上机实训01：初入C++

【上机目的】

1. 适应由C到C++的过渡
2. 熟悉在所用的系统上编辑、编译、连接和运行C++程序的方法

【实训内容】

输入以下程序，进行编译，观察编译情况，如果有错误，请修改程序，直到没有错误，先分析结果，然后运行 观察结果。

（1）源程序：

#include <iostream>

using namespace std;

int main();

{

int a,b,c;

cin>>a>>b;

c=a+b;

cout>>"a+b=">>a+b;

return 0;

}

（2）源程序：

#include <iostream>

using namespace std;

int add(int x,int y);

int main()

{

int a,b,c;

cin>>a>>b;

c=add(a,b)

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

return 0;

}

int add(int x,int y) ;

{int z;

z=x+y;

return(z);

}

【上机要求】

编写C++程序，成功运行源程序。

【上机指导】

（1）修改后的源程序：

#include <iostream>

using namespace std;

int main()

{

int a, b, c;

cin >> a >> b;

c = a + b;

cout << "a+b=" << a + b;

return 0;

}

该程序的cout右边应为“<<”，而不是>>。

（2）修改后的源程序：

#include <iostream>

using namespace std;

int add(int x, int y);

int main()

{

int a, b, c;

cin >> a >> b;

c = add(a, b);

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

return 0;

}

int add(int x, int y)

{

int z;

z = x + y;

return(z);

}

该程序的c = add(a, b)右边少了“;”，int add(int x, int y);

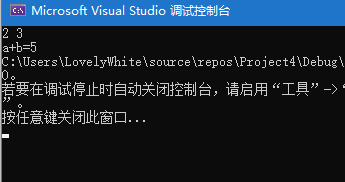
声明应放在定义之前;

函数定义后面不能加;，将int add(int x, int y);后的分号去掉。

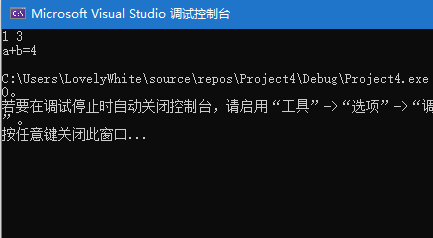
【运行结果】

编译运行此程序即可在命令行代码中得到以下结果。

（1）



（2）



【范例分析】

该程序通过cin输入数字，这是一个输入流对象，它重载了>>，让这个符号有了输入的功能。cout则刚好与cin相反，它的作用是输出。

在程序的开头调用了C++标准输入输出流。这和C语言里面的stdio.h有些类似。

上机实训02：基于函数的模块化程序设计

【上机目的】

1. 掌握函数的定义和调用
2. 掌握函数间的参数传送（传值、传地址）
3. 了解变量的作用域与存储类别

【实训内容】

1.函数默认值的使用

例：

int add(int x=5,int y=6)

{ return x+y; }

void main()

{ add(); //调用在实现后

}

编一个程序，用来求3个正整数中的最大数。

【上机要求】

（1）用不带默认参数的函数实现，3个整数通过参数传递

（2）用带有默认参数的函数实现，最后1个参数 默认值为 10；调用的采用带3个参数 和2个参数观察结果。

【上机指导】

#include<iostream>

using namespace std;

int findmax(int a, int b, int c = 10)

{

int max;

max = a > b ? a : b;

max = max > c ? max : c;

return max;

}

int main()

{

int x, y, z;

cin >> x >> y >> z;

cout << "传三个参数：\n";

cout<<findmax(x,y,z)<<endl;

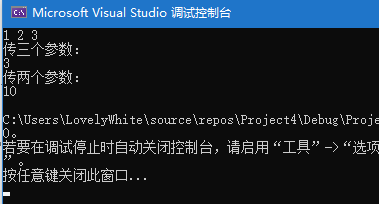
cout << "传两个参数：\n";

cout<<findmax(x, y)<<endl;

return 0;

}

【运行结果】



【范例分析】

在c语言中我们见得最多的就是模块化设计，在c++里我们使用的是类，在类中我们需要实现模块化设计。这样有助于程序的阅读与理解。

该程序的重点难点部分在于函数的调用和声明。在上面的程序当中，声明了一个带有默认参数的函数。C++中规定，必须最后几个参数才能带默认值。如果int findmax(int a, int b=3, int c)这样声明的方法是错误的。如果想让b带参数的话，正确的方法是：int findmax(int a, int c, int b=3)，将b放在后面。

上机实训03：类的定义和对象的创建

【上机目的】

1. 掌握类的声明和使用；
2. 掌握对象的声明和使用；
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。

【实训内容】

（1）以下是类的定义和对象的使用，请观察类的定义语法和对象的定义方法，并运行程序。

第一种成员函数定义：成员函数定义在类的内部

#include<iostream>

#include<string>

using namespace std;

class student {

private: //类的成员变量

int no;

string name ;

float score;

public:

void input(int no1,string name1,float score1) { //类的成员函数input定义

no=no1;

name=name1;

score=score1;

}

void output() {//类的成员函数output定义

cout<<" no="<<no<<endl;

cout<<" name="<<name<<endl;

cout<<" score="<<score<<endl;

}

void deal(int no1,string name1,float score1) {

input(no1,name1,score1);

output();

}

};

int main() {

student st;

int no1;

string name1;

float score1;

cin>>no1>>name1>>score1;

st.input(no1,name1,score1);

st.output();

st.deal(no1,name1,score1);

return 0;

}

另一种成员函数定义：成员函数定义在类外部

#include<iostream>

#include<string>

using namespace std;

class student {

private: //类的成员变量

int no;

string name ;

float score;

public: //类的成员函数原型

void input(int,string,float);

void output();

void deal(int,string,float);

};

void student::input(int no1,string name1,float score1); {

no=no1;

name=name1;

score=score1;

