第 3 章　实验内容与指导

实验一　熟悉实验环境

【实验目的】

1．了解和使用 VC 集成开发环境。
2．熟悉 VC 环境的基本命令和功能键。
3．熟悉常用的功能菜单命令。
4．学习使用 VC++环境的帮助。
5．熟悉完整的 C++程序开发过程。
6．理解简单的 C++程序结构。

【实验内容】

1．熟悉 Visual C++实验环境。

2．控制台应用。用 AppWizard 建立一个控制台应用，在终端上输出中“Hello”。

术语：“控制台应用程序”是一个在 DOS 窗口中运行的基于字符的程序。由于这种模式的应用程序比 Windows 程序简单，先选择利用 Visual C++来建立这样一个应用，这样使得我们可以将精力先投入到学习使用 C++编程语言，而不需要把过多的精力投入到学习复杂的 Windows 编程中去。

3．简单的 C++输入输出应用练习。输入并运行下列程序：

```
#include<iostream.h>
int main()
{
    int x,y,z;
    cin>>x>>y;
    z=x+y;
    cout<<"x+y="<<z<<endl;
    return 0;
}
```

【实验指导】

1．熟悉 Visual C++环境

（1）启动 Visual C++ 6.0 开发环境，单击“开始”菜单中的“程序”，选择 Microsoft Visual Studio 6.0 中的 Microsoft Visual C++ 6.0 选项，就可以启动集成开发环境 Developer Studio。看看初始化界面由哪些部分组成。

（2）查看各菜单项，看看都有哪些子菜单和命令。

（3）将鼠标放置于各工具栏图标上，系统会自动显示该图标代表的命令含义，了解一下都有哪些命令。

（4）在任意工具栏上单击鼠标右键，弹出式菜单上将显示所有可用的工具栏，选择其中没有用对号（√）标记的项，看看有什么效果，再选择有用对号标记的项，又有什么效果？

将鼠标移动到任意工具栏上，将鼠标放到图标间隙，按下鼠标左键不放，移动鼠标到屏幕中间，观察有什么现象发生？再将它拖回到原来位置，观察有什么现象发生？

将鼠标移动到左边的工作区窗口，按下鼠标左键不放，移动鼠标到屏幕中间，观察有什么现象发生？再将它拖回到原来位置，观察有什么现象发生？

（5）学习使用帮助系统。选择 Help（帮助）菜单中的 Contents（目录）选项，启动 MSDN 联机帮助系统，学习使用该帮助系统。联机帮助系统是一个相对独立的程序，它和 Developer Studio 是两个程序，但是它的启动和停止都受 Developer Studio 影响。MSDN 联机帮助系统运行的前提条件是 Developer Studio 在运行。

（6）选择 File（文件）菜单中的 Exit（退出）菜单项，退出 Developer Studio。

2. 用 AppWizard 创建一个控制台应用

（1）启动 Visual C++ 6.0 开发环境

单击“开始”菜单中的“程序”，选择 Microsoft Visual Studio 6.0 中的 Microsoft Visual C++ 6.0 选项，就可以启动集成开发环境 Developer Studio。

（2）新建工程文件

选择 File（文件）菜单中的 New（新建）命令，打开 New（新建）对话框，选择该对话框的 Projects（工程）选项卡，选择对话框左边的类型列表中的 Win32 Console Application 项，对话框右边的 Projects name:（工程名称）和 Location（位置）文本框中分别输入工程名和工程所存放的文件夹路径。如工程名为 Test1，存放在 D 盘上。单击 OK（确定）按钮，会出现 Win32 Console Application Step 1 of 1 对话框，该对话框主要向用户询问“创建什么类型的控制台程序？”，默认情况下已选择了 An Empty Project（一个空工程）单选按钮，因此不用更改，单击 Finish（完成）按钮，会出现 New Project Information 对话框，这是一个关于新建工程信息的简单列表，单击 OK 按钮完成 Test1 工程的创建工作。

（3）编辑程序文件

单击 File（文件）菜单中的 New（新建）菜单项，屏幕出现 New（新建）对话框，在该对话框中选择 Files（文件）选项卡。选择对话框左边的类型列表中的 C++ Source File 来创建 C++源程序。在 File（文件名）文本框中输入源程序文件名称（如 file1，扩展名可以省略，默认为.cpp），并在 Location（位置）文本框中选择要保存的文件路径，并确认 Add to Projects 复选框被选中，该复选框下面的下拉列表框中显示的工程名是 Test。如果不是，可以修改。

单击 OK（确定）按钮，该源程序文件（file1.cpp）被加入到工程 Test 中。这时就产生了一个源程序文件编辑窗口。

在源程序编辑窗口区输入如下源程序代码：

```
#include<iostream.h>
void main()
{
    cout<<"Hello"<<endl;
}
```

源程序代码输入完后，选择 File（文件）菜单中的 Save（保存）命令，将文件进行保存。

（4）编译和链接程序

单击 Build（编译链接）菜单中的 Compile[file1.cpp]命令，或按快捷键 Ctrl+F7 对程序进行编译。编译过程中，系统将发现的错误显示在屏幕下方的 Build 窗口中。用户可以根据这些错误信息进行修改。若没有错误，窗口显示信息如下：

```
Test.exe - 0 error(s), 0 warning(s)
```

（5）运行程序

当编译链接成功后，选择 Build（编译链接）菜单中的 Execute file1.exe，或按快捷键 Ctrl+F5 对程序进行运行。

运行该执行文件，结果显示在另一个显示执行文件输出结果的窗口中。

另外，在源程序输入完成后，也可以直接单击工具栏中的“!”按钮进行编译、链接和运行。

3. 简单的 C++输入输出应用练习

程序运行结果为：

```
12 34
x+y=46
```

实验二　C++基础练习

【实验目的】

1．掌握 C++程序的基本格式与规范，学会编写简单的 C++程序。

2．熟悉 C++程序基本的输入输出操作。

3．比较 C++与 C 语言中定义常量的不同方法。

4．掌握重载函数的概念及函数参数在重载函数中的作用。

5．熟悉作用域标识符的功能和基本使用方法。

6．掌握内联函数的使用方法。

7．理解引用的概念，掌握引用的使用方法。

8．掌握 C++字符串变量的定义和使用。

9．掌握 C++内存的动态分配与释放方法。

【实验内容】

1．编写一个简单程序，输出“Hello！C++”，并给程序加一行注释“fisrt C++ program”。

2．编写一个完整的包含输入和输出的简单 C++程序。

3．分别使用#define 和 const 定义圆周率 π=3.1415926 后，编写一个函数求圆的面积，并在主函数中输出面积。

4．利用函数重载，求分别输入的三个整型数、浮点数和长整型数中的最大数。

5．编写程序对一个整数数组求和，求和的结果使用外部变量 sum 存储，同时对整数中的奇数求和，结果使用内部变量 sum 存储，在主程序中输出两个结果。本题要求体会和理解作用域标识符的概念与基本使用方法。

6．编写内联函数求解 n！的值，n 为整数，并用主函数调用该函数。

7．编写程序，在主函数中输入两个整型数据，赋值给两个整型变量。通过函数调用，交换两个变量的值，参数传递采用引用传递方式，显示交换前和交换后的变量的值。

8．有 7 个字符串，要求对它们按由小到大的顺序排列，用 string 方法。

9．设计一个描述学生的结构类型。结构中应当包括学生的学号、姓名、性别、年龄、家庭住址等信息。编写程序为该结构动态分配内存，然后给出结构中所有成员的值并且显示输出。

【实验指导】

1．参考程序为：

```
//fisrt C++ program
#include<iostream.h>
int main()
{
    cout<<"Hello! C++"<<endl;
    return 0;
}
```

2．参考程序为：

```
#include<iostream.h>
int main()
{
    int x,y,z;
    cin>>x>>y>>z;
    cout<<"x="<<x<<endl;
    cout<<"y="<<y<<endl;
    z=x+y;
    cout<<"z(x+y)="<<z<<endl;
    return 0;
}
```

3．用#define 定义常量π的参考程序为：

```
#include<iostream.h>
#define PI 3.1415926
int main()
{
    double Area(int radio);//声明函数
    cout<<Area(2)<<endl;
    return 0;
}
double Area(int radio) //函数实现，计算圆面积
{
    return PI*radio*radio;
}
```

用 const 定义常量π的参考程序为：

```
#include<iostream.h>
const double PI=3.1415926;
```

```
int main()
{
    double Area(int radio);//声明函数
    cout<<Area(2)<<endl;
    return 0;
}
double Area(int radio) //函数实现，计算圆面积
{
    return PI*radio*radio;
}
```

说明：C 中是使用宏#define 定义常量，C++新增了用 const 来定义。二者的区别如下：

（1）const 是有数据类型的常量，而宏常量没有，编译器可以对前者进行静态类型安全检查，对后者仅是字符替换，没有类型安全检查，而且在字符替换时可能会产生意料不到的错误（边际效应）。

（2）有些编译器可以对 const 常量进行调试，不能对宏调试。

4．参考程序为：

```
#include<iostream.h>
int Max(int a,int b,int c)
{
    int max;
    max=a;
    if(b>max)
        max=b;
    if(c>max)
        max=c;
    return max;
}
long Max(long a,long b,long c)
{
    long max;
    max=a;
    if(b>max)
        max=b;
    if(c>max)
        max=c;
    return max;
}
float Max(float a,float b,float c)
{
    float max;
    max=a;
    if(b>max)
        max=b;
    if(c>max)
        max=c;
    return max;
```

```
}
int main()
{
    int x,y,z;
    long a,b,c;
    float m,n,k;

    cout<<"请输入三个整型数：";
    cin>>x>>y>>z;
    cout<<"三个整型数中最大值为："<<Max(x,y,z)<<endl;
    cout<<"请输入三个长整数：";
    cin>>a>>b>>c;
    cout<<"三个长整数中最大值为："<<Max(a,b,c)<<endl;
    cout<<"请输入三个浮点数：";
    cin>>m>>n>>k;
    cout<<"三个浮点数中最大值为："<<Max(m,n,k)<<endl;
    return 0;
}
```

5．参考程序为：

```
#include<iostream.h>
long sum = 0;
int main()
{
    long sum = 0;
    const int inArray[]={1,4,5,32016,64,642,70,540,45,608,403};
    for(int i=0; i<sizeof(inArray)/sizeof(int);++i)
    {
        ::sum+=inArray[i];
        if(inArray[i]%2==0)
            continue;
        else
            sum+= inArray[i];
    }
    cout<<"奇数和为: "<<sum<<endl;
    cout<<"总和为: "<<::sum<<endl;
    return 0;
}
```

