### ****What is an Algorithm?****

An **algorithm** is a **step-by-step procedure or set of rules** designed to perform a specific task or solve a particular problem.

### ****Key Characteristics of a Good Algorithm:****

1. **Input**: Takes one or more inputs.
2. **Output**: Produces at least one output.
3. **Definiteness**: Each step is clear and unambiguous.
4. **Finiteness**: It ends after a finite number of steps.
5. **Effectiveness**: Each step can be performed with basic resources.

**Algorithm**

**Algorithm: Fahrenheit to Celsius Conversion**

**Step 1: Start**

**Step 2: Input temperature in Fahrenheit (let's say F)**

**Step 3: Apply the conversion formula:**

    C = (F – 32) × 5 / 9

**Step 4: Output the temperature in Celsius (C)**

**Step 5: End**

## ****Algorithm: Simple and Compound Interest Calculation****

### ****Step 1:**** Start

### ****Step 2:**** Input the Principal amount → P

### ****Step 3:**** Input the Rate of interest (per annum) → R

### ****Step 4:**** Input the Time period (in years) → T

### ****Step 5:**** Calculate the Simple Interest using the formula:

    SI = (P × R × T) / 100

### ****Step 7:**** Calculate the Compound Interest using the formula:

### **Amount = P × (1 + R / (100 × N)) ^ (N × T)**

### **CI = Amount − P**

### ****Step 6:**** Display SI (Simple Interest) and CI (Compound Interest)

### ****Step 7:**** End

## ****Algorithm: Perform All Arithmetic Operations****

### ****Step 1:**** Start

### ****Step 2:**** Input two numbers → A and B

### ****Step 3:**** Perform the following operations:

* Sum = A + B
* Difference = A - B
* Product = A × B
* **If B ≠ 0 then**
  + Quotient = A ÷ B
* **Else**
  + Display "Division by zero is not allowed"
  + Remainder = A MOD B [A % B]

### ****Step 4:**** Display all results:

* Sum
* Difference
* Product
* Quotient (if division is valid)
* Remainder

### ****Step 5:**** End

### ****Algorithm: Larger between two Numbers****

**Step 1:** Start  
**Step 2:** Input two numbers: A, B  
**Step 3:**  
    if A > B then  
        → A is the larger

Large = A  
    else   
        → B is the larger

Large = B

**Step 4: Print Large**  
**Step 5:** Stop

### ****Algorithm: Largest of Three Numbers****

**Way 1:**

**Step 1:** Start  
**Step 2:** Input three numbers: A, B, C  
**Step 3:**  
    if A > B

If A > C  
         → A is the largest

Else

à C is the largest

else if B > C  
        → B is the largest  
    else  
        → C is the largest  
**Step 4:** Display the largest number  
**Step 5:** Stop

**Way 2:**

**Step 1:** Start  
**Step 2:** Input three numbers: A, B, C  
**Step 3:**  
    if A > B and A > C  
        → A is the largest  
    else if B > C  
        → B is the largest  
    else  
        → C is the largest  
**Step 4:** Display the largest number  
**Step 5:** Stop

## ****Algorithm for Checking the Type of Triangle****

### ****Step 1:**** Start

### ****Step 2:**** Input the three sides of the triangle: A, B, C

### ****Step 3:**** Check for triangle validity using the triangle inequality rule:

* **If** (A + B > C) AND (A + C > B) AND (B + C > A) then proceed
* **Else**
  + Display "Not a valid triangle"
  + **Go to Step 7**

### ****Step 4:**** Check if all sides are equal:

* **If** A == B AND B == C
  + Display "Equilateral Triangle"

### ****Step 5:**** Check if any two sides are equal:

* **Else If** A == B OR B == C OR A == C
  + Display "Isosceles Triangle"

### ****Step 6:**** If none of the sides are equal:

* **Else**
  + Display "Scalene Triangle"

