第 1 章　基本数据类型

程序设计是软件构造活动中的重要组成部分，它以某种程序设计语言为工具，给出解决特定问题的求解步骤。数据类型、常量、变量及运算是编写程序的基础。数据类型定义了一个对象（如常量和变量）应具有何种数值及对其可进行什么样的运算。常量和变量是程序中最基本的数据处理对象。运算用运算符来描述，决定运算对象应该进行何种运算。

1.1　程序与工程化程序设计

1.1.1　程序

从自然语言角度来讲，程序是解决问题方法步骤的描述；从计算机角度来讲，程序是用计算机语言描述问题的求解过程。

程序的执行过程是问题的解决过程，因此，要想解决问题，首先要写出解决问题的正确步骤。

例如，求一元二次方程 $ax^2+bx+c=0$（设 $a \neq 0$）实数根的步骤如下：

第一步　获得系数 a、b、c；

第二步　计算 $d = b^2-4ac$；

第三步　若 $d > 0$

　　计算：$x_1 = (-b+\sqrt{d})/(2a)$, $x_2 = (-b-\sqrt{d})/(2a)$

　　输出：两个实根 x_1 和 x_2，转第六步；

第四步　若 $d < 0$

　　输出：没有实根，转第六步；

第五步　计算：$x_1 = x_2 = (-b)/(2a)$

　　输出：两个相同的实数根 x_1，转第六步；

第六步　结束。

程序设计的过程就是分析、解决问题的方法和步骤，并将其记录下来的过程。在描述问题求解步骤时，就要使用程序设计语言，C 语言就是一种很好的程序设计语言。

1.1.2　工程化程序设计

随着计算机技术的发展，在计算机软件开发和维护过程中遇到了一系列严重问题，从而导致软件开发和维护日益复杂，这种现象称为软件危机。1968 年，北大西洋公约组织的计算机科学家在联邦德国召开国际会议，讨论软件危机问题，正式提出并使用了“软件工程”这个名词，从此诞生了“软件工程”学科。

软件工程是指导计算机软件开发和维护的工程学科，通过采用工程的概念、原理、技术和方法来开发与维护软件，把经过时间检验而证明正确的管理技术和当前能够得到的最好的软件开发方法结合起来。

软件工程的核心思想是把软件看成一个工程产品来处理，把需求分析、可行性研究、工程审核、质量监督等工程化概念引入软件生产中。软件工程包括 3 个要素：方法、工具和过程。方法是完成软件工程项目的技术手段；工具支持软件的开发、管理及文档的生成；过程支持软件开发中各个环节的控制和管理。通过这 3 个要素来达到工程项目的 3 个基本目标：进度、经费和质量。

同任何其他事物一样，一个软件产品也要经历孕育、产生、成长、成熟、衰亡等过程，这个过程一般称为软件生存期，即从软件产生直到消亡的时间间隔。可以把整个软件生存期划分为若干个生命周期，每经过一个生命周期，软件产品就经历一次升级。而一个生命周期又可分为若干个阶段，通常软件生命周期包括可行性分析、需求分析、系统设计（概要设计和详细设计）、编码、测试、维护等阶段，如图1.1 所示。

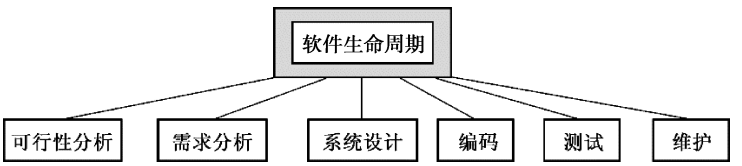

图 1.1　软件生命周期的若干个阶段

1．可行性分析

可行性分析决定“做还是不做”，是整个项目的第一步，其最根本的任务是对以后的行动方针提出建议。这一步涉及成本、人力资源、环境分析、预计回报等诸多因素。可行性分析不能以偏概全，必须为决策提供有价值的证据。

一般地，软件领域的可行性分析主要考虑 4 个要素：经济、技术、社会环境和人。

（1）经济可行性分析主要包括成本分析和收益分析。成本一般包括开发成本、维护成本。收益包括短期收益和长远收益。

（2）技术可行性分析就是分析参与开发人员的技术水平与软、硬件资源能否满足开发要求。技术可行性分析可以简单地表述为：是否能做？能否做得好？是否做得快？

（3）社会环境可行性分析至少包括两种因素：市场与政策。市场又分为未成熟市场、成熟市场和将要消亡市场。

（4）软件开发是复杂的过程，需要各种人员的相互配合，最好是人尽其才。

经过可行性分析，如果问题无法解决，分析员应该建议停止开发，以避免时间、资源、人力和资金的浪费；如果问题值得解决，分析员应该推荐一个较好的解决方案，并且为工程制订一个初步的计划。

2．需求分析

需求分析处于软件开发过程的初期，它对于整个软件开发过程及软件产品的质量至关重要。在该阶段，开发人员要准确理解用户的要求，进行细致的调查分析，将用户非形式的需求转化为完整的需求定义，再由需求定义转化到相应的形式功能规约（需求规格说明）。随着软件系统复杂性的提高及规模的扩大，需求分析在软件开发中的地位愈加突出，需求分析也就愈加困难。

需求分析阶段要完成以下几个主要任务。

1）问题识别

在功能需求上明确所要开发的软件必须具备的功能；在性能需求上明确所要开发的软件的技术性能指标；在环境需求上明确所要开发软件运行时所需要的软、硬件的要求；在用户界面需求上明确人机交互方式、输入/输出数据格式。

2）分析与综合，建立软件的逻辑模型

分析人员对获取的需求进行一致性的分析检查，然后在分析、综合中逐步细化软件功能，划分成各个子功能，以图文结合的形式建立起软件的逻辑模型。

3）编写文档

编写需求规格说明书，把双方共同的理解与分析结果用规范的方式描述出来，作为今后各项工作的基础。编写初步用户使用手册，着重反映所要开发软件的用户功能界面和用户使用的具体要求，用户手册能强制分析人员从用户的角度考虑软件。编写确认测试计划，作为今后确认和验收的依据。修改、完善软件开发计划，在需求分析阶段对所要开发的系统要有进一步的了解，能更准确地估计开发成本、进度及资源要求，所以要对原计划中不太合适的部分进行必要的修正。

