# 循环

# 数数几位数

- 程序要读入一个4位以下（含4位）的正整数，然后输出这个整数的位数。如：
- 输入：352，输出：3

# 人vs计算机

- 人的方式：眼睛一看就知道了
  - 352 —> 3位！
- 计算机的方式：判断数的范围来决定它的位数
  - 352∈[100,999] —> 3位
  - 人不擅长，因为人对数字的计算能力比文字弱

# 程序实现

```
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?

```c
int x;
int n =0;

scanf("%d", &x);

if ( x > 999 ) {
    n = 4;
} else if ( x > 99 ) {
    n = 3;
} else if ( x > 9 ) {
    n = 2;
} else if ( x > 0 ) {
    n = 1;
}

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

```c
int x;
int n =0;

scanf("%d", &x);

if ( x > 9999 ) {
    n = 5;
} else if ( x > 999 ) {
    n = 4;
} else if ( x > 99 ) {
    n = 3;
} else if ( x > 9 ) {
    n = 2;
} else if ( x > 0 ) {
    n = 1;
}

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

如何才是个头啊！！！

```c
int x;
int n =0;

scanf("%d", &x);

if ( x > 99999 ) {
    n = 6;
} else if ( x > 9999 ) {
    n = 5;
} else if ( x > 999 ) {
    n = 4;
} else if ( x > 99 ) {
    n = 3;
} else if ( x > 9 ) {
    n = 2;
} else if ( x > 0 ) {
    n = 1;
}

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

# 换个方式想

- 352 —> 3 很快，
  12381284326751827361827361267531 7是
  几位?
- 数数！

# 数数

- 1238128432675182736182736126753l7
- 人怎么数？从左往右数，一次划掉一个数字
- 计算机怎么划掉那个数字？

# 三位数逆序的题

- 352
  - 352 %100 —> 52
- 那么，
  12381284326751827361827361267531 7%
  10000000000000000000000000000000-> 2381284326751827361827361267531 7
- 怎么得到那个
  10000000000000000000000000000000?

# 人vs计算机

- 如果换一下，从右边开始划
- 123812843267518273618273612675317/10->
  12381284326751827361827361267531
- 去掉最右边的数。然后?
- 不断地划，直到没数可以划...
  - 在这个过程中计数

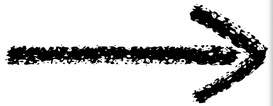
# 试试代码

```c
int x;
int n =0;

scanf("%d", &x);

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

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

- 这事儿还是没完没了…

# 试试代码

```c
int x;
int n =0;

scanf("%d", &x);

