Introduction to C++

C++ is a general-purpose, statically typed, free-form, compiled, multi-paradigm programming language. It supports procedural, object-oriented, and generic programming. It was developed by Bjarne Stroustrup at Bell Labs in 1979 as an extension of the C language.

Basic Building Blocks of C++

Data Types

Data types classify the type of data that a variable can hold. They determine the amount of memory allocated and the type of operations that can be performed.

* Primary (Built-in) Data Types: These are the fundamental data types.

  * int: Stores whole numbers (e.g., 10, -5).

  * float: Stores single-precision floating-point numbers (e.g., 3.14).

  * double: Stores double-precision floating-point numbers, offering more precision than float.

  * char: Stores a single character (e.g., 'A', 'c').

  * bool: Stores boolean values, either true or false.

  * void: An empty data type, used to specify that a function does not return a value.

Keywords

Keywords are reserved words with predefined meanings in C++. They cannot be used as identifiers (variable names, function names, etc.). Examples include int, if, for, while, return, class, etc.

Identifiers

Identifiers are user-defined names given to variables, functions, classes, etc. They must start with a letter or an underscore, followed by letters, numbers, or underscores. They are also case-sensitive. Examples: myVariable, sum_of_numbers, _name.

Variables

A variable is a named storage location that holds a value. The value of a variable can be changed during program execution. Before using a variable, it must be declared with a data type and an identifier.

Syntax: dataType variableName = value;

Operators and Their Priority

Operators are symbols that perform operations on variables and values.

* Arithmetic Operators: Perform mathematical calculations (+, -, *, /, %).

* Relational Operators: Compare two values and return a boolean result (==, !=, <, >, <=, >=).

* Logical Operators: Combine multiple conditions (&& (AND), || (OR), ! (NOT)).

* Assignment Operators: Assign a value to a variable (=, +=, -=, *=, /=, %=).

* Increment/Decrement Operators: Increase or decrease a variable's value by one (++, --).

Operator Precedence: The order in which operators are evaluated in an expression. For example, multiplication and division have higher precedence than addition and subtraction. Parentheses () can be used to override the default precedence.

Control Flow Statements

Conditional Statements

Conditional statements execute different blocks of code based on whether a condition is true or false.

* if-else statement: Executes a block of code if a condition is true, and an optional alternative block if it's false.

Syntax:

```
if (condition) {
  // code to execute if condition is true
} else {
  // code to execute if condition is false
}
```

* if-else if-else statement: Used for multiple conditions.

* switch statement: A multi-way branch statement that allows you to test the value of an expression against a list of possible constant values.

Syntax:

```
switch (expression) {
  case constant1:
    // code block 1
    break;
  case constant2:
    // code block 2
    break;
  default:
    // default code block
}
```

Looping Statements

Looping statements execute a block of code repeatedly as long as a condition is true.

* for loop: Used for a fixed number of iterations.

Syntax:

```
for (initialization; condition; update) {
  // code to repeat
}
```

* while loop: Repeats a block of code as long as a condition is true. The condition is checked at the beginning of the loop.

Syntax:

```
while (condition) {
  // code to repeat
```

}

* do-while loop: Similar to a while loop, but the code block is executed at least once before the condition is checked.

Syntax:

```
do {
  // code to repeat
} while (condition);
```

Unconditional Statements

These statements change the flow of control unconditionally.

* break: Terminates the innermost loop or switch statement and transfers control to the statement immediately following it.

* continue: Skips the rest of the current iteration of a loop and continues with the next iteration.

* goto: Transfers control to a labeled statement. Its use is generally discouraged as it can lead to unstructured and hard-to-read code.

Number Systems

A number system is a way of representing numbers.

* Decimal (Base 10): Uses digits 0 through 9. This is the most common system.

* Binary (Base 2): Uses digits 0 and 1. This is the fundamental system for computers.

* Octal (Base 8): Uses digits 0 through 7.

* Hexadecimal (Base 16): Uses digits 0 through 9 and letters A through F (representing values 10 through 15).

Types of Programming Languages

* Low-level Languages: Close to the hardware and require less translation.

* Machine Language: Consists of binary instructions (0s and 1s). Directly executable by the computer.

* Assembly Language: Uses mnemonics (e.g., ADD, MOV) for instructions. Requires an assembler to translate it into machine code.

* High-level Languages: Use a more human-readable syntax and are platform-independent. They require a compiler or interpreter to translate them into machine code. Examples: C++, Python, Java.

Types of Errors

Errors prevent a program from compiling or running correctly.

* Syntax Errors: Occur due to violations of the language's grammar rules. The compiler detects these errors. Example: forgetting a semicolon at the end of a statement.

* Logical Errors: The program compiles and runs, but the output is incorrect. These are harder to find as the program doesn't crash. Example: using + instead of - in a calculation.

* Runtime Errors: Occur during program execution. Example: division by zero or trying to access an invalid memory location.

Arrays

An array is a collection of elements of the same data type stored in contiguous memory locations.

* One-Dimensional Array: A list of elements.

Syntax: dataType arrayName[size];

* