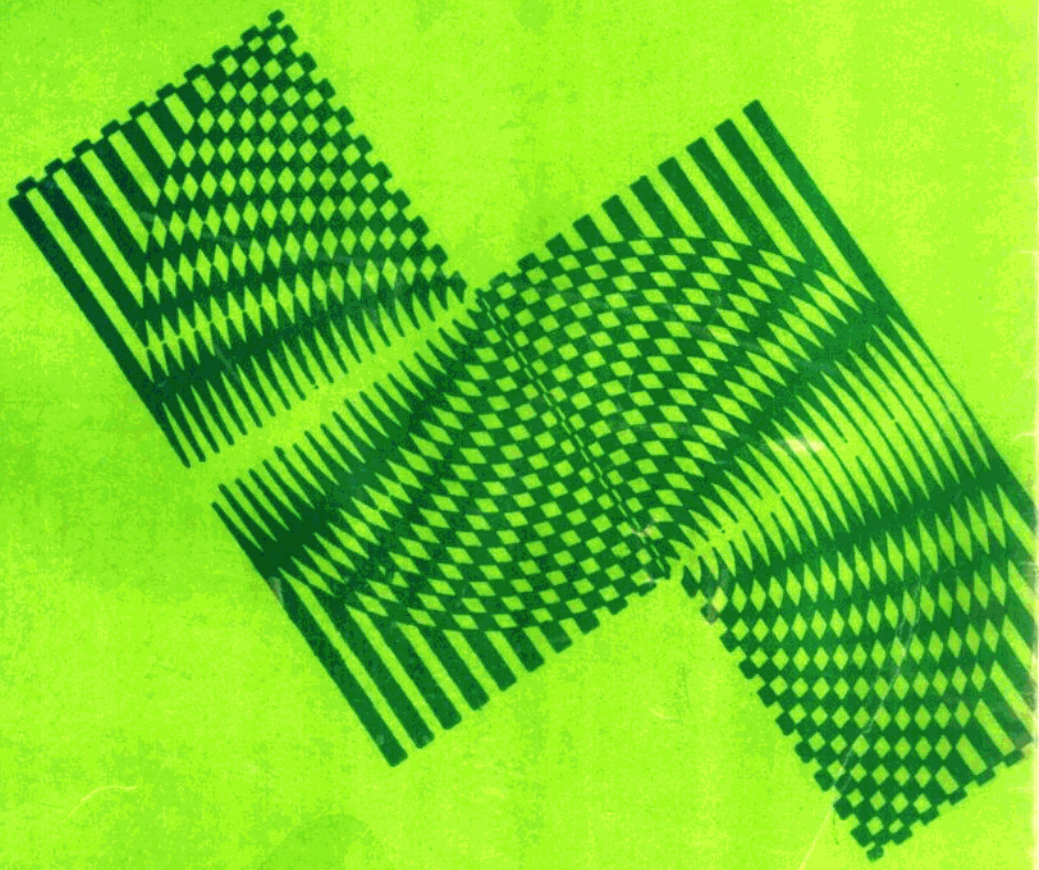
Turbo C++程序设计360例



何振邦 编著

西安电子科技大学出版社

Turbo C++程序设计360例

何振邦　编著

西安电子科技大学出版社
1993

(陕)新登字 010 号

内 容 简 介

Turbo C++是美国 Borland 公司在 Turbo C 的基础上在 IBM PC 机上实现的面向对象的程序设计的较新的编译程序。

本书是 Turbo C++的入门读物，全书共分十四章，在 Turbo C 的基础上通过360多个程序实例，由浅入深，循序渐进，较全面地讨论了输入输出流对象，C++对 C 的扩充，类封装中的数据成员和成员函数，类和对象，派生类和继承，指针，指针参数，引用类型，指向函数的指针，指向结构的指针，使用 C++库函数进行图形操作，友元，虚函数，C++和汇编语言的接口，面向对象的程序设计方法和技巧。每个程序实例均能在 Borland C++ 3.1下编译连接运行。

本书是作者《Turbo C(2.0版)程序设计及应用》的姊妹篇，可供大专院校师生和广大科技工作者用作学习 Turbo C++或 Borland C++的速成教材和参考书。

