# 12.ex1 lab4多态参考代码

#include<iostream>

#include<string>

#include <windows.h>

using namespace std;

const unsigned int days[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };//the days of month

int getmonth()

{//get the month of earth

SYSTEMTIME system;

GetLocalTime(&system);

return system.wMonth;

}

class Date

{

friend std::ostream &operator<<(ostream &, const Date &);

public:

Date(int m = 1, int d = 1, int y = 1900)

{

setDate(m, d, y);

}//default constructor

void setDate(int mm, int dd, int yy)

{

if (mm >= 1 && mm <= 12)

month = mm;

else

throw invalid\_argument("Month must be 1-12");//throw the error

if (yy >= 1900 && yy <= 2100)

year = yy;

else

throw invalid\_argument("Year must be>=1900 and<-2100");//throw the error

// test for a leap year

if ((month == 2 && leapYear(year) && dd >= 1 && dd <= 29) || (dd >= 1 && dd <= days[month]))

day = dd;

else

throw invalid\_argument("Day is out of range for current month and year");//throw the error

}// set month,day,year

Date &operator++()

{

helpIncrement();// increment date

return \*this;// reference return to create an lvalue

}//prefix increment operator

Date operator++(int)

{

Date temp = \*this;//hold current state of object

helpIncrement();//postfix increment operator

return temp;

}

Date &operator+=(unsigned int additiona1Days)

{

for (int i = 0; 1 < additiona1Days;++i)

helpIncrement();// add days,modify object

return \*this;

}

static bool leapYear(int testYear)

{

if (testYear % 400 == 0 || (testYear % 100 != 0 && testYear % 4 == 0))

return true; // a leap year

else

return false; // not a leap year

}// is date in a leap year?

bool endofMonth(int testDay)const

{

if (month == 2 && leapYear(year))

return testDay == 29;// last day of Feb. in leap year

else

return testDay == days[month];

}// is date at the end of month?

void helpIncrement()

{

//day is not end of month

if (!endofMonth(day))

++day; // increment day

else

if (month < 12)// day is end of month and month < 12

{

++month;// increment month

day = 1; // first day of new month

}//end if

else // 1ast day of year

{

++year; // increment year

month = 1; // first month of new year

day = 1; // first day of new month

}

}

int getmonthdate()const

{

return month;

}

virtual void print()const

{

cout << month << "-" << day <<"-"<< year ;

}

private:

unsigned int month;

unsigned int day;

unsigned int year;

};

class Employee:public Date

{

public:

Employee(const string &first, const string &last, const string &ssn, int m = 1, int d = 1, int y = 1900)

:firstName(first), lastName(last), socialSecurityNumber(ssn)

{

setDate(m, d, y);

}

virtual~Employee() {}// virtual destructor

void setFirstName(const string &first)

{

firstName = first;

}// set first name

string getFirstName() const

{

return firstName;

}// return first name

void setLastName(const string&Last)

{

lastName = Last;

}// set last name

string getLastName()const

{

return lastName;

}// return last name

void setSocialSecurityNumber(const string &ssn)

{

socialSecurityNumber = ssn;// should validate

}// set SSN

string getSocialSecurityNumber() const

{

return socialSecurityNumber;

}// return SSN

// pure virtual function makes Employee an abstract base class

virtual double earnings() const = 0;//pure virtual

virtual void print() const

{

cout << getFirstName() << ' ' << getLastName() << "\nsocial security number : " << getSocialSecurityNumber();

}//virtual

private:

string firstName;

string lastName;

string socialSecurityNumber;

};

class SalariedEmployee : public Employee

{

public:

SalariedEmployee(const string &first, const string &last, const string &ssn, double salary, int m = 1, int d = 1, int y = 1900) :Employee(first, last, ssn)

{

setWeeklySalary(salary);

setDate(m, d, y);

}// end SalariedEmployee constructor

void setWeeklySalary(double salary)

{

if (salary >= 0.0)

weeklySalary = salary;

else

throw invalid\_argument("Weekly salary must be>=0.0");

}// end function setWeeklySalary

// return salary

double getWeeklySalary()const

{

return weeklySalary;

}// end function getweeklySalary

// calculate earnings;

virtual double earnings()const override

{

if (getmonth() == getmonthdate())

{

cout << endl << "this month is his birthday,earning+100$" ;

return getWeeklySalary()+100;

}

return getWeeklySalary();

}// calculate earnings

virtual void print() const override

{

cout << "salaried employee:";

Employee::print();// reuse abstract base-class print function

cout <<endl<< "birthday:";

Date::print();

cout << "\nweekly salary:" << getWeeklySalary();

}// print object

private:

double weeklySalary;// salary per week

};// end class SalariedEmployee

class CommissionEmployee :public Employee

{

public:

CommissionEmployee(const string &first,const string&last, const string &ssn, double sales=0.0, double rate=0.0, int m = 1, int d = 1, int y = 1900)

: Employee(first, last, ssn)

{

setGrossSales(sales);

setCommissionRate(rate);

setDate(m, d,y);

}

virtual ~CommissionEmployee() {}// virtual destructor

void setGrossSales(double sales)

{

if(sales>=0.0)

grossSales = sales;

else

throw invalid\_argument("Gross sales must be>=0.0");

} // set commission rate

double getGrossSales()const

{

return grossSales;

}// return commission rate

void setCommissionRate(double rate)

{

if (rate > 0.0&& rate < 1.0)

commissionRate = rate;

else

throw invalid\_argument("Commission rate must be>0.0 and <1.0");

}// set gross sales amount

double getCommissionRate()const

{

return commissionRate;

}// return gross sales amount

virtual double earnings() const override

{

if (getmonth() == getmonthdate())

{

cout << endl<< "this month is his birthday,earning+100$" ;

return getCommissionRate()\*getGrossSales() + 100;

}

return getCommissionRate()\*getGrossSales();

}// calculate earnings

virtual void print() const override

{

cout << "commission employee:";

Employee::print();// code reuse

cout <<endl<< "birthday:";

Date::print();

cout << "\ngross sales:" << getGrossSales()<<":commission rate : "<<getCommissionRate();

}/// print object

private:

double grossSales; // gross weekly sales

double commissionRate;// commission percentage

};// end class CommissionEmployee

class BasePlusCommissionEmployee :public CommissionEmployee

{

public:

BasePlusCommissionEmployee(const string&first, const string &last, const string &ssn,double sales=0.0, double rate=0.0, double salary=0.0, int m = 1, int d = 1, int y = 1900)

:CommissionEmployee(first, last, ssn, sales, rate)

{

setBaseSalary(salary);// validate and store base salary

setDate(m, d, y);

}

virtual ~BasePlusCommissionEmployee() {}//virtual destructor

void setBaseSalary(double salary)

{

if (salary >= 0.0)

BaseSalary = salary;

else

throw invalid\_argument("Salary must be>=0.0");

}// set base salary

double getBaseSalary()const

{

return BaseSalary;

}// return base salary

// keyword virtual signals intent to override

virtual double earnings() const override

{

if (getmonth() == getmonthdate())

cout << endl << "this month is his birthday,earning+100$" ;

return getBaseSalary() + CommissionEmployee::earnings();

}

virtual void print()const override

{

cout << "base-salaried";

CommissionEmployee::print();// code reuse

cout << ";base salary:" << getBaseSalary();

}// print object

private:

double BaseSalary;// base salary per week

};

ostream &operator<<(ostream &output, const Date &d)