### ****Step 7:**** End

## ****Algorithm: Factorial of a Number****

### ****Step 1:**** Start

### ****Step 2:**** Input a number → N

### ****Step 3:**** Initialize a variable → Fact = 1

### ****Step 4:**** Use a loop from i = 1 to N

        Fact = Fact × i

### ****Step 6:**** Display Fact

### ****Step 7:**** End

## ****Algorithm: Prime Number Check****

### ****Step 1:**** Start

### ****Step 2:**** Input a number → N

### ****Step 3:**** If N ≤ 1

    Display "Not a Prime Number"  
    Go to Step 7

### ****Step 4:**** Initialize a flag → IsPrime = true

### ****Step 5:**** Loop from i = 2 to N/2

    If N mod i == 0  
        Set IsPrime = false  
        Break the loop

### ****Step 6:**** If IsPrime == true

    Display "Prime Number"  
Else  
    Display "Not a Prime Number"

### ****Step 7:**** End

## ****Algorithm: Display Prime Numbers in a Given Range****

### ****Step 1:**** Start

### ****Step 2:**** Input the starting number → Start

### ****Step 3:**** Input the ending number → End

### ****Step 4:**** Repeat for each number N from Start to End

    a. If N ≤ 1, skip to next number  
    b. Set IsPrime = true  
    c. For i = 2 to N/2  
        If N mod i == 0  
            Set IsPrime = false  
            Break  
    d. If IsPrime == true, then  
        Display N

### ****Step 5:**** End

**Pseudocode**

**What is Pseudocode?**

**Pseudocode** is a **simple, informal way** of describing an algorithm using **plain English-like statements**, without worrying about the exact syntax of a programming language.

**Key Features of Pseudocode:**

* **Easy to read and understand**
* Looks like a mix of English and programming logic
* Used to **plan out a program or algorithm** before writing actual code
* No strict rules, but follows **logical structure** (like IF, WHILE, FOR)

## ****Pseudocode: Fahrenheit to Celsius Conversion****

* + 1. Start
    2. Input Fahrenheit (F)
    3. Celsius ← (F - 32) × 5 / 9
    4. Display Celsius
    5. End

**Pseudocode: Largest of Three Numbers**

* 1. START
  2. INPUT A, B, C
  3. IF A > B AND A > C THEN

PRINT "A is the largest:", A

ELSE IF B > C THEN

PRINT "B is the largest:", B

ELSE

PRINT "C is the largest:", C

ENDIF

* 1. END

### What is a ****Flowchart****?

A **flowchart** is a **visual diagram** that represents the **sequence of steps** in a **process, algorithm, or program**. It uses **symbols** like arrows, rectangles, and diamonds to show actions, decisions, and flow of control.

### Why Use Flowcharts?

* To **visualize logic clearly**
* To **plan a program** before writing code
* To **communicate algorithms** effectively
* Easy to **understand, debug, and explain**

### Common Flowchart Symbols:

| **Symbol** | **Name** | **Meaning** |
| --- | --- | --- |
| 🔷 Oval | **Terminator** | Start / End |
| 🔲 Rectangle | **Process** | An action or instruction |
| 🔷 Diamond | **Decision** | A yes/no or true/false choice |
| ➡️ Arrow | **Flowline** | Shows direction of flow |
| 🔳 Parallelogram | **Input/Output** | Read input or display output |

**Flowchart for Fahrenheit to Celsius conversion**

## Flowchart Example: Temperature to Celsius Conversion | Flowchart Template

## ****Flowchart for simple interest****

## 

## ****Flowchart for checking the type of triangle****

## Draw flowchart to find the largest amoung3 numbers​ - Brainly.in

## ****Flowchart for calculating the factorial of a given number****

## Factorial of a Number using Loop in C++ - Dot Net Tutorials

## ****What is an Identifier?****

An **identifier** is the **name** used to identify **variables**, **functions**, **classes**, **arrays**, etc., in a program.

In short:

**All variable names are identifiers, but not all identifiers are variables.**

**Example:**

**int age;**

**float salary;**

**void calculate(); // Function name is also an identifier**

**class Employee // Class name is also an identifier**

**{**

**}**

## ****Rules for Naming Identifiers:****

1. Can contain **letters (A-Z, a-z)**, **digits (0-9)**, and **underscores (\_)**.
2. **Must begin with a letter or underscore** (not a digit).
3. **Cannot use C++ keywords** (e.g., int, return, class) as identifiers.
4. Case-sensitive: Total and total are different identifiers.
5. Should be meaningful (e.g., use marks instead of m).