需求分析在系统开发过程中的作用如图1.2 所示。

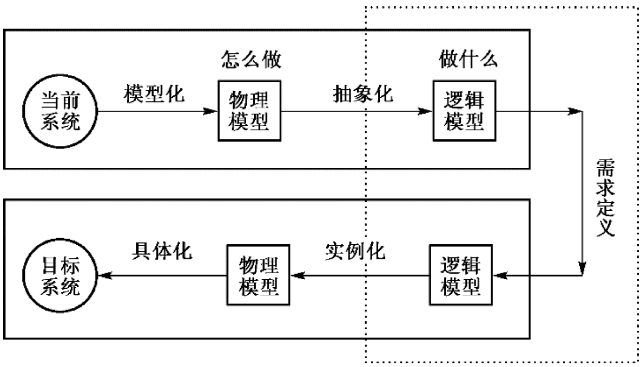

图 1.2　需求分析在系统开发过程中的作用

可以看出，需求分析的核心就是借助当前系统的逻辑模型导出目标系统的逻辑模型，解决目标系统“做什么”的问题。

需求分析有多种方法，常见的有面向数据流的结构化分析（SA）方法和面向对象的分析（OOA）方法。下面以 SA 方法为例来说明需求分析。

具体来说，SA 方法就是用抽象模型的概念，按照软件内部数据传递、变换的关系，自顶向下逐层分解，直到找到满足功能要求的可实现的软件为止。

SA 方法使用的工具包括数据流图、数据词典、判定表与判定树。

数据流图表达了数据处理过程中对数据加工的情况。数据流图中的主要图形元素如图 1.3 所示。描述银行取款过程的数据流图如图 1.4 所示。

数据流图（Data Flow Diagram，DFD）是 SA 方法中用于表示系统逻辑模型的一种工具，它以图形的方式描绘数据在系统中的流动和处理过程，由于它只反映系统必须完成的逻辑功能，所以它是一种功能模型。数据词典（Data Dictionary，DD）用来定义数据流图中的各个元素的具体含义，它以一种准确的、无二义性的说明方式为系统的分析、设计及维护提供了

有关元素一致的定义和详细的描述。它和数据流图共同构成了系统的逻辑模型，是需求规格说明书的主要组成部分。在数据词典中，必须对数据流图中每个被命名的图形元素加以定义，其内容有名字、编号、描述、定义等。

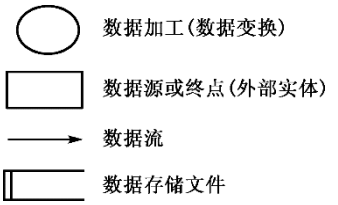

图 1.3 数据流图中的主要图形元素

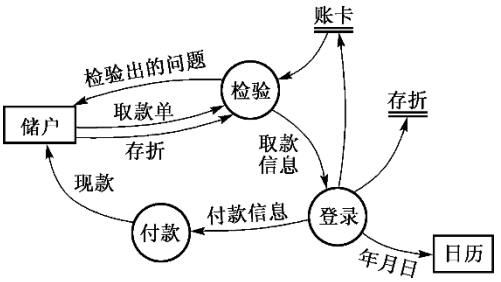

图 1.4 描述银行取款过程的数据流图

判定树是用来表达加工逻辑的一种工具，它比判定表更直观。

3．概要设计

在软件需求分析阶段，已经搞清楚了软件“做什么”的问题，并把这些需求通过需求规格说明书进行详细描述，这也是目标系统的逻辑模型。系统分析员审查软件计划、软件需求分析提供的文档，提出候选的最佳推荐方案供专家审定，审定后进行概要设计。

进入概要设计阶段，要把软件的逻辑模型变换为物理模型，即着手实现软件的需求，并将设计的结果反映在设计规格说明书中，所以概要设计是一个把软件需求转换为软件表示的过程，这种表示只是描述了软件总的体系结构，所以概要设计又称结构设计。

概要设计的基本过程如下。

（1）首先研究、分析和审查数据流图。

（2）然后从软件的需求规格说明书中厘清数据流加工的过程，对于发现的问题应及时解决。

（3）接下来根据数据流图决定数据处理问题的类型。数据处理问题可分为两种：变换型和事务型，针对数据处理问题的两种不同类型分别进行分析处理。

（4）最后由数据流图推导出系统的初始结构图，利用一些启发式原则来改进系统的初始结构图，直到得到符合要求的初始结构图为止。

4．详细设计

详细设计又称过程设计或软件算法设计，该阶段不进行程序编码，是编码的先导，并为以后编码做准备。在这一阶段，对主要模块内部实现细节用到的算法要进行精确的表达，包括以下 6 项主要任务。

（1）为每个模块进行详细的算法设计。用图形、表格、语言等工具将每个模块处理过程的详细算法描述出来。

（2）对模块内的数据结构进行设计。对需求分析、概要设计确定的概念性的数据类型进行确切的定义。

（3）对数据结构进行物理设计。确定数据库的物理结构，物理结构主要是指数据库的存储记录格式、存储记录安排和存储方法，这些都依赖于具体使用的数据库系统。

（4）其他设计。

（5）编写详细的设计规格说明书。

（6）评审。对处理过程的算法和数据库的物理结构都要进行评审。

5．编码

编码阶段的主要任务是使用选定的程序设计语言，把模块的过程性描述转换为用程序设计语言书写的源程序（源代码）。应根据项目的应用领域选择适当的编程语言和编程的软/硬件环境，同时应具有良好的程序设计风格。

6．测试

程序编写完成后要对其进行测试，这是保证软件质量的关键步骤，是对软件规格说明、设计和编码的最后复审，其工件量约占总工作量的 40%以上。测试的目的是尽可能发现并改正被测试程序中的错误，提高软件可靠性。测试的基本流程如图1.5 所示。