{

static string monthName[13] = { "", "January", "February","March", "April", "May", "June", "July", "August","September", "October", "November", "December" };

output << monthName[d.month] << ' ' << d.day << "," << d.year;

return output;// enables cascading

}

#include <iomanip>

#include <vector>

void virtualViaPointer(const Employee \*const baseClassPtr)

{

baseClassPtr->print();

cout << "\nearned $" << baseClassPtr->earnings() << "\n\n";

}

void virtualviaReference(const Employee &baseclassRef)

{

baseclassRef.print();

cout << "\nearned $" << baseclassRef.earnings() << "\n\n";

}

int main()

{

cout << fixed << setprecision(2);

SalariedEmployee salariedEmployee("John", "Smith", "111 - 11 - 1111", 800,12,4,2000);

CommissionEmployee commissionEmployee("Sue", "Jones", "333 - 33 - 3333", 10000, 0.06,3,1,1986);

BasePlusCommissionEmployee basePlusCommissionEmployee("Bob", "Lewis", "444 - 44 - 4444", 5000, .04, 300,2,12,2012);

cout << "Employees processed individually using static binding:\n\n";

salariedEmployee.print();

cout << "\nearned $" << salariedEmployee.earnings() << "\n\n";

commissionEmployee.print();

cout << "\nearned $" << commissionEmployee.earnings() << "n\n";

basePlusCommissionEmployee.print();

cout << "\nearned $" << basePlusCommissionEmployee.earnings() << "\n\n"; //create vector of three base - class pointers

vector< Employee \*> employees(3);

// initialize vector with pointers to Employees

employees[0] = &salariedEmployee;

employees[1] = &commissionEmployee;

employees[2] = &basePlusCommissionEmployee;

cout << "Enployees processed polymorphically via dynanic binding:\n\n";

//call virtualviapointer to print each Employee's information

cout << "Virtual function calls sade off base - class pointers :\n\n";

for (const Employee \*employeePtr : employees)

virtualViaPointer(employeePtr);

cout << "Virtual function calls made off base - class references :n\n";

for (const Employee \*employeePtr : employees)

virtualviaReference(\*employeePtr);// note dereferencing

}

# 12.ex2 lab4多态参考代码

#include<iostream>

#include<string>

#include<vector>

#define pi 3.14159

using namespace std;

class shape//最初的基类

{

public:

shape(string a = "not named") :id(a)//初始化器

{ }

virtual ~shape()//虚析构

{

cout << "the shape destructor have been used" << endl;

}

string getid()

{//返回图形名

return id;

}

void setid(string a)

{//设置图形名

id = a;

}

virtual void print() = 0;//纯虚函数

virtual double getarea() = 0;//纯虚函数

private:

string id;

};

class TwoDimensionalShape: public shape//二维派生类

{

public:

TwoDimensionalShape(string a = "not named TwoDimensionalShape") :shape(a)//船新的名字

{ }

virtual ~TwoDimensionalShape() //船新的虚析构

{

cout << "the TwoDimensionalShape destructor have been used" << endl;

}

};

class triangle :public TwoDimensionalShape//二维-三角形

{

public:

triangle(double x1 = 0.0, double x2 = 1.0, double x3 = 0.0, double y1 = 0.0, double y2 = 0.0, double y3 = 1.0, string a = "not named triangle") :

p1x(x1), p2x(x2), p3x(x3), p1y(y1), p2y(y2), p3y(y3), TwoDimensionalShape(a)//三个顶点+名字

{

}

virtual ~triangle()//三角形虚析构

{

cout << "the triangle destructor have been used" << endl;

}

void settriangle(double x1 = 0.0, double x2 = 1.0, double x3 = 0.0, double y1 = 0.0, double y2 = 0.0, double y3 = 1.0)

{//设置三角形三顶点

p1x = x1;

p2x = x2;

p3x = x3;

p1y = y1;

p2y = y2;

p3y = y3;

}

virtual double getarea()

{//计算面积

return fabs(p1x\*p2y + p2x \* p3y + p3x \* p1y - p1y \* p2x - p2y \* p3x - p3y \* p1x) / 2.0;

}

virtual void print()

{//重写，打印三角形的各项参数

cout << getid() << "(triangle): point1(" << p1x << "," << p1y << ");point2:(" << p2x << "," << p2y << ");point3:(" << p3x << "," << p3y << ");area:" << getarea() << endl;

}

private:

double p1x, p2x, p3x;

double p1y, p2y, p3y;

};

class circle :public TwoDimensionalShape//二维-圆

{

public:

circle(double ox = 0.0, double oy = 0.0, double a = 1.0, string b = "not named circle") :

pox(ox), poy(oy), r(a), TwoDimensionalShape(b)//分别需要一个圆心+半径即可确定

{

}

virtual ~circle()

{//圆的析构函数

cout << "the circle destructor have been used" << endl;

}

void setcircle(double ox = 0.0, double oy = 0.0, double a = 1.0)

{//设置圆心与半径

pox = ox;

poy = oy;

r = a;

}

virtual double getarea()//圆面积

{

return pi \* r\*r;

}

virtual void print()//输出圆的各项信息

{

cout << getid() << "(circle): point(" << pox << "," << poy << ");banjing:" << r << "area:" << getarea() << endl;

}

private:

double pox, poy;

double r;

};

class rectangle : public TwoDimensionalShape//二维-矩形

{

public:

rectangle(double x1 = 0.0, double x2 = 1.0, double y1 = 0.0, double y2 = 1.0, string a = "not named rectangle") :

ldx(x1), rux(x2), ldy(y1), ruy(y2), TwoDimensionalShape(a)//左下顶点+右上定点+名字

{

}

virtual ~rectangle()

{//长方形虚析构

cout << "the rectangle destructor have been used" << endl;

}

void setrectangle(double x1 = 0.0, double x2 = 1.0, double y1 = 0.0, double y2 = 1.0, string a = "not named")

{//设置参数

ldx = x1;

rux = x2;

ldy = y1;

ruy = y2;

setid(a);

}

virtual double getarea()

{//长承宽

return fabs((ldx - rux)\*(ldy - ruy));

}

virtual void print()

{//打印各项参数

cout << getid() << "(rectangle): leftdownpoint(" << ldx << "," << ldy << ");rightuppoint:(" << rux << "," << ruy << ");area:" << getarea() << endl;

}

protected:

double ldx, ldy;

double rux, ruy;

};

class square :public rectangle

{

public:

square(double x1 = 0.0, double x2 = 1.0, double y1 = 0.0, double y2 = 1.0, string a = "not named square") :

rectangle(x1, x2, y1, y2, a), bianchang(fabs(x1 - x2))//只是多了一个边长

{

}

virtual ~square()

{ //正方形析构函数

cout << "the square destructor have been used" << endl;

}

void setsquare(double x1 = 0.0, double x2 = 1.0, double y1 = 0.0, double y2 = 1.0, string a = "not named")

{//初始化各项参数

setrectangle(x1, x2, y1, y2, a);

bianchang = fabs(x1 - x2);

}

circle incircle()

{//计算一个内切圆（对角线中点）

circle c((ldx + rux) / 2, (ldy + ruy) / 2, bianchang / 2, "incircle");

return c;

}

virtual double getarea()

{

return bianchang \* bianchang;//直接边长的平方

}

virtual void print()//打印本参数与内切圆参数

{

cout << getid() << "(square): leftdownpoint(" << ldx << "," << ldy << ");rightuppoint:(" << rux << "," << ruy << ")area:" << getarea() << endl;

incircle().print();