}

void student::output() {

cout<<" no="<<no<<endl;

cout<<" name="<<name<<endl;

cout<<" score="<<score<<endl;

}

void student:deal(int no1,string name1,float score1) {

input(no1,name1,score1);

output();

}

int main() {

student st;

int no1;

string name1;

float score1;

cin>>no1>>name1>>score1;

st.input(no1,name1,score1);

st.output();

st.deal(no1,name1,score1);

return 0;

}

（2）请检查下面的程序，找出其中的错误（先不要上机，在纸面上作人工检查），并改正。然后上机调试，使之能正常运行。运行时从键盘输入时、分、秒的值，检查输出是否正确。

#include <iostream>

using namespace std;

class Time {

void set\_time(void);

void show\_time(void);

int hour;

int minute;

int sec;

};

Time t;

void set\_time(void) { //在main函数之前定义

cin>>t.hour;

cin>>t.minute;

cin>>t.sec;

}

void show\_time(void) { //在main函数之前定义

cout<<t.hour<<":"<<t.minute<<":"<<t.sec<<endl;

}

int main() {

set\_time();

show\_time();

return 0;

}

【上机要求】

无。

【上机指导】

1. 类中函数的定义可以有两种形式，第一种在类中定义，第二种在类外定义。在类外定义的时候需要先在类中声明函数原型。定义方法为：返回值 类名:函数名（参数）。

修改后的源程序：

#include<iostream>

using namespace std;

class time

{

public:

void set\_time();

void show\_time();

private:

int hour;

int min;

int sec;

};

void time::set\_time()

{

cin >> hour >> min >> sec;

}

void time::show\_time()

{

cout << hour << ":" << min << ":" << sec << endl;

}

int main() {

time t;

t.set\_time();

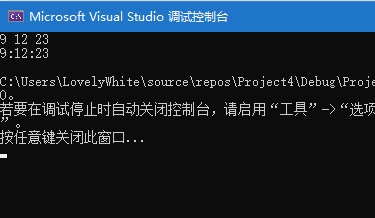
t.show\_time();

return 0;

}

【运行结果】

2



【范例分析】

在C++中对C语言中的结构体做了一个扩展，也就是类。类的定义需要调用其构造函数，如Time类，它没有显示的定义构造函数，但是我们知道它有一个默认构造函数Time()，这个函数的功能就是让Time类进行初始化。

上机实训04：基于数组构造的程序设计

【上机目的】

1．掌握一维数组和二维数组的定义、赋值和输入和输出的方法；

2．掌握字符数组和字符串函数的使用；

3．掌握与数组有关的算法

【实训内容】

（1）把下列代码复制到开发环境中，完善各个函数，并调用相关函数实现将一组double型数据的小数部分之和输出到屏幕上。

在main函数中，定义了如下变量：

double sm;//存储这组数的小数部分之和

double tt[M];//存储从键盘输入的数据

例如：输入103.91、212.87、334.79、23.93、15.4，则输出3.900000。

#include <conio.h>

#include <stdio.h>

#include <stdlib.h>

#define M 5

void PrintData(double sum);

void ReadData(double tt[M]);

double Acc( int k, double dd[] ) {//实现计算数组元素的小数部分之和

//写入你的代码

}

void main() {

int m;

double sm;

double tt[M];

system("cls");//清除屏幕数据

ReadData(tt); //读取数据

//此处添加求小数部分之和的函数调用语句

PrintData(sm); //结果显示在屏幕上

}

void PrintData(double sum) { //把sum数据输出到屏幕上

//写入你的代码

}

void ReadData(double tt[]) { //从键盘中输入5个double数据存于tt数据中

//写入你的代码

}

（2）对象数组的使用

对象数组的声明：类名 数组名[常量表达式]

Point p[2]; //定义了2个Point类型的对象（由C++运行时系统自动创建）

或者

Point p[2]={Point(),Point(1,2)};

对象数组的访问：数组名[下标].成员名

p[1].move(1,2);

观察下面的程序，仔细阅读每一句代码，了解其意义，然后预测运行输出结果，然后运行程序，看预测结果与实际运行是否一致

#include <iostream>

using namespace std;

class Point {//定义Point 类

public:

Point();

Point(int xx,int yy);

~Point();

void Move(int x,int y);

int GetX() {return X;}

int GetY() {return Y;}

private:

int X,Y;

};

Point::Point() {

X=Y=0;

cout<<"Default Constructor called."<<endl;

}

Point::Point(int xx,int yy) {

X=xx;

Y=yy;

cout<< "Constructor called."<<endl;

}

Point ::~Point() { cout<<"Destructor called."<<endl; }

void Point ::Move(int x,int y) {

cout<<"moving the point to("<<x<<","<<y<<")"<<endl;

X=x;

Y=y;

}

int main() {

int i;

cout<<"Entering main..."<<endl;

Point A[2]; //定义Point类型的对象数组

for( i=0;i<2;i++) A[i].Move(i+10,i+20);//通过下标使用对象数组元素

cout<<"print A..."<<endl;

Point B[2]={Point(),Point(1,2)}; //定义时初始化（注意无参对象的写法）

for(i=0;i<2;i++) B[i].Move(i+10,i+20);

cout<<"Exiting main..."<<endl;

return 0;

}

（3）编程：定义一个圆类Circle，具有数据成员r；有参和无参构造函数；成员函数float area()求面积。主函数中定义1个Circle类型具有2个元素的对象数组，并调用相关方法求每个对象的面积。

（4）定义ArrayData类，包含数据成员：整型量count、整型指针arr，分别记录数组元素的个数、保存数组指针；成员函数setdata()，对count赋值、用new动态创建数组，并把指针赋于arr；int sum()，返回数组元素之和；void sort()，对数组排序。

