计算机导论与程序设计 [CS006001-60]

段江涛

机电工程学院

2019 年 10 月

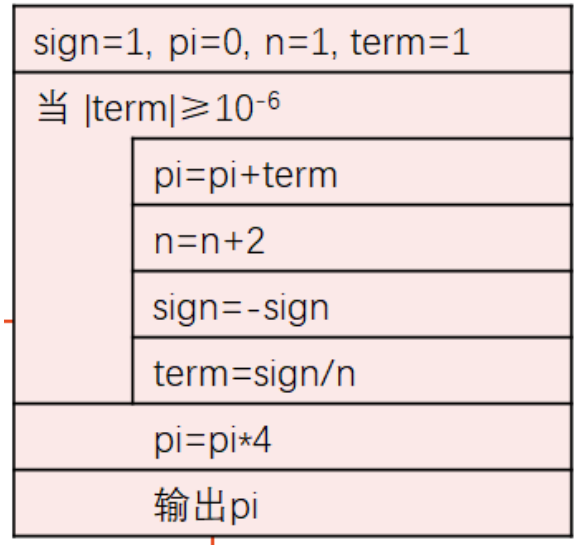
lecture-8 主要内容

循环结构程序设计举例

[例 5.7] 用公式 $\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$ 求 π 的近似值, 直到发现某一项的绝对值小于 10^{-6} 为止 (该项不累加)。

解题思路: 找规律

1. 每项的分子都是 1。
2. 后一项的分母是前一项的分母加 2。
3. 第 1 项的符号为正, 从第 2 项起, 每一项的符号与前一项的符号相反。在每求出一项后, 检查它的绝对值是否大于或等于 10-6。



```c
#include <stdio.h>
#include <math.h> //程序中用到数学函数fabs，应包含头文件math.h
int main()
{
  int sign=1; //sign用来表示数值的符号
  double pi=0.0,n=1.0,term=1.0; //pi开始代表多项式的值，最后代表π的值，n代表分
      母，term代表当前项的值
  while(fabs(term)>=1e-6) //检查当前项term的绝对值是否大于或等于10-6
  {
    pi=pi+term; //把当前项term累加到pi中
    n=n+2; //n+2是下一项的分母
    sign=-sign; //sign代表符号，下一项的符号与上一项符号相反
    term=sign/n; //求出下一项的值term
  }
  pi=pi*4; //多项式的和pi乘以4，才是π的近似值
  printf("pi=%10.8f\n",pi); //输出π的近似值
  return 0;
}
```

[例 5.8] 求 Fibonacci(斐波那契) 数列的前 40 个数。这个数列有如下特点: 第 1, 2 两个数为 1, 1。从第 3 个数开始, 该数是其前面两个数之和。即该数列为 1,1,2,3,5,8,13,..., 用数学方式表示为:

$$\begin{cases} F_1 = 1 & (n = 1) \\ F_2 = 1 & (n = 2) \\ F_n = F_{n-1} + F_{n-2} & (n \geq 3) \end{cases}$$

这是一个有趣的古典数学问题: 有一对兔子, 从出生后第 3 个月起每个月都生一对兔子。小兔子长到第 3 个月后每个月又生一对兔子。假设所有兔子都不死, 问每个月的兔子总数为多少?

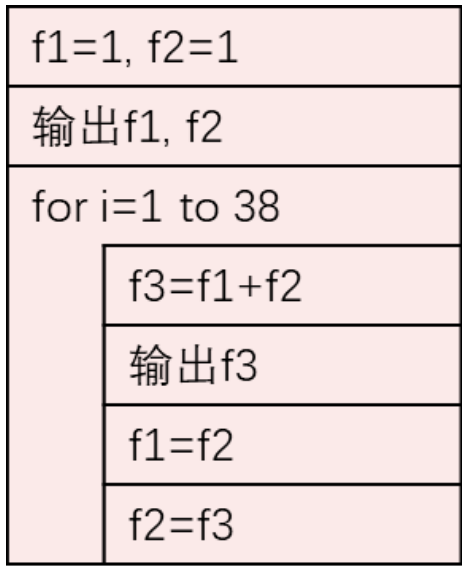
兔子繁殖的规律

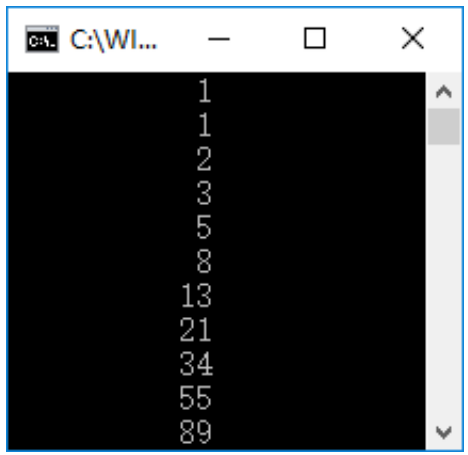
月数	小兔子对数	中兔子对数	老兔子对数	兔子总对数
1	1	0	0	1
2	0	1	0	1
3	1	0	1	2
4	1	1	1	3
5	2	1	2	5
6	3	2	3	8
7	5	3	5	13
⋮	⋮	⋮	⋮	⋮

