第 1 章　C 语言编程入门

本章的知识点：

- C 语言程序代码的基本框架。
- 数据类型。
- 算术运算符和算术表达式。
- 变量和常量。
- 其他运算符。
- 数据的输入与输出。

当你选择本书学习 C 语言时，我相信你已经了解到 C 语言是一种广泛使用的编程语言。有关 C 语言的历史和特点等内容在许多书籍和互联网资源中都有详细阐述，因此这里不再赘述。

本章的目的很简单，即引导你轻松进入 C 语言程序设计的世界。开始时，可以先模仿已有的程序实例编写程序，由浅入深地掌握程序设计的原理和计算思维。简而言之，学习 C 语言的关键在于“实践、实践再实践”。

开始学习 C 语言时，不必过于追求完全理解程序设计的语法，也不要过于纠结细节。随着编写程序数量的增多，很多不懂的东西会在学习过程中逐渐变得清晰。首先要确保你的程序可以成功运行，然后在有一定基础之后，再去深入探索程序背后的原理和细节。

过早地过于关注细节可能会阻碍你的学习进度。当然，在学习过程中遇到各种问题是很正常的，问题本身就是用来解决的。没有遇到问题的学习过程往往是有问题的。只要你不害怕困难，踏实地进行实践，相信你很快就能学有所成。

本章的实例较为简单，主要涉及编程基础的核心概念。对这些概念的理解和掌握都非常重要，因为它们是构建复杂程序的基石。

通过本章的实例，我们将学习到各种重要的内容，包括**数据类型、变量和常量、运算符、输入和输出、表达式及语句**等。无论程序的复杂程度如何，都会用到这些基本概念。

计算机的主要功能在于**存储和计算**。数据需要存储，程序代码也需要存储。而计算涵盖多个方面，包括**算术运算、赋值运算、关系运算、逻辑运算、位运算**等。本章所展示的实例主要集中在算术运算和赋值运算。

下面的代码是基本的 C 语言程序框架。

```
#include <stdio.h>
int main(void)
{
    return 0;
}
```

C 语言程序使用<stdio.h>头文件来包含标准输入和输出函数。

在这个程序中，定义了一个名为 **main** 的函数，它是程序的入口。main 函数没有任何参数（void），并且返回一个整数（int）。

如果希望在 main 函数中执行一些操作，可以在函数的花括号{}中添加代码。当前，花括号内除了 return 0;以外没有其他内容，这意味着程序没有执行任何具体的操作。

这是一个简单的 C 语言程序模板，你可以在 main 函数中添加自己的代码逻辑，用于实现想

要的功能。

一个程序应该至少有输出信息，并且在大部分情况下都需要输入数据，因此程序的开头通常包含**预处理指令**#include <stdio.h>。这个指令的主要目的是告诉 C 编译器，需要在编译时把 stdio.h 头文件的内容加入程序中。

在开始学习编程时，你可能对某些术语和语言语法不熟悉，这并没有关系。最好的学习方法是先模仿一些程序的写法，运行它们，然后再尝试理解其中的原理，这样能加速掌握编程技能。

实际上，计算机程序就是按照你所编写的“它懂得”的语句来执行**数据的存储和计算**。因此，在学习编程之前，你需要了解计算机如何存储数据，数据存储在哪里，以及如何处理数据。我们可以通过一些具体实例来解释这些概念。

一个 C 语言程序至少包含一个函数，即 main 函数。在执行程序时，操作系统会自动调用 main 函数。在接下来的三章编程实例中，所有程序只包含一个 main 函数，但会使用 C 语言的库函数，如输入/输出函数和数学函数等。

main 函数是每个 C 语言程序不可或缺的一部分。根据 ANSI（American National Standards Institute，美国国家标准协会）标准，应该在 main 函数后的圆括号中写上 void，表示 main 函数不接收任何参数。在 main 函数中，一对花括号中的语句称为函数体。通常情况下，程序从函数体的第一条语句开始执行，一直执行到最后一条语句。根据 ANSI 标准，main 函数体中不能缺少 return 语句。

在 C 语言中，函数体内的代码是以语句为单位的。C 语言中的语句是以分号作为结束符的。函数体中的每条语句都是独立的执行单元，执行完一条语句后才执行下一条语句，以此类推，直到函数执行完毕。

先来编写第一个程序实例，结果输出 Hello World!。同时，你可以通过这个实例搭建自己的编程环境，并开始使用 C 语言进行编程。


扫一扫，看视频

【实例 01-01】输出 Hello World!

