项目 1　C 语言程序设计概述

在学习任何程序设计之前，需要先了解一些基础性的知识。对于初学者，本项目可作为快速入门内容；对于有程序设计基础者，可以略过本项目，进而学习后续内容。本项目主要包括：程序设计的基本知识；计算机程序与计算机语言；程序设计的步骤和方法；C 语言的发展及其特性；通过简单的 C 语言例子，介绍了 C 语言程序的组成结构；详细介绍用 Visual Studio 2022 运行 C 语言程序的步骤与方法；给出学习 C 语言的建议和方法。

1.1　计算机程序

人与计算机不能直接互动和交流，如果想操纵计算机，就必须提前安排好各项任务并把任务转化成它能识别的东西，只有这样计算机才会自动进行所有的工作。其实，计算机的每一个操作都是根据人们事先指定的指令进行的。

计算机程序就是一组计算机能识别和执行的指令。每一条指令使计算机执行特定的操作，完成相应的工作。只要让计算机执行这个程序，计算机就会“自动”地执行各条指令，有条不紊地进行工作。为了使计算机系统能实现各种功能，需要成千上万个程序。这些程序大多数是由计算机软件设计人员根据需要设计好的，作为计算机软件系统的一部分提供给用户使用。此外，用户还可以根据自己的实际需要设计一些应用程序，如学生成绩管理程序、财务管理程序、工程中的计算程序等。

总之，计算机的一切操作都是由程序控制的，离开程序，计算机将一事无成。所以，计算机的本质是程序的机器，程序和指令是计算机系统中最基本的概念。只有懂得程序设计，才能真正了解计算机是怎样工作的，才能更深入地使用计算机。

1.2　计算机语言

在世界各国的交流中，不同国家的人使用不同的语言，如果他们之间想交流就需要“翻译”。“翻译”的职能就是懂得双方的语言，起到信息相互沟通的作用。人与计算机之间想进行交流，也需要“翻译”，即人与计算机都能识别的语言，这就是计算机语言。计算机语言一般分为 4 类：低级语言（机器语言、汇编语言）、高级语言（C、C++、Java 等）、专用语言（如 CAD 系统中的绘图语言和 DBMS 中的数据库查询语言）和脚本语言（如 JavaScript、Ruby、Python 等），本书只重点介绍前两类。计算机语言经历了以下几个发展阶段。

1. 机器语言

机器指令就是计算机能直接识别和接受的二进制代码（即 0 和 1 序列）。机器指令的集合就是该计算机的机器语言，它是计算机唯一能够直接识别和执行的语言。

因此，用机器语言编写的程序就是一个个二进制文件。一条机器语言称为一条指令，指令是不可分割的最小功能单元。而且，由于每台计算机的指令系统往往各不相同，所以，在不同计算机上执行同一程序，就必须编写不同版本，这样就造成了重复工作。但由于使用的是针对特定型号计算机的语言，故而运算效率是所有语言中最高的。机器语言是第一代计算机语言。

2. 汇编语言

汇编语言是人们用一些英文字母和数字表示一条指令。例如，用 ADD 代表加法，MOV 代表数据传递，等等。这样一来，人们很容易读懂并理解程序在干什么，纠错及维护都变得方便了。像这种计算机并不能直接识别和执行的符号语言的指令，需要用一种称为汇编程序的软件，把符号语言的指令转换为机器指令。机器指令转换的过程称为“代真”或“汇编”，因此，汇编语言又称为符号语言。

不同型号计算机的机器语言和汇编语言是互不通用的。用甲机器的机器语言编写的程序在乙机器上不能使用。机器语言和汇编语言是完全依赖具体机器特性的，其可读性和可移植性都很差，是面向机器的语言。由于它“贴近”计算机，或者说离计算机“很近”，称为计算机低级语言。

