

1.自增运算符

有一个运算符叫自增运算符 符号是 ++

```
1 // 
2 1. 自增运算符。
3
4     ++
5
```

2.自增表达式,由自增运算符组成的表达式

```
6
7
8 2. 自增表达式。
9
10 1).前自增表达式。
11     int num = 1;
12     ++num;
13
14 2).后自增表达式。
15     int num = 1;
16     num++;
```

3.自增表达式的共性

```
17
18 3).无论是前自增表达式还是后自增表达式,都是将自身的值增加1.
19
20
21
22
23
```

1>例子一

```
26 #include <stdio.h>
27
28 int main(int argc, const char * argv[])
29 {
30
31     int num = 1;
32     num++;|
33
34     printf("num = %d\n", num);
35
36
37
38
```

num = 2
Program ended with exit code: 0

```
26 #include <stdio.h>
27
28 int main(int argc, const char * argv[])
29 {
30
31     int num = 1;
32     ++num;
33
34     printf("num = %d\n", num);
35
36
37
38
39
```

Output window:
num = 2
Program ended with exit code: 0

4. 自增表达式的结果

```
20
21
22
23
24
25
26
27
```

3. 自增表达式是1个表达式，既然是1个表达式，那么这个自增表达式就一定有1个结果。
那么我们就可以使用1个变量把这个表达式的结果存储起来。

1>

```
23
24
25
26
27
```

后自增表达式的结果的计算方式：
先将自身的值取出来作为后自增表达式的结果，然后再将自身的值+1

```
37
38 #include <stdio.h>
39
40 int main(int argc, const char * argv[])
41 {
42
43     int i = 1;
44     int j = i++; // 结果是1，结果算完以后，在讲i的值+1
45
46     printf("i = %d\n", i);
47     printf("j = %d\n", j);
48 }
```

```
i = 2
j = 1
Program ended with exit code: 0
```

2>

```
06-交换两个变量的值 27  
07-算术运算符 28  
08-复合赋值运算符 29  
09-自增与自减运算 30  
c main.c 31  
Products 32  
06-交换两个变量的值 33
```

前自增表达式的结果的计算方式：
先将自身的值+1，然后再将自身的值取出来做为表达式的结果。|

```
01-复习 39 #include <stdio.h>  
02-数据类型转换 40  
03-scanf函数的使用 41 int main(int argc, const char * argv[]){  
04-scanf函数的其他用法 42  
05-scanf函数的缓冲区 43  
06-交换两个变量的值 44 int i = 1;  
07-算术运算符 45 //int j = i++; //结果是1 结果算完以后，在讲i的值+1  
08-复合赋值运算符 46  
09-自增与自减运算 47  
c main.c 48 int j = ++i;  
Products 49  
50 printf("i = %d\n",i);  
51 printf("j = %d\n",j);  
52  
53  
54  
55  
56  
57  
58
```

```
i = 2  
j = 2  
Program ended with exit code: 0
```

5. 复杂练习

1>一个注意点

```
02-数据类型转换 31  
03-scanf函数的使用 32  
04-scanf函数的其他用法 33  
05-scanf函数的缓冲区 34  
06-交换两个变量的值 35  
07-算术运算符 36
```

4. 自增运算符的优先级比算术运算符的优先级要高。

2>

```
06-交换两个变量的值 38  
07-算术运算符 39 #include <stdio.h>  
08-复合赋值运算符 40  
09-自增与自减运算 41 int main(int argc, const char * argv[]){  
c main.c 42  
Products 43  
44  
45 int i = 1;  
46 int j = 2;  
47  
48 int k = i++ + ++j + i++; ⚠ Multiple unsequenced modifications to i  
49  
50 printf("k = %d\n",k);  
51  
52
```

```
k = 6  
Program ended with exit code: 0
```

3>

The screenshot shows a C IDE interface with a project tree on the left containing chapters like "01-复习" through "09-自增与自减运算". The main area displays a C program with several self-incrementing statements. A warning message "Multiple unsequenced modifications to 'i'" is shown above the code at line 47. The code is as follows:

```
40
41 int main(int argc, const char * argv[])
42 {
43
44     int i = 1;
45
46
47     int j = i++ + ++i + i++ + ++i + i++;
48 //   printf("j = %d\n",j);
49
50
51
52
53
54
55
56 //   int i = 1;
57 //   int j = 2;
58 //
59 //   int k = i++ + ++j + i++;
60 //
```

The output window below shows the result of the execution: **j = 17**.

6. 自减运算和自增运算原理相同

The screenshot shows a C IDE interface with a project tree on the left containing chapters like "01-复习" through "09-自增与自减运算". The main area displays a C program with several self-decrementing statements. A warning message "Multiple unsequenced modifications to 'i'" is shown above the code at line 52. The code is as follows:

```
43
44 #include <stdio.h>
45
46
47 int main(int argc, const char * argv[])
48 {
49
50     int i = 1, j = 2;
51
52     int k = i++ + --j + j++ + --i;
53
54     printf("k = %d\n",k);
55
56
57
58 //   int i = 1;
59 //   //--i;
60 //
```

The output window below shows the result of the execution: **k = 4**. The status bar at the bottom indicates "Program ended with exit code: 0".

@"无色大师的笔记"