手把手教你学单片机的C语言程序设计(三)

C语言程序的基本结构

◆周兴华

C语言程序是由若干个函数单元组成的，每个函数都是完成某个特殊任务的子程序段，组成一个程序的若干个函数可以保存在一个源程序文件中，也可以保存在几个源程序文件中，最后再将它们连接在一起。C语言程序的扩展名为".c"，如"my-test.c"等。

实验一

在S3型试验板上实现：使LED1~LED8这8个二极管实现1、3、5、7及2、4、6、8的交替点亮，周期约1s。在我的文档中建立一个文件目录(cs1)，然后建立cs1.uv2的工程项目，最后建立源程序文件(cs1.c)。

输入下面的程序：

```
#include <REG51.H>// 序号(以下同):1
/*================2=========*/
void delay(void)          //3
{                         //4
    unsigned int i,j;     //5
    for(i=0;i<500;i++)    //6
    {                     //7
      for(j=0;j<120;j++)  //8
        {;}               //9
    }                     //10
}                         //11
//=================12=========
void ji_light(void)       //13
{                         //14
P0=0xaa;                  //15
}                         //16
//=================17=========
void ou_light(void)       //18
{                         //19
P0=0x55;                  //20
}                         //21
/*================22=========*/
void main(void)           //23
{                         //24
while(1)                  //25
    {                     //26
    ji_light();           //27
    delay();              //28
    ou_light();           //29
    delay();              //30
    }                     //31
}                         //32
```

编译通过后，将其烧录到89S51芯片中，将芯片插入到S3型LED输出试验板上，将P0口处的短路块全部拔出，S3实验板上接通5V稳压电源。发现LED1~LED8这8个二极管中的1、3、5、7及2、4、6、8的交替点亮，周期差不多为1s。图1为keil c51软件进行仿真时的界面。

下面先对程序进行分析，看看程序是怎样运行的，然后再介绍C语言程序的结构组成。

程序详释。

序号1(程序解释，以下同)：包含头文件REG51.H。

序号2：程序分隔或注释，在"/*"及"*/"之间的内容，程序不会去处理，因此通常可进行文字注释，能增加程序的可读性，当然也可作为程序语句模块之间的分隔。

序号3：定义函数名为delay的延时子函数。

序号4：delay延时子函数开始。

序号5：定义两个无符号整形变量i,j。

序号6~10：两个for语句循环体，作用是延时，由于我们还未学习for语句，因此这里可暂不理会。

序号11：delay的延时子函数结束。

序号12：程序分隔或注释，在"//"之后的内容，程序也不会去处理，因此也可进行文字注释，能增加程序的可读性，当然也能作为程序模块之间的分隔。但应注意，这种风格的注释，只对本行有效，所以在只需要一行注释的时候，往往采用"//……"这种格式。而"/*……*/"风格的注释，既可用于一行，也可用于多行。

序号13：定义函数名为ji_light的子函数，该子函数用于点亮1、3、5、7四个奇数号LED。

序号14：ji_light子函数开始。

序号15：向P0口送数AAH，目的是点亮1、3、5、7四个奇数号LED。

序号16：ji_light子函数结束。

序号17：程序分隔。

序号18：定义函数名为ou_light的子函数，该子函数用于点亮2、4、6、8四个偶数号LED。

序号19：ou_light子函数开始。

序号20：向P0口送数55H，目的是点亮2、4、6、8四个偶数号LED。

序号21：ou_light子函数结束。

序号22：程序分隔。

序号23：定义函数名为main的主函数

序号24：main的主函数开始。

序号25：while循环语句，这里进行无限循环。

序号26：while循环语句开始。

序号27：调用ji_light子函数模块。

序号28：调用延时子函数模块。

序号29：调用ou_light子函数模块。

序号30：调用延时子函数模块。

序号31：while循环语句结束。

序号32：main的主函数结束。

从上面我们可以看出，C语言程序的组成结构如下：

```
预处理命令          include<>
功能子函数1         delay()
                    {
                    函数体…
                    }
功能子函数2         ji_light()
                    {
                    函数体…
                    }
  ┆
功能子函数n         ou_light()
                    {
```

```
                函数体…
                }
主函数          main()
                {
                主函数体…
                }
```