}

private:

double bianchang;

};

class ThreeDimensionalShape : public shape//三维类型

{

public:

ThreeDimensionalShape(string a = "not named ThreeDimensionalShape") :shape(a)

{ }

virtual ~ThreeDimensionalShape()

{

cout << "the ThreeDimensionalShape destructor have been used" << endl;

}

virtual double getvolume() = 0;

};

class sphere :public ThreeDimensionalShape//三维-球

{

public:

sphere(double ox = 0.0, double oy = 0.0, double oz = 0.0, double a = 1.0, string b = "not named sphere") :x(ox),y(oy),z(oz),r(a)

{//一个球心+半径

}

virtual ~sphere()//球的虚析构

{

cout << "the sphere destructor have been used" << endl;

}

virtual void print()//打印函数

{

cout << getid() << "(sphere): point(" << x << "," << y <<" ,"<<z<<")banjing:" << r << "area:" << getarea() <<"volume:"<<getvolume()<< endl;

}

virtual double getarea()//对应的是表面积

{

return 4.0\*pi \* r\*r;

}

virtual double getvolume()//对应体积

{

return 4.0\*pi \* r\*r\*r/3.0;

}

private:

double x, y, z, r;

};

class cube:public ThreeDimensionalShape//三维-正方体

{

public:

cube(double x1 = 0.0, double x2 = 1.0, double y1 = 0.0, double y2 = 1.0, double z1 = 0.0, double z2 = 1.0, string a = "not named cube") :

ldx(x1), rux(x2), ldy(y1), ruy(y2),ldz(z1),ruz(z2), ThreeDimensionalShape(a)//对角两个顶点

{

}

virtual ~cube()

{//正方体虚析构

cout << "the cube destructor have been used" << endl;

}

void setrectangle(double x1 = 0.0, double x2 = 1.0, double y1 = 0.0, double y2 = 1.0, double z1 = 0.0, double z2 = 1.0,string a = "not named")

{//植入初始数据

ldx = x1;

rux = x2;

ldy = y1;

ruy = y2;

ldz = z1;

ruz = z2;

setid(a);

}

virtual double getarea()

{//对应表面积

return 2\*(fabs((ldx - rux)\*(ldy - ruy))+ fabs((ldx - rux)\*(ldz - ruz))+ fabs((ldz - ruz)\*(ldy - ruy)));

}

virtual void print()

{//对应打印各项函数

cout << getid() << "(cube): leftdownpoint(" << ldx << "," << ldy <<","<<ldz<< ");rightuppoint:(" << rux << "," << ruy <<","<<ruz <<");area:" << getarea() <<"volume:" << getvolume() << endl;

}

virtual double getvolume()

{//对应面积

return fabs((ldx - rux)\*(ldy - ruy)\*(ldz - ruz));

}

protected:

double ldx, ldy,ldz;

double rux, ruy,ruz;

};

int main()

{//测试函数

vector<shape\*>shap(6);

shap[0] = new triangle;

shap[1] = new circle;

shap[2] = new rectangle ;

shap[3] = new square;

shap[4] = new cube ;

shap[5] = new sphere ;

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

{

shap[i]->print();

delete shap[i];

}

}

# 12.ex3 lab4多态参考代码

#include<iostream>

#include<iostream>

#include<vector>

using namespace std;

class account//账户类型

{

public:

account(const double mon=0.0)

{//通过函数设置存款

setdebt(mon);

}

void setdebt(const double mon)

{//设置存款

if (mon >= 0)

debt = mon;//正常存

else

{

cerr << "debt must be >=0,set it zero"<<endl;//不能存负数

debt = 0.0;//自动设置为0.0

}

}

double getdebt()const

{//返回现有钱数

return debt;

}

virtual void debit(const double get)//可重写

{//取钱打印

if (get <= debt && get >= 0)//正常取钱且能去出来

{

cout << "you get " << get << "&" << endl;//我取出来xx钱

debt -= get;//减去钱数

}

else

cerr << "debit amount exceed account balance" << endl;//返回错误

}

virtual void credit(const double save)//可重写

{//存钱

cout << "you save " << save << "$"<<endl;//我存了多少钱

debt += save;//计算存款

}

virtual void print()//可重写

{

cout << "this account is a normal account" << endl;//打印这个类型的类型

}

void getbalance()const

{

cout << "now the debit own" << debt << "$" << endl;//打印输出当前的钱数

}

private:

double debt;

};

class savingaccount :public account//对账户类的更新（储户类）

{

public:

savingaccount(double bou=0.05, double deb=0)//加入了初始利率

{

setsavingaccount(bou, deb);//调用设置函数

}

void setsavingaccount(double bou,double deb)

{

setdebt(deb);//调用基类的set函数

if(bou>=0)//正常的利率

bouns = bou;//直接植入

else

{

cerr << "bouns must be larger than 0"<<endl;//利率不能小于0

bouns = 0;//无利率

}

}

double calculateinterest()

{//计算利率

return getdebt()\*(1.0+bouns);

}

virtual void credit(const double save)//重写过的存款函数

{

cout << "you save " << save << "$" << endl;//我存了xx钱

setdebt( getdebt()+save);//重设账户总钱数

setdebt(calculateinterest());//计算利率

}

virtual void print()//重写过的打印函数

{

cout << "this account is a saving account" << endl;//输出该账户的对应类型

}

private:

double bouns;

};

class checkingaccount :public account//支票类储户

{

public:

checkingaccount(double deb = 0.0, double cha = 0.0)

{

setcheckingaccount(deb, cha);//调用set函数

}

void setcheckingaccount(double deb = 0.0, double cha = 0.0)

{

if (cha >= 0)//手续费》0

charge = cha;//正常存入

else

{

cerr << "charge must be larger than 0"<<endl;//抛出error

charge = 0.0;//自动置为0

}

setdebt(deb);//调用基类set

}

virtual void debit(const double get)//重写的取款函数

{

if (get <= getdebt() && get >= 0)//能取出来（不计手续费）

{

cout << "you get " << get << "&" << endl;//打印

setdebt(getdebt() - get - charge);//重写 存款

}

else//抛出错误

cerr << "debit amount exceed account balance"<<endl;

}

virtual void credit(const double save)//重写的存款函数

{

cout << "you save " << save << "$" << endl;//打印我的存款函数

setdebt(getdebt() + save - charge);//重写存款

}

virtual void print()//重写打印函数

{

cout << "this account is a checking account" << endl;//这是一个支票函数

}

void getbalance()const//现在这里面有多少

{

cout << "now the debit own" << getdebt() << "$" << endl;

}

private:

double charge;

};

int main()

{

vector<account\*>acc(3);//测试函数

acc[0] = new account(100.0);

acc[1] = new savingaccount(0.05, 120.0);

acc[2] = new checkingaccount(120000.0, 5.0);

for (int i = 0;i <= 2;i++)

{

acc[i]->print();

acc[i]->credit(10.0);

acc[i]->debit(1.0);

acc[i]->getbalance();

}

}

LAB8 函数模板

1. #include<iostream>

2. #include<string>

3. using namespace std;

4. template<class ElementType,int numberOfElements>

5. class Array

6. {

7. public:

8. ElementType array1[numberOfElements];

9. Array()

10. {

11. cout << "Enter " << numberOfElements << " interge values" << endl;

12. for (int i = 0;i < numberOfElements;i++)

