Arithmetic Operators in C Programming

Introduction :

Arithmetic operators are used to perform mathematical operations on numeric values (constants or variables) in C programming. These operators help us to do basic calculations like addition, subtraction, multiplication, division and modulus (remainder).

List of Arithmetic Operators :

Operator	Name	Example	Description
+	Addition	a + b	Adds two values
−	Subtraction	a − b	Subtracts second value from first value
*	Multiplication	a * b	Multiplies two values
/	Division	a / b	Divides first value by second value
%	Modulus (Remainder)	a % b	Returns remainder of division

Examples :

Let's take two variables a = 10 and b = 3. Here's how each arithmetic operator works:

a + b → 10 + 3 = 13 (Addition)
a − b → 10 − 3 = 7 (Subtraction)
a * b → 10 * 3 = 30 (Multiplication)
a / b → 10 / 3 = 3 (Division - integer result)
a % b → 10 % 3 = 1 (Modulus - remainder)

Important Points :

1. In C, when both operands are integers, the result of division is also an integer (fractional part is ignored).
2. The modulus operator (%) can only be used with integers.
3. If we want a floating-point result in division, at least one operand should be a float or double.

Unary Operators in C Programming

Introduction :

Unary operators are operators that operate on a single operand (one value). They are used to perform operations such as incrementing, decrementing, changing the sign of a number, or getting the logical NOT of a value.

List of Unary Operators:

Operator	Name	Example	Description
++	Increment	++a or a++	Increases the value by 1.
--	Decrement	--a or a--	Decreases the value by 1.
+	Unary plus	+a	Returns the value as it is (positive).
-	Unary minus	-a	Changes the sign of the value.
!	Logical NOT	!a	Reverses the logical value (0 becomes 1, non-zero becomes 0).
~	Bitwise complement	~a	Inverts all the bits of the operand.

Examples :

```
++a  →  a = 5,  ++a = 6                (Pre-increment)
a++  →  a = 5,  a++ = 5, a = 6         (Post-increment)
--a  →  a = 5,  --a = 4                (Pre-decrement)
a--  →  a = 5,  a-- = 5, a = 4         (Post-decrement)
+a   →  a = -5, +a = -5                (Unary plus)
-a   →  a = 5,  -a = -5                (Unary minus)
!a   →  a = 0,  !a = 1                 (Logical NOT)
~a   →  a = 5,  ~a = -6                (Bitwise complement)
```

Important Points:

1. Unary operators work on a single operand.
2. The increment (++) and decrement (--) operators can be used in both prefix and postfix forms.
3. The unary minus (-) changes the sign of a number.
4. The logical NOT (!) gives 1 for 0 and 0 for any non-zero value.
5. The bitwise complement (~) is used for bit-level manipulation.

Relational Operators in C Programming

Introduction :

Relational operators are used to compare two values. They return a result in the form of a Boolean value (1 for true and 0 for false). Relational operators are commonly used in decision making and loop control statements.

List of Relational Operators :

Operator	Name	Example	Description
==	Equal to	a == b	Checks if two values are equal.
!=	Not equal to	a != b	Checks if two values are not equal.
>	Greater than	a > b	Checks if left value is greater.
<	Less than	a < b	Checks if left value is smaller.
>=	Greater than or equal to	a >= b	Checks if left value is greater or equal to right value.
<=	Less than or equal to	a <= b	Checks if left value is smaller or equal to right value.

Examples :

```
int a = 10, b = 20;
if (a > b)          // false
    printf("a is greater");
else
    printf("b is greater");   // executed
```

Important Points :

1. Relational operators return 1 (true) or 0 (false).
2. They are used in if, else, while, for and other control statements.
3. The result is always a Boolean value (integer).

Logical Operators in C Programming

Introduction :

Logical operators are used to combine two or more conditions. They return a Boolean value (1 for true and 0 for false). Logical operators are mainly used in decision making and in complex conditions.

List of Logical Operators :

Operator	Name	Example	Description
&&	Logical AND	a && b	Returns true if both conditions are true.
||	Logical OR	a || b	Returns true if any one condition is true.
!	Logical NOT	!a	Reverses the condition (true ↔ false).

Examples :

```
int a = 5, b = 10;
if (a > 0 && b > 0)  // true
    printf("Both are positive");

if (a > 10 || b > 20)  // false
    printf("At least one is greater");

if (!a)   // false (a is 5)
    printf("a is zero");
```

Important Points :

1. Logical AND (&&) returns true only if both conditions are true.
2. Logical OR (||) returns true if any one condition is true.
3. Logical NOT (!) changes true to false and false to true.
4. These operators are widely used in if, else if, while and for loops.

