BADJI MOKHTAR UNIVERSITY

SCIENCE AND TECHNIQUES DEPARTMENT
1st year ST

Course 5: Operators in C

by

Dr. Samira LAGRINI

lagrini.samira83@gmail.com

Course link : https://elearning-facsci.univ-annaba.dz/course/view.php?id=469

Introduction

C language has a wide range of classical operators, such as arithmetic, relational and logical operators, and original ones such as compound assignment and increment operators.

1. Assignment operator

- ❑ The assignment is an operator symbolized by the sign =
- ❑ Its syntax is as follows:

$$var = expression$$

- This means that the value of expression is assigned to the variable var
- The left term of the assignment must be a variable but not a constant.

Example:

X=3; means that the value of X is 3

X=X-2 ; means that the value of X becomes (3-2) therefore X=1 (the old value is overwritten).

2. Compound assignment operators

- The compound assignment operators are:

  += - = *= /= %=

- In C the two expressions are equivalent:

x+=y ⇔ x=x+y

x*=y ⇔ x=x*y

x/=y ⇔ x=x/y

x%=y ⇔ x=x%y

3. Arithmetic operators

The classic arithmetic operators are:

- ❑ Addition (+)
- ❑ the subtraction (-)
- ❑The multiplication (*)
- ❑ The division (/)
- ❑The modulo (%): gives the remainder of the division (it only applies to integer type operands.)

Note: in C, there is no operator performing the elevation to the power. you must use the pow(x,y) function from the math.h library to calculate X^y

4. Relational operators

operator	meaning
>	strictly superior
>=	Greater than or equal to
<	strictly inferior
<=	less or equal
==	equal to
!=	different

Relational operators in c

Example:

A>= b

A<B

4. Relational operators (2)

- The result of the comparison equals 1 if the comparison is true, and 0 otherwise.
- The four operators (<, <=,>,>=) have the same priority
- The two operators (==, !=) have a lower priority than the previous four.

Example:

A>=b == c<d

is interpreted as follows:

(A>=b) ==(c<d)

5. Logical operators

operator	meaning
&&	takes the value 1 (true) if all expressions are true
||	takes the value 1 (true) if one of the expressions is true
!	Inverts the value of an expression (returns the value 1 if the expression is 0, 0 if it is 1)

Logical operators in C

- ❑ In a logical expression, the evaluation is done from left to right and stops when the final result is determined.
- ❑ The value returned by a logical expression equal to 1 (true) or 0 (false).

Logical operators (2)

Example 1

```
 int x=1 , y=3 , z;
z= ((x >= 0) && (y <= 0))
```
➔z=0

Example 2

```
int x=2 , y=4 , z;
  z= ((x >= 0) | | (y +4<= 0))
```
→z=1

6. Increment and decrement operators ++ et --

- The **increment (++) and decrement (—) operators** in C are unary operators for incrementing and decrementing the numeric values **by 1** respectively.

What is the difference between **X++ et ++X?**

In both cases the variable X will be incremented, however in:

- **X++:** the returned value will be the value of X before incrementing (the old value)
- **++X:** the returned value will be the value of X after the increment (the new value)
- Similarly for --x et x--

6. Increment and decrement operators ++ et --

Example 1

```
int a = 3, b, c;
b = ++a; /*  b = 4 */
c = b++; /* c = 4 et b = 5 */
```

Example 2

```
i=10;
N=++i – 4;
```

This expression increments i by 1; so i=11 and N=11-4=7

Example 3

```
i=10;
N=i++-4
```

In this case i=11 et N= 10-4=6

7. The conditional operator (? :)

In C the expression:

if **(condition)** action1
else action2

Means:

If the condition is true then
execute *action1*
else execute *action 2*

This expression can be rewritten using the conditional operator as follows:

condition ? Action1 : action 2

The conditional operator first evaluates the condition. If the condition is true then an evaluation of action1 will be carried out, otherwise if the condition is false it is action2 which will be evaluated.

7. The conditional operator (? :)

Examples:

min=x<y ? x:y ; ⇔ if (x<y) min=x; else min =y;

Means : if x<y then min=x else min=y

valabsolue = x>0 ? X : -X ; ⇔ if (x>0) valabsolue=x; else valabsolue=-x;

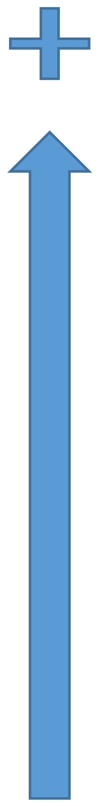
means: if x>o valabsolue=x else valabsolue=-x

X==y? i++:i-- ; ⇔ if (x==y) i++; else i--;

means: if x==y then i=i+1 else i=i-1

8. Operator precedence rules

The following table ranks operators in descending order of priority. Operators placed on the same line have the same priority.



Operator priority order
() []
! ++ --
* / %
+ -
< <= >= >
== !=
&&
| |
? :
%= /= *= -= += =

Exercise: What result does this program provide:

```
# include <stdio.h>
main()
{ int x = 5, y, z, r ;
y = x++; z=++y;
 printf(" y=%d  z=%d  x= %d \t",y, z, x);
 x = 2, y = 4, z = 5;
r = (x < 6 | | y ++ < 3) && (z++ == 6) ;
printf("r = %d  y=%d   z=%d   x=%d ",r, y,  z, x);
x = 2, y = 4, z = 5;
r = (++x > 6 | | y ++ < 3) && (z++ == 8) ;
printf("r = %d  y=%d   z=%d   x=%d ",r, y,  z, x);
}
```

Solution

This program displays:

```
y=6 z= 6 x=6
r=0 y=4 z=6 x=2
r=0 y=5 z=5 x=3
```