注意：代码 sizeof(inArray)/sizeof(int)用来求整型数组 inArray 中元素的个数。

6．参考程序为：

```
#include<iostream.h>
inline int sum(int n)//定义内联函数
{
    int X=1;
    for(int i=0;i<n;i++)
        X=X*n;
    return X;
```

```
}
int main()
{
    cout<<"输入 n 的值：";
    int n;
    cin>>n;
    cout<<"n 的阶乘为："<<sum(n)<<'\n';
    return 0;
}
```

7．参考程序为：

```
#include<iostream.h>
void swap(int &a,int &b)
{
    int temp;
    temp=a;
    a=b;
    b=temp;
}
int main()
{
    int a,b;
    cout<<"输入两个整数：";
    cin>>a>>b;
    cout<<"a="<<a<<endl;
    cout<<"b="<<b<<endl;
    swap(a,b);
    cout<<"交换后..."<<endl;
    cout<<"a="<<a<<endl;
    cout<<"b="<<b<<endl;
    return 0;
}
```

8．参考程序为：

```
#include<iostream>
#include <string>
using namespace std;
int main(){
    string x[7],t;
    //输入 string
    for(int i=0;i<7;i++)
        cin>>x[i];
    //用冒泡法对数组 X 排序
    for(i=0;i<7;i++)
    {
        for(int j=i+1;j<7;j++)
        {
            if(x[i]>x[j])
            {
```

```
                    t=x[i];x[i]=x[j];x[j]=t;
                }
            }
        }
        for(i=0;i<7;i++)
            cout<<x[i]<<endl;
        return 0;
    }
```

说明：string 是 C++的一个类，而不是真正意义上的变量类型，所以必须使用 iostream 和 string 不带.h 的头文件，作为 C++的头文件必须使用命名空间“using namespace std;”来声明，不能省略，否则编译报错。

9．参考程序为：

```
#include<iostream>
#include<string>
using namespace std;
struct student{
    int     id;                 //学号
    char    name[10];           //姓名
    char    sex;                //性别
    int     age;                //年龄
    char    address[30];        //家庭地址
};
int main()
{
    struct student *st;
    st=new struct student;
    cout<<"请输入..."<<endl;
    cout<<"姓名:";
    cin.getline(st->name,10);
    cout<<"地址:";
    cin.getline(st->address,30);
    cout<<"学号:";
    cin>>st->id;
    cout<<"性别:";
    cin>>st->sex;
    cout<<"年龄:";
    cin>>st->age;
    cout<<"显示学生信息"<<endl;
    cout<<"学号："<<st->id<<endl;
    cout<<"姓名："<<st->name<<endl;
    cout<<"性别："<<st->sex<<endl;
    cout<<"年龄："<<st->age<<endl;
    cout<<"地址："<<st->address<<endl;
    delete st;
    return 0;
}
```

说明：由于姓名和地址都是包含空格的字符串，为了保证空格的正常输入，采用 cin 流对象的成员函数 getline 来完成，它的语法为：

cin.getline(字符串变量,变量大小);

功能：从键盘输入一个以 ENTER 键标志结束的字符串。

实验三　类和对象（一）

【实验目的】

1．掌握类、类的数据成员、类的成员函数的定义方法。
2．理解类成员的访问控制方式。
3．掌握对象的定义和操作对象的方法。
4．理解构造函数和析构函数的定义与执行过程。
5．掌握重载构造函数的方法。
6．了解拷贝构造函数的方法。

【实验内容】

1．定义一个学生类，其中有 3 个数据成员包括学号、姓名、年龄，以及若干成员函数。同时编写主函数使用这个类，实现对学生数据的赋值和输出。

要求：

（1）使用成员函数实现对数据的输入、输出。

（2）使用构造函数和析构函数实现对数据的输入、输出。

2．声明一个时间类，时间类中有 3 个私有数据成员（hour、minute、second）和两个公有成员函数（settime 和 print_time）。settime 根据传递的 3 个参数为对象设置时间；print_time 负责将对象表示的时间显示输出。在主函数中，建立一个时间类的对象，设置时间为 9 点 20 分 30 秒并显示该时间。

3．使用构造函数代替上面的 settime 成员函数，并在主函数中使用构造函数设置时间为 10 点 40 分 50 秒，并显示该时间。

4．重载时间类的构造函数（不带参数）使时、分、秒缺省均为 0。

5．在时间类的析构函数中输出“goodbye!”。

6．对时间类定义拷贝构造函数并在主函数中调用。

7．在三种情况下系统自动调用拷贝构造函数：（1）当用类的一个对象去初始化该类的另一个对象时；（2）如果函数的形参是类的对象，调用函数进行形参和实参的结合时；（3）如果函数的返回值是类的对象，函数执行完成返回调用者时。针对这三种情况分别编写程序来说明。

8．定义一个类实现银行账户的概念，包括的变量有“账号”和“存款余额”，包括的方法有“存款”、“取款”和“查询余额”。在主函数中创建账户类的对象，并完成相应操作。

【实验指导】

1．使用成员函数实现输入输出的参考程序为：

```
#include<iostream.h>
class student{
public:
    void Input_Info()
    {
        cout<<"请输入..."<<endl;
        cout<<"姓名:";
        cin.getline(name,10);
        cout<<"学号:";
        cin>>id;
        cout<<"年龄:";
        cin>>age;
    }
    void Show_Info()
    {
        cout<<"显示学生信息"<<endl;
        cout<<"学号："<<id<<endl;
        cout<<"姓名："<<name<<endl;
        cout<<"年龄："<<age<<endl;
    }
private:
    int id;
    char name[10];      //姓名
    int   age;          //年龄
};
int main()
{
    student st;
    st.Input_Info();
    st.Show_Info();
    return 0;
}
```