Turbo C++程序设计360例

何振邦 编著

责任编辑 徐德源

西安电子科技大学出版社出版发行

西安电子科技大学印刷厂印刷

新华书店经销

开本787×1092 1/16 印张 30 12/16字数 736 千字

1993年5月第1版 1993年5月第1次印刷 印数1—10 000

ISBN7—5606—0214—2/TP·007 定价：20 元

前　言

Turbo C++是美国 Borland 公司在 Turbo C 的基础上在 IBM PC 上实现的面向对象的程序设计语言（Object oriented programming）。Turbo C++不但与 Turbo C 高度兼容，而且比 Turbo C 的功能更强大。Turbo C++对 Turbo C 进行了多方面的扩充，能够实现 Turbo C 所能实现的任何工作。

本书是在 Turbo C 2.0的基础上编写的，是作者的《Turbo C(2.0版)程序设计及应用》的姊妹篇。本书不仅可使熟悉其它程序设计语言的读者或初学者迅速学会先进的面向对象的 Turbo C++的程序设计及应用，而且对于熟悉 C++的读者进一步开发 C++程序也是有益的。

很多人对 C++感到陌生，望而生畏，而又很想学会它，因为 Turbo C++实在太有用了。本书是为速成学习 Turbo C++而写的。初学者应当从本书中每个程序实例中的主函数 main( )开始阅读，才容易理解。本书不要求读者精通 C 语言，甚至不熟悉 C 语言的读者也可以阅读本书。读过本书，不但可以学会 C++，而且可以更加熟悉 C 语言。

本书中的每个程序实例的起点较浅，相邻的程序实例差别不大，内容通俗易懂，深入浅出，而且有必要的汉字注释。尽量反复用相同的命题讨论 C++的不同的内容和应用，这对于初学者的快速入门和深入学习是很方便的。

为了让熟悉某种程序设计语言的读者尽快掌握先进的 Turbo C++的程序设计方法，本书通过360多个程序实例较全面地讨论了 Turbo C++的主要问题，每个程序实例均能在 Borland C++3.1下编译连接运行，从任何一个程序实例开始都可以作为学习 C++的起点。本书可供大专院校师生和科技工作者用作学习 Turbo C++或 Borland C++的速成教材或参考书。

