

de.pop\_front();

cout<<"popped from the front!\n";

}

// if there is no data, don't change cal\_num and de.

else

cout<<"Front Stack empty!\n";

break;

// pop from the back

case'4':if (de.size()-cal\_num > 0){

de.pop\_back();

cout<<"popped from the back!\n";

}

// if there is no data, don't change de.

else

cout<<"Back Stack empty!\n";

break;

// display all the elements in the doublestack

case'5':display(de, cal\_num);

break;

case'#':return 0;

default:cout<<"Error input! Please try again."<<endl;

}

}

}

// Display the doublestack.

void display(deque<int> de, int num)

{

// display the front stack.

int i = num;

cout<<"Front stack：";

while (i--)

cout<<de[i]<<' ';

cout<<endl;

// display the back stack.

// The size of the back stack is reckoned from num.

int j = num;

cout<<"Back stack：";

while (j < de.size()){

cout<<de[j]<<' ';

j++;

}

cout<<endl;

}

int main()

{

int choose(deque<int>);

deque<int> S;

choose(S);

return 0;

}