HZ BOOKS
华章教育

高等院校计算机教材系列

C++
面向对象编程基础

刁成嘉 刁奕 等编著

为教师提供电子教案

机械工业出版社
China Machine Press

高等院校计算机教材系列

C++
面向对象编程基础

刁成嘉 刁 奕 等编著

机械工业出版社
China Machine Press

本书系统地讲述了C++程序设计语言的基本语法格式和功能，还详细介绍了C++中的类的封装、继承和多态处理机制，以及微软基础类库（MFC）等高级C++编程技术，通过大量程序实例介绍如何利用C++语言的高级对象特性开发高效率、高质量的面向对象程序，并附有大量习题供初学者练习使用。

本书可以作为高等院校计算机和信息技术专业相关课程教材，也可作为广大软件开发人员学习面向对象C++编程技术的自学指导书和技术参考书。

图书在版编目（CIP）数据

C++面向对象编程基础/刁成嘉等编著. —北京：机械工业出版社，2007. 11
（高等院校计算机教材系列）
ISBN 978-7-111-22474-7

Ⅰ. C…　Ⅱ. 刁…　Ⅲ. C语言—程序设计—高等学校—教材　Ⅳ. TP312

中国版本图书馆CIP数据核字（2007）第152610号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：王春华
三河市明辉印装有限公司印刷 · 新华书店北京发行所发行
2008年1月第1版第1次印刷
184mm × 260mm · 20印张
定价：30.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线（010）68326294

前　言

C++是一种使用广泛的程序设计语言，只有全面深入地了解和掌握C++面向对象基础知识和基本编程技巧，才能高效率地开发出一个面向对象的软件系统。

本教材共分8章，各章内容如下：

第1～4章详细讲述了面向对象C++程序设计语言的基础语法格式和基本功能，通过大量程序实例介绍如何用C++语言编写一个面向对象的程序及一些编程技巧，重点介绍了C++语言中关于类的封装的基本原理和实现。

第5章介绍了C++语言中关于类的继承机制的描述和实现。

第6章重点介绍了C++语言中关于类的多态性的基本原理、机制和实现。

第7章详细介绍了C++的流处理机制。

第8章简要介绍了微软基础类库（MFC）的体系结构框架和使用MFC创建图形界面系统的方法。MFC实际上是一套开发模板，针对不同的应用和目的，程序员可以选择采用不同的模板。对程序的控制主要是由MFC框架完成，MFC提供了开发图形界面的大部分功能，并预定义或实现了许多事件和消息处理等。MFC是C++类库，程序员可通过使用、继承和扩展适当的类来实现特定的目的，从而大大减轻了开发人员的编程工作量。

本教材的内容安排按标准C++规范进行，第1～7章的程序示例在Visual C++ 6.0集成开发环境下通过调试运行。基于MFC图形界面系统的程序示例在Visual Studio.NET 2003集成开发环境下通过调试运行。

本书可作为高等院校计算机和信息技术专业相关课程的教材，也可作为广大软件开发人员学习面向对象C++编程技术的自学指导书和参考书。

刁奕编写了第1章和第8章，刁成嘉负责其他章节的编写。肖鹏、程玉鹏、金士英、郑伟、费志泉、郜业军、蓝炳伟、刘胜斐、郑莹莹、罗仕波、田新、漆方敏、陈惠、杨志真、宋雪松、赵青、高建国、旷昊等参加了部分章节示例和习题的编写。由于编者水平所限，加之时间仓促，疏漏、欠妥、谬误之处在所难免，敬请读者批评指正。