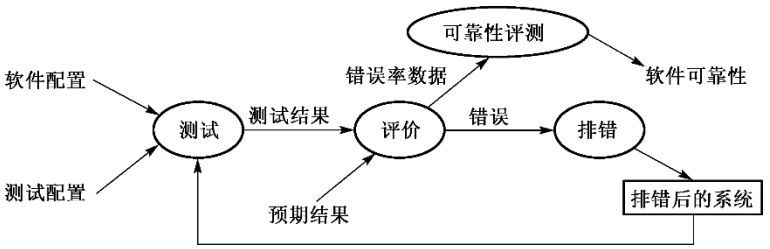

图 1.5　测试的基本流程

一般来说，测试主要分为两个层次：第一个层次是开发过程中的软件测试；第二个层次是第三方测试，测试与软件开发各阶段的关系如图1.6 所示。

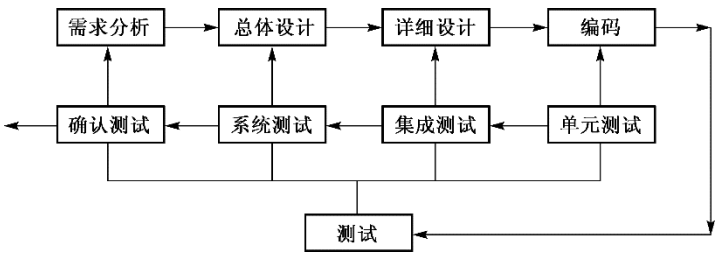

图 1.6　测试与软件开发各阶段的关系

1）开发过程中的软件测试

开发过程中的软件测试是由软件开发方进行的测试，包括单元测试、集成测试、系统测试 3 个主要环节，其目的主要在于发现软件的缺陷并及时修改。

（1）单元测试。单元测试针对编码过程中可能存在的各种错误，如用户输入验证过程中的边界值错误等。

（2）集成测试。集成测试主要针对详细设计中可能存在的问题，尤其是检查各单元与其他程序之间的接口上可能存在的错误。

（3）系统测试。系统测试主要针对概要设计，检查系统作为一个整体是否被有效地运行，如在软件设置中是否达到了预期的高性能。系统测试是保证软件质量的最后阶段。

2）第三方测试

经集成测试和系统测试后，已经按照设计要求把所有模块组装成一个完整的软件系统，接口错误也已经基本排除，接着就要进行第三方测试。第三方测试有别于开发人员或用户进行的测试，其目的是保证测试工作的客观性，主要包括确认测试和验收测试。

（1）确认测试。

确认测试又称有效性测试，是第三方测试机构根据软件开发商提供的用户手册，对软件进行的质量保证测试。确认测试的任务是验证软件的功能和性能及其他特性是否与用户的要求一致、是否符合国家相关标准法规、系统运行是否安全可靠等。

（2）验收测试。

验收测试是软件开发结束后，用户对软件产品投入实际应用前进行的最后一次质量检验活动。它不只是检验软件某个方面的质量，而是要进行全面的质量检验，并且要决定软件是否合格，因此验收测试是一项严格的正式测试活动，要根据事先制订的计划，进行软件配置评审、功能测试、性能测试等多方面检测。

3）常见的测试方法

从测试是否针对系统的内部结构和具体实现算法的角度来看，测试可分为黑盒测试和白盒测试。

（1）黑盒测试。

黑盒测试又称功能测试或数据驱动测试，它是在已知软件应具有功能的基础上，通过测试来检测每个功能是否都能正常使用。在测试时，把程序看成一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，测试者在程序接口进行测试，它只检查程序功能是否符合需求规格说明书的规定，程序是否能接收输入数据产生正确的输出信息并保持外部信息（如数据库或文件）的完整性。

黑盒测试是穷举输入测试，不仅要测试所有合法的输入，还要对不合法的可能输入进行测试。黑盒测试主要有等价类划分、边值分析、因果图、错误推测等方法，主要用于软件确认测试。

（2）白盒测试。

白盒测试又称结构测试或逻辑驱动测试，是指知道软件内部工作过程，通过测试来检测软件内部动作是否按照需求规格说明书的规定正常进行，按照程序内部的结构测试程序，检验程序中的每条通路是否都能按预定要求正确工作。白盒测试是穷举路径测试。在使用这一方案时，测试者必须检查程序的内部结构，从检查程序的逻辑着手，得出测试数据。

白盒测试的主要方法有逻辑覆盖法、基本路径测试法等，主要用于软件验证。

7．维护

维护是指在软件交付使用后，为了改正错误或满足用户新需求而修改软件的过程。要求进行维护的原因多种多样，归结起来有以下3类。

（1）改错需求。改正在特定使用条件下暴露出来的一些潜在程序错误或设计缺陷。

（2）软件移植需求。如果在软件使用过程中数据环境发生变化（如一个事务处理代码发生改变）或处理环境发生变化（如安装了新的硬件或操作系统），则要修改软件以适应这种变化。

（3）增加功能需求。用户和数据处理人员在使用软件时常会提出增加新功能及改善总体性能的要求，为满足这些要求，就要修改软件。

对于这 3 类问题主要进行的维护工作就是纠错性维护、适应性维护和完善性维护。

实践经验表明，各种维护占整个维护工作的比例大致为：纠错性维护占 20%左右，完善性维护占 50%左右，适应性维护占 25%左右，其他维护活动占 5%左右。根据这些统计可以看出，维护工作不仅是改错，大部分维护工作是围绕软件的完善展开的。

1.2　程序设计语言

1.2.1　程序设计语言的概念

程序设计语言是用于编写计算机程序的一组记号和一组规则。程序设计语言包含 3 个方面，即语法、语义和语用。语法表示程序的结构或形式，即表示构成程序的各个记号之间的组合规则，但不涉及这些记号的特定含义，也不涉及使用者。语义表示程序的含义，即表示按照各种方法所表示的各个记号的特定含义，也不涉及使用者。语用表示程序与使用的关系。

程序设计语言的基本成分：

- 数据成分——用以描述程序所涉及的数据；
- 运算成分——用以描述程序中所包含的运算；
- 控制成分——用以描述程序中所包含的控制；
- 传输成分——用以表达程序中数据的传输。

1.2.2　C 语言简介

C 语言是一种比较流行的计算机程序设计语言。它既具有高级语言的特点，又具有汇编语言的特点。它既可以作为系统程序设计语言，编写系统软件，也可以作为应用程序设计语言，编写应用软件。它的应用范围广泛，不仅应用在软件开发上，在其他计算机应用领域也都用到了 C 语言，如单片机及嵌入式系统开发。

