第五章　循环构造程序设计

5.1 while语句

5.2 do-while语句

5.3 for 语句

5.4 break、continue和goto语句

5.5 循环的嵌套

5.6 复合构造程序举例

教学目的和基本要求：

要求学生了解循环构造程序设计，掌握多种循环语句应用的特点及异同点，掌握循环嵌套及复合构造。

教学要点：

多种循环语句应用的特点及异同点。

编程处理这么的一种问题：

从键盘输入一百个学生的成绩，求总成绩。

从前面所学，有两种处理措施。

1.设一百个变量，分别输入学生的成绩，然后求和。

这种措施挥霍内存空间，显然不实际。

2.设一种变量，每次输入一种学生成绩，累加后再输入下一种学生成绩，如下：

```
scanf("%f",&a);
s=s+a;
scanf("%f",&a);
s=s+a;
.....
```

这么反复一百次，然后输出s的值。

这么写显然非常麻烦。我们注意到程序中的

```
scanf("%f",&a);
s=s+a;
```

两句话是一直反复的，假如能用一种语句，使这两句话能自动的反复执行一百次，就能够简化了书写的麻烦，这就是循环语句。

C语言有while、 do - while、和for语句三种循环构造语句。

前两个称为条件循环，即根据条件来决定是否继续循环；

后一种称为计数循环，即根据设定的执行次数来执行循环。

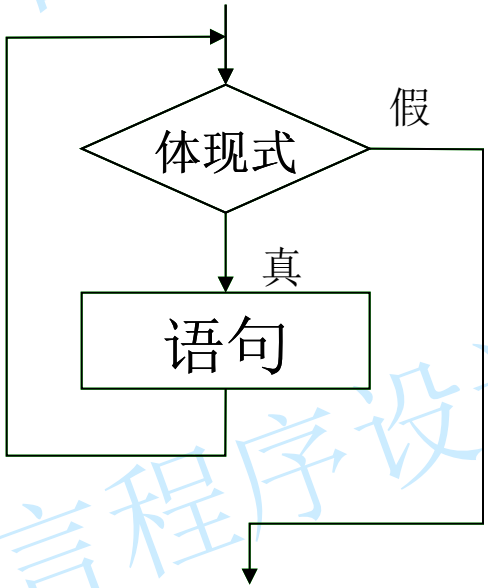
5.1 while语句

一般形式：

while(体现式) 语句

执行流程：

1. 计算体现式
2. 假如体现式的值为非零，执行语句
3. 返回第一步，重新计算体现式
4. 假如体现式的值为零，则结束循环



假如体现式的值一开始就为0，则语句一次也不会被执行。

while语句举例

问题: 求学生的平均成绩，以输入负数成绩为结束

算法分析:

1. 定义变量score存储学生成绩，定义s=0存储累加的成绩，

   定义n=0统计录入的成绩数目。

2. 输入第一种学生的score

3. 若score>=0,执行第4步，不然执行第7步

4. n++

5. s=s+score

6. 录入下一种score,并返回第3步

7. 假如n>0,输出s/n不然输出没有学生成绩

程序：

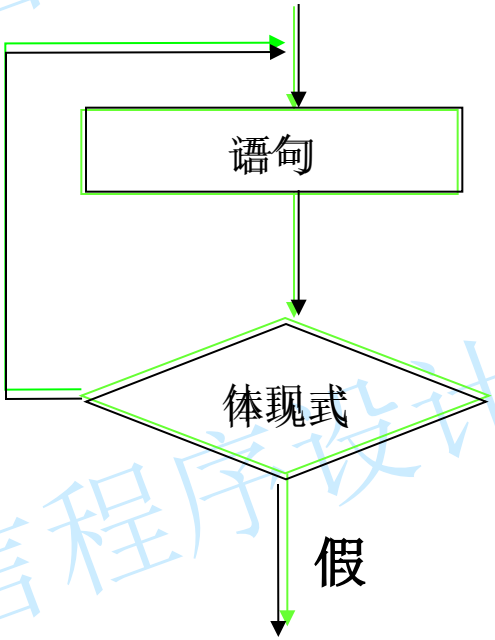
```
main( )
{ int  n=0 ;
  float s=0,score;
  scanf( "%f" ,&score);
  while (score >= 0 )
    {  n++;
       s=s+score;
       scanf( "%f" ,& score);
     }
  if(n>0)  printf (" \n %f", s/n);
  else   printf("no student score!");
}
```

