实验内容：

1. 掌握类的声明和使用
2. 掌握类的声明和对象的声明
3. 复习具有不同访问属性的成员的访问方式
4. 观察构造函数和析构函数的执行过程
5. 学习类的组合使用方法
6. 观察程序运行中变量的作用域、生存期和可见性。
7. 学习类的静态成员的使用。
8. 学习多文件结构在C++程序中的使用。
9. 学会使用Visual Studio 2008 开发环境中的debuge调试功能：单步执行、设置断点、观察变量值。

**实验一**：

声明一个CPU类，包含等级（rank），频率（frequency）,电压（voltage）等属性，有两个共有成员函数run,stop,其中，rank为枚举类型CPU\_Rank,声明为enumCPU\_Rank{P1=1,P2,P3,P4,P5,P6,P7},frequency为单位是MHz的整型数，voltage为浮点型的电压值，观察构造函数和析构函数的调用顺序。

// 实验一.cpp : 定义控制台应用程序的入口点。

//

#include "stdafx.h"

#include <iostream>

using namespace std;

enum CPU\_Rank {P1=1,P2,P3,P4,P5,P6,P7};

class CPU{

private:

CPU\_Rank rank;

int frequency;

float voltage;

public:

CPU(CPU\_Rank r, int f, float v){

rank = r;

frequency = f;

voltage = v;

cout << "constructing a CPU" << endl;

}

~CPU(){

cout << "destructing a CPU" << endl;

}

void run(){

cout << "CPU starts operating" << endl;

}

void stop(){

cout << "CPU stops operating" << endl;

}

};

int main(){

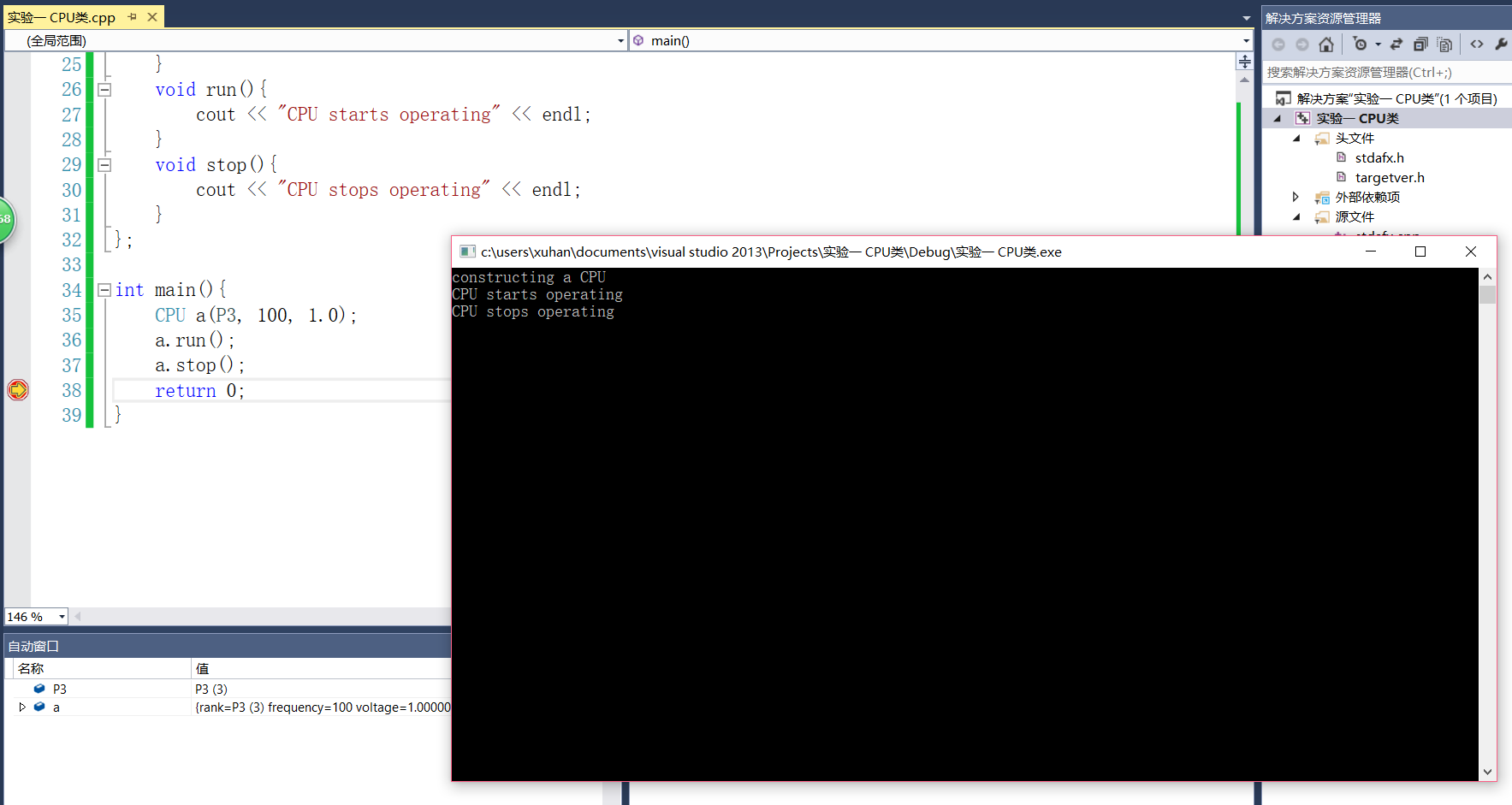
CPU a(P3, 100, 1.0);