描述：在计算机屏幕上输出 Hello World!。

分析：想要在屏幕上输出信息，需要用到标准输出函数 printf。

本实例的参考代码如下：

```
#include <stdio.h>              //包含 stdio.h 头文件，用于输入/输出操作
int main(void)
{
    printf("Hello World!");     //使用 printf 函数输出 Hello World!
    return 0;                   //返回整数值 0，表示程序正常结束
}
```

运行结果如下：

```
Hello World!
```

这个 main 函数中共有两条语句，以英文分号作为结束标记的代码称为语句。由双引号引起来的内容是字符串常量。

printf 是一个标准库函数，用于输出文本。在这里，它被用于输出 Hello World!。return 0; 语句表示程序正常结束，并将整数值 0 返回给操作系统。

通过编译和执行这段代码，你将得到一个可执行程序，它将在命令行或终端中输出 Hello World!。

当阅读程序代码时，通常会从程序的主函数 main 开始，然后按照从上到下的顺序逐行阅读。主函数是每个 C 语言程序的入口点，只能有一个。

C 语言是一种编译型语言，这意味着要运行上面的代码，首先需要将代码转换为机器可以理解的形式，这个转换过程由编译器完成。源代码是自己编写的代码，而可执行代码或机器代码是编译器生成的可以直接在计算机上执行的代码。

上面提供的代码可以保存为一个扩展名为.c 的文件，如 helloworld.c。

在开始编写程序之前，首先需要准备一个适合自己的开发环境，以帮助我们完成代码的编写、编译、调试和运行等任务。目前有许多不同的编程平台、**编译器**和**集成开发环境**（Integrated Development Environment，IDE）可供选择。对于初学者来说，选择一个简单易上手的开发工具是一个不错的选择。在积累了一些经验之后，可以尝试使用其他开发工具。

程序员使用计算机语言来表达自己的思想和算法。然而，高级语言（如 C 语言）并不能被计算机直接识别和执行。实际上，我们编写的代码需要经过一系列必要的步骤才能生成可执行程序。这个过程通常包括预处理、编译和链接三个步骤。在这个过程中，预处理器会处理程序中的各种预处理指令并生成预处理后的代码，编译器将预处理后的代码转换为汇编代码，然后汇编器将汇编代码转换为机器码，最后链接器将各个模块的机器码组合成最终的可执行程序。通过这个过程，我们的代码才可以在计算机上运行。

C 语言的预处理命令主要有三种形式：宏定义、文件包含和条件编译。

由于我们通常使用集成开发环境（IDE），预处理、编译和链接三个步骤通常是自动完成的，因此无须过多关注这些步骤的细节。IDE 会自动处理预处理指令、将源代码转换为可执行文件，并最终生成可在计算机上运行的程序。这样，我们可以专注于编写代码和实现算法，而无须手动执行这些步骤。

双斜线（//）后面的内容称为注释，程序代码中的注释是给程序员看的。编译器将源代码转换为目标代码后，完全忽略所有的注释和空白。也就是说，注释不会影响可执行程序。总之，应该在源代码中多加注释、多留空白，提高代码的可读性，方便后期维护。本书的代码都会有注释，目的也是帮助读者理解代码。