3. 高级语言

高级语言的出现，是为了克服低级语言的缺点，它很接近于人们习惯使用的自然语言和数学语言。程序中用到的语句和指令是用英文单词表示的，程序中所用的运算符、运算表达式和人们日常所用的数学式子差不多，很容易理解。这种语言功能很强，且不依赖具体机器，用它写出的程序对任何型号的计算机都适用（或只需做很少的修改），它与具体机器距离较远，故称为计算机高级语言。

当然，计算机也是不能直接识别高级语言程序的，要进行“翻译”。用一种称为编译程序的软件把用高级语言编写的程序（称为源程序，即 source program）转换为机器指令的程序（称为目标程序，即 object program），然后让计算机执行机器指令程序，最后得到结果。

高级语言经历了不同的发展阶段。

1954 年，第一个完全脱离机器硬件的高级语言 FORTRAN 问世；1969 年，提出了结构化程序设计方法；1970 年，第一个结构化程序设计语言 Pascal 出现，标志着结构化程序设计时期的开始。20 世纪 80 年代初开始，在软件设计思想上又产生了一次革命，其成果就是面向对象的程序设计。在此之前的高级语言几乎都是面向过程的，程序的执行像流水线似的，在一个模块被执行完成前，人们不能干别的事，也无法动态地改变程序的执行方向。像 C++、Java、Visual Basic、Delphi 等就是典型的面向对象程序设计语言的代表。

（1）非结构化的语言。初期的语言属于非结构化的语言，编程风格比较随意，只要符合语法规则即可，没有严格的规范要求，程序中的流程可以随意跳转。人们往往为了追求程序执行的效率而采用许多“小技巧”，使程序变得难以阅读和维护。早期的 BASIC、FORTRAN 和 ALGOL 等都属于非结构化的语言。

（2）结构化语言。为了解决以上问题，提出了“结构化程序设计方法”，规定程序必须由具有良好特性的基本结构（顺序结构、分支结构、循环结构）构成，程序中的流程不允许随意跳转，程序总是由上而下地顺序执行各个基本结构。这种程序结构清晰，易于编写、阅读和维护。QBASIC、FORTRAN 77 和 C 语言等属于结构化的语言，这些语言的特点是支持结构化程序设计方法。

以上两种语言都是基于过程的语言，在编写程序时需要具体指定每一个过程的细节。在编写规模较小的程序时，还能得心应手，但在编写规模较大的程序时，就显得捉襟见肘、力不从心了。

（3）面向对象的语言。在实践的发展中，人们又提出了面向对象的程序设计方法。程序面对的不是过程的细节，而是一个个对象，对象是由数据以及对数据进行的操作组成的。20 世纪 60 年代以来，在处理规模较大的问题时，开始使用面向对象的语言。C++、C#、Visual Basic 和 Java 等语言是支持面向对象程序设计方法的语言。

（4）面向应用的语言。面向应用的语言也是高级语言的下一个发展目标，也就是说，只需要告诉程序你要干什么，程序就能自动生成算法，自动进行处理，这也是非过程化的程序语言。

进行程序设计时，必须要用到计算机语言，人们根据任务的需要选择合适的语言，编写出程序，然后运行程序得到结果。

1.3　程 序 设 计

1.3.1　程序设计的定义

对复杂程度较高的问题，想直接编写程序是不现实的，必须从问题描述入手，经过对解题算法的分析、设计直至程序的编写、调试和运行等一系列过程，最终得到能够解决问题的计算机应用程序，此过程称为程序设计，也称计算机编程。在程序设计中，程序员需要了解各种开发语言和开发平台的优缺点，并且懂得如何根据问题的大小和难易程度选择最合适的开发工具。由于现在的实用程序越来越大，在大多数情况下，单用一种开发语言和开发平台已不能解决问题，需要多种开发工具的通力协作。

1.3.2　程序设计的基本步骤

简单程序设计的具体步骤如下。

（1）问题分析与方案确定。程序将以数据处理的方式解决客观世界中的问题，因此在程序设计之初，首先应该将实际问题用数学语言描述出来，形成一个抽象的、具有一般性的数学问题，从而给出问题的抽象数学模型，然后确定解决该模型所代表数学问题的算法。数学模型精确地阐述了模型本身所涉及的各种概念、已知条件、所求结果，以及已知条件与所求结果之间的联系等各方面的信息。数学模型是进一步确定解决所代表数学问题的算法的基础。数学模型和算法的结合将给出问题的解决方案。

