第1章 C++语言概述

本章从程序设计语言的发展讲起，概述了面向对象方法的概念、特点，面向对象程序设计语言的产生和特点等，并通过一个简单例题讲述了C++程序的组成及上机步骤。

1.1 程序设计语言概述

随着计算机技术的飞速发展，程序设计语言作为一种描述算法的手段，也在不断地充实、完善和发展。

当1945年第一台数字电子计算机ENIAC(Electronic Numerical Intergrator And Computer)产生的时候，并不存在现代意义上的程序设计语言。1946年冯·诺依曼机的出现，尤其是存储程序和程序控制概念的提出，是计算机发展史上的一个里程碑，于是出现了原始的程序设计语言，即机器语言。

机器语言程序设计要求程序员用数字代码和地址编写程序，这就要考虑许多与硬件有关的细节，而且机器语言的程序都是二进制代码，程序的阅读和修改相当麻烦，迫使人们去研究探索新的设计方法。

汇编语言(Assembly Language)是用指令的英语助记符代替了机器语言中的操作码，地址和数据除用二进制、八进制或十进制表示以外，还可用符号名表示。汇编程序是一个把符号语言翻译成等价的机器语言的翻译程序。它是计算机系统程序的一个组成部分。用汇编语言编制的程序是一个用户程序，需要通过汇编程序翻译（汇编）成机器可执行的机器语言代码。虽然汇编语言采用了助记符来编写程序，但由于它和机器语言指令仍存在一一对应关系，仍属于面向机器的语言，需要涉及许多硬件细节问题，不易被普通用户掌握及推广使用。

高级语言是一种采用命令或语句的语言，它屏蔽了机器细节问题，提高了语言的抽象层次，容易为人们理解和记忆，尤其是结构化程序设计语言的出现，给编程带来了方便。

第一个被广泛采用的高级程序设计语言是FORTRAN语言(Formula Translating System)。它是为工程和科学计算设计的，自20世纪50年代问世以来，人们用FORTRAN语言开发了大量的应用程序和系统。在土木工程界广泛应用的SAP优化程序，就是用FORTRAN语言开发完成的。

其他的高级语言如ALGOL、PASCAL、BASIC、C等许适用于不同的领域。但它们都有一个共同的特点，就是不能直接由计算机执行，必须经过一个翻译的步骤，就像两个不同语种的人交谈必须有翻译一样。这些高级语言是面向问题的语言。

高级语言执行的形式有两种，一种是解释执行方式，一种是编译执行方式。高级语言的解释执行方式是指，在计算机中配备了这种高级程序设计语言解释程序的前提下，将所

设计的程序放在计算机的存储器中，由该解释程序对所设计的高级语言程序逐句解释，并执行。编译执行方式是指由这种高级语言的编译程序将所设计的高级语言程序经过分步筛选和整体优化翻译，将这个高级语言程序翻译成完整的机器代码程序，可在同类型的任一台计算机上执行而无需借助于翻译程序。高级语言的解释和翻译程序都是计算机系统软件的组成部分。

面向对象的语言是一种更高级的语言。面向对象的设计使设计程序的出发点更注重于所要解决问题的对象及其相互关系，更接近于自然语言，这更符合人们对客观事物的认识，使得程序各个模块的独立性更强，程序的可读性和可理解性更好，程序代码结构更趋合理。从 20 世纪 90 年代起，各种可视化的程序设计语言应运而生，如目前应用广泛的 VB(Visual Basic)、VFP(Visual FoxPro)、VC++（Visual C++）、Delphi 等。

1.2 面向对象的程序设计概述

.2.1 面向对象的方法

在面向对象的概念提出之前，人们普遍使用的是面向过程的结构化程序设计方法。结构化程序设计的思路是“自顶向下，逐步细化”，其程序结构按功能划分为若干个基本模块，各个模块间尽可能简化关系，功能独立。模块内部均由三种基本结构（顺序结构、选择结构、循环结构）组成，由子程序（函数）具体实现。结构化程序设计可以把一个较为复杂的程序设计任务分解为许多易于控制和处理的子任务，分块解决，具有很多优点。但它把数据和处理数据的过程分离开来，当数据结构改变时，所有相关的处理数据过程都要进行相应的修改，程序的可重用性较差，对大型程序设计很难适应。

面向对象的设计则是把数据及对数据的处理方法放在一起，作为一个对象，对同类型的对象抽象出其共性形成类。类可以通过一个简单的外部接口与外界联系，对象与对象之间通过消息进行通信，加上类的继承和派生特性，使程序模块间关系更为简单，独立性更强，数据的安全性有了保障，提高了程序的可重用性，方便了软件的开发和维护。

