第 1 章　单片机的 C 语言概述

我们都知道，在单片机应用系统开发过程中，软件编程占有非常重要的地位。尤其是随着单片机技术的发展，嵌入式系统的推广和应用，硬件的集成化程度越来越高，同时对软件编程的要求也越来越高。这就要求单片机开发人员能在短时间内编写出执行效率高、运行可靠的代码。同时，由于实际系统的日趋复杂，对使用代码的规范性、模块化的要求越来越高，要方便多个工程师以软件工程的形式进行协同开发。在这种形势下，仅靠单片机在推广应用的初期使用的汇编语言来进行软件开发是远远不够的。

C 语言是近年来在国内外普遍使用的一种程序设计语言。C 语言能直接对计算机硬件进行操作，既有高级语言的特点，又有汇编语言的特点，因此在单片机应用系统开发过程中得到了非常广泛的应用。

在单片机应用系统设计与开发过程中，只要简单地熟悉相应单片机的硬件结构，利用 C 语言作为编程语言，就可以大大缩短开发周期。本章主要对单片机的 C 语言的基本问题进行概括的说明。

1.1　C 语言与 51 单片机编程

嵌入式单片机在开发过程中的编程语言主要有汇编语言和 C 语言。汇编语言作为传统的嵌入式系统的编程语言，已经不能满足实际需要了，而 C 语言的结构化和高效性成为电子工程师在进行嵌入式系统编程时的首选语言，并得以广泛应用。尤其是 C 语言编译系统的发展，更加促进了 C 语言的应用。1985 年出现了针对 8051 单片机的 C51 编译器，进而又出现了其他流行的嵌入式处理器系统，如 196 系列、PIC 系列、MOTORAL 系列、MSP430 系列、AD 公司和 TI 公司的 DSP 系列的 C 语言编译系统，以及丰富的 C 语言库函数。本书主要讨论 8 位嵌入式单片机——51 单片机及其派生产品的 C 语言编程问题，简称 C51 的程序设计。

1.1.1　单片机的 C 语言的特点

单片机的 C 语言的特点主要体现在以下几个方面：

① 无须了解机器硬件及其指令系统，只需初步了解 MCS-51 的存储器结构；

② C51 语言能方便地管理内部寄存器的分配、不同存储器的寻址和数据类型等细节问题，但对硬件控制有限，而汇编语言可以完全控制硬件资源；

③ C51 语言在小应用程序中产生的代码量大，执行速度慢，但在较大的程序中代码效率高；

④ C51 语言程序由若干函数组成，具有良好的模块化结构，便于改进和扩充；

⑤ C51 语言程序具有良好的可读性和可维护性，而汇编语言在大应用程序开发中，开发难度增加，可读性差；

⑥ C51 语言有丰富的库函数，可以大大减少用户的编程量，显著缩短编程与调试时间，大大提高软件开发效率；

⑦ 使用汇编语言编制的程序，当机型改变时，无法直接移植使用，而C语言程序是面向用户的程序设计语言，能在不同类型的机器上运行，可移植性好。

1.1.2 单片机的 C 语言和标准 C 语言的比较

标准 C 语言，或称为 ANSI C 语言。单片机的 C 语言和标准 C 语言之间有许多相同的地方，但也有其自身的一些特点。不同的嵌入式 C 语言编译系统之所以与 ANSI C 语言有不同的地方，主要是由于它们所针对的硬件系统不同，对于 MCS-51 系列单片机，称为 C51 语言。C51 语言与标准 C 语言的不同点主要体现在以下几方面。

（1）库函数

标准 C 语言定义的库函数是按照通用微型计算机来定义的，而 C51 语言中的库函数是按 MCS-51 单片机的应用情况来定义的。

（2）数据类型

在 C51 语言中增加了几种针对 MCS-51 单片机的特有数据类型。例如，MCS-51 系列单片机包含位操作空间和丰富的位操作指令，因此，C51 语言与 ANSI C 语言相比多了一种位类型，从而使其能同汇编语言一样，灵活地进行位指令操作。