作者

2007年5月于南开园

目 录

前言

第1章 C++的基本程序设计思想 ……1
1.1 C++语言简介 ……1
1.1.1 一个C++程序实例 ……2
1.1.2 C++字符集 ……3
1.1.3 ASCII码 ……3
1.2 C++程序的基本组成成分 ……4
1.2.1 词法记号 ……4
1.2.2 常量 ……6
1.2.3 变量 ……10
1.2.4 有名常量 ……11
1.2.5 运算符与表达式 ……12
1.3 C++数据类型 ……20
1.3.1 基本数据类型 ……20
1.3.2 枚举类型 ……23
1.3.3 结构体 ……24
1.3.4 联合体 ……27
1.3.5 位域 ……28
1.4 数据的输入与输出 ……29
1.4.1 I/O流 ……29
1.4.2 预定义的插入符和提取符 ……29
1.4.3 简单的I/O格式控制 ……30
1.5 条件控制 ……30
1.5.1 条件语句 ……31
1.5.2 switch语句 ……34
1.6 循环 ……36
1.6.1 for循环语句 ……36
1.6.2 while循环语句 ……38
1.6.3 do-while循环语句 ……39
1.6.4 转移语句 ……39
1.6.5 多重循环 ……40
1.7 数组 ……44
1.7.1 数组的概念 ……44
1.7.2 一维数组 ……45
1.7.3 二维数组 ……49
1.7.4 使用typedef语句定义数组类型 ……52
1.8 字符串 ……53
1.8.1 字符串概念 ……53
1.8.2 字符串函数 ……55
1.9 本章小结 ……57
习题 ……58
第2章 函数 ……64
2.1 函数的定义与使用 ……64
2.1.1 函数的定义与说明 ……64
2.1.2 函数的调用 ……65
2.1.3 函数的参数和返回值 ……67
2.2 传值调用与引用调用 ……72
2.2.1 传值调用 ……72
2.2.2 引用调用 ……72
2.3 函数和变量的作用域 ……77
2.3.1 函数的作用域 ……77
2.3.2 变量的作用域和生存期 ……77
2.3.3 C++的命名空间 ……78
2.4 内联函数和重载函数 ……80
2.4.1 内联函数 ……80
2.4.2 函数重载 ……81
2.5 函数的嵌套调用和递归调用 ……82
2.5.1 函数的嵌套调用 ……82
2.5.2 函数的递归调用 ……83
2.6 函数模板和使用C++系统函数 ……86
2.6.1 函数模板 ……86
2.6.2 使用C++系统函数 ……88
2.7 本章小结 ……88
习题 ……89
第3章 类与对象 ……95
3.1 类的定义和对象的创建 ……95

3.1.1 类设计的基本概念 ……95
3.1.2 类的定义格式 ……96
3.1.3 类的成员函数 ……97
3.1.4 类成员的访问控制 ……97
3.1.5 对象的声明与使用 ……98
3.2 构造函数和析构函数 ……99
3.2.1 构造函数与拷贝构造函数 ……99
3.2.2 析构函数 ……103
3.3 对象的生存期及类的作用域 ……104
3.3.1 对象的生存期 ……104
3.3.2 类的作用域 ……105
3.4 友元类和友元函数 ……106
3.4.1 友元函数 ……107
3.4.2 友元类 ……107
3.5 静态成员 ……109
3.5.1 静态数据成员 ……109
3.5.2 静态成员函数 ……110
3.6 类和对象的进一步应用 ……110
3.6.1 类对象作为成员 ……110
3.6.2 常对象 ……111
3.6.3 对象作为函数参数 ……112
3.6.4 对象数组 ……113
3.7 类模板 ……115
3.7.1 类模板的定义 ……115
3.7.2 类模板的使用 ……116
3.8 本章小结 ……117
习题 ……118
第4章 指针 ……122
4.1 指针的概念 ……122
4.2 指针的定义、赋值及运算 ……123
4.2.1 如何定义指针 ……123
4.2.2 “*”和“&”运算符 ……123
4.2.3 指针的赋值 ……123
4.2.4 指针的运算 ……125
4.3 指针与数组 ……127
4.3.1 用指针访问数组元素 ……127
4.3.2 指向数组的指针 ……129
4.3.3 指针数组 ……130
4.4 指针与字符串 ……131
4.4.1 字符数组与字符指针 ……131
4.4.2 字符串处理函数 ……133
4.4.3 string类 ……136
4.5 指针与函数 ……136
4.5.1 指针作为函数参数 ……136
4.5.2 指针函数 ……138
4.5.3 函数指针 ……139
4.6 指针与引用 ……140
4.7 其他类型的指针 ……141
4.7.1 const指针 ……141
4.7.2 对象指针 ……143
4.7.3 this指针 ……143
4.8 动态内存分配 ……145
4.8.1 堆内存 ……145
4.8.2 new 和delete 运算符 ……145
4.8.3 动态内存分配与释放函数 ……147
4.9 本章小结 ……148
习题 ……149
第5章 继承与派生 ……157
5.1 继承的含义 ……157
5.2 继承的语法 ……157
5.3 继承中的访问控制 ……158
5.3.1 公有继承 ……158
5.3.2 私有继承 ……160
5.3.3 保护继承 ……161
5.4 成员覆盖与作用域分辨 ……161
5.5 继承中对象的初始化与清除 ……163
5.5.1 初始化和清除 ……163
5.5.2 构造函数和析构函数的调用顺序 …164
5.6 向上映射 ……165
5.6.1 公有继承中的向上映射示例 ……166
5.6.2 私有继承、保护继承和向上映射 …168
5.6.3 向上映射的缺点 ……168
5.7 多重继承 ……169
5.7.1 多重继承的语法 ……169
5.7.2 多重继承中的构造函数与
析构函数 ……169
5.7.3 多重继承的二义性 ……171
5.7.4 虚基类 ……173