使用构造析构函数实现的参考程序为：

```
#include<iostream.h>
class student{
public:
    student()
    {
        cout<<"请输入..."<<endl;
        cout<<"姓名:";
        cin.getline(name,10);
        cout<<"学号:";
        cin>>id;
```

```
        cout<<"年龄:";
        cin>>age;
    }
    ~student()
    {
        cout<<"显示学生信息"<<endl;
        cout<<"学号："<<id<<endl;
        cout<<"姓名："<<name<<endl;
        cout<<"年龄："<<age<<endl;
    }
private:
    int id;
    char name[10];      //姓名
    int  age;           //年龄
};
int main()
{
    student st;
    return 0;
}
```

2．参考程序为：

```
#include <iostream.h>
class time{
public:
    void settime(int c,int m,int s)
    {
        clock=c;
        minute=m;
        second=s;
    }
    void print_time()
    {
        cout<<clock<<":"<<minute<<":"<<second<<endl;
    }
private:
    int clock,minute,second;
};
void main()
{
    time time1;
    time1.settime(9,20,30);
    time1.print_time();
}
```

3．参考程序为：

```
#include <iostream.h>
class time{
```

```
public:
    time(int c,int m,int s)
    {
        clock=c;
        minute=m;
        second=s;
    }
    void print_time()
    {
        cout<<clock<<":"<<minute<<":"<<second<<endl;
    }
private:
    int clock,minute,second;
};
void main()
{
    time time1(10,40,50);
    time1.print_time();
}
```

4．参考程序为：

```
#include <iostream.h>
class time{
public:
    time()
    {
        clock=0;
        minute=0;
        second=0;
    }
    time(int c,int m,int s)
    {
        clock=c;
        minute=m;
        second=s;
    }
    void print_time()
    {
        cout<<clock<<":"<<minute<<":"<<second<<endl;
    }
private:
    int clock,minute,second;
};
void main()
{
    time time1;
    time1.print_time();
```

```
    time time2(9,20,30);
    time2.print_time();
}
```

5．参考程序为：

```
#include <iostream.h>
class time{
public:
    time()
    {
        clock=0;
        minute=0;
        second=0;
    }
    time(int c,int m,int s)
    {
        clock=c;
        minute=m;
        second=s;
    }
    void print_time()
    {
        cout<<clock<<":"<<minute<<":"<<second<<endl;
    }
    ~time()
    {
        cout<<"good bye"<<endl;
    }
private:
    int clock,minute,second;
};
void main()
{
    time time1;
    time1.print_time();
    time time2(9,20,30);
    time2.print_time();
}
```

6．参考程序为：

```
#include <iostream.h>
class time{
public:
    time()
    {
        clock=0;
        minute=0;
        second=0;
```

```
    }
    time(int c,int m,int s)
    {
        clock=c;
        minute=m;
        second=s;
    }
    void print_time()
    {
        cout<<clock<<":"<<minute<<":"<<second<<endl;
    }
    ~time()
    {
        cout<<"good bye"<<endl;
    }
    time (const time&p)
    {
        clock=p.clock;
        minute=p.minute;
        second=p.second;
    }
private:
    int clock,minute,second;
};
void main()
{
    time time1;
    time1.print_time();
    time time2(9,20,30);
    time2.print_time();
    time time3(9,20,30);
    time time4(time3);
    time3.print_time();
    time4.print_time();
}
```

7．用类的对象进行初始化时，系统自动调用拷贝构造函数的参考程序为：

```
#include <iostream.h>
class Point{
public:
    Point(int xx = 0, int yy = 0)
    {
        X = xx; Y = yy;
        cout<<"调用构造函数\n";
    }
    Point(Point &p);
    int GetX(){ return X; }
```

```
    int GetY(){ return Y; }
private:
    int X,Y;
};
Point::Point(Point & p)
{
    X = p.X;
    Y = p.Y;
    cout<<"调用拷贝构造函数\n";
}
void main()
{
    Point A(3,4);
    Point B(A); //也可以写成 Point B=A;
    cout<<B.GetX()<<"\n";
}
```

调用形参是类的对象的函数时，系统自动调用拷贝构造函数的参考程序为：

```
#include <iostream.h>
class Point{
public:
    Point(int xx = 0, int yy = 0)
    {
        X = xx; Y = yy;
        cout<<"调用构造函数\n";
    }
    Point(Point &p);
    int GetX(){ return X; }
    int GetY(){ return Y; }
private:
    int X,Y;
};
Point::Point(Point & p)
{
    X = p.X;
    Y = p.Y;
    cout<<"调用拷贝构造函数\n";
}
void fun1(Point p)
{
    cout<<p.GetX()<<"\n";
}
void main()
{
    Point A(4,5);
    Point B(A);
    cout<<B.GetX()<<"\n";
```

```
    fun1(B);              //调用拷贝构造函数，实现形参和实参的结合
}
```

