**上机报告**

1. **测试代码：**

**生成一个C++控制台应用程序，Triplet**

**头文件：**

1、def.h内容如下

//函数结果状态代码

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASIBLE -1

#define OVERFLOW -2

//Status是函数类型，其值是函数结果状态代码

typedef int Status;

2、Triplet.h内容如下

#include "def.h"

typedef int ElemType;//额外添加类型名定义

//采用动态分配的顺序存储结构

typedef ElemType \*Triplet;//由InitTriplet分配3个元素存储空间

//基本操作的函数原型说明

Status InitTriplet(Triplet &T, ElemType v1, ElemType v2, ElemType v3);

//操作结果：构造了三元组T，元素e1,e2,和e3分别被赋以参数v1,v2和v3的值。

Status DestroyTriplet(Triplet&T);

//操作结果：三元组T被销毁

Status Get(Triplet T, int i, ElemType&e);

//初始条件：三元组T已经存在，1<=i<=3

//操作结果：用e返回T的第i个元的值

Status Put(Triplet &T, int i, ElemType e);

//初始条件：三元组T已经存在，1<=i<=3

//操作结果：改变T的第i个元的值为e

Status IsAscending(Triplet T);

//初始条件：三元组T已经存在

//操作结果：如果T的三个元素按升序排列，则返回1，否则返回0

Status IsDescending(Triplet T);

//初始条件：三元组T已经存在

//操作结果：如果T的三个元素按降序排列，则返回1，否则返回0

Status Max(Triplet T, ElemType &e);

//初始条件：三元组T已经存在

//操作结果：用e返回T的3个元素中的最大值

Status Min(Triplet T, ElemType &e);

//初始条件：三元组T已经存在

//操作结果：用e返回T的3个元素中的最小值

**3.基本操作的实现，文件名为Triplet.cpp**

#include"triplet.h"

#include<iostream>

using namespace std;

Status InitTriplet(Triplet &T, ElemType v1, ElemType v2, ElemType v3)

{

//构造了三元组T，依次置T的3个元素的初值为v1,v2和v3。

T = (ElemType\*)malloc(3 \* sizeof(ElemType));//分配3个元素的存储空间

if (!T)exit(OVERFLOW);//分配存储空间失败

T[0] = v1; T[1] = v2; T[2] = v3;

return OK;

}//InitTriplet

Status DestroyTriplet(Triplet &T)

{

//销毁三元组T

free(T);

T = NULL;

return OK;

}//DestroyTriplet

Status Get(Triplet T, int i, ElemType&e)

{

//1<=i<=3,用e返回T的第i个元的值

if (i<1 || i>3)return ERROR;

e = T[i - 1];

return OK;

}//Get

Status Put(Triplet &T, int i, ElemType e)

{

//1<=i<=3,置T的第i个元的值为e

if (i<1 || i>3)return ERROR;

T[i - 1]=e;

return OK;

}//Put

Status IsAscending(Triplet T)

{

//如果T的三个元素按升序排列，则返回1，否则返回0

return (T[0] <= T[1]) && (T[1] <= T[2]);

}//IsAscending

Status IsDescending(Triplet T)

{

//如果T的三个元素按降序排列，则返回1，否则返回0

return (T[0] >= T[1]) && (T[1] >= T[2]);

}//IsDescending

Status Max(Triplet T, ElemType &e)

{

//用e返回T的3个元素中的最大值

e = T[0];

if (T[1] > e)e = T[1];

if (T[2] > e)e = T[2];

return OK;

}//Max

Status Min(Triplet T, ElemType &e)

{

//用e返回T的3个元素中的最小值

e = T[0];

if (T[1] < e)e = T[1];

if (T[2] < e)e = T[2];

return OK;

}//Min

**4.test.cpp**

#include"triplet.h"

#include<iostream>

using namespace std;

int main()

{

Triplet T;

int a,b;

a = 1, b = 2;

cout << &a << endl;

//int b;

cout << &b << endl;

InitTriplet(T, 1, 2, 3);

cout << T << endl;

ElemType e;

int i;

i = 1;

cout << IsAscending(T) << endl;

cout << IsDescending(T) << endl;

//cout << DestroyTriplet(T) << endl;

Get(T, i, e);

cout << Get(T,i, e) << endl;

Put(T, i, e);

cout << Put(T, i, e) << endl;

Max(T, e);

cout << e << endl;