.2.2 面向对象的几个概念

1. 对象

一般来说，对象(object)是现实世界中存在的一个有形的或无形的事物，具有静态和动态特征。在程序设计中，对象是对要解决问题中的客观事物的抽象表示，它是构成程序的一个基本单位，由属性和行为构成。属性描述对象的静态特征、行为描述对象的动态特征的操作序列，一般用函数表示。

2. 类

类(class）是具有相同属性和服务的若干对象的集合。其内部有属性和行为两个主要部分。类实际上是一种自定义的类型，对象是类的一个实例。

3. 封装性

封装就是把对象的属性和服务结合在一起成为一个独立的系统单位，并尽可能屏蔽其内部细节，只能通过预留的有限接口对外发生联系。

4. 继承性

一个类（基类）可以派生出新的类，派生出的类可以全部或部分继承基类的数据或程序代码，这种特性称为继承和派生性。如世界上有各种各样的运输工具，可以将它们的共性抽象为运输工具类，再将各种各样的汽车的共性抽象出来，形成汽车类。汽车显然是运输工具的派生类，它具有运输工具的全部共性，另外还有其自身的特性。同样我们还可以将各种各样的轿车抽象出来，形成轿车类，它不仅具有运输工具类的全部共性，还有汽车类的全部共性，另有轿车自己的特性。所以轿车继承了运输工具和汽车的属性和服务。这时，汽车类就成了轿车类的基类，但它又是从运输工具类派生出来的。

继承和派生类一方面可以减少程序设计的错误（利用原来已调好的类），另一方面可以加快和简化程序设计，提高工作效率。

5. 多态性

多态性是指向不同对象发送同一消息，不同的对象具有不同的行为。所谓消息即为按一定格式传递的信息。由函数重载和运算符重载以及虚函数等实现。

封装性、继承和派生性、多态性是面向对象方法的三大特性。

.2.3 面向对象的软件开发

在面向对象的方法出现之前，面向过程的方法在软件开发中起到了很大的作用，解决了很多问题。但随着信息量的剧增，问题复杂程度的增加，很多问题用面向过程的方法无法解决。面向对象开发是以对象为中心的开发方法，是从客观世界中抽象出来的软件开发的新的思维方法，更符合人的思维方式，由于它具有抽象、封装、继承和多态等特性，这就保证了面向对象开发的软件的可重用性和易维护性。

面向对象方法包括分析、设计、编程、测试和维护五个阶段。

1. 面向对象的分析(Object Oriented Analysis)

从问题的陈述开始，逐步建立起客观事物的模型，分析模型中的对象，使得对象的描述与客观事物相一致，无论是对单个事物还是对事物之间的关系，都保持它们的原貌。相同特性的对象为一类，一般类和特殊类之间有继承关系。事物之间有静态联系和动态联系，这些都是客观事物的本来面目。所以，面向对象的系统分析对问题域的认识和分析是直接的，因此能够很好地映射客观事物。

2. 面向对象的设计(Object Oriented Design)

在设计阶段，主要是把在面向对象分析阶段建立的模型，用面向对象的方法产生一个具体实现。其中包括两方面的工作，一是把面向对象的分析模型直接到面向对象的设计作为面向对象设计的一部分；另外就是针对具体实现中的人机界面、数据存储、任务管理等因素补充一些与实现有关的部分，这些部分与分析模型采用相同的表示方法和模型结构。

3. 面向对象的编程（Object Oriented Programming)

编程是面向对象的软件开发最终落实的重要阶段。它主要是要用面向对象的语言实现

面向对象设计方案中的每个成员。

4. 面向对象的调试(Object Oriented Debugging)

调试的任务是发现软件中的错误，这是任何一个软件产品都必须经过的过程。在面向对象的调试中，以对象的类作为一个基本测试单位，查错范围是类定义之内的属性和服务以及接口所涉及的部分。这样可以更准确地发现错误，提高测试效率。

5. 面向对象的维护(Object Oriented Maintenance)

软件作为一件产品，无论经过多么严格的测试，总还会存在一些错误。所以，软件的维护是不能省略和停止的。

由于面向对象开发的程序和问题域是一致的，各阶段的表示也是一致的，从而可以减少软件维护人员的理解难度。发现程序中的错误可以直接追溯到问题域，因需求发生变化追踪到程序也比较容易。由于系统中对象的封装性，使得对某个对象的修改对其他对象的影响很小，这样就可以给软件的调试和维护带来极大的方便，提高工作效率。

由于面向对象的软件开发其分析问题的出发点与人们认识客观事物是一致的，所以更容易理解和掌握。关于面向对象的软件开发可参考有关书籍。

1.3 C++语言的发展和特点

1.3.1 C++ 语言的发展

C++语言是一种面向过程和面向对象都适用的混合型语言，它是在 C 语言的基础上逐步发展和完善起来的，而 C 语言则是在吸收了其他语言的一些优点后逐步发展为实用性很强的语言。