C 语言中有两种注释方式：一种是以/*开头、以*/结尾的块注释；另一种是以//开头、直到行末的行注释。行注释是在 C99 标准中引入的。

【实例 01-02】输出多行信息


扫一扫，看视频

描述：在计算机屏幕上输出两行信息。

分析：想要实现换行，需要使用转义字符\n。

本实例的参考代码如下：

```
#include <stdio.h>
int main(void)
{
    printf("我喜欢 C 语言!\n");
    printf("学好编程需要多写代码，实践、实践再实践!!!\n");
    return 0;
}
```

运行结果如下：

```
我喜欢 C 语言!
学好编程需要多写代码，实践、实践再实践!!!
```

这段代码调用两次 printf 函数，会在屏幕上输出两行信息。printf 函数双引号里的\n 称为转义字符，目的是告诉计算机本行输出结束时，后面的输出要另起一行。

你要做的就是在 IDE 中编辑、编译并运行实例 01-02，看一下运行结果。之后把第一个 printf 函数中的\n 去掉，再运行一下这个程序，看看是不是所有的信息都输出在同一行上了。

下面需要模仿前面两个实例编写你自己的程序，是不是觉得编写代码并不是很难，而且还很有趣！

【实练 01-01】编程输出直角三角形


扫一扫，看视频

```
*
**
***
****
```

【实练 01-02】编程输出个人信息

使用 printf 函数在计算机屏幕上输出自己的姓名、爱好以及来自哪里。


扫一扫，看视频

【实例 01-03】输出学生个人信息

描述：以表格的形式输出学生的学号、姓名、性别和年龄。

分析：可以使用转义字符\t 让信息对齐。


扫一扫，看视频

本实例的参考代码如下：

```
#include <stdio.h>
int main(void)
{
    printf("学号\t 姓名\t 性别\t 年龄\n");
    printf("001\t 李四\t 男\t20\n");
    return 0;
}
```

运行结果如下：

```
学号    姓名    性别    年龄
001     李四    男      20
```

转义字符可以提供特殊的格式控制。**转义字符是以反斜杠（\）开头的字符序列**。常用的转义字符见表 1.1。

表 1.1　常用的转义字符

转义字符	含　义
\t	水平制表
\v	垂直制表
\a	响铃
\b	后退一格
\0	空字符（Null），通常用作字符串结束标志
\?	问号
\n	换行
\r	回车
\"	双引号
\'	单引号
\\	反斜杠
\ddd	1~3 位八进制 ASCII 码值所表示的字符
\xhh	1~2 位十六进制 ASCII 码值所表示的字符

在 C 语言中，一些特殊字符具有特殊的含义，如双引号、单引号、反斜杠和问号。如果要在输出中显示这些特殊字符本身，而不是其特殊含义，可以使用转义字符反斜杠来实现。

反斜杠（\）可以告诉 printf 函数以特殊的方式解释下一个字符，如\n 代表换行符，\t 代表制表符。如果想要输出一个双引号字符，可以使用\"；而如果想要输出一个反斜杠字符，可以使用\\。这样，编译器就会知道应该输出这些特殊字符本身，而不是将其解释为其他含义。

请运行以下代码，以深入理解转义字符的使用方法。理解并熟练掌握转义字符的使用方法，有助于编写代码和处理字符串相关的任务。

```
#include <stdio.h>
int main(void)
{
    printf("Hello World!\r");       //回车符，注意不换行，使下一次输出从本行开始
```

```
    printf("China is good!");
    printf("\n");                    //换行符，将下一次输出移到新的一行
    printf("Hello World!");
    printf("\b\b\b\b\b\b");          //六个退格符
    printf("China!\n");              //输出 China!并换行到新的一行
    printf("It\'s my pleasure. \n"); //转义字符\'输出单引号
    return 0;
}
```