（2）算法描述。具体的解决方案确定后，需要对所采用的算法进行描述，算法的初步描述可以采用自然语言方式，然后逐步将其转化为程序流程图或其他直观方式。这些描述方式比较简单明确，能够比较明显地展示程序设计思想，是进行程序调试的重要参考。

（3）编写程序。使用计算机系统提供的某种程序设计语言，根据上述算法进行描述，将已设计好的算法表达出来。使得非形式化的算法转变为形式化的由程序设计语言表达的算法，这个过程称为程序编制（编码）。需要反复调试才能得到可以运行且结果“正确”的程序。

（4）程序测试。程序编写完成后必须经过科学的、严格的运行和测试，才能最大限度地保证程序的正确性。同时，通过测试可以对程序的性能作出评估。

（5）编写文档。程序文档一般包括程序名称、功能、运行环境、程序的装入和启动、需要输入的数据，以及使用的注意事项等。软件是计算机程序和程序文档的总称。

通过上述过程可以看到，程序设计过程是算法、数据结构和程序设计语言相统一的过程。问题和算法的最初描述，无论是在描述形式上还是在内容上，离最终以计算机语言描述的算法（程序）还有相当大的差距。对于一个规模不是很大的问题，程序设计的核心是算法设计和数据结构设计，只要成功地构造出解决问题的高效算法和数据结构，则完成剩下的任务就不存在太大困难。

1.3.3 C 语言程序设计的开发过程

1. 设计算法

针对具体的问题，分析、建立解决问题的物理或数学模型，并将解决方法采用某种方式描述出来，为进行 C 语言实际编程打下良好基础。

2. 编辑

使用一个文本编辑器编辑 C 语言源程序，并将其保存为文件扩展名为 .c 的文件。

3. 编译

编译就是将编辑好的 C 语言源程序翻译成二进制目标代码的过程。编译过程由 C 语言编译系统自动完成。编译时首先检查源程序的每一条语句的语法错误，当发现错误时，就在屏幕上显示错误的位置和错误类型信息。此时，要再次调用编辑器进行查错并修改，然后进行编译，直到排除所有的语言和语义错误。正确的源程序文件经过编译后，在磁盘上生成同名的目标文件（.obj）。