a.run();

a.stop();

return 0;

}



实验二：

实验描述：声明一个简单的Computer类，有数据成员芯片(cpu)，内存（ram），光驱（cdrom）,等，有两个共有成员函数run.stop.cpu为CPU类的一个对象，ram为RAM类的一个对象，cdrom为CDROM类的一个对象，声明并实现这个类。

// 实验二.cpp : 定义控制台应用程序的入口点。

//

#include "stdafx.h"

#include <iostream>

using namespace std;

enum CPU\_Rank {P1=1,P2,P3,P4,P5,P6,P7};

class CPU{

private:

CPU\_Rank rank;

int frequency;

float voltage;

public:

CPU(CPU\_Rank r, int f, float v){

rank = r;

frequency = f;

voltage = v;

cout << "constructing a CPU" << endl;

}

CPU(CPU&CPU1){

cout << "using the copy constructer to construct a CPU" << endl;

}

~CPU(){

cout << "destructing a CPU" << endl;

}

};

class RAM{

private:

int ram;

public:

RAM(int r){

ram = r;

cout << "constructing a RAM" << endl;

}

RAM(RAM&RAM1){

cout << "using the copy constructer to construct a RAM" << endl;

}

~RAM(){

cout << "destructing a RAM" << endl;

}

};

class CDROM{

private:

int cdrom;

public:

CDROM(int c){

cdrom = c;

cout << "constructing a CDROM" << endl;

}

CDROM(CDROM&CDROM1){

cout << "using the copy constructer to construct a CDROM" << endl;

}

};

class COMPUTER{

private:

CPU A;

RAM B;

CDROM C;

public:

COMPUTER(CPU cpu, RAM ram, CDROM cdrom);

~COMPUTER(){

cout << "destructing a COMPUTER" << endl;

}

void run(){

cout << "COMPUTER stars operating" << endl;

}

void stop(){

cout << "COMPTER stops operating" << endl;

}

};

COMPUTER::COMPUTER(CPU cpu, RAM ram, CDROM cdrom):A(cpu),B(ram),C(cdrom){

cout << "constructing a COMPUTER" << endl;

}

int main(){

CPU a(P3, 100, 1.0);

RAM b(1);

CDROM c(3);