【上机要求】

无。

【上机指导】

（1）在这段程序当中我们需要知道如何计算一个实数的小数部分，以及如何输出数组。

源代码：

#include <conio.h>

#include <stdio.h>

#include <stdlib.h>

#include<iostream> //c++输入输出

using namespace std; //c++命名空间

#define M 5 //数组大小

void PrintData(double sum); //输出函数声明

void ReadData(double tt[M]); //数组读入函数声明

double Acc(/\*int k, \*/double dd[]) {//实现计算数组元素的小数部分之和

//写入你的代码

double sum = 0;

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

sum += dd[i] - static\_cast<int>(dd[i]);

return sum;

}

void main() {

//int m;

double sm;

double tt[M];

system("cls"); //清除屏幕数据

ReadData(tt); //读取数据

sm = Acc(tt); //求小数部分之和的函数调用语句

PrintData(sm); //结果显示在屏幕上

return;

}

void PrintData(double sum) { //把sum数据输出到屏幕上

//写入你的代码

printf("sum = %lf\n", sum);

}

void ReadData(double tt[]) { //从键盘中输入5个double数据存于tt数据中

//写入你的代码

printf("请输入%d个实数：\n", M);

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

cin >> tt[i];

}

（2）在这段程序中我们需要掌握对象数组的声明以及创建。当我们声明有一个对象数组的时候同样也是调用它的构造方法。

#include<iostream>

using namespace std;

const float PI = 3.14;

class Cirlce

{

public:

Cirlce();

Cirlce(float radius);

~Cirlce();

float area();

private:

float r;

};

Cirlce::Cirlce()

{

cout << "Default constructor called." << endl;

}

Cirlce::~Cirlce()

{

cout << "Destructor called." << endl;

}

Cirlce::Cirlce(float radius) {

r = radius;

}

float Cirlce::area() {

return PI \* r\* r;

}

int main() {

float r[2];

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

cin >> r[i];

Cirlce test[2];

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

test[i] = Cirlce(r[i]);

cout << "area = " << test[i].area() << endl;

}

return 0;

}

（3）在写这道题之前我们需要掌握类的构造函数、如何声明和定义类成员函数、如何定义类数组。

源代码：

#include<iostream>

using namespace std;

const float PI = 3.14;

class Cirlce

{

public:

Cirlce();

Cirlce(float radius);

~Cirlce();

float area();

private:

float r;

};

Cirlce::Cirlce()

{

cout << "Default constructor called." << endl;

}

Cirlce::~Cirlce()

{

cout << "Destructor called." << endl;

}

Cirlce::Cirlce(float radius) {

r = radius;

}

float Cirlce::area() {

return PI \* r\* r;

}

int main() {

float r[2];

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

cin >> r[i];

Cirlce test[2];

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

test[i] = Cirlce(r[i]);

cout << "area = " << test[i].area() << endl;

}

return 0;

}

（4）这道题需要你会一种排序算法，整体来讲不算太难。重点在于排序算法。

源代码：

#include <iostream>

using namespace std;

class ArrayData {

private:

int count;

int\* arr;

public:

void setdata(int ar[], int m) {//设置数据

int i;

count = m;

arr = new int[count];

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

arr[i] = ar[i];

}

int sum() { //求和

int sum = 0;

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

sum += arr[i];

return sum;

}

void sort() { //排序

for (int i = 0; i < count; i++) {

for (int j = i + 1; j < count; j++) {

if (arr[i] > arr[j]) {

int t = arr[i];

arr[i] = arr[j];

arr[j] = t;

}

}

}

}

void output() { //输出结果

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

cout << arr[i] << " ";

}

~ArrayData() { delete[] arr; } //析构函数

};

void main() {

int a[5], i;

ArrayData X;

cout << "input data" << endl;

for (i = 0; i < 5; i++) cin >> a[i];

X.setdata(a, 5);

X.sort();

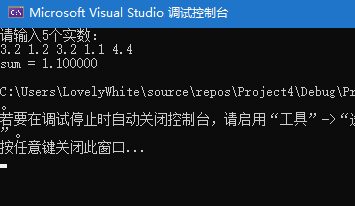
X.output();

cout << "sum = " << X.sum() << endl;

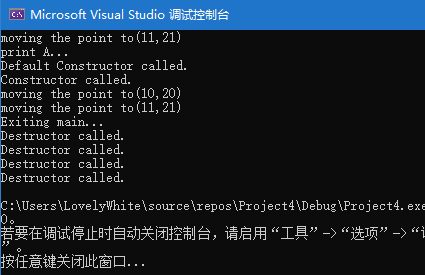
}

【运行结果】

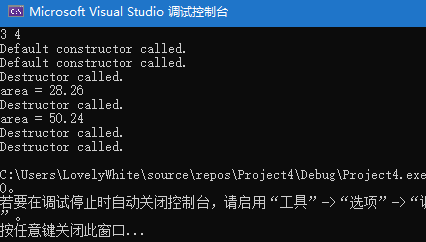
（1）



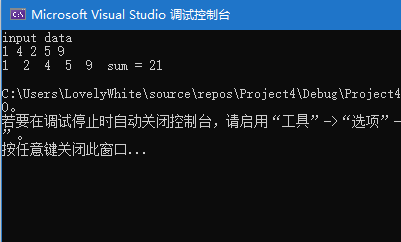
（2）



（3）



（4）



【范例分析】

数组是一种数据结构，在第一个实验中要求我们补充这些函数。其中用到了数组。数组不仅仅单纯的只能存数，它其实能够存任意类型的数据，比如类。在C语言中我们有结构体数组，在C++里相应的就成了对象数组。第二个实验当中我们定义了Circle的构造与析构函数，并且重载了它的构造函数，接受一个float类型的数据。这样做的目的是在创建这个对象的过程中就已经将r的值确定了。这也被称为是一种带参数的构造方法。第三个实验要求我们实现求面积的功能，在Circle类中定义了Cirlce(float radius);它能够接收一个浮点类型的参数，这个参数用来初始化圆的半径。定义了一个area函数，用于计算面积。第四个实验中运用了一种简单的排序算法。通过setdata函数来设置数据，通过sum函数来计算总和。sort函数来排序等等。

