11.1

#include<iostream>

using namespace std;

template <class T1,class T2>

void Sort(T1 &a, T2 size)

{

int temp;

for (int i = 0; i < size - 1; i++)

{

for (int j = 0; j < size - 1 - i; j++)

{

if (a[j] > a[j + 1])

{

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

}

}

}

for (int i = 0; i < size; i++)

cout << a[i] << " ";

}

int main()

{

int a[5] = { 5,6,3,8,4 };

double b[5] = { 1.2,3,5.6,7,9 };

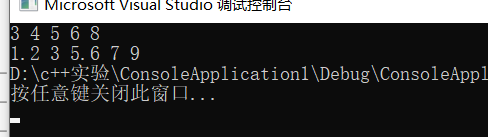
Sort(a, 5);

cout << endl;

Sort(b, 5);

return 0;

}



11.2

#include<iostream>

using namespace std;

template<class T1,class T2>

class Test

{

public:

void SetData1(T1 val) { data1=val; }

void SetData2(T2 val) {data2=val;}

T1 GetData1() { return data1; }

T2 GetData2() { return data2; }

private:

T1 data1;

T2 data2;

};

int main()

{

Test<int,double>t1;

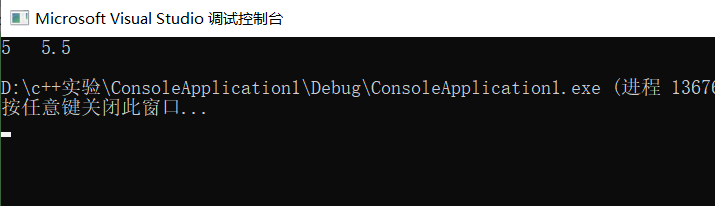
t1.SetData1(5);

t1.SetData2(5.5);

cout<<t1.GetData1()<<" "<<t1.GetData2()<<endl;

return 0;

}



11.3

#include<iostream>

using namespace std;

const int SIZE = 100; // 栈中能保存的最多元素个数

template <class T>

class IStack

{

public:

IStack() :stack{ 0 }, top(0){} // 栈的构造函数

void Push(T n); // 往栈顶增加元素

T Pop(); // 从非空栈的栈顶删除一个元素

T GetTop(); // 返回非空栈的栈顶元素

bool Empty(); // 判断栈是否为空

int Size(); // 返回栈中元素的个数

void ClearStack(); // 将栈清空

~IStack() {}; // 栈的析构函数

private:

T stack[SIZE]; // 保存栈中各元素的数组

int top; // 保存栈顶的当前位置

};

template <class T>

void IStack<T>::Push(T n)

{

if (top < SIZE)

stack[top++] = n;

else

{

cout << "错误" << endl;

return ;

}

}

template <class T>

T IStack<T>::Pop()

{

if (top >= 0)

{

top -= 1;

stack[top]=0;

}

else

{

cout << "错误" << endl;

return 0;

}

}

template <class T>

T IStack<T>::GetTop()

{

if (top >= 0)

return stack[top-1];

else

{

cout << "错误" << endl;

return 0;

}

}

template <class T>

int IStack<T>::Size()

{

return top;

}

template <class T>

bool IStack<T>::Empty()

{

if (top == 0)

return true;

else

return false;

}

template <class T>

void IStack<T>::ClearStack()

{

while (top > 0)

Pop();

}

int main()

{

IStack<int> s;

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

{

s.Push(i);

}

cout << "栈顶元素为" << s.GetTop() << endl;

cout << "栈长度为" << s.Size() << endl;

cout << "栈元素减一" << endl;

s.Pop();

cout << "栈顶元素为" << s.GetTop() << endl;

cout << "栈长度为" << s.Size() << endl<<endl;

s.ClearStack();

IStack<char> t;

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

{

t.Push('a'+i);

}

cout << "栈顶元素为" << (char)t.GetTop() << endl;

cout << "栈长度为" << t.Size() << endl;

cout << "栈元素减一" << endl;

t.Pop();

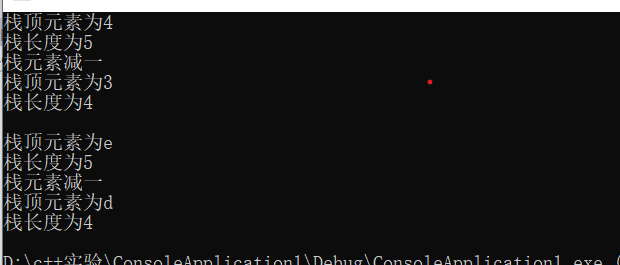
cout << "栈顶元素为" <<(char)t.GetTop() << endl;

cout << "栈长度为" << t.Size() << endl;

t.ClearStack();

return 0;

}



13.1

#include<iostream>

using namespace std;

class Complex

{

double real;

double imag;

public:

Complex() { real = 0; imag = 0; }

friend ostream &operator<<(ostream &out, Complex &c);

friend istream &operator>>(istream &in, Complex &c);

};

ostream &operator<<(ostream &out, Complex &c)

{

out << "输出复数" << endl;

out << c.real << "+" << c.imag << "i" << endl;

return out;

}

istream &operator>>(istream &in, Complex &c)

{

cout << "输入复数的实部和虚部:";

in >> c.real;

in >> c.imag;

return in;

}

int main()

{

Complex c1, c2;

cin >> c1;

cout << c1;

cout << endl;

cin >> c2;

cout << c2;

return 0;

}