运行结果如下：

```
China is good!
Hello China!
It's my pleasure.
```

仅能输出信息是不够的，计算机最重要的能力是计算，所以下面的程序实例介绍如何实现计算，先从最简单的数学计算开始。

【实例 01-04】计算 1+2 和 3+4 并输出结果

扫一扫，看视频

描述：在屏幕上输出 1+2 和 3+4 的计算结果。

分析：想要计算，就需要使用表达式。

本实例的参考代码如下：

```
#include <stdio.h>
int main(void)
{
    printf("1+2=%d\n", 1 + 2);      //输出 1+2 的计算结果 3
    printf("3+4=%d\n", 3 + 4);      //输出 3+4 的计算结果 7
    return 0;
}
```

运行结果如下：

```
1+2=3
3+4=7
```

在 C 语言中，**算术表达式是指使用算术运算符对数值进行计算的表达式**。常见的**算术运算符**包括加法运算符（+）、减法运算符（–）、乘法运算符（*）、除法运算符（/）和求余数运算符（%）。

代码中的 1 + 2 就是一个算术表达式，即使用加法运算符对两个整数进行相加计算。

本实例的 printf 函数由两部分组成，一部分是用双引号引起的**格式控制字符串**，另一部分是**输出项列表**。格式控制字符串的作用是控制输出项的格式和输出一些提示信息。格式控制字符串中的%d 表示输出项以十进制整数输出。**由百分号（%）和一个转换字符组成转换说明符**，转换说明符告诉 printf 函数如何解释待输出项的值，同时也起到**占位**的目的，指明在输出时对应值的输出位置，因此**格式控制字符串中的%也称为占位符**。

计算机处理的数据被分为不同的类型。在这段代码中，1 和 2 被称为**字面整型常量**，它们代表整数值。当这两个整数相加时，计算机将它们作为整数类型进行处理，并返回一个整数类型的结果。

编写 C 语言程序需要遵循语法和语义规则，以确保程序代码的正确性。初学者可能会犯一些简单的错误，如忘记写分号或**误将英文分号写成中文分号**，但只要遵循语法规则并注意语义错误，就能编写出正确的程序代码。

运行以下代码，并分析它的运行结果。

```
#include <stdio.h>
int main(void)
{
    printf("1+2=%d\n", 1 + 2); //输出 1+2=3
    printf("3-4=%d\n", 3 - 4); //输出 3-4=-1
    printf("5*6=%d\n", 5 * 6); //输出 5*6=30
    printf("8/7=%d\n", 8 / 7); //输出 8/7=1
```

```
    return 0;
}
```

运行结果如下：

```
1+2=3
3-4=-1
5*6=30
8/7=1
```

为什么 8/7 的结果是 1 呢？因为 8 和 7 是两个整数，在 C 语言里，除法运算中的两个运算数如果都是整数，结果也是整数，即这里的“/”是整除的含义。

如果把代码中的 “/”改成“%”会是什么结果呢？8 % 7 的结果是 1，因为“%”是求余数的意思，也就是整除后的余数，从数学的角度也就是 8 除 7 的商是 1 并且余数是 1，所以 14/3 的结果是 4，14%3 的结果是 2。可以自行编写代码测试一下。

通过上面的代码及练习，目前我们已经掌握了 5 个算术运算符，由运算符和运算数组成的式子称为**表达式**。在计算机编程中，表达式是算术、逻辑或其他类型的运算过程或计算公式。运算符表示操作的类型，运算数则是参与运算的值。


扫一扫，看视频

【实例 01-05】编程计算并输出(1+8)×2−9÷4+5 的结果

描述：在屏幕上输出(1+8)×2–9÷4+5 的计算结果。

分析：只需要写出正确的算术表达式，通过 printf 函数就可以让计算机输出正确的计算结果。

本实例的参考代码如下：

```
#include <stdio.h>
int main(void)
{
    printf("(1 + 8) * 2 - 9 / 4 + 5 = %d\n", (1 + 8) * 2 - 9 / 4 + 5); //结果为21

    return 0;
}
```