（3）变量的存储模式

C51 语言中变量的存储模式与 MCS-51 单片机的存储器紧密相关。从数据存储类型上，MCS-51 系列单片机有片内、片外程序存储器，片内、片外数据存储器。在片内程序存储器中，又有直接寻址区和间接寻址区之分，其分别对应 code、data、xdata、idata，以及根据 MCS-51 系列单片机特点而设定的 pdata 类型。使用不同存储器将会影响程序执行的效率，不同的模式对应不同的硬件系统和不同的编译结果。但 ANSI C 语言对存储模式要求不高。

（4）输入/输出

C51 语言中的输入/输出是通过 MCS-51 串行口来完成的，输入/输出指令执行前必须对串行口进行初始化。

（5）函数使用

C51 语言中有专门的中断函数，而标准 C 语言则没有。

1.1.3 单片机的 C 语言与汇编语言的优势对比

在国内，汇编语言在单片机开发过程中是比较流行的开发工具。长期以来对编译效率的偏见，以及不少程序员对使用汇编语言开发硬件环境的习惯性，使得 C 语言在很多地方遭到冷落。优秀的程序员写出的汇编语言程序的确有执行效率高的优点，但汇编语言其可移植性和可读性差的特点，使得使用其开发出来的产品在维护和功能升级时的确有极大的困难，从而导致整个系统的可靠性和可维护性比较差。而使用 C 语言进行嵌入式系统的开发，有着汇编语言不可比拟的优势。

（1）编程调试灵活方便

C 语言编程灵活，当前几乎所有的嵌入式系统都有相应的 C 语言级别的仿真调试系统，调试十分方便。

（2）生成的代码编译效率高

当前较好的C编译系统编译出来的代码效率只比直接使用汇编语言低 20%，如果使用优化编译选项甚至可以更低。

（3）模块化开发

目前的嵌入式系统的软硬件开发都向模块化、可复用性的目标集中。不管是硬件还是软件，都希望其有比较通用的接口，在以后的开发中如果需要实现相同或者相近的功能，就可以直接使用以前开发过的模块，尽量不做或少做改动，以减少重复劳动。如果使用 C 语言开发，那么数据交换可以方便

地通过约定实现，有利于多人协同进行大项目的合作开发。同时 C 语言的模块化开发方式使开发出来的程序模块可以不经修改而直接被其他项目使用，这样就可以很好地利用已有的大量 C 语言程序资源与丰富的库函数，从而最大程度地实现资源共享。

（4）可移植性好

由于不同系列嵌入式系统的 C 语言编译工具都是以标准 C 语言作为基础进行开发的，因此一种 C 语言环境下所编写的 C 语言程序，只需要将部分与硬件相关的地方和编译链接的参数进行适当修改，就可以方便地移植到另一种系列上。例如，在 C51 下编写的程序通过改写头文件及少量的程序行，就可以方便地移植到 196 或 PIC 系列上。也就是说，基于 C 语言环境下的嵌入式系统能基本实现平台的无关性。

（5）便于项目的维护

用 C 语言开发的代码便于开发小组计划项目、灵活管理、分工合作及后期维护，基本可以杜绝因开发人员变化而给项目进度、后期维护或升级所带来的影响，从而保证整个系统的品质、可靠性及可升级性。

下面通过一个例子对汇编语言和 C 语言进行比较。

【例 1-1】 将外部数据存储器的 000BH 和 000CH 单元的内容相互交换。

用汇编语言编程，源程序如下：

```
ORG     0000H
MOV     DPTR,#000BH
MOVX    A,@DPTR              ;将 000BH 内容读入 A
MOV     R7,A                 ;暂存 000BH 内容
INC     DPTR
MOVX    A,@DPTR              ;将 000CH 内容读入 A
MOV     DPTR,#000BH
MOVX    @DPTR,A
INC     DPTR
MOV     A,R7
MOVX    @DPTR,A
SJMP    $
END
```

