# COMPUTATIONAL THINKING FOR STRUCTURED DESIGN

Parul® University

# Operators in C

Parul® University

## Operators in C

Operators are the foundation of programming language.

C language provide different kind of operators.

Different types of operators in C are :

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment and decrement operators
6. Conditional operators
7. Bitwise operators

Parul® University

## Arithmetic Operators

Arithmetic operators are binary operators which are used to perform arithmetic operation.

These are :

| | |
|---|---|
| + | add |
| - | subtract |
| * | multiply |
| / | divide( divisor must be non zero ) |
| % | modulo(gives remainder after div) |

The parenthesis() are used to clarify complex operations. The operators + and - can be used as unary plus and unary minus arithmetic operators also. The unary – negates the sign of it's operand .

Note : C language has no operator for exponentiation.

The function pow(x,y) which exists in header file math.h of standard library and returns $X^y$

Following are some examples of arithmetic operators :

x+y, x-y, x*y, x/y, x%y, -x*y

Here x and y are operands that can take any value but % operator cannot be used on floating point data type.

## Arithmetic Expressions

An expression consisting of numerical values(either any number, variable or even some function call) joined together by arithmetic operators is known as an arithmetic expression. For example , consider the following expression :

(x-y)*(x+y)/5

Here x,y and 5 are operands and the symbols -,*,+,/ are operators.

The precedence of operators for the expression evaluation has been given by using parenthesis which will over rule the operators precedence. If x=25 and y=15,then the value of this expression will  be 80.

## Arithmetic Expressions

Consider the following expression :

3*((a%4)*(5+(b-2)/(c+3)))

Where a,b,c are integer variables and if a,b ,c have values 9 ,14 ,16 respectively then above expression would be evaluated as

3 * ((9%4) * (5 + (14 – 2) / (6 +3 )))

= 3 * ( 1 * (5 + (12 / 9) ) )

= 3 * ( 1 * (5 + 1))

= 3 * ( 1 * 6 )

= 3 * 6

= 18

## Arithmetic Operators Precedence

In C, the arithmetic operators have the priority as shown below:

First priority* / %

Second priority + - Third

priority =

The sequence of operations in evaluating an arithmetic expression is also known as hierarchy of operations. This is necessary to avoid any doubt while evaluating an expression. The following precedence rules are followed in expression evaluation :

## Arithmetic Operators Precedence

(i) All the subexpressions within the parentheses are evaluated first. Nested parenthesized subexpressions are evaluated inside-out, with the innermost expression being first to be evaluated.

(ii) Operators in the same sub expression are evaluated as given : *, /   ,% perform first +,  - performed next. Any function referenced (i.e.,invoked)in the expression gets the highest precedence over all the arithmetic operators.

(iii)Operators in the same expression with the same priority are evaluated from left to right.

## Arithmetic Operators Precedence

For example : consider the following expression for checking the operators precedence.

15 * 7 / ( 2 – 3 * 5 / 7 + 4 ) – 7 * 9 % 4

= 15 * 7 / (2 – 15 / 7 + 4 ) – 7 * 9 % 4

= 15 * 7 / (2 – 2 + 4) – 7 * 9 % 4

= 15 * 7 / 4 – 7 * 9 % 4

= 105 / 4 - 63 % 4

= 26 – 3

= 23

## Increment and Decrement Operator

C language has two useful operators called increment(++) and decrement (--) that operate on integer data only.

The increment (++) operator increments the operand by 1, while the decrement operator (--) decrements the operand by 1. There are two type of increment (++) operator : pre and post ,Similarly There are two type of decrement (--) operator : pre and post, for example :

```
        int a , b;
        a = 10;
    b = a++ ;
    printf(" %d %d ", a, b);
```

OUTPUT

```
        11 10 . First      a is assigned to b and then a is
```

incremented by 1 i.e.,post-increment takes place

Parul® University

## Increment and Decrement Operator

If we have :

```
int a, b ;
a = 20;
b = ++a;
printf("%d %d", a, b);
```

OUTPUT : 21 21. first a is incremented by 1 and then assignment take place i.e., pre-increment of a.

now, consider the example for (--) operator :

```
int a,b;
a=10;
b= a--;
printf("%d %d", a , b)
```

OUTPUT : 9 10. first *a* is assigned to *b* then *a* is decremented by 1. i.e.,post decrement takes place

## Decrement Operator

If we have :

```
        int i, j ;
                I = 20;
                j = --i;
                printf("%d %d", i, j);
```

OUTPUT : 19 19. first i is decremented by 1 and then assignment take place i.e., pre-decrement of i.

Note : on some compilers a space is required on both sides of ++i or i++ , i-- or --i

Parul® University

## Relational Operators

These are used to compare two variables or constants and return true or false . C has the following relational operators :

OPERATO R        MEANING

Equals

==        Not Equals

!=        Less than

<>        Greater than

<        Less than or

=

equals        Greater than or

>

=

equals

## Logical Operators

In C, we can have simple conditions (single) or compound conditions(two or more). The logical operators are used to combine conditions. The notations for these operators is given below :

| Operator | Notation in C |
|---|---|
| NOT | ! |
| AND | && |
| OR | \|\| |

The notation for the operator OR is given by two broken lines. These follow the same precedence as in other language. NOT(!) is evaluated before AND(&&) which is evaluatedbefore OR(||). Parenthesis( ) cab be used to change this order.

# Logical Operators

&& : returns true when all the condition under the consideration are true and returns false when anyone or more than one condition is false.

|| : returns true when one or more than one condition under the consideration are true and returns false when all the conditions are false

! : NOT operator is used to complement the condition under the consideration
Returns true when condition is false and returns false when condition is true

## Precedence of Relational Operators and Logical Operators

Each operator in C has a precedence of its own. It helps in evaluation of an expression. Higher the precedence of the operator, earlier it operates. The operators having same precedence are evaluated either from left to right or from right to left, depending on the level, known as the associativity of the operator.