上机实训05：基于指针构造的程序设计

【上机目的】

1．掌握指针的概念，会定义和使用指针变量

2．指针与对象结合的使用

【实训内容】

1．程序设计题：请编写函数fun(char \*s)，功能是把s所指字符串中的内容逆置。例如：字符串中原有的字符串为：abcdefg，则调用该函数后, 串中的内容为：gfedcba

2. 编译程序，改正编译错误

#include<iostream>

#include<string>

using namespace std;

class person {//person基类

private:

string name;

protected:

int age;

public:

void setvalue\_p(string name1,int age1) { name=name1; age=age1; }

void print() { cout<<"name="<<name<<" age="<<age<<endl; }

};

class student:public person { //student 派生类

private:

int no;

float score;

public:

void setvalue(string name1,int age1,int no1,float score1) {

name=name1;//私有成员在 派生类中不可直接访问

age=age1; //这里不报错，保护成员在派生类中可直接访问

no=no1;

score=score1;

}

void output() {

print();

cout<<"no="<<no<<" score="<<score<<endl;

}

};

void main() {

student st1;

st1.setvalue("张三",20,1011,93.1);

cout<<"----"<<endl;

st1.output();

cout<<"----"<<endl;

// st1.print();

}

3．赋值兼容性原则

* 派生类的对象可以赋值给基类对象。
* 派生类的对象可以初始化基类的引用。
* 指向基类的指针也可以指向派生类。
* 通过基类对象名、指针和引用只能使用从基类继承的成员

#include <iostream>

using namespace std;

class B0 { //基类B0声明

public:

void display(){ cout<<"B0::display()"<<endl; } //公有成员函数

};

class B1: public B0 {

public:

void display(){cout<<"B1::display()"<<endl;}

};

void fun(B0 \*ptr) { ptr->display(); //"对象指针->成员名" }

void main() { //主函数

B0 b0; //声明B0类对象

B1 b1; //声明B1类对象

B0 \*p; //声明B0类指针

p=&b0; //B0类指针指向B0类对象

fun(p);

p=&b1; //B0类指针指向B1派生类对象

fun(p);

b0=b1;//派生类对象赋值给基类对象

b0.display();

B0 &x=b1; //派生类对象初始化基类对象的引用

x.display();

b1.display ();

}

回答问题：为什么运行结果显示了4个B0::display() 和1个B1::display()？

【上机要求】

无。

【上机指导】

（1）源程序：

#include<string.h>

#include<stdio.h>

#include<iostream>

using namespace std;

#define N 81

void fun(char\* s) {

//写入你的代码

int len = strlen(s);

char str[N];

for (int i = 0; i < len; i++) {

str[i] = \*(s + len - 1 - i);

}

str[len] = '\0';

strcpy\_s(s, len + 1, str);

}

void main(void) {

char a[N];

printf("Enter a string : ");

cin >> a;

printf("The original string is : ");

puts(a);

fun(a);

printf("\n");

printf("The string after modified : ");

puts(a);

}

（2）修改后的源程序：

#include<iostream>

#include<string>

using namespace std;

class person {//person基类

private:

string name;

protected:

int age;

public:

void setvalue\_p(string name1, int age1) { name = name1; age = age1; }

void print() { cout << "name=" << name << " age=" << age << endl; }

};

class student :public person { //student 派生类

private:

int no;

float score;

public:

void setvalue(string name1, int age1, int no1, float score1) {

//name = name1;//私有成员在 派生类中不可直接访问

//age = age1; //这里不报错，保护成员在派生类中可直接访问

//修改代码

setvalue\_p(name1, age1);

no = no1;

score = score1;

}

void output() {

print();

cout << "no=" << no << " score=" << score << endl;

}

};

void main() {

student st1;

st1.setvalue("张三", 20, 1011, 93.1);

cout << "----" << endl;

st1.output();

cout << "----" << endl;

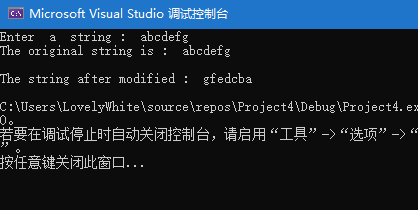
// st1.print();

}

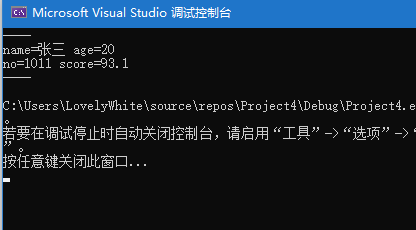
（3）答：在该程序中，声明了一个形参为基类B0类型指针的普通函数fun，根据类型兼容规则，可以将公有派生类对象的地址基类类型的指针，这样，使用fun函数就可以统一对这个类族中的对象进行操作.程序运行过程中，分别把基类对象和派生类对象B1赋值给基类类型指针p，但是，通过指针p，只能使用继承下来的基类成员。也就是说，尽管指针指向派生类B1的对象，fun函数运行时通过这个指针只能访问到B1类从B0类继承过来的display函数，而不是B1类自己的同名成员函数。因此，在主函数中前四次调用都是访问的B0的display函数，最后一次b1是B1类的对象，通过对象.成员函数访问时，会自动隐蔽基类中的display函数，因此访问到的是B1中的display函数。

【运行结果】

（1）



（2）



【范例分析】

第一个实验当中定义了一个fun函数，这个函数的作用就是倒置字符串，具体是重新申请一块区域，然后再原字符传中倒着取字符，依次放入这个新区域内，达到了倒置的功能。

第二个实验的错误原因是tudent类中的name是从person中继承的私有成员，不能在student中直接访问，所以将其注释掉。保护类成员在它的子类当中可以直接访问。s

