第一章 简 介

1.1 概 述

- 概述
- C 语言特点
- 关于本书的几点说明
- 教程概要

C 语言是一种成功的系统描述语言，用 C 语言开发的 UNIX 操作系统就是一个成功的范例；同时，C 语言又是一种通用的程序设计语言，在国际上具有巨大的影响。世界上很多著名的计算机公司，诸如 Microsoft、Borland 等公司都成功地开发了不同版本的 C 语言。很多优秀的应用程序也都是用 C 语言开发的。近年来，我国学习和使用 C 语言的用户与日俱增，许多科技工作者都试图在科研和生产领域中利用 C 语言开发自己的软件系统。然而要真正掌握 C 语言，充分发挥 C 语言的强大功能，远非一件轻而易举的事，因为 C 语言是一种非常灵活的程序设计语言。本书针对 Microsoft C 6.0/7.0，帮助用户深刻地理解 C 语言的精髓，以帮助用户得心应手地解决遇到的各种应用问题。

1.2 C 语言特点

- 语言表达能力强
- 语言简洁，使用灵活
- 具有构造数据类型的能力，并有很强的控制流结构
- 语言生成的代码质量高
- 可移植性好

C 语言的特点是多方面的，人们从不同的角度总结出众多特点。根据笔者的体会并参照流行的说法大致归结为以下几点：

1. 语言表达能力强。C 是面向结构程序设计的语言，通用性好，不局限于某种机器。它可以直接处理字符、数字、地址，可以完成通常由硬件实现的算术、逻辑运算(C 提供了字位运算的能力，可对变址的地址进行有关操作等)。它反映了当前计算机的性能，所以有效到足以取代汇编语言来编写各种系统软件和应用软件。最明显的例证是 UNIX 系统，UNIX 系统主要分三层：核心(操作系统)、与外层的接口 shell 命令解释程序以及外层大量的子系统，包

括各种软件工具、实用程序和应用软件。这些众多的程序中，除了核心内部的 1000 行左右（整个核心的 10%以下）因为效率、对机器硬件的表达等原因需要用汇编语言书写之外，其余的都是用 C 语言描述的。当然它同样可以表达数值处理、字处理功能，所以它又是一种通用语言。

2. 语言简洁，使用灵活，易于学习和应用。在表示方法上力求在明了易懂的前提下简单易行。如用一对{、}来代替复合语句括号 begin、end，利用赋值运算符（如+=、-=、*=、/=、&=、|=等）对运算和赋值的形式进行精简等等。另外，C 把一般语言的许多成分都通过显式函数调用来完成，使得编译程序小而精。比如 C 没有提供 I/O 设施，也没有并行操作、同步或协同程序等复杂控制，而是提供了大量有效的库函数来实现输入/输出、字符串处理及存储分配操作等功能，库函数可根据需要，方便地扩充。C 本身运行时所需支持少，存储空间占得也少。

3. 具有构造数据类型的能力，并有很强的控制流结构。它可以在简单类型（如字符、整数、浮点数等）的基础上按层次产生各种构造类型（如数组、指针、结构和联合等），因而 C 的数据类型很丰富。同时，它的各种控制语句如 if、while、for、switch、break 等，功能很强，足以描述结构良好的程序。此外，它的存储类说明（如 extern、static）在功能上有助于数据隐藏的模块化结构程序设计。

4. 语言生成的代码质量高。高级语言能否用来描述系统软件，特别是像操作系统、编译程序等，除要求语言表达能力强之外，很重要的一个因素是语言生成的代码质量如何。如果代码质量低，则系统开销就大，无实用价值。许多试验表明，针对同一问题用 C 编写程序，其生成代码的效率仅比用汇编语言写的代码低 10－20%。由于用高级语言描述比汇编语言描述问题编程迅速、工作量小、可读性好，易于调试、修改和移植，所以 C 就成了人们描述系统软件和应用软件比较理想的工具。在代码质量上 C 确实可以与汇编语言媲美。

