第1章　C语言概述

C语言是一种计算机程序设计语言，既具有高级语言的特点，又具有汇编语言的特点。C语言由AT&T贝尔实验室的D. M. Ritchie于1972年推出。1978年后，C语言已先后被移植到大、中、小及微型计算机上。它可以作为工作系统设计语言，编写系统应用程序；也可以作为应用程序设计语言，编写不依赖计算机硬件的应用程序。

熟练的编程技能是在不断的知识与经验积累的基础上发展起来的。初学编程的人一开始由于缺乏足够的语言知识和编程经验，对于很简单的问题往往也会感到无所适从，不知如何下手编写程序。本书建议读者从一开始学习C语言起就要试着编写程序，可以先模仿教科书中的程序，试着改写它并逐步体会语法逻辑、循序渐进，直到会独立地编写程序，解决比较复杂的问题。

为了使读者能逐步从简单的模仿中体会程序设计的基本思想和方法，而不拘泥于具体的语法细节，本章将简要介绍程序设计语言的功能、语法要素及C语言的特点。

1.1　一个C语言程序

为了使读者对C语言有一个直观的认识和初步了解，首先看下面这个程序。

例1-1　求两数中的较大值：由用户输入两个整数，程序执行后输出其中较大的数。

源程序：

```
#include <stdio.h>                          /*编译预处理①*/
int main(void)                              /*主函数②*/
{
    int x,y,max;                            /*变量说明③*/
    printf("请用户输入两个数:\n");          /*显示提示信息④*/
    scanf("%d%d",&x,&y);                    /*输入x,y值⑤*/
    if(x>y)
        max = x;
    else
        max = y;                            /*根据条件确定max的值⑥*/
    printf("maxnum = %d\n",max);            /*输出⑦*/
```

```
    return 0;
}
```

程序运行结果：

```
输入：4  6
输出：maxnum=6
```

本程序的功能是比较两个数，把较大的数赋给 max，然后将 max 的值输出到屏幕上。在程序的每行后用“/*”和“*/”括起来的内容为注释部分，程序不执行注释部分。以下是程序每个语句的详细说明：

① include 称为文件包含命令，表明程序要用到一些被包含文件中的功能，扩展名为.h 的文件称为头文件。

② main()是主函数的函数名，表示这是一个主函数。每一个 C 语言源程序都必须有且只能有一个主函数即 main()函数。

③ 定义三个整型变量存储数据，以备后面程序使用。

④ printf()函数的功能是把要输出的内容送到显示器显示。printf()函数是一个由系统定义的标准函数，可在程序中直接调用。

⑤ scanf()函数的功能是从键盘获得相应变量的值，它也是由系统定义的标准函数，可在程序中直接调用。

⑥ 比较 x、y 的大小，根据结果确定 max 存储 x 和 y 两者中的较大者。

⑦ 再次使用 printf()函数，其功能是按指定的格式输出 max 的值。

上例中程序的执行过程是：首先在屏幕上显示输入提示“请用户输入两个数”，回车(即按 Enter 键，下同)后由 scanf()函数语句接收这两个数送入变量 x、y 中，然后比较 x、y 的大小，把较大的值赋给变量 max，最后在屏幕上输出 max 的值。

由上面的程序可以看出：C 语言是一种结构化语言，它层次清晰，便于按模块化方式组织程序，接近于人类的自然语言，可读性强。读者即使没有学习过该语言，对于有些语句也可以根据英文的含义来理解。

1.2 计算机程序设计与程序设计语言

计算机程序(program)是指为实现特定目标或解决特定问题而用计算机语言编写的命令序列的集合，简单来讲，就是用某种程序设计语言告诉计算机如何完成一个具体的任务。而程序设计语言是人与计算机之间交换信息的工具，一方面，程序设计语言提供了表达数据和处理数据的功能；另一方面，编程人员必须按照规定的语法进行编程。

