**项目一：**

1、声明一个CPU类，包含等级(rank)、频率(frequency)、电压(voltage)等属性，有三个公有成员函数run、stop和setCPU。其中，rank为枚举类型CPU\_Rank，声明为enum CPU\_Rank{P1=1,P2,P3,P4,P5,P6,P7}；frequency为单位是MHz的整数，voltage为浮点型的电压值。run和stop函数分别显示“CPU开始运行！”和“CPU停止运行！”。setCPU用于CPU属性的设置。

2、创建一个CPU对象，并用setCPU函数进行属性设置。

3、使用构造函数代替上面的setCPU成员函数，并创建一个CPU对象。

4、重载CPU类的构造函数(不带参数)，使rank、frequency、voltage分别为P6、3000、220。

5、定义析构函数，在其中输出信息“Good bye！”。观察构造函数和析构函数的调用顺序。

6、定义拷贝构造函数并调用。

7、设计一个简单的主函数对CPU类进行调试。

**考核知识点：**

类的定义、构造函数、析构函数，类的重载

**设计类：**

创建CPU类，该类成员有void run（）、void stop（）、void setCPU（）、void show（）

**主函数：**

int main()

{

CPU a;

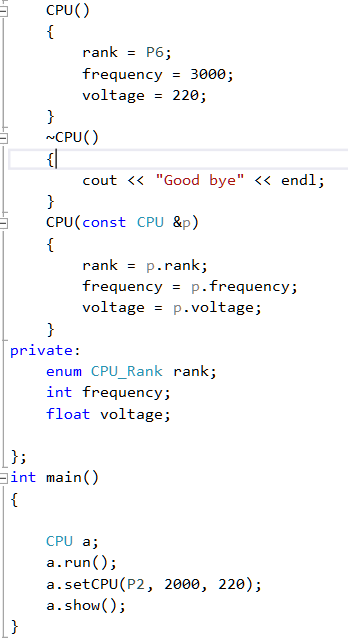
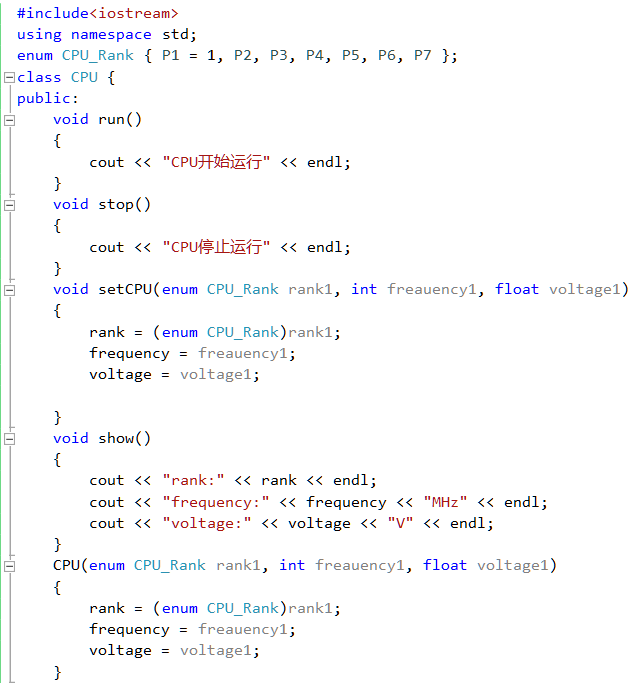
a.run();

a.setCPU(P2, 2000, 220);

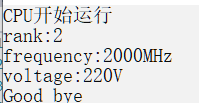
a.show();

}

**完整代码：**

****

**运行结果：**

****

**总结：**

学会了构造函数，对类的定义进一步巩固。

**项目二**：

1、定义一个Stock类，记录一支股票交易的基本信息，信息包括交易日序号(表示本月的第几个交易日，用整数表示)、当日最高价、当日最低价、当日开盘价和当日收盘价。尽量发挥想象力，为该类设计成员函数。