用 C 语言编程，C51 源程序如下：

```
#include<absacc.h>
void main(void)
{   char c;
    c=XBYTE[11];
    XBYTE[11]=XBYTE[12];
    XBYTE[12]=c;
    while(1);
}
```

上面的程序经过编译，生成的反汇编程序如下：

```
0x0000  020013  LJMP    STARTUP1(C:0013)     ;跳转
0x0003  90000B  MOV     DPTR,#0x000B
0x0006  E0      MOVX    A,@DPTR
```

```
0x0007  FF      MOV     R7,A
0x0008  A3      INC     DPTR
0x0009  E0      MOVX    A,@DPTR
0x000A  90000B  MOV     DPTR,#0x000B
0x000D  F0      MOVX    @DPTR,A
0x000E  A3      INC     DPTR
0x000F  EF      MOV     A,R7
0x0010  F0      MOVX    @DPTR,A
0x0011  80FE    SJMP    C:0011
0x0013  787F    MOV     R0,#0x7F            ;以下是清零部分
0x0015  E4      CLR     A
0x0016  F6      MOV     @R0,A
0x0017  D8FD    DJNZ    R0,IDATALOOP(C:0016)
0x0019  758107  MOV     SP(0x81),#0x07
0x001C  020003  LJMP    main(C:0003)
```

对照 C 语言编写的源程序与反汇编程序，可以看出：

① 进入 C 语言程序后，首先将 RAM 地址从 7FH 开始的 128 个单元清零，然后置 SP 为 07，因此如果要对内部 RAM 置初值，那么一定是在执行了一条 C 语言语句之后；

② 对于 C 语言程序设定的变量，C51 编译器自行安排寄存器或存储器作为参数传递区，通常在 R0～R7（一组或两组，根据参数多少而定），因此如果对具体地址置数，则应避开 R0～R7 这些地址；

③ 如果不特别指定变量的存储类型，那么变量通常被安排在内部 RAM 区。

下面再给出几个 C 语言和汇编语言对照的例子。

【例 1-2】 二进制数转换成十进制数（BCD 码）。将累加器 A 中给定的二进制数，转换成 3 个十进制数（BCD 码），并存入 Result 开始的 3 个单元。

汇编语言源程序如下：

```
Result      EQU     20H
            ORG     0000H
            LJMP    START
            ORG     0030H
START:      MOV     SP,#60H             ;主程序
            MOV     A,#123
            LCALL   BINTOBCD
            SJMP    $
BINTOBCD:   MOV     B,#100              ;设置转换子程序
            DIV     AB
            MOV     Result,A            ;除以 100 得百位数
            MOV     A,B
            MOV     B,#10
            DIV     AB
            MOV     Result+1,A          ;余数除以 10 得十位数
            MOV     Result+2,B          ;余数为个位数
            RET
            END
```

调试结果：

片内 RAM 20H、21H、22H 中的数值分别为 01H、02H、03H。

用 C 语言编程，C51 源程序如下：

```
void  main(void)
{   unsigned  char  Result[3];
    unsigned  char  Number;
    Number=123;
    Result[0]=Number/100;            //除以 100 得百位数
    Result[1]=(Number%100)/10;       //余数除以 10 得十位数
    Result[2]=Number%10;             //余数为个位数
    while(1);                        //等待暂停
}
```

调试结果：

片内 RAM 07H 中的数据为 7BH，08H、09H、0AH 中的数据分别为 01H、02H、03H。

【例 1-3】 二进制数转换成 ASCII 码程序。将累加器 A 中的内容分为两个 ASCII 码，并存入 Result 开始的两个单元。

汇编语言源程序如下：

```
Result      EQU     20H
            ORG     0000H
            LJMP    START
            ORG     0030H
START:      MOV     SP,#40H
            MOV     A,#00011010B
            LCALL   BINTOHEX
            SJMP    $
BINTOHEX:   MOV     DPTR,#ASCIITAB
            MOV     B,A
            SWAP    A
            ANL     A,#0FH              ;取 A 的高 4 位
            MOVC    A,@A+DPTR
            MOV     Result,A
            MOV     A,B
            ANL     A,#0FH              ;取 A 的低 4 位
            MOVC    A,@A+DPTR
            MOV     Result+1,A
            RET
ASCIITAB:   DB      '0123456789ABCDEF'
            END
```