Assignment Operators in C Programming

Introduction :

Assignment operators are used to assign a value to a variable. They store the result of an expression in a variable. The most commonly used assignment operator is = (simple assignment), but C also provides several compound assignment operators.

List of Assignment Operators :

Operator	Name	Example	Description
=	Simple assignment	a = b	Assigns the value of b to a.
+=	Add and assign	a += b	Adds b to a and assigns the result to a.
-=	Subtract and assign	a -= b	Subtracts b from a and assigns the result to a.
*=	Multiply and assign	a *= b	Multiplies a by b and assigns the result to a.
/=	Divide and assign	a /= b	Divides a by b and assigns the result to a.
%=	Modulus and assign	a %= b	Finds remainder of a % b and assigns the result to a.

Examples :

```
int a = 10, b = 5;
a += b;     // a becomes 15   (10 + 5)
a -= b;     // a becomes 5    (10 - 5)
a *= b;     // a becomes 50   (10 * 5)
a /= b;     // a becomes 2    (10 / 5)
a %= b;     // a becomes 0    (10 % 5)
```

Important Points :

1. The assignment operator (=) stores the right-hand value in the left-hand variable.
2. Compound assignment operators are shorthand way to write operations and assignment together.

Conditional Operators in C Programming

Introduction :

Conditional operators are used to check a condition and return a value based on the result (true or false). They are mainly used in decision making and expressions. The conditional operator ? : is called the ternary operator.

List of Conditional Operators :

Operator	Name	Example	Description
==	Equal to	a == b	Checks if two values are equal.
!=	Not equal to	a != b	Checks if two values are not equal.
>	Greater than	a > b	Checks if left value is greater.
<	Less than	a < b	Checks if left value is smaller.
>=	Greater than or equal to	a >= b	Checks if left value is ≥ right value.
<=	Less than or equal to	a <= b	Checks if left value is ≤ right value.
? :	Ternary operator	(a > b) ? a : b	Returns a if condition is true, otherwise returns b.

Examples :

```
int a = 10, b = 20;
if (a > b)   // checks if a is greater than b
    printf("a greater");
else
    // printf("b is greater");
```

Using ternary operator:

```
int max = (a > b) ? a : b;
  // max will be 20
```

Important Points :

1. Relational operators (==, !=, >, <, >=, <=) return 1 (true) or 0 (false).
2. The ternary operator ? : is a shorthand form of if-else and is very useful in simple conditions.

Library Functions in C Programming

Introduction :

Library functions are pre-written functions that are stored in header files. We can use these functions in our program to perform common and useful tasks without writing them from scratch. These functions save time and make the program shorter and easier to maintain.

Types of Library Functions :

In C, library functions are grouped into different header files. Some commonly used header files are:

Header File	Description	Example Functions
<stdio.h>	Input / Output functions	printf(), scanf(), gets(), puts()
<conio.h>	Console input / output (non-standard)	getch(), getche(), clrscr()
<stdlib.h>	General utility functions	malloc(), free(), exit(), rand()
<string.h>	String handling functions	strlen(), strcpy(), strcmp()
<math.h>	Mathematical functions	sqrt(), pow(), sin(), cos()
<ctype.h>	Character handling functions	isalpha(), isdigit(), tolower()
<time.h>	Date and time functions	time(), clock(), localtime()
<stdbool.h>	Boolean type (C99)	true, false

Examples of Library Functions :

1. printf() – Prints output on screen.
   Example:
   ```
   printf("Hello, World!");
   ```
2. scanf() – Reads input from keyboard.
   Example:
   ```
   scanf("%d", &num);
   ```
3. strlen() – Returns length of a string.
   Example:
   ```
   int len = strlen("Hello");
   ```
4. sqrt() – Returns square root of a number.
   Example:
   ```
   double r = sqrt(25);  // r = 5.0
   ```
5. strcpy() – Copies one string to another.
   Example:
   ```
   strcpy(dest, src);
   ```
6. strcmp() – Compares two strings.
   Example:
   ```
   int result = strcmp(s1, s2);
   ```
7. rand() – Generates a random number.
   Example:
   ```
   int r = rand();
   ```
8. time() – Returns current time.
   Example:
   ```
   time_t t = time(NULL);
   ```

Important Points :

1. To use library functions, we must include the corresponding header file at the beginning of the program using #include.
2. Example: #include <stdio.h>
3. Library functions help in reducing code length and improve efficiency.
4. Some functions are standard (portable), while others are non-standard (platform specific).