**Example**

int num = 10; // 'num' is a variable and an identifier

float price = 99.99; // 'price' is a variable and identifier

## ****What is a Variable?****

A **variable** is a **named storage location** in memory that holds a value, which can change during program execution.

### Example in C++:

**int age = 25;**

Here:

* int is the **data type**
* age is the **variable name**
* 25 is the **value stored**

### ****Data Types in C++****

In C++, **data types** specify the **type of data** a variable can hold. They help the compiler allocate memory and understand how to manipulate the data.

## ****1. Fundamental (Primitive) Data Types****

| **Data Type** | **Description** | **Example** |
| --- | --- | --- |
| int | Integer values | int x = 10; |
| float | Floating-point numbers (decimal) | float f = 5.5; |
| double | Double-precision floating-point | double d = 3.14159; |
| char | Single character | char ch = 'A'; |
| bool | Boolean values (true/false) | bool flag = true; |
| void | No value / used for functions | void display(); |

## ****2. Derived Data Types****

| **Type** | **Description** |  |
| --- | --- | --- |
| Arrays | Collection of elements of same type |  |
| Pointers | Stores address of another variable |  |
| Functions | Blocks of code performing tasks |  |
| Classes | Prototype for the objects [Combination of variables of different data types and functions] |  |
| Structures | Collection of elements of different data type |  |

**Operators in C++**

In **C++**, **operators** are special symbols used to perform **operations** on variables and values. They are grouped into different categories based on their purpose.

### 🔹 1. ****Arithmetic Operators****

Used to perform basic mathematical operations.

| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| + | Addition | a + b |
| - | Subtraction | a - b |
| \* | Multiplication | a \* b |
| / | Division | a / b |
| % | Modulus (remainder) | a % b |

### 🔹 2. ****Relational (Comparison) Operators****

Used to compare values. Returns true or false.

| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| == | Equal to | a == b |
| != | Not equal to | a != b |
| > | Greater than | a > b |
| < | Less than | a < b |
| >= | Greater or equal | a >= b |
| <= | Less or equal | a <= b |

### 🔹 3. ****Logical Operators****

Used for combining logical expressions.

| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| && | Logical AND | (a > 0 && b < 5) |
| || | Logical OR | (age >= 60 || gender == ’F’ ) |
| ! | Logical NOT | !(a > 0) |

### 🔹 4. ****Assignment Operators****

Used to assign values to variables.

| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| = | Assign | a = 10 |
| += | Add and assign | a += 5 (same as **a = a + 5**) |
| -= | Subtract and assign | a -= 5 |
| \*= | Multiply and assign | a \*= 5 |
| /= | Divide and assign | a /= 5 |
| %= | Modulus and assign | a %= 5 |

a= 10

b=20

b = b \* 10 + b / 2 è b \*= 10 + b / 2

### 🔹 5. ****Increment and Decrement Operators****

Used to increase or decrease value by 1.

| **Operator** | **Meaning** | **Example** |
| --- | --- | --- |
| ++ | Increment | a++ or ++a |
| -- | Decrement | a-- or --a |

Note: a = a + 1 è a++ à ++a à a += 1

### 🔹 6. ****Bitwise Operators****

Work at the bit level.

| **Operator** | **Meaning** |
| --- | --- |
| & | Bitwise AND |
| ` | ` |
| ^ | Bitwise XOR |
| ~ | Bitwise NOT |
| << | Left shift |
| >> | Right shift |

### 🔹 7. ****Other Operators****

| **Operator** | **Meaning** |
| --- | --- |
| sizeof | Returns size of a data type/ variable |
| ?: | Ternary conditional operator |
| , | Comma |
| -> | Member access through pointer |
| . | Member access |
| & | Address-of operator |
| \* | Pointer dereference |

**Programs**

**1. C++ Program: Fahrenheit to Celsius Conversion**

// Program for Fahrenheit to Celsius Conversion

#include <iostream>

using namespace std;

int main() {

int Fahrenheit;

float celsius;

// Input temperature in Fahrenheit

cout << "Enter temperature in Fahrenheit: ";

cin >> fahrenheit;

// Convert to Celsius

celsius = (fahrenheit - 32) \* 5 / 9;

// Output result

cout << "Temperature in Celsius: " << celsius << "°C" << endl;

return 0;