书中如有错误或不当之处，欢迎专家和读者不吝指正。

作者

1992年9月于

西安电子科技大学

目　录

第一章　流 …… 1

§1.1　流可以代表与数据传送有关系的事物 …… 1

§1.2　磁盘文件操作 …… 23

§1.3　Turbo C++ 流类库中的头文件 …… 32

§1.4　Turbo C++ 流类库中的成员函数 …… 33

§1.5　Turbo C++ 接受 .c 源程序与 Turbo C 2.0 高度兼容 …… 42

§1.6　Turbo C++ 的 .cpp 源程序接受 C 的库函数与 Turbo C 2.0 高度兼容 …… 46

§1.7　Turbo C++ 的 .cpp 源程序用标准输出流对象 cout 输出串常量 …… 50

§1.8　汉字串的使用 …… 53

§1.9　数学库的使用 …… 58

§1.10　格式化输出 …… 67

§1.11　用户自定义的格式化操纵函数 …… 95

§1.12　用户自定义的重载插入符<<的插入运算符函数 operator<< …… 96

§1.13　文本文件的读写操作 …… 105

§1.14　二进制文件的读写操作 …… 112

§1.15　seekg 可以从指定位置开始读文件 …… 126

§1.16　使用带有字符型参数的 get 可以访问下一个字符 …… 127

§1.17　Turbo C++运算符的优先级和同级运算符的求值次序 …… 130

第二章　类是结构和联合的扩充 …… 131

§2.1　用 struct 定义的结构类 …… 131

§2.2　结构类的 inline 内部函数 …… 156

第三章　控制和重复 …… 162

§3.1　for 循环 …… 162

§3.2　while 循环 …… 168

§3.3　do while 循环 …… 172

§3.4　应用练习 …… 176

§3.5　递归调用 …… 182

§3.6　switch 开关语句 …… 194

第四章　派生类 …… 200

§4.1　class 公有派生类 …… 200

§4.2　递归调用的应用 …… 217

第五章　指针 …… 248

§5.1　指针和数组 …… 248

§5.2 移指针访问二维数组 …… 263
§5.3 指针名参数 …… 267
§5.4 指针名参数用于 class 公有派生类 …… 285
§5.5 数组名参数用于 class 公有派生类 …… 295
第六章 结果变量的指针参数 …… 309
§6.1 用 C 风格引用传送普通单值结果变量的指针参数 …… 309
§6.2 递归调用过程引用传送结果变量的指针参数 …… 314
§6.3 叠代的应用 …… 319
§6.4 用 C++风格引用传送类的对象参数 …… 321
第七章 引用类型 …… 325
§7.1 C++的引用类型 …… 325
§7.2 递归调用过程引用传送结果自变量 …… 335
§7.3 叠代的应用 …… 341
第八章 指向函数的指针 …… 343
§8.1 函数名参数 …… 343
§8.2 递归调用的应用 …… 351
§8.3 引用传送指针参数或结果自变量带出结果值的过程 …… 368
§8.4 函数指针名函数 …… 384
§8.5 递归调用的应用 …… 389
§8.6 迭代的应用 …… 407
第九章 指向结构的指针 …… 412
§9.1 链表 …… 412
§9.2 结构指针参数 …… 414
第十章 图形操作 …… 416
§10.1 在屏幕上写字符 …… 416
§10.2 在屏幕上画图 …… 424
第十一章 友元 …… 430
§11.1 友元函数引用传送对象访问私有数据成员 …… 430
§11.2 递归调用的应用 …… 443
§11.3 在屏幕上画图 …… 451
第十二章 虚函数 …… 460
§12.1 虚函数允许派生类提供前辈基类成员函数的不同版本 …… 460
§12.2 用指向前辈基类对象的指针访问基类派生类的同名虚函数 …… 464
§12.3 纯虚函数 …… 469
第十三章 C++和汇编语言的接口 …… 475
§13.1 用 bcc 命令行环境的-S 开关由 .cpp 源文件生成等价的 .asm 文件 …… 475
§13.2 由 C++的 .cpp 源程序调用汇编语言的 .asm 子程序 …… 480
第十四章 编译连接运行环境 …… 481
§14.1 DOS 和内存环境 …… 481

§ 14.2 如何使用 Borland C++ 的集成开发环境 …… 481
§ 14.3 如何使用 Borland C++ 的 bcc 命令行环境 …… 483
§ 14.4 Borland 的目录结构 …… 484

第一章　流

§1.1　流可以代表与数据传送有关系的事物

一、什么是流

流是根据液体或气体可以流动，借用过来的名词。在C++中，流可以代表数据从内存传送到某个载体或设备中，叫做输出流；或者数据从某个载体或设备传送到内存缓冲区变量中，叫做输入流。它与液体或气流动之后要从原来的位置上消失不同，数据在不同的设备之间传送之后不一定消失。有时，也可以把与数据传送有关系的事物叫做流。例如，可以把文件变量，即指向FILE结构的指针叫做流。有时，还用流代表要进行传送的数据的结构、属性和特征，用一个名字代表它，叫做流的类。有时，还用流代表输入设备或者输出设备，叫做流类的对象。换句话说，流可以代表数据传送方面的事物。

二、用什么方法能使数据流动

1. 使用在标准输入输出头文件stdio.h中定义的库函数

例如，printf，fprintf，scanf，fscanf ……，可以使数据流动。

2. 数据重新定向

例如，输入定向、输出定向、管道输入输出，等等。它们的命令行和用途如下：

输入定向的命令行

C>无扩展名的可执行文件名<数据文件存盘名

用于从数据文件取数执行可执行文件的场合

输出定向的命令行

C>无扩展名的可执行文件名>结果文件存盘名

用于把运行结果存入结果文件的场合

管道输入输出的命令行

C>产生输出结果的无扩展名的可执行文件名|需要输入数据的无扩展名的可执行文件名

用于前者的输出结果恰为后者所需要的输入数据的场合

3. 使用在输入输出流类头文件iostream.h中定义的流对象cin和cout

面向对象的C++把参加数据传送操作的设备也视作对象。用cin代表输入设备，cout代表输出设备。预定义cin代表标准输入设备键盘，预定义cout代表标准输出设备显示器，预定义cerr和clog代表只输出错误信息的标准输出设备显示器。这是为了一旦遇到出错，只显示错误信息，而不把错误信息写入到任何文件中。cerr和clog的区别是，发送给cerr的错误信息立即输出，而发送给clog的错误信息只有当缓冲区满时才输出。

