学习任务

四

循环结构程序设计——输出九九表

现实生活中有很多需要重复做的事情，大多数人的今天就是在重复昨天的生活，只是时间、地点等有所不同。 要重复做的事情就是循环，在程序设计中，用到循环控制的地方有很多。 如要输入全班学生的成绩，每个成绩的输入就是重复操作，也就是循环。 又如求 1～100 之和也是重复地做加法运算的操作，几乎所有的实用程序都包含循环环节。 本学习任务将学习循环结构的程序设计。

学习目标

任务说明

学习循环结构程序设计的三种结构以及循环结构在程序设计中的运用。

知识和能力要求

知识要求

（1）掌握 while 语句的结构。
（2）掌握 do…while 语句的结构。
（3）掌握 for 语句的结构。
（4）掌握 continue 和 break 语句的用法。
（5）理解循环嵌套的知识。

能力要求

（1）能读懂循环结构程序流程图。
（2）能依据循环流程框图写出程序代码。
（3）能够分析循环程序的走向进而排查程序中的逻辑错误。
（4）能够为程序设计测试数据。
（5）能够与他人配合共同完成循环结构的程序设计和测试。

任务准备

一、while 循环结构

1. 语句形式

```
while(表达式)
    {语句}
```

2. 执行顺序

while 循环结构是一种当型循环结构，其执行过程是：先计算表达式的值，当表达式的值为真时执行循环体中的语句，然后反复执行，每次执行都会判断表达式的值是否为真，表达式的值为假时结束循环，接着执行循环体下面的语句。若循环体中的语句只有一条，则一对花括号可以省略。

3. 流程框图

图 4-1 和图 4-2 所示为 while 循环结构的程序流程图和 N-S 图。

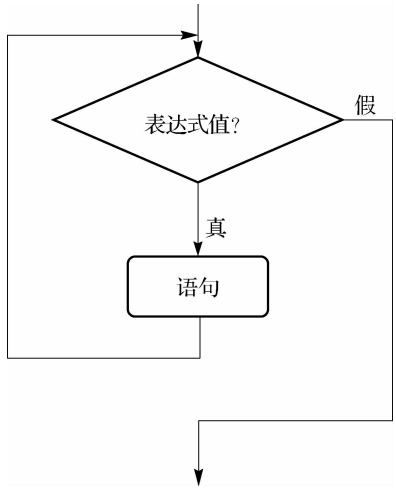

图 4-1　while 循环结构的程序流程图

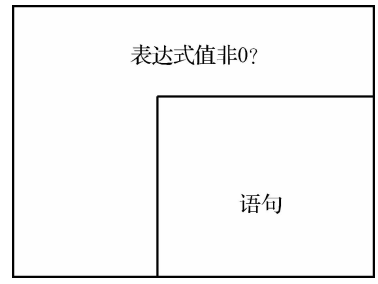

图 4-2　while 循环结构的 N-S 图

应用举例 4-1　计算 1～100 的自然数之和。

(1)分析。

先定义一个整型变量 i,初值是 1,再定义一个存放“和”的变量 s,初值为 0。s 的作用好比存钱罐,用前先清空。i 要从 1 变化到 100,每次变化之前将 i 值累加到 s 中,然后 i 增 1,这样的累加及增 1 的操作要反复 100 次,最后的 s 就是所求的值。

(2)绘制程序框图。

依据以上的分析绘制出程序流程 N-S 图,顺序结构用依次排列的长条格表示;循环结构是大方格中套小方格,小方格上面表示的是循环条件,小方格中的一条条内容是循环体语句。

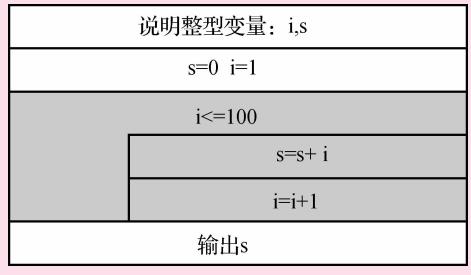

图 4-3　1～100 自然数之和的 N-S 图

(3)编写程序代码。

```
#include "stdio.h"
void main()
{
    int i,s;
    i=1;
    s=0;
    while(i<=100)
```

```
    {
        s+=i;                    //就是 s=s+i;
        i++;                     //就是 i=i+1;
    }
    printf("s= %d\n",s);
    getchar();
}
```

输出:s=5050

问题 4-1 使用当型循环结构编写程序计算 10!的值,即 1～10 的自然数之积。

(1)绘制程序框图。

(2)编写程序代码。

二、do…while 循环结构

1. 语句形式

```
do{
    语句
} while(表达式);                    //此处必有分号
```

2. 执行顺序

do…while 循环结构是直到型循环,其执行过程是:先执行循环体,然后计算表达式的值,

若表达式的值为真，则执行循环体中的语句，再计算表达式的值，若表达式的值为真，则继续执行循环体中的语句，如此反复，直到表达式的值为假，结束循环，接着执行循环下面的语句。

3. 流程框图

图 4-4 和图 4-5 所示为 do…while 循环结构的程序流程图和 N-S 图。

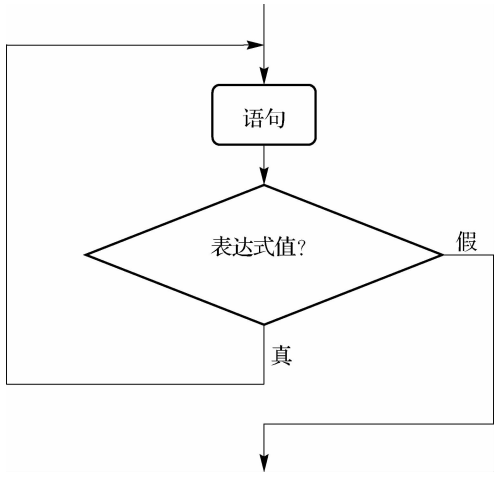

图 4-4　do…while 循环结构的程序流程图

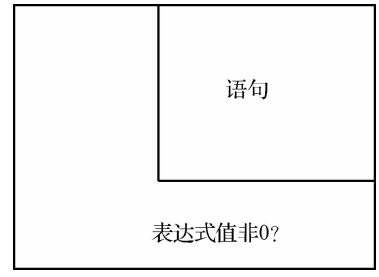

图 4-5　do…while 循环结构的 N-S 图

应用举例 4-2　用 do…while 循环计算 1～100 的自然数之和。

(1)绘制程序流程图。

依据应用举例 4-1 中的分析，绘制出程序流程图，如图 4-6 所示。

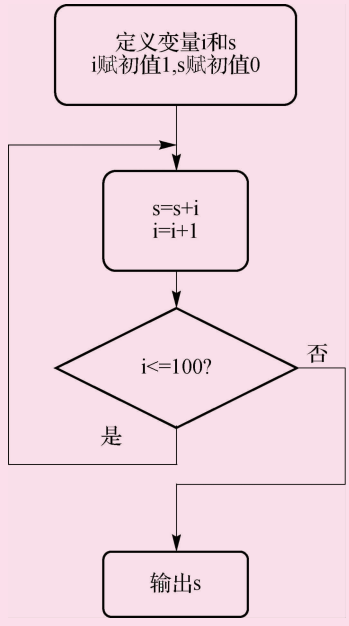

图 4-6　应用举例 4-2 的程序流程图

(2)编写程序代码。

```
#include "stdio.h"
void main()
{    int i=1,s=0;
     do
     {
          s=s+i;
          i++;
     }while(i<=100);
     printf("s= %d\n",s);
     getchar();
}
```

输出:s=5050

问题 4-2 使用直到型循环结构编写程序,计算 1～100 的所有偶数之和。

(1)绘制程序框图。

(2)编写程序代码。

三、for 循环结构

1. 语句形式

```
for(表达式 1;表达式 2;表达式 3)
    {语句}
```

2. 执行顺序

for 循环结构的执行顺序可用图 4-7 所示的程序流程图表示。

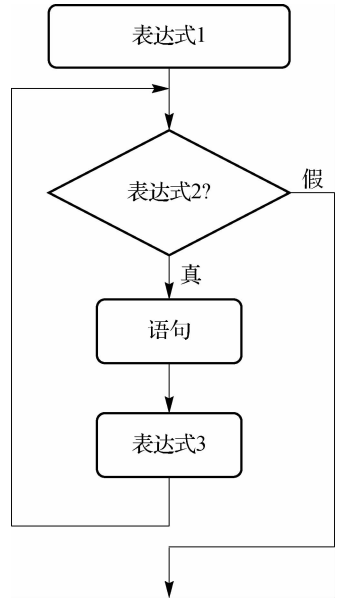

图 4-7　for 循环结构程序流程图

先计算表达式 1，然后计算表达式 2，若表达式 2 的值为真，则执行语句，然后执行表达式 3，然后再执行表达式 2，若表达式 2 的值为真，再次执行语句和表达式 3，然后再计算表达式 2，如此反复，直到表达式 2 的值为假，退出循环，执行循环下面的语句。

