

# Operators in C



**Instructor:** Maninder Kaur

**Email:** [maninder.kaur2@sheridancollege.ca](mailto:maninder.kaur2@sheridancollege.ca)

**Course:** PROG20799

# What are Operators?

- An operator is a symbol that tells the compiler to perform specific mathematical or logical operation.
- C language is rich in built-in operators and provides following type of operators:
  - Arithmetic Operators
  - Relational Operators
  - Logical Operators
  - Assignment Operators
  - Misc. Operators

# Arithmetic Operators

- Following table shows all the arithmetic operators supported by C language.
- Assume variable **A holds 10** and variable **B holds 20**:

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by de-numerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator increases integer value by one	A++ will give 11
--	Decrement operator decreases integer value by one	A-- will give 9

# Arithmetic Operators - Example

```
#include <stdio.h>

void main()
{
    int a = 21, b = 10, c;

    c = a + b;
    printf("Line 1 - Value of c is %d\n", c );

    c = a - b;
    printf("Line 2 - Value of c is %d\n", c );

    c = a * b;
    printf("Line 3 - Value of c is %d\n", c );

    c = a / b;
    printf("Line 4 - Value of c is %d\n", c );

    c = a % b;
    printf("Line 5 - Value of c is %d\n", c );

    c = a++;
    printf("Line 6 - Value of c is %d\n", c );

    c = a--;
    printf("Line 7 - Value of c is %d\n", c );
}
```

## Output:

Line 1 - Value of c is 31  
Line 2 - Value of c is 11  
Line 3 - Value of c is 210  
Line 4 - Value of c is 2  
Line 5 - Value of c is 1  
Line 6 - Value of c is 21  
Line 7 - Value of c is 22

# Relational Operators

- Following table shows all the relational operators supported by C language.
- Assume variable **A holds 10** and variable **B holds 20**:

Operator	Description	Example
==	Checks if the value of two operands is equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

# Relational Operators - Example

```
#include <stdio.h>

void main()
{
    int a = 21, b = 10;

    if( a == b )
        printf("Line 1 - a is equal to b. \n" );
    else
        printf("Line 1 - a is not equal to b. \n" );

    if ( a < b )
        printf("Line 2 - a is less than b. \n" );
    else
        printf("Line 2 - a is not less than b. \n" );

    if ( a > b )
        printf("Line 3 - a is greater than b. \n" );
    else
        printf("Line 3 - a is not greater than b.
\n" );
}
```

## Output:

Line 1 - a is not equal to b.  
Line 2 - a is not less than b.  
Line 3 - a is greater than b.

# Logical Operators

- Following table shows all the logical operators supported by C language.
- Assume variable **A holds 1** and variable **B holds 0**:

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	(A && B) is false.
	Called Logical OR Operator. If any of the two operands is non zero then condition becomes true.	(A    B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true.

# Logical Operators - Example

```
#include <stdio.h>

void main()
{
    int a = 5, b = 20;

    if ( a < 10 && b > 20 )
        printf("Line 1 - Condition is true\n" );

    if ( a == 10 || b == 5 )
        printf("Line 2 - Condition is true\n" );

    if (!(a == b))
        printf("Line 3 - Condition is true\n" );
}
```

## Output:

Line 3 - Condition is true



# Assignment Operators

- There are following assignment operators supported by C language:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	$C = A + B$ will assign value of $A + B$ into $C$
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	$C += A$ is equivalent to $C = C + A$
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	$C -= A$ is equivalent to $C = C - A$
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	$C *= A$ is equivalent to $C = C * A$
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	$C /= A$ is equivalent to $C = C / A$
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	$C \% = A$ is equivalent to $C = C \% A$

# Assignment Operators - Example

```
#include <stdio.h>

void main()
{
    int a = 21, c;

    c = a;
    printf("Line 1: = Operator Example, Value of c = %d\n", c );

    c += a;
    printf("Line 2: += Operator Example, Value of c = %d\n", c );

    c -= a;
    printf("Line 3: -= Operator Example, Value of c = %d\n", c );

    c *= a;
    printf("Line 4: *= Operator Example, Value of c = %d\n", c );

    c /= a;
    printf("Line 5: /= Operator Example, Value of c = %d\n", c );

    c = 200;
    c %= a;
    printf("Line 6: %= Operator Example, Value of c = %d\n", c );
}
```

## Output:

```
Line 1: = Operator Example, Value of c = 21
Line 2: += Operator Example, Value of c = 42
Line 3: -= Operator Example, Value of c = 21
Line 4: *= Operator Example, Value of c = 441
Line 5: /= Operator Example, Value of c = 21
Line 6: %= Operator Example, Value of c = 11
```

# Misc. Operators

- There are few other important operators including **sizeof** and **?:** supported by C Language.

Operator	Description	Example
sizeof()	Returns the size of an variable.	sizeof(a), where a is integer, will return 4.
&	Returns the address of an variable.	&a; will give actual address of the variable.
*	Pointer to a variable.	*a; will pointer to a variable.
?:	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

# Integer Types - Example

```
#include <stdio.h>


Void main()
{
    printf("Storage size for int: %d", sizeof(int));
}
```

- Check the output for yourself.

# Operators Precedence in C

- Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated.
- Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator:
- For example  $x = 7 + 3 * 2$ ; Here  $x$  is assigned 13, not 20 because operator  $*$  has higher precedence than  $+$  so it first get multiplied with  $3*2$  and then adds into 7.
- Operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

# Operators Precedence in C

Operators	Priority	Associativity
- , ++ , -- , !	Highest	Right to left
* , / , %		Left to right
+ , -		Left to right
< , <= , > , >=		Left to right
== , !=		Left to right
&&		Left to right
	Lowest	Left to right



**Any questions please?**