}

**2. C++ Program: Simple and Compound Interest Calculation**

// C++ Program: Simple and Compound Interest Calculation

#include <iostream>

#include <cmath> // for pow() function

using namespace std;

int main() {

double principal, rate, time;

int n; // Number of times interest is compounded per year

// Input from user

cout << "Enter Principal amount: ";

cin >> principal;

cout << "Enter Annual Interest Rate (%): ";

cin >> rate;

cout << "Enter Time (in years): ";

cin >> time;

cout << "Enter Number of times interest is compounded per year: ";

cin >> n;

// Simple Interest calculation

double simpleInterest = (principal \* rate \* time) / 100;

// Compound Interest calculation

double amount = principal \* pow(1 + (rate / (100 \* n)), (n \* time));

double compoundInterest = amount - principal;

// Output results

cout << "\n--- Interest Calculation Results ---" << endl;

cout << "Simple Interest: " << simpleInterest << endl;

cout << "Compound Interest: " << compoundInterest << endl;

cout << "Total Amount after Compound Interest: " << amount << endl;

return 0;

}

**3. C++ Program: Perform All Arithmetic Operations**

// Perform All Arithmetic Operations

#include <iostream>

using namespace std;

int main() {

float num1, num2;

// Input two numbers

cout << "Enter first number: ";

cin >> num1;

cout << "Enter second number: ";

cin >> num2;

// Perform operations

cout << "\nResults:" << endl;

cout << "Addition: " << num1 + num2 << endl;

cout << "Subtraction: " << num1 - num2 << endl;

cout << "Multiplication: " << num1 \* num2 << endl;

// Check for division by zero

if (num2 != 0) {

cout << "Division: " << num1 / num2 << endl;

} else {

cout << "Division: Not allowed (division by zero)" << endl;

}

return 0;

}

**3. C++ Program: Triangle Type Checking**

**//** Triangle Type Checking Program

#include <iostream>

using namespace std;

int main() {

float side1, side2, side3;

// Input the three sides of the triangle

cout << "Enter the three sides of the triangle: ";

cin >> side1 >> side2 >> side3;

// Check for triangle validity

if (side1 + side2 > side3 && side1 + side3 > side2 && side2 + side3 > side1) {

// Check for type of triangle

if (side1 == side2 && side2 == side3) {

cout << "The triangle is Equilateral." << endl;

}

else if (side1 == side2 || side2 == side3 || side1 == side3) {

cout << "The triangle is Isosceles." << endl;

}

else {

cout << "The triangle is Scalene." << endl;

}

} else {

cout << "The given sides do not form a valid triangle." << endl;

}

return 0;

}

**4. C++ Program: Factorial of a Number**

// Program for Factorial of a Number

#include <iostream>

using namespace std;

int main() {

int n;

unsigned long long factorial = 1;

// Input a number

cout << "Enter a positive integer: ";

cin >> n;

// Check for valid input

if (n < 0) {

cout << "Factorial is not defined for negative numbers." << endl;

} else {

// Calculate factorial

for (int i = 1; i <= n; ++i) {

factorial \*= i;

}

// Output the result

cout << "Factorial of " << n << " is " << factorial << endl;

}

return 0;

}

**5. C++ Program: Prime Number Check**

// Program for checking the number is Prime or Not

#include <iostream>

using namespace std;

int main() {

int num, i;

bool isPrime = true;

// Input the number

cout << "Enter a positive integer: ";

cin >> num;

// Handle edge cases

if (num <= 1) {

isPrime = false;

}

else if ( num == 2 ) {

isPrime = true;

}

else

{

// Check divisibility from 2 tonum/2

for (i = 2; i <= num/2; ++i) {

if (num % i == 0) {

isPrime = false;

break;

}

}

}

// Output result

if (isPrime) {

cout << num << " is a prime number." << endl;

} else {

cout << num << " is not a prime number." << endl;

}

return 0;

}

**6. C++ Program: Display Prime Numbers between a ranges**

// Program to list out Prime Numbers between a range

#include <iostream>

using namespace std;

bool isPrime(int num) {

if (num <= 1) return false;

for (int i = 2; i \* i <= num; ++i) {

if (num % i == 0)

return false;

}

return true;

}

int main() {

int start, end;

