## 实验一

### 程序

#include<iostream.h>

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

class CPU{

private:

CPU\_Rank rank;

int frequency;

float voltage;

public:

//构造函数

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

rank=r;

frequency=f;

voltage=v;

cout<<"构造了一个CPU"<<endl;

}

//拷贝构造函数

CPU(CPU &p){

rank=p.rank;

frequency=p.frequency;

voltage=p.voltage;

cout<<"拷贝构造了一CPU"<<endl;

}

//析构函数

~CPU(){cout<<"析构了一个CPU"<<endl;}

CPU\_Rank GetRank(){return rank;}

int Getfrequency(){return frequency;}

float Getvoltage(){return voltage;}

void setRank(CPU\_Rank r){rank=r;}

void setfrequency(int f){frequency=f;}

void setvoltage(float v){voltage=v;}

void run(){cout<<"CPU开始运行"<<endl;}

void stop(){cout<<"CPU stop"<<endl;}

};

void main(){

CPU a(P5,230,3.1);

a.run();

a.stop();

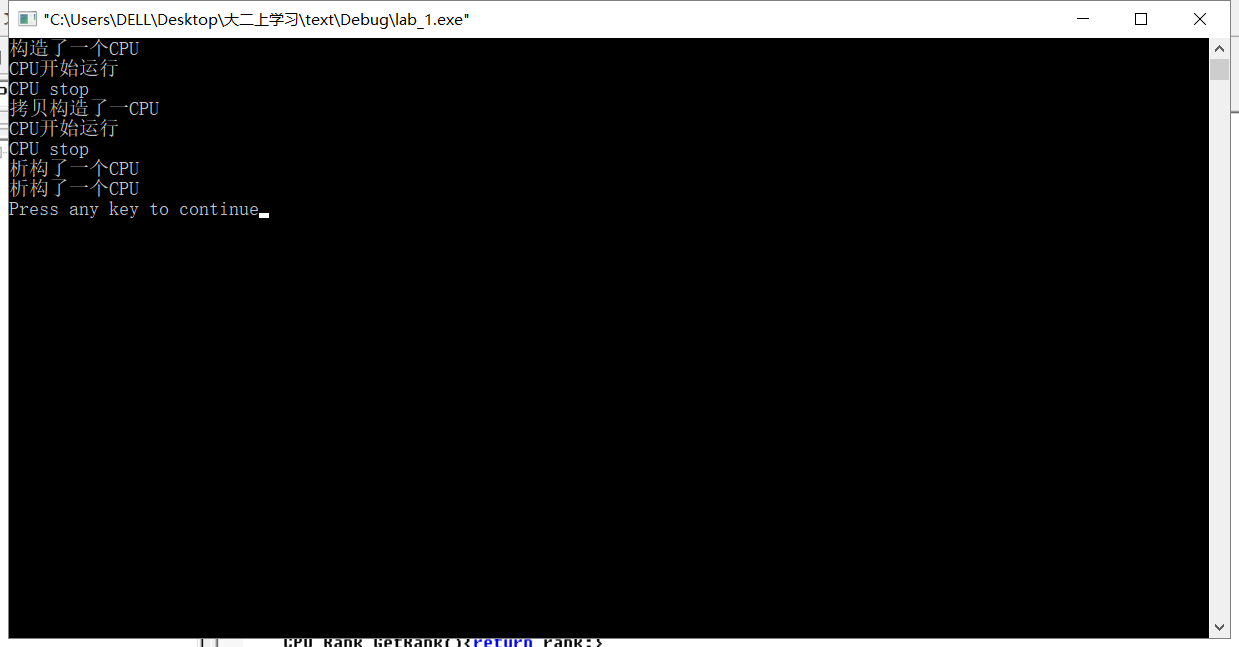
CPU b(a);

b.run();

b.stop();