1．C 语言的特点

同其他程序设计语言相比，C 语言具有以下主要特点。

（1）语言简洁、紧凑，使用方便、灵活。

C 语言一共只有 32 个关键字、9 种控制语句，压缩了一切不必要的成分，程序书写形式自由，语句简洁。

（2）运算符丰富，适用的范围也很广泛。

C 语言共有 34 种运算符，它把括号、赋值符号、强制类型转换符号等都作为运算符处理，从而使 C 语言的运算符类型丰富，表达式类型多样，灵活使用各种运算符可以实现用其他高级语言难以实现的运算和操作。

（3）数据结构丰富，具有现代化语言的各种数据结构。

C 语言的数据类型有整型、实型、字符型、数组类型、指针类型、结构体类型、共用

体类型等。这些丰富的数据类型能用来实现各种复杂的数据结构（如链、表、树、栈等）的运算。

（4）具有结构化的控制语句。

如 if…else 语句、switch 语句、while 语句、do…while 语句、for 语句等，这些语句可以实现程序中所有的控制结构。另外，函数是 C 语言程序的基本单位，将函数作为程序模块的基本单元，可以实现程序的模块化。

（5）可直接对硬件操作。

C 语言允许直接访问物理地址，能进行位操作，能实现汇编语言的大部分功能。这个特点使得 C 语言既具有高级语言的功能，又具有许多低级语言的特点，可以用来编写系统程序。

（6）目标代码质量好，程序执行效率高。

C 语言生成的目标代码一般只比汇编语言生成的目标代码的效率低 10%～20%。

（7）程序的可移植性好。

与汇编语言相比，用 C 语言编写的程序基本上不用被修改就能被用于各种型号的计算机和操作系统中，具备很好的移植性。

由于 C 语言灵活性大、功能强，可以编写出任何类型的程序，所以 C 语言一直是主流的程序设计语言。

2．C 语言程序的结构

下面通过几个简单的 C 语言程序，认识与体会 C 语言程序的结构。

【例 1.1】 在屏幕上输出一行字符串。

程序如下：

```
#include <stdio.h>                          /*引用标准输入和输出函数*/
#include <stdlib.h>                         /*引用标准 lib 库函数*/
void main( )                                /*主函数，无返回值*/
{
    printf("This is a C program.\n");       /*语句*/
    system("pause");                        /*暂停程序的执行，按任意键可以继续执行*/
}
```

此程序的功能是在屏幕上输出下面的一行信息：

```
This is a C program.
```

其中，main()是主函数，每个 C 语言程序都必须有一个 main()，表示整个 C 语言程序的入口。函数体用花括号“{ }”括起来。本例中主函数内有一个 printf 函数调用语句，它是 C 语言的输出函数，其功能是将双引号中的字符串原样输出。“\n”是换行控制符，即在“This is a C program.”输出之后回车换行。每个语句后要有一个分号（;），表示语句结束。

注意：

① 调用标准输出 printf 函数，要在程序的前面用文件包含命令“#include <stdio.h>”。

② system 函数可使在桌面上直接运行编译后的可执行程序暂停执行，这样就可以在运行的窗口中看到运行结果，按任意键可以继续执行。但调用该函数，要在程序前面用文件包含命令“#include <stdlib.h>”。

【例 1.2】 求两数之和。

```
#include <stdio.h>
#include <stdlib.h>
void main( )
{
    int a,b,sum;                    /*定义变量 a、b、sum*/
    a=123;                          /*给变量 a 赋值，123 为十进制常量*/
    b=0456;                         /*给变量 b 赋值，456 为八进制常量*/
    sum=a+b;                        /*计算 a+b 的和，并将结果赋给 sum 变量*/
    printf("sum is %d\n",sum);      /*输出结果，以十进制形式输出*/
    system("pause");
}
```

本程序的功能是求两个整数的和，并将其输出。“/*……*/”表示注释部分，其作用是提高程序的可读性。注释对编译和运行不起作用，可以加在程序中的任何位置。printf 函数中的“sum is %d\n”是输出格式字符串，其中“%d”是格式字符，用来指定输出时的数据类型和格式，“%d”表示十进制整数类型。在执行输出时，此位置代表一个十进制整型数值。printf 函数中最右端的“sum”表示要输出的变量，现在它的值为 425，因此该程序的执行结果：

```
sum is 425
```

【例 1.3】 输入 a 和 b 两个值，输出其中大者。

```
#include <stdio.h>
#include <stdlib.h>
void main( )                        /*主函数，无返回值*/
{   int max( );                     /*函数声明*/
    int a,b,c;                      /*定义变量*/
    scanf("%d,%d",&a,&b);           /*输入两个整型数给变量 a 和 b*/
    c=max(a,b),                     /*调用 max 函数，将返回值赋给 c*/
    printf("max=%d",c);             /*输出 c 的值*/
    system("pause");
}
int max(int x,int y)                /*定义 max 函数，返回值为整型，x 和 y 为 int 型形式参数*/
{   int z;                          /*定义 max 函数中用到的变量 z*/
    if (x>y)
        z=x;
    else
        z=y;
    return(z);                      /*将 z 的值作为 max 函数返回值带回调用处*/
}
```

本程序由 main 函数和 max 函数组成。

说明：C 语言源程序由函数构成，包含一个主函数和若干个其他函数。函数可以是系统定义的库函数（如主函数中的 printf 函数、scanf 函数，只能调用，无须定义），也可以是用户自己定义的函数（如此例中的 max 函数）。

max 函数的功能是将 x 和 y 中的较大者赋给变量 z。return 语句将 z 的值返回给 main 函数。返回值通过函数名 max 带回到 main 函数的调用处。在 main 函数中，调用了 scanf 系统函数，其作用是通过键盘输入 a 和 b 的值。“&”的含义是“取地址”，表示将输入的值放到 a 和 b 所代表的空间中。

在调用 max 函数时，将实际参数 a 和 b 的值分别传送给 max 函数中的形式参数 x 和 y。经过执行 max 函数得到一个返回值，把这个值赋给变量 c，然后输出 c 的值。

程序的执行结果如下：

```
10,20↙          （输入 10、20 并按回车键，符号↙表示回车）
max=20          （输出 c 变量的值）
```