2、在主函数中建立两个股票对象，分别存储该股票昨天和今天两天的信息，以当日收盘价计算该股票今天的涨幅。

3、找出当日涨幅最大的股票。（需要用到静态数据成员）

**考核知识点：**

随机数的使用，成员在对象间的传递

设计类：

Stock类，成员：Stock(string n, double o) 、Stock(string n) 、void set()、void show()、void compare()、void setopen(double n)、double returnclose()、

name、open、close、max、min

主函数：

int main()

{

double n;

Stock yesterday[5] = {

{ "Tencent", 652.00 },

{ "单元式住宅", 660.50 },

{ "公寓式住宅", 157.07 },

{ "复式住宅", 107.72 },

{ "跃层式住宅", 146.60 },

};

for (int i = 0; i < 5; i++)

{

yesterday[i].set();

}

Stock today[5] = {

{ "Tencent" },

{ "单元式住宅" },

{ "公寓式住宅" },

{ "复式住宅" },

{ "跃层式住宅" },

};

for (int i = 0; i < 5; i++)

{

n = yesterday[i].returnclose();

today[i].setopen(n);

today[i].set();

}

for (int i = 0; i < 5; i++)

{

today[i].compare();

}

time\_t rawtime;

struct tm\* ptminfo;

time(&rawtime);

ptminfo = localtime(&rawtime);

cout << ptminfo->tm\_mday << "日涨幅最大股票为：" << endl;

today[c].show();

return 0;

}

完整代码：

#include<iostream>

#include<time.h>

#include<cstdlib>

#include<string>

#include<windows.h>

using namespace std;

double a = 0.0;

int c = 1;

class Stock

{

public:

Stock(string n, double o) :name(n), open(o)

{

}

Stock(string n) :name(n)

{

}

void set();

void show();

void compare();

void setopen(double n);

double returnclose();

private:

string name;

double open;

double close;

double max;

double min;

};

void Stock::show()

{

cout << name << " ";

cout << "开盘价为" << open << ".00";

cout << " 收盘价为" << close << ".00";

cout << " 涨幅为" << (close - open) / open \* 100 << "%" << endl;

}

void Stock::set()

{

srand(650);

max = rand() / 10 + 0.00;

if (max<open)

{

do

{

max = rand() / 10 + 0.00;

} while (max >= open);

}

min = rand() / 10 + 0.00;

if (min>open)

{

do

{

min = rand() / 10 + 0.00;

} while (min <= open);

}

close = rand() / 10 + 0.00;

if (close<min || close>max)

{

do

{

close = rand() / 10 + 0.00;

} while (close <= max || close >= min);

}

}

void Stock::compare()

{

double aa = (close - open) / open \* 100;

if (aa > a)

{

a = aa;

c++;

}

}

double Stock::returnclose()

{

return close;

}

void Stock::setopen(double n)

{

open = n;

}

int main()

{

double n;

Stock yesterday[5] = {

{ "Tencent", 652.00 },

{ "单元式住宅", 660.50 },

{ "公寓式住宅", 157.07 },

{ "复式住宅", 107.72 },

{ "跃层式住宅", 146.60 },

};

for (int i = 0; i < 5; i++)

{

yesterday[i].set();

}

Stock today[5] = {

{ "Tencent" },

{ "单元式住宅" },

{ "公寓式住宅" },

{ "复式住宅" },

{ "跃层式住宅" },

};

for (int i = 0; i < 5; i++)

{

n = yesterday[i].returnclose();

today[i].setopen(n);

today[i].set();

}

for (int i = 0; i < 5; i++)

{

today[i].compare();

}

time\_t rawtime;

struct tm\* ptminfo;

time(&rawtime);

ptminfo = localtime(&rawtime);

cout << ptminfo->tm\_mday << "日涨幅最大股票为：" << endl;

today[c].show();

return 0;

}

**运行结果：**

****

**项目三：**

1、声明一个简单的Computer类，有数据成员芯片(cpu)、内存(ram)、光驱(cdrom)等，有两个公有成员函数run和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。（提示：CPU类为项目一中所定义的。RAM类与CDROM类仿照CPU类进行设计。）