5.8 本章小结 ……176
习题 ……177
第6章 多态 ……187
6.1 多态概述 ……187
6.1.1 多态的类型 ……187
6.1.2 多态的实现机制 ……187
6.2 运算符重载 ……188
6.2.1 运算符重载的语法和规则 ……188
6.2.2 一元运算符重载 ……189
6.2.3 二元运算符重载 ……191
6.2.4 几个特殊运算符的重载 ……194
6.2.5 运算符重载与类型转换 ……198
6.3 虚函数 ……199
6.3.1 问题的提出 ……200
6.3.2 虚函数的运用 ……201
6.3.3 虚函数的实现 ……204
6.3.4 虚函数和构造函数 ……205
6.3.5 虚析构函数 ……206
6.3.6 对象切片问题 ……209
6.3.7 纯虚函数 ……210
6.3.8 抽象类 ……211
6.4 本章小结 ……214
习题 ……215
第7章 输入/输出流类 ……223
7.1 流类及流类间的关系 ……223
7.2 从标准输入/输出流中读/写数据 ……225
7.2.1 从标准输入获取数据 ……226
7.2.2 将数据写入标准输出 ……229
7.3 顺序文件的输入/输出 ……231
7.3.1 文件的打开与关闭 ……231
7.3.2 将数据写入文件 ……233
7.3.3 从文件中读取数据 ……234
7.4 文件的随机读取 ……238
7.4.1 文件的读取指针 ……238
7.4.2 随机读取数据文件 ……239
7.4.3 从串流中读取数据 ……240
7.4.4 自定义输入运算符 ……241
7.5 文件的随机写入 ……242
7.5.1 文件的写指针 ……242
7.5.2 随机写 ……243
7.5.3 将数据写入串流 ……245
7.5.4 输出运算符重载 ……246
7.6 二进制文件的读取 ……247
7.7 本章小结 ……249
习题 ……249
第8章 Windows C++编程基础 ……254
8.1 Windows编程基础知识 ……254
8.1.1 窗口 ……255
8.1.2 句柄 ……255
8.1.3 消息 ……255
8.1.4 事件驱动 ……256
8.1.5 MFC类库简介 ……256
8.2 MFC应用程序基本架构 ……258
8.2.1 用MFC“应用程序向导”自动生成框架程序 ……258
8.2.2 MFC程序的类结构 ……259
8.2.3 MFC程序的文件组成 ……260
8.2.4 应用程序类及其主要成员函数InitInstance() ……261
8.2.5 文档类、视图类及文档/视图设计模式 ……262
8.2.6 框架窗口类 ……263
8.2.7 子窗口类 ……264
8.2.8 MFC的消息处理机制 ……264
8.3 菜单、快捷键、工具栏和状态栏 ……265
8.3.1 菜单 ……265
8.3.2 几个建立菜单的程序实例 ……266
8.3.3 快捷键 ……270
8.3.4 工具栏 ……272
8.3.5 状态栏 ……276
8.4 图形界面编辑 ……278
8.4.1 图形设备接口 ……278
8.4.2 伪设备 ……279
8.4.3 设备语义 ……279
8.4.4 CDC类 ……280
8.4.5 触发WM_PAINT绘图消息 ……281
8.4.6 采用CDC类绘图的实例 ……281
8.4.7 字体类和文本输出实例 ……282

8.5 文件操作 ……283
8.5.1 文件与CFile类……283
8.5.2 文件操作方法 ……283
8.5.3 序列化 ……284
8.6 对话框 ……291
8.6.1 特殊的窗口——对话框 ……291
8.6.2 对话框的运行机制 ……291
8.6.3 控件 ……292
8.6.4 创建基于对话框的MFC应用程序 …293
8.6.5 对话框数据交换和数据验证（DDX/DDV） ……299
8.6.6 基本消息对话框 ……303
8.6.7 通用对话框 ……303
8.7 本章小结 ……307
习题 ……307

第1章 C++的基本程序设计思想

本章将对C++语言的基本知识进行说明分析，以帮助初学者尽快学会使用C++语言进行程序设计。本章介绍了C++的诞生、特点、程序实例及其字符集；C++数据类型；数据的输入与输出；各种控制语句；数组以及字符串的定义及使用。

本章目的

- 了解C++的诞生、特点、程序实例及其字符集
- 了解C++数据类型
- 了解数据的输入与输出
- 熟练使用各种控制语句
- 掌握数组的定义、一维数组和二维数组的使用及用typedef语句定义数组类型
- 掌握字符串的定义及使用

1.1 C++语言简介

C++是从C语言发展而来的。在C语言诞生之前，系统软件主要是用汇编语言编写的。由于汇编语言程序依赖于计算机硬件，其可读性和可移植性都很差。但一般的高级语言又难以实现对计算机硬件的直接操作（这正是汇编语言的优势），于是人们盼望有一种兼有汇编语言和面向过程的高级语言特性的新语言，C语言由此应运而生。