通过对以上几个例子的分析，可以总结出 C 语言程序的结构有以下特点。

（1）C 语言程序由函数构成。

一个 C 语言程序至少要包括一个 main 函数，即程序是由一个 main 函数和若干个其他函数构成，因此，函数是 C 语言程序的基本单位。被调用的函数可以是系统提供的库函数，如 printf 函数和 scanf 函数，也可以是用户自定义的函数。C 语言的系统函数库十分丰富，编译系统提供了 300 多个库函数。C 语言的这种特点易于实现程序的模块化。

（2）每个函数都由两部分组成：函数说明部分和函数体。

函数的一般格式如下：

```
函数类型 函数名（形式参数表）
{
    函数体;
}
```

其中：

- 函数说明部分包括函数名、函数类型、形式参数表。一个函数名后面必须跟一对圆括号，可以没有参数，如 main()，如果有参数，则应说明每个参数的类型和名字；
- 函数体即函数说明部分下面的花括号“{ }”内的部分，如果一个函数中有多对花括号，则最外层的一对花括号为函数体的范围。函数体一般包括变量定义部分和执行部分。

（3）main()是整个 C 语言程序的入口。

一个 C 语言程序总是从 main 函数开始执行，在 main 函数中结束，其他函数通过被调用而执行。main 函数可以在程序的任何位置。

（4）程序书写格式自由。

一行内可以写几个语句，一个语句也可以分开写在多行上。语句以分号（;）作为结束标志。分号是 C 语言语句的必要组成部分，不可省略，即使是程序的最后一个语句，也必须有分号。

（5）C 语言本身没有输入/输出语句。

在 C 语言程序中，输入/输出功能是由 scanf 和 printf 等库函数来实现的，即 C 语言对输入/输出实行“函数化”。

（6）可以用“/*……*/”对 C 语言程序中的任何部分进行注释，以提高程序的可读性。

1.3　数据类型与常量

1.3.1　数据类型

根据数据的一些共同特性对数据进行归纳分类，抽象出共同点（取值和操作的集合），从而得到了数据类型，即数据类型是一个数据集数据的抽象化描述。

1．基本数据类型

在 C 语言中，无论常量还是变量，都必须有各自的数据类型。在一个具体的 C 语言系统中，每个数据类型都确定了数据 3 个方面的属性：数据范围、在内存中的存放形式及所占存储空间的大小。

C 语言的基本数据类型如图1.7 所示。在程序中用关键字表示对应的类型。

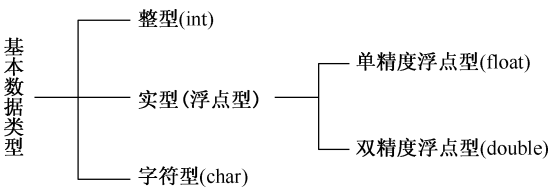

图 1.7　C 语言的基本数据类型

其中，char（表示字符型）、int（表示整型）、float（表示单精度浮点型）和 double（表示双精度浮点型）都是关键字。另外，在某些基本类型之前还可以使用修饰词来改变它们原来的含义，这些修饰词包括 short（短型）、long（长型）、signed（有符号）和 unsigned（无符号）。

例如：

- short int 表示短整型；
- unsigned char 表示无符号字符型；
- long int 表示长整型；
- unsigned short int 表示无符号短整型。

2．不同类型数据的取值范围

在计算机中，由于机器字长的位数通常是固定的（如 32 位或 64 位），因此，计算机中数的表示范围（允许取值范围）是有限的。另外，数有正数和负数之分，其正号（+）或负号（−）在机器中用一位二进制数来表示，称为符号位。通常这个符号位放在二进制数的最高位，一般规定 0 代表正号（+），1 代表负号（−），按该格式存放的数据称为符号数或机器数（用 signed 修饰）。不设符号位的数据称为无符号数（用 unsigned 修饰）。

因此，一个无符号型变量只能存放不带符号的数据，即无符号型变量只能存放一个大于或等于零的数。无符号型变量不能存储负数，它比有符号变量存储的正数的范围大了一倍多。

例如，int 型数据在 VC 6.0 系统中用 4 字节存放，则 signed int 型变量的取值范围为−2 147 483 648～2 147 483 647，而 unsigned int 型变量的取值范围为 0～4 294 967 295。

修饰符 long 一般指存储空间扩大了 int 型的一倍，而修饰符 short 一般指存储空间缩小到

int 型的一半。但不同 C 语言编译系统的具体规定有所不同。VC 系统中基本数据类型、字宽和范围如表 1.1 所示。

表 1.1 VC 系统中基本数据类型、字宽和范围

类 型	类型名（类型说明符）	VC 系统	
		空间（字节）	取值范围
字符型（2 种）	[signed] char	1	−128～127
	unsigned char	1	0～255
整型（6 种）	[signed] int	4	−2 147 483 648～2 147 483 647
	[signed] short [int]	2	−32 768～32 767
	[signed] long [int]	4	−2 147 483 648～2 147 483 647
	unsigned [int]	4	0～4 294 967 295
	unsigned short [int]	2	0～65 535
	unsigned long [int]	4	0～4 294 967 295
实型（3 种）	float	4	可精确到小数点后 7 位有效数字
	double	8	可精确到小数点后 16 位有效数字
	long double	8	可精确到小数点后 16 位有效数字

表注：① 在表中出现的“[ ]”，表示该部分可以省略。如 int 与 signed int 都是表示有符号整型变量的说明符；long 与 long int 都是表示有符号长整型变量的说明符。