1.2.1 程序与指令

指令是指挥计算机工作的指示和命令，程序就是一系列按一定顺序排列的指令。指令是计算机处理数据最基本的单元，一条指令只能完成一个最基本的功能，如加法运算、数据复制或者大小判别。虽然计算机指令所能完成的功能很基本，计算机系统所包含的指令的数量也有限，但一系列指令组合却能完成很复杂的功能。目前我们使用的计算机操作系统、

各种办公软件、影音播放器和各种计算机游戏等都是由这些基本指令构成的。

假设一个计算机系统有以下 7 条基本指令，每条指令第一部分是指令名，第二部分是需要处理的数据。

指令 1：Input X，将当前输入数据存储到内存的 X 单元。

指令 2：Output X，将内存 X 单元的数据输出。

指令 3：Add X Y Z，将内存 X 单元的数据与 Y 单元的数据相加并将结果存储到 Z 单元。

指令 4：Sub X Y Z，将内存 X 单元的数据与 Y 单元的数据相减并将结果存储到 Z 单元。

指令 5：BranchEq X Y P，比较 X 与 Y，若相等则程序跳转到 P 处执行，否则继续执行下一条指令。

指令 6：Jump P，程序跳转到 P 处执行。

指令 7：Set X Y，将内存 Y 单元的值设为 X。

一般来说，计算机系统中是不含乘法操作这个基本指令的，因此进行乘法操作必须用其他指令组合实现。下面通过例题说明，如何用简单的指令进行不同组合来实现复杂的功能。

例 1-2 输入两个数 A 和 B，求 A * B 的结果。

由基本指令构成的程序如下，每条指令均有一个编号，右侧用“/ * ”和“ * /”括起来的内容为说明文字：

```
1. Input A;              /* 输入第 1 个数据到存储单元 A 中 */
2. Input B;              /* 输入第 2 个数据到存储单元 B 中 */
3. Set 0 X;              /* 将 X 设为 0，此处 X 用以统计 A 累加的次数 */
4. Set 0 Z;              /* 将 Z 设为 0，此处 Z 用以存放 A * B 的结果 */
5. BranchEq X B 9;       /* 判别 X 与 B 是否相等；若相等说明 A 已累加了 B 次，*/
                         /* 程序跳转到第 9 条指令，输出结果 */
6. Add Z A Z;            /* Z = Z + A */
7. Add 1 X X;            /* X = X + 1 */
8. Jump 5;               /* 程序跳转到第 5 条指令，继续循环执行第 6 条、第 7 条指令 */
9. Output Z;             /* 输出 Z 的值，该值等于 A * B */
```

把 A 累加 B 次就可以得到 A * B 的结果，因此以上程序反复执行加法运算 Z=Z+A，执行的次数则由另一个乘数 B 决定。头两条指令首先用 A 和 B 来存储这两个要相乘的数。第 4 条指令用 Z 来存放乘法运算的结果，初始值为 0。随后反复执行第 6 条指令，不断将 A 累加到 Z 上。累加的次数通过一个中间变量 X 控制。第 3 条指令设 X 初始值为 0，随后每累加一次 A 程序就用第 7 条指令将 X 加 1，然后通过第 8 条指令跳转回到第 5 条指令，判断 X 是不是被累加了 B 次(实际上也就是 A 累加到 Z 上达到了 B 次)。若已达到就跳转到第 9 条指令，此时 Z 的值就是 A * B 的结果了，最后输出 Z 的值。

为什么程序一定要由计算机中的指令组成呢？一方面，通过设计计算机可直接实现的指令集使得程序在计算机中的执行变得简单，计算机硬件系统只要实现了指令就能方便地实现相应的程序；另一方面，需要计算机实现的任务成千上万，如果每一个任务都相对独立，与其他程序之间没有公共的内容，编程工作将十分困难。这就是计算机科学中很重要的一个概念——“重用”的体现。

