Operator in programming C

Operator is a symbol that tells the compiler to perform a certain mathematical operation or logical operation. Language C has a lot of operators and provides operator types.

Operator is a symbol that tells the compiler to perform a certain mathematical operation or logical operation. The C language has many operators available and provides the following operator types:

1. Arithmetic operator
2. Relational operator
3. Logical operators
4. Bit comparison operator
5. Assignment operator
6. Mixed operator

The tutorial will explain arithmetic, relational, logical, bit, assignment and other operators, one by one.

Arithmetic operator in C

The table below shows all arithmetic operators supported by C language. Suppose variable **A** has value 10 and variable **B** has value 20:

OperatorDescription For example + Add two operands A + B that will give the result 30 - Subtract the second operand value from the first operand A - B will result in -10 * Multiply the two operands A * B will result The result is 200 / Divide the integer two operands B / A will give a result of 2% Divide the remainder B% A will give a result of 0 ++ The incremental value of the extra 1 unit of A ++ will give the result The result is 11 - The decrease in the operand value of a unit A - will result in 9

Relational operator

The table below shows all relational operators supported by C language. Suppose that variable **A** has value 10 and variable **B** has value 20, we have:

OperatorDescriptionExample == Check if 2 operands are equal or not. If equal, the condition is true. (A == B) is not correct. ! = Check if two operands have different values. If not, the condition is true. (A! = B) is true. > Check if the left operand is greater than the right operand. If larger, the condition is true. (A> B) is not correct. If it is smaller then true. (A **= Check if the left operand has a value greater than or equal to the value of the right operand. If true, true. (A> = B) is not correct. = Check if the left operand is less than or equal to the right operand. If true, true. (A = B) is true.**

Logical operators

The table below shows all logical operators supported by C language. Suppose variable **A** has value 1 and variable **B** has value 0:

OperatorDescription && Called the logical AND operator (and). If both operators have values ??other than 0, the condition becomes true. (A && B) is false. || Called logical operators OR (or). If either operator is non-zero, then the condition is true. (A || B) is true. ! Called the NOT operator. Use to reverse the logic state of that operand. If the operand condition is true, the negative will be false. ! (A && B) is true.

Bit comparison operator

The bit comparison operator works on bit units, calculates bitwise comparison expressions. The following table is about &, |, and ^ as follows:

pqp & qp | qp ^ q0 0 0 0 0 0 1 0 1 1 1 1 1 1 0 1 0 0 1 1
Suppose if A = 60; and B = 13; then in binary format they will be as follows:

A = 0011 1100

B = 0000 1101

A&B = 0000 1100

A | B = 0011 1101

A ^ B = 0011 0001

~ A = 1100 0011

Bit comparison operators supported by C language are listed in the table below. The price we use has variable A having the value 60 and variable B having the value 13, we have:

Operator DescriptionExample & binary AND (and) operator copy a bit to the result if it exists in both operands. (A & B) will result in 12, ie 0000 1100 | The binary OR operator (or) copies a bit to the result if it exists in one or two operands. (A | B) will result in 61, ie 0011 1101 ^ Binary copy XOR operator, which only exists in an operand and not both. (A ^ B) will result in 49, ie 0011 0001 ~ Bitwise operator (turn 1 bit into 0 bit and vice versa). (~ A) will result in -61, ie 1100 0011. Left shift operator. The left operand value is shifted left by the number of bits specified by the right operand. A 2 will give the result 240, ie 1111 0000 (left shift of two bits) >> Right shift operator. The left operand value is shifted right by the number of bits specified by the right operand. A >> 2 will result in 15, ie 0000 1111 (translate to two bits right)

Assignment operator

These are the assignment operators supported by the C language:

OperatorDescription = Simple assignment operator. Assign the right operand value to the left operand. C = A + B will assign the value of A + B to C + = Add the operand value to the left operand and assign that value to the left operand. C + = A is equivalent to C = C + A - = Subtract the right operand value from the left operand and assign this value to the left operand. C - = A is equivalent to C = C - A * = Multiply the right operand value by

the left operand and assign this value to the left operand. C * = A is equivalent to C = C * A / = Divide the left operand for the right operand and assign this value to the left operand. C / = A is equivalent to C = C / A% = Take the remainder of the left operand division for the right operand and assign the left operand. C% = A is equivalent to C = C% A = Left shift of left-handed math to position number is the right operand value. C = 2 is equivalent to C = C 2 >> = Right shift of left operand to position number is the right operand value. C >> = 2 is equivalent to C = C >> 2 & = AND bit C & = 2 is equivalent to C = C & 2 ^ = OR operation excludes bit C ^ = 2 equivalent to C = C ^ 2 | = OR bit. C | = 2 is equivalent to C = C | 2

Of sizeof & ternary mixed operators

Is there some important mixed operator is **sizeof** and **?** : supported by C language

DescriptionOf operatorSizeof () Returns the size of a sizeof (a) variable, with a being an integer, which returns the result of 4. & Returns the address of a variable. & a; will give the actual address of the variable a. * Point to a variable. * a; will point to variable a. ? : Conditional expression If the condition is true? then the value X: If not, the value Y

Operator priority in C

Operator priority in C determines how the expression is calculated. For example, the multiplication operator takes precedence over the addition operator, and it is done first.

For example, x = 7 + 3 * 2; Here, x is assigned a value of 13, not 20 because the * operator has a higher priority than the + operator, so first it performs multiplication 3 * 2 and then adds 7.

The following table lists the priority order of operators. Operators with the highest priority appear at the top of the table, and the operators with the lowest priority are at the bottom of the table. In an expression, the highest priority operators are first calculated.

Type EntriesPriority Order Postfix () [] ->. ++ - - Left to right Unary + -! ~ ++ - - (type) * & sizeof Right to left Multiplication * /% Left to right Addition + - Left to right Move >> Left to right Relationship =>> = Left to right Scales equals ==! = Left to right AND bit & Left to right XOR bit ^ Left to right OR bit | Left to right AND logic && Left to right Logical OR || Left to right Condition: Right to left Assign = + = - = * = / =% = >> = = & = ^ = | = Right to left Comma, Left to right
According to Tutorialspoint

Previous lesson: Storage class in C programming

Next lesson: Flow control in C programming