第2章 C运算符和表达式

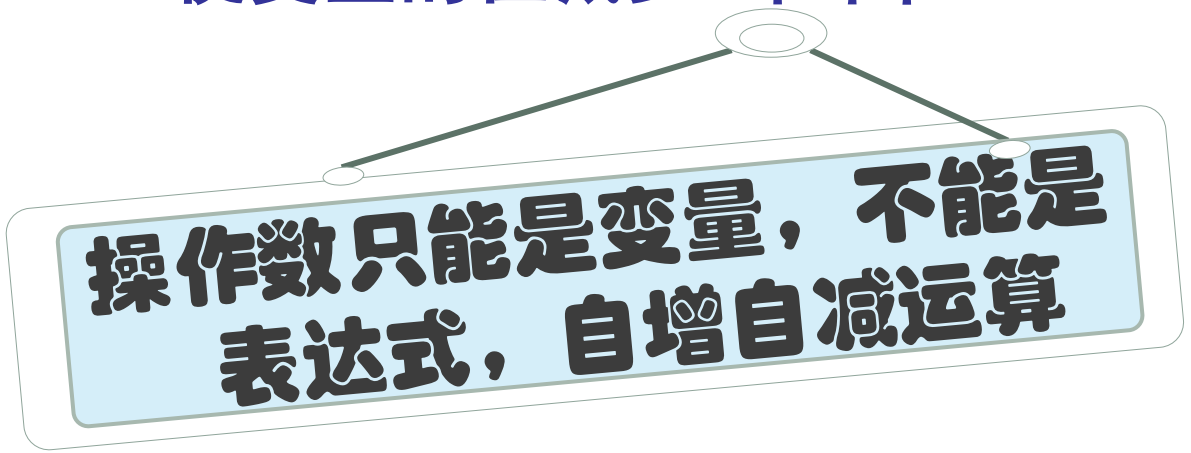
——增1和减1运算符

本节要讨论的主要问题

- 增1和减1运算符作为前缀和后缀运算符时有何不同？
- 为什么良好的程序设计风格不提倡在一个表达式中使用过多的增1和减1运算符？

增1和减1运算符

- 增1运算符(Increment) ++
  - 使变量的值增加1个单位
- 减1运算符(Decrement) --
  - 使变量的值减少1个单位



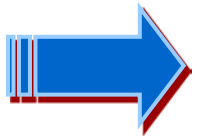
- 一元运算符
  - 前缀(prefix)
    - `++n` → `n = n + 1`
    - `--n` → `n = n – 1`
  - 后缀(postfix)
    - `n++` → `n = n + 1`
    - `n--` → `n = n – 1`

前缀增1/减1运算符

- 作为前缀(prefix)运算符时
  - ++n，--n
  - 先对n增1/减1，然后再使用n的值

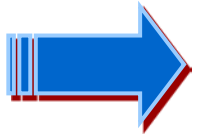
用增1和减1运算生成的代码运行速度更快

```
m = ++n;
```



```
n = n + 1;
m = n;
```

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

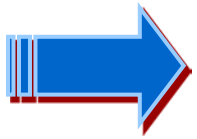


```
n = n + 1;
printf("%d",n);
```

后缀增1/减1运算符

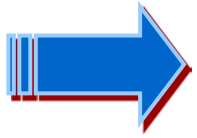
- 作为后缀(postfix)运算符时
  - n++，n--
  - 先使用n的值，然后再对n增1/减1

```
m = n++;
```



```
m = n;
n = n + 1;
```

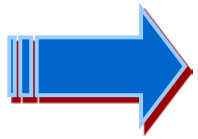
```
printf("%d",n++);
```



```
printf("%d",n);
n = n + 1;
```

前缀与后缀对变量和表达式的影响

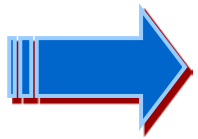
```
m = ++n-2;
```



```
n = n + 1;
m = n - 2;
```

n	6
m	4

```
m = n++-2;
```



```
m = n - 2;
n = n + 1;
```

n	6
m	3

操作数的值是相同的
但表达式的值是不同的

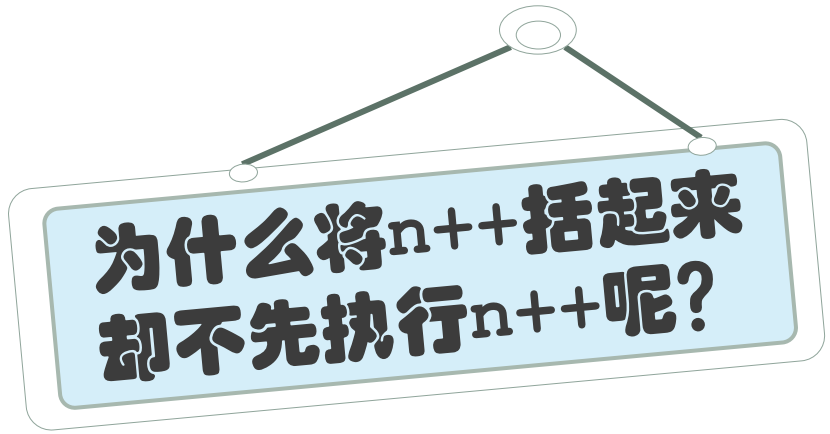
稍微复杂一点的例子

```
printf("%d",-n++);
```

```
printf("%d",-(n++));
```

```
n = n + 1;
```

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



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

```
n = n + 1;
```

n 6

屏幕输出 -5

稍微复杂一点的例子

```
printf("%d",-n++);
```

```
printf("%d",-(n++));
```

```
n = n + 1;
printf("%d",-n);
```

```
printf("%d",(-n)++);
```

```
-n = -n + 1;
```

```
printf("%d",-n);
n = n + 1;
```

n 6

屏幕输出 -5

增1和减1运算符

- 优点
  - 增1和减1运算生成的代码效率更高一些
- 问题：过多的增1和减1运算混合会产生什么结果？
  - 可读性差，例如(++n)+(++n)，(n++)+(n++)
  - 不同编译器产生的运行结果不同
- 良好的程序设计风格提倡
  - 在一行语句中，一个变量只出现一次增1或减1运算