4. 链接

将目标文件和库函数等链接在一起形成一个扩展名为 .exe 的可执行文件。如果函数名称写错或漏写包含库函数的头文件，则可能提示错误信息，从而得到程序错误数据。

C 语言程序的运行

5. 运行

可以脱离 C 语言编译系统，直接在操作系统下运行。若运行程序后达到预期的目的，则 C 语言程序的开发工作到此完成，否则要进一步修改源程序，重复编辑—编译—链接—

运行的过程，直到取得正确结果为止。C 语言程序上机过程如图 1-1 所示。

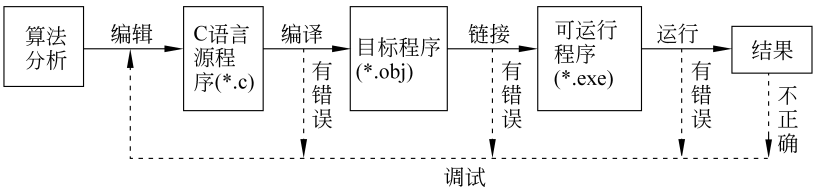

图 1-1　C 语言程序上机过程

1.4　C 语言的发展及其特性

1.4.1　C 语言的发展过程

1967 年，英国剑桥大学的马丁·理查兹（Martin Richards）推出了没有类型的 BCPL（basic combined programming language），被称为 C 语言的祖先。1970 年，美国 AT&T 贝尔实验室的肯·汤普森（Ken Thompson）以 BCPL 为基础，设计出了很简单且很接近硬件的 B 语言（取 BCPL 的第一个字母）。1972—1973 年，美国贝尔实验室的 D. M. 里奇（D. M. Ritchie）在 B 语言的基础上设计出了 C 语言。最初的 C 语言只是为给描述和实现 UNIX 操作系统提供一种工作语言而设计的。1973 年，肯·汤普森和 D. M. 里奇合作把 UNIX 90% 以上的源代码用 C 语言改写，即 UNIX 第 5 版（原来的 UNIX 操作系统是 1969 年由美国贝尔实验室的肯·汤普森和 D. M. 里奇成功开发的，是用汇编语言编写的）。随着 UNIX 的日益广泛应用，C 语言也迅速得到推广。1978 年以后，C 语言先后移植到大、中、小和微型计算机上，C 语言便很快风靡全世界，成为世界上应用最广泛的程序设计高级语言之一。

1989 年，ANSI 公布了一个完整的 C 语言标准 ANSI X3. 159—1989（常称 ANSI C 或 C89）。1990 年，国际标准化组织（International Standard Organization，ISO）接受 C89 作为国际标准 ISO/IEC 9899：1990，它和 ANSI 的 C89 基本上是相同的。

1995 年，ISO 对 C90 做了一些修订，即 1995 基准增补 1（ISO/IEC 9899/AMDI:1995）。1999 年，ISO 又对 C 语言标准进行修订，在基本保留原来 C 语言特征的基础上，针对应用的需要增加了一些功能，尤其是 C++ 中的一些功能，命名为 ISO/IEC 9899:1999。2001 年和 2004 年先后进行了两次技术修正，即 2001 年的 TCl 和 2004 年的 TC2。ISO/IEC 9899:1999 及其技术修正被称为 C99，C99 是 C89（及 1995 基准增补 1）的扩充。

2011 年 12 月 8 日，国际标准化组织和国际电工委员会再次发布了 C 语言的新标准 ISO/IEC 9899:2011，简称 C11 标准，这是 C 语言的第三个官方标准。C11 标准提高了对 C++ 的兼容性，并增加了一些新的特性。因为 C11 和 C99 区别并不大，而 C99 的影响更深远，所以业内交流通常只会提 C99。

2018 年 6 月，国际标准化组织正式发布了 C 语言的新标准 ISO/IEC 9899:2018，即 C18。C18 没有引入新的语言特性，只对 C11 进行了补充和修正。

2022 年 9 月 3 日，ISO 于 Open Standards（计算机标准开放组织）网站上发布了新的 C 语言标准定稿，称为 ISO/IEC 9899:2023，简称 C23。

目前，在实际的使用过程当中，受限于 C 语言编译器的支持度及使用习惯，业界依然以 C89 和 C99 为主。

1.4.2 C 语言的特性

C 语言经历了不断的发展和完善，成为当今计算机界公认的一种优秀程序设计语言，有着其他语言不可比拟的特点。

（1）生成目标代码质量高，程序执行效率高，适合开发系统软件。C 语言最初是为了编写 UNIX 操作系统，它既具有高级语言的易学、易用、可移植性强的特点，又具有低级语言执行效率高、可对硬件进行操作等优点。既可以开发应用软件，也可以开发系统软件。许多以前只能用汇编语言处理的问题，后来可以改用 C 语言来处理。目前，C 语言的主要用途之一是编写嵌入式系统程序。由于具有上述优点，C 语言应用面十分广泛，许多应用软件也用 C 语言编写。

（2）结构化的程序设计语言。C 语言是一种结构化语言，它提供了编写结构化程序的基本控制语句，并以具有独立功能的函数作为模块化程序设计的基本单位。有利于以模块化方式进行设计、编码、调试和维护。

（3）丰富的数据类型和表达式。C 语言不仅本身提供了大量的数据类型，如整型、实型、字符型等，还可以由用户根据自己设计需要定义特殊的数据类型。同时还允许大多数数据类型之间进行转换。运算符和表达式的类型丰富多样，包括赋值、条件、计算、逗号等 30 余种。这使得 C 语言具有极强的表现能力和处理能力，几乎可以完成所有的事务描述。