调试结果：

片内 RAM20H、21H 中的数据分别为 31H、41H。

用 C 语言编程，C51 源程序如下：

```
code unsigned char ASCIITAB[16]="0123456789ABCDEF";
void  main(void)
{   unsigned char Result[2];
    unsigned  char Number;
```

```
    Number=0x1a;
    Result[0]=ASCIITAB[Number/16];          //高 4 位
    Result[1]=ASCIITAB[Number&0x0f];        //低 4 位
    while(1);
}
```

调试结果：

片内 RAM07H 中的数据为 1AH，08H、09H 中的数据分别为 31H、41H。

1.2　C51 程序

1.2.1　C51 的程序结构

单片机 C51 语言继承了 C 语言的特点，其程序结构与一般 C 语言结构没有差别。C51 源程序文件扩展名为“.c”，如 test.c、function.c 等。每个 C51 源程序都包含一个名为“main( )”的主函数，C51 程序的执行总是从 main( )函数开始。当主函数所有语句执行完毕，则程序执行结束。例 1-4 和例 1-5 是两个典型的 C51 源程序的例子。例 1-4 是通过串口窗口 Srial#1 输出结果，例 1-5 是通过硬件电路输出结果。

【例 1-4】 C51 源程序参考示例。

```
#include<reg52.h>                   //预处理命令，reg52.h 是一个库文件
#include<stdio.h>                   //stdio.h 是一个库文件
void Function1(void);               //自定义函数 Function1 声明
unsigned int ch;                    //全局变量声明
void main(void)                     //主函数
{   SCON=0x50;                      //SCON:模式 1,8bit 异步串口通信
    TMOD=0x20;                      //TMOD:定时器 1 为模式 2,8bit 自动装载方式
    TH1=221;                        //TH1:1200bit/s 的装载值,16MHz
    TR1=1;                          //TR1:timer1 运行
    TI=1;                           //TI:设置为 1,以发送第一个字节
                                    //以上 5 条语句是为串口调试设置的
    while(ch<=5)
    {   Function1( );               //调用自定义函数
        printf("char=%d\n",ch);     //程序语句
    }
    while(1);
}
void Function1(void)                //自定义函数 Function1
{   unsigned char ps;               //自定义函数内部变量声明
    ps=1;
    ch=ch+ps;
}
```

调试结果：

在串口窗口 Srial#1 中，输出结果：

```
char=1
char=2
```

```
char=3
char=4
char=5
char=6
```

【例 1-5】 电路如图 1-1 所示，发光二极管 D1 经限流电阻接至 P1.0，编程使该灯以一定的时间间隔闪烁。

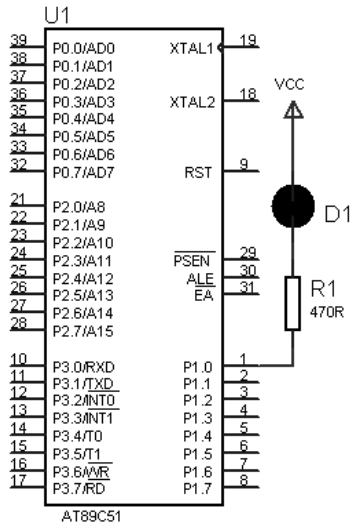

图 1-1　例 1-5 电路原理示意图

```
/*****************************
  ; 说明：这是一个学习 C51 的例程
  ; 功能：使 P1.0 口的 LED 按照设置的时间间隔闪烁
  ; 设计者：JZH
  ; 设计日期：2013 年 9 月 27 日
  ; 修改日期：2014 年 11 月 20 日
  ; 版本序号：V1.0.0
  ; *****************************/
   #include<reg51.h>                    //寄存器定义
  #include<stdio.h>                     //一般 I/O 口定义
  /***以下是全局变量定义*********/
  sbit LED=P1^0;                        //LED 灯连接在 P1.0 上
  int data i;                           //定义一个整型全局变量
  /*********主程序开始***************/
  void  main(void)
  {  while(1)
     {  LED=0;                          //LED 灯点亮
        for(i=0;i<1000;i++);            //延时
        LED=1;                          //LED 灯熄灭
        for(i=0;i<1000;i++);            //延时
      }
   }
```