C 语言是在 B 语言的基础上发展起来的，它的根源可以追溯到 ALGOL60。ALGOL60 是 1960 年出现的一种面向问题的高级语言，它离硬件比较远，不适合编写系统程序。 1963 年英国剑桥大学推出了意在更接近硬件的 CPL(Combined Programming Language)语言。由于 CPL 语言规模较大，难以实现，因此 1967 年英国剑桥 Matin Richards 对 CPL 语言作了简化，推出了 BCPL(Basic Combined Programming Language) 语言。同年美国贝尔实验室的 K. Thompson 在 BCPL 的基础上进一步作了简化，设计出了简单而又接近硬件的 B 语言，并用 B 语言写了一个 UNIX操作系统。由于 B 语言过于简单，功能有限，且目标代码运行速度慢， 1972 年，美国贝尔实验室的 D.M.Ritchie 在 B 语言的基础上设计出了 C 语言，并在第二年和 K.Thompson 合作用 C 语言重写了 UNIX 操作系统。现在的 UNIX 操作系统就是在此基础上发展起来的。

后来的 C 语言又经过多次改进，在 1977 年出现了不依赖于具体机器的 C 语言编译程序，并开始移植到非 UNIX 环境中，逐步成为一种独立的程序设计语言。 1988 年美国国家标准协会 ANSI 在继承和发展 D.M.Ritchie和 K.Thompson 的C语言的基础上进行了标准化，产生了 ANSI C。目前所有的C 语言编译系统都是以它为基础的。

C 语言和其他高级语言一样，从程序设计角度看是“面向过程”的，从语言的处理方法上看是“面向问题”的。 20 世纪 80 年代提出的面向对象(Object Oriented）的概念以来，

也产生了许多面向对象的程序设计语言，如 Smalltalk、Actor 等纯面向对象的程序设计语言，C++、Delphi 等混合型的面向对象的程序设计语言，即在传统的过程型程序设计语言中加入了面向对象的成分。随着 C 语言的应用推广，C 语言的一些缺陷和不足也开始表露出来，如对数据类型的检查机制较为薄弱，缺少支持代码重用的结构等。

1980 年美国贝尔实验室的 Bjarne Stroustrup 等对 C 语言进行了改进和扩充，并引入了类的概念，加入了运算符重载、引用、虚函数等功能，使 C++成为目前最为流行的一种混合型程序设计语言。

1.3.2 C++语言的特点

C++语言具有下列主要特点：

① C++完全兼容 C，具有 C 语言的“简捷、紧凑，运算符丰富，可直接访问机器的物理地址，使用灵活方便，程序书写形式自由”的特点。大多数的 C 语言代码略作修改或不修改就可在 C++ 的集成环境下运行。

② C++作为一种面向对象的程序设计语言，它使程序的各个模块间更具独立性，程序的可读性更好，代码结构更加合理，对设计和编制大型软件更为方便。

③ 用 C++ 设计的程序扩充性强。

C++在面向对象的程序设计语言方面获得主导地位，很大程度上是由于它继承了 C 语言的主要特征，再加上它加强了数据类型的检查机制，引入了类的概念等。目前影响最广的可能是 Microsoft 的 Visual C++ 和 Borland 公司的 Borland C++等。

1.4 简单的 C++程序介绍

1.4.1 简单的 C++例子

下面先介绍一个 C++程序，说明 C++程序的基本组成结构和特点。

例 1–1 程序名为 SA1.CPP。

```
#include <iostream.h>
void main()
{ int j，k，l;
   cout<<"j="<<"k=";
   cin>>j>>k;
   l=j+k;
   cout<<"l="<<l<< '\n' ;
}
```

该程序经过编译、连接、运行时，屏幕上出现提示： j= k=

这时从键盘键入两个整数 100，200，再键入回车后(↙)，屏幕上显示

l=300

这就是程序的运行结果。

这个源程序要求以.CPP 为扩展名，即 C++程序一般要求以.CPP 为扩展名。

由这个程序我们可以看到 C++源程序的一般格式为：

预处理指令　　#include <iostream.h>

main 函数　　main()

　　{

C++代码块　　　C++程序的语句

　　}

下面简单介绍源程序的格式：

(1)包含文件或编译预处理指令

在程序的第一行#include <iostream.h>是一条 C++编译指令，包含了 C++的标准输入输出流，它不是 C++的语句，不需要分号结束。

(2)主函数 main()

任何一个 C++程序都要有一个主函数 main()，而且只能有一个。main()可以出现在程序的任何位置，是程序执行的起点。一个程序中可以包含有一个或多个除 main()以外的其他函数,函数不是命令而是一个有名的程序段。