4. 插入操作符<<用于输出内存中的数据到设备

插入符<<的左操作数是代表输出设备的对象，右操作数是要输出的内容。

在这里，插入符<<是重新定义的向左移位符。在C++中，重新定义叫做重载。因此，插入符<<是重载的向左移位符。C++编译程序首先检查<<的左操作数和右操作数来判别应该执行向左移位还是执行插入操作，因此，不会发生二义性。

这样，把内容写入到预定义代表显示器的标准输出流对象 cout 的格式可以是

```
cout << 变量名;
```

为了记忆的方便，可以把插入符<<视作箭头。这样，可以认为插入符的作用是使内存中的数据沿着箭头<<所指引的方向流入到输出流对象所代表的输出设备中。

5. 抽取操作符>>用于从设备输入数据到内存缓冲区变量中

抽取符>>的左操作数是代表输入设备的对象，例如 cin，右操作数是内存缓冲区变量。

抽取符>>是重载的向右移位符。C++编译程序首先检查>>的左操作数和右操作数来判别应该执行向右移位还是执行抽取操作，因此，不会发生二义性。

也就是说，从预定义代表键盘的标准输入流对象 cin 给变量送数的格式可以是

```
cin >> 变量名;
```

为了记忆的方便，可以把抽取符>>视作箭头。这样，可以认为抽取符的作用是使输入流对象所代表的输入设备中的数据沿着箭头>>所指引的方向流入到内存缓冲区变量中。

三、内部类型

按格式输出函数 printf 中从%开始的格式转换符所能表示的类型，叫做内部类型，如各种整型，各种浮点型，各种字符型，表示字符串的 char *，和表示地址的 void *，等等。

按格式输出函数 printf 只能输出内部类型的数据。

C++的抽取符>>和插入符<<默认支持内部类型的数据。

以任何方式重新定义之后的抽取符>>和插入符<<仍然支持内部类型的数据。

四、C++源程序中的注释

1. 注释是很有用的非语法成分

在C++的源程序中，凡是可以书写空格的地方，都可以书写注释。编译程序不理睬注释的内容，因此，遇到注释，C++不执行任何动作。打印源程序清单时，可以打印出源程序中的注释内容。

注释可以是作者对源程序的解释和说明。虽然注释不是C++中的语法成分，它却非常有用。注释可以增加源程序的可读性。如果在源程序中含有足够多的注释，对于读者尽快和准确地理解源程序的功能，会起到事半功倍的效益。

2. 单行注释

单行注释从双斜线//开始一直到行尾。

3. 多行或不到行尾的注释

多行注释或不到行尾的注释仍使用/*注释内容*/的方式。

五、类

1. 什么是类

在生物学中用类区分和继承生物群体的构造、属性与特征。C++借用类这个名词，区

分和继承数据的结构、属性和特征。

2. 为什么要用类

用类区分和继承数据的结构、属性和特征更便于进行数据处理和科学计算。如果不用类，对于每个数据和对象，都得定义它们的所有的性质和特征。如果使用了类，一些数据和对象可以继承和共享另外一些数据和对象的相同的性质和特征，只需要定义有区别的性质和和特征。数据和对象的性质和特征还可以有多层次的、多重的继承和派生。

3. 类是结构和联合的扩充

C 中已有的各种整型，各种浮点型、字符型……虽然也能区分数据的属性和特征，但是这还不够。C 中的结构和联合虽然能够更好地区分数据的属性和特征，这也不够，还要进行扩充。C++中的类是结构和联合的扩充。

4. 在 C++中有什么类

在 C++中有 struct 类、union 类和 class 类。

六、对象

对象是由类描述其属性和特征的数据。对象可以拥有操作和操纵类中的数据的函数和代码。

七、struct 类

1. 什么是 struct 类

struct 类由结构扩充而来。其中，不但含有对象的数据成员，而且含有可对数据成员进行操作的成员函数。

2. struct 类的定义

struct 类的定义从关键字 struct 开始，定义语句可以是：

```
struct 结构类名
{ //结构类定义体开始
   数据成员的访问控制修饰词：
      取值类型名  存放对象的第 1 个数据成员取值的变量名1；
      取值类型名  存放对象的第 2 个数据成员取值的变量名2；
      ……
   成员函数的访问控制修饰词：
      用于初始化数据成员的构造函数说明语句或定义语句//构造函数名与类名同名
      用于对数据成员进行操纵或操作的第 1 个成员函数1 的说明或定义语句1
      用于对数据成员进行操纵或操作的第 2 个成员函数2 的说明或定义语句2
      ……
}; //结构类定义体结束
```