从上面的例子可以看出，一个典型的 C51 源程序包含预处理命令、自定义函数声明、主函数 main()和自定义函数。这几部分与 C 语言的程序结构完全类似，各部分的功能如下。

① 预处理命令部分常用#include 命令来包含一些程序中用到的头文件。这些头文件中包含了一些库函数，以及其他函数的声明与定义。

② 自定义函数声明部分用来声明源程序中自定义的函数。

③ 主函数 main()是整个 C51 程序的入口。不论 main()函数位于程序代码中的哪个位置，C51 程序总是从 main()函数开始执行。

④ 自定义函数部分是 C51 源程序中用到的自定义函数的函数体。

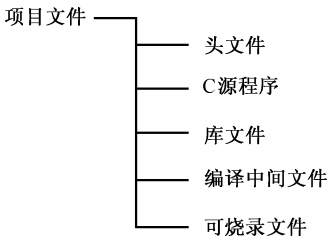

图 1-2　Keil μVision2 项目结构示意图

除了扩展名为“.c”的源程序文件外，C51 程序还支持扩展名为“.h”的头文件以及扩展名为“.lib”的库文件等。在一般的编译系统中，通常以项目（工程）结构来管理复杂的 C51 程序文件。例如 Keil μVision2 编译环境中，整个项目结构如图 1-2 所示。

在这里，整个项目由项目文件管理，项目文件扩展名为“.UV2”。整个工程项目中可以包含如下几类文件。

① 头文件用来包含一些库函数，系统变量声明并将不同的 C 文件连接起来。

② C 源文件是 C51 程序的主要部分，用来实现特定的功能。C 源文件可以有一个，也可以按照不同的功能分成多个，但所有这些 C 源文件中有且仅有一个可以包含一个 main()主函数。

③ 库文件是实现特定功能的函数库，供 C 源文件调用。

④ 编译中间文件是源程序在编译链接过程中生成的中间文件，其中包含了文件编译调试的信息。

⑤ 可烧录文件是编译系统生成的可以烧录到单片机内部供执行的文件，类似于“.exe”可执行文件。在 C51 语言中，一般扩展名为“.hex”或者“.bin”等。

在这些文件中，C 源文件是必需的，其他文件可以根据用户实际的需要而使用。

1.2.2　C51 编程规范及注意事项

在学习任何一种编程语言的时候，按照一定的规范培养良好的编程习惯很重要。良好的编程规范可以帮助开发人员理清思路、方便整理代码，同时也便于他人阅读、理解，以促进代码的交流。在进行 C51 语言程序设计时，应该注意以下几方面的编程规范。

1．注释

任何编程语言都支持注释语句。注释语句只对代码起到功能描述的作用，在实际的编程链接过程中不起作用。在 C51 语言中可以通过两种方式表示注释内容。

（1）用“//”开头来注释一行

例如“//变量声明”。本方法简明、方便，“//”符号可以在一行的开始，这样整行都表示注释内容；“//”也可以在某行的执行语句后面，“//”符号后面的内容是对该语句的注释内容。

（2）用“/*”符号开头，并以“*/”符号结束

例如“/*声明整型变量 ch*/”。本方法灵活多变，可以注释多行，如下：

```
/************************************************************/
/*       Function.c：使用 C51 编译器的自定义功能函数库       */
/************************************************************/
```

用户也可以在程序语句的内部进行注释，示例如下：

```
printf("ch=%d\n",/*整型变量 ch*/ch);
```

一个好的 C51 源程序应该添加必要的注释内容。这样，可以增加程序的可读性，方便日后修改或者与他人的代码交流。注释内容一般包括程序的功能、实现方式、自定义函数的功能描述、语句的功能等。