(3)花括号{}

花括号用来将函数体括起来，任何函数体均以“{”开始,以“}”结束。函数体可以包含有若干条语句，每一条语句都以分号结束。一般一行写一条语句，当然短语句也可以一行写多个，长语句可以一条写多行，长语句一行写不完自动转下行，一般不需要加续行符“\”。如果要加续行符“\”，一般加在行尾,两个单词的中间,不能将一个单词分开或将一个字符串分开。

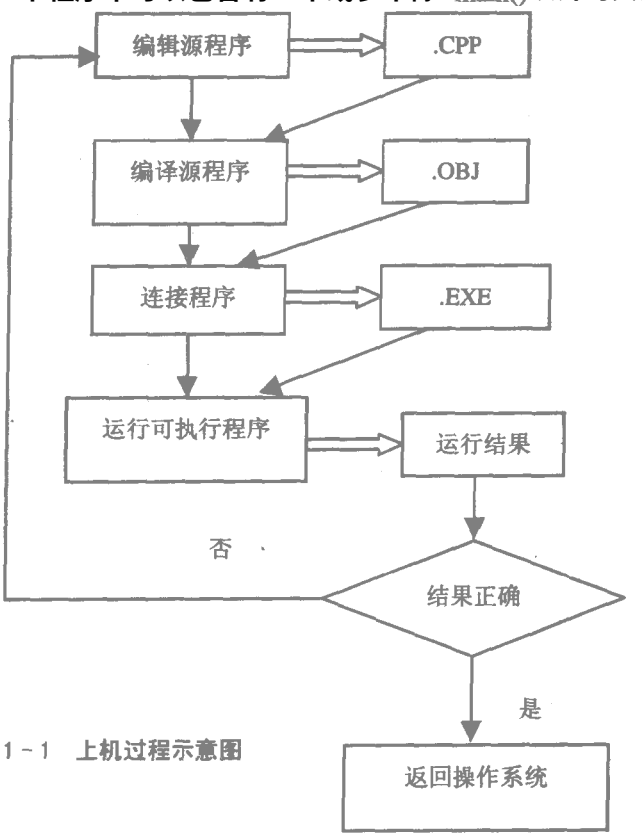

图 1－1　上机过程示意图

在程序 SA1 的函数体中,用到了输入、输出流的对象 cin 和 cout 它们分别和插入运算符“>>”、提取运算符“<<”连用来完成输入和输出的操作，它们不是 C++语言的组成部分。

1.4.2 开发步骤

在前面我们介绍过高级程序语言的执行方式有解释性和编译性两种。C++语言是一种编译性语言，在设计好一个源程序后，要经过编辑、编译、连接，生成可执行文件后，才能运行，见上页图 1-1。

有关上机的具体步骤，可参见本书附录 A。

小 结

本章简单介绍了面向对象设计方法和程序设计语言的特点等。

语言是一套具有语法、词法规则的系统，可以描述人类的行为和思维过程。计算机程序设计语言是计算机能识别的语言，可以完成人和计算机的交流。它可以分为机器语言、汇编语言、高级程序设计语言和面向对象的语言。目前，C++是一种应用最广的面向结构和面向对象的混合型语言。

程序设计语言的学习过程是一个动手操作很强的过程，在学习时大家不能只掌握语法规则，而忽视上机操作。一定要多思考、多动手操作，才能学好本课程。

思考与练习

1-1 试述 C++语言的发展史。

1-2 试述面向对象语言的特点。

1-3 什么是封装？如何理解继承和派生？

1-4 简述程序的基本组成。

1-5 简述面向对象方法的几个过程。

第 2 章　数据类型

一个程序应包括数据的描述和动作的描述两方面内容，即程序设计应包括结构特性的设计和行为特性的设计。结构特性的设计指在问题的求解过程中，计算机所处理的数据以及数据之间联系的表示方法；而行为特性的设计是指完整描述问题求解的全过程，并精确定义每个解题步骤，即算法设计。

本章主要讨论 C++ 的基本数据类型。

2.1 基本数据类型

不论用何种语言设计程序，都要涉及到数据的描述和处理。数据类型决定了该类型对象的存储和对该类型对象能执行的操作或运算。

C++中的数据类型比较丰富，可以分为基本数据类型和派生数据类型两类。基本数据类型是 C++ 系统中预定义的内部数据类型，有 char、int、float、double、void 型。派生数据类型则是根据用户需要，按照 C++ 规则由基本数据类型构造出来的数据类型，有指针、数组、结构体、联合体、枚举、类等类型。

表 2-1 列出了 C++中的基本数据类型。字符型用来存放一个 ASCII 码字符或一个 8 位的二进制数；整型用来存放一个整数，其占用的字节数一般为 4 个(32 位机)字节 实型用来存放实数，双精度型用来存放双精度数，它们占用的字节数根据机型不同而有所不同：无值型将在后面章节介绍。