如果程序设计直接用0、1序列的计算机指令来写，那将是难以忍受的。因此，人们设计了程序设计语言，用这种语言来描述程序，同时应用一种软件（如编译系统）将程序设计语言描述的程序转换成计算机能直接执行的指令序列。

1.2.2 程序设计语言

程序设计的任务就是用计算机理解的语言即程序设计语言编写程序，然后交给它去执行。程序设计语言分为低级语言和高级语言。低级语言包括机器语言和汇编语言，它们的特点是与特定的机器有关，效率高但使用复杂、烦琐、费时、易出差错。高级语言的表示方法比低级语言更接近人类自然语言的表达习惯，其特点是在一定程度上与具体机器无关，易学、易用、易维护。

计算机所能执行的指令的集合称为计算机的指令系统，它将计算机可以完成的动作编制成一个指令表，每种动作赋予一个二进制代码，并为机器的每种动作设计一种通用的格式：由指令码和内存地址组成。一条指令就是一个固定长度的由指令码和地址码组成的二进制位串，这就是计算机唯一可以读懂的语言，一般称作机器语言。

“机器语言”太“低级”了，机器容易懂，但对人来说却太不方便。于是，一种“汇编系统”程序问世了，这种程序的功能是把一种“汇编语言”翻译为“机器语言”。汇编语言是用人比较习惯的符号来代替指令编码，例如，用“ADD”代替“001”表示加法操作，用“Move”代替“010”表示数据移动。用符号代替二进制地址表示参加操作的数据，这样大大减少了编程工作中的困难。

汇编语言和机器语言都属于低级语言，这是因为其都是以面向机器的指令序列形式为主，与人的习惯语言方式距离较远，它们的共同缺点是：

（1）依赖于机器，可移植性差；

（2）代码冗长，不易于编写大规模程序；

（3）可读性差，可维护性差。

总之，即使是汇编语言，也没有解决计算机编程难的基本问题。低级语言必然被高级语言取代，高级语言极大地推动了今天计算机应用的网络化、多媒体化和智能化等发展。高级语言是目前绝大多数编程者的选择。和汇编语言相比，高级语言不但将许多相关的机器指令合成为单条指令，而且去掉了与具体操作有关但与完成工作无关的细节，如使用堆栈、寄存器等，这样就大大简化了程序中的指令。由于省略了很多细节，因此编程者也不需要具备太多的专业知识。

高级语言主要是相对于汇编语言而言的，它并不是特指某一种具体的语言，而是包括很多编程语言，如目前流行的BASIC、C、C++、Pascal、FORTRAN、Java等，这些语言的语法、命令格式都各不相同。

1.3 C语言的发展与应用现状

由于C语言的强大功能和各方面的优点逐渐为人们认识，到了20世纪80年代，C语言开始移植到其他操作系统，并很快在各类大、中、小、微型计算机上得到了广泛的使用，成为当代最优秀的程序设计语言之一。

C 语言由早期的编程语言 BCPL(Basic Combined Programming Language)发展演变而来。1970 年,AT&T 贝尔实验室的 Ken Thompson 根据 BCPL 语言设计出较先进的 B 语言,1972 年 D. M. Ritchie 在 B 语言的基础上设计出了 C 语言,同年首次在 DEC PDP-11 计算机上使用。

发明 C 语言的目的是描述 UNIX 操作系统。1973 年,Ken Thompson 和 D. M. Ritchie 用 C 语言重写了 UNIX,即 UNIX 系统,这使得 UNIX 设计得非常轻巧,改变了操作系统的历史。从此,C 语言的应用越来越广泛,成了各种计算机共同使用的程序设计语言。

