循环

2020年7月10日 19:43

程序写的是步骤，不是关系！！！要一步一步认识计算机是怎么执行的！！

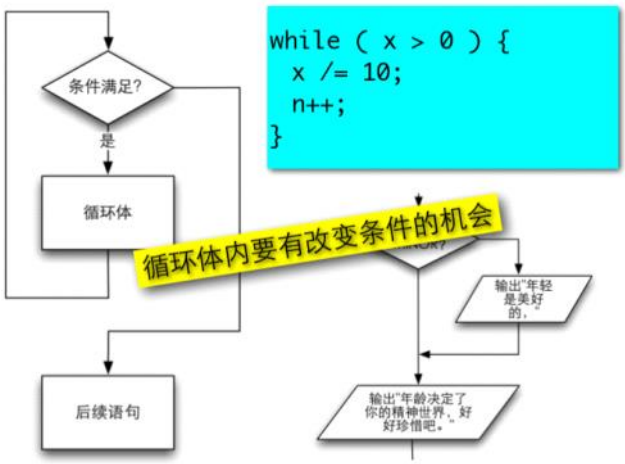
```
int x;
int n =1;

scanf("%d", &x);

if ( x > 999 ) {
    n = 4;
} else if ( x > 99 ) {
    n = 3;
} else if ( x > 9 ) {
    n = 2;
}

printf("%d\n", n);
```

- 因为题目明确了4位数及以下的正整数，所以可以简化一些判断
- 因为从高处往下判断，所以不需要判断上限了
  - 反过来不行
- 问题：任意范围的正整数怎么办？

while循环

```
if ( x > 0 ) {
  x /= 10;
  n++;
}
```

```
while ( x > 0 ) {
  x /= 10;
  n++;
}
```




```
while ( x > 0 ) {
  x /= 10;
  n++;
}
```

- 如果我们把while翻译作“当”，那么一个while循环的意思就是：当条件满足时，不断地重复循环体内的语句。
- 循环执行之前判断是否继续循环，所以有可能循环一次也没有被执行；
- 条件成立是循环继续的条件。

- 测试程序常使用边界数据，如有效范围两端的数据、特殊的倍数等
  - 个位数；
  - 10；
  - 0；
  - 负数。

数位数的算法

1. 用户输入x；
2. 初始化n为0；
3. x = x / 10，去掉个位；
4. n ++；
5. 如果x>0，回到3；
6. 否则n就是结果。

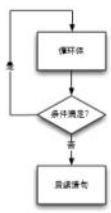
```
int x;
int n =0;

scanf("%d", &x);

n++;
x /= 10;
while ( x > 0 ) {
    n++;
    x /= 10;
}

printf("%d\n", n);
```

不能去掉!

do-while循环

- 在进入循环的时候不做检查，而是在执行完一轮循环体的代码之后，再来检查循环的条件是否满足，如果满足则继续下一轮循环，不满足则结束循环

```
do
{
    <循环体语句>
} while ( <循环条件> );
```



- do-while循环和while循环很像，区别是在循环体执行结束的时候才来判断条件。也就是说，无论如何，循环都会执行至少一遍，然后再来判断条件。与while循环相同的是，条件满足时执行循环，条件不满足时结束循环。

```
int x;
scanf("%d", &x);
int n = 0;
do
{
  x /= 10;
  n ++;
} while ( x > 0 );
printf("%d", n);
```

小套路

计算之前先保存原始的值，后面可能有用

如果要模拟运行一 个很大次数的循环，可以模拟较少的循环次数，然后作出推断

猜数游戏

- 让计算机来想一个数，然后让用户来猜，用户每输入一个数，就告诉它是大了还是小了，直到用户猜中为止，最后还要告诉用户它猜了多少次。
- 因为需要不断重复让用户猜，所以需要用到循环
- 在实际写出程序之前，我们可以先用文字描述程序的思路
- 核心重点是循环的条件
  - 人们往往会考虑循环终止的条件

```
srand(time(0));
int number = rand()%100+1;
int count = 0;
int a = 0;
printf("我已经想好了一个1到100之间的数。");
do {
  printf("请猜这个1到100之间数：");
  scanf("%d", &a);
  count ++;
  if ( a > number ) {
    printf("你猜的数大了。");
  } else if ( a < number ) {
    printf("你猜的数小了。");
  }
} while (a != number);
printf("太好了，你用了%d次就猜到了答案。\n", count);
```

for (初始动作; 条件; 每轮的动作) {

}

- for中的每一个表达式都是可以省略的

for (; 条件;) == while (条件)