2．命名

在进行程序设计时，经常需要自定义一些函数或变量。一般来说，只要符合 C51 命名规范即可通过编译。但是，为了便于源程序的理解和交流，在进行命名时应注意以下几点：

① 自定义函数或者变量的名称最好能反映该函数或变量的功能用途。因此，需要选用有意义的单词或者字母组合来表示。例如 MAX 表示最大值、MIN 表示最小值等。

② 变量名通常加上表示数据类型的前缀，例如“ucSendData”的前缀“uc”表示 unsigned char。

③ 在命名时不要和系统保留的标志符以及关键字产生冲突或者歧义。

3．格式

为了程序阅读方便，在进行 C51 程序设计时，在程序结构以及语句书写格式方面应注意以下几点：

① 虽然 C51 语言对 main( )函数放置位置没有限定，但为了程序阅读的方便，最好把它放置在所有自定义函数的前面，即依次为头文件声明、自定义函数以及全局变量声明、main( )函数、自定义函数。

② C51 语句可以写在一行上也可以写在多行上。为了程序理解的方便，最好将每条语句单独写在一行中，并加以注释。有时某几条相连的语句或者共同执行某个功能则可以放置在一行中。

③ 对于源程序文件不同结构部分之间要留有空行。例如，头文件声明、自定义函数声明、main( )函数以及自定义函数之间均要空一行，来明显区分不同结构。

④ 对于 if、while 等块结构语句中的“{”和“}”要配对对齐，以便于程序阅读时能够理解该结构的起始和结束。

⑤ 源代码安排时可以通过适当的空格以及 Tab 键来实现代码对齐。

以上是一些常用的编程规范，读者可以参考借鉴。

1.2.3　C51 的标志符与关键字

标志符和关键字是一种编程语言最基本的组成部分，C51 语言同样支持自定义的标志符及系统保留的关键字。在进行 C51 程序设计时，需要了解标志符和关键字的使用规则。

1．标志符

标志符常用来声明某个对象的名称，如变量和常量的声明、数组和结构的声明、自定义函数的声明，以及数据类型的声明等。示例如下：

```
int count;
void Function1();
```

在上面的例子中，count 为整型变量的标志符，Function1 为自定义函数的标志符。

在 C51 语言中，标志符可以由字母、数字（0～9）或者下画线“_”组成，最多可支持 32 个字符。

C51 标志符的第一个字符必须是字母或者下画线“_”，例如 unt、ch_1 等都是正确的标志符，而 5count 则是错误的标志符。另外，C51 的标志符区分大小写，例如 count 和 COUNT 代表两个不同的标志符。使用标志符时应注意以下几点。

① 在命名 C51 标志符时，需要能够清楚地表达其功能含义，这样有助于阅读和理解源程序。

② C51 的标志符原则上可以使用下画线开头，但有些编译系统的专用标志符或者预定义项是以下画线开头的。为了程序的兼容性和可移植性，建议一般不使用下画线开头来命名标志符。

③ 尽量不要使用过长的标志符，以便于使用和程序理解方便。

④ 自定义的 C51 标志符不能使用 C51 语言保留的关键字，也不能和用户已使用的函数名或 C51 库函数同名。例如 char 是关键字，所以不能作为标志符使用。

2．关键字

关键字是 C51 语言的重要组成部分，是 C51 编译器已定义保留的专用特殊标志符，有时也称为保留字。这些关键字通常有固定的名称和功能，如 int、if、for、do、while、case 等。C51 语言中常用的关键字如表 1-1 所示。