其中，花括号{ }和它所包围的内容叫做结构类定义体。通常，用分号结束结构类定义体。

在关键字 struct 和由{ }所包围的结构类定义体之间是用户用标识符自定义的结构类名。结构类名在 C 中有时叫做成员表名、结构名、结构类型名或结构标志。结构类名代表由{ }包围的结构类定义体中的全部内容。

struct 类的数据成员的访问控制修饰词可以是 private、protected、public、或缺省。

若取 private，是私有的数据成员，只能由同一个类的对象或成员访问。

若取 protected，是受保护的数据成员，可由同一个类或它的派生类的对象或成员访问。

若取 public，是公有的数据成员，在类外与此 struct 类定义在同一作用域中的任何地方可由对象和句号圆点符访问。结构类的数据成员或成员函数的缺省的访问控制是公有的。

结构类的成员函数的访问控制通常设置成为公有的，在类外可由对象和句号圆点符访问。

八、数据成员

1. 什么是数据成员

对象的数据有地方存放时才能够使用。内存中的存储单元可以存放数据。存储单元与变量是一一对应的。在类定义体中定义的变量具有存储单元，能够存放对象的数据，叫做对象的数据成员，简称数据成员。

2. 如何定义数据成员

在类内定义数据成员的格式是

```
取值类型名  变量名;
```

3. 如何访问数据成员

在类内的成员函数可以访问同一个类定义体中的数据成员。

在类外可由对象和句号圆点符访问公有的数据成员：

```
类的对象名.数据成员名
```

九、成员函数

1. 什么是成员函数

在类定义体中定义的函数，或者在类的作用域中定义的函数叫做成员函数。

2. 成员函数有什么用

成员函数可以对同一个类定义体中的数据成员进行操纵和操作。

3. 如何在类定义体内定义成员函数

在类定义体内定义成员函数的格式是

```
结果值类型  函数名(类型名1  形式参数名1,……,类型名n  形式参数名n)
{ //成员函数定义体开始
  //成员函数定义体中的语句序列实现它的功能
} //成员函数定义体结束
```

4. 如何在类外定义成员函数

在类定义体外定义成员函数的格式是

```
结果值类型  类名::函数名(类型名1  形式参数名1,……,类型名n  形式参数名n)
{ //成员函数定义体开始
  //成员函数定义体中的语句序列实现成员函数的功能
} //成员函数定义体结束
```

其中，类作用域分辨符::的左操作数是类名，右操作数是成员函数名。

5. 如何访问成员函数

用对象和句号圆点符可以访问公有的成员函数。访问公有的成员函数的调用语句可以是

```
类的对象名.成员函数名(实在参数表达式表);
```

十、构造函数 constructor

1. 什么是构造函数

函数名与类名同名的成员函数叫做构造函数。

2. 构造函数有什么用

构造函数可以自动读取对象的各个初值，并依次用它们初始化对象的各个数据成员。

3. 如何在类定义体内定义构造函数

在类定义体内定义构造函数的格式是

```
构造函数名(类型名1  形式参数名1,……,类型名n  形式参数名n)
  //函数名与类名同名的是构造函数
  //构造函数不返回任何结果值,冠以函数名不能写任何结果值类型
{ //构造函数定义体开始
  //构造函数定义体中的语句序列实现初始化数据成员的功能
  //C++的编译程序负责把对象的第1个初值自动传送给形式参数名1
  用户负责书写把形式参数名1的取值传送给数据成员1的语句
  //C++的编译程序负责把对象的第2个初值自动传送给形式参数名2
  用户负责书写把形式参数名2的取值传送给数据成员2的语句
  //如此,对象的第i个初值经形式参数名i传给数据成员i,依此类推,初始化各个数据成员。
} //构造函数定义体结束
```

4. 如何在类外定义构造函数

在类定义体外定义构造函数的格式是

```
类名::构造函数名(类型名1  形式参数名1,……,类型名n  形式参数名n)
  //函数名与类名同名的是构造函数
{ //构造函数定义体开始
  //由形式参数给对应的数据成员传值的语句序列实现初始化数据成员的功能
} //构造函数定义体结束
```