// Input the range

cout << "Enter the start of the range: ";

cin >> start;

cout << "Enter the end of the range: ";

cin >> end;

cout << "Prime numbers between " << start << " and " << end << " are:\n";

// Loop through the range and print primes

for (int i = start; i <= end; ++i) {

if (isPrime(i)) {

cout << i << " ";

}

}

cout << endl;

return 0;

}

1. **C++ Program: Factorial Using Function**

// Program to calculate Factorial Using Function

#include <iostream>

using namespace std;

// Function to calculate factorial

unsigned long long factorial(int n) {

unsigned long long fact = 1;

for (int i = 1; i <= n; ++i) {

fact \*= i;

}

return fact;

}

int main() {

int number;

// Input from user

cout << "Enter a positive integer: ";

cin >> number;

// Check for valid input

if (number < 0) {

cout << "Factorial is not defined for negative numbers." << endl;

} else {

// Call the function and display result

cout << "Factorial of " << number << " is " << factorial(number) << endl;

}

return 0;

}

1. **C++ Program: nCr Using Function**

// Program to calculate nCr Using Function

#include <iostream>

using namespace std;

// Function to calculate factorial

unsigned long long factorial(int n) {

unsigned long long fact = 1;

for (int i = 1; i <= n; ++i) {

fact \*= i;

}

return fact;

}

// Function to calculate nCr

unsigned long long nCr(int n, int r) {

if (r > n)

return 0;

return factorial(n) / (factorial(r) \* factorial(n - r));

}

int main() {

int n, r;

// Input values

cout << "Enter value of n: ";

cin >> n;

cout << "Enter value of r: ";

cin >> r;

// Check for valid input

if (n < 0 || r < 0) {

cout << "n and r should be non-negative integers." << endl;

} else if (r > n) {

cout << "r cannot be greater than n." << endl;

} else {

// Display result

cout << "nCr (" << n << "C" << r << ") = " << nCr(n, r) << endl;

}

return 0;

}

1. **C++ Program: Recursive Factorial Function**

#include <iostream>

using namespace std;

// Recursive function to calculate factorial

unsigned long long factorial(int n) {

if (n == 0 || n == 1)

return 1; // Base case

else

return n \* factorial(n - 1); // Recursive call

}

int main() {

int number;

// Input from user

cout << "Enter a positive integer: ";

cin >> number;

// Check for valid input

if (number < 0) {

cout << "Factorial is not defined for negative numbers." << endl;

} else {

// Call and display the result

cout << "Factorial of " << number << " is " << factorial(number) << endl;

}

return 0;

}

1. **C++ Program: Sort Names Alphabetically**

#include <iostream>

#include <string>

using namespace std;

int main() {

int n;

// Input number of names

cout << "Enter the number of names: ";

cin >> n;

string names[n];

// Input names

cout << "Enter " << n << " names:\n";

for (int i = 0; i < n; ++i) {

cin >> names[i];

}

// Sorting names alphabetically using simple bubble sort

for (int i = 0; i < n - 1; ++i) {

for (int j = i + 1; j < n; ++j) {

if (names[i] > names[j]) {

// Swap names[i] and names[j]

string temp = names[i];

names[i] = names[j];

names[j] = temp;

}

}

}

// Output sorted names

cout << "\nNames in alphabetical order:\n";

for (int i = 0; i < n; ++i) {

cout << names[i] << endl;

}

return 0;

}

**C++ Program: Palindrome Check (Without Using Reverse Function)**

// Palindrome Check (Without Using Reverse Function)

#include <iostream>

#include <string>

using namespace std;

int main() {

string str;

bool isPalindrome = true;

cout << "Enter a string: ";

cin >> str;

int length = str.length();

// Compare characters from start and end

for (int i = 0; i < length / 2; i++) {

if (str[i] != str[length - 1 - i]) {

isPalindrome = false;

break;

}

}

if (isPalindrome)

cout << "The string is a palindrome." << endl;

else

cout << "The string is not a palindrome." << endl;

return 0;

}

**C++ Program: Palindrome Check using reverse function**

// C++ Program: Palindrome Check using reverse function

#include <iostream>

#include <string>

using namespace std;

int main() {

string str, reversedStr;

cout << "Enter a string: ";

cin >> str;