不满 1 个月的为小兔子, 满 1 个月不满 2 个月的为中兔子, 满 2 个月以上的为老兔子。

解法一: 利用递推 (迭代) 公式: $F_1 = F_2 = 1; F_3 = F_1 + F_2; F_1 = F_2; F_2 = F_3;$

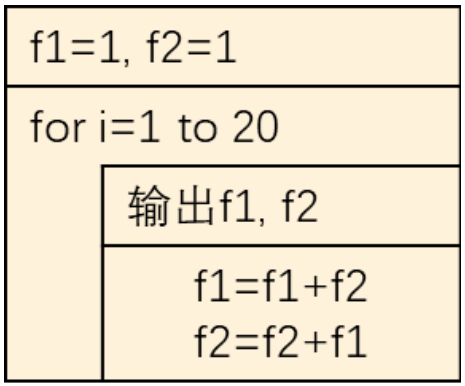
```
#include <stdio.h>
int main()
{
    int f1=1,f2=1,f3;
    int i;
    printf("%12d\n%12d\n",f1,f2);
    for(i=1; i<=38; i++)
    {
        f3=f1+f2;
        printf("%12d\n",f3);
        f1=f2;
        f2=f3;
    }
    return 0;
}
```

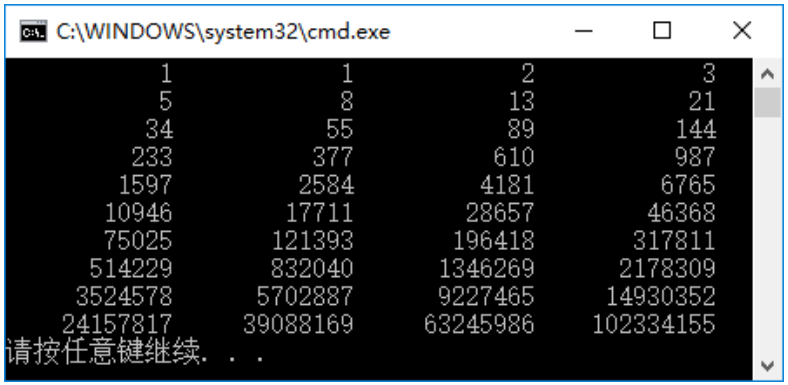




解法二: 利用递推 (迭代) 公式: $F_1 = F_2 = 1; F_1 = F_1 + F_2; F_2 = F_1 + F_2;$

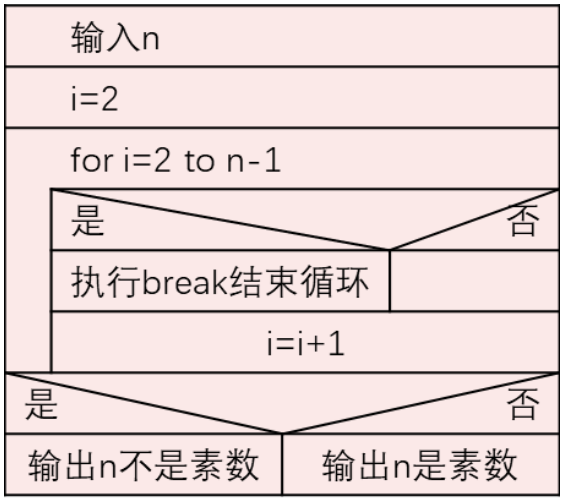
```
#include <stdio.h>
int main()
{
  int f1=1,f2=1;
  int i;
  for(i=1; i<=20; i++)
  {
    printf("%12d%12d",f1,f2);
    if(i%2==0) // 等效 if(!(i%2))
       printf("\n");
    f1=f1+f2;
    f2=f2+f1;
  }
  return 0;
}
```





[例 5.9] 输入一个大于 3 的整数 n,判定它是否为素数 (prime,又称质数)。

```
#include <stdio.h>
int main()
{
  int n,i;
  printf("please␣enter␣a␣integer␣number,n=?");
  scanf("%d",&n);
  for (i=2;i<n;i++)
    if(n%i==0) break;
  if(i<n) // for提前结束
    printf("%d␣is␣not␣a␣prime␣number.\n",n);
  else // for正常结束
    printf("%d␣is␣a␣prime␣number.\n",n);
  return 0;
}
```



只要在循环结束后检查循环变量 i 的值,就能判定循环是提前结束还是正常结束的。从而判定 n 是否为素数。这种判断循环结束的方法以后会常用到。

优化: n 不必被 $2 \sim (n-1)$ 内的各整数去除, 只须将 n 被 $2 \sim \sqrt{n}$ 之间的整数除即可。因为 n 的每一对因子, 必然有一个小于 n, 另一个大于 n。

```
#include <stdio.h>
#include <math.h>
int main()
{
  int n,i,k;
  printf("please␣enter␣a␣integer␣number,n=?");
  scanf("%d",&n);
  k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
  for (i=2;i<=k;i++)
    if(n%i==0) break;
  if(i<=k)
    printf("%d␣is␣not␣a␣prime␣number.\n",n);
  else
    printf("%d␣is␣a␣prime␣number.\n",n);
  return 0;
}
```