函数的返回值是类的对象，函数执行完返回时系统自动调用拷贝构造函数的参考程序为：

```
#include <iostream.h>
class Point{
public:
    Point(int xx = 0, int yy = 0)
    {
        X = xx; Y = yy;
        cout<<"调用构造函数\n";
    }
    Point(Point &p);
    int GetX(){ return X; }
    int GetY(){ return Y; }
private:
    int X,Y;
};
Point::Point(Point & p)
{
    X = p.X;
    Y = p.Y;
    cout<<"调用拷贝构造函数\n";
}
Point fun2()
{
    Point Temp(10,20);          //调用构造函数
    return Temp;
}
void main()
{
    Point A(4,5);
    Point B(A);
    cout<<B.GetX()<<"\n";
    B = fun2();
    cout<<B.GetX()<<"\n";
}
```

8．参考程序为：

```
#include<iostream.h>
#include<string.h>
class BankAccount{
public:
    BankAccount(char *acno,double amt)
    {
        strcpy(Acc_no,acno);
        leftmoney=amt;
    }
```

```
    double getleftmoney()
    {
        return leftmoney;
    }
    void savemoney(double money)
    {
        leftmoney+=money;
    }
    void getmoney(double money)
    {
        if(money<=leftmoney)
            leftmoney-=money;
        else
            cout<<"只能取："<<leftmoney<<endl;
    }
private:
    char Acc_no[20];
    double   leftmoney;
};
void main()
{
    BankAccount ba("888123",1000);
    ba.savemoney(21000);
    cout<<"存入 21000 元后余额为："<<ba.getleftmoney()<<endl;;
    ba.getmoney(11500);
    cout<<"取出 11500 元后余额为："<<ba.getleftmoney()<<endl;
    ba.getmoney(20000);
}
```

实验四　类和对象（二）

【实验目的】

1．掌握对象数组的定义与使用方法。
2．理解对象指针的概念，学会使用指针引用对象。
3．了解 this 指针的工作方式。
4．掌握静态数据成员和静态成员函数的基本使用方法。
5．理解友元与友元函数的作用，掌握其使用方法。

【实验内容】

1．建立一个对象数组，存放 5 个学生的数据（学号、成绩），定义函数 max，用指向对象的指针作为函数参数，在 max 函数中找出 5 个学生成绩中的最高者。

2．定义一个圆类，计算圆的面积和周长。

要求：分别用成员函数和友元函数来求圆的面积和周长。

3．定义类 stock，记录一支股票交易的基本信息，包括交易日期序号（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。再定义一个对象数组存放连续 5 个交易日的股票信息。用指针引用对象数组中的两个对象，在主函数中计算从第 2 个交易日开始每天的当日涨幅。

4．在 stock 类中定义一个静态数据成员，记录当前 stock 对象的数量。

5．设计一个成员函数 assign_stock 对对象赋值，其中的形参是对另一个 stock 对象的引用，使用 this 指针避免对自己赋值，在主函数中显示用 assign_stock 赋值的对象。

6．定义一个友元函数计算 stock 对象的当日开盘价是否高于当日收盘价。若是返回真，否则返回假。

【实验指导】

1．参考程序为：

```
#include<iostream.h>
class Student{
private:
    int id;
    float score;
public:
    Student()
    {
        id = 0;
        score = 0;
    }
    Student(int id, float score)
    {
        this->id = id;
        this->score = score;
    }
    float getScore()
    {    return score;}
    int getID()
    {    return id;}
};
void max(Student* s, int size)
{
    if(s==NULL || size < 1)
        return;
    int max_i=0;
    for(int i = 1; i < size; i++)
    {
        if (s[i].getScore()>s[max_i].getScore())
            max_i = i;
    }
```

```
        cout << "Student with ID "<<s[max_i].getID()<<" has the largest grade."<<endl;
}
int main()
{
        const int num = 5;
        Student students[num]={Student(1001, 78),Student(1002, 92),Student(1004, 81),
              Student(1005, 89),Student(1006, 68)};
        max(students, 5);
        return 0;
}
```

2．参考程序为：

```
#include<iostream.h>
const double PI=3.14159;
class circle{
public:
        circle()
        {       r=0;       }
        circle(float x)
        {       r=x;       }
        double get_delimeter()
        {       return 2*PI*r;       }
        double get_area()
        {       return PI*r*r;       }
        friend void cal_circle(circle &ci);
private:
        float r;
};
void cal_circle(circle &ci)
{
        cout<<"友元函数计算圆周长为："<<2*PI*ci.r<<endl;
        cout<<"友元函数计算圆面积为："<<PI*ci.r*ci.r<<endl;
}
int main()
{
        circle object(5);
        cout<<"成员函数计算圆周长为："<<object.get_delimeter()<<endl;
        cout<<"成员函数计算圆面积为："<<object.get_area()<<endl;
        cal_circle(object);
        return 0;
}
```

3．参考程序为：

```
#include<iostream.h>
const n=5;
class stock{
public:
        stock() {}
```

```
        stock(int n,float ma,float mi,float b,float e);
        void set_stock(int n,float ma,float mi,float b,float e);
        void set_stock();
        float get_end();
        void show_stock();
private:
        int number;
        float max,min,begin,end;
};
stock::stock(int n,float ma,float mi,float b,float e)
{
        number=n;max=ma;
        min=mi;begin=b;
        end=e;
}
void stock::set_stock(int n,float ma,float mi,float b,float e)
{
        number=n;max=ma;
        min=mi;begin=b;end=e;
}
float stock::get_end()
{
        return    end;
}
void stock::show_stock()
{
        cout<<number<<"\t";
        cout<<max<<"\t";
        cout<<min<<"\t";
        cout<<begin<<"\t";
        cout<<end<<endl;
}
void stock::set_stock()
{
        cout<<"number:";
        cin>>number;
        cout<<"max:";
        cin>>max;
        cout<<"min:";
        cin>>min;
        cout<<"begin:";
        cin>>begin;
        cout<<"end:";
        cin>>end;
}
void main()
```

```
{
    int   i;
    stock s1[5];
    stock   *p;
    for(i=0, p=s1; i<n; i++, p++)
        p->set_stock();
    for(i=0, p=s1; i<n; i++, p++)
        p->show_stock();
    for(i=1, p=s1+1; i<n; i++, p++)
        cout<<"\n"<< (p->get_end()-(p-1)->get_end())/(p-1)->get_end()*100<<"%";
}
```