② C 语言标准没有规定各类数据所占用的内存位数，所以不同 C 语言编译系统的各类数据所占用的内存位数是不同的。

注意：在本书后面的描述中都是针对 VC 系统来讲的。

1.3.2 常量

常量是指程序在运行时其值不能改变的量，常量有多种不同的数据类型，如图1.8 所示，不同数据类型的常量有不同的表示方式。

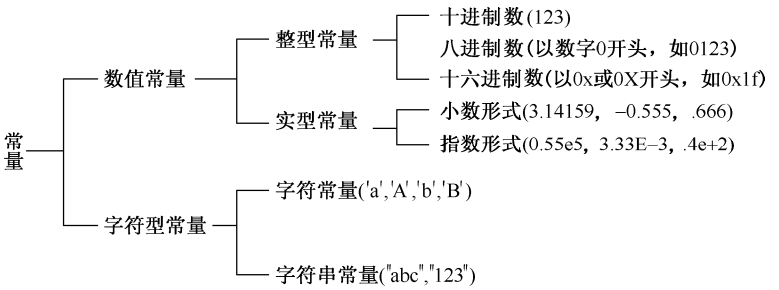

图 1.8 常量的数据类型

常量的类型决定了常量所占内存的大小和取值范围。在 C 语言程序中，常量直接以自身的存在形式体现其值和类型。

例如，在 VC 系统中，123 是一个十进制整型常量，占 4 字节存储空间；123.0 是双精度实型常量，占 8 字节存储空间。

1．整型常量

整型常量有以下 3 种表示形式。

1）十进制整型常量

十进制整型常量是由 0～9 这 10 个数字组成的数值，没有小数部分，书写时它被直接书写出来，如 345、12。

2）八进制整型常量

八进制整型常量是由 0～7 这 8 个数字组成的数值，书写时它以“0”开头，表示它是一个八进制整型常量，如 0345、012。

3）十六进制整型常量

十六进制整型常量是由 0～9 和 A～F（a～f）这 16 个符号组成的数值，书写时它以 0x 或 0X 开头，表示它是一个十六进制整型常量，如 0xdf1。

注意：

① 无符号整型常量要在常数的末尾增加一个字母 u 或 U，如 345u、238U。

② 无符号长整型常量要在常数的末尾增加两个字母 ul 或 UL、lu、LU、Ul、uL，如 256ul、12 345 678UL。

2．实型常量

实型常量又称浮点数。在 C 语言中，实型常量有两种表示形式：小数表示法和指数表示法（又称科学计数法）。

1）小数表示法

浮点数由整数和小数两部分组成。这两部分可以省略其中的一部分，但不能同时省略（小数点不能省略）。例如，12.35、35. 、.689 都是 double（双精度浮点）型常量。

2）指数表示法

指数表示法的表示格式如下：

```
<尾数>E<指数部分>
```

指数部分可正可负，但必须是整数。例如，1e−2、0.5E10、35.56E-3、7.e−2，它们都是 double（双精度浮点）型常量，分别代表 $1*10^{-2}$、$0.5*10^{10}$、$35.56*10^{-3}$、$7.0*10^{-2}$。

注意：

① 用指数形式表示的浮点数必须有尾数且指数必须是整数，如 e4、.e3 和.2e-1.5 的写法都是错误的。

② 浮点数后面的字母 F 或 f 表示该浮点数为 float（单精度浮点）型常量；浮点数后面的字母 L 或 l 表示该浮点数为 long double（长双精度浮点）型常量。

例如：

- 1e−2f 表示 float 型常量；
- 3.14E5L 表示 long double 型常量。

③ 浮点数后面不加字母表示是该浮点数为一个 double（双精度浮点）型常量。

3．字符型常量

1）普通字符型常量

普通字符型常量表示字符本身，书写时要在字符的两边加单引号（'），如''（空字符）、' '（空格字符）、'a'（字符 a）、'x'（字符 x）、'D'（字符 D）等。

在 ANSI C 中，普通字符型常量在计算机内部存放的是该字符的 ASCII 编码。

例如，字符'0'在计算机内部存放的是 48，字符'A'在计算机内部存放的是 65。

注意：普通字符型常量是 1 个字符，如'x'和'y'是合法的，而'xy'是不合法的。单引号（'）是定界符，不属于普通字符型常量的一部分。

2）转义字符

除了普通字符型常量外，还有一类特殊的字符型常量，这就是转义字符，它是以在其两边加单引号并以反斜线（\）开头的字符序列来表示的，意思是将反斜线（\）后面的字符转换成另外的意义。转义字符一般表示“控制字符”，例如，'\n'代表一个“换行”控制符，而不是字符 n。常用转义字符如表 1.2 所示。

表 1.2 常用转义字符

字 符 形 式	含 义	ASCII 编码
\0	空字符（NULL）	0
\a	响铃（BEL）	7
\b	退格符（BS）	8
\t	水平制表符（即跳到下一个输出区）（HT）	9
\n	回车换行（LF）	10
\v	竖向跳格（VT）	11
\r	回车符（CR）	13
\"	双引号字符（"）	34
\'	单引号字符（'）	39
\\	反斜线字符（\）	92
\ddd	1～3 位八进制数所代表的字符	—
\xhh	1～2 位十六进制数所代表的字符	—

注意：普通字符型常量也可以用转义字符形式来表示，其方法就是用反斜线“\”开头，后面跟字符的 ASCII 编码（用八进制数或十六进制数表示），其具体表示方法如下。

① 用'\ddd'表示，其中 ddd 代表不超过 3 位的八进制数。

例如，'\101'代表字符'A'，因为 A 的 ASCII 编码为$(101)_8$；'\60'代表字符'0'（零），因为 0 的 ASCII 编码为$(60)_8$。

② 用'\xhh'表示，其中 hh 代表不超过 2 位的十六进制数。

例如，'\x41'代表字符'A'，因为 A 的 ASCII 编码为$(41)_{16}$；'\x30'代表字符'0'（零），因为 0 的 ASCII 编码为$(30)_{16}$。

【例 1.4】 转义字符示例。

程序如下：

```
#include<stdio.h>
#include <stdlib.h>
void   main( )
{
      printf("%c,%c\n",'\101','\60');
      printf("%c,%c\n",'\x41','\x30');
      system("pause");
}
```

输出结果：

```
A,0
A,0
```

4．字符串型常量

字符串型常量表示字符串本身，是用一对双引号括起来的零个或多个字符的序列。字符串中可以包含空字符、空格字符、转义字符和其他字符，也可以包含汉字。

例如：

```
"How are you! "        表示字符串“How are you!”
""                     表示空字符串
" "                    表示字符串“空格”
"a"                    表示字符串“a”
```

