目　录

第一部分　预备知识

第二部分 案例拓展实训

第三部分 编程实训

第一部分

预备知识

第一部分介绍 ANSI C 的编程基础和基于接口的设计的概念。如果已经学过了 C 语言编程，那么这些内容是对所学知识的一个复习。

主要内容：

第 1 章　ANSI C 语言概述

第 2 章　C 的数据类型

第 3 章　库和接口

第 1 章　ANSI C 语言概述

本章学习目标：

- 了解典型 C 程序的各个组成部分。
- 理解 C 语言中预定义的数据类型以及它们如何在程序中存储信息。
- 熟悉简化形式的工具和简单的标准输入输出库，并用它们读取输入的数据和显示的结果。
- 理解 C 语言中表达式的结构，并掌握使用常见的运算符来表达计算过程。
- 了解 if、switch、while 和 for 语句的格式，并使用它们编写简单的程序。
- 认识到将程序分解成独立的函数的重要性，并理解那些函数的工作机制。
- 能够综合使用本章中的控制机制编写简单的程序。

1.1　C 语言简介

1.1.1　C 语言出现的历史背景

C 语言是国际上流行的、很有发展前途的计算机高级语言。C 语言适合于作为“系统描述语言”。它既可以用来编写系统软件，也可以用来编写应用程序。

以前操作系统等系统软件主要采用汇编语言编写。汇编语言依赖于计算机硬件，程序的可读性、可移植性都比较差。为了提高可读性和可移植性，人们希望采用高级语言编写这些软件，但是一般的高级语言难以实现汇编语言的某些操作，特别是针对硬件的一些操作（如：内存地址的读写 - 直接硬件、二进制位的操作）。人们设法寻找一种既具有一般高级语言特性，又具有低级语言特性的语言，C 语言就在这种情况下应运而生的。

C 语言的发展见下图：

ALGOL60 - > CPL - > BCPC - > B - > C - > 标准 C - > ANSI C - > ISO C

- ALGOL60：一种面向问题的高级语言。ALGOL60 离硬件较远，不适合编写系统程序。

• CPL（Combined Programming language，组合编程语言）：CPL 是一种在 ALGOL60 基础上更接近硬件的一种语言。CPL 规模大，实现困难。

• BCPL（Basic Combined Programming language，基本的组合编程语言）：BCPL 是对 CPL 进行简化后的一种语言。

• B 语言：是对 BCPL 进一步简化所得到的一种很简单接近硬件的语言。B 语言取 BCPL 语言的第一个字母。B 语言精练、接近硬件，但过于简单，数据无类型。B 语言诞生后，Unix 开始用 B 语言改写。

• C 语言：是在 B 语言基础上增加数据类型而设计出的一种语言。C 语言取 BCPL 的第二个字母。C 语言诞生后，Unix 很快用 C 语言改写，并被移植到其他计算机系统。

• 标准 C 、ANSI C、ISO C：C 语言的标准化。

注：最初 Unix 操作系统是采用汇编语言编写的，B 语言版本的 Unix 是第一个用高级语言编写的 Unix。在 C 语言诞生后，Unix 很快用 C 语言改写，C 语言良好的可移植性很快使 Unix 从 PDP 计算机移植到其他计算机平台，随着 Unix 的广泛应用，C 语言也得到推广。从此 C 语言和 Unix 像一对孪生兄弟，在发展中相辅相成，Unix 和 C 很快风靡全球。

从 C 语言的发展历史可以看出，C 语言是一种既具有一般高级语言特性（ALGOL60 带来的高级语言特性），又具有低级语言特性（BCPL 带来的接近硬件的低级语言特性）的程序设计语言。C 语言从一开始就是用于编写大型、复杂系统软件的，当然 C 语言也可以用来编写一般的应用程序。也就是说：C 语言是程序员的语言！

IBM PC 微机 DOS、Windows 平台上常见的 C 语言版本有：

• Borland 公司：

Turbo C，Turbo C + + ，Borland C + +

C + + Builder（Windows 版本）

• Microsoft 公司：

Microsoft C

Visual C + + （Windows 版本）

1.1.2 C 语言的特点

C 语言是从“组合编程语言”CPL 发展而来，C 语言既具有一般高级语言特性（ALGOL60 带来的高级语言特性），又具有低级语言特性（BCPL 带来的接近硬件的低级语言特性）。C 语言具有以下特点（其中 1 - 6 属于高级语言特性，7，8 属于低

