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。

#### 代码：

#include<iostream>

using namespace std;

class CPU{

public:

CPU() { cout << "constructer called"<<endl; };

CPU(int fre, int vol) :frequency(fre), voltage(vol) { cout << "constructer called"<<endl; };

~CPU() { cout << "destructer called"<<endl; };

private:

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

int frequency;

int voltage;

};

int main() {

CPU cpu;

return 0;

}

#### KKY`X@FO`_P3[CA6L_)7~AC图片：

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。

###### 代码：

#include<iostream>

using namespace std;

class CPU {

public:

CPU() { cout << "constructer called" << endl; };

CPU(int fre, int vol) :frequency(fre), voltage(vol) { cout << "constructer called" << endl; };

~CPU() { cout << "destructer called" << endl; };

private:

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

int frequency;

int voltage;

};

class RAM{

public:

RAM() { };

void setRam(bool b);

void showRam();

private:

bool isOpen;

};

void RAM::setRam(bool b) {

isOpen = b;

};

void RAM::showRam() {

cout << "RAM:" << isOpen << endl;

};

class CDROM {

public:

CDROM() { };

void setCDRom(bool b);

void showCDRom();

private:

bool isOpen;

};

void CDROM::setCDRom(bool b) {

isOpen = b;

}

void CDROM::showCDRom() {

cout << "CDROM:" << isOpen << endl;

}

class Computer {

public:

Computer() {};

Computer(CPU nCpu, RAM nRam, CDROM nCdrom) :cpu(nCpu), ram(nRam),cdrom(nCdrom) {};

void run();

void stop();

private:

CPU cpu;

RAM ram;

CDROM cdrom;

};

void Computer::run() {

ram.setRam(true);

cdrom.setCDRom(true);

ram.showRam();

cdrom.showCDRom();

};

void Computer::stop() {

ram.setRam(false);

cdrom.setCDRom(false);

ram.showRam();

cdrom.showCDRom();

};

int main() {

CPU cpu;

RAM ram;

CDROM cdrom;

Computer com(cpu, ram, cdrom);

com.run();

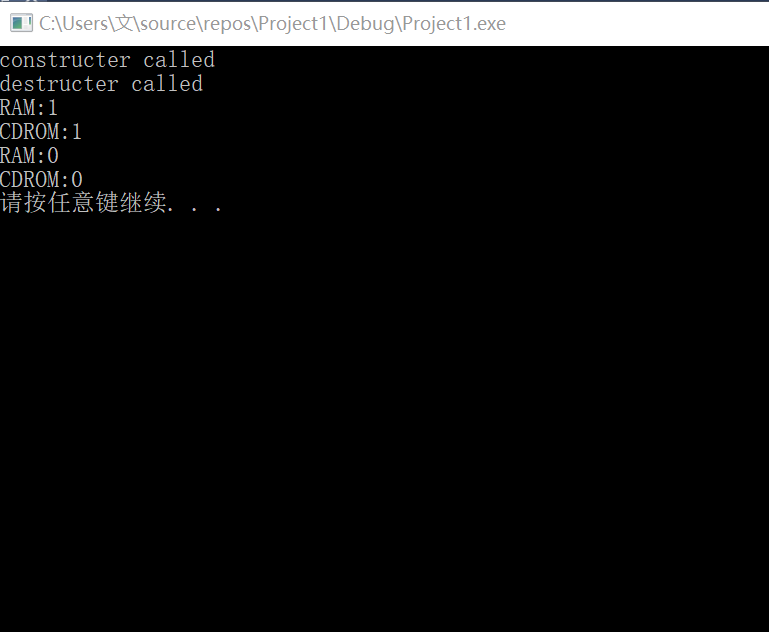
com.stop();

system("pause");

return 0;

}

###### 图片：



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()

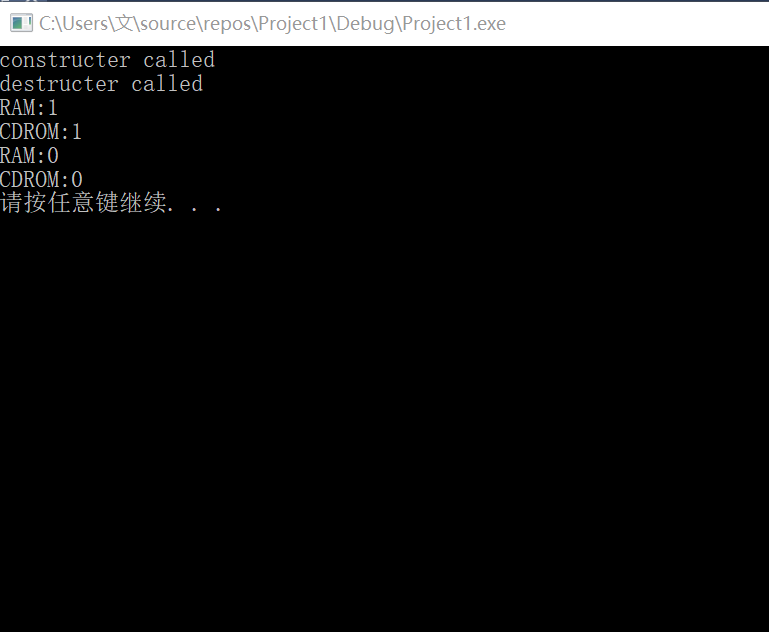
{

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

using namespace std;

class CLIENT {

public:

CLIENT() {};

CLIENT(int nID, string nName);

void setName(string nName);

void setID(int nID);

void showClient();

void static showCount();

private:

int id;

string name;

int static count;

};

###### client.cpp

#include<iostream>

#include<string>

#include"client.h"

using namespace std;

CLIENT::CLIENT(int nID, string nName) {

id = nID;

name = nName;

count++;

}

void CLIENT::showClient(){

cout << "id is:" << id <<" "<<"name is:"<<name<< endl;

};

void CLIENT::setID(int nID) {

id = nID;

}

void CLIENT::setName(string nName) {

name = nName;

}

void CLIENT::showCount() {

cout << "the count is:" << count << endl;

}

###### lab5\_2.cpp

#include<iostream>

#include<string>

#include"client.h"

#include<iostream>

using namespace std;

int static count = 0;

int main() {

CLIENT client1(01,"Lily");

CLIENT::showCount();

CLIENT client2(02, "Rose");

CLIENT client3(03, "Jack");

client1.showClient();

client2.showClient();

client3.showClient();

client3.setID(5);

client3.setName("Bamboo");

client3.showClient();

CLIENT::showCount();

system("pause");

return 0;

}