2、要求采用多文件组织，资料参见教材4.8小节。类的接口定义存在头文件中、类的实现存储在cpp文件中。主函数所在文件要求包含头文件。

3、构造函数多样化。（至少想出两种需要初始化的方法）

例如：可以是一台组装机，已有内存、芯片、光驱，组装一台新机器。

可以是一台品牌机，参数有：内存大小、cpu的主频、光驱品牌。

**考核知识点：**

多文件组织的使用，不同的初始化方法，类的组合

**设计类：**

computer类，成员：computer（）、void run（）、void stop（）、void show（）

CPU类，成员：CPU()、void show()、CPU\_rank rank、 frequency、voltage1

RAM类，成员：RAM（）、void show()、velocity1、capacity1、voltage2;

CDROM类，成员：CDROM（）、void show()、velocity2、capacity2、voltage3;

**主函数：**

int main()

{

computer a(P1, 2000, 220, 120, 512, 5, 150, 128, 5);

computer b;

a.run();

a.show();

b.show();

a.stop();

return 0;

}

**完整代码：**

//头文件

#include <iostream>

using namespace std;

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

class CPU

{

public:

CPU(enum CPU\_rank r, int f, float v);

CPU();

void show();

private:

enum CPU\_rank rank;

int frequency;

float voltage1;

};

class RAM

{

public:

RAM(int ve, int c, int vo);

RAM();

void show();

private:

int velocity1;//单位 ns 60

int capacity1;//单位MB 512

int voltage2;

};

class CDROM

{

public:

CDROM(int ve1, int cc, int vo1);

CDROM();

void show();

private:

int velocity2;//单位 kb/s 150

int capacity2;//单位k 128

int voltage3;

};

class computer

{

public:

computer();

computer(CPU\_rank r, int f, float v, int ve, int c, int vo, int ve1, int cc, int vo1);

void run();

void stop();

void show();

private:CPU cpu; RAM ram; CDROM cdrom;

};

//cpp文件

#include "xiangmusan.h"

CPU::CPU(enum CPU\_rank r, int f, float v)

{

rank = (enum CPU\_rank)r; frequency = f; voltage1 = v;

}

void CPU::show()

{

cout << "芯片性能：" << endl;

cout << "等级：" << rank << " " << "频率：" << " " << frequency << "电压：" << " " << voltage1 << endl;

}

CPU::CPU()

{

rank = P2;

frequency = 3000;

voltage1 = 220;

}

RAM::RAM(int ve, int c, int vo)

{

velocity1 = ve;

capacity1 = c;

voltage2 = vo;

}

RAM::RAM()

{

velocity1 = 80;

capacity1 = 128;

voltage2 = 5;

}

void RAM::show()

{

cout << "内存性能：" << endl;

cout << "运行速度：" << velocity1 << " " << "容量：" << capacity1 << " " << "电压：" << voltage2 << endl;

}

CDROM::CDROM(int ve1, int cc, int vo1)

{

velocity2 = ve1; capacity2 = cc; voltage3 = vo1;

}

CDROM::CDROM()

{

velocity2 = 200;

capacity2 = 256;

voltage3 = 5;

}

void CDROM::show()

{

cout << "光驱性能：" << endl;

cout << "运行速度：" << velocity2 << " " << "容量：" << capacity2 << " " << "电压：" << voltage3 << endl;

}

void computer::show()

{

this->cpu.show();

this->ram.show();

this->cdrom.show();

}

void computer::run()

{

cout << "computer类开始运行" << endl;

}

void computer::stop()

{

cout << "computer类停止运行!" << endl;

}

computer::computer(CPU\_rank r, int f, float v, int ve, int c, int vo, int ve1, int cc, int vo1) :cpu(r, f, v), ram(ve, c, vo), cdrom(ve1, cc, vo1)

{}

computer::computer()

{

}

int main()