C语言是贝尔实验室于20世纪70年代初研制出来的，后来又被多次改进，并出现了多种版本。20世纪80年代初，美国国家标准化协会（ANSI）根据C语言问世以来各种版本对C语言的发展和扩充，制定了ANSI C标准（1989年做了修订）。

C语言具有许多优点，如：

- 语言简洁、紧凑，使用方便、灵活。
- 运算符极其丰富。
- 生成的目标代码质量高，程序执行效率高。
- 可移植性好（较之汇编语言）。
- 可以直接操纵硬件。

随着上世纪80年代面向对象方法成为程序设计的主流方法以来，任何一种高级程序设计语言如果不具备面向对象的功能就没有市场。顺应潮流，C++便在C语言基础上增加了支持面向对象的功能，它是在1980年由AT&T贝尔实验室的Bjarne Stroustrup博士创建的。

与纯粹的面向对象语言（如Java、Smalltalk等）不同，C++语言是一种混合编程语言，既具有面向过程的功能又具有面向对象的功能。

C++语言从C语言中继承了其独有的那种为程序员所喜爱的简明高效的表达式形式，比较容易地解决了目标代码高质量高效率的问题。C++保持了C的简洁、高效和接近汇编语言等特点，对C的类型系统进行了改革和扩充，因此C++比C更安全。由于它的兼容性，使得许多C代码不经修改就可以直接为C++所用。

C++语言最有意义的方面是支持面向对象的特征。虽然与C的兼容使得C++具有面向过程和面向对象的双重特点，但它在概念上是和C完全不同的语言，我们应该注意按照面向对象的思维方式去编写C++程序。

为了有利于C++语言的发展，C++语言的设计者和开发商还注意解决了与C语言相关的一些问题。一方面，在相当长的时间内，C与C++语言及其编译系统同时发售，促进了C程序员向C++语言的转化过程。另一方面，对于C语言的语法成分，做了许多卓有成效的取代工作，例如：

- 引入const有名常量和内联函数概念，取代宏定义。
- 引入reference（引用）概念，部分取代C语言中过于灵活、影响安全性的指针。
- 引入动态内存分配运算符，取代比较低级的相关库函数。
- 引入用于数据I/O的流类，取代可读性差的C语言I/O库函数等。

1.1.1 一个C++程序实例

编写程序的目的就是要计算机做一件事，下面的程序例1-1以计算机做一件小事（在屏幕上显示一串字符“hello world!”）为例，使读者初识C++程序。

例1-1：在屏幕上显示一串字符“hello world!”。

```
1 //*************************
2 //*      例1-1.cpp          *
3 //*************************
4   #include <iostream.h>
5   void main(void) {
6         cout << "hello world!" ;
7 }
```

程序在VC++ 6.0环境下的运行结果：

```
hello world!
```

此程序由7行代码组成，不包含每行左端的行号。每行代码说明如下。

1）第1～3行为注释行。程序的每行如出现符号“//”，则在其右的所有字符为注释。注释是帮助读者阅读程序的说明，与程序的运行没有关系，在程序编译时，注释被当作空格处理。这些注释行指出本程序以文件名“程序例1-1.cpp”进行存储。

2）第4行的#include是一条编译预处理指令，它告诉编译系统在编译本程序时把系统提供的“标准的输入/输出流类”头文件<iostream.h>的内容插入到第4行的位置，它在程序中的作用与第6行的输出语句有关。注意，由系统提供的头文件名要用一对尖括号<>围起。

3）第5~7行是程序的主体，由一个主函数组成。

第5行是主函数头，其中main是主函数名，第一个void指出该函数无返回值。括号( )表示函数，括号内应为函数的参数表，但此函数无参数，故用void表示，它与空白( )效果相同。main( )函数是C++程序执行的起点，也称入口语句。

第6~7行称为函数体，用{ }括起来，函数体内可以包含任意多行语句。

第6行是本程序中主函数唯一要执行的语句：在屏幕输出（显示）一串字符。cout是一个标准输出文件名，这里表示屏幕。符号“<<”是运算符，它指示计算机把其两端用双引号括起来的一个字符串输送到cout文件即屏幕上。由于cout、<<的定义说明都在系统提供的头文件iostream.h之中，因此，凡是程序中需要使用cout、<<等标准输入/输出机制时，第4行的包含指令就必须列出。

4）此程序的执行结果在屏幕上显示为：

```
hello world!
```

当我们编写完程序文本后，要将它存储到磁盘上，形成后缀为.cpp的文件，称为C++源文件。再经过编译系统的编译、连接后，产生后缀为.exe的可执行文件。本书的例题都是在Microsoft Visual C++6.0集成环境下开发的。