3. 提示

for 循环结构书写灵活，括号中是两个分号断开三个表达式，三个表达式均可以省略，但两个分号不能省略。省略表达式 1，可将相应部分移到 for 循环之上，而表达式 1 采用逗号表达式，可减少初始值的设置语句；省略表达式 2，则认为表达式 2 的值为真；省略表达式 3，可将表达式 3 移到循环体中，也可将循环体中的语句通过逗号表达式并在表达式 3 中。表达式 1 和表达式 3 可以是任何有效的表达式。

应用举例 4-3　用 for 循环计算 1～100 的自然数之和。

```
#include "stdio.h"
void main()
{
    int i,s=0;
```

```
    for(i=1;i<=100;i++)
        s=s+i;                    //循环体只有一条语句,故省略了一对花括号
    printf("s= %d\n",s);
}
```

输出:s=5050

老师 不论哪种循环,循环中的语句可以是一条,也可以是用一对花括号括起来的若干条语句,甚至可以是一个循环语句。

问题 4-3 **使用 for 循环结构编写程序,计算 1～100 的所有奇数之和。**

(1)绘制程序框图。

(2)编写程序代码。

四、循环嵌套

如果循环体中的某条语句是循环语句，那么这就是循环的嵌套。初学程序设计的同学写程序比较困难，特别是循环程序，下面将通过若干连续变化的例子说明循环并不复杂。

应用举例 4-4 循环嵌套。

(1)下面这段程序是在屏幕上显示一个“*”，其核心语句就一条。

```
#include "stdio.h"
void main()
{
    printf("*");                    //核心语句
    getchar();
}
```

输出：*

(2)将上述程序中的核心语句写进一个固定循环次数的循环中，就可以将输出一个“*”的事情重复若干遍，这就是循环，下面这段程序是输出 10 个“*”。

```
#include "stdio.h"
void main()
{
    int i;
    for(i=1;i<=10;i++)
        printf("*");
    getchar();
}
```

输出：**********

(3)修改上面这段程序，使输出“*”的个数由输入决定。我们需要再定义一个变量 n，用于存放输入的一个整数，这个循环的次数由输入的数决定。

```
#include "stdio.h"
void main()
{
    int i,n;
    printf("请输入一个正整数:");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        printf("*");
    getchar();      //两条“getchar();”语句是为了运行可执行文件时看到结果
    getchar();
}
```

输入 5 时的输出：*****

(4)将前面第二个例子中输出10个“*”的语句“for(i=1;i<=10;i++)printf("*");”看成一个整体,当成一条语句,再写一个换行语句,将这两条语句作为一个循环体,放在循环中,这就是循环的嵌套。下面这段程序就是将输出10个“*”并换行这件事做五遍。

```
#include "stdio.h"
void main()
{
    int i,j;                        //使用两个变量,分别控制内外层循环次数
    for(j=1;j<=5;j++)
    {
        for(i=1;i<=10;i++) //循环体中的循环语句
            printf("*");
        printf("\n");              //循环体中的换行语句
    }
    getchar();
}
```

输出:

```
**********
**********
**********
**********
**********
```

(5)修改第四个例子,使内层循环的次数由外层的循环控制变量决定。

```
#include "stdio.h"
void main()
{
    int i,j;
    for(j=1;j<=5;j++)
    {
        for(i=1;i<=2*j;i++)
            printf("*");
        printf("\n");
    }
    getchar();
}
```

输出:

```
**
****
******
```

```
********
**********
```

问题 4-4 使用循环嵌套编写程序使其输出如下图形。

```
*
**
***
****
*****
******
```

五、break 和 continue 语句

break 语句常用在 switch 语句和循环中，其作用是跳出本层循环或本 switch 语句。continue 语句只用在循环中，作用是结束本次循环直接进入下次循环。图 4-8 至图 4-13 所示为遇到 continue 和 break 语句后程序的走向。

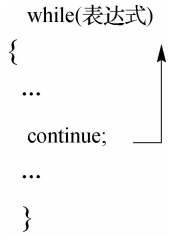

图 4-8 while 语句中的 continue

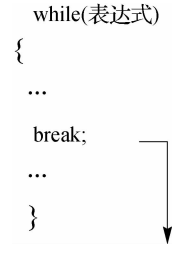

图 4-9 while 语句中的 break

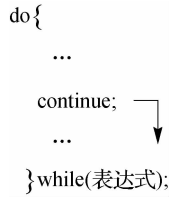

图 4-10　do…while 语句中的 continue

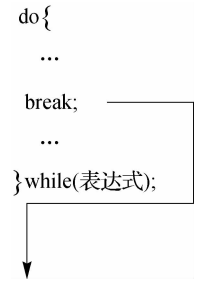

图 4-11　do…while 语句中的 break

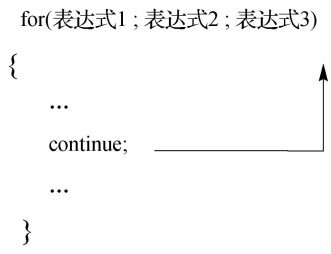

图 4-12　for 语句中的 continue

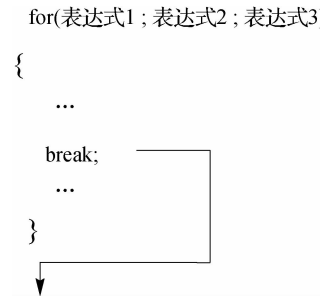

图 4-13　for 语句中的 break

老师　continue 和 break 一般出现在循环体中的 if 语句中。

应用举例 4-5　分析以下程序代码的功能。

(1)程序代码。

```
#include "stdio.h"
void main()
{
    int i;
    for(i=100;i>0;i--)              //从 100 开始向下搜
        if(i%13==0)                 //循环体只有一条 if 语句,省略了花括号
            break;
    printf("%d\n",i);
    getchar();
}
```

(2)解读程序。

通过阅读程序和分析程序的功能进而学习其中的设计技巧,这些都是程序设计人员所必备的基本技能。通过回答以下几个问题来分析程序的功能。

①程序预期的循环次数是多少? 100 次。

②程序中是否存在强制退出循环的语句? 存在,就是 break 语句。

③循环体有几条语句? 一条 if 语句。

④程序中有几条输出语句？一条，是出循环后的输出。

填写下表，分析循环走向。

循环执行次序	i 值	i%13 的值	表达式 i%13==0 值	是否执行 break 语句
第 1 次	100	9	0	不执行
第 2 次	99	8	0	不执行
第 3 次	98	7	0	不执行
第 4 次	97	6	0	不执行
第 5 次	96	5	0	不执行
第 6 次	95	4	0	不执行
第 7 次	94	3	0	不执行
第 8 次	93	2	0	不执行
第 9 次	92	1	0	不执行
第 10 次	91	0	1	执行

(3)程序功能。

通过分析得出结论：这个程序的功能是找出 100 以内能被 13 整除的最大数。

(4)模仿。

依据以上的例子，分析下面程序代码的功能。

```
#include "stdio.h"
void main()
{
    int i;
    for(i=100;i>0;i--)
        {if(i%13!=0)
            continue;
         printf("%3d",i);
        }
    getchar();
}
```

①程序预期的循环次数是多少？.....

②遇到 continue 语句程序转向哪里？.....

③循环体有几条语句？.....

④程序中有几个输出，是一个还是若干个？.....

填写下表，分析循环走向。

循环次序	i 值	i%13 的值	i%13==0 值	是否执行 continue 语句	输出
第 1 次					
第 2～9 次					
第 10 次					
第 11～22 次					
第 23 次					
第 24～35 次					
第 36 次					
……	……	……	……	……	……

续填表格至第 36 次，得出程序的功能是_____

问题 4-5 补充程序，实现输出 1～50 中能被 7 整除的数的功能，并画出流程框图。

```
#include "stdio.h"
void main( )
{
    int i;
    for (i=1;i<=50;i++)
        {if (i%7!=0)
        ____________;
        printf("%d\n";i);
        }
}
```

绘制流程框图。

任务实施

任务一 累加问题的程序设计

工作内容及要求

已知 π/4 的近似值为 1－1/3＋1/5－1/7＋1/9－…1/n，编写一个程序实现以下功能：依据输入的计算精度求 π。

编程训练 4-1-1　计算 1～n 的自然数之和。

(1)分析。

模仿应用举例 4-3，完成 1～n 的自然数之和的程序设计。

(2)编写程序代码。

```
#include "stdio.h"
void main()
{
    int i,s=0,n;
    printf("请输入一个正整数:");

    ..........                          //输入一个正整数赋给 n
    for(i=1;i<=      ;i++)
                                        //循环体语句,计算累加和
    printf("s= %d\n",s);
}
```

(3)调试。

在 VC 6.0 环境中调试程序，将预期值和实际输出值填写在下表中。

次数	输入数据	期望输出值	实际输出值
1	5		
2	10		
3	100		

编程训练 4-1-2　计算 1～n 的倒数之和。

(1)分析。

由于是倒数之和，存放和的变量 s 应为 double 类型，输出格式为%f。

(2)编写程序代码。

```
#include "stdio.h"
void main()
{
    int i,n;
    ..........                          //定义 s 为 double 类型,并赋初值 0
```

```
    printf("请输入一个正整数:");
    ..........                    //输入一个正整数赋给 n
    for(i=1;i<=.....;i++)
    ..........                    //循环体语句,计算累加和
    printf("s=%.....\n",s);
}
```