5. 可移植性好。可移植是指程序可以从一个环境不加或稍加改动就可搬到一个完全不同的环境上运行。汇编语言因为依赖于机器硬件，所以根本不可移植，而一些高级语言如 Fortran 等其编译也不可移植。目前许多不同机器都配有 Fortran，基本上都是根据国际标准重新实现的。而 C 语言目前在许多机器上出现，大部分却是由 C 语言编译移植得到的。统计资料表明，不同机器上的 C 编译程序 80%的代码是公共的。C 语言的编译程序便于移植，也就使在某个环境上用 C 编写的许多程序可以很方便地移到另一个环境上。可移植性良好是 C 语言最为人称赞的特点。诚如 83 年度图灵奖的评论中所说的"对于 UNIX 系统可移植性的关键贡献是 Ritchie 开发的 C 程序设计语言"。

1.3 关于本书的几点说明

- 幻灯
- 教学重点
- 教学指导
- 交互性

1.3.1 幻灯

本书每一节的开头都列举了该节的内容概要，我们建议教师在授课时把这些内容制成幻灯。书中所提到的幻灯即是指这些内容。有必要指出，本书没有提供幻灯片，只在各个章节提供了幻灯的内容。

1.3.2 教学重点

本书每一节的内容概要之后都有一个关键语句，它是每一节内容的重点。

1.3.3 教学指导

教学指导中给出了该节内容的难点、教授的重点以及一些行之有效的教学方法。当然这仅是我们的一家之言，广大教师应该批判地吸收。教学指导对学生自学也是有帮助的。

1.3.4 交互性

讲授本书的最佳方法是教、学交互。条件许可的地方，应该给每一个学生都配备计算机。教师把学生要用的程序和工具输入计算机。教师在讲课时，学生在计算机上进行操作。即使无法满足上述条件，学生也应该在课后，在计算机上进行操作。

1.4 教程概要

- C 程序框架
- C 数据类型
- 获取和显示文本
- 算术和赋值运算符
- C 预处理器
- 比较运算符和流程控制

- 函数和原型
- 标识符作用域和存储类型
- 宏的使用
- 使用文件
- 使用标准输入输出

- 一维数组
- 指针
- 使用指针修改函数参数
- 字符串和常用字符串函数

- 指针数组

- 内存模式
- 位操作
- 动态内存分配

- 结构和联合
- 链表
- 函数指针
- 位域

第二章　C 程序框架

2.1　概　述

- Main()
- 函数和库
- C 编程规则
- 变量和函数的声明和初始化
- 保留字
- 语句
- 编译和连接

2.1.1　概述

在本章中，您将了解 C 程序的主要组成部分和用于建立 C 程序的约定。

2.1.2　前提

本章讨论的前提要求你已经具备至少一种语言的编程经验。

2.1.3　总体目标

学完本章后，你将会了解 C 程序的轮廓，其中包括编译伪指令、Main 函数、函数、库和 de facto C 编程规则。

2.1.4　学习目标

学完本章后，你将会：

1. 解释函数的目的和格式。
2. 使用 de facto C 编程规则，演示好坏两种风格的程序。
3. 编写使用 C 语言保留字的代码。
4. 编写正确声明变量的代码。
5. 用命令行正确地编译并连接 C 程序。

2.2　main ()

```
#include <stdio.h>
#define MILLIMETERS_PER_INCH 25.4
```

```
void main( )
    {
    }
```

1. main()函数是 C 程序的入口点。
2. 编译时不需要 main()函数，但连接时必须提供 main()函数。
3. 典型的 C 程序结构如下所示：

```
#include(s)
#define(s)
main()
所有其它函数
```

4. 使用显式常量的目的。
5. 初步理解头文件：它包括函数的描述，而不是实际代码。在所演示的例子中，stdio.h 给出了函数 printf()和 scanf()的描述。