{

computer a(P1, 2000, 220, 120, 512, 5, 150, 128, 5);

computer b;

a.run();

a.show();

b.show();

a.stop();

return 0;

}

**运行结果：**

****

**总结：**

熟练运用多文件组织，掌握类的组合。

**项目四：**

1、定义一个Employee类，其中包括姓名、街道地址、城市和邮编等属性，以及change\_name()和display()等函数。display()显示姓名、街道地址、城市和邮编等属性，change\_name改变对象的姓名属性。定义包含5个元素的对象数组，每个元素都是Employee类型的对象。

2、将姓名构造成类Name。其名和姓在该类中为保护数据成员，其构造函数为接受一个指向完整姓名字符串的指针，其Display()函数输出姓名。然后将Employee类中的姓名成员（字符数组）换成Name类对象。

3、构成完整的类定义，要求类定义与类的成员函数定义分开。

4、设计一个简单的主函数对Employee类、Name类进行调试。

**考核知识点：**

类的组合

**设计类：**

Employee类：

class Employee

{

private:

string name;

string street;

string city;

string zip;

public:

Employee(string name1, string street1, string city1,string zip1);

void change\_name(string name1);

void display();

};

Name类：

class Name

{

protected:

string xing;

string ming;

public:

void Display();

Name(string xing1, string ming1);

};

主函数：int main()

{

Employee a[5] = {

{"黎明", "翻斗大街", "洛杉矶", "063000"},

{"刘德华","天安门","北京市","100000"},

{"郭富城","朝阳区","北京市","100000"},

{"张学友","浦东","上海市","200135"},

{"张益达","静安区","上海市","200040"},

};

a[2].change\_name("周润发");

for (int i = 0; i<5; i++)

{

a[i].display();

}}

int main()

{

Employee x("黎","明", "翻斗大街", "洛杉矶", "063000");

x.display();

}

完整代码：

1. #include <iostream>

#include <string>

using namespace std;

class Employee

{

private:

string name;

string street;

string city;

string zip;

public:

Employee(string name1, string street1, string city1, string zip1);

void change\_name(string name1);

void display();

};

Employee::Employee(string name1, string street1, string city1, string zip1)

{

name = name1;

street = street1;

city = city1;

zip = zip1;

}

void Employee::change\_name(string name1)

{

name = name1;

}

void Employee::display()

{

cout << name << " " << street << " " << city << " " << zip << endl;

}

int main()

{

Employee a[5] = {

{"黎明", "翻斗大街", "洛杉矶", "063000"},

{"刘德华","天安门","北京市","100000"},

{"郭富城","朝阳区","北京市","100000"},

{"张学友","浦东","上海市","200135"},

{"张益达","静安区","上海市","200040"},

};

a[2].change\_name("周润发");

for (int i = 0; i < 5; i++)

{

a[i].display();

}

}

2#include <iostream>

#include <string>

using namespace std;

class Name

{

protected:

string xing;

string ming;

public:

void Display();

Name(string xing1, string ming1);

};

Name::Name(string xing1, string ming1) :xing(xing1), ming(ming1)

{

}

void Name::Display()

{

cout << xing << ming;

}

class Employee

{

private:

Name name;

string street;

string city;

string zip;

public:

Employee(string xing1, string ming1, string street1, string city1, string zip1);

void display();

};

Employee::Employee(string xing1, string ming1, string street1, string city1, string zip1) :name(xing1, ming1)

{

street = street1;

city = city1;

zip = zip1;

}

void Employee::display()

{

name.Display();

cout << street << " " << city << " " << zip << endl;

}

int main()

{

Employee x("黎", "明", "翻斗大街", "洛杉矶", "063000");

x.display();

}

**项目五：**

1、设计一个用于人员管理的People(人员)类。考虑到通用性，这里只抽象出所有类型人员都具有的属性：number(编号)、sex(性别)、birthday(出生日期)、id(身份证号)等。其中“出生日期”声明为一个“日期”类内嵌子对象。用成员函数实现对人员信息的录入和显示。要求包括：构造函数和析构函数、拷贝构造函数、内联成员函数、类的组合。