// Reverse the string

reversedStr = string(str.rbegin(), str.rend());

// Check if original and reversed strings are equal

if (str == reversedStr)

cout << "The string is a palindrome." << endl;

else

cout << "The string is not a palindrome." << endl;

return 0;

}

/\*

Multi line comment

This is the first program

\*/

#include <iostream>

int main()

{

int a, b, sum;

double average;

std::cout<< "Enter 2 numbers";

std::cin >> a >> b;

sum = a + b;

//average = (double)sum / 2;

average = sum / 2.0;

std::cout<< "Sum = " << sum << "\n" ;

std::cout<< "Average = " << average;

return 0;

}

Program with Type Casting

**// Program for Fahrenheit to Celsius Conversion**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int fahrenheit;**

**float celsius;**

**// Input temperature in Fahrenheit**

**cout << "Enter temperature in Fahrenheit: ";**

**cin >> fahrenheit;**

**// Convert to Celsius**

**celsius = ((float)fahrenheit - 32) \* 5.0 / 9;**

**// Output result**

**cout << "Temperature in Celsius: " << celsius << "°C" << endl;**

**return 0;**

**}**

- - - - -

// Find out the largest number

#include <iostream>

using namespace std;

int main() {

int a , b, c;

int large;

// Input the 3 numbers;

cout << "Enter the three numbers: ";

cin >> a >> b >> c;

// Ternary operator ----- ? :

// (Conditional Expression ) ? true part : False part

// Larger between 2 numbers

large = (a > b) ? a : b;

cout << "Larger between a and b = " << large;

// Largest among 3 numbers -- Way

large = (a > b && a > c) ? a : ( (b > c) ? b : c );

cout << "\n \n Largest among 2 - Way 1 = " << large;

// Largest among 3 numbers

large = (a > b ) ? ( (a>c) ? a : c ) : ( (b > c) ? b : c );

cout << "\n \n Largest among 3 - Way 2 = " << large;

return 0;

}

- - - -

// Program for Factorial of a Number -- Using FOR, While and DO… WHILE

#include <iostream>

using namespace std;

int main() {

int n;

unsigned long factorial = 1, fact1 = 1, fact2 = 1;

int i, i1, i2;

// Input a number

cout << "Enter a positive integer: ";

cin >> n;

// Check for valid input

if (n < 0) {

cout << "Factorial is not defined for negative numbers." << endl;

} else {

// Calculate factorial -- Using FOR loop

for ( i = 100; i <= n; ++i)

{

factorial = factorial \* i;

}

// Calculate factorial -- Using WHILE loop

i1 = 100;

while ( i1 <= n)

{

fact1 = fact1 \* i1 ;

i1++;

}

// Calculate factorial -- Using WHILE loop

i2 = 100;

do

{

fact2 = fact2 \* i2 ;

i2++;

}while ( i2 <= n) ;

// Output the result

cout << "Factorial of " << n << " is " << factorial << "--Using for " << endl;

cout << "Factorial of " << n << " is " << fact1 << "--Using WHILE " << endl;

cout << "Factorial of " << n << " is " << fact2 << "--Using DO.... WHILE " << endl;

}

return 0;

}

- - - - - -

// Program for checking the number is Prime or Not --- In a given range

#include <iostream>

using namespace std;

int main() {

int num1, num2 , n , i;

bool isPrime = true;

// Input the number

cout << "Enter 2 positive integers: ";

cin >> num1 >> num2;

// Check divisibility from 2 tonum/2

for (n = num1; n <= num2 ; n++) // Loop for range

{

isPrime = true;

for (i = 2; i <= n/2; ++i) // Loop for prime checking

{

if (n % i == 0) {

isPrime = false;

break;

}

}

if (isPrime) // Display prime or not

{

cout << n << " is a prime number." << endl;

}

else

{

cout << n << " is not a prime number." << endl;

}

}

return 0;