级语言特性）

1. C 语言的语言成分简洁，紧凑，书写形式自由

例：将 C 语言程序段与实现同样功能的 PASCAL 语言程序段进行比较。

	C 语言	PASCAL 语言	含义	说明
1	{…}	BEGIN…END	复合语句（或：语句块）	PASCAL 显得啰唆
2	if（e）S;	IF（e）THEN S;	条件语句	PASCAL 至少多了一个 THEN 关键词
3	int i;	VAR i：INTEGER	定义 i 为整型变量	PASCAL 至少多了一个 VAR 关键词
4	int a［10］;	VAR a：ARRAY［1. .10］OF INTEGER	定义 a 为整型一维数组，10 个元素	PASCAL 多了 VAR、ARRAY、OF 等关键词
5	int f（）;	FUNCTION f（）：INTEGER	定义 f 为返回值为整型的函数	PASCAL 至少多了一个 FUNCTION 关键词
6	int ＊p;	VAR p：^INTEGER	定义 p 为指向整型变量的指针变量	PASCAL 至少多了一个 VAR 关键词
7	i＋＝2;	i：＝i＋2	赋值语句	C 中如果将一个变量与另外一个操作数运算后赋值给原来的变量，使用复合的运算符可以不要重复书写此变量。C 形式上更加简洁
8	i＋＋	i＝i＋1	i 自增 1	C 定义了常用的自增 1、自减 1 运算符。形式上显得相当简洁

2. C 语言拥有丰富的数据类型

C 语言具有整型、实型、字符型、数组类型、指针类型、结构体类型、共同体类型等数据类型。能方便地构造更加复杂的数据结构（如：使用指针构造链表、树、栈）。

3. C 语言的运算符丰富、功能更强大

例如：

（1）C 语言具有复合的赋值运算符“＋［－＊/%］＝”（加等、减等、乘等、除等），“＞＞＝”“＜＜＝”（右移等、左移等），“&［^|］＝”（与等、或等、非等）。

x＋＝5 等价于 x＝x＋5

（2）C 语言有条件运算符“?:”可代替简单的 if/else 语句。

如果需要表示："如果 x 小于或等于 0，y 为 0；否则 y 为 1" 可以采用：

```
y = x < =0? 0: 1;
```

如果用一般的程序设计语言表示就应该像下面这样：

```
if (x < =0) y =0;
else y =1;
```

C 语言避免了一切可能的啰唆。

（3）C 语言中连赋值这种操作都定义为运算符，也就是说赋值操作本身可以作为表达式的一部分，参与运算。如：

```
if ( (p = malloc (sizeof (int))) = =NULL) {printf ( "Error!"); exit (1);}
```

如果改写为一般形式：

```
p = malloc (sizeof (int));
if (p = =NULL) {printf ( "Error!"); exit (1);}
```

又如下面算式是正确的：

```
x = y = z =6;
```

如果改写为一般形式：

```
z =6; y =6; x =6
```

4. C 语言是结构化的程序设计语言

- C 语言具有结构化的控制语句（if/else，switch/case，for，while，do…while）
- 函数是 C 语言程序的模块单位。

5. C 语言对语法限制不严格，程序设计灵活

C 语言不检查数组下标越界，C 语言不限制对各种数据转化（编译系统可能对不合适的转化进行警告，但不限制），不限制指针的使用，程序正确性由程序员保证。

实践中，C 语言程序编译时会提示："警告错误""严重错误"。警告错误表示你使用的语法可能有问题，但是你有时可以忽略，你的程序仍然可以完成编译工作，然后运行。（但是一般情况下警告错误，往往意味着程序真的有问题，你应该认真地检查）"严重错误" 是不能忽略的，编译系统发现严重错误，就不会产生目标代码。

灵活和安全是一对矛盾，对语法限制的不严格可能也是 C 语言的一个缺点，比如：黑客可能使用越界的数组攻击你的计算机系统。JAVA 语言是优秀的网络应用程序开发语言，它必须保证安全性，它绝对不允许数组越界。此外 JAVA 不使用指针，不能直接操作客户计算机上的文件，语法检查相当严格，程序正确性容易保证，但是 JAVA 在编程时却缺乏灵活。

6. C 语言编写的程序具有良好的可移植性

编制的程序基本上不需要修改或只需要少量修改就可以移植到其他的计算机系统

或其他的操作系统。

7. C 语言可以实现汇编语言的大部分功能

- C 语言可以直接操作计算机硬件如寄存器，各种外设 I/O 端口等。
- C 语言的指针可以直接访问内存物理地址。
- C 语言类似汇编语言的位操作可以方便地检查系统硬件的状态。
- C 语言适合编写系统软件。

8. C 语言编译后生成的目标代码小，质量高，程序的执行效率高

有资料显示只比汇编代码效率低 10% －20% 。

1.1.3　C 语言基本语法成分

1. C 语言字符集：

字符是 C 语言的最基本的元素，C 语言字符集由字母、数字、空白、标点和特殊字符组成（在字符串常量和注释中还可以使用汉字等其他图形符号）。由字符集中的字符可以构成 C 语言进一步的语法成分（如，标识符，关键词，运算符等）。

（1）字母：A －Z，a －z

（2）数字：0 －9

（3）空白符：空格，制表符（跳格），换行符（空行）的总称。空白符除了在字符，字符串中有意义外，编译系统忽略其他位置的空白。空白符在程序中只是起到间隔作用。在程序的恰当位置使用空白将使程序更加清晰，增强程序的可读性。