随着微型计算机的日益普及,出现了许多 C 语言版本。由于没有统一的标准,这些 C 语言之间出现了不一致。为了改变这种情况,美国国家标准学会(ANSI)于 1983 年制定了 C 语言标准,即 ANSI C,并于 1987 年开始实施。Turbo C 是美国 Borland 公司的产品,1987 年该公司首次推出 Turbo C 1.0 产品,其中使用了全新的集成开发环境,即使用了一系列下拉式菜单,将文本编辑、程序编译、连接以及程序运行一体化,大大方便了程序的开发。1991 年为了适应 Microsoft 公司的 Windows 3.0 版本,Borland 公司又将 Turbo C++做了更新,即 Turbo C 的新一代产品 Borland C++问世。

1990 年,国际标准化组织(International Organization for Standardization,ISO)接受了 ANSI C89 为 ISO C 的标准(ISO/IEC 9899:1990)。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,针对需要增加了一些功能,尤其是对 C++中的一些功能,修订后的标准命名为 ISO/IEC 9899:1999。目前 C 语言的最新官方标准是 ISO/IEC 9899:2024。目前流行的 C 语言编译系统大多是以 ANSI C 为基础进行开发的,但不同版本的 C 编译系统所实现的语言功能和语法规则略有差别。

C 语言的应用范围非常广泛。对操作系统和系统实用程序以及在需要对硬件进行操作的场合,用 C 语言明显优于其他解释型高级语言,有一些大型应用软件也是用 C 语言编写的。C 语言具有绘图能力强、可移植性高的特点,并且具备很强的数据处理能力,因此适于编写系统软件、三维/二维图形和动画。它也是适合做数值计算的高级语言。

1.4 C 语言的特点

1.4.1 C 语言的优势

1. 简洁紧凑、灵活方便

C 语言一共只有 32 个关键字、9 种控制语句,程序书写形式自由,且严格区分大小写。它兼具高级语言的基本结构和语句,以及低级语言的实用性。

2. 运算符丰富

C 语言的运算符种类很丰富,共有 34 种运算符。C 语言把括号、赋值、强制类型转换等都作为运算符处理,因此运算类型极其丰富,表达式类型多样化。灵活使用各种运算符可以实现在其他高级语言中难以实现的运算。

3. 数据类型丰富

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,可用于实现各种复杂的数据结构的运算;引入了指针概念,使程序执行效率更高。另

外，C 语言具有强大的图形功能，支持多种显示器和驱动器，且计算功能、逻辑判断功能强大。

4. C 是结构式语言

结构式语言的显著特点是代码及数据的分隔化，即程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰，便于使用、维护及调试。C 语言是以函数形式提供给用户的，这些函数可方便地调用，并具有多种循环、条件语句控制程序流向，从而使程序完全结构化。

5. 语法限制不太严格，程序设计自由度大

虽然 C 语言也是强类型语言，但它的语法比较灵活，允许程序编写者有较大的自由度。

6. 允许直接访问物理地址，对硬件进行操作

由于 C 语言允许直接访问物理地址，可以直接对硬件进行操作，因此它既具有高级语言的功能，又具有低级语言的许多功能，能够像汇编语言一样对位、字节和地址进行操作，而这三者是计算机最基本的工作单元，可用来编写系统软件。

7. 生成目标代码质量高，程序执行效率高

C 语言一般只比用汇编程序生成的目标代码效率低 10%～20%。

8. 适用范围广，可移植性好

C 语言一个突出的优点就是适合多种操作系统，如 DOS、UNIX、Windows 系列，也适用于多种机型。

1.4.2 学习 C 语言程序设计的方法

对于将 C 语言作为学习的第一门编程语言的读者来说，最关心的问题应该是如何尽快学会用 C 语言进行程序设计。为此，首先对程序设计语言，特别是 C 语言本身要有所了解，更重要的是通过不断的编程实践，逐步领会和掌握程序设计的基本思想与方法。熟练的编程技能是在不断地进行知识与经验积累的基础上发展起来的。