其中，类作用域分辨符::的左操作数是类名，右操作数是构造函数名。

十一、析构函数 destructor

1. 什么是析构函数

函数名是类名冠以波浪符～的成员函数叫做析构函数。析构函数没有任何结果值类型，不返回任何结果值，也没有任何自变量参数。

2. 析构函数有什么用

析构函数在撤销对象时被调用。析构函数在删除对象之前，先删除对象的成员。对象被撤销时，要释放原先分配给它的内存，这是由析构函数来完成的。析构函数是构造函数的配对物，用于清理不再需要的对象，解除在定义对象时由构造函数给它动态分配的内存空间。析构函数被调用的次序恰好和构造函数相反。如果你没有定义一个析构函数，C++自动使

用一个隐含的由内部生成的析构函数实现释放内存。为了释放内存，析构函数也还可以执行另外的操作，例如把数据成员的内容写入到磁盘文件中，然后关闭文件，等等。

3. 如何在类定义体内定义析构函数

在类定义体内定义析构函数的格式是

```
析构函数名( )
  //函数名与类名同名但要冠以波浪符~的是析构函数
  //析构函数不返回任何结果值，不能写任何结果值类型，也没有任何自变量参数
{ //析构函数定义体开始
  //析构函数定义体中的语句序列实现释放对象的内存空间的功能
} //析构函数定义体结束
```

4. 如何在类外定义析构函数

在类定义体外定义析构函数的格式是

```
类名::析构函数名( )
  //类名冠以波浪符~作为函数名的函数是析构函数
{ //析构函数定义体开始
  //析构函数定义体中的语句序列实现释放对象的内存空间的功能
} //析构函数定义体结束
```

其中，类作用域分辨符::的左操作数是类名，右操作数是析构函数名。

十二、结构类的对象的定义

冠以结构类名可以定义结构类的对象。定义结构类的对象的定义语句可以是：

```
结构类名  对象名标识符(成员初值表);
```

或者，也可以冠以结构类定义体定义结构类的对象：

```
struct 结构类名 { /*结构类定义体*/ } 对象名标识符(成员初值表);
```

十三、union 类

1. 什么是 union 类

union 类由联合扩充而来。其中，不但含有对象的数据成员，而且含有可对数据成员进行操作的成员函数。

2. union 类的定义

union 类的定义从关键字 union 开始，定义语句可以是：

```
union 联合类名
{ //联合类定义体开始
  //联合类的数据成员只有缺省的公有的访问控制，用户不可改变
     取值类型名  存放对象的第1 个数据成员取值的变量名1;
     取值类型名  存放对象的第2 个数据成员取值的变量名2;
     ……
  成员函数的访问控制修饰词:
     用于初始化数据成员的构造函数说明语句或定义语句//构造函数名与类名同名。
     用于对数据成员进行操纵或操作的第1 个成员函数1 的说明或定义语句1
     用于对数据成员进行操纵或操作的第2 个成员函数2 的说明或定义语句2
```

```
    ……
};//联合类定义体结束
```

花括号{ }和它所包围的内容叫做联合类定义体。通常，用分号结束联合类定义体。

在关键字 union 和由{ }所包围的联合类定义体之间是用户用标识符自定义的联合类名。联合类名在 C 中有时叫做成员表名，联合名，联合类型名或联合标志。联合类名代表由{ }包围的联合类定义体中的全部内容。

联合类的数据成员的缺省的访问控制是公有的，用户不可改变。

联合类的成员函数的访问控制通常设置成为公有的。缺省的访问控制是公有的。

公有的数据成员和成员函数在类外可由对象和句号圆点符访问。

十四、联合类的对象的定义

冠以联合类名可以定义联合类的对象。定义联合类的对象的定义语句可以是：

```
联合类名  对象名标识符(成员初值表);
```

或者，也可以冠以联合类定义体定义联合类的对象：

```
union 联合类名 { /*联合类定义体*/ } 对象名标识符(成员初值表);
```

十五、class 类

1. 什么是 class 类

class 类由结构扩充而来。其中，不但含有对象的数据成员，而且含有可对数据成员进行操作的成员函数。

2. class 类的定义

class 类的定义从关键字 class 开始，定义语句可以是：

```
class 用户自定义的 class 类名
{ //class 类定义体开始
   数据成员的访问控制修饰词:
      取值类型名  存放对象的第 1 个数据成员取值的变量名1;
      取值类型名  存放对象的第 2 个数据成员取值的变量名2;
      ……
   成员函数的访问控制修饰词:
      用于初始化数据成员的构造函数说明语句或定义语句
      用于对数据成员进行操纵或操作的第 1 个成员函数 1 的说明或定义语句 1
      用于对数据成员进行操纵或操作的第 2 个成员函数 2 的说明或定义语句 2
      ……
};//class 类定义体结束
```