上机实训06：继承设计

【上机目的】

1．了解派生类的构造函数和析构函数。

2．进一步理解继承与派生的概念。

3．掌握多重继承的使用。

【实训内容】

执行下面的程序回答问题：

#include<iostream>

#include<string>

using namespace std;

class person { //person基类

private:

string name;

protected:

int age;

public:

void printname();

void setname(string);

person() {

name="基类test";

cout<<"name="<<name<<" 基类构造函数执行"<<endl;

}

person(string name1) {

name=name1;

//cout<<"name="<<name<<" 基类构造函数执行"<<endl;

}

~person() {

//cout<<"name="<<name<<" 基类person析构函数执行"<<endl;

}

};

void person::printname() { cout<<"基类name="<<name<<endl; }

void person::setname(string name1) { name=name1; }

class student:public person {//student 继承person 是派生类

private:

int no;

float score;

//person p1;

protected:

string sex;

public:

void output();

student()//:person("mm"),p1("4")

{

no=1;

score=80.0;

age=20;

cout<<"no="<<no<<" 派生类构造函数执行"<<endl;

}

~student() { // cout<<"no="<<no<<" "<<"派生类析构函数"<<endl;

}

};

void student::output() {

printname();

cout<<"age="<<age<<endl;

}

void main() {

student st; //说明在创建st对象时，调用了哪个构造函数

//是否调用了基类构造函数？如果是，调用的是哪个基类的构造函数

cout<<"st----"<<endl;

st.printname();

}

1. 运行程序、观察结果，输出结果的第一行是执行的哪个类的构造函数，第二行又执行的哪个类的构造函数，把结果显示在下面。
2. 在（1）步的基础上，将

class student:public person {//student 继承person 是派生类

private:

int no;

float score;

//person p1;

…

}

中的注释符号去掉。观察运行结果，为什么会多出一个构造函数的执行？说明在派生类中，构造函数的执行顺序，把运行结果显示在下面。

（3）在（2）步的基础上，将

person(string name1) {

name=name1;

//cout<<"name="<<name<<" 基类构造函数执行"<<endl;

}

和

student()//:person("mm"),p1("4")中的注释去掉。

运行程序，观察每行的输出结果，与前面有什么不同？将运行结果显示在下面。

（4）在（3）步的基础上，将

~person() {

//cout<<"name="<<name<<" 基类person析构函数执行"<<endl;

}

和

~student() { // cout<<"no="<<no<<" "<<"派生类析构函数"<<endl; }

中的注释去掉，运行并观察结果，将结果显示在下面，然后总结析构函数的执行顺序。

【上机要求】

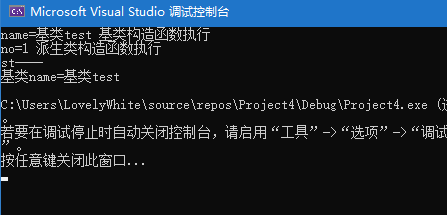
无。

【上机指导】

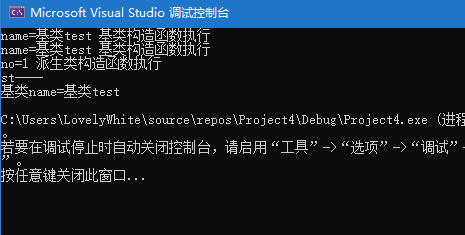
答：第一行是基类的构造函数执行； 第二行是派生类的构造函数执行。

【运行结果】

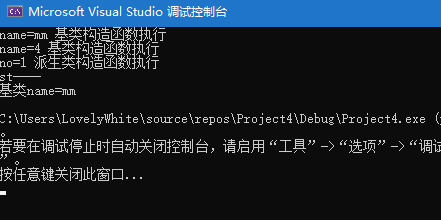
（1）



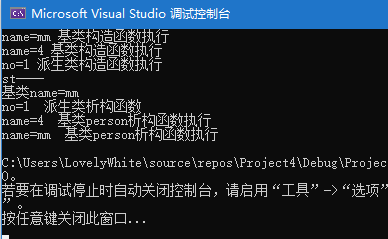
（2）



（3）



（4）



【范例分析】

在派生类student中，语句person p1表示创建一个基类的person的对象p1，此时需要调用基类的构造函数初始化；

基类构造函数总是被优先调用，这说明创建派生类对象时，会先调用基类构造函数，再调用派生类构造函数，如果继承关系有好几层的话，例如：

A –> B –> C

那么创建 C 类对象时构造函数的执行顺序为：

A类构造函数 –> B类构造函数 –> C类构造函数

构造函数的调用顺序是按照继承的层次自顶向下、从基类再到派生类的。而构函数的执行顺序与构造函数的执行顺序相反。

上机实训07：运算符重载

【上机目的】

掌握运算符重载方法。

【实训内容】

观察下列程序，掌握双目运算符的类成员函数重载方式，模仿这种重载的定义格式和调用格式，完成后面的编程题。

#include<iostream>

using namespace std;

class complex {

double real, imag;

public:

complex(double r = 0, double i = 0) { real = r; imag = i; }

complex operator + (complex &); //运算符重载的成员函数原型

//运算符结果类型 operator @(参数表列); 其中，@表示运算符

void print();

};

/\*运算符重载实现：

函数返回类型 类名::operator @(参数表列)

{

//代码

}\*/

complex complex::operator+(complex &c) { //operator+ 是类complex的成员函数

complex sum;

sum.real = real + c.real;

sum.imag = imag + c.imag;

return sum;

}

void complex::print() {

cout<<"real="<<real<<endl;

cout<<"imag="<<imag<<endl;