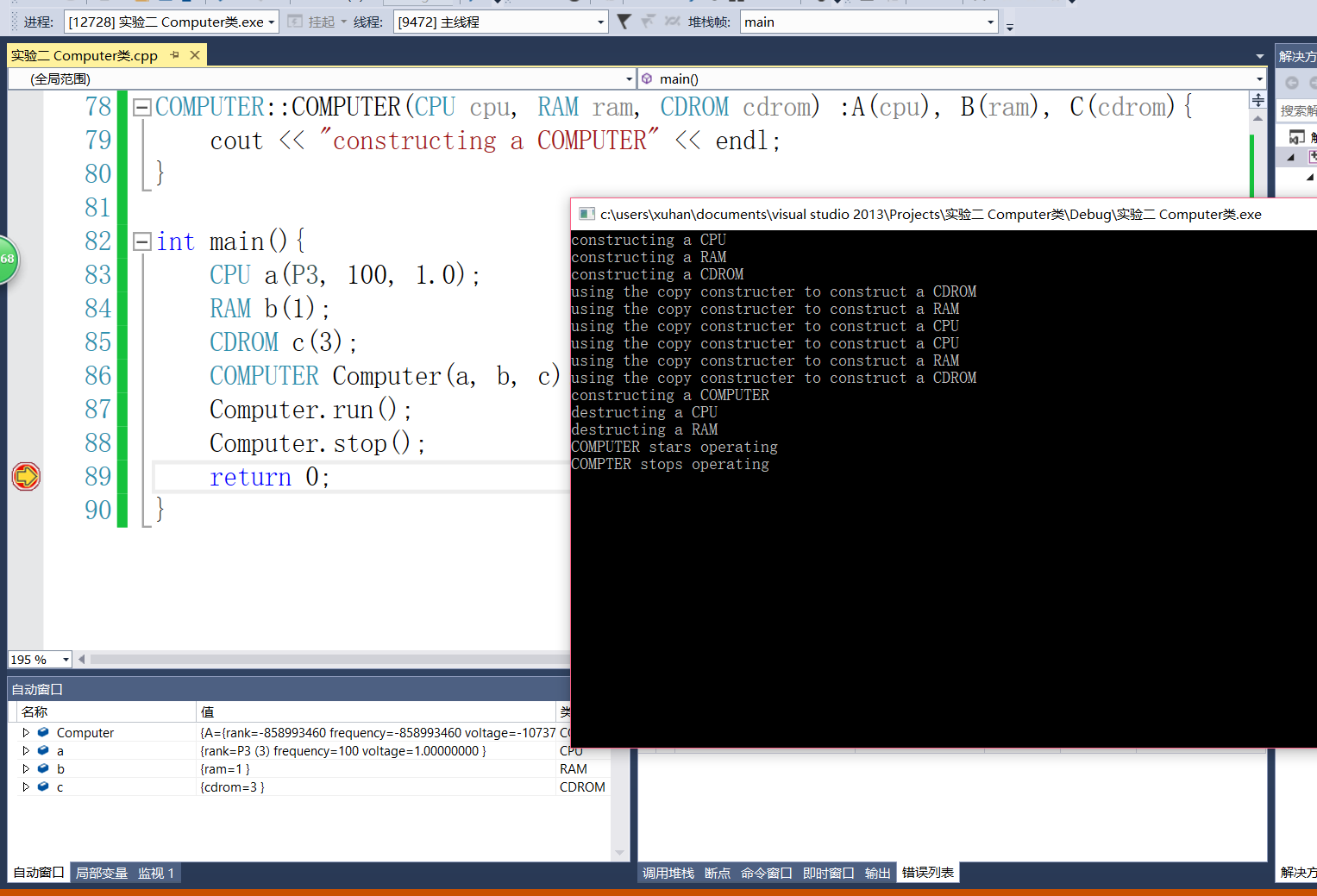
COMPUTER Computer(a, b, c);

Computer.run();

Computer.stop();

return 0;

}



实验三：

实验描述：

1. 运行以下程序，观察程序输出。

#include<iostream>

using namespace std;

void fn1();

int x=1,y=2;

int main()

{

cout<<"Begin..."<<endl;

cout<<"x="<<x<<endl;

cout<<"y="<<y<<endl;

cout<<"Evaluate x and y in main()..."<<endl;

int x=10,y=20;

cout<<"x="<<x<<endl;

cout<<"y="<<y<<endl;

cout<<"Step int fn1()..."<<endl;

fn1();

cout<<"Back in main"<<endl;

cout<<"x="<<x<<endl;

cout<<"y="<<y<<endl;

return 0;

}

void fn1()