(3)调试。

在 VC 6.0 环境中调试程序,将预期值和实际输出值填写在下表中。

次数	输入数据	期望输出值	实际输出值
1	2		
2	3		
3	5		

编程训练 4-1-3　计算 1～*n* 的奇数的倒数和。

(1)分析。

将编程训练 4-1-2 中的语句“for(i=1;i<=n;i++)”改为“for(i=1;i<=n;i=i+2)”,i 的变化顺序是 1,3,5,…,*n*。

(2)编写程序代码。

```
..........                        //文件包含
..........                        //主函数
{
    ..........                    //定义 i,n 为整型变量
    ..........                    //定义 s 为 double 类型,并赋初值
    ..........                    //提示输入
    ..........                    //输入一个正整数赋给 n
    ..........                    //for 循环
        ..........                //循环体语句,计算累加和
    ..........                    //输出 s
}
```

(3)调试。

在 VC 6.0 环境中调试程序,将预期值和实际输出值填写在下表中。

次数	输入数据	期望输出值	实际输出值
1	2		
2	3		
3	5		
4	10		

编程训练 4-1-4　正负相间问题：计算 1－1/3＋1/5－1/7＋1/9－…1/*n* 的值。

(1)分析。

将编程训练 4-1-3 中的循环体“for(i＝1;i＜＝n;i＝i＋2) s＝s＋1.0/i”改为“s＝s＋f＊1.0/i”，而 f 在＋1 和－1 之间交替变化，即在循环体中增加一条语句“f＝－1＊f;”，如果赋给 n 一个较大值，并将累加和的 4 倍输出，这就是 π 的近似值。

(2)编写程序代码。

```
.....                                   //文件包含
.....                                   //主函数
{
    .....                               //定义 i,n,f 为整型变量，为 f 赋初值 1
    .....                               //定义 s 为 double 类型，并赋初值
    .....                               //提示输入
    .....                               //输入一个正整数赋给 n
    for(i＝1;i＜＝n;i＝i＋2)             //for 循环
        {.....                          //循环体语句，计算累加和
        .....                           //更新 f 值
        }
    .....                               //输出 s 和 4＊s
}
```

(3)调试。

在 VC 6.0 环境中调试程序，在组员的监督下，将输出值填在下表中。

次数	输入数据	1/n 的值	s 输出值	4＊s 输出值(π)
1	100			
2	10 000			
3	50 0000			
4	1000 000			
填表人：			监督人：	

编程训练 4-1-5　根据 π/4≈1－1/3＋1/5－1/7＋1/9－…1/*n* 求 π，计算精度由输入获得。

(1)分析。

将编程训练 4-1-4 中的循环体“for(i＝1;i＜＝n;i＝i＋2)”改为“for(i＝1;1.0/i＞＝e;i＝i＋2)”，当 n＝100 时，e 为 0.01；当 n 为 10 000 时，e 为 0.000 1。输入较大的 n 值与输入较小的 e 值相对应，并将累加和的 4 倍输出，这就是 π 的近似值。

(2)编写程序代码。

```
.....                                   //文件包含
.....                                   //主函数
{
    .....                               //定义 i,f 为整型变量,并为 f 赋初值 1
    .....                               //定义 s,e 为 double 类型,并为 s 赋初值
    .....                               //提示输入计算精度
    .....                               //输入一个计算精度赋给 e
    .....                               //for 循环
        {.....                          //循环体语句,计算累加和
        .....                           //更新 f
        }
    .....                               //输出 4 * s
}
```

(3)调试。

在 VC 6.0 环境中调试程序,在组员的监督下,将输出值填在下表中。

次数	输入数据 e	输出值 4 * s(π)
1	0.01	
2	0.000 1	
3	0.000 002	
4	0.000 001	
填表人:		监督人:

老师 对比编程训练 4-1-4 与编程训练 4-1-5 的结果。

编程训练 4-1-6　程序改错。

为了能够接收多次输入,某学生针对编程训练 4-1-5 编写了以下程序代码,其输出结果如图 4-14 所示,为什么结果不对? 请查找原因。

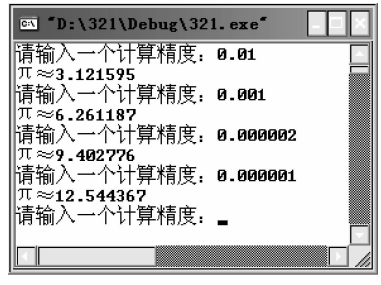

图 4-14　编程训练 4-1-6 图

```
#include "stdio.h"
void main()
{
    int i,f=1;
    double s=0,e;
    while(1)
    {
        printf("请输入一个计算精度:");
        scanf("%lf",&e);
        for(i=1;1.0/i>=e;i=i+2)
        {
            s=s+f*1.0/i;
            f=-f;
        }
        printf("π≈%f\n",4*s);
    }
}
```

错误原因是：

应改为：

学生 程序设计好玄妙啊！接下来该学九九表了吧？

任务二 输出九九表

工作内容及要求

编写一个程序产生一个可执行文件，如图 4-15 所示，当运行这个文件时，显示九九乘法表，如图 4-16 所示。

图 4-15 九九乘法表可执行文件

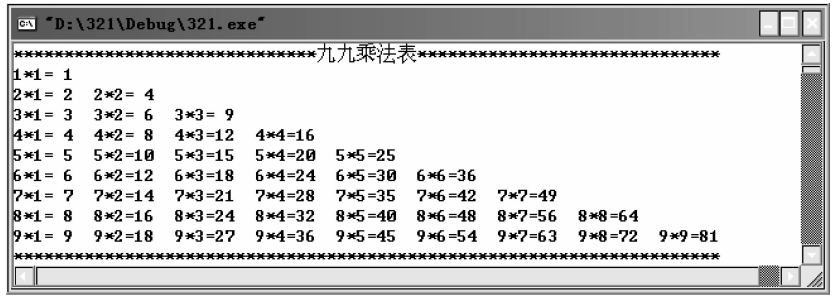

图 4-16 九九乘法表

(1)任务提示。

观察分析下面这段一层循环程序,程序输出为

1＊1＝1　1＊2＝2　1＊3＝3　…　1＊9＝9

```
void main()
{
    int j;
    for(j=1;j<=9;j++)
        printf("1＊ %d= %2d  ",j,1＊j);          /＊若构成九九表,此处的1是1～9变化的＊/
}
```

将上段程序的核心部分“for(j=1;j<=9;j++)　printf("1＊%d=%2d ",j,1＊j);”看做一个整体,考虑是某数乘1～9,这个某数要从1到9变化,将语句“printf("1＊%d=%2d ",j,1＊j);”中的“1”改为变量i,即“printf("%d＊%d=%2d ", i,j,i＊j);”,i在1～9变化,故再来一层循环,设循环变量为i,仍是1～9变化,考虑到变量j每循环一次要换一行,因此在内层循环中再增加一条换行语句,并将它们用一对花括号括起来。

(2)方形的九九表程序代码。

```
.....                                   //文件包含
.....                                   //主函数
{
    .....                               //定义i,j为整型变量
    .....                               //外层循环i,从1到9变化
    .....                               //内层循环j,从1到9变化
    {
        .....                           //输出一行
        .....                           //换行
    }
}
```

(3)输出下三角形的九九表。

```
.....                                   //文件包含
.....                                   //主函数
{
    .....                               //定义i,j为整型变量
    .....                               //外层循环i,从1到9变化
```

```
        .....                                   //内层循环 j,从 1 到 i 变化
        {
            .....                               //输出一行
            .....                               //换行
        }
    }
```

(4)为输出增加修饰。

```
    .....                                       //文件包含
    .....                                       //主函数
    {
        .....                                   //定义 i,j 为整型变量
        .....                                   //输出“****九九表 ****”
        .....                                   //外层循环 i,从 1 到 9 变化
        .....                                   //内层循环 j,从 1 到 i 变化
            {
            .....                               //输出一行
            .....                               //换行
            }
            .....                               //输出“**************”
        getchar();
    }
```