}

### 程序运行结果



## 实验二

### 程序

#include<iostream.h>

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

enum RAM\_Rank{G1=1,G2,G4,G8};

enum CDROM\_Rank{M700=1,M750,M800};

enum CDROM\_Speed{X24=1,X36,X40,X50};

class CPU{

private:

CPU\_Rank rank;

int frequency;

float voltage;

public:

//构造函数

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

rank=r;

frequency=f;

voltage=v;

cout<<"构造了一个CPU"<<endl;

}

//拷贝构造函数

CPU(CPU &p){

rank=p.rank;

frequency=p.frequency;

voltage=p.voltage;

cout<<"拷贝构造了一CPU"<<endl;

}

//析构函数

~CPU(){cout<<"析构了一个CPU"<<endl;}

CPU\_Rank GetRank(){return rank;}

int Getfrequency(){return frequency;}

float Getvoltage(){return voltage;}

void setRank(CPU\_Rank r){rank=r;}

void setfrequency(int f){frequency=f;}

void setvoltage(float v){voltage=v;}

};

class RAM{

private:

RAM\_Rank ramRank;

float speed;

public:

//构造函数

RAM(RAM\_Rank r, float s){

ramRank=r;

speed=s;

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

}

//拷贝构造函数

RAM(RAM &p){

ramRank=p.ramRank;

speed=p.speed;

cout<<"拷贝构造了一个RAM"<<endl;

}

//析构函数

~RAM(){cout<<"析构了一个RAM"<<endl;}

RAM\_Rank getRank(){return ramRank;}

float getspeed(){return speed;}

void setRank(RAM\_Rank r){ramRank=r;}

void setspeed(float s){speed=s;}

};

class CDROM{

private:

CDROM\_Rank cdromRank;

CDROM\_Speed cdromSpeed;

public:

//构造函数

CDROM(CDROM\_Speed s,CDROM\_Rank r){

cdromSpeed=s;

cdromRank=r;

cout<<"构造了一个CDROM"<<endl;

}

//拷贝构造函数

CDROM(CDROM &p){

cdromSpeed=p.cdromSpeed;

cdromRank=p.cdromRank;

cout<<"拷贝构造了一个CDROM"<<endl;

}

//析构函数

~CDROM(){cout<<"析构了一个CDROM"<<endl;}

CDROM\_Speed getcdromSpeed(){return cdromSpeed;}

CDROM\_Rank getcdromRank(){return cdromRank;}

void setcdromSpeed(CDROM\_Speed s){cdromSpeed=s;}

void setcdromRank(CDROM\_Rank r){cdromRank=r;}

};

class computer{

private:

CPU cpu;

RAM ram;

CDROM cdrom;

public:

//构造函数

computer(CPU c,RAM r,CDROM cd):cpu(c),ram(r),cdrom(cd){

cout<<"构造了一个computer"<<endl;

}

//拷贝构造函数

computer(computer &p):cpu(p.cpu),ram(p.ram),cdrom(p.cdrom){

cout<<"拷贝构造了一个computer"<<endl;

}

//析构函数

~computer(){cout<<"析构了一个computer"<<endl;}

void run(int a){cout<<"computer"<<a<<"开始运行"<<endl;}

void stop(int a){cout<<"computer"<<a<<"stop"<<endl;}

};

void main(){

CPU cpu1(P5,220,3.8);

RAM ram1(G2,200);

CDROM cdrom1(X50,M800);

computer computer1(cpu1,ram1,cdrom1);

computer1.run(1);

computer1.stop(1);