（4）标点符号、特殊字符：

!	#	%	^	&	+	－	*	/	=	~	<	>	\	|	.	,	;	:	?	‘	“	(	)	[	]	{	}

2. 标识符（名字）：

用来标识变量名、符号常量名、函数名、数组名、类型名等实体（程序对象）的有效字符序列。标识符由用户自定义（取名字）。

C 语言标识符定义规则：

1）标识符只能由字母、数字和下划线三种字符组成，且第一个字符必须为字母或下划线。

例如：

合法的标识符：sum，average，_ total，Class，day，stu_ name，p4050

不合法的标识符：M. D. John，$123，#33，3D64，a > b

2）大小写敏感。C 程序员习惯：变量名小写，常量名大写，但不绝对，如 windows 编程，应当使用匈牙利表示法（大小写混用，每个单词词首第一个大写，其余小写，如 WinMain）。

例如：sum 不同 Sum。BOOK 不同 book。

3）ANSI C 没有限制标识符长度，但各个编译系统都有自己的规定和限制（TC 32 个字符，MSC 8 个字符）。

例如：student_ name，student_ number 如果取 8 个，这两个标识符是相同的。

4）标识符不能与“关键词”同名，也不与系统预先定义的“标准标识符”同名。

5）建议：标识符命名应当有一定的意义，做到见名知义。

3. 关键词（保留字）：

C 语言规定的具有特定意义的字符串。参阅 P234 附录 V。

4. 运算符：

运算符将常量、变量、函数连接起来组成表达式，表示各种运算。运算符可以由一个或多个字符组成。参阅 P233 附录 IV。

运算符根据参与运算的操作数的个数分为：单目、双目、三目运算符。

5. 分隔符：

逗号，空格。起分隔、间隔作用。

6. 注释符：

“/*”和“*/”构成一组注释符。编译系统将/*... */之间的所有内容看作为注释，编译时编译系统忽略注释。

- 注释在程序中的作用：提示、解释作用。

注释与软件的文档同等重要，要养成使用注释的良好习惯，这对软件的维护相当重要。记住：程序是要给别人看的，自己也许还会看自己几年前编制的程序（相当别人看你的程序），清晰的注释有助于他人理解您的程序、算法的思路。

- 在软件开发过程中，还可以将注释用于程序的调试 - 暂时屏蔽一些语句。

例如，在调式程序时暂时不需要运行某段语句，而你又不希望立即从程序中删除它们，可以使用注释符将这段程序框起来，暂时屏蔽这段程序，以后可以方便地恢复。

1.2 C 语言程序的编辑、编译、运行

1.2.1 源程序、 目标程序、 可执行程序的概念

1. 程序：为了使计算机能按照人们的意志工作，就要根据问题的要求，编写相应的程序。程序是一组计算机可以识别和执行的指令，每一条指令使计算机执行特定的操作。

源程序：程序可以用高级语言或汇编语言编写，用高级语言或汇编语言编写的程

序称为源程序。C 程序源程序的扩展名为“. c”

事实上我们编写的程序，不管采用什么计算机语言，都是源程序，有几个人还会用机器语言去编程！

源程序不能直接在计算机上执行，需要用“编译程序”将源程序翻译为二进制形式的代码。

2. 目标程序：源程序经过“编译程序”翻译所得到的二进制代码称为目标程序。目标程序的扩展名为“. obj”

目标代码尽管已经是机器指令，但是还不能运行，因为目标程序还没有解决函数调用问题，需要将各个目标程序与库函数连接，才能形成完整的可执行的程序。

3. 可执行程序：目标程序与库函数连接，形成完整的可在操作系统下独立执行的程序称为可执行程序。可执行程序的扩展名为“. exe”（在 dos/windows 环境下）

1.2.2　C 语言程序的上机步骤

输入与编辑源程序 - >编译源程序，产生目标代码 - >

连接各个目标代码、库函数，产生可执行程序 - >运行程序。

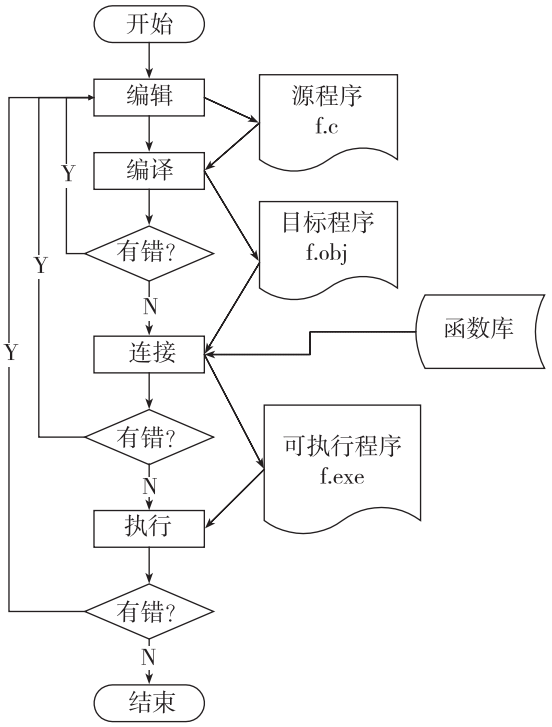


图 1.1　C 程序的上机步骤

1.2.3 Turbo C 的使用 (DOS 环境)

Turbo C2.0 是 Borland 公司开发的微机上一个 C 语言集成开发环境。可以在 Turbo C 中完成 C 语言程序的编辑、编译、连接、运行、调试。