其中，花括号{ }和它所包围的内容叫做 class 类定义体。通常，用分号结束 class 类定义体。

在关键字 class 和由{ }所包围的 class 类定义体之间是用户用标识符自定义的 class 类名。class 类名代表由{ }包围的 class 类定义体中的全部内容。

访问控制修饰词可以是 private，protected，public，或缺省。

private　是私有的，只能由同一个类的对象或成员访问。

protected是保护的，可由同一个类或它的派生类的对象或成员访问。

public　是公有的，可由类外与此 class 类定义在同一作用域中的 任何地方访问。

class 类的缺省的访问控制是私有的。

class 类的成员函数的访问控制通常可以设置成为公有的。

十六、class 类的对象的定义

冠以 class 类名可以定义 class 类的对象。定义 class 类的对象的定义语句可以是：

用户自定义的 class 类名　对象名标识符(成员初值表)；

或者，也可以冠以 class 类定义体定义对象：

class　用户自定义 class 类名 { /＊class 类定义体＊/ } 对象名标识符(成员初值表)；

十七、C++的派生类

1. 什么是C++的派生类

C++的派生类是在继承和共享类中的数据成员和成员函数的基础上，又附加了新的数据成员和成员函数的新类。

2. 什么是C++的基类

被继承的C++的类可以叫做基类，父类或者前辈基类。

派生类也可以叫做子类。

3. 如何定义C++的派生类

定义C++的派生类的定义语句可以是

class　派生类名：派生访问修饰词　被继承的前辈基类名 { /＊派生类定义体成员表＊/}；

或

struct　派生类名：派生访问修饰词　被继承的前辈基类名 { /＊派生类定义体成员表＊/}；

派生类可以作为新的基类继续生成新的派生类。

其中，在派生类定义体中，只需定义在历届的前辈基类中不能共享的新的附加成员。

4. 派生类能从前辈基类中继承和访问什么

派生类能够从前辈基类中继承所有的成员，但派生类的对象只能够访问前辈基类中受的保护的和公有的成员。派生类的对象所能访问的前辈基类中的成员示于表1.1。

表 1.1　派生类的对象所能访问的前辈基类中的成员

前辈基类中的访问修饰词	派生访问修饰词	派生类继承的前辈基类中的成员的访问控制级别
public	public	public
private	public	不可访问
protected	public	protected
public	private	private
private	private	不可访问
protected	private	private

class 派生类的缺省的派生访问修饰是 private。struct 结构派生类的缺省派生访问修饰是 public。union 类既不能够作为前辈基类，也不能够作为派生类。

派生访问修饰词用于修饰被继承的前辈基类名，可以控制前辈基类中的成员在派生类

中的可访问性，示于表 1.1。

前辈基类中的访问修饰词 private，protected 和 public 的访问级别不同。存取权可以在父亲、儿子、孙子之间传递或抑制。通常，不需要把你的数据暴露给非家族或非成员者。C++可以做到这一点。由前辈基类所看到的其中的成员的访问级别不必和派生类所继承到的它的访问级别一样。也就是说，在定义派生类的定义语句中，使用派生访问修饰词修饰基类名，使你能够控制派生类从前辈基类继承过来的成员的访问级别。

一个派生类可以从它的前辈基类私有的或公有的派生。私有派生时，可使前辈基类中的公有的和受保护的成员转变成为私有的访问控制级别。公有派生时，不会改变前辈基类中原有的访问控制级别。