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

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



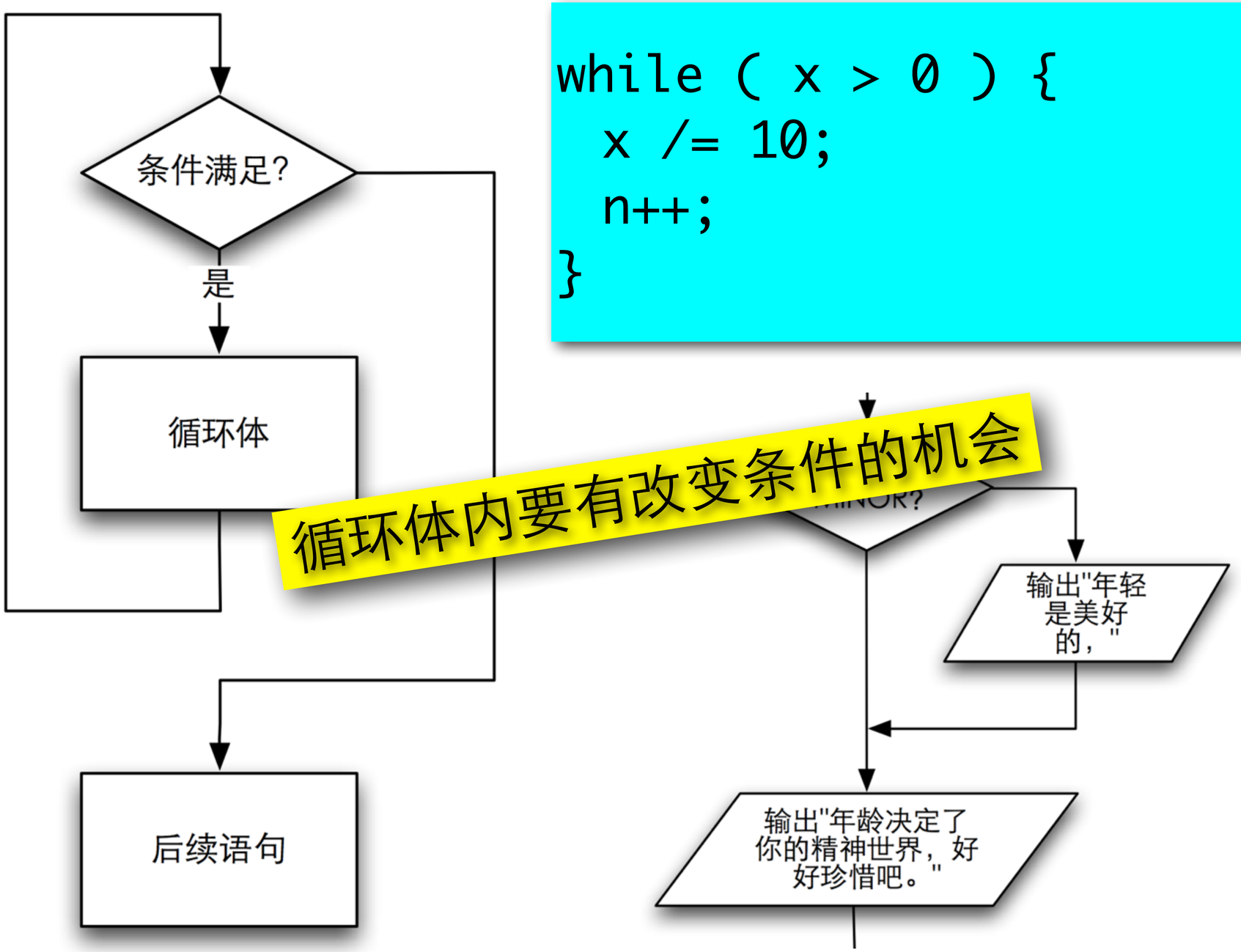
```c
int x;
int n =0;

scanf("%d", &x);

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

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

别拿123812843267518273618273612675317去试！

# while循环

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

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




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

循环体内要有改变条件的机会

# while循环

- 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。

# 再想想

```c
int x;
int n =0;

scanf("%d", &x);

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

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

```c
int x;
int n =0;

scanf("%d", &x);

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

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

- 如果没有外面的运算?

# 看程序运行结果

- 人脑模拟计算机的运行，在纸上列出所有的变量，随着程序的进展不断重新计算变量的值。当程序运行结束时，留在表格最下面的就是程序的最终结果

# 验证

- 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。

# 调试

```c
int x;
int n =0;

scanf("%d", &x);

n++;
x /= 10;
while ( x > 0 ) {
    printf("x=%d, n=%d\n", x, n);
    n++;
    x /= 10;
}

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

- 在程序适当的地方插入printf来输出变量的内容

# do-while循环

# 数位数的算法

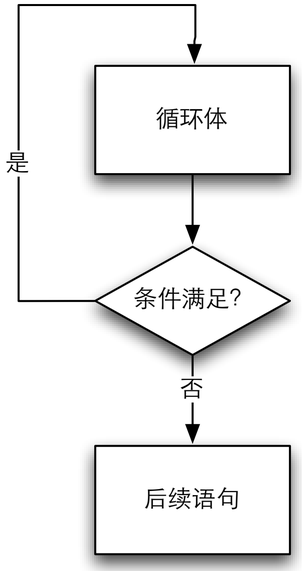
1. 用户输入x；
2. 初始化n为0；
3. x = x / 10，去掉个位；
4. n ++；
5. 如果x>0，回到3；
6. 否则n就是结果。