重点菜单项：

File - >new，load，save，write to

Compile - >compile to obj，make exe file，build all

Run - >run，go to cursor，trace into，step over，user screen，program reset

Debug - >Evaluate

1.2.4 Visual C 的使用 （Windows 环境）

Visual C 是 windows 环境下的 C 应用程序，C + +应用程序，Windows 应用程序集成开发环境。

1.3 C 语言调试环境实训要点

1.3.1 实训目的和要求

1. 掌握开发环境的操作以及有效管理用户程序文件。

2. 熟悉 C 程序的实现过程和方法。

（1）掌握编辑、编译、连接、运行程序的过程和方法。

（2）了解常见的两种语法错误（Error/Warning）。

3. 了解集成开发环境强大的功能。

1.3.2 实训内容和具体步骤

1. VC + +6.0 开发环境

（1）启动

安装完 VC + +6.0 后，双击桌面快捷方式或从开始菜单，“程序——VC + +6.0”中点击该程序，得到图 1.3.7 窗口界面。具体英文注释如黄标。

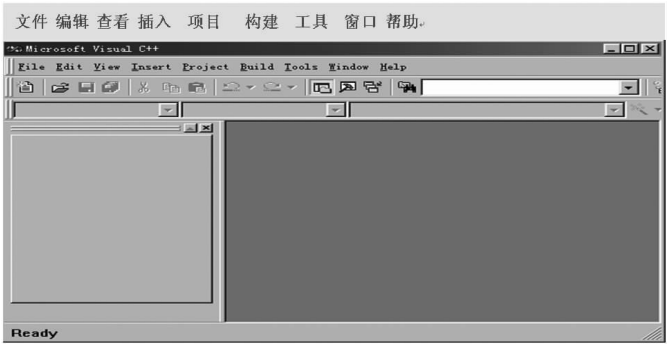


图 1.3.7 VC + +6.0 窗口界面

（2）编辑源程序

先从文件（File）菜单下拉菜单选建立新文件（New），如图 1.3.8 所示过程。选择后出现图 1.3.9 所示界面，选中 C + + Sourse File，在图右侧位置，起好文件名及保存位置，注意文件类型以. C 命名。之后就出现图 1.3.10 所示界面。在此界面就可以输入源程序了。也可以选择打开已有 C 程序，从文件（File）菜单下拉菜单选打开文件（open），选择相应文件即可。