5.2 do—while语句

一般形式:

```
do{
    语句
}while (体现式);
```

注意:
分号不能丢



执行流程:

1. 执行语句
2. 计算体现式
3. 体现式的值为非零，返回第1步
4. 体现式的值为零，结束循环

语句至少被执行一次。

用do-while语句编写统计学生平均成绩的程序：

```
main( )
{ int  n=0 ;
  float s=0,score;
  do { scanf( "%f" ,& score);
       n++;
       s=s+score;
     }while(score>=0);
  if(n>1)  printf (" \n %f" ,(s-score)/(n-1));
  else   printf("no student score!");
}
```

因为do-while语句至少要被执行一次，特别要注意n和s的取值问题！

想想这么的一段循环语句的执行成果:

```
i=1;
while (i<=100)
  putchar('*');
i++;
```

这个循环永远不会结束，因为循环控制变量i没有在循环体内被变化，i++; 不属于循环体。

应该改为:

```
i=1;
while (i<=100)
{ putchar('*');
  i++;
}
```

循环语句中一定要注意体现式的值是否能在循环执行过程中被变化，以免造成死循环。

5.3 for循环语句

一般形式：

for(体现式1;体现式2;体现式3)　循环体语句

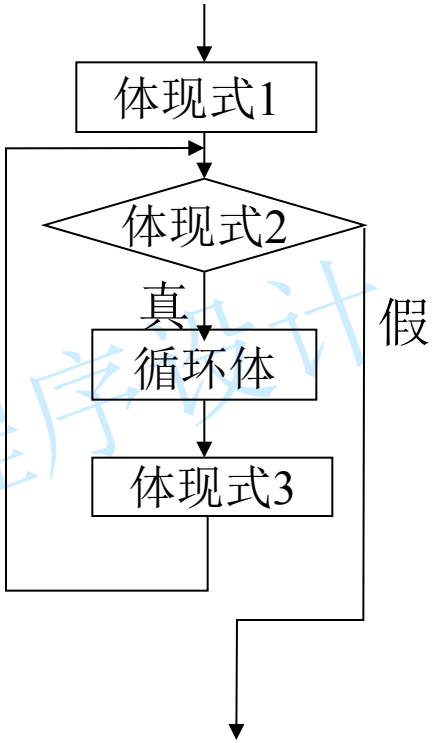
执行流程

1.计算体现式1，一般用于循环开始前设置变量初值。

2.计算体现式2，值为0则结束循环，不然执行第3步。

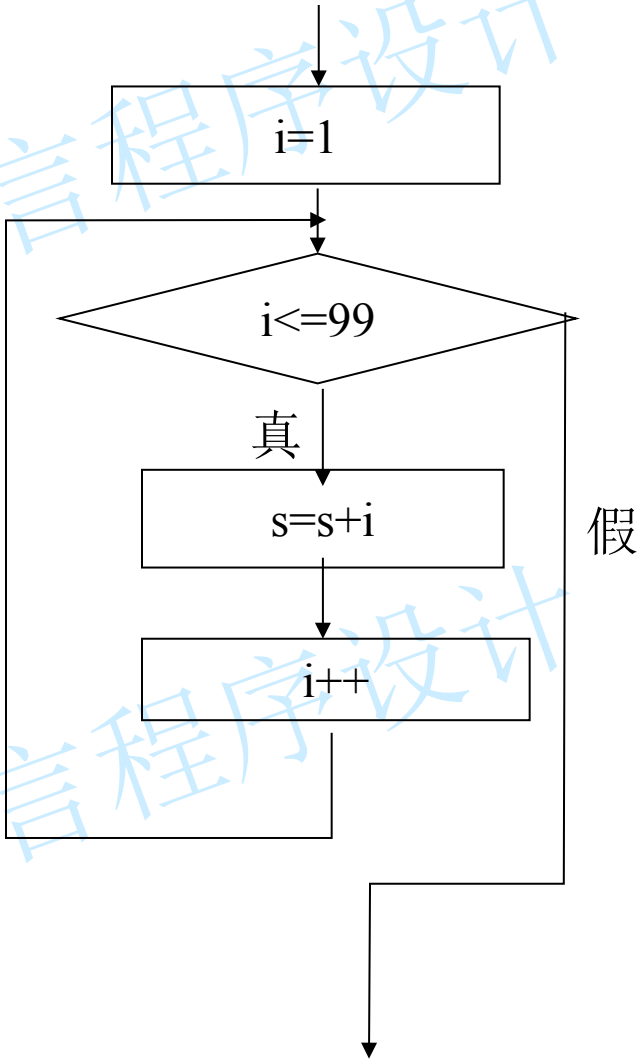
3.执行循环体语句。

4.计算体现式3，返回第2步。



例: 求 1+2 +…+99

分析：用变量i从1到99循环，把i的值累加到变量s中，最终输出s的值。



程序如下：

```
#include<stdio.h>
main( )
{  int i,s=0;
    for(i=1;i<=99;i++)
        s=s+i;
    printf("s=%d",s);
}
```

从上面的程序我们看到，for语句中：

体现式1：一般是给循环变量赋初值

体现式2：循环是否继续执行的鉴别体现式，这个体现式一般与某一种（或多种）变量的值有关，伴随这个（些）变量的值的变化，体现式的成果发生变化，这个（些）变量被称为循环因变量。

体现式3：一般用于变化循环因变量的值。

在某些情况下，for语句中的体现式1、2、3都能够省略，而改用其他的方式来实现这些功能。我们还用上面的例子阐明for语句省略体现式的情形。

1.省略体现式1

```
#include<stdio.h>
 main( )