结论：C 语言程序是由函数构成的，一个 C 源程序至少包括一个函数(主函数)，一个 C 源程序有且只有一个名为 main()的函数，也可能包含其它函数，因此，函数是 C 程序的基本单位。函数后面一定有一对大括号 " {……}"，在大括号里书写程序。C 语言程序总是从 main 主函数开始执行的，而不管物理位置上这个 main()放在什么地方。主函数通过直接书写语句和调用其它功能子函数来实现有关功能，这些功能子函数可以是由 C 语言本身提供给我们的库函数，也可以是用户自己编写的函数。那么库函数和用户自定义子函数有什么区别呢？简单地说，库函数就是针对一些经常使用的算法，经前人开发、归纳、整理形成的通用功能子函数集供大家使用。而自己编写的功能子函数则称用户自定义功能子函数，显然，用户自定义功能子函数是用户根据自己需要而编写的。可以看出，使用 C 语言开发产品，可以大量使用库函数而减少用户自己编写程序的工作量，这样，产品开发的速度和质量是汇编语言绝对不能相比的。Keil C51 内部有数百个库函数可供我们使用。调用 Keil C51 的库函数时只需要包含具有该函数说明的相应的头文件即可。

刚才我们谈到过，一个 C 源程序至少包括一个函数(主函数)，一个 C 源程序有且只有一个名为 main()的函数，那么我们将实验一的程序改动一下，只用主函数完成，而不用自定义功能子函数。

实验二

在 S3 型试验板上实现：使 LED1~LED8 这 8 个二极管实现 1、3、5、7 及 2、4、6、8 的交替点亮，周期约 1s。在我的文档中建立一个文件目录(cs2)，然后建立 cs2.uv2 的工程项目，最后建立源程序文件(cs2.c)。

输入下面的程序：

```
#include <REG51.H>// 序号(以下同):1
/*==================2===*/
void main(void)                 //3
{    unsigned int i,j;          //4
while(1)                        //5
     {                          //6
     P0=0xaa;                   //7
//=================8
     for(i=0;i<500;i++)         //9
     {                          //10
          for(j=0;j<120;j++)    //11
          {;}                   //12
     }                          //13
     //=================14
     P0=0x55;                   //15
     //=================16
     for(i=0;i<500;i++)         //17
     {                          //18
          for(j=0;j<120;j++)    //19
          {;}                   //20
     }                          //21
     //=================22
     }                          //23
}                               //24
```

将其编译通过，将其烧录到 89S51 芯片中，将芯片插入到 S3 型 LED 输出试验板上，将 P0 口处的短路块全部拔出，S3 实验板上接通 5V 稳压电源。发现试验效果与实验一完全相同。但是我们能看出，此程序的结构条理没有实验一的程序来得清晰，一大堆实现各种功能的语句全部放在主函数内，显得有点乱。因此一个设计合理的 C 语言程序，不仅语句要流畅，而且结构也要简洁明了。

我们把程序解释一下。

序号 1（程序解释，以下同）：包含头文件 REG51.H。

序号 2：程序分隔。

序号 3：定义函数名为 main 的主函数。

序号 4：主函数开始。定义两个无符号整形变量 i,j。

序号 5：while 循环语句，进行无限循环。

序号 6：while 循环语句开始。

序号 7：向 P0 口送数 AAH，目的是点亮 1、3、5、7 四个奇数号 LED。

序号 8：程序语句分隔。

序号 9~13：两个 for 语句循环体，作用是延时。

序号 14：程序语句分隔。

序号 15：向 P0 口送数 55H，目的是点亮 2、4、6、8 四个偶数号 LED。

序号 16：程序语句分隔。

序号 17~21：两个 for 语句循环体，作用是延时。

序号 22：程序语句分隔。

序号 23：while 循环语句结束。

序号 24：main 的主函数结束。

我们还没有解释过预处理命令中的包含头文件 REG51.H 即 include <REG51.H>。

包含头文件即为文件包含处理。

所谓“文件包含”是指一个文件将另外一个文件的内容全部包含进来，所以这里的预处理命令虽然只有一行，但 C 编译器在处理的时候却要处理几十乃至几百行。包含头文件 REG51.H 的目的是为了要使用 P0 这个符号，即通知 C 编译器，程序中所写的 P0 是指 80C51 单片机的 P0 端口而不是其它变量。