1.1.2 C++字符集

字符集是构成C++语言的基本元素。用C++语言编写程序时，除字符型数据外，其他所有部分都只能由字符集中的字符构成。C++语言的字符集由下列字符构成。

英文字母：A～Z，a～z

数字字符：0～9

特殊字符：空格 ! # % ^ & * _(下划线) + = - ~ < > / \ ‘’ “” ; . , () [] {}

1.1.3 ASCII码

ASCII是美国标准信息交换码（American Standard Code for Information Interchange）的英文缩写，ASCII码表把95个基本（可打印）符号和33个控制字符共128个字符与七位二进制数0000000～1111111共128个数码建立了对应关系，实际上任何一个基本符号在计算机内的表示形式都是这样一个二进制数码。计算机本身不能区分不同的字母、数字或特殊符号，它是根据每个符号对应的码值来识别这些基本符号的。ASCII码表如表1-1所示。

表1-1 ASCII码表

代码	字符	名称	代码	字符	名称	代码	字符	名称	代码	字符	名称
000	NUL	无效	019	DC3	设备控制	038	&		057	9	
001	SOH	标题始	020	DC4	设备停机	039	'	单引	058	:	冒号
002	STH	正文始	021	NAK	信息否认	040	(	圆括号	059	;	分号
003	ETX	正文尾	022	SYN	同步	041	)	圆括号	060	<	小于
004	EOT	传递止	023	ETB	块传送止	042	*	乘号	061	=	等于
005	ENQ	查询	024	CAN	作废	043	+	加号	062	>	大于
006	ACK	信号确认	025	EM	纸用完	044	,	逗号	063	?	问号
007	BEL	响铃	026	SUB	置换	045	-	减号	064	@	
008	BS	退格	027	ESC	换码	046	.	圆点	065	A	
009	HT	水平制表	028	FS	文件分离	047	/	除号	066	B	
010	LF	换行	029	GS	分组	048	0		067	C	
011	VT	垂直制表	030	RS	记录分离	049	1		068	D	
012	FF	换页	031	US	元素分离	050	2		069	E	
013	CR	回车	032		空格	051	3		070	F	
014	SO	移出	033	!		052	4		071	G	
015	SI	移入	034	“	双引	053	5		072	H	
016	DLE	换码	035	#		054	6		073	I	
017	DC1	设备控制	036	$		055	7		074	J	
018	DC2	设备控制	037	%		056	8		075	K	

（续）

代码	字符	名称	代码	字符	名称	代码	字符	名称	代码	字符	名称
076	L		089	Y		102	f		115	s	
077	M		090	Z		103	g		116	t	
078	N		091	[		104	h		117	u	
079	O		092	\		105	i		118	v	
080	P		093	]		106	j		119	w	
081	Q		094	^		107	k		120	x	
082	R		095	_		108	l		121	y	
083	S		096	`		109	m		122	z	
084	T		097	a		110	n		123	{	括号
085	U		098	b		111	o		124	|	竖线
086	V		099	c		112	p		125	}	括号
087	W		100	d		113	q		126	~	波纹
088	X		101	e		114	r		127	DEL	删除

其中，0~31和127共33个二进制码所对应的是控制字符，例如008 BS为“退格”，013 CR为“回车符”，这些字符与计算机的控制有关。

从032~126共95个可打印字符构成了C++基本字符集。

由于计算机的最基本存储单元是字节，一个字节有8个二进制数位，一般一个基本字符的代码占用一个字节。用8位二进制编码可表示0~255共256个字符。在不同的计算机中有不同的扩展ASCII码表。在扩展的ASCII码表中，128~255号字符一般是不同语言的特殊字符，字符制表符及其他特殊符号。增加了这些符号，方便了制表输出和特殊符号输出，但它们仍不能作为C++语言的基本符号。

在C++语言中，有一个关于字符的特别处理方法，就是把字符和它的编码等同看待，在表示上不予区分。

例1-2：一个字符类型的数据可以代表该字符本身，也可以代表该字符的ASCII码。

```
#include<iostream.h>
void main( ){
  char  ch='H';                    //定义一个字符型变量ch，并将其内容初始化为“H”
  int   j=ch;                      //定义整型变量j，并初始化为ch的内容“H”对应的ASCII码值
  cout<< ch <<endl<< j;            //在屏幕上显示ch和j的内容
}
```

程序在VC++ 6.0环境下的运行结果为：

```
H
72
```

由于ch为字符型变量，并被赋值为'H'，因此输出ch的结果为'H'。而j定义为整型变量，j又被赋值为'H'，H字符对应的ASCII码是72，所以j输出为72。

1.2 C++程序的基本组成成分

1.2.1 词法记号

词法记号是最小的词法单元，主要包括C++的标识符、关键字、文字、运算符、分隔符和空白符。

1. 关键字

关键字是C++编译器预定的具有特定含义的保留字，在程序中它们不能被作为标识符使用。C++增加了C不具有的关键字，并且不同版本C++编译器含有不同的关键字。以下是标准C++中主要的关键字：

asm	auto	bool	break	case	catch	char
class	const	const_cast	continue	default	delete	do
double	dynamic_cast	else	enum	explicit	exterm	false
float	for	friend	goto	if	inline	int
long	nutable	new	namespace	operator	private	protected
public	reinterpret_cast	register	return	short	signed	sizeof
static	static_cast	struct	switch	template	this	throw
true	try	typedef	typeid	typename	union	unsigned
using	virtual	void	volatile	while		