4．参考程序为：

```
#include<iostream.h>
const n=5;
class stock{
public:
    stock() {}
    stock(int n,float ma,float mi,float b,float e);
    void set_stock(int n,float ma,float mi,float b,float e);
    void set_stock();
    float get_end();
    void show_stock();
    int   get_n_count();
private:
    static   int   n_count;
    int number;
    float max,min,begin,end;
};
int   stock::n_count=0;
stock::stock(int n,float ma,float mi,float b,float e)
{
    n_count++;
    number=n;max=ma;
    min=mi;begin=b;
    end=e;
}
void stock::set_stock(int n,float ma,float mi,float b,float e)
{
    n_count++;
    number=n;max=ma;
    min=mi;begin=b;end=e;
}
float stock::get_end()
{
    return   end;
}
```

```
void stock::show_stock()
{
    cout<<number<<"\t";
    cout<<max<<"\t";
    cout<<min<<"\t";
    cout<<begin<<"\t";
    cout<<end<<endl;
}
void stock::set_stock()
{
    n_count++;
    cout<<"number:";
    cin>>number;
    cout<<"max:";
    cin>>max;
    cout<<"min:";
    cin>>min;
    cout<<"begin:";
    cin>>begin;
    cout<<"end:";
    cin>>end;
}
int stock::get_n_count()
{
    return n_count;
}
void main()
{
    int   i;
    stock s1[5];
    stock   *p;
    for(i=0, p=s1; i<n; i++, p++)
        p->set_stock();
    for(i=0, p=s1; i<n; i++, p++)
        p->show_stock();
    for(i=1, p=s1+1; i<n; i++, p++)
        cout<<"\n"<<(p->get_end()-(p-1)->get_end())/(p-1)->get_end()*100<<"%";
    cout<<"\n"<<p->get_n_count()<<endl;
}
```

5．参考程序为：

```
#include<iostream.h>
class stock{
public:
    stock() {}
    stock(int n,float ma,float mi,float b,float e);
    void set_stock(int n,float ma,float mi,float b,float e);
```

```
        void set_stock();
        float get_end();
        void show_stock();
        void assign_stock(stock& p);
private:
        static  int  n_count;
        int number;
        float max,min,begin,end;
};
int  stock::n_count=0;
stock::stock(int n,float ma,float mi,float b,float e)
{
        n_count++;
        number=n;max=ma;
        min=mi;begin=b;
        end=e;
}
void stock::set_stock(int n,float ma,float mi,float b,float e)
{
        n_count++;
        number=n;max=ma;
        min=mi;begin=b;end=e;
}
float stock::get_end()
{
        return  end;
}
void stock::show_stock()
{
        cout<<number<<"\t";
        cout<<max<<"\t";
        cout<<min<<"\t";
        cout<<begin<<"\t";
        cout<<end<<endl;
}
void stock::set_stock()
{
        n_count++;
        cout<<"number:";
        cin>>number;
        cout<<"max:";
        cin>>max;
        cout<<"min:";
        cin>>min;
        cout<<"begin:";
        cin>>begin;
```

```
    cout<<"end:";
    cin>>end;
}
void stock::assign_stock (stock& p)
{
    if   (this!=&p)
    {
        n_count++;
        number=p.number;
        max=p.max;
        min=p.min;
        begin=p.begin;
        end=p.end;
    }
}
void main()
{
    stock s1(4,7.88,7.48,7.56,7.68);
    stock s2;
    s2.assign_stock(s1);
    s2.show_stock();
}
```

6．参考程序为：

```
#include<iostream.h>
const n=5;
class stock{
public:
    stock() {}
    stock(int n,float ma,float mi,float b,float e);
    void set_stock(int n,float ma,float mi,float b,float e);
    void set_stock();
    float get_end();
    void show_stock();
    friend int get_stock(stock *s1);
private:
    int number;
    float max,min,begin,end;
};
stock::stock(int n,float ma,float mi,float b,float e)
{
    number=n;max=ma;
    min=mi;begin=b;
    end=e;
}
void stock::set_stock(int n,float ma,float mi,float b,float e)
{
```

```
    number=n;max=ma;
    min=mi;begin=b;end=e;
}
float stock::get_end()
{
    return   end;
}
void stock::show_stock()
{
    cout<<number<<"\t";
    cout<<max<<"\t";
    cout<<min<<"\t";
    cout<<begin<<"\t";
    cout<<end<<endl;
}
void stock::set_stock()
{
    cout<<"number:";
    cin>>number;
    cout<<"max:";
    cin>>max;
    cout<<"min:";
    cin>>min;
    cout<<"begin:";
    cin>>begin;
    cout<<"end:";
    cin>>end;
}
int get_stock(stock *s1)
{
    cout<<endl<<s1->begin<<"\t"<<s1->end;
    if ((s1->begin)>(s1->end))
        return 1;
    else
        return 0;
}
void main()
{
    int   i;
    stock s1[5];
    stock   *p;
    for(i=0, p=s1; i<n; i++, p++)
        p->set_stock();
    for(i=0, p=s1; i<n; i++, p++)
        p->show_stock();
    for(i=1, p=s1+1; i<n; i++, p++)
```

```
            cout<<"\n"<<(p->get_end()-(p-1)->get_end())/(p-1)->get_end()*100<<"%";
        for(i=0,p=s1;i<n;i++,p++)
            cout<<"\n"<<get_stock(p)<<endl;
    }
```