运行结果如下：

```
(1 + 8) * 2 - 9 / 4 + 5 = 21
```

在表达式中，**运算符具有优先级**。如果一个表达式中存在多个算术运算符，按照数学计算规则，乘除法的优先级高于加减法。当然，也可以使用括号改变运算的顺序，先计算括号内的表达式，再计算括号外的表达式。这与数学计算的规则是相同的。因此，对于表达式(4 – 3)* 2，计算结果是 2。因为先计算括号内的减法，得到 1；然后再乘以 2，得到 2。而对于表达式 4 – 3 * 2，计算结果是–2。这是因为先计算乘法，得到 6；然后再计算减法，得到–2。

在这些算术运算符中，求余数运算（%）和乘除运算（*/）的优先级是相同的。当优先级相同时，按照从左到右的顺序进行计算。

由于参与运算的运算数都是整数，因此整个表达式的计算结果也会是整数。对于本实例中的表达式，计算结果是 21。


扫一扫，看视频

【实例 01-06】输出数字的平方和立方

描述：编写一个程序，以表格的形式输出数字 1～5 的平方和立方的计算结果。

分析：通过乘法运算计算数字 1～5 的平方和立方，并使用水平制表符（\t）将相关结果对齐，最终以表格的形式输出。

本实例的参考代码如下：

```
#include <stdio.h>
int main(void)
{
    printf("number square cube\n");
    printf("%d\t%d\t%d\n", 1, 1 * 1, 1 * 1 * 1);
```

```
    printf("%d\t%d\t%d\n", 2, 2 * 2, 2 * 2 * 2);
    printf("%d\t%d\t%d\n", 3, 3 * 3, 3 * 3 * 3);
    printf("%d\t%d\t%d\n", 4, 4 * 4, 4 * 4 * 4);
    printf("%d\t%d\t%d\n", 5, 5 * 5, 5 * 5 * 5);
    return 0;
}
```

运行结果如下：

```
number square cube
1       1       1
2       4       8
3       9       27
4       16      64
5       25      125
```

这段代码的重复性很高，等学会使用循环语句后，可以对这段代码进行重构，写出更简洁且具有良好可读性的代码。

【实例 01-07】十进制整数 13 的八进制数和十六进制数

扫一扫，看视频

描述：输出十进制整数 13 的十进制数、八进制数和十六进制数。

分析：一个整数不同的进制数的输出可以通过格式控制转换说明实现。

本实例的参考代码如下：

```
#include <stdio.h>
int main(void)
{
    printf("十进制整数 13 的十进制数是：%d\n", 13); //十进制
    printf("十进制整数 13 的八进制数是：%o\n", 13); //八进制
    printf("十进制整数 13 的十六进制数是：%x", 13); //十六进制

    return 0;
}
```

运行结果如下：

```
十进制整数 13 的十进制数是：13
十进制整数 13 的八进制数是：15
十进制整数 13 的十六进制数是：d
```

整型常量通常使用**十进制（Decimal）**数字表示，但有时也会用**十六进制（Hexadecimal）**或**八进制（Octal）**数字表示。然而，在计算机内存中，这些整数值实际上被转换为**二进制（Binary）**形式来存储和处理。不同进制的整型常量的表示形式见表 1.2。

表 1.2　不同进制的整型常量的表示形式

进 制	实 例	特　　点
十进制	13	以 10 为基数。由 0~9 数字序列组成
二进制	1101	以 2 为基数。由 0~1 数字序列组成
八进制	015	以 8 为基数。由数字 0 开头，后跟 0~7 数字序列
十六进制	0xd	以 16 为基数。由 0x 或 0X 开头，后跟 0~9、a~f 或 A~F 序列

当需要在程序中输出整数时，可以使用格式控制的方式来指定所需的输出格式，如十进制、八进制或十六进制等。通过在输出函数中使用相应的格式控制字符串，如**%d 表示十进制格式，%o 表示八进制格式，%x 或%X 表示十六进制格式**，程序可以根据需要以特定的形式输出整数。

运行以下代码，以深入理解 %#o、%#x 和 %#X 这些转换说明符的含义。

```
#include <stdio.h>
int main(void)
{
```