用记事本打开 C:\Keil\C51\Inc\Reg51.h(打开时的文件类型改为所有文件)可以看到这样的一些内容：

```
/*--------------------------
REG51.H
Header file for generic 80C51 and 80C31
microcontroller.
Copyright (c) 1988-2002 Keil Elektronik
GmbH and Keil Software, Inc.
All rights reserved.
--------------------------*/
#ifndef __REG51_H__
#define __REG51_H__
/* BYTE Register */
sfr P0   = 0x80;
sfr P1   = 0x90;
sfr P2   = 0xA0;
sfr P3   = 0xB0;
sfr PSW  = 0xD0;
sfr ACC  = 0xE0;
sfr B    = 0xF0;
sfr SP   = 0x81;
sfr DPL  = 0x82;
sfr DPH  = 0x83;
sfr PCON = 0x87;
sfr TCON = 0x88;
sfr TMOD = 0x89;
sfr TL0  = 0x8A;
sfr TL1  = 0x8B;
sfr TH0  = 0x8C;
```

```
sfr TH1  = 0x8D;
sfr IE   = 0xA8;
sfr IP   = 0xB8;
sfr SCON = 0x98;
sfr SBUF = 0x99;
/* BIT Register */
/* PSW */
sbit CY   = 0xD7;
sbit AC   = 0xD6;
sbit F0   = 0xD5;
sbit RS1  = 0xD4;
sbit RS0  = 0xD3;
sbit OV   = 0xD2;
sbit P    = 0xD0;
/* TCON */
sbit TF1  = 0x8F;
sbit TR1  = 0x8E;
sbit TF0  = 0x8D;
sbit TR0  = 0x8C;
sbit IE1  = 0x8B;
sbit IT1  = 0x8A;
sbit IE0  = 0x89;
sbit IT0  = 0x88;
/* IE */
sbit EA   = 0xAF;
sbit ES   = 0xAC;
sbit ET1  = 0xAB;
sbit EX1  = 0xAA;
sbit ET0  = 0xA9;
sbit EX0  = 0xA8;
/* IP */
sbit PS   = 0xBC;
sbit PT1  = 0xBB;
sbit PX1  = 0xBA;
sbit PT0  = 0xB9;
sbit PX0  = 0xB8;
/* P3 */
sbit RD   = 0xB7;
sbit WR   = 0xB6;
sbit T1   = 0xB5;
sbit T0   = 0xB4;
sbit INT1 = 0xB3;
sbit INT0 = 0xB2;
sbit TXD  = 0xB1;
sbit RXD  = 0xB0;
/* SCON */
sbit SM0  = 0x9F;
sbit SM1  = 0x9E;
sbit SM2  = 0x9D;
sbit REN  = 0x9C;
sbit TB8  = 0x9B;
sbit RB8  = 0x9A;
sbit TI   = 0x99;
sbit RI   = 0x98;
#endif
```

因为之前我们都已学过了《手把手教你学单片机》讲座，对80C51内部结构比较熟悉，因此从中可看出，这里都是一些符号的定义，即规定符号名与地址的对应关系。其中有“sfr P0 = 0x80;”一行，即定义P0口与地址0x80对应（0x80是C语言中十六进制数的写法，相当于汇编语言中的80H）。

美国国家标准化协会（ANSI）制订的C语言标准（ANSI C）中规定，函数必须要“先说明，后调用”，我们在实验一的程序中，是先定义（说明）了几个功能子函数，然后在主函数中进行调用，这样当然没问题。如果将顺序倒一下，将几个功能子函数放在主函数的后面，那么编译时就会出错。这时我们需要进行“先说明，后调用”。下面就是“先说明，后调用”的例子。

```
#include <REG51.H>// 序号（以下同）:1
//=========================2
void delay(void);                    //3
void ji_light(void);                 //4
void ou_light(void);                 //5
//=========================6
void main(void)                      //7
{                                    //8
while(1)                             //9
    {                                //10
    ji_light();                      //11
    delay();                         //12
    ou_light();                      //13
    delay();                         //14
    }                                //15
}                                    //16
/*====================17==*/
void delay(void)                     //18
{                                    //19
    unsigned int i,j;                //20
    for(i=0;i<500;i++)               //21
    {                                //22
        for(j=0;j<120;j++)           //23
        {;}                          //24
    }                                //25
}                                    //26
//=========================27
void ji_light(void)                  //28
{                                    //29
P0=0xaa;                             //30
}                                    //31
//=========================32==
void ou_light(void)                  //33
{                                    //34
P0=0x55;                             //35
}                                    //36
```

以上的程序中，序号3~5为自定义功能子函数的说明，其它可参考实验一或实验二的程序解释，由于符合ANSI C规定的“先说明，后调用”原则，因此编译时通过。所以，一个好的习惯是，不管自定义的功能子函数放在什么位置，在程序的开始处总是先进行说明。

下面是通用的C语言程序组成结构（主函数可放在功能子函数说明之后的任意位置）：

```
预处理命令          include<>
功能子函数1说明
功能子函数2说明
┆
功能子函数n说明

功能子函数1        delay()
                   {
                   函数体…
                   }
主函数             main()
                   {
                   主函数体…
                   }
功能子函数2        ji_light()
                   {
                   函数体…
                   }
┆
功能子函数n        ou_light()
                   {
                   函数体…
                   }
```