2. 标识符

标识符是程序员或系统定义的符号，用以标识变量、类型和函数等。以下是使用标识符时的注意事项。

1）标识符由字母、数字和下划线组成，它必须以字母及下划线开始。

2）C++中大小写字母被认为是两个不同的字符。

3）标识符不能是C++的关键字。

4）为标识符取名时，习惯使用有意义（能够反映其用途）的单词或缩写，这样可以提高程序的可读性。

5）虽然标识符长度可以是任意的，但不同的C++编译器能识别的最大长度是有限的。对于超长度的标识符，编译器忽略其多余的字符，而不给出语法错误信息。

例如：line、Draw_point_01、Stu_1都是合法的标识符，而point.1，1则不是合法的标识符。

3. 字面常量

字面常量是程序中直接使用符号表示的数据，包括数字、字符、字符串和布尔字面常量，在本章后面部分将介绍各种字面常量。

4. 操作符（运算符）

操作符是用于实现各种运算的符号，例如：+、－、×、/……在后面部分将会介绍。

5. 分隔符

分隔符用于分隔各个语法成分的单词，程序中的分隔符有点像文章中的标点符号。例如：“”表示一个字符串的开始与结束；“；”表示一个语句的结束。C++分隔符包括：

() {} , : ;

6. 空白

在程序编译时的词法分析阶段将程序正文分解为词法记号和空白。空白是空格、制表符（按tab键产生的字符）、换行符（按Enter键所产生的字符）和注释的总称。

空白符用于指示词法记号的开始和结束位置，但除了这一功能之外，其余的空白将被忽略。因此，C++程序可以不必严格地按行书写，凡是可以出现空格的地方，都可以出现换行。例如：

```
int m;
int
m;
```

是等价的。尽管如此，我们在书写程序时，仍要力求清晰易读。因为一个程序不仅要让机器读懂，还要让人来阅读，以便于修改和维护。

注释是对程序的注解和说明，以便于理解和阅读源程序，而对程序的功能实现不起任何作用，编译系统在对源程序进行编译时不理会注释部分，因此，该注释内容不会影响最终产生的可执行程序的大小和效率。适当地使用注释，能够提高程序的可读性。

在C++中，可以使用两种注释方法。

方法一：使用“/*”和“*/”围起的注释行，这时延用C语言的规则。例如：

```
int m=6;            /*m为整型变量并初始化为6*/
cout << m;          /*输出m的值*/
```

方法二：使用“//”。例如：

```
int m=6;            //m为整型变量并初始化为6
cout << m;          //输出m的值
```

1.2.2 常量

常量是指在程序执行中不变的量，它分为字面常量和有名常量（又称标识符常量）。如25，－3.26，‘a’，“constant”等都是字面常量，即字面本身就是它的值。有名常量是一个标识符，对应着一个存储空间，该空间中保存的数据就是该有名常量的值，这个数据是在定义有名常量时赋予的，以后不能改变。如C++保留字中的true和false就是系统预先定义的两个有名常量，它们的值分别为数值1和0。

1. 整型常量

整型常量简称整数，它有十进制、八进制和十六进制三种表示。

（1）十进制整数

十进制整数由正号（+）或负号（－）开始，首位为非0的若干个十进制数字所组成。若前缀为正号则为正数，若前缀为负号则为负数，若无符号则默认为正数。

例如：38、－25、+120、74286等都是符合书写规定的十进制整数。

（2）八进制整数

八进制整数由首位数字为0，后接若干个八进制数字（借用十进制数字中的0～7）组成。八进制整数不带符号位，隐含为正数。

例如：0、012、0377、04056等都是八进制整数，对应的十进制整数依次为0、10、255和2094。

（3）十六进制整数

十六进制整数由数字0和字母x（大、小写均可）开始，后接若干个十六进制数字（0～9，A～F或a～f）组成。同八进制整数一样，十六进制整数也均为正数。

例如：0x0、0X25、0x1ff、0x30CA等都是十六进制整数，对应的十进制整数依次为0、37、511和4298。

（4）在整数末尾使用u和L字母

对于任一种进制的整数，若后缀有字母u（大、小写等效），则硬性规定它为一个无符号整型（unsigned int）数；若后缀有字母L（大、小写等效），则硬性规定它为一个长整型