（4）可移植性好。由于 C 语言程序本身并不依赖机器硬件，且 UNIX、Windows 等主要的操作系统都支持 C 语言编译器，因此 C 程序可以被广泛地移植到各种类型的计算机上。

（5）语句简洁、功能强。C 语言中只提供了 30 余个关键字、9 种控制语句，编程风格灵活，语法限制少，它的表达式可以组成语句以及将其他高级语言中不必要的表示去掉。对于相同的操作，C 语言较之简单、灵活，易于阅读和维护。

（6）具有预处理功能和丰富的库函数。预处理的使用为程序的修改、阅读、移植和调试提供了方便。同样大量的库函数可供程序设计人员直接调用，省去了重复编写这些函数的时间和精力，大幅提高了程序设计的效率并保证了程序设计的质量。

（7）C 语言允许直接访问物理地址。C 语言能进行位（bit）操作，能实现汇编语言的大部分功能，可以直接对硬件进行操作。因此 C 语言既具有高级语言的功能，又具有低级语言的许多功能，可用来编写系统软件。C 语言的这种双重性，使它既是成功的系统描述语言，又是通用的程序设计语言。

1.5 C 语言程序的组成结构

下面通过一个简单的例子说明一般 C 语言程序的组成结构。

1.5.1　C 语言程序举例


第一个
C 语言程序

【例 1-1】 要求在屏幕上输出以下一行信息。

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

分析：在主函数中用 printf( ) 函数原样输出以上文字。

```
#include <stdio.h>                          // 编译预处理指令
int main()                                  // 定义主函数
{                                           // 函数开始的标志
  printf("This is a C program. \n");        // 输出所指定的一行信息
  return 0;                                 // 函数执行完毕时返回函数值 0
}                                           // 函数结束的标志
```

程序运行结果为

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

程序分析：

（1）程序第 2 行中的 main( ) 是“主函数”，int 表示此函数的类型是整型类型，它规定在执行主函数后会得到一个整型值（即函数值）。

（2）程序第 4 行中 printf( ) 函数的作用是将双撇号内的字符串 "This is a C program." 输出到屏幕上，\n 是光标换行。

（3）程序第 5 行中的 return 语句将 0 作为函数的返回值。

说明：

（1）每一个 C 语言程序都必须有一个 main( ) 函数，函数体由大括号 {} 括起来。

（2）stdio.h 是系统提供的一个文件名，文件扩展名 .h 的意思是头文件（header file），因为这些文件都是放在程序各文件模块的开头的。输入 / 输出函数的相关信息已事先放在 stdio.h 文件中。现在，用 #include 指令把这些信息调入以供使用。对于编译预处理指令 #include，在此读者可先不必深究，只要记住：在程序中如要用到标准函数库中的输入 / 输出函数，应该在本文件模块的开头写上下面一行。

```
#include <stdio.h>
```

（3）“//”则表示从此处开始到本行结束是“注释”，用来对程序有关部分进行必要的说明。在编译时注释部分不产生目标代码，注释对运行不起作用。注释只用来进行说明，计算机是不会执行的。

C 语言允许用两种注释方式：以“//”开始的单行注释，注释的范围从“//”开始，以换行符结束，即这种注释不能跨行；以“/*”开始、以“*/”结束的块式注释，注释可以包含多行内容。

（4）C99 建议把 main( ) 函数指定为 int 型（整型），它要求函数返回一个整数值。在 main( ) 函数中，在执行的最后设置一个“return 0;”语句。当主函数正常结束时，得到的函数值为 0；当执行 main( ) 函数过程中出现异常或错误时，函数值为一个非 0 的整数。