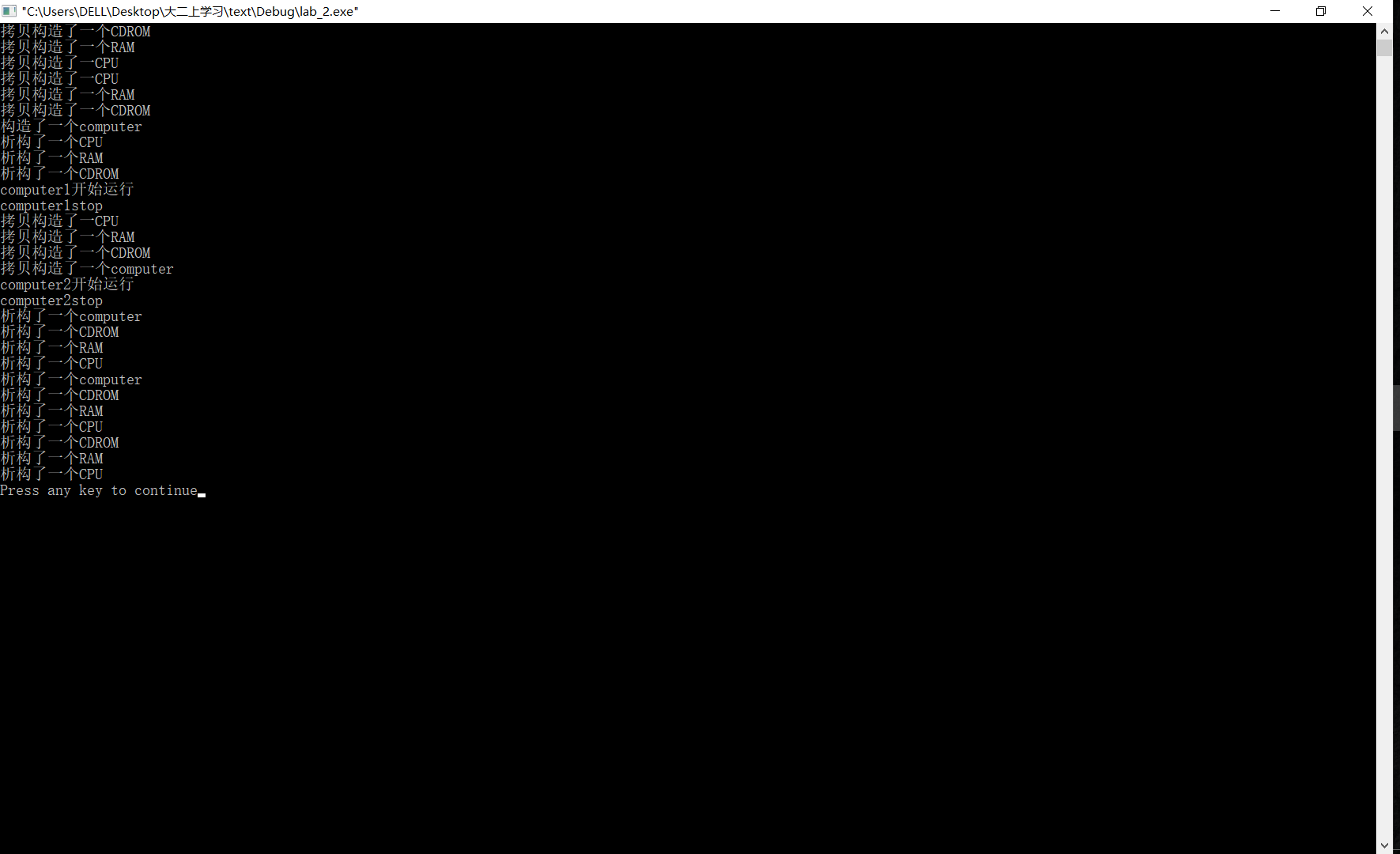
computer computer2(computer1);

computer2.run(2);