}

void main() {

complex a(5.0,6.0), b(4.0,5.0);

complex c;

c=a+b;

//c=5+a;

c.print();

}

说明：+是双目运算符，成员函数operator+(complex &c)的参数c是+的右操作数。在main函数中，c=a+b等价c=a.operator+(b);

仿照上面双目运算符的重载方式，编写如下程序：

定义people类，具有数据成员name(姓名)、id(身份证号)，均为字符数组或string类型。定义成员函数：（1）带参构造函数，实现给数据成员赋初值；（2）重载运算符==，实现对象比较（如果两个people类对象的身份证id相等，则结果为1，否则结果为0）。在main函数中，通过if语句测试程序正确性。

（注意：如果数据成员为字符数组，则用标准库函数strcpy实现赋值、strcmp实现字符串大小的比较；如果数组成员为string类型，则用运算符=实现赋值、==实现大小比较，并添加#include <string>）

【上机要求】

无。

【上机指导】

（1）源程序：

#include<iostream>

#include<string>

using namespace std;

class People

{

public:

People() {};

People(string namw1, string id1);

~People();

int operator==(People&);

private:

string name;

string id;

};

People::People(string name1, string id1)

{

name = name1;

id = id1;

}

People::~People()

{

}

int People::operator==(People& c) {

if (name == c.name && id == c.id)

return 1;

else

return 0;

}

int main() {

string name, id;

cout << "please input name and id of people a:\n";

cin >> name >> id;

People a(name, id);

cout << "please input name and id of people b:\n";

cin >> name >> id;

People b(name, id);

if (a == b)

cout << 1 << endl;

else

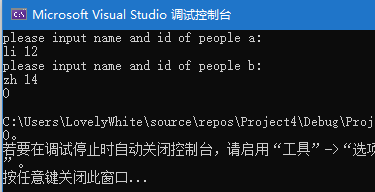
cout << 0 << endl;

return 0;

}

（2）

【运行结果】



【范例分析】

运算符重载是一种特殊的重载运算。函数名是由关键字 operator 和其后要重载的运算符符号构成的。与其他函数一样，重载运算符有一个返回类型和一个参数列表。

上机实训08：动态多态性

【上机目的】

1．了解多态性的概念。

2．进一步理解继承与派生的概念。

3．了解虚函数的作用及使用方法。

【实训内容】

（1）派生类和基类的成员函数同名的处理

#include<iostream>

#include<string>

using namespace std;

class A {

private:

int a;

public:

A(int x) { a=x; }

void setA(int x){ a = x; }

void show() { cout<<"基类A="<<a<<endl; }

};

class C:public A { //公有继承 A

private:

int c;

public:

C(int a,int b,int c1):A(a) { c=c1; } //派生类构造函数

void setC(int x,int y) { c = x; }

void showC() { cout<<"c="<<c<<endl; }

void show() { cout<<"派生类C="<<c<<endl; }

};

void main() {

A ta(12);

C tc(1,2,3);

ta.show();

tc.show();

}

回答：

1. main函数中的ta.show() 和tc.show()分别调用的哪个类中的show函数？

2)将C类中的show函数修改为：

void show() {

A::show();

cout<<"派生类C="<<c<<endl;

}

将main函数中的ta.show()语句注释掉，运行观察输出结果。说明在派生类中，调用基类的同名函数的调用方法。

（2）赋值兼容原则

#include<iostream>

#include<string>

using namespace std;

class A {

private:

int a;

public:

A(int x) { a = x; }

void setA(int x) { a = x; }

void show() { cout << "基类A=" << a << endl; }

};

class C :public A { //公有继承A

private:

int c;

public:

C(int a, int c1) :A(a) { c = c1; } //派生类构造函数

void setC(int x) { c = x; }

void showC() { cout << "c=" << c << endl; }

void show() { //A::show(); cout<<"派生类C="<<c<<endl;

}

};

void main() {

A ta(12), \* pa;

C tc(1, 3);

A& ya = tc; //赋值兼容原则

pa = &tc;

cout << "pa-show" << endl;

pa->show(); //输出什么结果

cout << "ya.show" << endl;

ya.show(); //输出什么结果

ta = tc;

cout << "ta.show" << endl;

ta.show(); //输出什么结果

}

1. 回答：派生类对象的地址是否可以赋值给基类的指针变量？派生类对象是否可以赋值给基类的引用对象和基类对象本身？
2. 预测输出结果，然后运行，说明main函数中的三个show函数分别调用的是哪个类中的show成员函数。

（3）编程实现

定义基类Base，其功能是计算图形的面积，它只有一个公有的成员虚函数area。从Base类公有派生三角形类Triangle和圆类Circle，在类Triangle和类Circle中，分别定义自己的area函数，用于计算各自的面积。在main函数中，声明Base类的对象指针变量，分别指向类Triangle和类Circle的对象，通过指针调用area函数显示相应对象的面积。（用虚函数实现多态）

【上机要求】

无。

【上机指导】

1.1答：ta.show()调用的基类A的show函数，tc.show()调用的派生类C的show函数；

1.2答：调用方法：类名::同名函数；

2.1答：①派生类对象的地址可以赋值给基类的指针变量；②派生类对象可以赋值给基类的引用和基类本身；

2.2答：都调用的基类的show函数；

3 源代码：

#include<iostream>

#include<cmath>

#include<iomanip>

using namespace std;

class Base {

public:

virtual double area() {

cout << "计算图形的面积" << endl;

return 0;

}

};

class Triangle :public Base {

private:

int a, b, c;

public:

double area() {

int p = (a + b + c) / 2;

double s = sqrt(p \* (p - a) \* (p - b) \* (p - c));

return s;

}

void set\_abc() {

cout << "input a,b,c:" << endl;

cin >> a >> b >> c;

}

};

class Circle :public Base {

int r;

public:

double area() {

const double pi = 3.1415;

return pi \* r\* r;