派生类能够继承前辈基类中的所有的成员，但只能使用它们原来在基类中是受保护和公有的成员。前辈基类中私有成员不能由派生类的成员直接使用。

5. 什么是多重继承的派生类

从一个以上的前辈基类生成的派生类可以继承每个前辈基类中的所有成员，但只能使用其中的公有的和受保护的数据成员和成员函数，叫做多重继承的派生类。

6. 如何定义多重继承的派生类

定义多重继承的派生类的定义语句可以是

```
class 派生类名:public 被继承的前辈基类名1,private 被继承的前辈基类名2,……,被继承的前辈基类名 n
{ /* 派生类定义体成员表 */ };
或
class 派生类名:private 被继承的前辈基类名1,private 被继承的前辈基类名2,……,被继承的前辈基类名 n
{ /* 派生类定义体成员表 */ };
或
struct 派生类名:public 被继承的前辈基类名1,private 被继承的前辈基类名2,……,被继承的前辈基类名 n
{ /* 派生类定义体成员表 */ };
或
struct 派生类名:private 被继承的前辈基类名1,private 被继承的前辈基类名2,……,
                被继承的前辈基类名 n
{ /* 派生类定义体成员表 */ };
```

7. 修饰前辈基类的派生访问修饰词可以改变多重继承的派生类所能访问的基成员 chass 派生类的缺省派生访问修饰词是 private，struct 派生类的缺省派生访问修饰词是 public。protected 不能 用作派生访问修饰词。

union 既不能作为多重继承的前辈基类，也不能够作为派生类。

在定义派生类的定义语句中，写在前辈基类名字之前的派生访问修饰词，可以改变派生类从前辈基类继承到的基类成员的存取属性和访问控制级别，但不能改变基类 对基成员的访问控制级别。

由 public 基类生成的派生类可继承基类中的 public 成员成为派生类的 public 成员。由 puublic 基类生成的派生类可继承基类中的 protected 成员成为派生类 的 protected 成员。派生类不可访问基类的 private 成员。

由 private 基类生成的派生类可以继承基类的 public 和 protected 成员 成为 派生类的 private 成员。派生类不可访问基类中的 private 成员。

8. 多重继承的派生类定义示例

```
class a:b                    //class 派生类 a 默认为是由 private 基类 b 派生
{ /*派生类定义体成员表 */ };
class c:d,public e
//class 派生类 c 默认为是由 private 基类 d 和 public 基类 e 派生
{ */派生类定义体成员表 */ }
struct f:g                   //struct 派生类 f 默认为是由 public 基类 g 派生
{ */派生类定义体成员表 */ }
struct h:i,private j
//struct 派生类 h 默认为是由 public 基类 i 和 private 基类 j 派生
{ */派生类定义体成员表 */ }
```

9. 在多重继承的派生类定义体中应当含有什么成员

在多重继承的派生类定义体中，只需定义在历届的前辈基类中继承不到的新的附加的成员。

10. 派生类中的公有的和受保护的访问控制说明可以改变继承所得的私有的访问级别

举例程序段如下：

```
    ……
class b                      //用户自定义的 class 类名为 b
{                            //class 类 b 的定义体开始
    int a;                   //class 类的缺省访问控制是私有的
  public:                    //显式说明访问控制是公有的，否定缺省的访问控制
    int c,e;                 //公有的数据成员 c 和 e 的取值类型说明
    int bf(void);            //公有的成员函数 bf 说明
};                           //class 类 b 的定义体结束

class d:private b            //class 派生类 d 是由 private 基类 b 派生的
{                            //派生类 d 的定义体开始
                  //派生类 d 由 private 基类 b 中继承得到的成员 a,c,e 和 bf 的访问控制是私有的
                             //前辈基类 b 中的私有成员 a 在派生类 d 中是不可访问的
    int f;                   //class 派生类的缺省访问控制是 private
  public:                    //显式说明访问控制是公有的，否定缺省的访问控制
    b::e;                    //由 private 基类 b 继承得到 e 的访问控制是 private，现改为 public
    int g;                   //公有的数据成员 g 的取值类型说明
    int df(void);            //公有的成员函数 df 说明
};                           //派生类 d 的定义体结束
……
int ef(d & p);
    //在基类 b 和派生类 d 以外，含有 class 派生类 d 的引用形式对象参数 p 的函数 ef 的说明
    ……
```

类外的函数 ef 只能使用访问控制为公有的成员 e，g 和 df。

成员函数 df 在 class 派生类 d 中。派生类 d 是由 private 基类 b 派生的。因此，df 可以访问：