表 2-1　基本数据类型

类　型	名　称	占用字节数	存储值范围
char	字符型	1	−128~127
int	整型	4	-2^{31}~$(2^{31}-1)$
float	实型	4	-10^{38}~10^{38}
double	双精度型	8	-10^{308}~10^{308}
void	无值型	0	无值

表 2-1 中的基本数据类型可以通过关键字对其进行修饰。关键字（keyword）是 C++ 中预留的有特殊含义和用途的英文单词，在程序中不能另作它用。C++共有 48 个标准关键字(详见附录 D)。例如 short 表示短，long 表示长，signed 表示有符号，unsigned 表示无符号等。这四个修饰词和基本数据类型组合后的数据类型见表 2-2。用 short、long、unsigned、signed 这四个关键字修饰 int 类型时，int 可以省略，例如：unsigned int 可简写为 unsigned。

无修饰词的 int 类型和 char 类型，编译程序认为是有符号的。即相当于 signed int 和 signed char。

表 2-2 C++中所有基本数据类型

类 型	名 称	占用字节数	存储值范围
char	字符型	1	-128~127
signed char	有符号字符型	1	-128~127
unsigned char	无符号字符型	1	0~255
short int	短整型	2	-32768~32767
signed short int	有符号短整型	2	-32768~32767
unsigned short int	无符号短整型	2	0~65535
int	整型	2	-2^{31}~$(2^{31}-1)$
signed int	有符号整型	4	-2^{31}~$(2^{31}-1)$
unsigned int	无符号整型	4	-0~$(2^{32}-1)$
long int	长整型	4	-2^{31}~$(2^{31}-1)$
signed long int	有符号长整型	4	-2^{31}~$(2^{31}-1)$
unsigned long int	无符号长整型	4	-0~$(2^{32}-1)$
float	实型	4	-10^{38}~10^{38}
double	双精度型	8	-10^{308}~10^{308}
long double	长双精度型	8	-10^{308}~10^{308}

2.2 常量和标识符常量

在程序执行过程中，其值不能改变的量称为常量。 C++中除了常量外，还有一种在程序中不能变化的量，叫标识符（符号）常量。常量和标识符（符号）常量是两个不同的概念，下面分别讨论。

2.2.1 常量

常量在程序中一般直接给出数值或字符。根据数据类型，常量可分为整型、实型、字符型和字符串型四种。可以用十进制、八进制和十六进制表示。

1. 整型常量

整型常量是指不含小数点的整数，它可以有正负号。如果是正号，可以省略。

十进制整数的表示和日常表示一样，由 0~9 组成。如：100，34，-15 等都是十进制数。

八进制整数的表示以 0 开头，由 0~7 组成。如：012、0456 、-045 都等是八进制数。

十六进制整数的表示以 0X 或 0x 开头，由 0~9，a~f(或 A~F)字母组成。如：0x2a、0X10AB、0XEDD9 等都是十六进制数。

长整型常量可以在数值后加后缀 l(或 L)表示，无符号型常量可以在数值后加后缀 u 或

(U)表示。如：12345l 是一个长整型常量；3452ul 是一个无符号长整型常量。

2. 实型常量

实型常量是指包含有小数点或 10 的方幂的数，也称浮点数。有十进制小数和十进制指数两种。

十进制小数形式由数字和小数点组成（必须有小数点）。例如：1.25、3.14、0.0 -123.9 等都是实数。

指数形式是指以 10 的方幂表示的数，也称科学表示法。由小数和指数两部分组成，两者缺一不可。指数部分用 e 来表示方幂。注意方幂 e 前面必须是数字，e 后面必须为整数。例如：1.24e3 表示 1.24×10^3，-4.3e-2 表示 -4.3×10^{-2}。但 1.3e、e2、1.2e3.1 等都是不合法的实数。

3. 字符常量

字符常量是用单引号（撇号）括起来的一个字符。例如： 'e'、't'、'B'、'E'等表示一个字符常量。

要注意的是'e'和'E'是不同的字符常量，它们代表不同的 ASCII码值。

还有一部分字符不能直接显示或者不能从键盘上输入，这时必须采取转义序列表示。转义序列是字符常量的另外一种表示方法。转义序列就是用转义符“ \”开始，后跟一个字符或一个整型常量（字符的 ASCII 码值）的方法来表示一个字符。若转义符后边跟的是一个整型常量，则必须是一个以 0 为前缀的八进制或以 x 为前缀的十六进制数，其大小在 0~255 之间。若转义符后跟八进制数时，前缀 0 可以省略。如'\021'、'\x78'、'\0'、'\56'等都是合法的字符型常量。