(5)调试。

某学生在反复调试程序时,编译通过,运行时出现如图 4-17 所示的错误提示,根据屏幕内容找出错误原因。

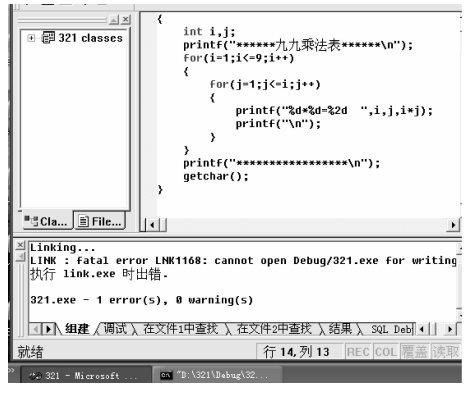

图 4-17 错误提示

错误原因是：

应改为：

老师 在组员的监督下运行可执行文件，输出九九表。

评价与考核

课程名称：C 语言学习与应用		授课地点：		
学习任务四：循环结构程序设计——输出九九表		授课教师：	授课学时：	
课程性质：理实一体课程		综合评分：		
知识掌握情况评分（35 分）				
序号	知识考核点	教师评价	配分	得分
1	while 语句		5	
2	do…while 语句		5	
3	for 语句		10	
4	break 和 continue 语句的用法		5	
5	循环嵌套的知识		10	
工作任务完成情况评分（65 分）				
序号	能力操作考核点	教师评价	配分	得分
1	能编写累加和的程序		10	
2	能在两种循环间变通		15	
3	会运用技巧解决正负相间等问题		15	
4	物理和逻辑错误的排查能力		15	
5	与同组成员的协作能力		10	
违纪扣分（20 分）				
序号	违纪考查点	教师评价	配分	得分
1	课上吃东西		5	
2	课上打游戏		5	
3	课上打电话		5	
4	其他扰乱课堂秩序的行为		5	

任务测试模拟试卷

一、选择题(每小题 1 分,共 20 分)

1. C 语言中 while 和 do…while 循环的主要区别是(　　)。

A. do…while 的循环体至少无条件执行一次

B. while 的循环控制条件比 do…while 的循环控制条件严格

C. do…while 允许从外部转到循环体内

D. do…while 的循环体不能是复合语句

2. 以下程序段中,while 循环执行的次数是(　　)。

```
int k=0;
while(k=1)  k++;
```

A. 无限次　　B. 有语法错,不能执行

C. 一次也不执行　　D. 执行一次

3. 以下程序的执行结果是(　　)。

```
main()
{
    int i,sum;
    for(i=1;i<=3;sum++) sum+=i;
    printf("%d\n",sum);
}
```

A. 6　　B. 3　　C. 死循环　　D. 0

4. 执行以下程序后,sum 的值是(　　)。

```
main()
{
    int i,sum;
    for(i=1;i<6;i++) sum+=i;
    printf("%d\n",sum);
}
```

A. 15　　B. 14　　C. 不确定　　D. 0

5. 以下程序段的输出结果是(　　)。

```
main()
{
    int n=9;
    while(n>6) {n--;printf("%d",n);}
}
```

A. 987　　B. 876　　C. 8765　　D. 98765

6. 有以下程序段,变量已正确定义,其输出结果是(　　)。

```
i=0;
do
```

```
printf("%d,",i);
while(i++);
printf("%d\n",i);
```

A. 0,0　　B. 0,1　　C. 1,1　　D. 程序进入无限循环

7. 以下程序的执行结果是(　　)。

```
main()
{
    int x=23;
    do
    {printf("%d",x);}
    while(!x);
}
```

A. 321　　B. 23

C. 不输出任何内容　　D. 陷入死循环

8. 以下程序的运行结果是(　　)。

```
#include <stdio.h>
main()
{
    int y=9 ;
    for( ;y>0;y--)
        if(y%3==0)  printf("%d",--y) ;
}
```

A. 741　　B. 963

C. 852　　D. 875421

9. 以下不构成无限循环的语句或语句组是(　　)。

A. `n=0;`
`do{++n;}while(n<=0);`

B. `n=0;`
`while(1){n++;}`

C. `n=10;`
`while(n);{n--;}`

D. `for(n=0,i=1; ;i++) n+=i;`

10. 以下程序段的输出结果是(　　)。

```
int x=3;
do
{printf("%d",x-=2);}
while(!(--x));
```

A. 1　　B. 3 0

C. 1 −2　　D. 死循环

11. 以下程序运行的结果是(　　)。

```
#include <stdio.h>
main()
{
```

```
    int i=5;
    do
    { if(i%3==1)
        if(i%5==2)
          { printf("* %d",i);break;}
      i++;
    }while(i!=0);
    printf("\n");
}
```

A. *7　　B. *3*5　　C. *5　　D. *2*6

12. 以下循环体的执行次数是(　　)。

```
main()
{
    int i,j;
    for(i=0,j=1; i<=j+1; i+=2, j--)  printf("%d\n",i);
}
```

A. 3　　B. 2　　C. 1　　D. 0

13. 以下叙述中正确的是(　　)。

A. break 语句只能用于 switch 语句体中

B. continue 语句的作用是使程序的执行流程跳出包含它的所有循环

C. break 语句只能用在循环体内和 switch 语句体内

D. 在循环体内使用 break 语句和 continue 语句的作用相同

14. 以下程序的运行结果是(　　)。

```
#include <stdio.h>
main()
{ int x=8;
  for( ;x>0;x--)
    {if(x%3){printf("%d,",x--);continue;}
    printf("%d,",--x);
    }
}
```

A. 7,4,2,　　B. 8,7,5,2,

C. 9,7,6,4,　　D. 8,5,4,2,

15. 以下程序的运行结果是(　　)。

```
main()
{int i,j,n=0;
  for(i=0;i<2;i++)
  for(j=0;j<2;j++)
    n++;
  printf("%d",n);
```

```
}
```

A. 4　　B. 2　　C. 2　　D. 0

16. 以下程序运行后的输出结果是(　　)。

```
main()
{int i,j;
  for(i=1;i<4;i++)
    {for(j=i;j<4;j++)
        printf("%d*%d=%d",i,j,i*j);
      printf("\n");
      }
}
```

A. 1*1=1 1*2=2 1*3=3
2*1=2 2*2=4
3*1=3

B. 1*1=1 1*2=2 1*3=3
2*2=4 2*3=6
3*3=9

C. 1*1=1
1*2=2 2*2=4
1*3=3 2*3=6 3*3=9

D. 1*1=1
2*1=2　2*2=4
3*1=3　3*2=6　3*3=9

17. 以下程序的运行结果是(　　)。

```
#include <stdio.h>
main()
{int i,j,m=55;
    for(i=1;i<=3;i++)
        for(j=3;j<=i;j++) m=m%j;
    printf("%d\n",m);
}
```

A. 0　　B. 1　　C. 2　　D. 3

18. 以下程序运行的结果是(　　)。

```
#include <stdio.h>
#include <stdio.h>
main()
{int i,j;
  for (i=3;i>=1;i--)
    {for(j=1;j<=2;j++)
     printf("%d",i+j);
     printf("\n");
    }
}
```

A. 2 3 4
3 4 5

B. 4 3 2
5 4 3

C. 2 3
3 4
4 5

D. 4 5
3 4
2 3

19. 下面程序的功能是计算 1～10 的奇数之和及偶数之和，应补充的语句是(　　)。

```
#include <stdio.h>
main()
{
    int a,b,c,i;
    a=c=0;
    for(i=0;i<=10;i+=2)
        {a+=i;
        _____;
        c+=b;
        }
    printf("偶数之和=%d\n",a);
    printf("奇数之和=%d\n",c-11);
}
```

A. i+=2　　B. i++

C. b=b+1　　D. b=i+1

20. 若输入字符串“abcde”后按 Enter 键，则以下 while 循环体执行的次数是(　　)。

```
while((ch=getchar())=='e') printf("*");
```

A. 0 次　　B. 4 次

C. 6 次　　D. 1 次

二、填空题(每空 2 分，共 20 分)

1. 以下程序的输出结果是_____。

```
#include <stdio.h>
main()
{ int n=12345,d;
  while(n!=0){d=n%10;printf("%d",d);n/=10;}
}
```

2. 若有定义“int k;”，则以下程序段的输出结果是_____。

```
for(k=2;k<6;k++,k++)  printf("##%d",k);
```

3. 有以下程序段，且变量已正确定义和赋值。

```
for(s=1.0,k=1;k<=n;k++)
s=s+1.0/(k*(k+1));
printf("s=%f\n\n",s);
```

填空使下面程序段的功能与上述程序段的功能完全相同。

```
s=1.0;k=1;
while(_____)
{s=s+1.0/(k*(k+1));_____;}
printf("s=%f\n\n",s);
```

4. 填入一个整数使以下程序段输出 10 个整数。

```
for(i=0;i<=_____;printf("%d\n",i+=2));
```

5. 填空使下面程序的功能为输出 100 以内能被 3 整除且个位数为 6 的所有整数。

```
#include <stdio.h>
main()
{int i,j;
  for(i=0;_____;i++)
    {j=i*10+6;
    if(_____) continue;
    printf("%d",j);
    }
}
```

6. 以下程序的输出结果是_____。

```
#include <stdio.h>
main()
{int i,j,sum;
  for(i=3;i>=1;i--)
    {sum=0;
    for(j=1;j<=i;j++) sum+=i*j;
    }
  printf("%d\n",sum);
}
```

7. 根据前面 1～100 之和的程序，求 1＋2＋…＋n≥3 000 中的 n 值，即从 1 开始的连续自然数之和，累加到 n 时，其值大于或等于 3 000。填空使下面的程序实现上述功能。

```
#include "stdio.h"
main()
{
  int i=1,s=0;
  while(i<=100)
    {
    s=s+i;
    if(s>=3000)
        _____;
    i++;
    }
  printf("s=%d,i=%d\n",s,i);
}
```

8. 上题中也可以将 s>=3 000 作为循环的终止条件，即当 s<3 000 时循环，程序运行结果一致。

```
#include "stdio.h"
main()
{
```

```
    int i=0,s=0;
    while(_____)
      {
      i++;
      s=s+i;
      }
  printf("s= %d,i= %d\n",s,i);
  }
```

三、程序改错(每错 10 分,共 20 分)

以下是一个能接受两次输入且判断输入是否为素数的程序,程序中在 found 下一行有错,请改正。

```
#include "stdio.h"
main()
{int x,i=2,j;
  /*********found*********/
  while(i++)                    //加一个循环,目的是执行两遍判断素数的问题
  {printf("请输入一个正整数:");
    scanf("%d",&x);
    for(j=2;j<x;j++)            //让 x 遍除 2~(x-1),一旦除开结束循环
      if(x%j==0)
  /*********found*********/
          continue;
      if(i==x)                  /*出循环,从 i 值可判断是否除开,i 等于 x 说明没
                                  有除开,是素数*/
          printf("%d 是素数\n",x);
      else
          printf("%d 不是素数\n",x);
   }
  getchar();
}
```

四、编程题(每题 20 分,共 40 分)

1. 编程计算 $n!$。

```
.....                                   //文件包含
.....                                   //主函数
{

    .....
    .....
    .....
```

.....