}

void set\_r() {

cout << "input r:" << endl;

cin >> r;

}

};

int main() {

//定义三角形对象并计算面积

Triangle\* pa = new Triangle();

pa->set\_abc();

cout << "area = " << pa->area() << endl;

//定义圆类对象并计算面积

Circle\* pb = new Circle();

pb->set\_r();

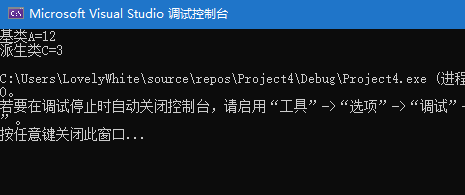
cout << "arae = " << pb->area() << endl;

return 0;

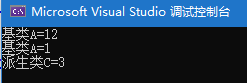
}

【运行结果】

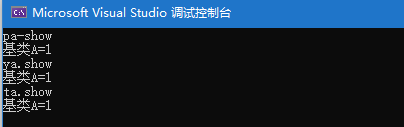
1.1



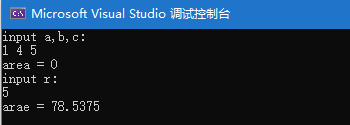
1.2

、

2.1



3



【范例分析】

第一个实验中当子类中出现与父类相同的函数时，声明类时调用的谁的构造函数就调用谁的同名函数。修改以后加入了A::show(); 这段函数的作用就是调用父类的show方法。这也说明了父类当中的相同函数并没有消失，而是被作用域隐藏了。

第二个实验得出的结论是子类的地址是可以赋值给父类指针的，当父类指针指向子类的时候调用的是父类的相同函数，当父类引用是子类的时候调用的也是父类的相同函数，当父类变量的值为子类时，调用的依然是父类的相同函数。

上机实训09：类属设计

【上机目的】

掌握类模板和模板类的用法。

【实训内容】

1.函数模板和模板函数

例子程序

#include<iostream>

using namespace std;

int fun1(int x, int y) { return x + y; }

float fun1(float x, float y) { return x + y; }

void main() {

int a1 = 2, b1 = 5;

float a2 = 5.3f, b2 = 4.3f;

cout << fun1(a1, b1) << endl << fun1(a2, b2) << endl;

}

1）以上程序改写为函数模板形式。

2）仿照以上例子编程：用函数模板实现函数max，用于找出2个整型数据的最大值和2个浮点型数据的最大值，在主函数中定义2个整型变量和两个浮点型变量，输入它们的数值，然后调用这两个函数。

2．掌握类模板和模板类的用法

(1) 观察下面的类Compare\_int和类Compare\_float的区别，了解类模板的使用。

#include <iostream>

using namespace std;

class Compare\_int {

public:

Compare\_int(int a,int b) {x=a;y=b;}

int max() { return (x>y)?x:y; }

int min() { return (x<y)?x:y; }

private:

int x,y;

};

class Compare\_float{

public:

Compare\_float(float a,float b) {x=a;y=b;}

float max() {return (x>y)?x:y;}

float min() {return (x<y)?x:y;}

private:

float x,y;

};

void main() {

Compare\_int cmp1(3,7);

Compare\_float cmp2(3.5,7.4);

cout<<cmp1.max()<<" is the Maximum of two inteder numbers."<<endl;

cout<<cmp1.min()<<" is the Minimum of two inteder numbers."<<endl<<endl;

cout<<cmp2.max()<<" is the Maximum of two inteder numbers."<<endl;

cout<<cmp2.min()<<" is the Minimum of two inteder numbers."<<endl<<endl;

}

比较以上的Compare\_int 和Compare\_float类，只有数据成员类型不同，其他函数功能都类似，像这样的类就可以把数据类型进行参数化，构建类模板如下：

#include <iostream>

using namespace std;

template<class T>

class Compare {

public:

Compare(T a,T b) {x=a;y=b;}

T max() {return (x>y)?x:y;}

T min() {return (x<y)?x:y;}

private:

T x,y;

};

void main() {

Compare<int> cmp1(3,7);

Compare<float> cmp2(3.5,7.5);

cout<<cmp1.max()<<" is the Maximum of two inteder numbers."<<endl;

cout<<cmp1.min()<<" is the Minimum of two inteder numbers."<<endl<<endl;

cout<<cmp2.max()<<" is the Maximum of two inteder numbers."<<endl;

cout<<cmp2.min()<<" is the Minimum of two inteder numbers."<<endl<<endl;

}

以上定义中，成员函数是放在类模板内定义的，其定义格式如下：

template<class numtype> //numtype 由用户指定 ，class是类型参数标志关键字

class 类名 {

numtype 成员函数名1（参数类别） { ….. }

numtype 成员函数名2（参数类别） { ….. }

}

(2)根据以上例子，对下面的imax类和fmax类用类模板进行改写。

#include<iostream>

using namespace std;

class imax {

private:

int x,y;

public:

imax(int x1,int y1) { x=x1; y=y1; }

int showmax() { return x>y?x:y; }

int showmin() { return x<y?x:y; }

};

class fmax {

private:

float x,y;

public:

fmax(float x1,float y1) { x=x1; y=y1; }

float showmax() { return x>y?x:y; }

float showmin() { return x<y?x:y; }

};

void main(void) {

imax a(3,5);

fmax b(5.0f,6.7f);

cout<<"a.max="<<a.showmax()<<endl;

cout<<"b.max="<<b.showmax()<<endl;

}

用类模板实现以上类，成员函数在类内部实现。

【上机要求】

无。

【上机指导】

1.1源程序：

#include<iostream>

using namespace std;

template<class T, class T1>

T fun1(T x, T y, T1 z) { return x + y + z; }

int main() {

int a1 = 2, b1 = 5;

float a2 = 5.3f, b2 = 4.3f;

cout << fun1(a1, b1, 5) << endl << fun1(a2, b2, 6) << endl;

cout << fun1<int,int>(a1, b1, 5) << endl << fun1<float,int>(a2, b2, 5) << endl;

return 0;