Tips for loops

- 如果有固定次数，用for
- 如果必须执行一次，用do_while
- 其他情况用while

break vs continue

判断素数

```
} while ( x > 0 );
printf("%d", n);
```

小心

1. 计算机随机想一个数，记在变量number里；
2. 一个负责计次数的变量count初始化为0；
3. 让用户输入一个数字a；
4. count递增（加一）；
5. 判断a和number的大小关系，“大”；如果a小就输出“小”；
6. 如果a和number是不相等的（无论大还是小），程序转回到第3步；
7. 否则，程序输出“猜中”和次数，然后结束。

循环的条件是a和number不相等

随机数

每次召唤rand()就得到一个随机整数

x % n 的结果是[0, n-1]的一一个整数

for循环

for循环像一个计数循环：设定一个计数器，初始化它，然后在计数器到达某值之前，重复执行循环体，而每执行一轮循环，计数器值以一定步进进行调整，比如加1或者减1

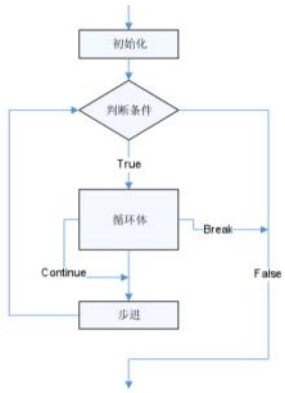
```
for ( i=0; i<5; i=i+1 ) {
  printf("%d", i);
}
```

- 做求和的程序时，记录结果的变量应该初始化为0，而做求积的变量时，记录结果的变量应该初始化为1
- 循环控制变量i只在循环里被使用了，在循环外面它没有任何用处。因此，我们可以把变量i的定义写到for语句里面去

```
int n;

scanf("%d", &n);
int fact = 1;
for ( int i=1; i<=n; i++ ) {
    fact *= i;
}

printf("%d!=%d\n", n, fact);
```

C99 ONLY!

```
int main()
{
    int x;
```



判断素数

i<x? ?
如果输入的是一个极大值的素数，就会极耗运算时间。
效率降低
可读性变差

```c
int main()
{
    int x;

    scanf("%d", &x);

    int i;
    int isPrime = 1;    //  x是素数
    for ( i=2; i<x; i++ ) {
        if ( x % i == 0 ) {
            isPrime = 0;
            break;
        }
    }
    if ( isPrime == 1 ) {
        printf("是素数\n");
    } else {
        printf("不是素数\n");
    }
    return 0;
}
```

50个素数　一个计数器

```c
int x;
int cnt = 0;

// for ( x=1; cnt<50; x++ ) {
x = 1;
while ( cnt <50 ) {
    int i;
    int isPrime = 1;    //  x是素数
    for ( i=2; i<x; i++ ) {
        if ( x % i == 0 ) {
            isPrime = 0;
            break;
        }
    }
    if ( isPrime == 1 ) {
        cnt ++;
        printf("%d\t", x);
        if ( cnt %5 == 0 ) {
            printf("\n");
        }
    }
```

- 如何用1角、2角和5角的硬币凑出10元以下的金额呢？

接力break

```c
int x;
int one, two, five;
int exit = 0;

scanf("%d", &x);
for ( one = 1; one < x*10; one++ ) {
    for ( two = 1; two < x*10/2; two++ ) {
        for ( five = 1; five < x*10/5; five++ ) {
            if ( one + two*2 + five*5 == x*10 ) {
                printf("可以用%d个1角加%d个2角加%d个5角得到%d元\n",
                    one, two, five, x);
                exit = 1;
                break;
            }
        }
        if ( exit == 1 ) break;
    }
    if ( exit == 1 ) break;
}
```

设置一个标志int exit = 0;，然后满足某种条件后，改变标志exit = 1，进而改变程序控制流程

goto

```c
int x;
int one, two, five;

scanf("%d", &x);
for ( one = 1; one < x*10; one++ ) {
    for ( two = 1; two < x*10/2; two++ ) {
        for ( five = 1; five < x*10/5; five++ ) {
            if ( one + two*2 + five*5 == x*10 ) {
                printf("可以用%d个1角加%d个2角加%d个5角得到%d元\n",
                    one, two, five, x);
                goto out;
            }
        }
    }
}
out:
    return 0;
```

少用goto，在这种循环嵌套情况下跳出多层循环时用

$$f(n) = 1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\ldots++\frac{1}{n}$$

$$f(n) = 1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\ldots+\frac{1}{n}$$

```c
int n;
int i;
double ret=0.0;

scanf("%d", &n);
for ( i=1; i<=n; i++ ) {
    ret += 1.0/i;
}
printf("%f\n", ret);
```