13. {

14. cin >> array1[i];

15. }

16. }

17. void show()

18. {

19. cout << "The value in array are:" << endl;

20. for (int i = 0;i < numberOfElements;i++)

21. {

22. cout << array1[i] << " ";

23. }

24. }

25. };

26. int main()

27. {

28. Array<int, 5> a1;

29. a1.show();

30. Array<string, 7>a2;

31. a2.show();

32. return 0;

33. }

#include<iostream>

#include<string>

using namespace std;

template<class T>

bool IsEqualTo(T &a, T &b)

{

if (a == b)

return true;

else

return false;

}

int main()

{

int a, b;

cin >> a >> b;

cout << IsEqualTo(a, b) << endl;

string c, d;

cin >> c >> d;

cout << IsEqualTo(c, d) << endl;

return 0;

}

#include<iostream>

#include<string>

using namespace std;

class Complex

{

public:

int a;

int b;

Complex(int k1, int k2)

{

a = k1;

b = k2;

}

};

template<class T>

bool IsEqualTo(T &a, T &b)

{

if (a == b)

return true;

else

return false;

}

int main()

{

Complex a1(1, 2);

Complex a2(2, 3);

cout << IsEqualTo(a1, a2) << endl;

return 0;

}

#include<iostream>

#include<string>

using namespace std;

class Complex

{

public:

int a;

int b;

Complex(int k1, int k2)

{

a = k1;

b = k2;

}

bool operator==(Complex& temp)

{

if (this->a == temp.a && this->b == temp.b)

return true;

else

return false;

}

};

template<class T>

bool IsEqualTo(T &a, T &b)

{

if (a == b)

return true;

else

return false;

}

int main()

{

Complex a1(1, 2);

Complex a2(2, 3);

cout << IsEqualTo(a1, a2) << endl;

Complex a3(2, 3);

cout << IsEqualTo(a3, a2) << endl;

return 0;

}

#include<iostream>

#include<string>

using namespace std;

template<class T>

class Vector

{

public:

T array[5];

};

template<class T>

ostream& operator<<(ostream& output, const Vector<T>& temp)

{

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

output<< temp.array[i] << " ";

output << endl;

return output;

}

template<class T>

istream& operator>>(istream& input,Vector<T>& temp)

{

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

input >> temp.array[i];

return input;

}

int main()

{

Vector<int>v1;

cin >> v1;

cout << v1;

return 0;

}

Lab3构造与析构

EX1

(1)

…BaseClass1 Object is created! //对象a的BaseClass1类构造函数

…BaseClass1 Object is created! //对象b的BaseClass1基类类构造函数

…First layer derived Object is created! //对象b的Myderivrd1派生类构造函数

…BaseClass1 Object is created! //对象c的BaseClass1基类的基类构造函数

…First layer derived Object is created! //对象c的Myderivrd1基类构造函数

…Second layer derived Object is created! //对象c的Myderivrd11派生类构造函数

…Second layer derived Object is destroyed! //对象c的Myderivrd11派生类析构函数

…First layer derived Object is Destroyed! //对象c的Myderivrd1基类析构函数

…BaseClass1 Object is destroyed! //对象c的BaseClass1基类的基类析构函数

…First layer derived Object is Destroyed! //对象b的Myderivrd1派生类析构函数

…BaseClass1 Object is destroyed! //对象b的BaseClass1基类析构函数

…BaseClass1 Object is destroyed! //对象a的BaseClass1类析构函数

(2)

…BaseClass1 Object is created! //对象a的BaseClass2类private成员a1的构造函数

…BaseClass2 Object is created! //对象a的BaseClass2类构造函数

…BaseClass1 Object is created! //对象b的BaseClass2基类private成员的构造函数

…BaseClass2 Object is created! //对象b的BaseClass2基类构造函数

…BaseClass1 Object is created! //对象b的Myderived2派生类中private成员a1的构造函数

…First layer derived Object is created! //对象b的Myderived2派生类的构造函数

…BaseClass1 Object is created! //对象c的BaseClass2基类的基类private成员的构造函数

…BaseClass2 Object is created! //对象c的BaseClass2基类的基类构造函数

…BaseClass1 Object is created! //对象c的Myderived2基类类中private成员a1的构造函数

…First layer derived Object is created! //对象c的Myderived2基类的构造函数

…Second layer derived Object is created! //对象c的Myderived21派生类的构造函数

…Second layer derived Object is destroyed! //对象c的Myderived21派生类的析构函数

…First layer derived Object is Destroyed! //对象c的Myderived2基类的析构函数

…BaseClass1 Object is destroyed! //对象c的Myderived2基类类中private成员a1的析构函数

…BaseClass2 Object is destroyed! //对象c的BaseClass2基类的基类析构函数

…BaseClass1 Object is destroyed! //对象c的BaseClass2基类的基类private成员的析构函数

…First layer derived Object is Destroyed! //对象b的Myderived2派生类的析构函数

…BaseClass1 Object is destroyed! //对象b的Myderived2派生类中private成员a1的析构函数

…BaseClass2 Object is destroyed! //对象b的BaseClass2基类析构函数

…BaseClass1 Object is destroyed! //对象b的BaseClass2基类private成员的析构函数

…BaseClass2 Object is destroyed! //对象a的BaseClass2类析构函数

…BaseClass1 Object is destroyed! //对象a的BaseClass2类private成员a1的析构函数

在创建派生类对象时会依次自上而下调用从基类到派生类的构造函数，销毁时逆序调用析构函数。若类中存在其他类的对象成员，则先调用该成员对应类的构造函数

EX2

(1)

…BaseClass31 Object is created! //对象a的类BaseClass31的构造函数

1 2 3 //对象a中成员a,b,c赋值后的输出

…BaseClass32 Object is created! //对象b的类BaseClass32的构造函数

-858993460 -858993460 -858993460 //对象b的成员x,y,z的初始值，但没有人为设置初始值

4 5 6 //对象b中成员x,y,z赋值之后的输出

…BaseClass32 Object is destroyed! //对象b的BaseClass32类析构函数

…BaseClass31 Object is destroyed! //对象a的BaseClass31类析构函数

(2)

Correct:

#include <iostream>

using namespace std;

class MyBase31 {

int a, b, c;

public:

MyBase31(int x, int y, int z) :a(x), b(y), c(z)

{

cout << "…BaseClass31 Object is created!" << endl;

cout << a << " " << b << " " << c << endl;

}

~MyBase31() { cout << "…BaseClass31 Object is destroyed!" << endl; }

};

class MyBase32 {

int x, y, z;

public:

MyBase32(int a, int b, int c)

{

cout << "…BaseClass32 Object is created!" << endl;

cout << x << " " << y << " " << z << endl;

x = a, y = b, z = c;

cout << a << " " << b << " " << c << endl;

}

int getx() { return x; }//用于访问对象xyz的public函数成员

int gety() { return y; }

int getz() { return z; }

~MyBase32() { cout << "…BaseClass32 Object is destroyed!" << endl; }

};

class MyDerived1 : public MyBase31 {

MyBase32 a; //创建属于Mubase32类对象的私有成员

int c; //int私有成员

public:

MyDerived1(int x) : c(x), a(x, 8, 9), MyBase31(x, 8, 9) //构造函数，使用初始化器，使c=x,构造a(x,8,9)和基类构造函数(x,8,9)

{

cout << "…Base Object has been created!" << endl;

cout << "…Member Object has been created! "