2.2.1 样板程序

```
1   /*   INCHES.C   */
2
3   #include <stdio.h>
4   #define MILLIMETERS_PER_INCH 25.4
5
6   int millimeters;
7
8   void main()
9   {
10      double inches;
11      printf ("Enter the number of inches:");
12      scanf("%lf",&inches);
13
14      millimeters=convert(inches);
15
16      printf ("%lf inches=%d millimeters. \n",
17      inches,millimeters);
18  }
19
20  int convert (double d_inches)
21  {
22      int temp;
23      temp=d_inches * MILLIMETERS_PER_INCH;
24      return temp;
25  }
```

要使一个 C 程序能通过编译和连接，它必须有一个称为 main( )的函数。main( )作为程序的入口。典型地，main( )作为一个"驱动"函数，程序的实际工作由 main( )调用的函数完成。虽然技术上 main 不是 C 语言的保留字，但它绝对不能在命名入口函数以外的其它地方使用。

程序的实际代码必须放在函数的花括号之间。上述例子有四个函数：

- 两个用户编写的函数：main( )和 convert( )。
- 两个库函数：printf( )和 scanf( )。

C 程序也包含下面关键元素：

- 注释
- 显式常量
- 全局变量
- 局部变量
- include 文件

2.2.2 定义

所有 C 程序必须有一个称为 main( )的函数。

2.3 函数和库

- 函数
- 库

教学重点：**C** 程序利用库和函数。

教学指导：**解释函数和库的区别。库是目标模块的集合。每个目标模块由一个或多个编译后的函数组成。C 程序员接触的第一个库是 SLIBCE. LIB。printf( )中的代码驻留在 SLIBCE. LIB 库，由连接器抽取并放置到. EXE 文件中。**

C 程序员可以利用用户编写的函数，也可以利用标准库中的函数。

2.3.1 函数

1. 把程序分成逻辑完整的单独模块，使程序易于编写和调试。
2. 由于函数使用的私有变量对主程序和其它函数是不可见的，从而避免了不想要的副作用。
3. 函数避免了对用于经常操作任务的代码的不必要重复。
4. 函数相当于其它语言的子程序、过程或段落。
5. 函数是把一段计算封装在一个黑匣子中的便利方法。
6. 函数是可重复使用的。
7. C 语言中，函数常用于所有的 I/O。

2.3.2 使用函数的优点

1. 调试
2. 可重复使用性
3. 模块化

函数分为两个范畴：用户编写的函数和库函数。库是目标模块的集合，目标模块则是一个或多个编译后函数的集合。缺省使用的库命名为 SLIBCE. LIB。它包括四百多个单独的函数。

库函数分为两个范畴，ANSI 和非 ANSI。大概有一百五十个 ANSI 函数。ANSI 函数在其名字中绝对不能有下划线。但不是所有的非 ANSI 函数名中都有下划线。

例如，printf( )和 scanf( )是 ANSI 函数。getch( )、chdrive 和 bios_equiplist( )则不是。要了解 SLIBCE. LIB 和编译器所带的其它库中的函数的细节，请参考有关文档。

通常，C 程序是库函数和用户编写函数的混合。

2.4 C 编程规则

- 不带空格的例子
- 带空格的例子

教学重点：C 编译器忽略源文件中的空格。

教学指导：观察一个低质 C 程序，着重注意分号作为语句分隔符的作用。强调下列内容：

■缩格和空行对编译器不起作用。

■对编译器来说，2.4.2 节和 2.4.3 节的程序逻辑上是相同的。

■分号是语句结束符。

■简要指出 2.4.3 节中的程序第 6 行的函数原型的作用。

■前一个程序没有函数原型。

解释

■没有函数原型是个失策，程序只能在某些情形下正确运行(当函数返回整型数时，程序才能正确运行)。

2.4.1 C 程序中的 de facto 规则

1. C 忽略空格。
2. 一行一条语句。
3. 括号内语句缩格。
4. 括号在竖直方向上对齐。

2.4.2　不带空格的程序例子

```
/* VOLUME.C Calculate sphere's volume. */
#include <stdio.h>
#define PI 3.14F
float sphere(float);void main() {float volume;float
radius=3.0F;volume=sphere(radius);printf(
"Volume: %f\n",volume);}float sphere(float rad) {float
result;result=rad*rad*rad;result=4.0F*PI*result;result=
result/3.0F;return result;}
```