图 1.3.8　新建文件过程

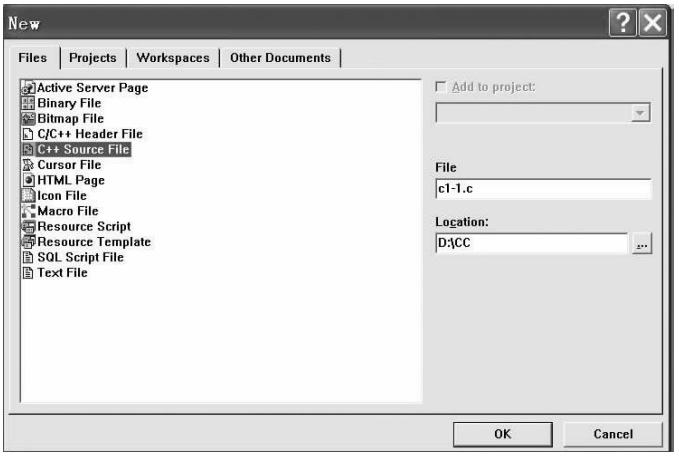


图 1.3.9　文件类型选择界面

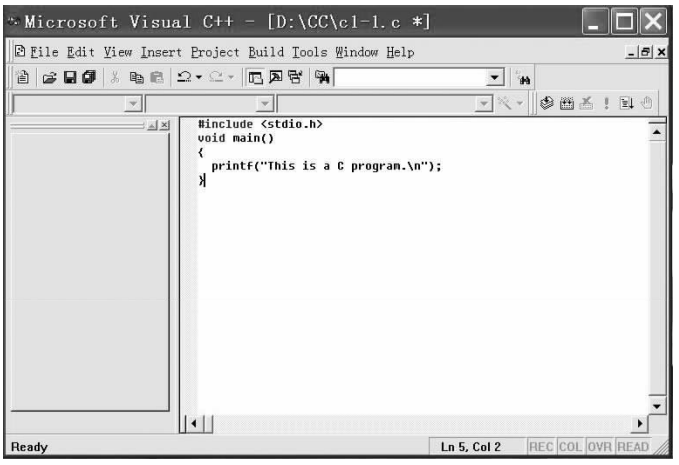


图 1.3.10　文件编辑界面

（3）保存文件。

在文件（File）菜单下点击 Save ，如图 1.3.11 或直接点击图标 ，文件就以您命名的名字保存在定义的路径下了。

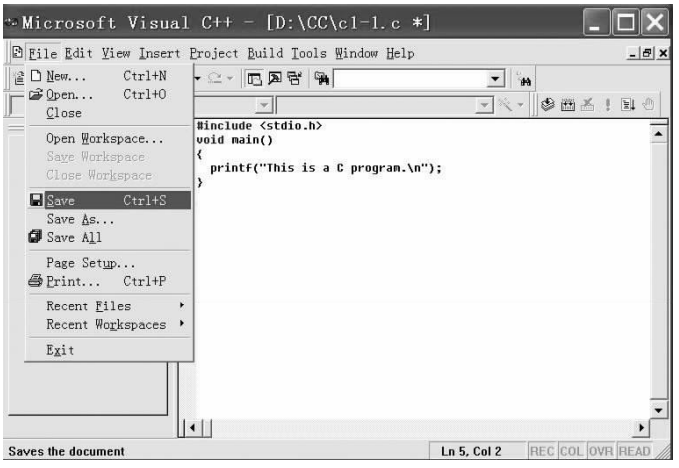


图 1.3.11　文件保存过程

（4）源程序编译

从 Build 菜单下选择 Compile，或单击图标，对程序进行编译，过程如图 1.3.12 所示。

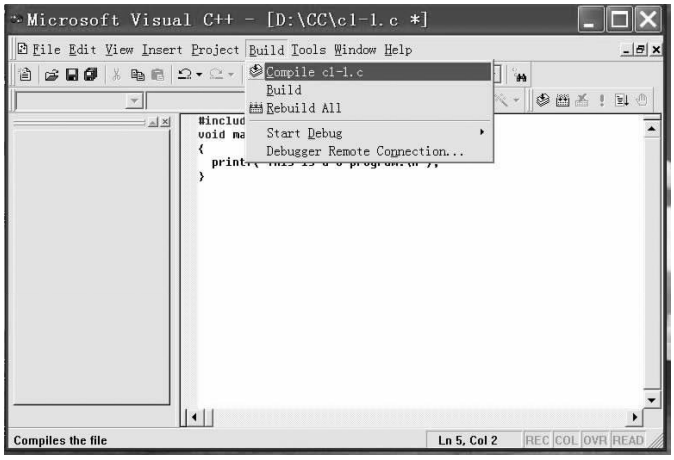


图 1.3.12　编译过程

（5）连接

从 Build 菜单下选择 Build，对程序进行连接，过程如图 1.3.13 所示。

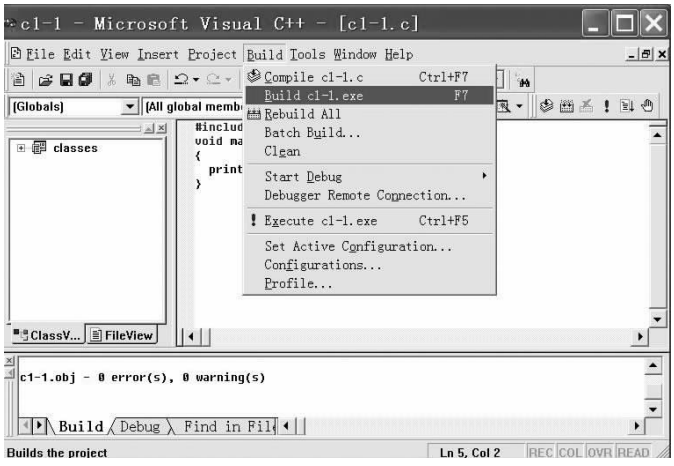


图 1.3.13　连接过程

（6）运行

从 Build 菜单下选择 Run，或单击图标，运行程序，得到结果如图 1.3.14 所示。按任意键即可推出此界面。

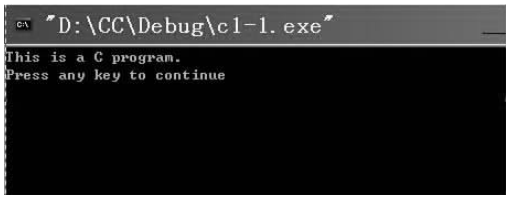


图 1.3.14　运行结果界面

2. 程序调试过程

所谓程序调试是指对程序的查错和排错。调试程序步骤：

（1）先进行人工检查，即静态检查。

（2）上机调试。

（3）在改正语法错误和“警告”后，程序经过连接（link）就得到可执行的目标程序。运行程序，输入程序所需数据，就可得到运行结果。

1.4 C 语言程序的结构实训要点

1.4.1 简单的 C 程序介绍

```
例 1.1：
int main（void）
{
    printf（" This is a C program. \ n"）;
    return 0;
}
```

说明：本程序的功能是输出一行信息：This is a C program.