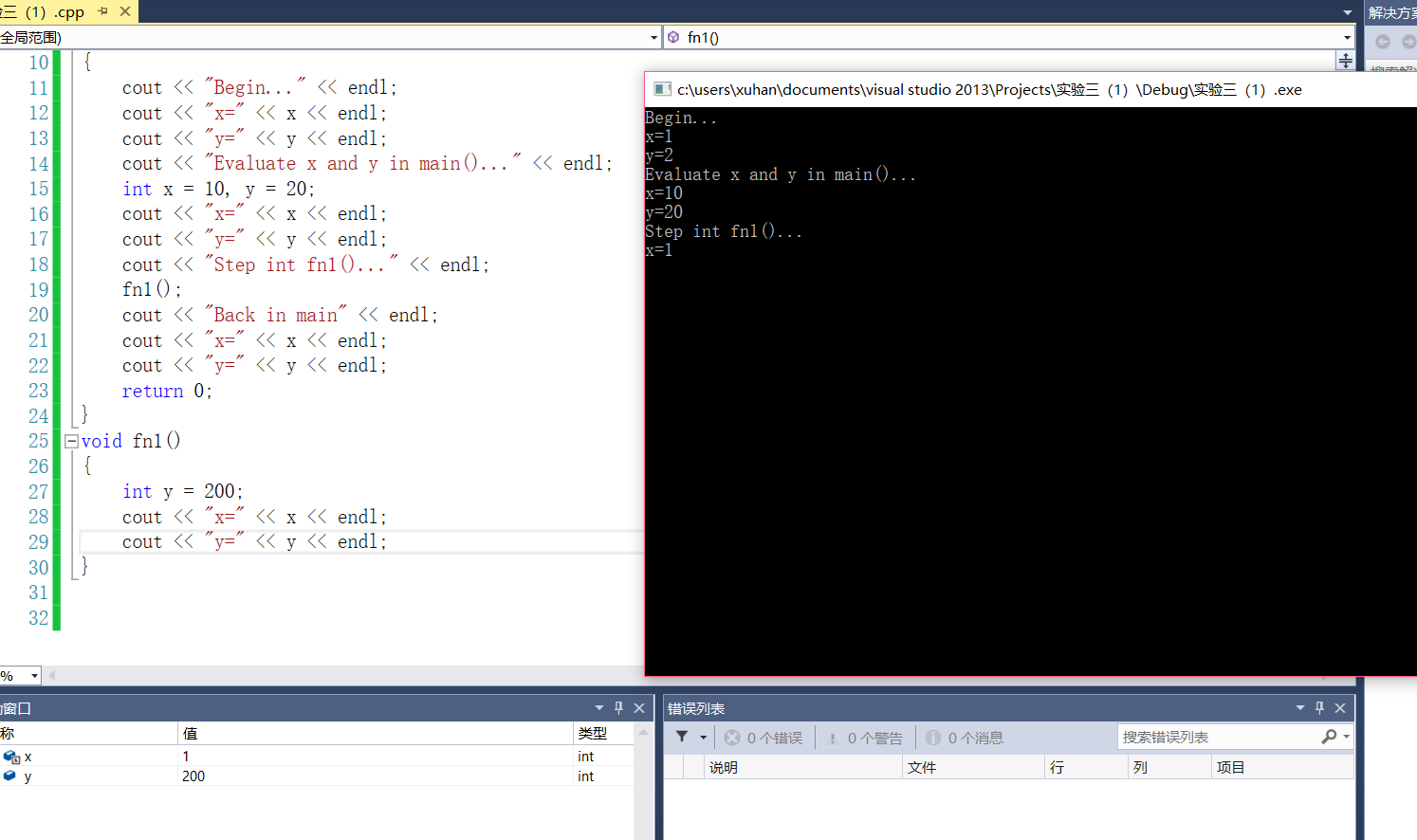
{

int y=200;

cout<<"x="<<x<<endl;

cout<<"y="<<y<<endl;

}



1. 新建一个空的项目lab5\_2,添加头文件client.h，在其中声明类CLIENT,注意使用编译预处理命令；再添加源程序文件client.cpp,在其中实现CLIENT类，注意静态成员变量的使用方法；再添加文件lab5\_2.cpp,在其中定义main（）函数，测试CLIENT类，观察相应的成员变量取值的变化情况。

//client.h

#ifndef \_\_CLIENT\_H\_\_

#define \_\_CLIENT\_H\_\_

class CLIENT{

public:

CLIENT(int ID);

virtual ~CLIENT();

virtual int GetID();

virtual void SetID(int ID1);

private:

int ID;

static int CLIENTNUM;

};

#endif// \_\_CLIENT\_H\_\_

/ client.cpp : 定义控制台应用程序的入口点。

//

#include <iostream>

#include "client.h"

using namespace std;

CLIENT::CLIENT(int ID) :ID(ID){

CLIENTNUM++;

}

int CLIENT::CLIENTNUM = 0;

CLIENT::~CLIENT(){

CLIENTNUM--;

}

int CLIENT::GetID(){

return ID;

}

void CLIENT::SetID(int ID1){

ID = ID1;

}

//lab5\_2.cpp

#include "client.h"

#include<iostream>

using namespace std;

int main(){

CLIENT X(2017310718);

CLIENT Y(2017310719);

cout << "X" << ":" << X.GetID() << endl;

cout << "Y" << ":" << Y.GetID() << endl;

return 0;

}