}

1.2源程序：

#include<iostream>

using namespace std;

template<class T>

T max(T a, T b) {

return(a > b ? a : b);

}

int main() {

int a, b;

float x, y;

cout << "input two number(int):" << endl;

cin >> a >> b;

cout << "max = " << max(a, b) << endl;

cout << "input two number(float):" << endl;

cin >> x >> y;

cout << "max = " << max(x, y) << endl;

return 0;

}

2.2源程序：

#include<iostream>

using namespace std;

template<class T>

class max {

private:

T x, y;

public:

max(T x1, T y1) {

x = x1;

y = y1;

}

T showmax() { return x > y ? x : y; }

T showmin() { return x < y ? x : y; }

};

void main(void) {

max<int> a(3, 5);

max<float> b(5.0f, 6.7f);

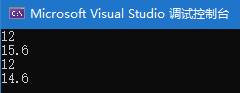
cout << "a.max=" << a.showmax() << endl;

cout << "b.max=" << b.showmax() << endl;

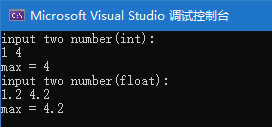
}

【运行结果】

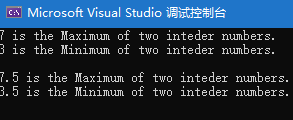
1.1



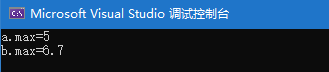
1.2



2.1



2.2



【范例分析】

在这里需要明确这些概念：

函数模板提供了一类函数的抽象，它提供了任意类型为参数及返回值。函数模板经实例化后生成的具体函数成为模板函数。函数模板代表了一类函数，模板函数表示某以具体函数。

函数模板的格式：

template <模板参数表>

返回值类型 函数名 ( 参数表 ) { //函数体 }

模板参数表的格式：

<class 类型参数1, class 类型参数2, …>

其中 class 是关键字，不是类标志。

上机实训10：输入输出流设计

【上机目的】

1．深入理解C++的输入输出含义及其实现方法

2．掌握文本文件的输入和输出

3．掌握二进制文件的输入和输出

【实训内容】

（1）实现将数据输出到文本文件中

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ofstream outfile("d:\\myfile2.txt",ios::app); //定义输出文件类对象，同时打开指定文件

if(outfile.fail()) { //fail函数判断是否打开成功

cout<<"D:\\file can not open"<<endl;

return;

}

float x=6, y=7;

outfile<<x<<" "<<y<<endl;

outfile.close();

return 0;

}

到D盘找到myfile2.txt，打开文件，观察里面的数据。

A．为什么outfile<<x<<" "<<y<<endl; x和y间需要输出空格

B．outfile.open("myfile2.txt", ios::app );表示什么意思？

（2）编程题：仿照以上例子，编程实现：输入5个整数，并保存到文本文件中。

（3）从文本文件中读取2个数据到变量中，仔细观察程序和运行过程

#include <iostream>

#include <fstream>

#include<string>

using namespace std;

int main() {

ifstream infile("myfile2.txt"); //定义输入文件类对象

float x , y;

infile>>x>>y;

cout<<x<<" "<<y;

infile.close();

return 0;

}

编程题：仿照以上例子，编程实现。从（1）中创建的文件中，把5个整数读取到变量或数组中，然后求和并输出到outfile1.txt文件中。（当不知道文件中存储的整数个数的时候，需要读出所有数据。改写以上代码）

提示：

while ( infile>>x ) { cout<<x<<" "<<y<<endl; }

或者

infile>>x; //必须先读取一次，否则读出来的数据会多一个

while(!infile.eof()) {

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

infile>>x;

}

【上机要求】

无。

【上机指导】

1.A：答：区分两个数据，避免输出为一个数67；

1.B: 答：打开"myfile2.txt"这个文件，并在其末尾追加内容；

1.2源程序：

#include<iostream>

#include<fstream>

using namespace std;

int main() {

fstream outfile1("outfile1.txt", ios::app);

if (outfile1.fail())

{

cout << "open file failure!" << endl;

return 0;

}

ifstream inflile("test\_file.txt");

int\* a = new int[5];

inflile >> a[0];

int i = 1;

while (!inflile.eof())

{

inflile >> a[i++];

}

int n = i - 1;

int sum = 0;

cout << "the value from test\_file:" << endl;

for (i = 0; i < n; i++) {

sum += a[i];

cout << a[i] << " ";

}

cout << endl;

cout << "sum = " << sum << endl;

delete[] a;

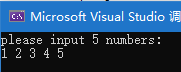
inflile.close();

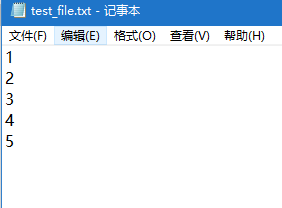
return 0;

}

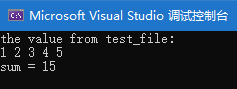
【运行结果】

1.1





1.2



【范例分析】

第一题要求将数据输出到文件当中，这里我们可以直接使用<<运算符，<<运算符已经重载了输出文件的功能。它的左侧为输出流对象，右侧为要输出的内容。所以我们先要创建一个输出流对象 这里通过ofstream的构造函数创建了一个输出流对象，然后对输出流对象进行操作，最后别忘了调用close函数关闭输出流。

第二题与第一题很类似，只不过这里输出到文件的数据是从键盘输入的。

第三题和第一题相反，是从文件中读取数据，相同的我们需要实例化一个输入流对象，然后使用>>重载运算符，向内存中输入数据。

附：文件的打开方式：