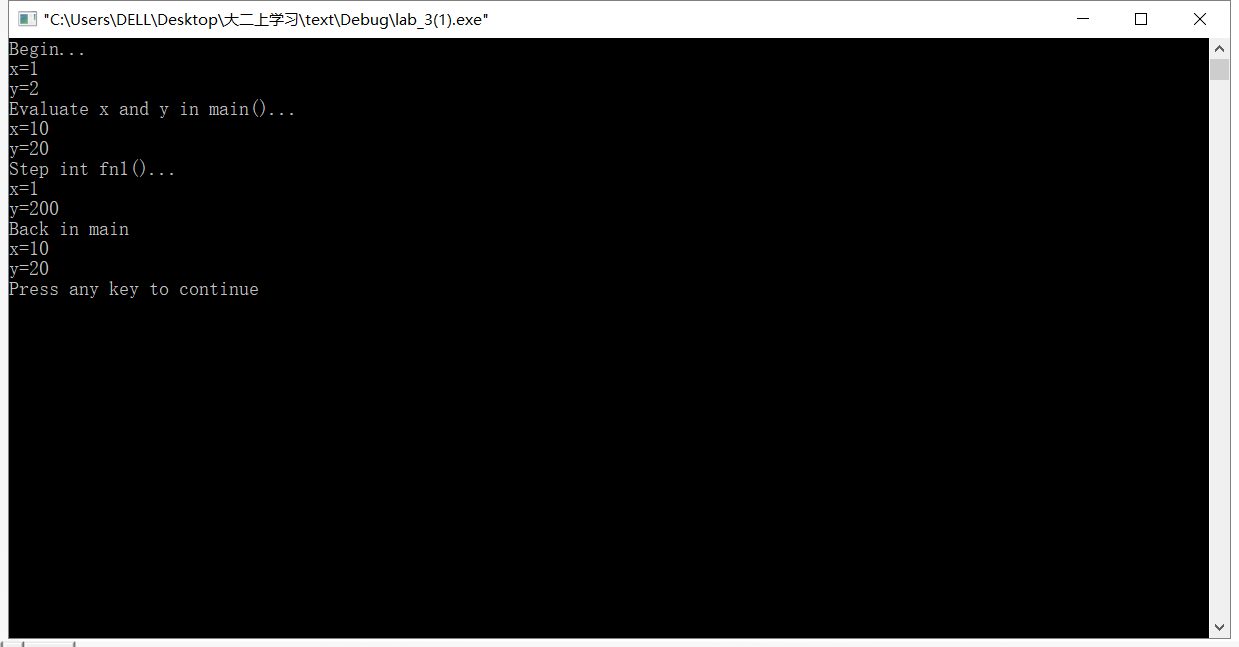
computer2.stop(2);

}

### 程序运行结果



## 实验三

1. （1）实验运行结果

### 运行结果分析

全局变量的作用域为文件作用域，在整个程序运行期间有效，但是局部模块中声明同名变量后，可见的是局部变量，这时id全局变量不可见。

而局部变量的生存期只限于相应的程序模块中，离开该程序模块，局部变量x,y就不再存在，此时同名的全局变量重新可见。

### (1)程序

Client.h

#ifndef CLIENT\_H

#define CLIENT\_H

//其中静态成员为类的属性，为所有的类的对象共同拥有

class client{

public:

client();

~client();

static void ChangeServerName(char ss);//改变服务器名称

static void showServeName();

static void showClientNum();

private:

static char ServeName;//保存服务器名，引用说明

static int ClientNum;//记录已经定义的客户（对象）的数量，引用说明

}；

#endif

Client.cpp

#include<iostream.h>

#include"client.h"

client::client(){ClientNum++;}//增加一个对象

client::~client(){ClientNum--;}//减少一个对象

void client::ChangeServerName(char sn){ServerName=sn;}

void client::showServerName(){cout<<"服务器名："<<ServerName<<endl;}

void client::showClientNum(){cout<<"客户总数："<<ClientNum<<endl;}

char client::ServerName='A';

int client::ClientNum=0;

Lab\_3(2).cpp

#include<iostream.h>

#include"client.h"

void main(){

client::showServerName();//初始状态

client::ChangeClientNum();//类名引用静态成员函数

client::ChangeServerName('B');

client a;//增加一个用户

a.ShowServerName();

a.showClientNum();

{

client b;

b.ShowServerName();

b.showClientNum();

}//减少一个用户

client::ShowServerName();

client::showClientNum();

}

### 运行结果