使用标志变量, 判断循环结束条件。

```
int n,i,k,flag=1; // flag: 标志变量
k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
for (i=2;i<=k;i++)
  if(n%i==0) { flag=0; break; }
if(!flag) // for提前结束
  printf("%d is not a prime number.\n",n);
else // for正常结束
  printf("%d is a prime number.\n",n);
```

```
int n,i,k,flag=1; // flag: 标志变量
k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
for (i=2;i<=k && flag;i++)
  if(n%i==0) { flag=0; }
if(!flag)
  printf("%d is not a prime number.\n",n);
else
  printf("%d is a prime number.\n",n);
```

使用标志变量, 判断循环结束条件。

```
int n,i,k,flag=1; // flag: 标志变量
k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
for (i=2;i<=k;i++)
  if(n%i==0) { flag=0; break; }
if(!flag) // for提前结束
  printf("%d is not a prime number.\n",n);
else // for正常结束
  printf("%d is a prime number.\n",n);
```

```
int n,i,k,flag=1; // flag: 标志变量
k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
for (i=2;i<=k && flag;i++)
  if(n%i==0) { flag=0; }
if(!flag)
  printf("%d is not a prime number.\n",n);
else
  printf("%d is a prime number.\n",n);
```

[例 5.10] 求 100 ～ 200 间的全部素数。

```
#include <stdio.h>
#include <math.h>
int main()
{
  int n,i,k;
  for (n=101;i<=200;n+=2) //n从101变化到200, 对每个奇数n进行判定
  {
    k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
    for(i=2;i<=k;i++)
      if(n%i==0) break;
    if(i>k)
      printf("%d␣",n);
  }
  printf("\n");
  return 0;
}
```

[例 5.10] 求 100 ∼ 200 间的全部素数。

```
#include <stdio.h>
#include <math.h>
int main()
{
  int n,i,k;
  for (n=101;i<=200;n+=2) //n从101变化到200, 对每个奇数n进行判定
  {
    k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
    for(i=2;i<=k;i++)
      if(n%i==0) break;
    if(i>k)
      printf("%d␣",n);
  }
  printf("\n");
  return 0;
}
```

[例 5.11] 译密码。为使电文保密,往往按一定规律将其转换成密码,收报人再按约定的规律将其译回原文。例如,可以按以下规律将电文变成密码: 将字母 A 变成字母 E,a 变成 e,即变成其后的第 4 个字母,W 变成 A,X 变成 B,Y 变成 C,Z 变成 D。

```
char c;
c=getchar(); //输入一个字符给字符变量c
while(c!='\n') //检查c的值是否为换行符'\n'
{
  if((c>='a' && c<='z') || (c>='A' && c<='Z')) //c如果是字母
  {
    if((c>='W' && c<='Z') || (c>='w' && c<='z')) c = c-22; //如果是26个字
        母中最后4个字母之一就使c-22
    else c =c + 4; //如果是前面22个字母之一, 就使c + 4
  }
}
printf("%c",c); //输出已改变的字符
c=getchar(); //再输入下一个字符给字符变量c
}
printf("\n");
```

[例 5.11] 译密码。为使电文保密,往往按一定规律将其转换成密码,收报人再按约定的规律将其译回原文。例如,可以按以下规律将电文变成密码: 将字母 A 变成字母 E,a 变成 e,即变成其后的第 4 个字母,W 变成 A,X 变成 B,Y 变成 C,Z 变成 D。

```
char c;
c=getchar(); //输入一个字符给字符变量c
while(c!='\n') //检查c的值是否为换行符'\n'
{
  if((c>='a' && c<='z') || (c>='A' && c<='Z')) //c如果是字母
  {
    if((c>='W' && c<='Z') || (c>='w' && c<='z')) c = c-22; //如果是26个字
        母中最后4个字母之一就使c-22
    else c =c + 4; //如果是前面22个字母之一, 就使c + 4
  }
}
printf("%c",c); //输出已改变的字符
c=getchar(); //再输入下一个字符给字符变量c
}
printf("\n");
```

在循环条件中接收输入的字符是一种常见技巧。

```
char c;
while((c=getchar())!='\n') //检查c的值是否为换行符'\n'
{
   if((c>='a' && c<='z') || (c>='A' && c<='Z')) //c如果是字母
   {
      if((c>='W' && c<='Z') || (c>='w' && c<='z')) c = c-22; //如果是26个
          字母中最后4个字母之一就使c-22
      else c =c + 4; //如果是前面22个字母之一，就使c + 4
   }
   printf("%c",c); //输出已改变的字符
}
printf("\n");
```

欢迎批评指正！