初学编程的人一开始由于缺乏足够的语言知识和编程经验，对于很简单的问题往往也会感到无所适从，不知如何下手编写程序。建议读者从一开始学习 C 语言起就试着编写程序，可以先模仿教科书中的程序，试着改写它并逐步体会、循序渐进，直到会独立地编写程序解决比较复杂的问题。

编写程序就像用某种自然语言写文章一样，首先语法要通，即要符合语言所规定的语法规则。当然，语法通了并不意味着文章就符合要求了，有可能词不达意、离题万里。因此还需进行程序调试(查错)，即找出程序中的错误(非语法错误)，这是一个非常需要耐心和经验的过程。而语法错误的检查则相对容易得多。在初学 C 语言时，可以与学习英语进行类比，因为 C 语言就是接近于英语的计算机语言，很多语句可以望文生义，但 C 语言的“单词”和“语句”比英语少得多。学习 C 语言时，可以从下列三个层次逐步掌握这个程序设计工具：

(1) 了解程序设计语言，掌握语法、结构和流程控制；

(2) 从模仿、改写到自己编写，逐步提高程序设计能力；

(3) 通过不断的编程实践，逐步领会和掌握程序设计的基本思想与方法。

1.4.3 C 语言的语法特点

程序员利用程序设计语言编写想要的程序以处理相应的问题。在程序中，可能要表达

数据，包括定义相应的用于存储数据的变量，还要用程序设计语言来描述需要的数据处理过程，包括语句间的控制和子程序间的控制。为了使计算机理解程序员在程序中所描述的这些工作，用程序设计语言所写的程序必须符合相应语言的语法(grammar)。

一般我们将用程序设计语言编写的未经编译的程序称为源程序(source code，又称源代码)。从语法的角度看，源程序实际上是一个字符序列。这些字符序列按顺序分别组成一系列“单词”。这些“单词”包括语言事先约定好的保留字(reserved words，如用于描述分支控制的 if、else，用于描述数据类型的 int 等)、常量(constant)、运算符(operator)、分隔符以及程序员自己定义的变量名、函数名等。

在这些“单词”中，除了运算符(如＋、－、＊、/)、普通常量(如－12、12.34、'a')、分隔符[如“;”“(”“)”等]外，其他主要是用来标识(表示)变量、函数、数据类型、语句等的符号，这些标识符号叫标识符(idendifier)。任何程序设计语言对标识符都有一定的定义规范，即只有满足这些规范的字符的组合才能构成该语言所识别的标识符。

1. C 语言的主要“单词”

(1) 标识符。C 语言的标识符由字母、数字以及下画线组成，且第一个字符必须是字母或下画线。如，_name1 是一个合法的标识符，而 left&right 是非法的(“&”符号非法)。在 C 语言中，标识符中的字母大小写是有区别的。最主要的标识符是保留字和用户自定义标识符。

(2) 保留字。也称关键字，是 C 语言规定的、赋予它们以特定含义、有专门用途的标识符。这些保留字也主要与数据类型和语句有关。如 int(整数类型)、float(浮点数类型)、char(字符类型)、typedef(类型定义)以及与语句相关的 if、else、while、for、break 等。

(3) 自定义标识符。包括在程序中定义的变量名、数据类型名、函数名以及符号常量名。一般来说，为了便于程序阅读，经常以有意义的英文单词作为用户自定义标识符(如 n、fact 等)。

(4) 常量。常量是有数据类型的，如整型常量 123、实型常量 12.34、字符常量'a'、字符串常量“hello world!”等。

(5) 运算符。其代表对各种数据类型实际数据对象的运算。如＋(加)、－(减)、＊(乘)、/(除)、%(求余)、＞(大于)、＞＝(大于或等于)、＝＝(恒等于)、＝(赋值)等。绝大多数运算为双目运算(涉及两个运算对象)，也有单目(涉及一个运算对象)和三目(涉及三个运算对象)运算，如 C 语言中的条件运算“?:”就是一个三目运算。