在存储时，一个字符型常量在计算机内部占 1 字节。而字符串型常量在内存中占有连续字节空间，每个字节空间放一个字符。为了标识字符串的结束位置，系统存储时，会自动在字符串最后一个字符的后面存储一个结束符（'\0'），这里的'\0'是空字符的转义字符。也就是说，一个字符串存储时，除了依次存储字符串中所有字符外，还要存储结束符。其所占存储空间为串长+1。

例如，'A'存储时仅占 1 字节，存储字符 A 的 ASCII 编码为 65；而"A"在存储时占 2 字节，第一字节放字母 A 的 ASCII 编码（65），第二字节放结束符（'\0'）。因而，字符型常量'A'与字符串型常量"A"占据的存储空间是不一样的，如图1.9所示。

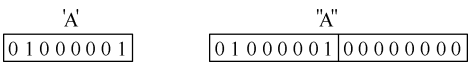

图 1.9　'A'与"A"的存储空间示意图

5．符号型常量

C 语言中，可以通过标识符来表示某个常量，在程序中要使用该常量时，可直接引用标识符。

C 语言中用宏定义命令对符号型常量进行定义，其定义格式如下：

```
#define 标识符 常量
```

其中，#define 是宏定义命令的专用定义符，标识符是对常量的命名，常量可以是任何一种常量的表示形式（标识符和常量之间要有空格）。目的是用指定的标识符来代表指定的常量，这个被指定的标识符就称为符号型常量。

例如：

```
#define PAI 3.1415927
```

在编译预处理时，系统将程序中宏定义命令之后出现的所有符号型常量用宏定义命令中对应的常量一一替代。例如，对于以上两个宏定义命令，预处理时，系统首先将程序中除这两个宏定义命令之外的所有 PAI 替换为“3.1415927”。

习惯上，符号型常量用大写字母表示，而将变量名用小写字母表示。

【例 1.5】 已知圆半径 r，求圆周长 c 和圆面积 s。

程序如下：

```
#define PI 3.1416
main( )
{    float r,c,s;
     scanf("%f",&r);        /*从键盘输入一个实型数给变量 r*/
```

```
    c=2*PI*r;              /*编译时用“3.1416”替换 PI */
    s=PI*r*r;              /*编译时用“3.1416”替换 PI */
    printf("c=%6.2f,s=%6.2f\n",c,s);
}
```

1.4 变量

1.4.1 标识符

程序中有许多对象，为了便于区分和使用，程序语言都规定了在程序中描述对象名字的规则，这些名字包括变量名、常数名、数组名、函数名等，这些为标识对象而为对象取的名字通常称为标识符。

C 语言规定，标识符由字母、数字和下画线（_）字符组成，其中，第一个字符必须是字母或下画线。

例如：

- 下面是合法的 C 语言标识符：

```
my_name    test39    _string1
```

- 下面是不合法的 C 语言标识符：

```
call…name      非字母、数字或下画线组成的字符序列
39test         非字母或下画线开头的字符序列
-string1       非字母或下画线开头的字符序列
```

注意：

① C 语言标识符区分大小写，即同一个字母的大写与小写被看成不同的标识符。

例如，a 和 A、AB 和 Ab 是不相同的标识符。

② ANSI C 规定标识符的长度可达 31 个字符，但一般系统使用的标识符，其有效长度不超过 8 个字符。

③ 标识符不能采用 C 语言系统关键字（又称保留字），C 语言系统关键字见附录 D。

1.4.2 变量的定义

1．变量的概念

在程序运行中，允许改变值的量称为变量。任何一个变量具有 3 个基本要素：变量名、数据类型和值。

变量都有一个区别于其他变量的名字，这就是变量名；每个变量都具有某种数据类型（在定义时指定），编译系统根据变量的数据类型，在内存分配一定的存储空间；在变量所占空间可以存放一个具体的值，这就是变量的值。

变量可以看成一个存储数据的容器，它的功能就是存储数据。

对变量有两个基本操作：

① 向变量中存入数据值，这个操作称为给变量“赋值”。

② 获取变量当前值，以便在程序运行过程中使用，这个操作称为“取值”。

变量具有保存值的性质，也就是说，如果在某个时刻给某变量赋了一个新值，此后使用这个变量时，得到的将是这个新值。

2．变量名

变量通过变量名来区分，变量名命名的规则遵守标识符的命名规则。

使用变量名时，应注意：

① 命名变量名时应尽量做到“见名知意”，这样有助于增加程序的可读性。

② 下画线（_）符号一般是系统函数常用的开始符号，故一般不要用它作为变量名的第一个字符。

③ 习惯上一般变量用小写字母命名，而符号型常量则用大写字母命名。

3．变量定义的语法规则

变量在使用前必须先定义，定义位置一般在函数体的说明部分。可以一次定义一个或多个同型变量，其格式如下：

```
<类型说明符>  <变量名表>;
```

其中：

① 类型说明符是一个有效的数据类型说明符，如整型类型说明符 int、字符型类型说明符 char 等。

② 变量名表示的形式是：变量名 1, 变量名 2, …, 变量名 *n*; 即用逗号分隔的变量名的集合。

③ 分号表示结束定义。

例如：

```
int a,b,c;                /*定义 a、b、c 为整型变量，每个变量在内存中占 4 字节*/
unsigned short d;         /*定义 d 为无符号短整型变量*/
long e,f;                 /*定义 e 和 f 为长整型变量*/
char cc;                  /*定义 cc 为字符型变量，在内存中占 1 字节*/
float m,n;                /*定义变量 m 和 n 为单精度实数*/
double x,y;               /*定义 x 和 y 为双精度实型变量，每个变量在内存中占 8 字节*/
long double g;            /*定义变量 g 为长双精度实数*/
```

1.5　基本运算

1.5.1　运算符与表达式

1．运算符

C 语言提供了丰富的运算符，从而保证操作的方便实现。但是在所提供的运算符中，有些不同功能的运算符使用了相同的符号。另外，运算符还具有优先级和结合性等。所以在使用运算符时，要掌握每个运算符的功能、优先级和结合性，以及使用时的格式规定。