{  int i=1,s=0;
    for( ;i<=99;i++)
        s=s+i;
     printf(“s=%d”,s);
}
```

2.省略体现式3：

```
#include<stdio.h>
main( )
{  int i,s=0;
   for(i=1;i<=99; )
       {s=s+i;i++;}
    printf(“s=%d”,s);
}
```

注意：体现式省略，分号不省略。

3. 同步省略体现式1、3

```
#include<stdio.h>
main( )
{  int i=1,s=0;
    for( ;i<=99; )
        {s=s+i;i++;}
     printf(“s=%d”,s);
}
```

体现式2也能够省略但在循环体中要借助break;语句来实现循环的结束，我们将在背面简介。

for语句中的体现式能够是一切形式的体现式，逗号运算符参加的体现式也能够利用在for语句中，一般利用于体现式1和体现式3。

如上面的例子能够改写为：

```
#include<stdio.h>
main( )
{  int i,s;
    for(s=0,i=1;i<=99;s=s+i,i++);
    printf("s=%d",s);
}
```

注意此处的分号。

此处，体现式1用逗号体现式的形式，给多种变量赋初值。体现式3用逗号体现式把循环体也写入其中。注意体现式3书写顺序不能互换。

例：求 1/100+2/99+…+1

分析：用变量i从1开始循环，每次增长1；用变量j从100开始循环，每次降低1。累加i/j的值到s中。当i>j时结束循环（即i<=j时继续循环）。最终输出s。

程序如下：

```
#include<stdio.h>

main( )

{ int i,j;

  float s=0;

  for( i=1,j=100 ; i<=j ; i++,j--)

      s=s+(float)i/j;

  printf("\ns=%f",s);

}
```

for、while、do-while的比较

全部需要用到循环构造的程序，都能够用for、while、do-while中的任何一种来实现，区别只在于某些问题用哪种语句更以便。

例如求1+2+…+99的问题我们也能够分别用while与do-while语句编写如下：

用while:

```
#include<stdio.h>
main( )
{ int i=1,s=0;
  while(i<=99)
     {s=s+i;  i++; }
  printf("\ns=%d",s);
}
```

用do-while：

```
#include<stdio.h>
main( )
{ int i=1,s=0;
  do{s=s+i;
       i++; }while(i<=99);
  printf("\ns=%d",s);
}
```