(6) 分隔符，如“;”“[”“]”“(”“)”“＃”等。

2. C 语言的主要语法单位

(1) 表达式。运算符与运算对象(可以是常量、函数、变量等)的有意义组合就形成表达式，如 2＋3＊4、i＋2＜j 等。表达式中可以包含多种数据类型的运算符，不同运算符间有不同的运算优先顺序，如 i＋2＜j 中，“＋”的优先级高于“＜”。

(2) 变量定义。变量也都有数据类型，因此定义变量时要说明相应变量的类型。变量的类型不同，它在内存中所占的空间大小也会有所不同。变量定义的最基本形式是：“类型名 变量名;”，如“int i;”就定义了一个整型变量 i。

(3) 语句。语句是程序最基本的执行单位，程序的功能就是通过对一系列语句的执行来实现的。C 语言中的语句有多种形式，包括表达式语句、分支语句、循环语句和复合语句，

这些内容将在后面的章节中详细讲解。

(4) 函数定义与调用。函数是完成特定任务的独立模块,是C语言唯一的一种子程序形式。通常,函数的作用是接收0个或多个数据(称为函数的参数),并至多返回一个结果(称为函数的返回值)。函数的使用主要涉及函数的定义与调用。函数定义的主要内容是通过编写一系列语句来规定其所完成的功能。完整的函数定义涉及函数头和函数体。其中,函数头包括函数的返回值类型、函数名、参数类型;函数体是一个程序模块,规定了函数所具有的功能。函数调用则通过传递函数的参数并执行函数定义所规定的程序过程,实现相应功能。

(5) 输入输出。C语言没有输入输出语句,它通过标准库函数中的有关函数,如printf()和scanf()函数,实现数据的输入与输出。这种处理方式为C语言在不同硬件平台上的可移植性提供了良好的基础。

1.4.4 C语言的结构特点

一个C语言源程序可以由一个或多个源文件组成。每个源文件可由一个或多个函数组成。一个源程序无论由多少个文件组成,都有一个且只能有一个main()函数,即主函数。源程序中可以有预处理命令(include命令仅为其中的一种),预处理命令通常放在源文件或源程序的最前面。每一个说明、每一条语句都必须以分号结尾。但预处理命令中,函数头和花括号"}"之后不能加分号。标识符、关键字之间必须至少加一个空格以示间隔。若已有明显的间隔符,也可不用加空格来间隔。

当要解决的问题比较复杂时,程序的控制过程也会变得十分复杂。一种比较典型的程序设计方法是:将复杂程序划分为若干个相互独立的模块(module),使完成每个模块的工作变得单纯而明确,在设计一个模块时不受其他模块的牵连。同时,通过现有模块积木式的扩展就可以形成复杂的、更大的程序模块或程序。这种程序设计方法就是结构化的程序设计方法(structured programming)。C语言就是支持这种设计方法的典型语言。在结构化程序设计方法中,一个模块可以是一条语句(statement)、一段程序或一个函数(function)(子程序)等。

一般来说,从程序流程的角度看,模块只有一个入口、一个出口。这种单入、单出的结构为程序的调试提供了良好的条件。多入多出的模块结构将使得程序的调试变得异常困难。按照结构化程序设计的观点,任何程序都可以将模块通过三种基本的控制结构进行组合来实现。这三种基本的控制结构如下:

(1) 顺序控制结构(sequential control structure):一个程序模块执行完后,按自然顺序执行下一个模块。

(2) 分支控制结构(branch control structure):又称选择结构,计算机在执行程序时,一般是按照语句顺序执行的,但在许多情况下需要根据不同的条件来选择所要执行的模块。例如,检测某种条件是否满足,如果条件满足则执行某些指令,否则执行另外一些指令。例如,我们在周末根据天气情况决定去郊游还是在房间里学习,就是一种分支控制结构。