.....

.....

.....

}

2. 已知小红今年 12 岁，父亲比她大 20 岁，问多少年之后，父亲的年龄是小红的二倍？编写一个程序求解这个问题。

..... //文件包含

..... //主函数

{

.....

.....

.....

.....

.....

.....

.....

}

项目实训一

直流电动机控制系统设计

实训目标

了解单片机 C 语言编写的独特之处；

了解直流电动机控制系统电子电路的工作原理；

掌握用 C 语言控制直流电动机的基本方法；

掌握用 C 语言编写项目程序的基本思路。

项目分析

1. 任务说明

直流电动机由于其良好的起动、制动和调速性能，在一些要求较高的电力拖动系统中有着广泛的应用。本项目实训要求用 C 语言程序驱动单片机实现对直流电动机转速、转向的控制。具体要求如下。

(1)编写程序，通过 AT89S51 单片机输出 PWM 波，经驱动电路使直流电动机按固定方向和固定转速旋转。

(2)在此基础上使用一个单刀双掷开关控制直流电动机转向。

(3)在此基础上使用一个电位器调节直流电动机的转速。

2. 设计方案

本系统由转速控制模块、转向控制模块、单片机核心控制系统、电动机驱动电路和直流电动机五部分组成。直流电动机控制系统结构框图如图 1 所示，其中，控制核心是单片机部分，它采集转速控制模块和转向控制模块提供的输入信号，经过内部程序的运算处理，分别通过不同的 I/O 口输出转速和转向控制信号给直流电动机驱动电路，从而控制直流电动机的运行状态。

(1)转向控制模块设计方案。采用一个单刀双掷开关，分别使其开关状态为 1 和 0 信号，驱动电动机正转和反转。

(2)转速控制模块设计方案。使用电位器作为转速输入设备，用手调动电位器，使之改变输入电压，将变化的模拟量电压值经过 AD 转换芯片 ADC0831 离散为数字信号输入单片机 I/O 口，单片机经过程序运算后输出占空比不同的 PWM 波，从而达到调整直流电动机转速的目的。

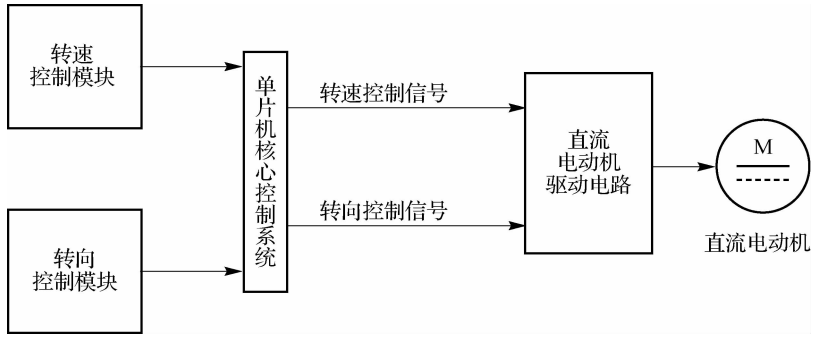

图 1　直流电动机控制系统结构框图

(3)单片机选择方案。选择市面上常见的 AT89S51 单片机。

(4)电动机驱动电路设计方案。由于采用 12 V 直流电动机，所以需设驱动电路。本项目采用不同功率三极管搭建成的驱动电路，可以同时控制电机的方向和转速。

系统硬件原理图如图 2 所示。

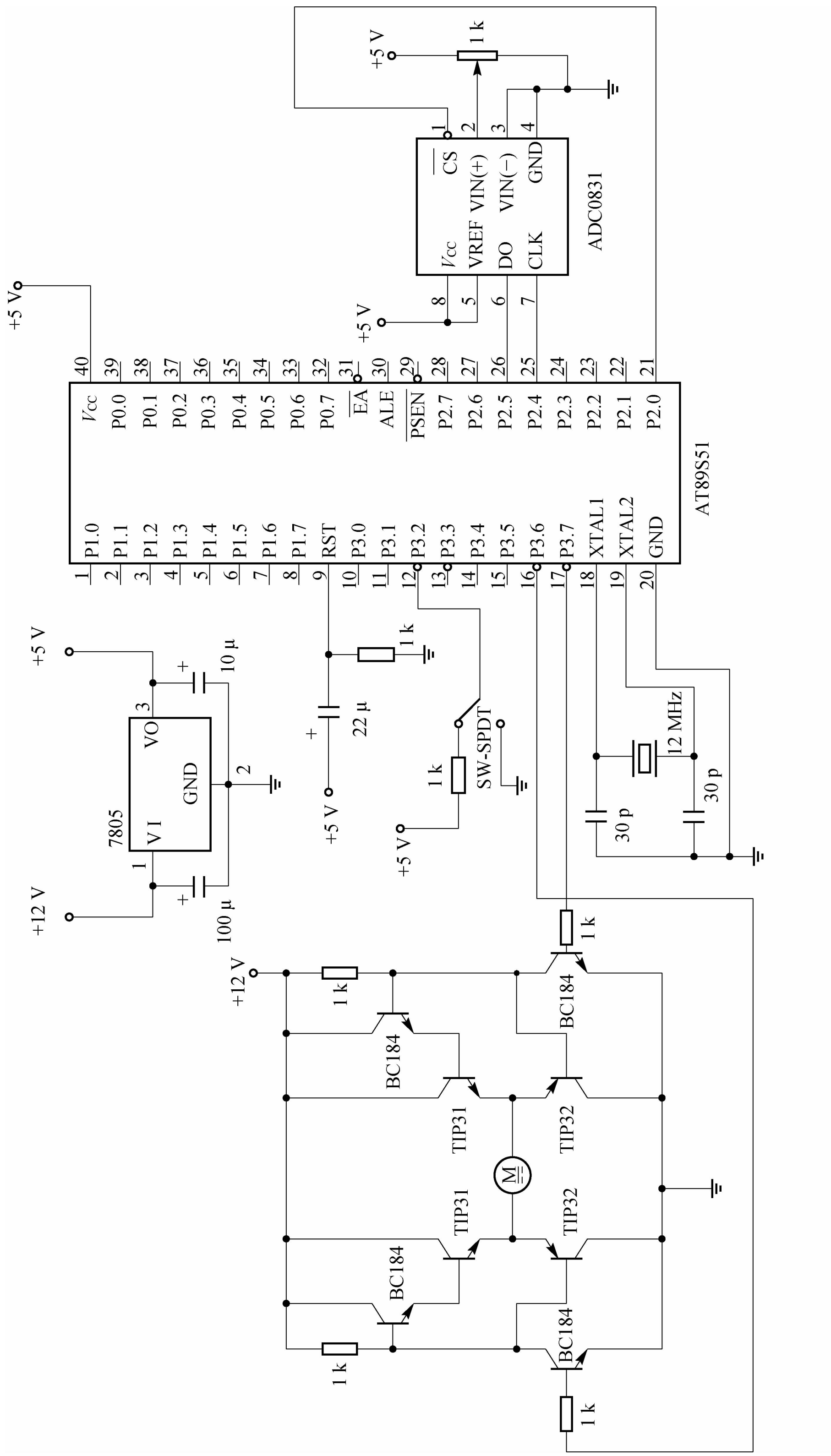

图2 直流电动机控制系统硬件原理图

知识准备

1. AT89S51 单片机的硬件结构

单片机是一种采用超大规模集成电路技术把具有数据处理能力的中央处理器、存储器、I/O 口和总线等集成到一块硅片上的计算机系统。由于它特有的稳定性高、功耗低、面积小、价格低廉等优点，在生产生活中，特别是工业领域中，有着广泛的应用。单片机的处理速度由产生之初的 4 位、8 位逐步发展成现在的 32 位，并且还在继续发展和变化中。