求两个整数中的较大者

【例 1-2】 求两个整数中的较大者。

分析：用一个函数来实现求两个整数中的较大数，然后在主函数中调用该函数并输出结果。

```
#include<stdio.h>          //编译预处理指令
int main()                 //定义主函数
{                          //函数开始
  int max(int x,int y);    //对被调用函数max()的声明
  int a,b,c;               //定义变量a、b、c
  scanf("%d,%d",&a,&b);    //输入变量a和b的值
  c=max(a,b);              //调用max()函数，将得到的值赋给c
  printf("max=%d\n",c);    //输出c的值
  return 0;                //返回函数值为0
}                          //主函数体结束
int max(int x,int y)       //定义max()函数，函数值为整型，形式参数x和y为整型
{
  int z;                   //定义变量z
  if (x>y)z=x;             //若x>y成立，将x的值赋给变量z
  else z=y;                //否则（即x>y不成立），将y的值赋给变量z
  return(z);               //将z的值作为max()函数值，返回到调用max()函数的位置
}
```

程序运行结果为

```
8,5↙
max=8
```

程序分析：

（1）本程序包括 2 个函数，即主函数 main( ) 和被调用的函数 max( )。

（2）函数的功能。max( ) 函数的作用是将 x 和 y 中较大者的值赋给变量 z，并用 return 语句把 z 的值作为 max( ) 函数值，返回到调用 max( ) 函数的位置。

（3）对被调用函数的声明。为什么要作 max( ) 函数声明呢？因为在主函数中要调用 max( ) 函数时，max( ) 函数的定义却在 main( ) 函数之后，对程序的编译是自上而下进行的，如果没有声明，在对程序“c=max(a,b);”进行编译时，编译系统无法知道 max 是什么，因而无法把它作为函数进行调用处理。

（4）变量的输入函数。程序中 scanf 是输入函数的名字，该 scanf( ) 函数的作用是输入变量 a 和 b 的值。scanf 后面括号中包括两部分内容：一是双撇号中的内容，它指定输入的数据按什么格式输入。“%d”的含义是以十进制整数形式；二是输入的数据准备放到哪里，即赋给变量 a 和 b。在 C 语言中“&”是地址符，&a 的含义是“变量 a 的地址”。执行 scanf( ) 函数，从键盘读两个整数，送到变量 a 和 b 的地址，然后把这两个整数分别赋给变量 a 和 b。

小提示：在使用 Visual Studio 2022 调用 scanf( ) 函数时，会提示 scanf( ) 函数不安全，此时需要关闭安全检查。要关闭安全检查，需要在源代码文件顶部添加 #define_CRT_SECURE_NO_WARNINGS。

（5）函数的调用。程序用 max(a,b) 调用 max( ) 函数，在调用时将 a 和 b 作为 max( ) 函数的参数（称为实际参数）的值传送给 max( ) 函数中的参数 x 和 y（称为形式参数），

然后执行 max( ) 函数的函数体，使 max( ) 函数中的变量 z 得到一个值（即 x 和 y 中较大者的值），return(z) 的作用是把 z 的值作为 max( ) 函数值返回到程序第 7 行的右侧 [主函数调用 max( ) 函数的位置]，取代 max(a,b)，然后把这个值赋给变量 c。

（6）结果输出。在执行 printf( ) 函数时，对双撇号括起来的 max= %d\n 是这样处理的：将“max=”原样输出，%d 由变量 c 的值取代之，\n 执行换行。

本例用到了函数调用、实际参数和形式参数等概念，只作了简单的解释。读者如对此不太理解，可以先不予深究，在以后学到有关内容时自然迎刃而解。

1.5.2　C 语言程序的结构特点

C 语言程序的结构特点

通过以上两个程序例子，可以看到一个 C 语言程序的结构有以下特点。

（1）一个 C 语言程序是由一个或多个函数组成的，其中必须包含一个 main( ) 函数 [且只能有一个 main( ) 函数]。并且，无论 main( ) 函数在程序中处于什么位置，程序的执行总是从 main( ) 函数开始。