2.4.3　带空格的程序例子

```
/* VOLUME.C */

#include <stdio.h>
#define PI 3.14F

float sphere (float);

void main()
{
    float volume;
    float radius=3.0F;
    volume=sphere(radius);
    printf ("Volume:%f\n",volume);
}

float sphere(float rad)
{
    float result;
    result=rad*rad*rad;
    result=4.0F*PI*result;
    result=result/3.0F;
    return result;
}
```

这个程序常量中使用了 F，使常量为浮点数或为四个字节。C 中缺省为双精度浮点数或为 8 个字节，详细内容将在第三章中讲解。如果没有 F，编译时这个程序产生警告。

2.5　定义变量和声明函数

- 定义和初始化变量
- 声明和定义函数
- 声明新类型

教学重点：C 语言中，一个变量必须在声明后才能使用。

教学指导：学习怎样声明变量。同样，一个函数在使用前，必须声明给编译器。强调 2.5.

4 节中程序的以下内容：

■第 5 行的函数原型的作用。
■在第 9 行进行初始化的三个变量。
■除非进行显式初始化，否则变量内容没有定义，变量之中不是 0。
■实参名(x,y,z)不必与形参名(any,old,name)相同。

2.5.1 定义和初始化变量

1. 类似于 **COBOL** 的数据部分。
2. 在程序(在#**define** 之后)和函数中首先完成。
3. 在可执行代码前定义。
4. 如果在函数体外定义，则变量是全局可见的，否则变量只在函数之中可见(可见性将在其它章中详细介绍)。

2.5.2 声明和定义函数

1. 在 main 前面说明函数原型或声明函数。
2. 在 main 之后定义函数。
3. 函数声明使编译器能进行类型检查。这也称为函数原型化。

2.5.3 声明新类型

1. 创建用户定义的数据类型。
2. 在"结构"章中将进行详细介绍。

2.5.4 样板程序

```
1   /* SHOWME.C */
2
3   #include <stdio.h>
4
5   void showme (int,int,int);
6
7   void main()
8   {
9       int x=10,y=20,z=30;
10      showme (x,y,z);
11      printf ("x=%d y=%d z=%d\n",x,y,z);
12  }
13
14  void showme (int any,int old,int name)
15  {
16      printf("any=%d old=%d name=%d\n",
17      any,old,name);
18  }
```

注意：在 C 语言中，每个函数对于程序中其它函数通常都是可见的。也就是，任何函数都可以被包括它自己的所有函数所调用。用户不能定义函数的唯一之处是在另一个函数的定义之中。仅仅是由于 main( )启动和终止程序运行，所以函数定义可以以任何顺序出现在程序中。main()函数通常第一个出现在程序中，而其它函数定义紧随其后。

2.6　保留字

保留字

- ANSI
- Microsoft
- C++

教学重点：保留字分为两个范畴。

教学指导：展示 32 个 ANSI 保留字以及 Microsoft 保留字。强调 Microsoft 保留字不能移植，只能用于 Microsoft 或 Intel 微处理器。由于 Microsoft 保留字中有一个前导的下划线，因此，用户总能区别出 Microsoft 保留字。C7.0 的保留字有两个下划线，但是只有一个下划线的老格式也是可接受的。

