实验报告

金融实验班17 汪雨佳 2017311307

**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。

// Lab1.cpp

#include "pch.h"

#include <iostream>

using namespace std;

enum CPU\_Rank

{

P1 = 1, P2, P3, P4, P5, P6, P7

};

class CPU

{

public:

CPU();

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

~CPU();

float run() { return voltage ; }

int stop() { return rank ; }

private:

CPU\_Rank rank;

int frequency;

float voltage;

};

CPU::CPU()

{

cout << "construcing CPU" << endl;

}

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

{

rank = r;

frequency = f;

voltage = v;

}

CPU::~CPU()

{

cout << "destructing CPU" << endl;

}

int main()

{

CPU c;

CPU a(P3, 50, 220);

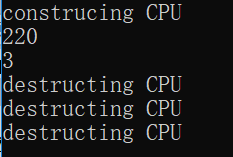
CPU b = a;

cout << a.run() << endl;

cout << b.stop() << endl;

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。

// Lab2.cpp

#include "pch.h"

#include <iostream>

using namespace std;

enum CPU\_Rank

{

P1 = 1, P2, P3, P4, P5, P6, P7

};

class CPU

{

public:

CPU();

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

~CPU();

float run() { return voltage; }

int stop() { return rank ; }

private:

CPU\_Rank rank;

int frequency;

float voltage;

};

CPU::CPU()

{

cout << "construcing CPU" << endl;

}

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

{

rank = r;

frequency = f;

voltage = v;

}

CPU::~CPU()

{

cout << "destructing CPU" << endl;

}

class RAM

{

public:

RAM() { cout << "constructing RAM" << endl; }

~RAM() { cout << "destructing RAM" << endl; }

};

class CDROM

{

public:

CDROM() { cout << "constructing CDROM" << endl; }

~CDROM() { cout << "destructing CDROM" << endl; }

};

class Computer

{

public:

Computer() { cout << "constructing Computer" << endl; }

~Computer() { cout << "destructing Computer" << endl; }

private:

CPU cpu;

RAM ram;

CDROM cdrom;

};

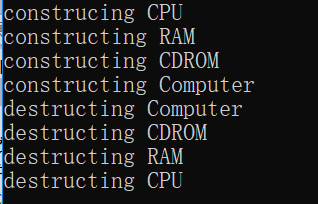
int main()

{

Computer c;

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()

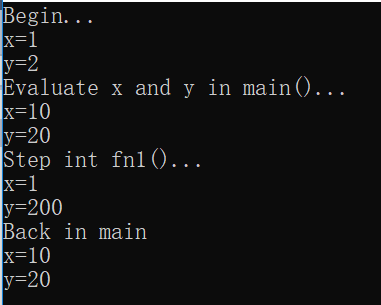
{

int y=200;

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

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

}



最开始的x和y是全局变量，所以输出的值分别为1、2。

第二个x和y在main函数中重新声明了一次，因此之前的x、y在函数中不可见，所以输出的值为10,20。

然后调用fn1函数，输出了值为1的全局变量x和值为200的局部变量y。

最后由于fn1中的变量具有动态生存期，随着函数返回而失效，所以输出的x、y依然是main函数中声明的变量，值为10、20。

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(){}

CLIENT(int date, int id, int age);

~CLIENT(){}

int getid() const;

int getage() const;

static int gettotal();

void show() const;

private:

int id;

int age;

int date;

static int total;

};

#endif // CLIENT\_H

//client.cpp

#include "client.h"

#include <iostream>

using namespace std;

int CLIENT::total = 0;

CLIENT::CLIENT(int date, int id, int age):id(id), age(age), date(date) {

cout << date << " #" << id << " is created" << endl;

total++;

}

int CLIENT::getid() const { return id; }

int CLIENT::getage() const { return age; }

int CLIENT::gettotal() { return total; }

void CLIENT::show() const {

cout << "id#" << id << " age:" << age << " date:" << date << endl;

}

// Lab3-2.cpp

#include "client.h"

#include <iostream>

using namespace std;

int main()

{

CLIENT c1(24, 2017311307, 19);

CLIENT c2(56, 2017310212, 20);

CLIENT c3(108, 2018243735, 18);

cout << c1.getid() << endl;

cout << c2.getage() << endl;

cout << c2.gettotal() << endl;

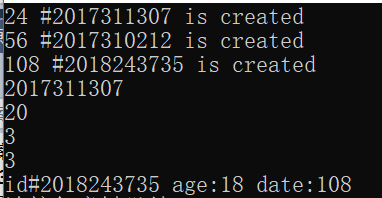
cout << c3.gettotal() << endl;

c3.show();

system("pause");

return 0;

}



变量total被static修饰，具有静态生存周期，每创建一个客户的账户，就会增加一个单位，可以用来计算总客户数量，且不管通过哪个对象调用，值都不会变化。

其他变量具有动态生存周期，用不同的对象调用时，值不同。