}

* **Bitwise Operators in C++**
* Bit operators work on 32 bits numbers.
* Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a C++ number.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Same as** | **Result** | **Decimal** |
| & | AND | 5 & 1 | 0101 & 0001 | 0001 | 1 |
| | | OR | 5 | 1 | 0101 | 0001 | 0101 | 5 |
| ~ | NOT | ~ 5 | ~0101 | 1010 | 10 |
| ^ | XOR | 5 ^ 1 | 0101 ^ 0001 | 0100 | 4 |
| << | left shift | 5 << 1 | 0101 << 1 | 1010 | 10 |
| >> | right shift | 5 >> 1 | 0101 >> 1 | 0010 | 2 |
| >>> | unsigned right shift | 5 >>> 1 | 0101 >>> 1 | 0010 | 2 |

Example:

#include <iostream>

using namespace std;

int main() {

int a = 5; // Binary: 0101

int b = 3; // Binary: 0011

cout << "a & b = " << (a & b) << endl; // 0101 & 0011 = 0001 -> 1

cout << "a | b = " << (a | b) << endl; // 0101 | 0011 = 0111 -> 7

cout << "a ^ b = " << (a ^ b) << endl; // 0101 ^ 0011 = 0110 -> 6

cout << "~a = " << (~a) << endl; // ~0101 = 1010 (in 2's complement: -6)

cout << "a << 1 = " << (a << 1) << endl; // 0101 << 1 = 1010 -> 10

cout << "a >> 1 = " << (a >> 1) << endl; // 0101 >> 1 = 0010 -> 2

return 0;

}

**Output**

a & b = 1

a | b = 7

a ^ b = 6

~a = -6

a << 1 = 10

a >> 1 = 2

### Note:

* Bitwise operations are **only applicable to integral types** (like int, char, short, long, etc.).
* For signed integers, the result of right shift (>>) may vary between **arithmetic** and **logical** shifts depending on the compiler.
* Smaller types like char or short are **promoted to int** before the bitwise operation.
* If the value is already int, it stays 32-bit. If it's smaller (like char or short), it's promoted to 32-bit int before the operation.

**Constants in C++**

In **C++**, a **constant** is a value that **cannot be changed after it is defined**. Constants improve **code safety, readability**, and **prevent accidental modification**.

**🔹 Types of Constants in C++**

| **Type** | **Example** | **Description** |
| --- | --- | --- |
| const keyword | const int x = 10; | Declares a constant variable |
| #define macro | #define PI 3.14 | Preprocessor macro constant |
|  |  |  |
| enum | enum Color { RED, GREEN }; | Integer constants using enum |
|  |  |  |

**1. const Keyword**

#include <iostream>

using namespace std;

int main() {

const int x = 10;

// x = 20; // ❌ Error: cannot modify a const variable

cout << x;

return 0;

}

* **Const must be initialized when declared.**
* Commonly used for safety.

**2. #define Preprocessor Constant**

#include <iostream>

using namespace std;

#define PI 3.14159

const int AGE = 18;

int main() {

int AGE = 23;

cout << "PI = " << PI << endl;

cout << "Vote Age = " << ::AGE << endl;

cout << "AGE = " << AGE << endl;

if (AGE >= ::AGE)

cout << "Eligible to Vote";

else

cout << "Not Eligible to vote";

return 0;

}

**NOTE**

**No type safety when using #define  
Simple text replacement by the preprocessor  
Not recommended for modern C++ (use const)**

**NOTE:** ::AGE à :: is the Global variable accessing operator

**Increment and Decrement operators**

[Pre-increment ( ++a ) and Post-increment ( a++ ) operators]

### ++A and A++ Operators in C++

Both ++A and A++ are **increment operators** used to increase the value of a variable by 1.  
However, there's a key difference in **when** the increment happens.

### Syntax and Meaning

| **Operator** | **Name** | **Description** |
| --- | --- | --- |
| ++A | Pre-increment | Increments the value **before** using it |
| A++ | Post-increment | Uses the value **before** incrementing it |

### Example:

#include <iostream>

using namespace std;

int main() {

int A = 5;

cout << "Initial A = " << A << endl;

cout << "Using ++A: " << ++A << endl; // A becomes 6, then prints 6

cout << "After ++A, A = " << A << endl;

cout << "Using A++: " << A++ << endl; // Prints 6, then A becomes 7

cout << "After A++, A = " << A << endl;

return 0;

}

### Output:

Initial A = 5

Using ++A: 6

After ++A, A = 6

Using A++: 6

After A++, A = 7

A= 5;

B= 10;

C = ++A + B--;

D = ++A + --B;