<< a.getx() << " " << a.gety() << " " << a.getz() << endl;//使用函数成员返回xyz的值

cout << "…Derived Object is created! " << c << endl;

}

};

int main()

{

MyBase31 a(1, 2, 3);

MyBase32 b(4, 5, 6);

MyDerived1 c(88);

return 0;

}

…BaseClass31 Object is created! //对象a的Mybase31类构造函数

1 2 3 //输出对象a的成员a,b,c赋值后的值

…BaseClass32 Object is created! //对象b的Mybase32类构造函数

-858993460 -858993460 -858993460 //输出对象b的成员x,y,z初始值，但未人为设置初始值

4 5 6 //输出对象b的成员x,y,z赋值后的值

…BaseClass31 Object is created! //对象c基类Mybase31的构造函数

88 8 9 //该Mybase31对象的输出值

…BaseClass32 Object is created! //对象c成员a的Mybase32类构造函数

-858993460 -858993460 -858993460 //该对象a的成员x,y,z的初始值，未人为设置初始值

88 8 9 //该对象a的成员x,y,z赋值后的值输出

…Base Object has been created! //对象c的派生类Myderived1类构造函数

…Member Object has been created! 88 8 9 //输出对象a的x,y,z值

…Derived Object is created! 88 //输出对象c其中成员c的值

…BaseClass32 Object is destroyed! //对象c的成员a的Mybase32类析构函数

…BaseClass31 Object is destroyed! //对象c基类Mybase31的析构函数

…BaseClass32 Object is destroyed!//对象b的Mybase32类析构函数

…BaseClass31 Object is destroyed!//对象a的Mybase31类析构函数

派生类不能直接访问基类的私有成员（题目原代码中的错误），应当使用对应类的成员函数来间接访问其对象。

在初始化时不仅对自身类进行初始化，对基类也进行初始化

EX4

//1)To create a base class as following :

class MyBase {

int x;

public:

MyBase(int a) :x(a) {};

int getX() { cout << " " << endl; return x; }

};

//2)To create a derived class as following :

class MyDerived : public MyBase {

int y;

public:

MyDerived(int a) :y(a), MyBase(a + 4){};

int getY() { cout << " " << endl; return y; }

};

//3)To create a test program as following and analyze the result.

int main()

{

MyBase a(2), \*p = &a; //类指针p应指向a所在的地址

MyDerived b(4), \*q = &b; //类指针q应指向b所在的地址

MyBase &c = a;

MyBase &d = b;

cout << a.getX() << " " << p->getX() << endl;

cout << b.getY() << " " << q->getY() << b.getX() << " " << q->getX() << endl;

a = b;

cout << a.getX() << " " << a.getY() << endl;

//此处，a是基类对象，不能够访问派生类的getY成员函数

p = q;

cout << p->getX() << " " << p->getY() << endl;

//此处，p是指向基类对象的指针，同样不能够访问派生类的getY成员函数

cout << c.getX() << " " << d.getX() << " " << d.getY() << endl;

//d本身是基类对象，虽然引用了派生类b，但并不能访问派生类的成员函数getY

b = a;

//对象不能由基类转化至派生类

cout << b.getX() << " " << b.getY() << endl;

}

类之间的传递只存在同类→同类，派生类→基类，而不存在基类→派生类，

类指针访问类成员时应使用->，引用类时，以引用的对象类型为准而不是被引用的对象

EX5

#include<iostream>

using namespace std;

class Date

{

public:

Date(int a = 2014, int b = 1, int c = 1)//默认构造函数，默认日期2014-1-1

{

year = a;

month = b;

day = c;

}

void date\_print()

{

cout << year << "-" << month << "-" << day;//打印日期 year-month-day

}

void setDate(Date date)//设置日期的函数

{

year = date.year;

month = date.month;

day = date.day;

}

private:

int year, month, day;

};

class FinalTest:public Date

{

public:

FinalTest(const char\* n, Date date)//派生类FinalTest的构造函数

{

name = n;

setDate(date);//调用基类的函数

}

FinalTest(const char\* n=NULL)//重载构造函数

{

name = n;

}

void print()

{

cout << "Title: " << name << endl;

cout << "Test Date: ";

date\_print();//调用基类函数输出日期

cout << endl;

}//打印考试信息

private:

const char\* name;

};

int main()

{

FinalTest item1("C++ Test", Date(2014, 6, 2));

item1.print();

FinalTest item2("Java");

item2.print();

item2.setDate(Date(2014, 6, 10));

item2.print();

return 0;

}

实现派生类和基类之间交流可以活用set,get这类函数，可以避免派生类无法访问基类成员的情况，同时，可以嵌套调用函数来减少代码量，实现代码的高效利用

Ex6

#include<iostream>

#include<cmath>

#include<string>

using namespace std;

//坐标类

class Coordinate

{

friend ostream &operator<<(ostream &output, const Coordinate &a)

{

cout << " ( " << a.x << " , " << a.y << " ) ";

return output;

}

public:

Coordinate(double a = 0, double b = 0)

{

x = a;

y = b;

}

~Coordinate() {}

void setxy(double a, double b)//设置坐标

{

x = a;

y = b;

}

void print() { cout << " ( " << x << " , " << y << " ) "; }//打印坐标格式( x , y )

double getx() { return x; }

double gety() { return y; }

double distance(Coordinate a)

{

return sqrt((x - a.x)\*(x - a.x) + (y - a.y)\*(y - a.y));

}

Coordinate operator\*=(const Coordinate&a)

{

x = a.x;

y = a.y;

}

private:

double x, y;

};

//定义Shape类

class Shape

{

public:

Shape(string name = "") { id = name; }

void setid(string name) { id = name; }

~Shape() {}

void printname() { cout << "object name: " << id << endl; }

private:

string id;

};

//定义派生类Cricle

class Circle :public Shape

{

friend class Square;//声明友元

public:

Circle(Coordinate c = Coordinate(0, 0), string name = "", double rad = 0) :Shape(name), center(c), radius(rad)

{}

~Circle() {}

double area() { return 3.14159 \* radius\*radius; }//pi取近似值3.14159

void print()

{

printname();

cout << "radius: " << radius << endl;

cout << "center: " << center << endl;

cout << "area: " << area() << endl << endl;

}

private:

double radius;

Coordinate center;

};

//定义派生类Triangle

class Triangle :public Shape

{

public:

Triangle(Coordinate a = Coordinate(0, 0), Coordinate b = Coordinate(0, 0), Coordinate c = Coordinate(0, 0), string name = "NoName")

:Shape(name), point1(a), point2(b), point3(c)

{

side1 = point1.distance(point2);

side2 = point1.distance(point3);

side3 = point2.distance(point3);

//计算三边长

}//设置三角形三顶点位置

~Triangle() {}

double area()

{

double p = (side1 + side2 + side3) / 2;

return sqrt(p\*(p - side1)\*(p - side2)\*(p - side3));//Heron公式计算三角形面积

}

void print()

{

printname();

cout << "point 1: " << point1 << endl;

cout << "point 2: " << point2 << endl;

cout << "point 3: " << point3 << endl;

cout << "length of side1: " << side1 << endl;

cout << "length of side2: " << side2 << endl;

cout << "length of side3: " << side3 << endl;

cout << "area: " << area() << endl << endl;

}

private:

double side1, side2, side3;

Coordinate point1, point2, point3;

};

//定义派生类Rectangle

class Rectangle :public Shape

{

public:

Rectangle(Coordinate a = Coordinate(0, 0), Coordinate b = Coordinate(0, 0), string name = "")

:Shape(name), lefttop(a), rightbottom(b)

{

length = lefttop.gety() - rightbottom.gety();

width = rightbottom.getx() - lefttop.getx();//计算长和宽

}

~Rectangle() {}

double area() { return length \* width; }

void print()

{

printname();

cout << "point of lefttop: " << lefttop << endl;

cout << "point of rightbottom: " << rightbottom << endl;

cout << "length: " << length << endl;

cout << "width: " << width << endl;

cout << "area: " << area() << endl << endl;

}

private:

Coordinate lefttop, rightbottom;

double length, width;

};

//定义派生类Square

class Square :public Rectangle

{

public:

Square(Coordinate a, string name = "", double side = 0)

:Rectangle(Coordinate(a.getx(), side + a.gety()), Coordinate(a.gety(), a.getx() + side), name), point(a), side(side)

{}

~Square() {}

double area() { return side \* side; }

Circle incircle()

{

Circle circle(Coordinate(point.getx() + side / 2, point.gety() + side / 2), "", side / 2);

return circle;

}

void print()

{

printname();

cout << "point of leftbottom: " << point << endl;

cout << "length of side: " << side << endl;

cout << "area: " << area() << endl;

cout << "information of its incircle:" << endl;

cout << "\t" << "center: " << incircle().center << endl;

cout << "\t" << "radius: " << incircle().radius << endl;

cout << "\t" << "area: " << incircle().area() << endl << endl;

}

private:

double side;

Coordinate point;

};

int main()

{

Circle obj1(Coordinate(2, 2), "circle", 2);

Triangle obj2(Coordinate(0, 0), Coordinate(3, 1), Coordinate(1, 2), "triangle");

Rectangle obj3(Coordinate(0, 5), Coordinate(7, 0), "rectangle");

Square obj4(Coordinate(2, 2), "square", 6);

obj1.print();

obj2.print();

obj3.print();

obj4.print();

return 0;

}

要注意派生类不能访问基类的私有成员（而公有成员可以），而基类不能访问派生类。类的继承和友元类都可以实现类与类之间的交流

Lab5文件

14.ex1

#include<iostream>

#include<cstdlib>

#include<fstream>

#include<iomanip>

#include<random>

#include<ctime>

#include<array>

using namespace std;

const unsigned int days[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };//12月每月对应的天数

class Time//时间类 时：分：秒

{

public:

Time(const int h = 0, const int m = 0, const int s = 0)

{//默认构造函数，自动赋值为0：0：0

setTime(h, m, s);//调用构造函数

}

void setTime(const int h, const int m, const int s)

{//赋值函数（三个一起）

setHour(h);//时赋值函数

setMinute(m);//分赋值函数

setSecond(s);//秒赋值函数

}

void setHour(const int h)//时赋值函数

{

if (h >= 0 && h < 24)//属于正常的时间（0-23）

hour = h;

else//弹出报错

throw invalid\_argument("hour must be 0-23");//返回报错

}

void setMinute(const int m)

{

if (m >= 0 && m < 60)//属于正常的时间（0-59）

minute = m;

else//弹出报错

throw invalid\_argument("minute must be 0-59");//返回报错

}

void setSecond(const int s)

{

if (s >= 0 && s < 60)//属于正常的时间（0-59）

second = s;

else//弹出报错

throw invalid\_argument("second must be 0-59");

}

unsigned int getHour() const

{//返回值函数，返回小时

return hour;

}

unsigned int getMinute() const

{//返回值函数，返回分钟

return minute;

}

unsigned int getSecond() const

{//返回值函数，返回秒

return second;

}

virtual void print() const

{//标准-输出函数。输出各项信息(可重载）

cout << setfill('0') << setw(2) << getHour() << ":" << setw(2) << getMinute() << ":" << setw(2) << getSecond();

}

virtual void print1(ofstream &a) const

{//标准-输出函数。输出各项信息（可重载）

a << setfill('0') << setw(2) << getHour() << ":" << setw(2) << getMinute() << ":" << setw(2) << getSecond();

}

void printStandard() const

{//12小时制-输出函数

cout << ((getHour() == 0 || getHour() == 12) ? 12 : getHour() % 12) << ":" << setfill('0') << setw(2) << getMinute() << ":" << setw(2) << getSecond() << (hour < 12 ? " AM" : " PM") << endl;

}

virtual int getcompare()

{//这是个用于获取总秒数的比对函数

return hour \* 3600 + minute \* 60 + second;

}

friend void tick(Time &);//友元函数

private:

unsigned int hour; //0 - 23

unsigned int minute; //0- 59

unsigned int second; //0- 59

};

class Date :public Time//包含时间的日期类

{

friend std::ostream &operator<<(ostream &, const Date &);//经典输出友元函数

public:

Date(int m = 1, int d = 1, int y = 1900,const int h = 0, const int min = 0, const int s = 0)

{//初始化函数，初始天分与时间

setTime(h, min, s);//时分秒初始化函数

setDate(m, d, y);//年月天初始化函数

}//default constructor

void setDate(int mm, int dd, int yy)//年月天初始化函数

{

if (mm >= 1 && mm <= 12)//

month = mm;

else

throw invalid\_argument("Month must be 1-12");

if (yy >= 1900 && yy <= 2100)

year = yy;

else

throw invalid\_argument("Year must be>=1900 and<-2100");

// test for a leap year

if ((month == 2 && leapYear(year) && dd >= 1 && dd <= 29) || (dd >= 1 && dd <= days[month]))

day = dd;

else

throw invalid\_argument("Day is out of range for current month and year");

}// set month,day,year

Date &operator++()

{

helpIncrement();// increment date

return \*this;// reference return to create an lvalue

}//prefix increment operator

Date operator++(int)

{

Date temp = \*this;//hold current state of object

helpIncrement();//postfix increment operator

return temp;

}

Date &operator+=(unsigned int additiona1Days)

{

for (int i = 0; 1 < additiona1Days;++i)

helpIncrement();// add days,modify object

return \*this;

}

static bool leapYear(int testYear)

{

if (testYear % 400 == 0 || (testYear % 100 != 0 && testYear % 4 == 0))

return true; // a leap year

else

return false; // not a leap year

}// is date in a leap year?

bool endofMonth(int testDay)const

{

if (month == 2 && leapYear(year))

return testDay == 29;// last day of Feb. in leap year

else

return testDay == days[month];

}// is date at the end of month?

void helpIncrement()

{

//day is not end of month

if (!endofMonth(day))

++day; // increment day

else

if (month < 12)// day is end of month and month < 12

{

++month;// increment month

day = 1; // first day of new month

}//end if

else // 1ast day of year

{

++year; // increment year

month = 1; // first month of new year

day = 1; // first day of new month

}

}

int getmonthdate()const

{//返回月份

return month;

}

virtual void print()const

{//重载后的标准-输出函数

cout << month << "-" << day << "-" << year;

}

virtual void print1(ofstream &a)const

{//重载后的文件操作-输出函数

a << month << "-" << day << "-" << year;

}

virtual int getcompare()

{//重载后的返回比对值函数，不同的是，此值无意义

return year \* 10000 + month \* 100 + day;

}

private:

unsigned int month;

unsigned int day;

unsigned int year;