目前，常用的单片机系列有 MCS-51 单片机、AVR 单片机、PIC 单片机、NEC 单片机等，其中初学者接触最多的是 MCS-51 单片机，80C51 单片机是该系列中的一个产品。图 3 为 80C51 单片机的引脚排列。

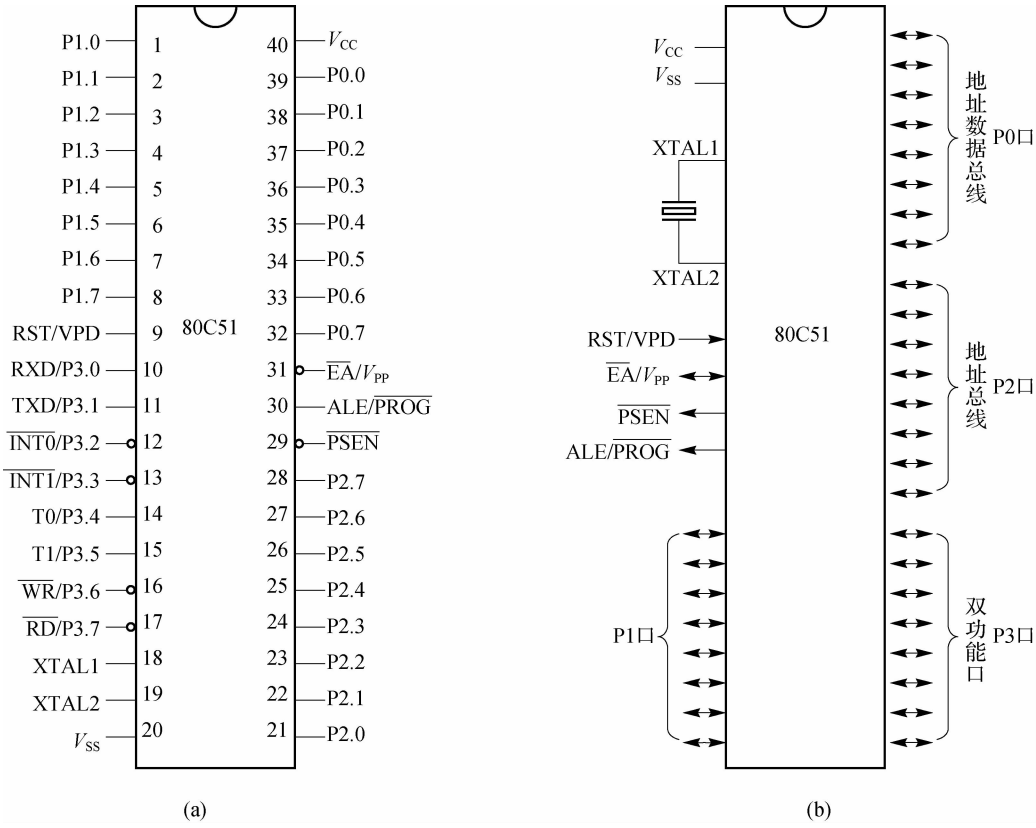

图 3　80C51 单片机引脚排列

本项目采用的 AT89S51 单片机采用常用的 MCS-51 架构，是一种低功耗、高性能的 CMOS 8 位微控制器，具有 4 KB 在系统可编程 Flash 存储器。使用 Atmel 公司高密度非易失性存储器技术制造，与工业 80C51 单片机产品指令和引脚完全兼容。片上 Flash 允许程序存储器在系统可编程，亦适用于常规编程器。在单芯片上拥有灵巧的 8 位 CPU 和在系统可编程 Flash，使得 AT89S51 单片机在众多嵌入式控制应用系统中得到广泛应用。AT89S51 单片机的具体标准功能为：4 KB 的 FlashROM，128 B 的 RAM，32 位 I/O 口线，数据指针，两个 16 位定时/计数器，一个 6 向量二级中断结构，全双工串行口，片内晶振及时钟电路。另外，AT89S51 单片机可降至 0 Hz 静态逻辑操作，支持两种软件，可选择节电模式。

空闲模式下，CPU 停止工作，允许 RAM、定时/计数器、串行口、中断继续工作。掉电保护方式下，RAM 内容被保存，振荡器被冻结，单片机一切工作停止，直到下一个中断或硬件复位为止。

AT89S51 单片机的引脚功能介绍如下。

1）P0 口～P3 口

P0 口～P3 口是单片机的 I/O 口，即输入/输出接口，单片机与外界的信号传输就是通过这些 I/O 口进行的，简单地说，就是通过它接受外部信号和驱动外部设备。MCS-51 系列单片机的 I/O 口共 32 位，以下为具体说明。

（1）P0 口。P0 口是一个 8 位漏级开路双向 I/O 口，每只引脚可吸收八个 TTL 门电流。P0 口能够用于外部程序数据存储器，它可以被定义为数据/地址的低 8 位。在 Flash 编程时，P0 口作为原码输入口，当 Flash 进行校验时，P0 口输出原码，此时 P0 口外部必须被拉高。

（2）P1 口。P1 口是一个内部提供上拉电阻的 8 位双向 I/O 口，P1 口缓冲器能接收、输出四个 TTL 门电流。P1 口管脚写入“1”后，被内部上拉为高，可作为输入，P1 口被外部下拉为低电平时输出电流，这是由于内部上拉的缘故。在 Flash 编程和校验时，P1 口作为第 8 位地址接收。

（3）P2 口。P2 口是一个内部上拉电阻的 8 位双向 I/O 口，P2 口缓冲器可接收输出四个 TTL 门电流，当 P2 口被写“1”时，其引脚被内部上拉电阻拉高，可作为输入。P2 口作为输入时，其引脚被外部拉低，将输出电流，这是由于内部上拉。P2 口用于外部程序存储器或 16 位地址外部数据存储器进行存取时，P2 口输出地址的高 8 位，在给出地址“1”时，它利用内部上拉优势。当对外部 8 位地址数据存储器进行读写时，P2 口输出其特殊功能寄存器的内容。P2 口在 Flash 编程和校验时接收高 8 位地址信号和控制信号。

（4）P3 口。P3 口引脚是八个带内部上拉电阻的双向 I/O 口，可接收输出四个 TTL 门电流。当 P3 口写入“1”后，它们被内部上拉为高电平，可作为输入。作为输出时，由于外部下拉为低电平，P3 口将输出电流。P3 口除了作为普通 I/O 口，还有第二功能，具体内容见表 1。

表 1　P3 口的第二功能

P3 口引脚	第二功能
P3.0	RXD——串行输入口
P3.1	TXD——串行输出口
P3.2	$\overline{\text{INT0}}$——外部中断 0
P3.3	$\overline{\text{INT1}}$——外部中断 1
P3.4	T0——T0 定时器的外部计数输入
P3.5	T1——T1 定时器的外部计数输入
P3.6	$\overline{\text{WR}}$——外部数据存储器的写选通
P3.7	$\overline{\text{RD}}$——外部数据存储器的读选通

P3 口同时可为闪烁编程和编程校验接收一些控制信号。

I/O 口作为输入时有两种工作方式，即所谓的读端口与读引脚。读端口实际上并不从

外部读入数据，而是把端口锁存器的内容读入内部总线，经过某种运算或变换后再写回端口锁存器。只有读端口时才真正地把外部的数据读入内部总线。AT89S51 单片机的 P0 口、P1 口、P2 口、P3 口作为输入时都是准双向口。除 P1 口外，P0 口、P2 口、P3 口都还有其他的功能。

2）RST

RST 引脚为复位输入端，高电平有效。当振荡器复位器件时，要保持 RST 引脚两个机器周期的高电平时间。

3）ALE/$\overline{\text{PROG}}$

ALE/$\overline{\text{PROG}}$为地址锁存允许/编程脉冲信号端。当访问外部存储器时，地址锁存允许的输出电平用于锁存地址的低位字节。在 Flash 编程期间，此引脚用于输入编程脉冲。在平时，ALE 引脚以不变的频率周期输出正脉冲信号，此频率为振荡器频率的 1/6，因此它可用做对外部输出的脉冲或用于定时。然而，每当 ALE 用做外部数据存储器时，将跳过一个 ALE 脉冲。如想禁止 ALE 的输出，可在 SFR 的 8EH 地址上置 0。此时，ALE 只有在执行 MOVX、MOVC 指令时 ALE 才起作用。另外，该引脚被略微拉高。如果微处理器在外部执行状态 ALE 禁止，置位无效。

4）$\overline{\text{PSEN}}$

$\overline{\text{PSEN}}$为外部程序存储器的选通信号，低电平有效。在由外部程序存储器取指令期间，每个机器周期两次$\overline{\text{PSEN}}$有效。但在访问外部数据存储器时，这两次有效的$\overline{\text{PSEN}}$信号将不出现。

5）$\overline{\text{EA}}/V_{\text{PP}}$

$\overline{\text{EA}}/V_{\text{PP}}$引脚功能为外部程序存储器访问允许。当$\overline{\text{EA}}$保持低电平时，访问外部程序存储器（0000H～FFFFH），不管是否有内部程序存储器，加密方式 1 时，$\overline{\text{EA}}$将内部锁定为 RESET；当$\overline{\text{EA}}$端保持高电平时，访问内部程序存储器。在 Flash 编程期间，此引脚也用于施加 12 V 编程电源（V_{PP}）。