例：任意输入一种自然数，把它反序输出。（如：原数为123，输出321）。

分析：此题不拟定循环执行的次数，也不涉及一种规律变化的变量，一般用while或do-while来编写。又因为第一次就要判断输入的是否是自然数，一般用while来实现。算法环节如下：

1. 定义整型变量a用于存储输入的自然数，定义t初值为0用于存储a的反序数，定义i用于依次存储求出的a的每一位的数值。
2. 输入一种自然数赋值给变量a
3. 若a>0,执行第4步，不然执行第7步
4. i=a%10
5. t=t*10+i
6. a=a/10，并返回第3步
7. 输出t

程序：

```
#include<stdio.h>
main( )
{  long  a,i,t=0;
   scanf(“%ld”,&a);
   while(a>0)
      {   i=a%10;
          t=t*10+i;
          a=a/10;   }
   printf(“\n%ld”,t);
}
```

在这里因为a的值可能很大所以用到了long型定义变量a,假如希望取到的值更大，能够用unsigned long型。

问：假如a用double型，并把i=a%10改为i=(long)a%10;把a=a/10改为a=(long)a/10 能够吗？

答：不能够！！

例：有数列2/3、4/5、6/9、10/15……求此数列前30项的和。

算法分析：

对于数列的题，首先要找出通项公式，或前后项的计算关系公式，根据公式求所需。因为数列的题一般执行次数能拟定，用for语句来编写比较以便。

此题，前后项的关系是：后一项的分子是前一项的分母加1，后一项的分母是前一项的分子加分母。解题思绪是用循环语句求各项，并把值累加，因为是求前30项的和，循环执行30次。

1. 初值i=2,j=3,s=0;
2. 用n从1到30循环
    3. s=s+ i/j;
    4. c=i; i=j+1; j=c+j;
5. 输出s;

程序：

```
#include<stdio.h>
main( )
{  int i=2,j=3,n,c;
   float  s=0;
   for(n=1;n<=30;n++)
        {   s=s+(float)i/j;
            c=i;
            i=j+1;
            j=c+j;
        }
   printf("\n%f",s);
}
```

此题中的n与循环体中的执行语句没有数值上的联络，仅仅用做决定循环执行的次数。

5.4 break、continue、goto语句

- 此类语句的功能是使程序从其所在的位置转向另一处。
- goto语句使程序的构造性和可读性都变差，要求尽量防止使用，此处不做简介。

5.4.1 break语句

一般形式：

```
break;
```

它的作用是把流程转向所在构造之后。 在switch分支构造中，使用break语句能够使流程跳出switch分支构造。 一样的，在循环构造中，使用break语句使流程跳出目前的循环层,转向执行该循环构造背面的语句。

例：前面讲到的计算1+2+…+99的程序，能够同步省略for循环的三个体现式，改写成如下形式：

```
main( )
{
  int s=0,i=1:
  for (; ;)
  {  if (i>99)     break;

     s = s+i;      i++;
   }
  printf("s=%d",s);
}
```

本程序中，当i>99时，利用break语句强行终止for循环，继续执行for语句后的下一条语句。

5.4.2 continue语句

一般形式：

continue;

该语句被称为继续语句。在循环构造中执行continue语句,使此次循环提前结束，即跳过循环体中continrue语句下面的还未执行的循环体语句，但不结束整个循环，继续进行下一次循环的条件鉴别，条件为真，继续进行执行循环语句。

例：下面这个程序，想想它实现的是什么功能？

```
#include<stdio.h>
main( )
{  int i,s=0;
   for(i=1;i<=100;i++)
      {if(i%5==0) continue;
       s=s+i;
      }
   printf("\n%d",s);
}
```

在左边的程序中，i从1到100循环，当i是5的倍数时，直接进入下一种i,当i不是5的倍数时，把i累加到s,最终输出s。所以，这个程序实现的是求1~100中间全部非5的倍数的数之和。

5.5 循环的嵌套

循环体语句能够是任何形式的语句，简朴语句、空语句、复合语句、流程控制语句都可作为循环体语句。

当循环体语句又是一条循环语句，或作为循环体的复合语句中又包括循环语句时称为循环的嵌套。嵌套能够是两层或多层。While、do-while、for三种循环都能够相互嵌套。