转义符后面跟有字符时，其字符必须小写，大写只能表示其自身。如果转义字符后面跟大写字符，则忽略反斜杠，作为一个一般的符号处理。例如： '\E'，则认为就是'E'。常 用的转义序列如表 2-3。

表 2-3 C++中常用的转义字符及意义

转义字符	名　　称	功能或用途
\a	响铃（报警）	输出
\b	退格（Backspce 键）	退回一个字符
\f	换页	输出(用于打印机)
\n	换行	输出
\r	回车	输出
\t	水平制表符(Tab 键)	输出
\v	纵向制表符	输出
\\	反斜线	用于输出或文件的路径名
\'	单引号	输出
\"	双引号	输出
\0	空字符	输出

表中的转义字符“ \ ”、“\' ”、“ \" ”虽然既可从键盘上输入，又可显示，但由于它们在

C++中的特殊用途，所以当它们作为字符常量出现时，一般也要采用转义序列表示。

例如可以用'\"'、'\42'、'\0x22'来表示字符常量双引号（42、0x22 是“ ”的 ASCII 码的八进制和十六进制值），但若将双引号用于字符串时，必须采用转义序列表示法。

4. 字符串常量

用一对双引号将 0 个或若干个字符括起来，称为字符串常量。例如： "Nanjing is a beautiful city\n"、"abc"、"a"、" "等都是合法的字符串。

字符串常量以 ASCII 码 0值结尾。即以“ 0 ”作为字符串常量的结束标志。所以 "X"和'X'是不同的。前者是一个字符串常量，占两个字节（其中有一个字符串结束标志'\0'），后者是一个字符常量，占一个字节。

编译系统在处理字符串常量时，会自动在字符串常量的尾部加上'\0'。

2.2.2 标识符常量（常变量）

标识符是用户定义的一个名字，用以表示函数、变量和常变量等。一个合法的 C++标识可以是一个由字母开头，后跟若干个字母、数字等有效符号的名字，长度可达 32 位。C++中是区分大小写字母的，number1 和 NUMBER1 是两个不同的标识符。

标识符的选取建议要“见名知意”，尽量和要表示的内容有关，以便于阅读程序，不能用 C++关键字作为标识符。

标识符常量是指用一个标识符来表示一个常量。 C++中可用两种方法表示，一种是用类型修饰符 const；另一种是用编译预处理指令 define 定义。

1. const 说明符

const 说明常量的格式为：

const 类型标识符= 常量值

例如：

```
const int   A=100;
const float PI=3.14;
const float price=0.88;
```

但 const int class=20 ；是错误的，因为 class 是 C++中的关键字，不能作为标识符。

在定义标识符常量时，要注意：

① 一般将标识符常量大写，以便和普通变量相区分。

②关键字不能作为标识符。

③取名时尽量做到“见名知意”。

2. 用预处理的宏定义指令

define 定义常量的格式为：

#define 标识符 常量值

例如：

```
#define PI 3.1415926
#define N 100
#define ST "Nanjing"
```

上面定义了实型常量 PI 的值为 3.1415926，整型常量 N 的值为 100,字符串常量 ST 为 "Nanjing"。

要注意，在程序中，标识符常量必须先定义、后使用，而且只能使用，不能改变其值。

有关编译预处理指令和 const 的详细使用，我们在后面的章节中将陆续介绍。

2.3 变量

在程序执行过程中，其值可以改变的量称为变量。其作用是存储程序中需要处理的数据。变量有三个特性：

①每个变量必须有一个名字。

②每个变量属于一种类型。

③每个变量只能存放其类型允许的值。

变量名作为变量的标识，应是一个合法的 C++标识符。

2.3.1 变量类型的说明

因为变量的类型决定了该变量所能执行的运算，以及编译程序为其分配的存储空间的大小，并和变量名一一对应起来，所以变量必须在使用前先定义、后使用。

C++中变量定义的格式一般如下：

存储类别 数据类型 变量名 1，变量名 2，…，变量名 n;

其中，存储类别是可选项，可根据需要确定有无；数据类型是 C++中允许的类型,包括基本数据类型和后面讲到的派生数据类型；变量名要求是合法的标识符，在一条语句中，可以定义一个变量也可以同时定义若干个变量。例如：

```
int l，m，n;
float w，x，y，z;
char ch1;
double z1;
```

第一条语句定义了三个整型变量 l、m 和 n。第二条语句定义了四个浮点型变量 w、x、y和 z。第三条语句定义了一个字符变量 ch1。第四条语句定义了双精度型变量 z1。在 C++中，变量说明是作为说明语句来处理的。它可以出现在程序中任何语句可以出现的地方，不一定要放在程序的开头，只要在使用前说明即可。一个变量在程序中只能说明一次。