! , < , <= , > , >=, ==, !=, ==, !=, &&, ||

# The Conditional Operator

This operator ? And : together forms a ternary operator called as the conditional operator.

Syntax : (test-expression) ? T-expr : F-expr ;

Let us see example program :

# The conditional operator

```
#include<stdio.h>
main()
{
int n;
clrscr();
printf)("Enter value of
n"); scanf("%d",&n);
(n%2==0)? Printf("n is even"):printf("n is not
even");
getch();
}
```

## Bitwise Operators

These are used to perform bitwise operations such as testing the bits, shifting the bits to left to right, one’s complement of bits etc. these operations can be applied only on int and char data types but not on float and double data types. Various bitwise operators in C language are :

| | |
|---|---|
| ~ | Bitwise (1’s) complement) |
| | shift left |
| < | |
| < | shift right |
| > | bitwise |
| > | |
| | AND |
| & | |
| | bitwise XOR(Exclusive |
| ^ | |
| | OR) bitwise OR |
| \| | |

## Special Operator

C provides the following special operators :

- Comma Operator
- sizeof operator
- Address operator
- Dereferencing operator
- Dot operator
- Member selection operator
- Pointer

## The Comma Operator

The comma operator (,) has the lowest precedence.

The comma operator is mainly used in for statement. For example :

```
int i , j;
for(i=1 , j=400 ; i<=10 ; ++I , j/=2)
printf("%d\n", i+j ) ;
```

The initial value of i is 1 and that of j is 400 and every time the value of i is incremented by 1 and that of j is divided by 2 after execution of the body of the for loop .

The distinct expression on either side of the comma operator are evaluated from left to right.

The associativity of comma operator is from left to right .

## The sizeof Operator

It is a unary operator which provides the size , in bytes, of the given operand. The syntax of sizeof operator is :

```
sizeof(operand)
```

Here the operand is a built in or user defined data type or variable.

The sizeof operator always precedes its operand.

For example, sizeof (float) returns the value 4 .

The sizeof operator mainly used in dynamic memory allocation for calculating the number of bytes used by some user defined data type.

## Precedence of operators among themselves and across all the sets of operators.

The TURBO C operators are divided into the following 16 categories : these are ordered from the highest precedence to the lowest precedence. The operation within each category have equal precedence.

Parul® University

## Precedence of operators among themselves and across all the sets of operators.

| Category | Operator | What it does ? |
|---|---|---|
| 1. Highest precedence | () | Function call |
| | [] | Array subscript |
| | -> | C indirect component selector |
| 2.Unary | ! | NOT |
| | ~ | Bitwise(1's) component |
| | + | Unary plus |
| | - | Unary minus |
| 3.Member acces | .* | Dereference |
| | ->* | Dereference |
| 4.Multiplication | * | Multiply |
| | / | Divide |
| | % | Remainder (Modulus) |
| | | |

Parul® University

| Category | Operator | What it does ? |
|---|---|---|
| 5.Additive | + | Binary plus |
| | - | Binary minus |
| 6.Shift | << | Shift left |
| | >> | Shift right |
| 7.Relational | < | Less than |
| | <= | Less than or equal to |
| | > | Greater than |
| | >= | Greater than equal to |
| 8.Equality | == | Equal to |
| | != | Not equal to |
| 9.Bitwise AND | & | Bitwise AND |
| 10.Bitwise XOR | ^ | Bitwise XOR |
| 11.Bitwise OR | \| | Bitwise OR |

## Precedence of operators among themselves and across all the sets of operators.

## Precedence of operators among themselves and across all the sets of operators.

## The Associativity of Operators

In C , the operators having the equal precedence are evaluated either from left to right or from right to left, depending on the level. It is known as associativity property of the operator.

Parul® University

# Associativity of the Operator

| Category | Operator | Associativity |
|---|---|---|
| 1.Highest precedence | ( ) | Left to Right |
| | [ ] | |
| | -> | |
| | :: | |
| | . | |
| 2.Unary | ! | Right to left |
| | ~ | |
| | + | |
| | - | |
| | ++ | |
| | -- | |
| | & | |
| | * | |
| | sizeof | |

Parul® University

## Associativity of the Operator

| Category | Operator | Associativity |
|---|---|---|
| 3.Member access | .* | Left to Right |
| | ->* | |
| 4.Multiplication | * | Left to right |
| | / | |
| | % | |
| 5.Additive | + | Left to Right |
| | - | |
| 6.Shift | << | Left to Right |
| | >> | |
| 7.Relational | < | Left to Right |
| | <= | |
| | > | |
| | >= | |

Parul® University

## Associativity of the Operator

| Category | Operator | Associativity |
|---|---|---|
| 8.Equality | == | Left to Right |
| | != | |
| 9.Bitwise AND | & | Left to right |
| 10.Bitwise XOR | ^ | Left to Right |
| 11.Bitwise OR | \| | Left to Right |
| 12.Logical AND | && | Left to Right |
| 13.Logical OR | \|\| | Left to Right |
| 14.Conditional | ?: | Right to Left |
| 15.Assignment | = | Right to Left |
| | *= | |
| | /= | |
| | %= | |
| | += | |
| | -= | |

| Category | Operator | Associativity |
|---|---|---|
| | &= | Right to Left |
| | ^= | |
| | \|= | |
| | <<= | |
| | >>= | |
| 16.Comma | . | Left to Right |

## Associativity of the Operator

Thank
You