(long int) 数。在一个整数的末尾，可以同时使用u和L，并且对排列无要求。如25U、0327UL、0x3ffbL、648LU等都是整数，其类型依次为unsigned int、unsigned long int、long int和unsigned long int。

2. 字符常量

字符常量简称字符，它以单引号作为起止标记，中间为一个或若干个字符。如‘a’、‘%’、‘\n’、‘\012’、‘\125’、‘\x4F’等都是合乎规定的字符常量。每个字符常量只表示一个字符，当字符常量的一对单引号内多于一个字符时称为转义字符，则将按规定解释为一个字符。如‘a’表示字符a、‘\125’解释为字符U（稍后便知是如何解释的）。

因为字符型数据的长度为1，值域范围是－128～127或0～255，而在计算机领域使用的是ASCII字符，其ASCII码值为0～127，正好在C++字符型值域内。所以，每个ASCII字符均是一个字符型数据，即ASCII字符集中的一个值。

对于ASCII字符集中的每个可显示字符（个别字符除外），对应的C++字符常量就是它本身，对应的值就是该字符的ASCII码，表示时用单引号围起来。对于像“回车”、“换行”那样具有控制功能的字符，以及对于像单引号、双引号那样作为特殊标记使用的字符，就无法采用上述的表示方法。为此引入了“转义”字符的概念，其含义是：以反斜线作引导的下一个字符失去了原来的含义，而转义为具有某种控制功能的字符。如‘\n’中的字符n通过前面使用的反斜线转义后就成为一个换行符，其ASCII码为10（换行符）。为了表示用作特殊标记使用的可显示字符，也需要用反斜线字符引导。如‘\’’表示单引号字符，若直接使用‘’’表示单引号是不行的，因为此时的单引号具有二义性。另外，还允许用反斜线引导一个具有1至3位的八进制整数，或一个以字母x作为开始标记的具有1至2位的十六进制整数，对应的字符就是以这个整数作为ASCII码的字符。如‘\0’，‘\12’，‘\73’，‘\146’，‘\x5A’等对应的字符依次为空字符（其ASCII码为0。注意：它不同于空格字符，空格字符的ASCII码为32），换行符，‘;’，‘f’和‘Z’等。

由反斜线字符开始的符合上述使用规定的字符序列称为转义字符序列，C++语言中的所有转义字符序列如表1-2所示。

表1-2 转义字符序列对应表

转义序列	ASCII码	对应功能或字符	转义序列	ASCII码	对应功能或字符
\a	007	响铃	\\	092	反斜线
\b	008	退格	\'	039	单引号
\f	012	换页	\"	034	双引号
\n	010	换行	\?	063	问号
\r	013	回车	\0ddd	0ddd	八进制整数
\t	9	水平制表	0xhh	hh	十六进制整数
\v	11	垂直制表	\0	000	串尾符

转义字符序列不但可以作为字符常量，也可以同其他字符一样使用在字符串中。如“abc\n”字符串中含有四个字符，最后一个为换行符，“\tx=”中的首字符为水平制表符，当输出它时将使光标后移8个字符位置。

对于一个字符常量，当用于输出显示时，将显示出字符本身或体现出相应的控制功能，当出现在计算表达式中时，将使用它的ASCII码值。例如下列一段程序：

```
char ch = 'A';
```

```
int B = ch+2;
if(ch <'M') cout << ch << '<' << 'M' << endl;
cout << "hello\n" ;
```

第一条语句定义字符变量ch并把字符A赋给它作为其初值，实际是把字符A的ASCII码65赋给ch。第二条语句定义整型变量B并把ch+2的值67赋给它。第三条语句首先进行ch< 'M'比较，实际上是取出各自的值（即对应的ASCII码值）进行比较，因条件成立，所以执行其后的输出语句，将向屏幕输出"A<M"。第四条语句输出一个字符串，即原样输出hello并使光标移到下一行开始位置。

3. 逻辑常量

逻辑常量是逻辑类型的值，C++用保留字bool表示逻辑类型，该类型只含有两个值，即整数0和1，用0表示逻辑假，用1表示逻辑真。在C++中还定义了这两个逻辑值所对应的有名常量false和true：false的值为0，表示逻辑假；true的值为1，表示逻辑真。

由于逻辑值是整数0和1，所以它也能够像其他整数一样出现在表达式里，参与各种整数运算。

4. 枚举常量

枚举常量是枚举类型中的值，即枚举值。枚举类型是一种用户定义的类型，只有用户在程序中定义后它才能被使用。用户通常利用枚举类型定义程序中需要使用的一组相关的有名常量。枚举类型的定义格式为：

```
enum  <枚举类型名> { <枚举表> };
```

这是一条枚举类型定义语句，该语句以enum保留字开始，接着为枚举类型名，它是用户命名的一个标识符，以后就直接使用它表示该类型。枚举类型名后为该类型的定义体，它是由一对花括号和其中的枚举值表所组成，枚举表为一组用逗号分开的由用户命名的有名常量，每个有名常量又称为枚举常量或枚举值。