2.3.2 变量的值

变量的值有两个方面，一个是变量本身所表示的数值；另一个是变量所占内存空间的地址值。

一旦对一个变量进行了说明，在编译或运行时，系统就在内存中开辟一存储单元用来存放变量的值。例如：

```
int m，n；
m=10；
n=20；
```

在编译时，系统会给 m 分配一个 4 个字节的空间存放 10 这个值，给 n 也分配一个 4 个字节的空间存放 20 这个值。

一个变量一旦获得一个值，便一直保留，直到有新的值来更新它为止。

在定义变量的时候赋值称为给变量赋初值或称为变量的初始化，在变量定义以后再赋值称为赋值。两者都是给变量赋值的，效果一样。

例 2–1　计算公路的设计交通量，计算公式为 $N_d=N_0(1+\gamma)^{n-1}$，其中：N_d为远景设计交通量（辆/日）；N_0表示起始年交通量（辆/日）；γ表示年平均增长率；n 为远景设计年限。

```
#include <iostream.h>
#include <math.h>
void main()
{
 long int nd，n0，n；    // 定义了三个变量
 float gama；    // 定义变量
 cout<<"Please input N0，n，gama="<<'\n'；    // 输出提示符
 cin>>n0>>n>>gama；    // 输入变量值
 nd=n0*pow((1+gama)，n-1)；    // 计算公式
 cout<<"Nd="<<nd<<'\n'；    // 输出运算结果
}
```

运行情况：

```
Please input N0，n，gama=
3000 10 1.5↙
Nd=11444091
```

在程序中，定义了三个长整型的变量 nd、n0、n 和一个浮点型变量 gama，其作用是通知编译系统，在内存中分配三个长整型的空间和一个浮点型空间存放这四个值。 γ 因在键盘上无法表示，故用字符 gama 表示。

2.4 类型转换

C++中类型转换有算术类型转换（隐式转换）和强制类型转换两种方式。

2.4.1 算术类型转换

在表达式中，一般要求运算符的两个操作数类型相同。如果两个操作数的类型不同，则需要进行转换，使其类型一致。这种转换的原则是由低向高自动转换，是可靠的。

例如：

```
int a=1;
float b=2.2;
```

则 a+b 这个表达式由于“ + ”两侧的操作数 a 和 b 类型不同，需要转换为同种类型。在 C++ 中， int 的类型低于 float 类型，所以将 a 转换为和 b 相同的 float 类型，然后再进行计算。

这种由低类型向高类型转换，数据精度不受影响。

各种类型的高低顺序如下（由低到高）：

int →unsigned →long →unsigned long →double →long double

转换时 char 型和 short 型自动转换为 int 型； float 型自动转换为 double 型。

2.4.2 强制类型转换

强制转换是指将表达式类型强制转换为指定类型。

强制类型转换格式为：

强制转换类型（表达式）

或（强制转换类型）表达式

强制类型转换可以由低类型向高类型转换，精度不受损失；也可以由高类型向低类型转换，此时要注意数据精度一般要受损失。编写程序时应尽量避免这种由高级到低级的转换。例如：

```
int a=2;
float b=3.2，c;
a=int (b);
```

将 b 强制转换为 int 型，所以 a 的值等于 3，而非 3.2 ，数据精度受到损失。

```
c=float (a);
```

将 a 强制转换为 float 型，所以 c 的值等于 2.0，数据精度没有受到损失。

2.5 枚举类型(enum)

枚举类型是 C++中的一种派生数据类型。它是若干个枚举常量的集合。每个枚举常量有一个整型（或无符号字符型）的值。枚举变量只能取这些枚举常量的值，即每个枚举变量只能有几种可能的取值。

枚举类型的格式为：

enum 枚举标记 {枚举常量 1,枚举常量 2，...,枚举常量 n};

其中关键字 enum 表示枚举型声明的开始，分号表示结束。

在声明了枚举类型后，枚举标记就成了一个类型名，可以通过枚举标记直接定义该枚举类型的变量。例如：

```
enum week{sun，mon，tue，wed，thu，fri，sat}  ;
enum week workday，weekend;
```

上面第一条语句先把 week 声明为枚举类型，第二条语句把 workday 和 weekend 定义为 week 枚举类型的变量。这两个变量的值只能取 sun 到 sat 中的一个。例如：

workday=mon; weekend=sun;

或在声明枚举类型时直接定义枚举变量。例如：

enum week{sun，mon，tue，wed，thu，fri，sat} workday，weekend;

也是把 workday 和 weekend 定义为 week 枚举类型的变量。

枚举元素作为常量，其值在没有特别指定时为 0，1，2，…(按顺序排列,例如式中 sun=0，mon=1，…，sat=7)；在定义时可以指定枚举元素的值。例如：

enum week(sun=7，mon=1，tur，wed，thu，fri，sat);