一个函数由函数首部和函数体两部分组成。函数首部即函数的第 1 行，包括函数名、函数类型、函数参数名、参数类型。函数体，即函数首部下面的大括号内的部分。如果在一个函数中包括有多层大括号，则最外层的一对大括号是函数体的范围。函数体一般包括以下两部分：声明部分，定义在本函数中所用到的变量，对本函数所调用函数进行声明。执行部分，由若干个语句组成，指定在函数中所进行的操作。

（2）分号是 C 语言语句的必要组成部分，在每个数据声明和语句的最后必须有一个分号。

（3）程序中的 #include<stdio.h> 通常称为预处理命令，预处理命令必须用“#”开头，行尾不能加“;”，它不是 C 语言程序的语句。

（4）程序应当包含注释，分为两种，即行注释“//”和块注释“/* 注释内容 */”。一个好的、有使用价值的源程序都应当加上必要的注释，以增加程序的可读性。

（5）C 语言本身不提供输入和输出语句。输入和输出的操作是由库函数 scanf( ) 和 printf( ) 等函数来完成的。

（6）一个程序由一个或多个源程序文件组成。一个规模较小的程序，往往只包括一个源程序文件。在一个源程序文件中可以包括 3 个部分：①预处理指令，如 # include <stdio. h>（还有一些其他预处理指令，如 #define 等）；②全局声明，即在函数之外进行的数据声明；③函数定义。

1.5.3　C 语言字符集

任何一门西方语言都有其固定的字母，整个语言就是由这些字母（要素）构成的。C 语言表达一个程序同样需要定义一些基本的要素，这些要素称为字符集（character-set）。

C 语言基本字符集包括英文字母、阿拉伯数字以及一些特殊符号。很多特殊符号具有多重含义，在后续内容中将会逐步介绍，这里向大家介绍常规分类如下。

（1）英文字符（52 个）：a~z 和 A~Z。

（2）阿拉伯数字（10 个）：0、1、2、3、4、5、6、7、8、9。
（3）其他特殊符号如下。
① 下画线（1 个）：_。
② 括号（6 个）：(、)、[、]、{、}。
③ 四则运算符号（7 个）：+、-、*、/、%、++、--。
④ 关系运算符号（6 个）：<、>、=、!=、>=、<=。
⑤ 逻辑运算符号（7 个）：!、||、&&、^、~、|、&。
⑥ 移位符号（2 个）：<<、>>。
⑦ 转义符号（1 个）：\。
⑧ 条件表达符号（2 个）：?、:。
⑨ 界定符号（4 个）："、'、,、;。
⑩ 成员运算符号（2 个）：.、→。

1.5.4 C 语言标识符

1. 标识符的定义

用来对变量、符号常量名、函数、数组、类型等命名的有效字符序列统称为标识符。也可以说，标识符就是一个对象的名字。标识符命名规则是：标识符可以由字母、数字和下画线组成，并且第一个字符必须为字母或下画线。C 语言规定，在命名标识符时，必须遵循以上规则。area、IP、_ini、a_array、s243 是合法的标识符；456P、cade -、a&b 是非法的标识符。

说明：

（1）标识符必须以字母或下画线开头。
（2）标识符除首字符外，其余字符可以是字母、数字、下画线。
（3）大小写被认为是不同的字符。
（4）标识符不能与关键字（key words）重名。

对于标识符的长度，不同的编译程序有不同的规定，Visual Studio 2022 开发环境中没有限制。读者在命名标识符时，应注意自己所用系统对标识符长度的规定。对于初学者，建议在程序中对标识符的命名不要太长，要尽量做到见名知义，以提高程序的可读性。

2. 标识符分类

C 语言标识符可以分为以下 3 类。

（1）关键字。在 C 语言中已经预先规定了一批标识符，它们在程序中都代表着固定的含义，不能另作他用，这一类标识符叫作关键字。在程序中，用户不能再用它们来定义新的函数、变量等，或者说，用户所定义的标识符不能和关键字重名。根据 ANSI（American National Standard Institute）标准，C 语言共有 32 个关键字：

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while