其中：

1. main 表示“主函数”。每个 C 语言程序都必须有一个 main 函数，它是每一个 C 语言程序的执行起始点（入口点）。main（）表示“主函数” main 的函数头。

2. 用 {} 括起来的是“主函数” main 的函数体。main 函数中的所有操作（或：语句）都在这一对 {} 之间。也就是说 main 函数的所有操作都在 main 函数体中。

3. “主函数” main 中只有一条语句，它是 C 语言的库函数，功能是用于程序的输出（显示在屏幕上），本例用于将一个字符串“This is a C program. \ n”的内容输出。即在屏幕上显示：

This is a C program.

_ （回车/换行）

4. 注意：每条语句用“;”号结束语句。

```
例 1.2：
int main（void）/ * 计算两数之和 * /
{
  int a，b，sum; / * 这是定义变量 * /
```

```
    a = 123; b = 456; /* 以下 3 行为 C 语句 */
    sum = a + b;
    printf ("sum = %d\n", sum);
    return 0;
}
```

说明：本程序计算两数之和，并输出结果。

1. 同样此程序也必须包含一个 main 函数作为程序执行的起点。{} 之间为 main 函数的函数体，main 函数所有操作均在 main 函数体中。

2. /* */括起来的部分是一段注释，注释只是为了改善程序的可读性，在编译、运行时不起作用（事实上编译时会跳过注释，目标代码中不会包含注释）。注释可以放在程序任何位置，并允许占用多行，只是需要注意“/*”、“*/”匹配，一般不要嵌套注释。

3. int a，b，sum；是变量声明。声明了三个具有整数类型的变量 a，b，sum。C 语言的变量必须先声明再使用。

4. a = 123；b = 456；是两条赋值语句。将整数 123 赋给整型变量 a，将整数 456 赋给整型变量 b。a，b 两个变量分别为 123，456。注意这是两条赋值语句，每条语句均用“;”结束。

也可以将两条语句写成两行，即：

```
a = 123;
b = 456;
```

由此可见 C 语言程序的书写可以相当随意，但是为了保证容易阅读要遵循一定的规范。

1. sum = a + b；是将 a，b 两变量内容相加，然后将结果赋值给整型变量 sum。此时 sum 的内容为 579。

2. printf（“sum = %d\n”，sum）；是调用库函数输出 sum 的结果。%d 为格式控制表示 sum 的值以十进制整数形式输出。程序运行后，输出（显示）：

```
sum = 579
_ （回车/换行）
```

```
例 1.3
int main（void）/* 主函数 */
{/* main 函数体开始 */
  int a，b，c；/*声明部分定义变量*/
  scanf（“%d,%d”，&a，&b）；

  /* 调用 max，将调用结果赋给 c */
  c = max（a，b）；

  printf（“max = %d”，c）；
  eturno；
} /* main 函数体结束 */
```

```
int max（int x，int y）/* 计算两数中较大的数 */
{/* max 函数体开始 */
  int z；/* 声明部分，定义变量 */
  if（x > y）z = x；
  else z = y；
  return z；/* 将 z 值返回，通过 max 带回调用处 */
} /* max 函数体结束 */
```

说明：输入两个整数，计算两者较大的数，并输出。

1. 本程序包括两个函数。其中主函数 main 仍然是整个程序执行的起点。函数 max 计算两数中较大的数。

2. 主函数 main 调用 scanf 函数获得两个整数，存入 a，b 两个变量，然后调用函数 max 获得两个数字中较大的值，并赋给变量 c。最后输出变量 c 的值（结果）。

3. int max（int x，int y）是函数 max 的函数头，函数 max 的函数头表明此函数获得两个整数，返回一个整数。

4. 函数 max 同样也用 {} 将函数体括起来。max 的函数体是函数 max 的具体实现。从参数表获得数据，处理后得到结果 z，然后将 z 返回调用函数 main。

5. 本例还表明函数除了调用库函数外，还可以调用用户自己定义，编制的函数。

1.4.2 注释

“注释（comments）”是用来记录和描述程序的文本语句。注释语句可有可无，是一种非执行语句（即计算机不预处理）置于程序中，起到解释程序的作用和编码方式。在 C 语言中注释是以“/*”符号作为开头；以“*/”符号作为结尾。

注释是非常重要的一种机制。千万不能被忽视了。没有注释的程序不能算合格的程序。要建立这样的观念：程序是给别人编的，让人家看懂是第一位重要的事情。特别是将来你可能要参加一个团队，几十个人或几百人一起合作编程，相互协同，就更须将注释写得清清楚楚，明明白白。例如，微软公司在开发 Windows2000 时，就软件开发人员 1700 余人，可想而知，程序注释的重要性。

关于程序注释内容通常由以下几项组成：

（1）程序名称

（2）程序功能

（3）程序的思路和特点

（4）编程的人与合作者

（5）一些特殊变量的说明

（6）编程的日期及修改后的版本

（7）模块接口等

以下是某程序段，请大家注意其中的注释部分。

1.4.3　包含的库文件

C 语言本身不提供输入/输出语句，输入/输出的操作是通过调用库函数（scanf，printf）完成。输入/输出操作涉及具体计算机硬件，把输入/输出操作放在函数中处理，可以简化 C 语言和 C 的编译系统，便于 C 语言在各种计算机上实现。不同的计算机系统需要对函数库中的函数做不同的处理，以便实现同样或类似的功能。不同的计算机系统除了提供函数库中的标准函数外，还按照硬件的情况提供一些专门的函数。因此不同计算机系统提供的函数数量、功能会有一定差异。