实验五　派生和继承

【实验目的】

1．理解类的继承的概念，能够定义和使用类的继承关系。
2．掌握派生类的声明与定义方法。
3．熟悉公有派生、保护派生和私有派生的访问特性。
4．理解虚基类在解决二义性问题中的作用。

【实验内容】

1．仔细阅读下列程序，写出运行结果。

```
#include <iostream.h>
class A{
public:
    A(int m){ cout<<"A constructor:"<<m<<endl; }
    ~A(){ cout<<"A destructor"<<endl; }
};
class B{
public:
    B(int n){ cout<<"B constructor:"<<n<<endl;}
    B(){cout<<"B destructor"<<endl;}
};
class C: public B,public A{
public:
    C(int a,int b,int c,int d,int e):A(b),B(e),bb(c),aa(d)
    { cout<<"C constructor:"<<a<<endl; }
    ~C(){ cout<<"C destructor"<<endl; }
private:
    A aa;
    B bb;
};
int main()
{
    C cc(1,2,3,4,5);
}
```

2．定义 Person 类，数据成员包含姓名变量 name，以及输出姓名的成员函数 PrintName()。再从 Person 类派生出 Worker 类，该类包括数据成员 number 用来记录对象的工号、sex 用来记录对象的性别、age 用来记录对象的年龄、add 用来记录对象的家庭住址；包括函数成员

printinfo()用来输出对象的个人信息。

要求：

（1）构造 Worker 类对象输出该对象的工号、年龄、家庭住址等信息。

（2）在 Worker 类的 printinfo()成员函数中须调用 Person 类的成员函数 PrintName()。

3．定义基类 myarray 来存放一组整数，基类中有构造函数、析构函数、输入数据和输出数据的成员函数。

4．定义类 sortarray 继承自 myarray，在该类中定义成员函数实现排序功能。

5．定义类 rearray 继承自 myarray，在该类中定义成员函数实现逆转功能。

6．定义类 averarray 继承自 myarray，在该类中定义成员函数求解数组中的所有整数的平均值。

7．定义 newarray 类，同时继承自 sortarray、rearray 和 averarray，使得 newarray 类的对象同时具有排序、逆转和求平均值的功能。在继承 myarray 的过程中声明为虚基类，体会虚基类在解决二义性问题中的作用。

【实验指导】

1．分析：注意派生类和基类中构造函数和析构函数的调用顺序。

程序运行结果如下：

```
B constructor:5
A constructor:2
A constructor:4
B constructor:3
C constructor:1
C destructor
B destructor
A destructor
A destructor
B destructor
```

2．参考程序为：

```
#include <iostream.h>
class Person{
public:
    Person(char * n ){ name = n; }
    void PrintName(){ cout<<name<<' '; }
private:
    char * name;
};
class Worker:public Person{
public:
    Worker(char * n,int nu,char * s,int ag,char *add):Person(n)
    { num = nu; sex = s; age = ag; address = add;}
    void PrintInfo()
    {
        PrintName();
```

```
        cout<<num<<' '<<sex<<' '<<age<<' '<<address<<' '<<endl;
    }
private:
    int num;
    char * sex;
    int age;
    char * address;
};
int main()
{
    Worker b("张三",10168,"男",28,"中山路 128 号");
    b.PrintInfo();
    return 0;
}
```

3．参考程序为：

```
#include <iostream.h>
class myarray{
public:
    myarray(int leng);
    ~myarray();
    void input();
    void display();
protected:
    int *alist;      //指向动态申请的一组空间
    int length;      //数组中元素的个数
};
myarray::myarray(int leng)
{
    alist=new int[leng];
    length=leng;
}
myarray::~myarray()
{
    delete [] alist;
}
void myarray::input()
{
    cout<<"enter numbers:"<<endl;
    int i;
    int *p=alist;
    for (i=0;i<length;i++)
        cin>>p[i];
}
void myarray::display()
{
    int i;
```

```
    int *p=alist;
    for (i=0;i<length;i++)
        cout<<p[i]<<" ";
    cout<<endl;
}
void main()
{
    myarray a(5);
    a.input();
    a.display();
}
```

4．参考程序为：

```
#include <iostream.h>
class myarray{
public:
    myarray(int leng);
    ~myarray();
    void input();
    void display();
protected:
    int *alist;
    int length;
};
myarray::myarray(int leng)
{
    alist=new int[leng];
    length=leng;
}
myarray::~myarray()
{
    delete [] alist;
}
void myarray::input()
{
    cout<<"enter numbers:"<<endl;
    int i;
    int *p=alist;
    for (i=0;i<length;i++)
        cin>>p[i];
}
void myarray::display()
{
    int i;
    int *p=alist;
    for (i=0;i<length;i++)
        cout<<p[i]<<" ";
```

```
    cout<<endl;
}
class sortarray:public myarray{
public:
    sortarray(int length):myarray(length){}
    void sort();
};
void sortarray::sort()
{
    int i,j,temp;
    for (i=0;i<length-1;i++)
        for (j=0;j<length-i-1;j++)
            if(alist[j]>alist[j+1])
            {
                temp=alist[j];
                alist[j]=alist[j+1];
                alist[j+1]=temp;
            }
}
void main()
{
    sortarray s(5);
    s.input();
    s.display();
    s.sort();
    s.display();
}
```