注意数据类型!

```c
int n;
int i;
double ret=0.0;
int sign = 1;

scanf("%d", &n);
for ( i=1; i<=n; i++ ) {
    ret += 1.0*sign/i;
    sign = -sign;
}
printf("%f\n", ret);
```

正序分解整数

- 输入一个非负整数，正序输出它的每一位数字
- 输入：13425
- 输出：1 3 4 2 5

```c
int x;
```

注意数据类型!

```
int x;
scanf("%d", &x);

do {
    int d = x % 10;
    printf("%d", d);
    if ( x>9 ) {
        printf(" ");
    }
    x /= 10;
} while ( x > 0 );
printf("\n");
```

- 但是是逆序的!

```
int x;
scanf("%d", &x);

x = 13425;
int mask = 10000;
do {
    int d = x / mask;
    printf("%d", d);
    if ( mask > 9 ) {
        printf(" ");
    }
    x %= mask;
    mask /= 10;
} while ( mask>0 );

printf("\n");
```

- 如果能有这么一个mask

```
x = 12345;
int mask = 10000;
int n=0;
do {
    x /= 10;
    n++;
} while ( x > 0 );
printf("n=%d\n", n);
```

- 计算x的位数

```
x = 12345;
int mask = 10000;
int n=0;
do {
    x /= 10;
    n++;
} while ( x > 0 );
printf("n=%d\n", n);
mask = pow(10,n);
printf("mask=%d\n", mask);
```

- pow?
- #include <math.h>
- pow是浮点运算，慢

```
x = 12345;
int mask = 1;
do {
    x /= 10;
    mask *=10;
} while ( x > 0 );
printf("mask=%d\n", mask);
```

- 直接算mask
- mask=100000?
- 因为第一轮mask就是10了
- 怎么办?

```
x = 12345;
int mask = 1;
do {
    x /= 10;
    mask *=10;
} while ( x > 9 );
```

- 改变循环的条件，让它少做一轮

```
x = 12345;
int mask = 1;
do {
    x /= 10;
    mask *=10;
} while ( x > 9 );
```

- 改变循环的条件，让它少做一轮
- 但是最后的结果为什么不对?

```
mask=10000
00001[Finished in 0.1s]
```

```
int x;
scanf("%d", &x);

x = 13425;
int mask = 1;
int t = x;
do {
    t/=10;
    mask *= 10;
} while (t>9);
printf("mask=%d\n", mask);
do {
    int d = x / mask;
    printf("%d", d);
    if ( mask > 9 ) {
        printf(" ");
    }
    x %= mask;
    mask /= 10;
} while ( mask>0 );
```

- 因为x在第一个循环中被改变了
- 需要用另外的变量代替x做计算

```
int x;
scanf("%d", &x);

x = 13425;
int mask = 1;
int t = x;
while ( t>9 ) {
    t /= 10;
    mask *=10;
}
printf("x=%d, mask=%d\n", x, mask);
do {
    int d = x / mask;
    printf("%d", d);
    if ( mask > 9 ) {
        printf(" ");
    }
    x %= mask;
    mask /= 10;
} while ( mask > 0 );
printf("\n");
```

```
int a,b;
int min;

scanf("%d %d", &a, &b);
if ( a<b ) {
    min = a;
} else {
    min = b;
}
int ret = 0;
int i;
for ( i = 1; i < min; i++ ) {
    if ( a%i == 0 ) {
        if ( b%i == 0 ) {
            ret = i;
        }
    }
}
printf("%d和%d的最大公约数是%d.\n", a, b, ret);
```

枚举

1. 设t为2;
2. 如果u和v都能被t整除，则记下这个t
3. t加1后重复第2步，直到t等于u或v;
4. 那么，曾经记下的最大的可以同时整除u和v的t就是gcd

辗转相除法

1. 如果b等于0，计算结束，a就是最大公约数；
2. 否则，计算a除以b的余数，让a等于b，而b等于那个余数；
3. 回到第一步。

辗转相除法

```c
int a,b;
int t;

scanf("%d %d", &a, &b);
int origa = a;
int origb = b;
while ( b != 0 ) {
    t = a%b;
    a = b;
    b = t;
}
printf("%d和%d的最大公约数是%d.\n", origa, origb, a);
```

注意

1. 注意边界情况
2. 循环好好分析，注意执行过程，循环的条件，循环的边界，循环的标记