1.4.4　函数原型

例 1.3 中的 max 函数定义在 main 函数之后，这就需要我们提前通知 main 函数，max 函数中需要两个形式参数，并且还返回 int 型的值。因此需要声明，int max（int x，int y）。

像这样明确记述了函数返回值类型，以及形参的类型和个数等的声明称为函数原型声明，需要注意的是该声明要以分号结尾。

函数原型的声明只声明了函数的返回值和形参等相关信息，并没有定义函数的实体。用来定义实体的是函数定义。

另外，如果 max 函数的返回值和形参等发生了改变，那么函数定义和函数原型声明两部分都必须进行修改。

但是，在编写程序的时候，如果把 max 函数放在 main 函数之前，就不用特意使用函数原型声明了。

一般，把 main 函数放在程序的最后位置，而把被调用的函数放在程序前部是比较好的选择。

1.4.5　函数定义

一个函数由函数首部和函数体两部分组成（以例 1.3 的 max 函数说明）

（1）函数首部：一个函数的第一行。

返回值类型 函数名（［函数参数类型 1 函数参数名 1］［，…，函数参数类型 2，

函数参数名 2]）

注意：函数可以没有参数，但是后面的一对（）不能省略，这是格式的规定。

（2）函数体：函数首部下用一对｛｝括起来的部分。如果函数体内有多个｛｝，最外层是函数体的范围。函数体一般包括声明部分、执行部分两部分。

｛

[声明部分]：在这部分定义本函数所使用的变量。

[执行部分]：由若干条语句组成命令序列（可以在其中调用其他函数）。

｝

1.4.6 主程序

1. C 程序由函数构成（C 是函数式的语言，函数是 C 程序的基本单位）（以例 1.3 说明）

一个 C 源程序至少包含一个 main 函数，也可以包含一个 main 函数和若干个其他函数。函数是 C 程序的基本单位。被调用的函数可以是系统提供的库函数，也可以是用户根据需要自己编写设计的函数。C 是函数式的语言，程序的全部工作都是由各个函数完成。编写 C 程序就是编写一个个函数。C 函数库非常丰富，ANSI C 提供 100 多个库函数，Turbo C 提供 300 多个库函数。

2. main 函数（主函数）是每个程序执行的起始点（以例 1.3 说明）

一个 C 程序总是从 main 函数开始执行，而不论 main 函数在程序中的位置。可以将 main 函数放在整个程序的最前面，也可以放在整个程序的最后，或者放在其他函数之间。

3. C 程序书写格式自由

- 一行可以写几个语句，一个语句也可以写在多行上。
- C 程序没有行号，也没有 FORTRAN，COBOL 那样严格规定书写格式（语句必须从某一列开始）。
- 每条语句的最后必须有一个分号“;”表示语句的结束。

1.4.7 C 程序设计风格

“风格（style）”通常上指画家、作家们在创作自己作品时所表达自己作品题材的一种方式。在这里的风格（coding style）则指的是计算机程序员在设计程序时所表现出来的特点、习惯及逻辑思路等。一个公认的、良好的 coding style 可以减少编码的错误，减少读程序的时间，从而提高软件开发的效率。

从 70 年代以来，编码的目标从强调效率转变到强调清晰。与此相反，程序设计的风格也从追求“聪明”和“技巧”，变为“简明”和“直接”。人们逐渐认识到，良好的程序设计风格能在一定程度上弥补语言存在的缺点。反之，如果不注意风格，

即使使用了结构化现代语言，也很难写出高质量的程序。特别是对于一个软件开发团队来讲，尤其需要强调良好的和一致的 coding style。

以下从两个方面来阐述 C 程序设计风格问题：

1. 源代码的书写格式

对于 C 程序设计语言的书写格式来讲，它的书写格式规范非常灵活，即一行可允许写多条语句；也可以一条语句分多行书写，其优点是灵活性比较大，非常随意，但这也意味着给程序的可读性带来不便，如下面的 C 程序的书写格式。

```
main(){long int s[2][5]={150001,150002,150003,150004,150005, 1000,500
,230,700,998};char ok='y'; while (ok=='y'){ printf("input number(150001~
150005):\n"); scanf("%d\n", &x);if((x<150001)&&(x>150005)){ printf(" D
ata error!  \n" ); goto LB; }  while(x!=s[0][i]) i++; printf("%d,%d\n", s[0][i]
, s[1][i]); LB: printf(" Continue(y/n)?");  scanf("%d\n", &ok); }  }
```