**考核知识点：**

内嵌子对象、类的基本成员函数的定义；

设计类：

Date类:私有成员：year，month,day,成员函数：Date(int y, int m, int d)，set(),display()

Person类：私有成员num,sex,ID,子对象birthday;成员函数：Person(int n, int y, int m, int d, char id[18], char s = 'm'），Person(const Person& p)，input（），output(),~Person

1. 主函数

定义Person类的对象，并调用成员函数

完整代码：

#include <iostream>

#include <string.h>

#pragma warning(disable : 4996)

using namespace std;

class Date

{

private:

int year;

int month;

int day;

public:

Date() {}

Date(int y, int m, int d) {

year = y;

month = m;

day = d;

}

void set(int y, int m, int d)

{

year = y;

month = m;

day = d;

}

void display()

{

cout << year << "年" << month << "月" << day << "日" << endl;

}

};

class Person

{

private:

int num;

char sex;

Date birthday;

char ID[18];

public:

Person() {}

Person(int n, int y, int m, int d, char id[18], char s = 'm') :birthday(y, m, d)

{

num = n;

sex = s;

strcpy(ID, id);

}

Person(Person& p)

{

num = p.num;

sex = p.sex;

birthday = p.birthday;

strcpy(ID, p.ID);

}

void input()

{

cout << "录入数据：" << endl;

cout << "编号：";

cin >> num;

cout << "性别：";

cin >> sex;

cout << "生日：";

int y, m, d;

cin >> y >> m >> d;

birthday.set(y, m, d);

cout << "身份证号：";

cin >> ID;

ID[18] = '\0';

cout << endl;

}

void output()

{

cout << "编号：" << num << endl;

cout << "性别：" << sex << endl;

cout << "生日：";

birthday.display();

cout << endl;

cout << "身份证号：" << ID << endl;

}

~Person()

{

cout << "录入成功，调用析构函数" << endl;

}

};

int main()

{

Person p1;

p1.input();

p1.output();

return 0;

}

**项目六：**

1、使用C++的类建立一个简单的玩具销售程序。类内必须具有玩具编号、单价、销售量、种类（每类玩具使用一个编号）等属性，并为该类建立一些必要的成员函数。在主程序中建立若干个玩具类对象，用户可以选择购买玩具，不仅显示每种玩具的销售金额，如果销售金额大于某一值（自己设定），显示“XX玩具畅销”字样。并且计算出最畅销的玩具并输出此玩具基本信息。

2、编写一个有关股票的程序，其中有两个类：一个是深圳类Shen\_Stock，另一个是上海类Shang\_Stock。类中有3项私有数据成员：普通股票个数general、ST股票个数st、PT股票个数pt，每一个类分别有自己的友元函数来计算并显示深圳或上海的股票总数（3项的和）。两个类还共用一个count()，用来计算深圳和上海总共有多少股票并输出。

1.

考核知识点：

友元函数

设计类：

ToySell类：

class ToySell

{

public:

ToySell(int b, int p, int a,string type1);

void show();

void show1();

int compare();

private:

int bianhao;

int price;

int amount;

string type;

};

主函数：

int main()

{

int b;

ToySell toy[5] = {

{ 1, 30, 200, "玩偶" },

{ 2, 20, 150, "机器人" },

{ 3, 50, 120, "赛车" },

{ 4, 60, 120, "动漫人物" },

{ 5, 70, 100, "音乐盒" }

};

cout << "请问您是否要购买玩具？" << endl;

cout << "yes:购买 no：不买" << endl;

string daan;

cin >> daan;

if (daan == "no")

{

cout << "再见！" << endl;

return 0;

}

else

{

for (int i = 0; i < 5; i++)

{

toy[i].show();

b = toy[i].compare();

}

toy[b].show1();

cout << "请输入您要购买的玩具编号：";

cin >> b;

cout << "下单成功！请稍等，正在出据发票，请凭发票到柜台缴费！" << endl;

cout << "欢迎下次使用！" << endl;

return 0;