2.6.1　ANSI 保留字

1. 都是小写字母
2. 也称为关键字

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

2.6.2　Microsoft 规定的保留字

QuickHelp 中的部分列表：

_asm	loadds
_based	_near

_cdecl	_pascal
_export	_saveregs
_far	_segment
fastcall	_segname
_fortran	_self
huge	_stdcall
_inline	syscall
interrupt	

为了和老版本的 Microsoft 编译器兼容，C 可以识别关键字的单下划线格式，如同无下划线的格式。优先使用双下划线格式。

下面的不是关键字，但在 Microsoft C 或 C++中有特殊的意义：

argc
_emit
main
_setenvp
argv
envp
_setargv
_set_new_handler

因为你将来有可能把 C 代码用到 C++代码中，你应避免使用下列 C++关键字，即使它在 C 程序是合法的。

class
new
public
delete
operator
this
friend
private
virtual
inline
protected

2.7 语 句

- 语句
- 注释

教学重点：分号，而不是新行号，是 C 语言的语句分隔符。

教学指导：演示一条 C 语句对于 C 编译器的意义，并演示在 C 程序插入注释的两个方法。指出简单语句和诸如 if 语句的复杂结构，对于 C 编译器都是单语句。有两种插入注释的方法，"//"不是 ANSI 标准。

2.7.1　语句

以分号结束。

2.7.2　样板语句

```
Printf("%S","This is string printing");
if(steve<=5)
    printf("%d is less than five",steve);
```

2.7.3　注释

1. 用/* */输入。
2. 注释不能嵌套。
3. Microsoft 也支持另外一种插入注释的方法：双斜杠。

```
// this is a comment
```

这种方法不是 ANSI 标准，它更适合于短注释。从//至本行末的所有内容都是注释。

2.8　编译和连接

■ 在一个操作中完成编译和连接

```
cl filename.c
```

■ 分别进行编译和连接

```
cl /c filename.c
link filename.obj
```

教学重点：可以在一次操作中完成编译和连接，也可以通过两个命令把编译和连接分成两个不同的操作。

教学指导：演示怎样完成简单的编译和连接。CL.EXE 本身不是编译器和连接器，而是调用不同遍数编译器的小驱动程序。如果用户没有指定/C 选项，则 CL.EXE 调用连接器。

不要拘泥于为什么可以把连接当作单独的操作。有关 NMAKE 的附录将对此进行详细介绍。

所显示的选项对于本课程将进行的大多数编译来说是足够的。强调：使用/qc 选项将加快编译。让学生在这里建立一个调用带有正确选项的编译的批处理文件。把该批处理文件称为 C.BAT，并把它置于路径中，通常在\TOOLS 子

目录中。如下所示：

```
@echo off
cl /qc %1.c
```

然后，在课堂上让学生编译程序，在提示符下键入：

```
c filename
```

文件名不带扩展名。此时，他们可以建立另一个批处理文件：D.BAT，即使 CodeView 将在下文中进行讨论。

```
@ echo off
cl /qc /Zi /Od %1.c
```

上面显示了编译一个包含由单个源文件构成的简单程序的两种方法。CL.EXE 程序本身不是编译器。它只是一个驱动程序，首先调用可运行编译程序，然后调用连接器。第一个选择是最经常使用的。

第二个选择显示了怎样只调用编译器以生成目标文件：FILENAME.OBJ。/c 选项使 CL.EXE 只调用编译器，它只生成目标文件。目标文件用于连接程序以生成 FILENAME.EXE。

在第二个例子中扩展名.c 对于 CL.EXE 的调用是必需的。由于 LINK 所用文件的扩展名缺省为.obj，所以在调用 LINK.EXE 时扩展名.obj 是可选的。一旦你由单个源文件生成一个程序，通常如第一个例子，那么在一次操作中就调用编译器和连接器。