这段代码虽然节省了书写空间，但它却给大大降低了程序的可读性，同时也增强了程序修改的难度，显然这不是好 coding style。

比较好的代码书写格式是采用“缩进式”的书写风格，即对每一组语句进行层次上划分，这样可以大大提高程序的可读性，也为程序的调试修改带来很大的方便。如下面采用的是缩进格式书写的 C 程序。

```
main()
 { long int s[2][5]={150001,150002,150003,150004,150005,
                     1000,500 ,230,700,998};
   char ok='y';
   while (ok=='y')
     { printf("input number(150001~150005):\n");
       scanf("%d\n", &x);
       if((x<150001)&&(x>150005))
           { printf(" Data error!  \n" );
             goto LB;
           }
       while(x!=s[0][i])
          i++;
       printf("%d,%d\n", s[0][i], s[1][i]);
LB:    printf(" Continue(y/n)?" );
       scanf("%d\n", &ok);
     }
 }
```

两段代码内容完全一样，只是在书写格式上不同，可以从可读性的角度来评价这两段代码，哪一个更 OK?

1.4.8　程序的运行效率

“效率（efficiency）”一般是指对处理机时间和存储空间的使用效率（即时间和空间的问题）。通常选择良好的设计方法才是提高程序效率的根本途径，设计良好的数据结构与算法，都是提高程序效率的重要方法。

下面以一个典型的小程序段来说明如何利用好时间和空间的问题。

举例：请编程实现求解下列问题。

$$\sum_{i=1}^{20}\sum_{j=1}^{3}(i+1)^2(j+2)$$

这是一个嵌套求累加和的问题。如果不考虑效率的问题，通常程序员一定会按下列方法设计程序：

```
main()
 { int i,j; float sum; sum=0.0;
   for(i=1;i<=20;i++)
     for(j=1;j<=3;j++)
       sum=sum+(i+1)*(i+1)*(j+2);
  printf("sum=%d\n",sum);
 }
```

在对上述的程序的分析中，从书写风格来说，程序的可读性比较好，程序所完成功能可以说是一目了然，定义变量的数量也比较得当。但是如果要从效率来分析的话，就会发现有下列问题存在：

①从循环次数来看，内循环要执行 20 * 4 = 80 次，再加上外循环所要执行的 21 次，所以总循环为 101 次。如果我们做一次调整，则循环次数会有所变化，请观察以下被调整后的 C 程序：

```
main()
 { int i,j; float sum; sum=0.0;
   for(j=1;j<=3;j++)
     for(i=1;i<=20;i++)
       sum=sum+(i+1)*(i+1)*(j+2);
  printf("sum=%d\n",sum);
 }
```

可以发现整体的循环次数减少了，即内循环减到 3 * 21 = 63 次，外循环减到 4 次，总循环为 67 次。所以，通过运行调整后，我们可以得出，应该把循环变量变化大的值位于内循环。

② 从求和的表达式来看，其中的 j + 2 运算只与外循环的变量 j 有关系，j + 2 运算共执行了 3 * 20 = 60 次，其中有 57 次是重复的运算。如果从运算效率的角度来做一次语句的调整，则情况会有所不同，请分析调整后的 C 程序。

从对调整后的程序分析得出，j + 2 运算只执行了 3 次。从而可以总结出：应尽量把与本循环无关的运算移到循环外面。

关于本程序，还可以继续对其效率分析。在求和表达式的 i + 1 运算中，每做一次循环则要执行 2 次 i + 1 运算，这样总共就要执行 2 * 60 = 120 次，显然要有 60 次是重复计算的。进一步调整程序：

```
main()
 { int i,j,k,m; float sum; sum=0.0;
   for(j=1;j<=3;j++)
     { k=j+2;
       for(i=1;i<=20;i++)
          { m=i+1;
            sum=sum+m*m*k;
       printf("sum=%d\n",sum);
     }
 }
```

对于调整后的程序中的 i +1 运算只要执行 60 次，减少了一倍。

以上提到的两个方面，对编程人员来讲是必须具备的。在这里有必要要求大家养成良好的编程习惯：

- 程序构思要有说明
- 强调可读性
- 变量要加注释
- 学会如何调试程序
- 对运行结果要做正确与否的分析

1.5　变量、值和类型实训要点

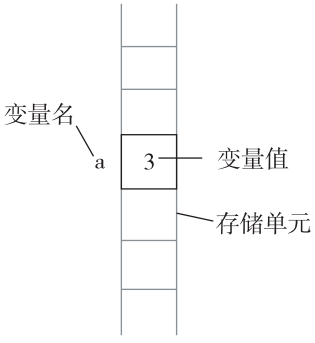
1.5.1　变量

变量：在程序的运行过程中，其值可以改变的量称为变量。

注意：

1. 变量名（用标识符表示）、变量在内存中占据的存储单元、变量值三者关系。

变量名在程序运行过程中不会改变，变量的值可以改变。变量名遵守标识符准则。



2. C 语言中变量："先定义，后使用"。就是说，C 要求对所有用到的变量做强制定义。

1）只有申明过的变量才可以在程序中使用，这使得变量名的拼写错误容易发