(3) 循环控制结构(loop control structure):经常需要重复地执行某些相同的处理过程,即重复执行某个模块。当然,重复执行这些模块一般是有条件的。也就是说,检测某些条件,如果条件满足就重复执行相应的模块。

1.4.5 C 语言的编译与编程环境

其实，计算机硬件真正能理解的只是计算机的指令。用程序设计语言编写的程序并不能被计算机直接接受，需要一个软件把相应的程序转换成计算机直接能理解的指令序列。对 C 语言等许多高级程序设计语言来说，这种软件就是编译器(compiler)。编译器首先对源程序进行词法分析，然后进行语法与语义分析，最后生成可执行的代码。如果程序中有语法错误，编译器会直接指出程序中的语法错误。当然，由编译器生成可执行的代码后并不意味着程序就没有错误了。对于程序中的逻辑错误，编译器是发现不了的，必须通过程序的调试进一步发现。

编写一个程序需要做很多工作，包括编辑程序(edit)、编译(compile)、调试(debug)、连接(link)等过程。因此，许多程序设计语言都有相应的编程环境。程序员可以直接在该环境中完成上述工作，以提高编程的效率。

对于初学者而言，Microsoft Visual C++是一款比较好的软件，其界面友好，功能强大，调试也很方便。它是微软公司开发的一个 C 语言集成开发环境(IDE)，分为企业版和学生版等。它以拥有“语法高亮”、自动编译功能以及高级调试功能而著称。比如，允许用户进行远程调试、单步执行等；还允许用户在调试期间重新编译被修改的代码，而不必重新启动正在调试的程序。在 Microsoft Visual C++编译及内置系统中，提供了预编译头文件、最小重建及增量连接等特征，这些特征可以明显缩短程序编辑、编译及连接的时间，对于大型软件开发尤其显著。

习题

一、选择题

1. 一个 C 语言程序的执行总是从(　　)。
   A. main()函数开始，直到 main()函数结束
   B. 第一个函数开始，直到最后一个函数结束
   C. 第一条语句开始，直到最后一条语句结束
   D. main()函数开始，直到最后一个函数结束
2. C 程序中的注释可以加在(　　)。
   A. 程序的任何位置
   B. 程序的开始
   C. 程序的末尾
   D. 任何空格、制表符或换行符可以出现的位置
3. 以下叙述正确的是(　　)。
   A. 在 C 程序中，main()函数必须位于程序的最前面
   B. C 程序的每行只能写一条语句
   C. C 语言本身没有输入输出语句
   D. 在对一个 C 程序进行编译的过程中，可发现注释中的拼写错误
4. 以下叙述不正确的是(　　)。

A. 一个C源程序可由一个或多个函数组成

B. 一个C源程序必须包含一个main()函数

C. C程序的基本组成单位是函数

D. 在C程序中,注释说明只能位于一条语句的后面

5. 下面四个选项中,均是不合法的用户标识符的选项是(　　)。

A. A　P_0　do　　B. float　ta0　-A

C. b-a　return　int　　D. -123 INT temp

6. 一个C程序中,main()函数在程序中的位置(　　)。

A. 必须在程序的开头　　B. 必须在最后

C. 只能在中间　　D. 可以任意

7. 在C语言中,下列描述正确的是(　　)。

A. 标识符只能由字母和下画线组成

B. 标识符由任意字符组成

C. 标识符只能由字母、数字和下画线组成

D. 标识符只能由字母组成

8. 下面的标识符中,可以作为合法变量名的是(　　)。

A. m@q　　B. Mo_day　　C. mn/5　　D. char

二、简答题

1. 简述C语言标识符的相关规定,并指出下列标识符中哪些是合法的,哪些是不合法的:

begin_,3dmax,eri@sc,Counter1,total,_debug,Large&Tall, float。

2. 简述C语言在程序设计上的优势。

3. 简述C语言三种基本的程序结构。