5．参考程序为：

```
#include <iostream.h>
class myarray{
public:
    myarray(int leng);
    ~myarray();
    void input();
    void display();
protected:
    int *alist;
    int length;
};
myarray::myarray(int leng)
{
    alist=new int[leng];
    length=leng;
}
myarray::~myarray()
{
```

```
    delete [] alist;
}
void myarray::input()
{
    cout<<"enter numbers:"<<endl;
    int i;
    int *p=alist;
    for (i=0;i<length;i++)
        cin>>p[i];
}
void myarray::display()
{
    int i;
    int *p=alist;
    for (i=0;i<length;i++)
        cout<<p[i]<<" ";
    cout<<endl;
}
class rearray:public myarray{
public:
    rearray(int leng):myarray(leng)
    {}
    void reverse();
};
void rearray::reverse()
{
    int i,temp;
    for(i=0;i<length/2;i++)
    {
        temp=alist[i];
        alist[i]=alist[length-1-i];
        alist[length-1-i]=temp;
    }
}
void main()
{
    rearray r(5);
    r.input();
    r.display();
    r.reverse();
    r.display();
}
```

6．参考程序为：

```
#include <iostream.h>
class myarray{
public:
```

```
    myarray(int leng);
    ~myarray();
    void input();
    void display();
protected:
    int *alist;
    int length;
};
myarray::myarray(int leng)
{
    alist=new int[leng];
    length=leng;
}
myarray::~myarray()
{
    delete [] alist;
}
void myarray::input()
{
    cout<<"enter numbers:"<<endl;
    int i;
    int *p=alist;
    for (i=0;i<length;i++)
        cin>>p[i];
}
void myarray::display()
{
    int i;
    int *p=alist;
    for (i=0;i<length;i++)
        cout<<p[i]<<" ";
    cout<<endl;
}
class averarray:public myarray{
public:
    averarray(int leng):myarray(leng)
    {}
    double aver();
};
double averarray::aver()
{
    int i,j;
    double s=0;
    for (i=0;i<length;i++)
        s=s+alist[i];
    return s/length;
```

```
}
void main()
{
    averarray a(5);
    a.input();
    a.display();
    cout<<"the aver is:"<<a.aver()<<endl;
}
```

7．参考程序为：

```
#include <iostream.h>
class myarray{
public:
    myarray(int leng);
    ~myarray();
    void input();
    void display();
protected:
    int *alist;
    int length;
};
myarray::myarray(int leng)
{
    alist=new int[leng];
    length=leng;
}
myarray::~myarray()
{
    delete [] alist;
}
void myarray::input()
{
    cout<<"enter numbers:"<<endl;
    int i;
    int *p=alist;
    for (i=0;i<length;i++)
        cin>>p[i];
}
void myarray::display()
{
    int i;
    int *p=alist;
    for (i=0;i<length;i++)
        cout<<p[i]<<" ";
    cout<<endl;
}
class averarray:virtual public myarray{
```

```
public:
    averarray(int leng):myarray(leng)
    {}
    double aver();
};
double averarray::aver()
{
    int i,j;
    double s=0;
    for (i=0;i<length;i++)
        s=s+alist[i];
    return s/length;
}
class sortarray:virtual public myarray{
public:
    sortarray(int length):myarray(length){}
    void sort();
};
void sortarray::sort()
{
    int i,j,temp;
    for (i=0;i<length-1;i++)
        for (j=0;j<length-i-1;j++)
            if(alist[j]>alist[j+1])
            {
                temp=alist[j];
                alist[j]=alist[j+1];
                alist[j+1]=temp;
            }
}
class rearray:virtual public myarray{
public:
    rearray(int leng):myarray(leng)
    {}
    void reverse();
};
void rearray::reverse()
{
    int i,temp;
    for(i=0;i<length/2;i++)
    {
        temp=alist[i];
        alist[i]=alist[length-1-i];
        alist[length-1-i]=temp;
    }
}
```

```
class newarray:public sortarray,public rearray,public averarray{
public:
    newarray(int leng):myarray(leng),sortarray(leng),rearray(leng),averarray(leng)
    {}
};
void main()
{
    newarray n(5);
    n.input();
    n.display();
    cout<<"the aver is:"<<n.aver()<<endl;
    n.sort();
    n.display();
    n.reverse();
    n.display();
}
```

实验六　多态性和虚函数

【实验目的】

1．理解多态性的概念。
2．了解编译时的多态和运行时的多态。
3．掌握虚函数的定义及实现，掌握虚析构函数的使用方法。
4．了解纯虚函数和抽象类的关系及用法。

【实验内容】

1．下面是一个计算矩形面积的小程序。仔细理解静态绑定和动态绑定的区别，并应用虚函数使程序能够输出正确结果。

```
#include<iostream.h>
class Point{
public:
    Point(double i, double j)
    {x=i; y=j;}
    double Area() const
    {   return 0.0;}
private:
    double x, y;
};
class Rectangle:public Point{
public:
    Rectangle(double i, double j, double k, double l);
    double Area() const
    {   return   w*h;}
```