（1）定义枚举常量和枚举变量

例如：

```
enum color{red, yellow, blue};
enum day{Sun, Mon, Tues, Wed, Thur, Fri, Sat};
```

第一条语句定义了一个枚举类型color，用来表示颜色，它包含三个枚举值red、yellow和blue，分别代表红色、黄色和兰色。

第二条语句定义了一个枚举类型day，用来表示日期，它包含7个枚举值，分别表示星期日、星期一至星期六。

一种枚举类型被定义后，可以像整型等C++固有的类型一样在允许出现数据类型的任何地方使用。如可以利用它定义变量和给该变量赋相应的枚举值：

```
enum color c1, c2,c3;              //定义3个color枚举类型变量c1, c2, c3
enum day today, workday;           //定义2个day枚举类型变量today, workday
c1=red;                            //给枚举变量c1赋值为red
workday= Fri;                      //给枚举变量workday赋值为Fri
```

第一条语句开始的保留字enum和类型标识符color表示上述定义的枚举类型color，其中enum可以省略不写，后面的三个标识符c1、c2和c3表示是该类型的三个枚举变量，每一个枚举变量可以用来表示该类型枚举值表中列出的任一个值。

第二条语句开始的两个分量（分量之间的空格除外）表示前面定义的枚举类型day，同样

enum可以省略不写，后面用逗号分开的两个标识符today和workday表示是该类型的两个枚举变量，每一个枚举变量用来表示该类型枚举值表中列出的七个值中的任一个值。

第三条语句把枚举值red赋给枚举变量c1，第四条语句把枚举值Fri赋给枚举变量workday。

（2）定义枚举常量的值

在一个枚举类型的枚举值表中列出的每一个枚举常量都对应着一个整数值，该整数值可以由系统自动确认，也可以由用户指定。若用户在枚举值表中一个枚举常量后加上赋值号和一个整型常量，则表示该枚举常量被赋予了这个整型常量的值。

例如：

```
enum color{red, yellow, blue};
```

用户没有特殊说明，则花括号{}围起的枚举常量由C++编译系统自动赋值，每个枚举常量按其排列顺序从0开始依次赋一个整数值。在本语句中，red = 0、yellow = 1、blue = 2。如果用户在枚举表中特别指定某个枚举常量的整数值，则其后的每个枚举常量按其排列顺序从该整数值+1开始依次被赋予一个整数值。

例如：

```
enum day{Sun=7, Mon=1, Tues, Wed, Thur, Fri, Sat};
```

用户指定了Sun的值为7，Mon的值为1。对于这个被修改定义的day枚举类型，各枚举常量的值依次为7、1、2、3、4、5、6。

若用户没有给一个枚举常量赋初值，则系统给它赋予的值是它前一项枚举常量的值加1，若它本身就是首项，则被自动赋予整数0。

由于各枚举常量的值是一个整数，所以可把它同一般整数一样看待，参与整数的各种运算。又由于它本身是一个有名常量，所以当作为输出数据项时，输出的是它的整数值，而不是它的标识符，这一点同输出其他类型的有名常量是一致的。

例如：

```
enum weekday { Sun,Mon,Tue,Wed,Thu,Fri,Sat}w1 = Wed ,w2 = Sat;
```

对于被定义的weekday枚举类型，各枚举常量的值依次为0、1、2、3、4、5、6。该语句同时又定义两个该枚举类型变量w1和w2，并给w1赋值为3，w2赋值为6。

5. 实型常量

实型常量简称实数，它有十进制的定点和浮点两种表示方法，不存在其他进制的表示。

（1）定点表示

定点表示的实数简称定点数，它是由一个符号（正号可以省略）后接若干个十进制数字和一个小数点所组成，这个小数点可以处在任何一个数字位之前或之后。例如：.12、1.2、12.、0.12、−12.40、+3.14、−.02037、-36.0等都是符合书写规定的定点数。

（2）浮点表示

浮点表示的实数简称浮点数，它是由一个十进制整数或定点数后接一个字母e（大、小写均可）和一个1至3位的十进制整数所组成，字母e之前的部分称为该浮点数的尾数，之后的部分成为该浮点数的指数，该浮点数的值就是它的尾数乘以10的指数幂。例如：3.23E5、+3.25e−8、2E4、0.376E−15、1e-6、−6.04E+12、.43E0、96.e24等都是合乎规定的浮点数，它们对应的数值分别为：3.23×105、3.25×10^{-8}、20000、0.376×10^{-15}、10^{-6}、-6.04×10^{12}、0.43、96×10^{24}等。

对于一个浮点数，若将其尾数中的小数点调整到最左边第一个非零数字的后面，则称它