};

void tick(Time &c)//已经提前声明

{

int t;

t = 3600 \* c.hour + 60 \* c.minute + c.second;//将其转化为一天中的总秒数，无需if

t++;

t = t % 86400;//一天只有86400秒

c.hour = t / 3600;

c.second = t % 60;

c.minute = t % 3600 / 60;

}

class ipv4 :public Date//一个ipv4类

{

public:

ipv4(int i1=0,int i2=0,int i3=0,int i4=0, int m = 1, int d = 1, int y = 1900, const int h = 0, const int min = 0, const int s = 0)

{//初始构造函数

setTime(h, min, s);

setDate(m, d, y);

setipv4(i1, i2, i3, i4);

}

void setipv4(int i1 = 0, int i2 = 0, int i3 = 0, int i4 = 0)

{//设置IP，仅当0-255时可用

if (i1 >= 0 && i1 <= 255)

ip1 = i1;

else

ip1 = 0;

if (i2 >= 0 && i2 <= 255)

ip2 = i3;

else

ip2 = 0;

if (i4 >= 0 && i3 <= 255)

ip3 = i3;

else

ip1 = 0;

if (i4>= 0 && i4 <= 255)

ip4 = i4;

else

ip4 = 0;

}

void setip(int i1 = 0, int i2 = 0, int i3 = 0, int i4 = 0, int m = 1, int d = 1, int y = 1900, const int h = 0, const int min = 0, const int s = 0)

{//将其中所蕴含的12个数据全初始化一边

setTime(h, min, s);

setDate(m, d, y);

setipv4(i1, i2, i3, i4);

}

virtual void print()const

{//重载的标准-输出函数，输出该相对应的全部信息

cout << ip1 << ":" << ip2 << ":" << ip3 << ":" << ip4<<" ";

Date::print();

cout << " ";

Time::print();

cout << endl;

}

virtual void print1(ofstream &a)const

{//重载的文件操作-输出函数，输出该相对应的全部信息

a << ip1 << ":" << ip2 << ":" << ip3 << ":" << ip4 << " ";

Date::print1(a);

a << " ";

Time::print1(a);

a << endl;

}

bool operator>(ipv4& ip)

{//比对函数

int a1, a2;

a1 = Date::getcompare();

a2 = ip.Date::getcompare();

if (a1 != a2)//如果日期更大则直接结束判定

return ((a1 - a2) / fabs(a1 - a2)+1)/2;

else

{

a1 = Time::getcompare();//进行时间判定

a2 = ip.Time::getcompare();

if (a1 != a2)

return ((a1 - a2) / fabs(a1 - a2) + 1) / 2;

else

return false;

}

}

private:

int ip1, ip2, ip3, ip4;

};

void test001()

{

srand(time(0));//重载随机数

array<ipv4, 101>i\_p;//IP

int sor[101];

for (int i = 1; i <= 100; i++)

sor[i] = 0;

sor[1] = 1;

int side = 1;

int k;

ofstream here("test.txt", ios::out);//文件操作

for (int j = 1;j <= 100;j++)

{

const unsigned int num[11] = { 0,256,256,256,256,12,10000,120,24,60,60 };//对应的IP数据上界

unsigned int num1[11] = { 0,0,0,0,0,1,0,1901,0,0,0 };//对应的日期测试数据

int i;

for (i = 1;i <= 10;i++)

num1[i] += rand() % num[i];//随机生成日期，IP

num1[6] = num1[6] % ::days[num1[5]]+1;//月份日期，较为特殊

i\_p[j].setip(num1[1], num1[2], num1[3], num1[4], num1[5], num1[6], num1[7], num1[8], num1[9], num1[10]);//将其载入IP

for (k = 1; k < j; k++)

{

if (i\_p[sor[k]] > i\_p[j])//把这一个以升序顺序载入IP数组

{

for (int p = j; p >k;p--)

sor[p] = sor[p-1];

sor[k] = j;

k = j + 1;

}

}

if (k == j)

sor[k] = k;

}

for (k = 1; k <= 100; k++)

{

i\_p[sor[k]].print1(here);//输出去

}

here.close();

}

void test002()

{

ifstream there("test.txt", ios::in);//读入

int sum[6] = { 0,0,0,0,0,0 };

int i = 1;

int ip1;

for (i = 0;i < 100;i++)

{

there >> ip1;//通过重载得到读入来读入并分类

there.ignore();//忽略 换行符

if (ip1 <= 127)//根据IP数据范围来统计数据

sum[1]++;

else if (ip1 <= 191)

sum[2]++;

else if (ip1 <= 223)

sum[3]++;

else if (ip1 <= 239)

sum[4]++;

else sum[5]++;

for (int j = 1; j <= 9; j++)

{

there >> ip1;

there.ignore();//再次读入一个IP

}

}

for (i = 1; i <= 5; i++)

cout << sum[i] << endl;

}

int main()

{

test001();

test002();

}

14.ex2

#include<iostream>

#include<fstream>

#include<iomanip>

#include<string>

using namespace std;

class student

{

public:

student(int n=0, char\* fn=nullptr, char\* ln = nullptr, char\* idnum = nullptr)

{//构造函数，依次是学号，fistname，lastname，id

setstudent(n, fn, ln, idnum);

}

/\*~student()

{

}\*/

void setstudent(int n=0, char\* fn = nullptr, char\* ln = nullptr, char\* idnum = nullptr)

{//student的批量赋值函数

setnum(n);//依次调用各自的赋值函数

setfirstname(fn);

setlastname(ln);

setid(idnum);

}

void setnum(int n)//学号赋值函数

{

num = n;

}

int getnum()//学号返回函数

{

return num;

}

void setfirstname(char\* fn)//firstname赋值函数

{

int i = 0;

if(fn!=nullptr)//如果说有可复制的

while (i <= 16 && \*(fn + i) != '\0')

{//最多复制位

firstname[i] = \*(fn + i);

i++;

}

firstname[i] = '\0';

}

void setlastname(char\* ln)

{//lastname赋值函数

int i = 0;

if (ln != nullptr)//如果说有可复制的

while (i <= 16 && \*(ln + i) != '\0')//做多赋值16位

{

lastname[i] = \*(ln + i);

i++;

}

lastname[i] = '\0';

}

void setid(char\* idnum)

{//初始化18位的身份证号

if (idnum != nullptr)

for (int i = 1; i <= 18; i++)

id[i] = \*(idnum + i);

}

void print()//统一制表输出

{

cout.setf(ios::left);

cout << setw(5) << num;//编号

for (int i = 1; i <= 18; i++)

cout << id[i];//身份证号

cout << " ";

cout <<setw(17) << firstname<<setw(17)<<lastname;//主要是为了对齐

cout << endl;

}//num++id++++++++++++++++++firstname++++++++

private:

int num;

char firstname[17], lastname[17];

char id[19];

};

void changeid(fstream &here,int n,char\* p)//更改id

{

student stu(n);//声明一个空student

here.seekg((n - 1) \* sizeof(student));//跳转到n-1对应的学生开头

here.read(reinterpret\_cast<char\*>(&stu), (sizeof(student)));//读入对应长度的身份证号（注：这个是当时纠错用的）

stu.setid(p);//设置

here.seekp((n - 1) \* sizeof(student));//将put指针移动到哪里

here.write(reinterpret\_cast<char\*>(&stu), sizeof(student));//覆盖重写

}

void changefirstname(fstream& here, int n, char\* fn)