}

**完整代码：**

#include<iostream>

using namespace std;

int temp = 0, a = 1;

class ToySell

{

public:

ToySell(int b, int p, int a, string type1);

void show();

void show1();

int compare();

private:

int bianhao;

int price;

int amount;

string type;

};

ToySell::ToySell(int b, int p, int a, string type1) :bianhao(b), price(p), amount(a), type(type1)

{}

void ToySell::show()

{

cout << bianhao << "号玩具 销售金额：" << price \* amount << " 元 ";

if (price \* amount >6999)

cout << bianhao << "号玩具畅销";

cout << endl;

}

int ToySell::compare()

{

if (price \* amount > temp)

{

temp = price \* amount;

a++;

}

return a;

}

void ToySell::show1()

{

cout << "最畅销玩具：" << bianhao << "号 单价：" << price << " 销售量：" << amount << " 类型：" << type << endl;

}

int main()

{

int b;

ToySell toy[5] = {

{ 1, 30, 200, "玩偶" },

{ 2, 20, 150, "机器人" },

{ 3, 50, 120, "赛车" },

{ 4, 60, 120, "动漫人物" },

{ 5, 70, 100, "音乐盒" }

};

cout << "请问您是否要购买玩具？" << endl;

cout << "yes:购买 no：不买" << endl;

string daan;

cin >> daan;

if (daan == "no")

{

cout << "再见！" << endl;

return 0;

}

else

{

for (int i = 0; i < 5; i++)

{

toy[i].show();

b = toy[i].compare();

}

toy[b].show1();

cout << "请输入您要购买的玩具编号：";

cin >> b;

cout << "下单成功！请稍等，正在出据发票，请凭发票到柜台缴费！" << endl;

cout << "欢迎下次使用！" << endl;

return 0;

}

2.

设计类：

深圳类：

class shen\_stock

{

public:

shen\_stock(int g, int s, int p);

~shen\_stock();

int sum();

friend void count(shen\_stock& a,shang\_stock& b);

private:

int general, st, pt;

};

上海类：

class shang\_stock

{

public:

shang\_stock(int g, int s, int p);

~shang\_stock();

int sum();

friend void count(shen\_stock& a,shang\_stock& b);

private:

int general, st, pt;

};

主函数：

int main()

{

shen\_stock shenzhen(111, 45, 67);

shang\_stock shanghai(222, 54, 88);

shenzhen.sum();

shanghai.sum();

count(shenzhen, shanghai);

return 0;

完整代码：

#include<iostream>

using namespace std;

class shang\_stock;

class shen\_stock

{

public:

shen\_stock(int g, int s, int p);

~shen\_stock();

int sum();

friend void count(shen\_stock& a,shang\_stock& b);

private:

int general, st, pt;

};

shen\_stock::shen\_stock(int g, int s, int p) {

general = g; st = s; pt = p;

}

int shen\_stock::sum()

{

int y = general + st + pt;

cout << "深圳股票总数：" << y << endl;

return y;

}

shen\_stock::~shen\_stock()

{

}

class shang\_stock

{

public:

shang\_stock(int g, int s, int p);

~shang\_stock();

int sum();

friend void count(shen\_stock& a,shang\_stock& b);

private:

int general, st, pt;

};

shang\_stock::shang\_stock(int g, int s, int p) {

general = g; st = s; pt = p;

}

int shang\_stock::sum()

{

int y = general + st + pt;

cout << "上海股票总数：" << y << endl;

return y;

}

shang\_stock::~shang\_stock()

{

}

void count(shen\_stock& a,shang\_stock& b)

{

int y=a.general+a.st+a.pt+b.general+b.st+b.pt;

cout<<"深圳和上海股票总数为："<<y<<endl;

}

int main()

{

shen\_stock shenzhen(111, 45, 67);

shang\_stock shanghai(222, 54, 88);

shenzhen.sum();

shanghai.sum();

count(shenzhen, shanghai);

return 0;

}

**总结：对友元函数的理解进一步加深。**