6）XTAL1

XTAL1 为片内振荡器反相放大器和时钟发生器的输入端。

7）XTAL2

XTAL2 为片内振荡器反相放大器的输出端。

关于 AT89S51 单片机的具体信息，网上有很多资料可供下载，请读者自行参阅，本书不再赘述。

2. AT89S51 单片机编程语言简介

对于 MCS-51 单片机，现有四种语言支持，分别为汇编、PL/M、C 和 BASIC。其中 C 语言由于使用非常方便而得到广泛的支持。C 语言本身不依赖于机器硬件系统，基本上不用修改就可移植过来。

MCS-51 单片机使用的 C 语言基本上等同于通用 C 语言，但有其独特的地方，所以也有将 MCS-51 单片机所使用的 C 语言称为 C51 语言的说法。具体介绍详见本项目实训项目实例的“程序释疑”部分。

3. 直流电动机的方向和转速控制

直流电动机驱动电路如图 4 所示。

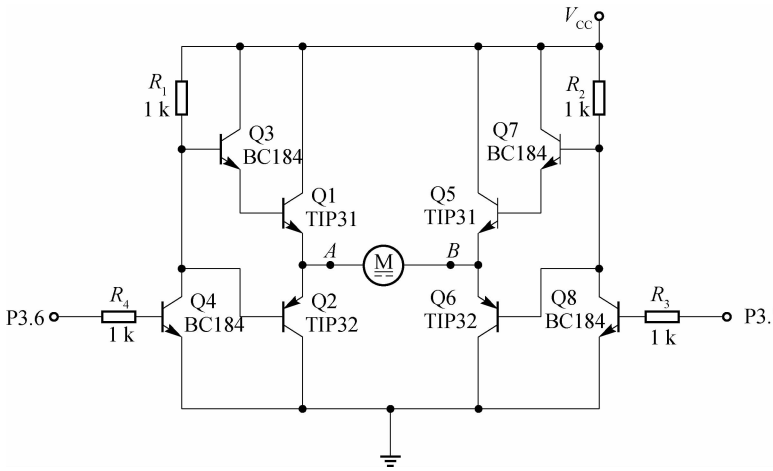

图 4　直流电动机驱动电路

根据直流电动机的原理，改变流过电枢绕组的电流方向即可改变电动机的电磁转矩，从而改变旋转方向，改变电枢电压可以调节转速。本实验通过程序使单片机输出可变占空比的 PWM 波来改变电枢电压从而改变电动机转速。

单片机的 P3.6 引脚控制直流电动机的转向，P3.7 引脚控制转速。根据电子学知识，具体工作原理如下。

(1)当 P3.6 引脚输出高电平时，Q2 和 Q4 导通，Q1 和 Q3 截止，电动机左端(*A* 点)电位被拉低。

P3.7 引脚为高电平时，Q6 和 Q8 导通，Q5 和 Q7 截止，电动机右侧(*B* 点)电位也被拉低，故电动机电枢中无电流流过；P3.7 引脚为低电平时，Q6 和 Q8 截止，Q5 和 Q7 导通，*B* 点电位为高电平，电动机电枢中的电流方向为从右至左，电动机正转。

(2)当 P3.6 引脚输出低电平时，Q2 和 Q4 截止，Q1 和 Q3 导通，电动机左端(*A* 点)电位被抬高。

P3.7 引脚为高电平时，Q6 和 Q8 导通，Q5 和 Q7 截止，电动机右侧(*B* 点)电位被拉低，电动机电枢中的电流方向为从左至右，电动机反转。P3.7 引脚为低电平时，Q6 和 Q8 截止，Q5 和 Q7 导通，*B* 点电位也为高电平，故电动机电枢中无电流流过。

由此可看出：P3.6＝1 时，电动机正转；P3.6＝0 时，电动机反转。

使用延时函数控制 P3.7 引脚高低电平时间长短的变化来控制加在直流电动机电枢绕组上 PWM 波的占空比，进而控制电压高低，达到控制转速的目的。

4. A/D 转换芯片 ADC0831 工作原理简介

ADC0831 芯片是常用的八位串行逐次逼近式 A/D 转换芯片，转换速度高，功耗低，适合在袖珍式智能仪器中使用，可作为微处理器接口或独立操作。ADC0831 芯片使用 5 V 参考电压，支持单通道或双通道使用。通常情况下，参考输入电压与 V_{CC} 相等。

图 5 所示为 ADC0831 芯片引脚图。

(1)$\overline{CS}$为片选信号引脚，一旦为低电平则选中该芯片工作。

(2)$V_{IN}+$和 $V_{IN}-$是差分输入端，若输入一路信号，则 $V_{IN}-$接地。

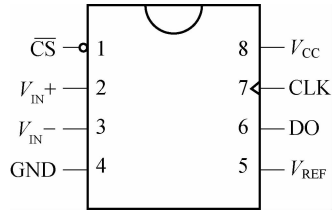

图 5　ADC0831 芯片引脚图

(3)GND 为芯片接地端。

(4)V_{REF}为参考电压输入端,使用时一般将其与供电 V_{CC}接到一起。

(5)DO 为数字信号输出端,信号从该引脚以串行方式输出。

(6)CLK 为时钟信号的输入引脚,数字信号就是靠时钟信号带动输出的。

(7)V_{CC}为芯片的供电引脚,本芯片采用 5 V 直流电压供电。

ADC0831 芯片工作过程如下。

(1)将 CLK 引脚电平拉低,之后将片选引脚置低电平,A/D 转换启动。

(2)在第一个时钟信号的下降沿来到时,DO 端输出低电平,为数字信号的输出做好准备。

(3)第二个时钟信号下降沿到来时,ADC0831 芯片才开始转换数据,这样一直到第九个时钟信号下降沿,转换出 8 位数据,即 1 字节,输出位的顺序为从最高位至最低位。单片机可以通过某一引脚接收该串行信号,并通过数据移位的方法将该数存入某一变量中进行处理。

图 6 为 ADC0831 芯片工作时序图。

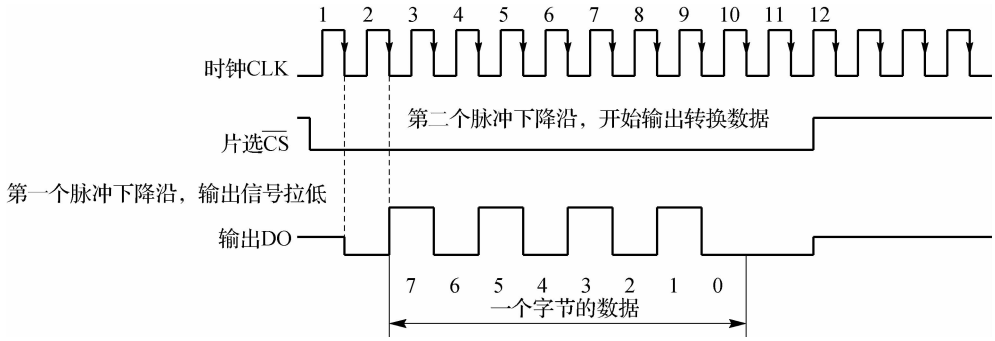

图 6　ADC0831 芯片工作时序图

项目实例

1. 实例的技术要求

请编写程序,使单片机 P3.7 引脚输出占空比为 0.5 的 PWM 波,驱动直流电动机匀速正转运行。

2. 流程图和源程序

(1)流程图。

图 7 所示为直流电动机正转程序流程图。

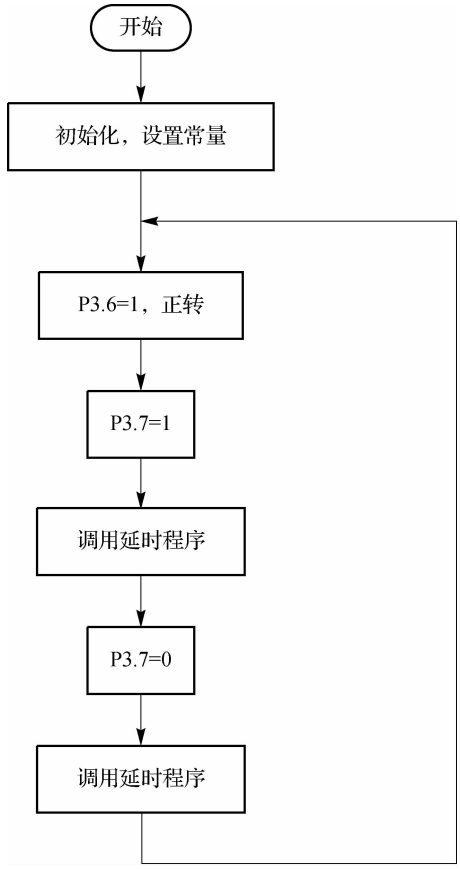

图 7　直流电动机正转程序流程图

(2)源程序。

```
/*************************************************************/
/*文件包含,程序开始*/
/*************************************************************/
1     #include <reg51.h>
2     #define uchar unsigned char
3     #define uint unsigned int
4     sbit PWM=P3^7;
5     sbit DIR=P3^6;
/*************************************************************/
/*延时子程序,ms 是形式参数*/
/*************************************************************/
6     void delay(uchar ms)
7     {
8     int i;
9     while(ms--)
```

```
10        {
11         for(i=0;i<38;i++);
12        }
13      }
/*****************************************************************/
/*正转主程序*/
/*****************************************************************/
14      void main()
15      {
16        while(1)
17      { DIR=1;
18        PWM=1;
19        delay(128);
20        PWM=0;
21        delay(128);
22        }
23     }
```