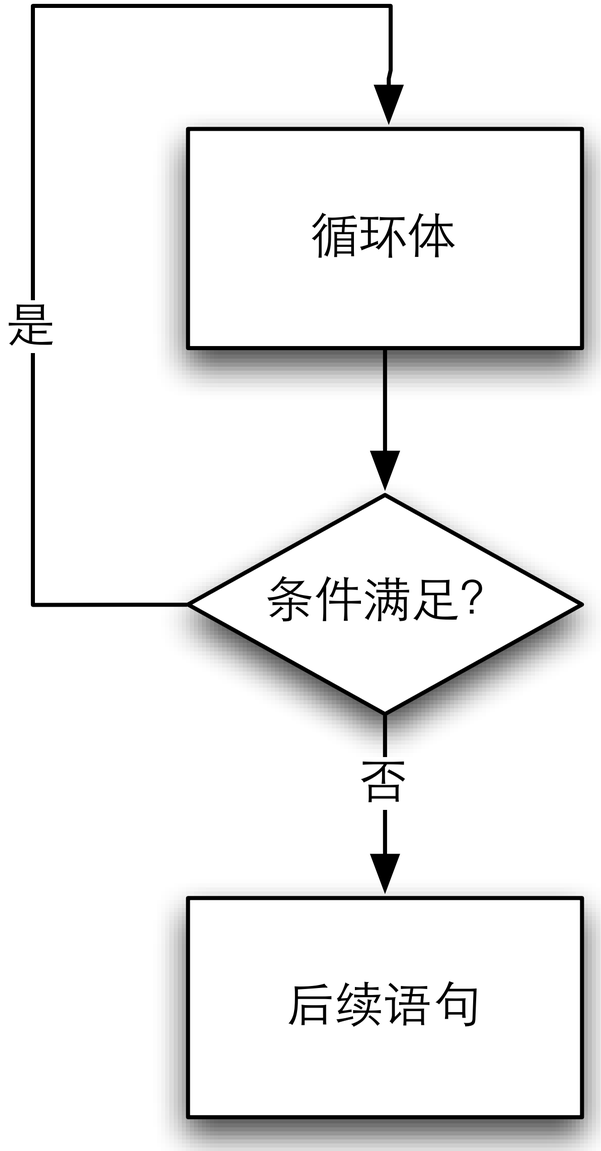
```c
int x;
int n =0;

scanf("%d", &x);

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

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

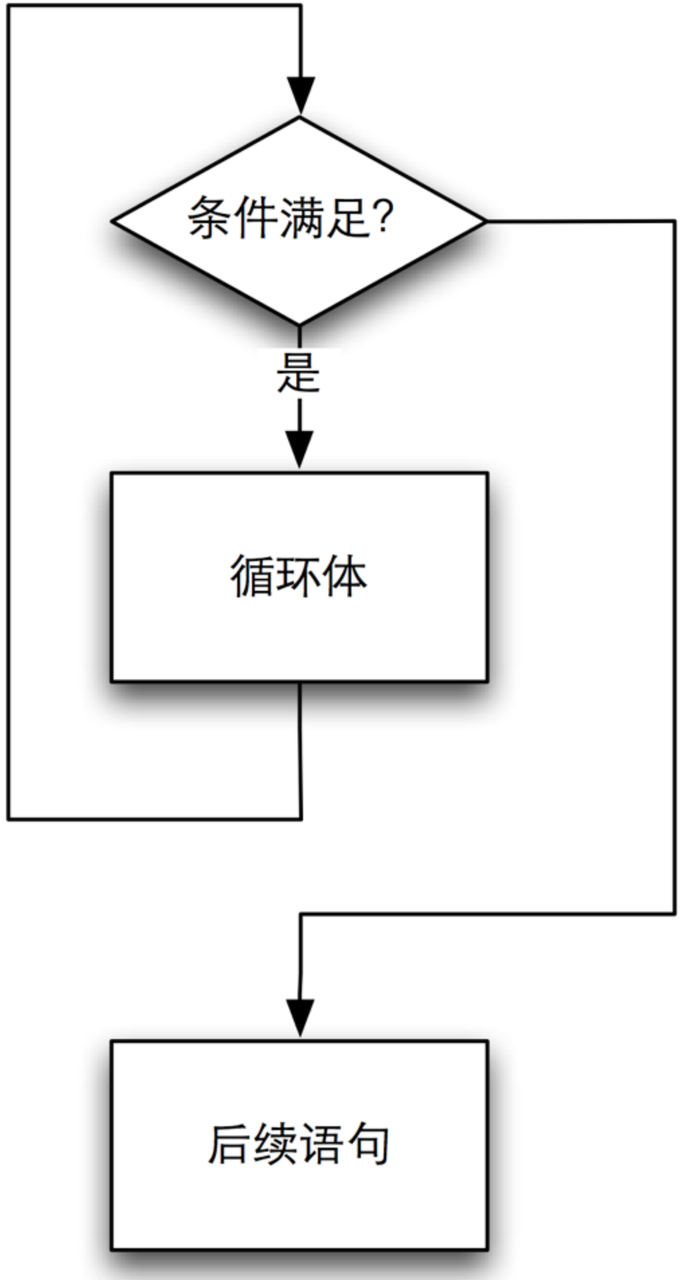
# do-while循环

- 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> );
```




循环体
是
条件满足?
否
后续语句


条件满足?
是
循环体
后续语句

# 两种循环

- 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);
```