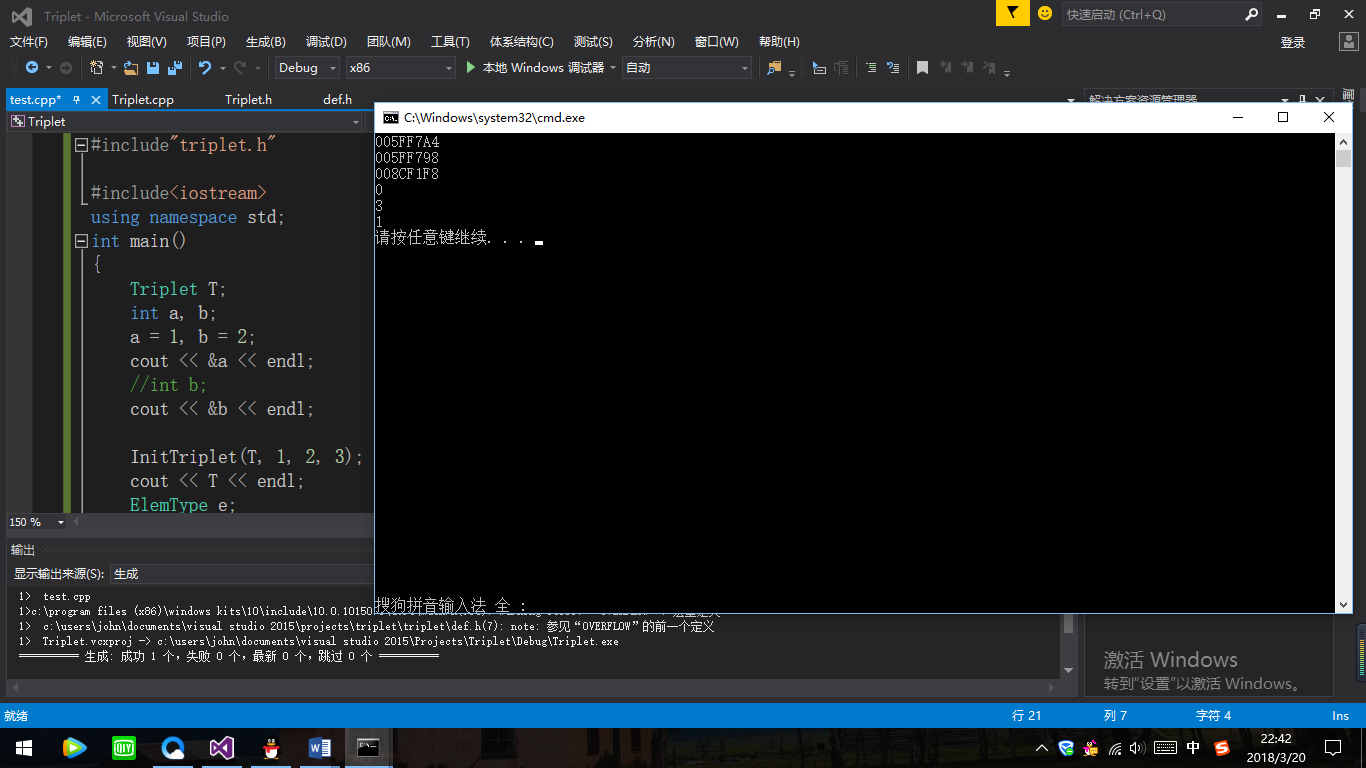
Min(T, e);

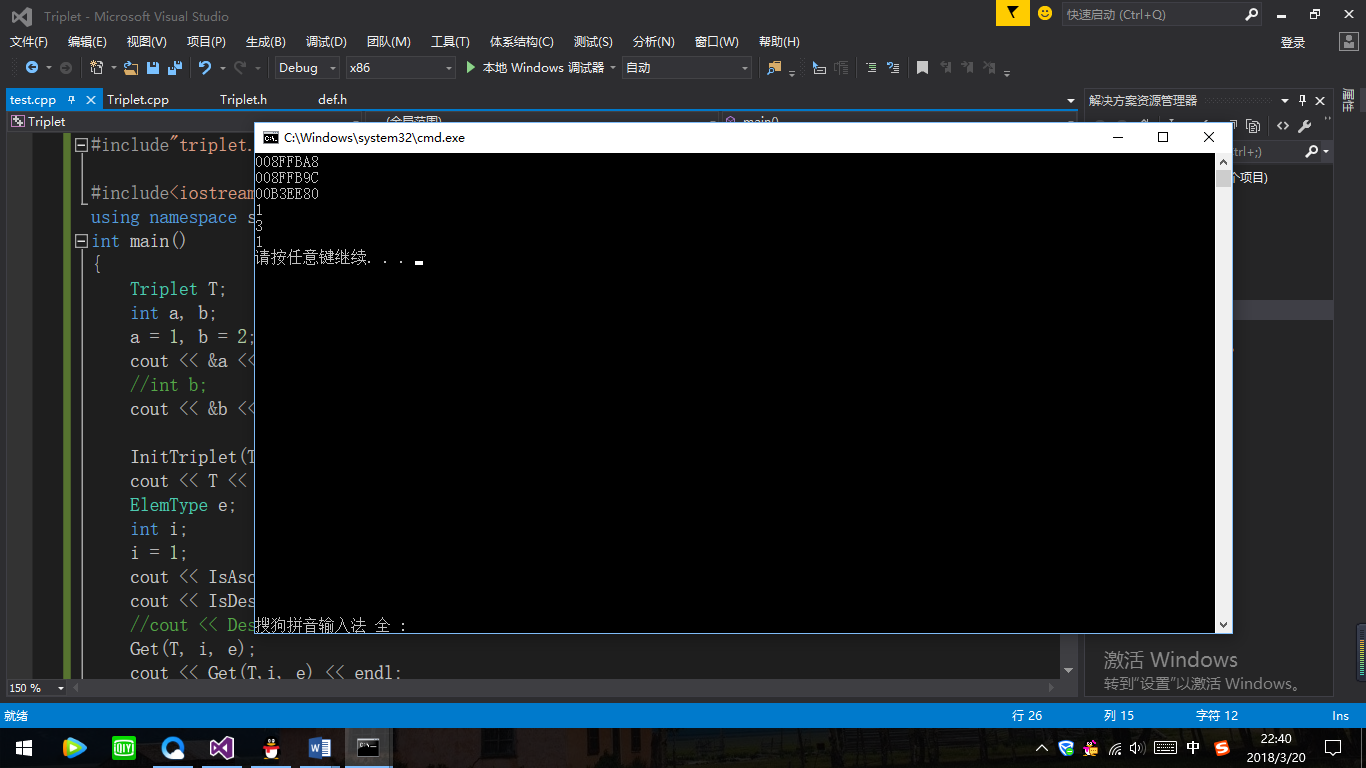
cout << e << endl;