(3)程序释疑。

单片机的C语言是由通用C语言继承而来的,运行于单片机平台。因此单片机C语言既具有结构清晰、便于学习的优点,又具有汇编语言的硬件操作能力。对于具有通用C语言编程基础的读者,能够轻松地掌握单片机C语言的程序设计。

单片机的C语言常用语法比通用C语言语法要少,所以易读性非常强。但有些地方有些独特之处。现针对本实例作简要介绍。

程序第1行是“文件包含”,意思是本文件将另外一个文件全部包含进来。被包含的文件称为头文件,扩展名为.h,其中reg51.h主要介绍了单片机的简单框架结构,规定了符号名与地址的对应关系,一般都应写上。

程序第2行和第3行是宏定义语句,#define命令用它后面的第一个字母组合代替该字母组合后面的所有内容,即给原内容重新命名一个相对简单易懂的新名称,这样以后在程序中就可以用这种简单易懂的名称代替原有繁琐复杂的名称了。如第2行中,就是用uchar来代替无符号字符型unsigned char,在后面出现该类型变量时,都是用uchar来替代unsigned char进行定义的。第3行也是同样的意思。

程序的第4、5行分别用易理解的符号来表示不同的I/O口。sbit是单片机编译软件Keil C51的保留字,为了让单片机理解PWM就是P3^7,DIR就是P3^6,就必须用此种格式将所有需要替换的名称全部定义一遍。

程序的6～13行是延时函数。ms是形参,由于其数据类型是uchar,因此实参取值范围为0～255。本实例要求单片机输出占空比为0.5的PWM波,故主程序中实参为128。

程序的14～23行为主函数部分。在单片机C语言中,while(1)是主函数中一个常用的语句,“1”是条件永远为真的意思,就是无限循环。这是因为单片机程序是裸机跑程序,里面的程序要不断地循环运行(除非编写者只想让程序运行一遍),不断地扫描,所以要加上这一

条无限循环的语句。其后面的花括号中的内容就是循环内容，即单片机的工作内容。

项目实施

工作任务一 使直流电动机按固定方向和固定转速旋转

请参考“项目实例”写出流程图和程序，要求单片机 P3.7 引脚输出占空比为 0.25 的 PWM 波，驱动直流电动机匀速反转运行。

(1)流程图。

(2)源程序。

工作任务二 控制直流电动机的转向

若在硬件电路中用 P3.2 引脚加装一个单刀双掷开关，使 P3.6 引脚控制直流电动机的旋转方向，应在以下程序的基础上如何完善程序？

温馨提示

现在已经可以编写正反转的程序，可将这两个程序编成函数，分别命名为 forward 和 reversal，在编程时，给 P3.2 引脚赋 1 或清 0，以此调用不同的正反转函数，即可配合硬件电路控制直流电动机的转向。

```
/*************************************************************/
/* 包含文件,程序开始 */
/*************************************************************/
#include <reg51.h>
#define uchar unsigned char
#define uint unsigned int
sbit PWM=P3^7;
sbit DIR=P3^6;
sbit SW=P3^2;
/*************************************************************/
/* 延时子程序,ms 是形式参数 */
/*************************************************************/
void delay(uchar ms)
{
 int i;
 while(ms--)
 {
   for(i=0;i<38;i++);
 }
}
/*************************************************************/
/* 直流电动机正转函数 */
/*************************************************************/
void forward()
{
.....
.....
.....
.....
.....
}
/*************************************************************/
```

```
/*直流电动机反转函数*/
/************************************************************/
void reversal()
{
  ..........
  ..........
  ..........
  ..........
  ..........
}
/************************************************************/
/*主程序*/
/************************************************************/
void main()
{
 while(1)
 {
  ..........
  ..........
  ..........
  ..........
  ..........
 }
}
```

工作任务三 控制直流电动机的转速

在电路中加装一个电位器，要求编写程序，通过调节电位器改变直流电动机的转速。

温馨提示

可使用 ADC0831 芯片对电位器的输出电压模拟量进行实时转换，单片机读取转换后的数字量，作为 PWM 波的延时时间常数，调节 PWM 波的占空比，从而调节电动机转速。

ADC0831 芯片转换数据的读入：单片机通过 P2.4 引脚发出脉冲至 ADC0831 芯片的 CLK 引脚，用 P2.5 引脚接收芯片输出的串行信号，并通过数据移位的方法将该数存入某一变量中进行处理。

转速的控制：由图 2 可知，通过调节 1 KB 的电位器控制 ADC0831 芯片输入电压的大

小，由 P2.4 引脚输出时钟脉冲，P2.5 引脚接收转换后的串行数字信号。

由于 ADC0831 芯片为 8 位串行逐次逼近式 A/D 转换芯片，所以转换出的数据最大为 255，设输出数字为 Dout，则产生 PWM 波的方法可以使 P3.7 引脚输出高电平，然后延时时间常数为 255－Dout，再输出低电平，延时时间常数设为 Dout，依此往复循环即可。

由于输出 Dout 是根据电位器分得电压转换而得，所以可通过调节电位器来改变 Dout，从而改变 PWM 波的占空比，最终达到改变直流电动机转速的目的。

图 8 是根据图 6 所画的 ADC0831 芯片转换数据读入函数流程图，请尝试补全后面的 AD 转换函数。程序中部分释疑如下。

(1)_nop_函数是空操作函数，该函数什么都不做，只是消耗一个机器周期的时间，以上程序用该函数做一个脉冲的延时，相当于汇编语言中的 NOP 空指令。该函数包含在库函数 intrins.h 中，若要用到该函数，在程序开始部分将该头文件包含进来即可。

(2)由图 2 可知，程序中的变量 CS、CLK 和 DO 分别是 P2.0 引脚、P2.4 引脚和 P2.5 引脚。

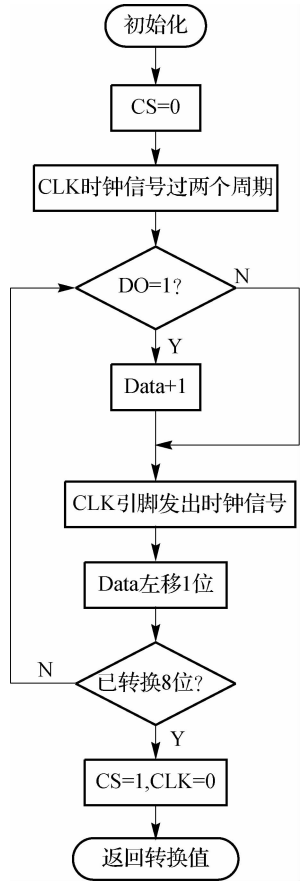

图 8　ADC0831 芯片转换数据读入函数流程图

```
uchar AD_CONV(void)     //ADC0831 转换函数名称
{
  uchar i;
```

```
    uchar Data;
    CLK=0;                  //过两个脉冲
    CS=0;
    _nop_();                /*C51 库函数,包含在 intrins.h 中,消耗一个机器周期时
                              间*/
    CLK=1;
    _nop_();
    CLK=0;
    _nop_();
    CLK=1;
    _nop_();
    CLK=0;
    _nop_();
    for(i=8;i>0;i--) //开始取数
    {
     ..........
     ..........
     ..........
     ..........
     ..........
     ..........
     }
    CS=1;
    CLK=0;
    for(i=40;i>0;i--)  //延时一段时间,返回读取到的值
     {
       _nop_();
     }
     return(Data);
}
```

请编写完整的流程图和程序,使得可以对直流电动机进行正反转控制,并可以调速。

（1）流程图。

(2)源程序。

总结与思考

1. 项目总结

(1)由本项目可以看出,单片机 C 语言编写的项目程序在语法结构上并不复杂,需要特别关注的是电路的硬件结构和芯片的工作方式。在编写单片机 C 语言程序前,必须有足够的了解,才能编写出正确合理的程序。

(2)编写单片机程序时要养成画软件模块流程图的好习惯,在此基础上编写程序可以事半功倍。编写顺序上可以先确定主程序,再编写函数。确定主程序可以在框架上作整体把握,定下核心任务,程序就不会出大问题。

2. 项目思考

(1)单片机 C 语言与通用 C 语言有什么异同?

(2)为什么单片机程序中常用 while(1)语句?

(3)能否编写程序,在现有电路中实现电动机先正转 10 s 后自动反转 10 s,循环往复?

(4)能否保持现有硬件电路不变,自己设计工艺流程,编写一个豆浆机控制程序?