编译器和连接器有很多选项，它们将在另外的章节中讨论。至此，你会发现下述将对你有所帮助：

Microsoft C 编译器实际是四个编译程序：

1. 单遍编译器，它很快，完成最少的优化，常用于开发。
2. 多遍编译器，它较慢，但能完成较广泛的优化，用于生成程序的最后版本。
3. 多遍 C++编译器。
4. 单遍 C++编译器。

缺省的 C 编译使用单遍编译器。它不同于 Microsoft C 编译器以前的版本，老版本缺省的是多遍编译器。

可以用/f 选项显式要求编译器进行单遍编译。

如果你要用 MS Code View 调试程序，编译时必须关闭所有优化，并加入调试符号。相应的选项是/Od 和/Zi。

如果你愿意用命令行调用编译器，你将发现下面三个批处理文件对于本课程是很有用的。把它们放到 TOOLS 子目录中。你可以键入批处理文件名(不带扩展名)来调用它们。

C.BAT 完成通常的单遍编译：

```
@ echo off
cl /f %1.C
```

D. BAT 完成用于调试的编译：

```
@ echo off
cl /f /Od /Zi %1.C
```

/f 选项是缺省的，所以它不是必需的，但包含它对于编写文档是有好处的。

O. BAT 完成所有优化编译：

```
@ echo off
cl /Ozaxb2 /G2r %1.c
```

如果在准备调试时希望把编译和连接分开，把 D. BAT 修改成如下：

```
@echo off
cl /c /Od /Zi %1.c
link /CO %1.obj
```

编译选项区分字母大小写，而连接选项则不然。

通过键入下列命令来熟悉编译器和连接器的其它选项：

```
cl /help
link /help
```

第三章 C 数据类型

3.1 概 述

- 数据类型
- 十六进制和八进制记数
- 命名变量和函数
- 常量
- 转义序列

3.1.1 概述

C 程序中最基本的元素是数据类型。本章向你介绍 C 程序中常用的十一种数据类型。

3.1.2 前提

本章是基本数据类型的简要回顾。它假设你已经精通其它主要编程语言，并假设你对所熟悉语言与 C 语言之间的区别感兴趣。

3.1.3 总体目标

本章的总体目标是为你提供编写 C 程序所需的基本工具集之一。

3.1.4 学习目标

在结束本章时，你将能够：

1. 解释十一种数据类型的正确使用。
2. 正确使用十六进制和八进制记数。
3. 正确命名变量。
4. 正确规定常量。
5. 正确使用转义序列。

3.2 数据类型

- 八种整型数据类型
- 三种浮点数据类型

教学重点：C 语言所支持的数据类型是很基本的。

教学指导：显示十一种标准数据类型。所有的数据类型都建立在这十一种基本类型之上。

对于字符类型，强调指出字符实际上只是一个字节的整型数。解释给一个字符变量赋值的六种方法与机器语言完全相同。

强调：一个整型量的尺寸在不同机器上、不同编译器之间是不同的。**Microsoft C7.0** 编译器产生十六位整数。

整数到字符的赋值是危险的。学生必须明白只有整数中的低字节赋值给了字符。对浮点型和双精度型的讨论是有趣的。对学生强调数据类型的范围、精度和内部格式，但并不要求记住。应强调指出，浮点型占用 **4** 字节存储量，双精度占用 **8** 个字节。

3.2.1　八种整型数据类型

char	字符型
int	整型
short	短整型
long	长整型
unsigned char	无符号字符型
unsigned short	无符号短整型
unsigned int	无符号整型
unsigned long	无符号长整型

3.2.2　三种浮点数据类型

double	双精度型
float	浮点型
long double	长双精度型

然而长双精度(long double)并不是所有系统都支持的。

标准的数据类型是：

- 基元，它易于移植。
- 所有处理器都支持。

3.2.3　字符(char)数据类型

特点：

- 缺省带有符号。
- 存储时占用 8 位或 1 个字节
- 带符号的字符变量可以是−128～127
- 无符号字符为 0～255
- 字符所有性质与 1 字节整型量相同