return 0;

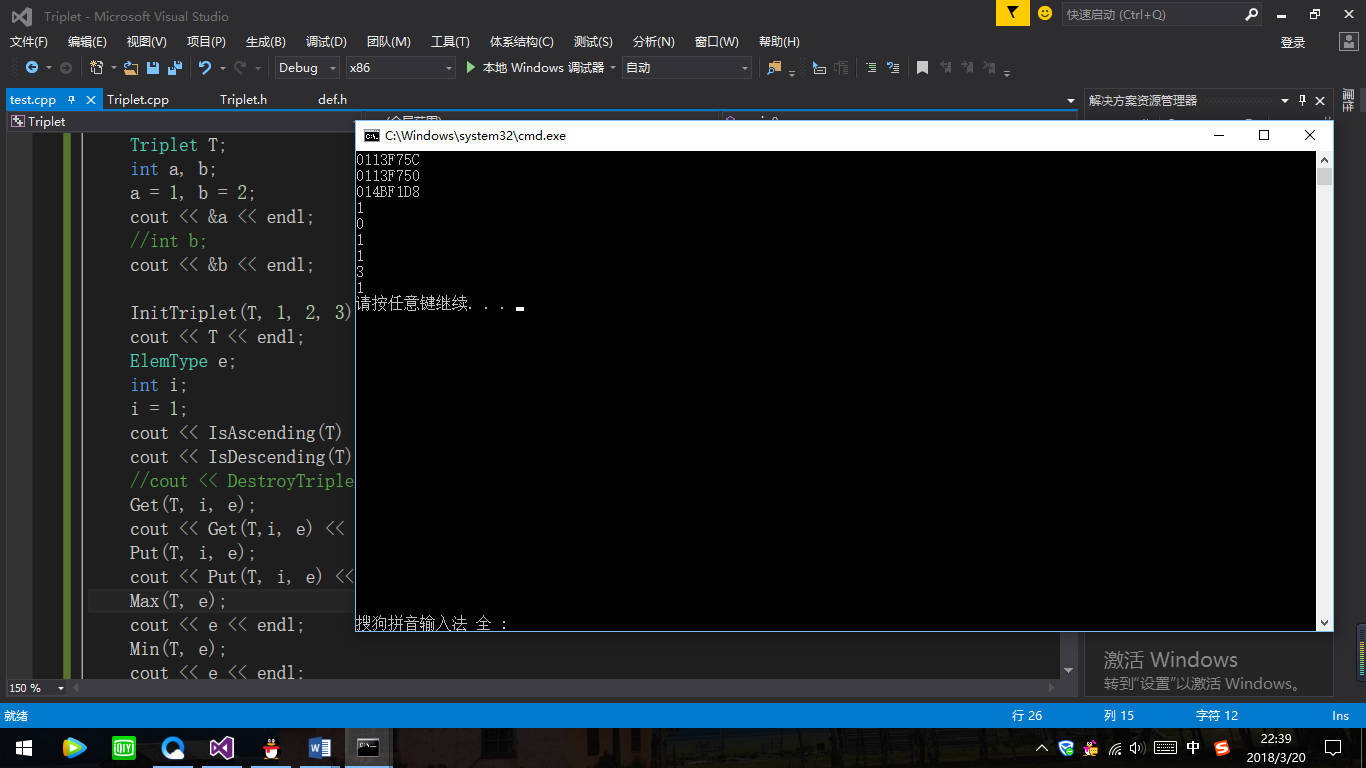
}

1. **测试过程：**





1. **测试结果：**



1. **结果分析：**

**0113F75C :输出的a的地址**

**0113F750 :输出的b的地址**

**014BF1D8 :给三元组T的三个元素分配空间**

1. **:** **T的三个元素按升序排列，返回1**
2. **：T的三个元素按降序排列，返回0**
3. **：用e返回T的第1个元的值**
4. **:置T的第1个元的值为e**

**3 :用e返回T的元素的最大值**

**1 ：用e返回T的元素的最小值**