表 1-1　C51 语言中常用的关键字

类　别	关 键 字	类　型	用 途 说 明
ANSI C 标准关键字	auto	存储种类说明	常用于声明局部变量，默认值为此类型
	break	程序语句	无条件退出循环程序最内层循环
	case	程序语句	switch 选择语句中的选择项
	char	数据类型说明	单字节整型数据或字符型数据
	const	存储类型说明	定义不可更改的常量值
	continue	程序语句	中断本次循环，并转向下一次循环
	default	程序语句	switch 选择语句中的默认选择项
	do	程序语句	用以构成 do…while 循环
	double	数据类型说明	声明双精度浮点型数据
	else	程序语句	用以构成 if…else 选择结构
	enum	数据类型说明	枚举
	extern	存储种类说明	在其他程序模块中说明了的全局变量
	float	数据类型说明	定义单精度浮点型数据
	for	程序语句	构成 for 循环语句
	goto	程序语句	构成 goto 转移语句
ANSI C 标准关键字	if	程序语句	用以构成 if…else 选择结构
	int	数据类型说明	声明基本整型数据
	long	数据类型说明	声明长整型数据
	register	存储种类说明	CPU 内部寄存的变量
	return	程序语句	用于返回函数的返回值
	short	数据类型说明	声明短整型数据
	signed	数据类型说明	声明有符号数，二进制数表示的最高位为符号位
	sizeof	运算符	计算表达式或数据类型的占有字节数
	static	存储种类说明	声明静态变量
	shruct	数据类型说明	声明结构类型数据
	switch	程序语句	构成 switch 选择语句
	typedef	数据类型说明	重新定义数据类型
	union	数据类型说明	声明联合数据类型

续表

类　别	关 键 字	类　型	用 途 说 明
ANSI C 标准关键字	unsigned	数据类型说明	声明无符号数据
	void	数据类型说明	声明无类型数据
	volatile	数据类型说明	该变量在程序执行中可被隐含地改变
	while	程序语句	用以构成 do…while 或 while 循环结构
C51 扩展关键字	bit	位变量声明	声明一个位变量以及位类型的函数
	sbit	位变量声明	声明一个可位寻址的变量
	sfr	特殊功能寄存器声明	声明一个 8 位的特殊功能寄存器
	shr16	特殊功能寄存器声明	声明一个 16 位的特殊功能寄存器
	data	存储器类型说明	直接寻址的单片机片内数据存储器
	bdata	存储器类型说明	可位寻址的单片机片内数据存储器
	idata	存储器类型说明	间接寻址的单片机片内数据存储器
	pdata	存储器类型说明	分页寻址的单片机片内数据存储器
	xdata	存储器类型说明	单片机片外数据存储器
	code	存储器类型说明	单片机程序存储器
	interrupt	中断函数说明	定义一个中断服务函数
	reentrant	再入函数说明	定义一个再入函数
	using	寄存器组定义	定义单片机的工作寄存器

从表 1-1 中可以看出，单片机 C51 程序语言不仅继承了 ANSI C 定义的 32 个关键字，还根据 C51 语言及单片机硬件的特点扩展了相应的关键字。在 C51 语言程序设计中，用户自定义的标志符不能和这些关键字冲突，否则就无法正确通过编译。

本 章 小 结

本章主要对单片机的 C 语言进行概述，使读者对单片机的 C 语言有一个初步的认识，包括两部分内容。

（1）C 语言与 51 单片机编程

单片机的编程语言主要有汇编语言和 C 语言，而单片机的 C 语言和标准的 C 语言又有一定的区别。本节主要介绍单片机的 C 语言的特点、单片机的 C 语言和标准 C 语言的比较、单片机的 C 语言和汇编语言的优势对比，目的是让读者认识到 C 语言是单片机编程的主要语言。

（2）C51 结构

C51 程序结构和标准 C 语言结构大同小异，二者差别较小。本节介绍 C51 的结构、编程规范、注意事项、标志符、关键词。读者可以发现，很多方面都和标准 C 语言是一样的。目的是让读者对 C51 有一个初步的认识。

通过本章的学习，应该掌握以下几个基本内容：

（1）单片机 C 语言与标准 C 语言的不同。

（2）C51 的程序结构。

（3）C51 的编程规范及注意事项。

习　题

1．简述单片机的C语言的特点。
2．单片机的C语言和标准C语言相比，有哪些不同点？
3．单片机的C语言和汇编语言相比，有哪些优势？
4．简述单片机的C语言程序的构成。