2．表达式

表达式是由运算符和运算对象组成的式子，运算对象就是在程序中要处理的各种数据。最简单的表达式就是一个常量或变量。在表达式中，可以使用圆括号来改变优先级，任何一

个表达式经过计算都有一个确定的值和类型。

在计算一个表达式时，应注意：

① 应首先确定运算符的功能，然后确定计算顺序。

一个表达式的计算顺序是由运算符的优先级和结合性决定。优先级高的先计算，优先级低的后计算。在优先级相同的情况下，由结合性决定，在多数情况下，从左至右；在个别情况下，从右至左。

② 一个表达式的类型由运算符的种类和运算对象的类型决定。

C 语言规定，只有同类型的数据才能进行计算，不同类型的数据要先转换成同一类型的数据，然后进行计算。在一般情况下，系统会自动进行数据类型转换来满足计算需求。

1.5.2 赋值运算

1．变量初始化

在定义变量的同时给变量赋予初始值称为变量的初始化。变量的初始化是变量赋值的一种方式，初始值必须是常量，常量的类型应该和变量的数据类型一致，若不同，系统会自动进行类型转换。

例如：

```
int x;                          /*变量 x 没有初始化，其值是不确定的*/
int   a=10.0;                   /*变量 a 初始化，其值最终是 10*/
float   f=5.55f;
char   c='a';
int   a=15,b=15,d=15;           /*注意不能写为 int a=b=d=15;*/
```

当一个变量初始化后，其值将被保存到分配给该变量的内存单元中，直到重新给该变量赋值。定义变量时，系统为其分配相应的存储空间，而在所分配存储空间内，必然存在一个先前保留下来的无意义值（对于静态变量系统有默认初始化值，其值如 0 或'\0'等）。也就是说对于没有初始化的变量，其值是不确定的。

2．变量赋值

变量赋值是改变变量值的基本方法，给变量赋新值，原来的值会被取代。

变量赋值的语法格式如下：

```
变量名=表达式;
```

其中，赋值运算符“=”是一种能够改变变量值的运算符。即它在计算表达式的值后，还将会改变变量的值。赋值运算符是双目运算符，赋值运算符的左边必须是变量。

简单赋值运算符如表 1.3 所示。

表 1.3 简单赋值运算符

对 象 数	名 称	运 算 符	运 算 规 则	运 算 对 象	运 算 结 果	结 合 性
双目	赋值	=	将表达式值赋给变量	任何类型	变量的类型	自右向左

赋值运算符的运算规则：将赋值运算符右边表达式的值赋给左边的变量；赋值运算符的结合性是自右向左的；赋值运算符的优先级比算术运算符的优先级低。

例如：

```
int x=6;
x=5+10;
```

第一个语句定义了一个 int 型的变量 x，并且初始化为 6。第二个语句是将赋值运算符右边表达式 5+10 的计算结果 15 赋给左边变量 x，从而使变量 x 的值由原来的 6 变为 15。

例如：

```
int x,y,z;
x=y=z=5+6;
```

上面第二个语句的执行过程是：先计算 z=5+6 的值，计算后变量 z 被更新为 11，表达式取 z 的值，即 11。接着，将表达式 z=5+6 的值 11 赋给变量 y。最后将表达式 y=z=5+6 的值 11 赋给变量 x。经过上面的连续赋值，使变量 x、y、z 的值都更新为 11。

3．赋值表达式的类型转换

赋值运算符两边的数据类型若不一致，系统会自动进行类型转换。

转换原则：在赋值表达式中，当左值（赋值运算符左边的变量）和右值（赋值运算符右边的表达式）类型不同时，一律将右值类型转换为左值类型。

（1）浮点型数据赋给整型变量时，舍弃小数部分。

例如：

```
int x;
float y=5.26;
x=y+1.3;
```

将 y+1.3 的计算结果 6.56 赋给整型变量 x 时，由于 6.56 是一个 double 型数值，舍弃小数部分，将整数部分 6 赋给整型变量 x。

注意：当赋值运算符右值大于 int 型的有效范围时，x 将被赋予一个无意义的数值。

（2）将字符型数据赋给整型变量时，将字符的 ASCII 编码赋给 int 型变量。

例如：

```
int x;
x='a';
```

它是将字符'a'的 ASCII 编码 97 赋给 int 型变量 x。

看下面的一个例子：

```
int x=5,y;
y=x+6.89;
```

在表达式 y=x+6.89 中，要做两次隐含转换，先将变量 x 的值转换成 double 类型，然后与 6.89 进行相加运算，最后将相加的结果 11.89（double 型）转换成 int 型数值 11（取整）赋给变量 y，y 的值为 11。

（3）将 int 型或 long 型数据赋给 char 型变量时，将低 8 位传送到 char 型变量（发生截断）。

例如：

```
int i=289;          /*i 中存放的二进制数是 00000000000000000000000100100001*/
```

```
char c;
c=i;                /*将 i 变量中低 8 位二进制数 00100001 存入 c 变量中，即十进制数为 33*/
```

（4）将 signed 型数据赋给长度相同的 unsigned 型变量时，将存储单元内容原样赋值（连原有的符号位也作为数值一起传送）。

【例 1.6】 将 signed（有符号）型数据赋给 unsigned（无符号）型变量。

程序如下：

```
#include<stdio.h>
#include <stdlib.h>
void   main( )
{      unsigned a;
       int b=-1;
       a=b;
       printf("%u\n",a);
       system("pause");
}
```

1.5.3 算术运算

1．算术运算符

算术运算符用来对数据进行简单算术运算。要注意字符型数据也可以看成整型数据参加基本算术运算。算术运算符如表 1.4 所示。

表 1.4 算术运算符

对 象 数	名 称	运 算 符	运 算 规 则	运 算 对 象	运 算 结 果	结 合 性
单目	正	+	取原值	整型或实型	整型或实型	自右向左
	负	-	取负值			
双目	加	+	加法			自左向右
	减	-	减法			
	乘	*	乘法			
	除	/	除法			
	模（求余）	%	整除取余	整型	整型	

这 7 个算术运算符的优先级规定如下：

- 单目运算符优先于双目运算符；
- *、/、%优先于+、-；
- 同级单目运算符的结合性是自右向左；
- 同级双目运算符的结合性是自左向右。

加、减、乘、除四则运算对 int 型、float 型和 double 型变量都适用，而%运算符只用于 int 型运算对象。另外，两个整数相除，结果为整数。

例如：

```
5/2     （结果为 2）
2/5     （结果为 0）
```