{

student stu(n);//声明一个空student

here.seekg((n - 1) \* sizeof(student));//跳转到n-1对应的学生开头

here.read(reinterpret\_cast<char\*>(&stu), (sizeof(student)));//读入对应长度的firstname

stu.setfirstname(fn);//设置

here.seekp((n - 1) \* sizeof(student));//将put指针移动到哪里

here.write(reinterpret\_cast<char\*>(&stu), sizeof(student));//覆盖重写

}

void changelastname(fstream& here, int n, char\* ln)

{

student stu(n);//声明一个空student

here.seekg((n - 1) \* sizeof(student));//跳转到n-1对应的学生开头

here.read(reinterpret\_cast<char\*>(&stu), (sizeof(student)));//读入对应长度的lastname

stu.setlastname(ln);//设置

here.seekp((n - 1) \* sizeof(student));//将put指针移动到哪里

here.write(reinterpret\_cast<char\*>(&stu), sizeof(student));//覆盖重写

}

void add(student stu,fstream& here)//添加

{

here.seekp((stu.getnum() - 1) \* sizeof(student));//移动put指针

here.write(reinterpret\_cast<char\*>(&stu), sizeof(student));//将整个student写上去

}

void picture(fstream& here)

{

student stu;//空置student

int i;

i = 1;

cout << "num id firstname lastname " << endl;//初始制表（标题）

here.seekg(0);

here.read(reinterpret\_cast<char\*>(&stu), sizeof(student));//初始读入

while (i<100)

{

i++;//1至100遍历

if(stu.getnum()!=0)

stu.print();//打印

here.read(reinterpret\_cast<char\*>(&stu), sizeof(student));//向下挪动一位

here.seekg((i - 1) \* sizeof(student));

}

}

void map(fstream &here)

{//把每一个点位都创建出来，并且添加终止符

student stu;

for(int i=1;i<=100;i++)

here.write(reinterpret\_cast<char\*>(&stu), sizeof(student));

here << EOF;

}

void delete1(fstream& here, int num)

{//将某个点位对应的数据洗掉

student stu;

here.seekp((num - 1) \* sizeof(student));

here.write(reinterpret\_cast<char\*>(&stu), sizeof(student));

}

int main()

{//快乐测试

char p[19] = {'1','2','3','6','2','3','9','1','2','3','6','2','3','9','9','1','7','7','7'};

char fn1[19] = { 'z','h','a','n','g','\0' };

char ln1[19] = { 's','a','n','\0' };

char fn2[19] = { 'w','a','n','g','\0' };

char ln2[19] = { 'f','o','u','r','\0'};

student stu(77,fn1,ln1,p);

fstream here("student.dat", ios::in | ios::out | ios::binary);

//fstream there("window.txt", ios::in | ios::out | ios::binary);

map(here);

add(stu, here);

stu.setstudent(20, fn2, ln2, p);

add(stu,here);

picture(here);

changefirstname(here, 77, ln2);

picture(here);

delete1(here,77);

picture(here);

here.close();

//there.close();

}

Lab7异常

#include<iostream>

#include<stdexcept>

using namespace std;

class TestExcept :public runtime\_error

{

public:

TestExcept():runtime\_error("attempted to divide by zero"){}

};

double test01(int a, int b)

{

if (b == 0)

{

throw TestExcept();

}

return a / b;

}

int main()

{

int number1;

int number2;

cout << "Enter two integers(end\_of\_file to end):";

while (cin >> number1 >> number2)

{

try

{

double result = test01(number1, number2);

cout << "This is a test" << endl;

}

catch (TestExcept& testexcept)

{

throw TestExcept();

}

}

return 0;

}

2.

#include<iostream>

#include<stdexcept>

using namespace std;

class Item

{

public:

int value;

Item(int a):value(a)

{

cout << "Item" << a << "constructr is called" << endl;

if (value == 3)

{

throw runtime\_error("An exception was thrown");

}

}

~Item()

{

cout << "Item" << value << "is destructor is called" << endl;

}

};

int main()

{

try

{

Item a1(1);

Item a2(2);

Item a3(3);

}

catch (runtime\_error& r1)

{

cout << "An exception is called" << endl;

}

return 0;

}

3.

#include<iostream>

#include<stdexcept>

#include<exception>

using namespace std;

class TestExcept:public runtime\_error

{

public:

TestExcept():runtime\_error("This is an except")

{

}

};

void g()

{

try

{

throw TestExcept();

}

catch (exception& ptr)

{

cout << "Exception caught in function g().Rethrowing...."<< endl;

throw TestExcept();

}

}

int main()

{

try

{

g();

}

catch (exception &ptr)

{

cout << "Exception caught in main function" << endl;

}

return 0;

}

4.

#include<iostream>

#include<stdexcept>

#include<exception>

using namespace std;

class TestExcept:public runtime\_error

{

public:

TestExcept():runtime\_error("This is an except")

{

}

void what()

{

cout << "This is an exception" ;

}

};

void h()

{

throw TestExcept();

}

void g()

{

h();

}

void f()

{

g();

}

int main()

{

try

{

f();

}

catch (TestExcept& ptr)

{

ptr.what();

cout << " be caught in main function" << endl;

}

return 0;

}

Lab6容器

2. #include<iostream>

3. #include<vector>

4. #include<algorithm>

5. #include<string>

6. using namespace std;

7. int compare(vector<string> v1)

8. {

9. string s;

10. for (int i = 0;i < 11;i++)

11. {

12. cin >> s;

13. v1.push\_back(s);

14. }

15. for (vector<string>::iterator str1 = v1.begin();str1 != v1.end();str1++)

16. {

17. cout << \*str1 << " ";

18. }

19. cout << endl;

20. vector<string>::iterator str2 = v1.begin();

21. vector<string>::iterator str3 = v1.end() - 1;

22. for (int i = 0;i < v1.size();i++)

23. {

24. if (\*(str2 + i) != \*(str3 - i))

25. {

26. cout << "不是回文" << endl;

27. return 0;

28. }

29. }

30. cout << "是回文" << endl;

31. return 0;

32. }

33. int main()

34. {

35. vector<string>v1,v2;

36. compare(v1);

37. compare(v2);

38.

39. return 0;

40. }

#include<iostream>

#include<vector>

#include<algorithm>

#include<string>

using namespace std;

void makezero(vector<int>& v,int g)

{

for (int i = 2;i \* g < 1001;i++)

{

v[i \* g] = 0;

}

}

int main()

{

vector<int> v1(1001, 1);

for (int i = 2;i < 1001;i++)

{

int flag = 1;

for (int j = 2;j < i;j++)

{

if (i % j == 0)

{

flag = 0;

}

}

if (flag)

makezero(v1, i);

}

int counter = 0;

cout << "The prime number between 2 and 999" << endl;

for (int i = 2;i < 1001;i++)

{

if (v1[i] == 1)

{

cout << i << "\t ";

counter++;

if (counter % 10 == 0)

{

cout << endl;

}

}

}

cout << endl;

int number;

while (1)

{

cout << "Enter a number from 2 to 999: " ;

cin >> number;

Flag:

if (number < 2 || number > 999)

{

cout << "Please record" << endl;

cin >> number;

goto Flag;

}

if (v1[number] == 1)

{

cout << number << "is a prime" << endl;

}

else

{

cout <<number<< " = ";

for (int i = 2;i < number;i++)

{

if (number % i == 0 && v1[i] == 1)

{

int temp = number;

while (temp % i == 0)

{

temp /= i;

cout << i;

if (temp/i != 0)

cout << "\*";

}} }

cout << endl;}}}