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。

程序：

#include<iostream>

using namespace std;

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

class CPU{

public:

   static CPU\_Rank rank;

    int frequency;

    float voltage;

    CPU();

    CPU(CPU\_Rank rank,int frequency,float voltage);

    ~CPU(){

        cout<<"调用CPU的析构函数"<<endl;}

};

CPU\_Rank CPU::rank=P1;

CPU::CPU(){

    rank=P1;

    frequency=100;

    voltage=10000;

     cout<<"调用CPU的构造函数"<<endl;

}

CPU::CPU(CPU\_Rank rank,int frequency,float voltage){

    this->rank=rank;

    this->frequency=frequency;

    this->voltage=voltage;

         cout<<"调用CPU的构造函数"<<endl;

}

int main(){

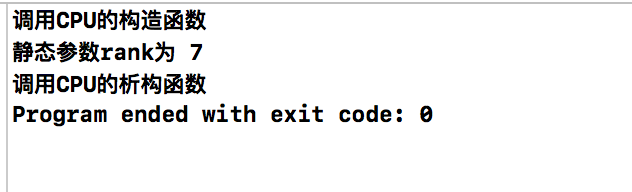
    CPU a();

    CPU b(P7,666,6666);

    cout<<"静态参数rank为 "<<CPU::rank<<endl;

}

运行结果：



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。

程序：

#include<iostream>

using namespace std;

class CPU{

public:

    CPU(){

        cout<<"构造了一个CPU对象"<<endl;}

    ~CPU(){

        cout<<"调用析构函数:CPU类"<<endl;

    }

};

class RAM{

public:

    RAM(){

        cout<<"构造了一个RAM对象"<<endl;

    }

    ~RAM(){

        cout<<"调用析构函数:RAM类"<<endl;

    }

};

class CDROM{

    public :

    CDROM(){

        cout<<"构造了一个CDROM对象"<<endl;

    }

    ~CDROM(){

        cout<<"调用析构函数:CDROM类"<<endl;

    }

};

class Computer{

public:

    CPU cpu;

    RAM ram;

    CDROM cdrom;

    void run();

    void stop();

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

    ~Computer(){

        cout<<"调用析构函数:Computerl类"<<endl;

    }

};

Computer::Computer(CPU cpu,RAM ram,CDROM cdrom):cpu(cpu),ram(ram),cdrom(cdrom){

    cout<<"构造了一个Computer对象"<<endl;

}

int main(){

    CPU a;

    RAM b;

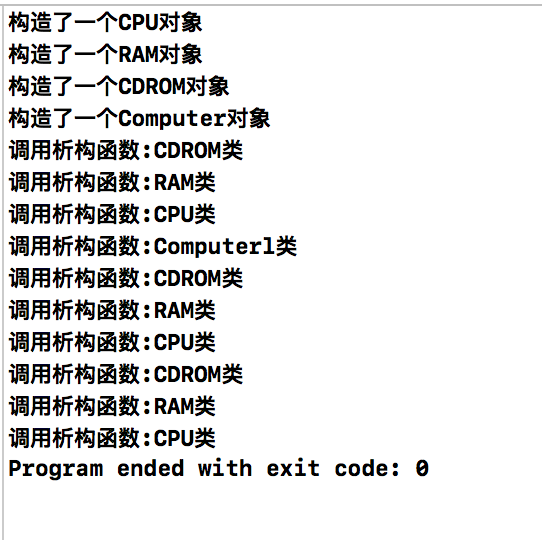
    CDROM c;

    Computer d(a,b,c);

    return 0;

}

运行结果：



**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;

}

运行结果为：

**Begin...//cout函数输出，函数开始运行**

**x=1//输出了命名空间中的X**

**y=2//输出了命名空间中的y**

**Evaluate x and y in main()...//cout函数输出，开始观察main函数中的变量**

**x=10//输出了拥有类作用域的x**

**y=20//输出了拥有类作用域的y**

**Step int fn1()...//cout函数输出，开始观察函数fn1**

**x=1//函数fn1中没有改动拥有命名空间作用域的x，仍为一开始的1**

**y=200//函数fn1中改动了拥有命名空间作用域的y,由2变为200**

**Back in main//cout函数输出，再次观察main函数中的变量**

**x=10//输出了拥有类作用域的x，仍为10**

**y=20//输出了拥有类作用域的y，仍为20**

**Program ended with exit code: 0**

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.hpp**

#include<string>

using namespace std;

class CLIENT{

public:

    CLIENT(string name);

    CLIENT(string s,int i,int a);

 string getName();

int getAge();

    int getIdNumber();

private:

    string name;

    unsigned int idNumber;

    int age;

};

**Client.cpp**

#include "client.hpp"

#include<string>

#include<iostream>

using namespace std;

CLIENT::CLIENT(string name){

    this->name=name;

    cout<<"创建用户信息"<<endl;

    idNumber=0;

    age=0;

}

CLIENT::CLIENT(string s,int i,int a){

    name=s;

    idNumber=i;

    age=a;

    cout<<"创建用户信息"<<endl;

}

string CLIENT::getName(){

    return name;

}

int CLIENT::getAge(){

    return age;

}

int CLIENT::getIdNumber(){

    return idNumber;

}

**Lab5\_2.cpp**

#include "client.hpp"

#include<iostream>

using namespace std;

int main(){

    CLIENT a("中央财经大学");

    CLIENT b("蜡笔小新",123456,4);

    cout<<"对象a的name为"<<a.getName()<<endl;

    cout<<"对象a的age为"<<a.getAge()<<endl;

    cout<<"对象a的idNumber为"<<a.getIdNumber()<<endl;

    cout<<"对象b的name为"<<b.getName()<<endl;

    cout<<"对象b的age为"<<b.getAge()<<endl;

    cout<<"对象b的idNumber为"<<b.getIdNumber()<<endl;

    return 0;

}