则 sun 的值为 7，mon 为 1，thu 为 2，…，sat 为 6。

需要注意的是：

① 在 C++编译中，对枚举元素按常量处理，称为枚举常量。在程序中，不能对枚举常量赋值。

②枚举元素可比较大小。例如 (if workday>=mon) …

③一个整数不能直接赋给一个枚举变量，应进行强制类型转换才能赋值。例如：workday=(enum week) 2;

④可用枚举常量对枚举变量赋值。例如： workday=tur;

2.6 类型定义(typedef)

C++中的数据类型除了基本数据类型和派生数据类型外，还可以使用 typedef 来定义类型名。typedef定义类型的一般格式为：

typedef 类型说明符标识符；

类型说明是 C++中基本数据类型的说明或派生类型的说明，标识符是为该类型规定的新的符号名。

类型名定义中，标识符作为类型名说明，一般用大写字母以提高程序的可读性。例如：

typedef unsigned char BYTE;

即将 BYTE 定义成了无符号类型说明符，之后程序中既可用 BYTE 也可用 unsigned char 作为无符号类型的类型说明符。即 unsigned char ch；和 BYTE ch ；是等价的，都说明了 ch 是一个无符号字符变量。

要特别注意的是：类型名定义 typedef 并不产生新的数据类型，只是给已有的数据类型定义一个别名，以简化程序的书写和提高程序的可读性，方便程序在不同系统之间的移植。

小 结

从本章开始,进入 C++基本语法的学习。

数据是描述对象本身属性的，正是因为数据的不同，一个对象和另一个对象才能相区

分。计算机中的数据类型有基本类型和派生类型两种。基本类型是 C++中预定义的类型，它可以和修饰词一起使用限定变量在计算机中的存储方式和存储空间。 C++中基本数据类型除 void 之外，有整型和实型两大类。字符型本质上是整型，它是长度为一个字节的整数，一般用来存放字符的 ASCII 码，双精度型也是描述的实型数。

枚举类型是一种派生（自定义）的类型，它的变量取值只能取枚举常量值。利用类型定义语句还可以给一个已有的类型取一个别名，以方便程序在不同系统之间的移植。

思考与练习

2-1 C++ 中的基本数据类型有哪些？除基本数据类型外还有什么类型？

2-2 什么是变量？变量的三要素是什么？

2-3 什么是常量？C++ 中常用的常量有些什么类型？

2-4 字符常量和字符串常量有何不同？举例说明。

2-5 枚举类型如何定义？使用时要注意些什么？

2-6 下列变量名中，哪些是合法的？

Mycar my_car all 55A a-bcd while const daf-32 _xyz x.12 var(3) max&min your name

2-7 下列常量中，哪些是合法的？

78 063 c56 0x98 '\07' "\"b" " "abc\n" "\abs"

第 3 章　语句和运算符

C++中对数据进行处理，和其他程序设计语言一样，也是通过运算符完成的。运算符就是完成对常量、变量等操作对象不同运算的符号。运算符的对象称为操作数。对一个操作数进行运算的称为一元（单目）运算符，对两个操作数进行运算的称为二元（双目）运算符，对三个操作数进行运算的称为三元（三目）运算符。

常量、变量等数据类型通过运算符组合在一起构成了 C++的表达式，每个符合 C++规则的表达式将有一个确定的结果，这个结果的类型一般取决于操作数的类型，表达式是构成程序的一个重要元素。当多个运算符组合成一个复合表达式时，运算符的求值次序根据运算符的优先级和结合规则来确定。

语句是 C++中构成程序的基本单位，表达式语句是最简单的 C++语句。

C++是运算符最丰富的语言之一，本章介绍算术运算符、关系运算符、逻辑运算符以及位运算符的优先级和结合规则以及表达式语句。

3.1 基本运算符

3.1.1 算术运算符和算术表达式

1. 算术运算符

C++中有一元算术运算符和二元算术运算符两种，分别完成单目的正负运算以及双目的加、减、乘、除四种运算和求模运算。 C++中没有乘幂运算符，乘幂运算通过多次自乘或函数 pow()完成。

一元算术运算符有：

+　　正数运算符，一般可省略，例如+3，+9。

　　负数运算符，例如-10，-100。

二元算术运算符有：

+　　加法运算符，例如 a+b，3+12，a+55。

-　　减法运算符，例如 a-b，34-c，d-100。

*　　乘法运算符，例如 a*b，11*c，f*10。

／除法运算符，例如 a/b，100/4，30.0/4.0。

%　　求模运算符，例如 2%5，10%3，6%3。

对于加、减、乘法运算，如果运算符两边的操作数都是整数，其运算结果也是整数；如果两个操作数中有一个是实数，其运算结果就是实数。对于除法运算，如果运算符两边的操作数都是整数，其运算结果也是整数，即两数整除；如果两个操作符中有一个是实数，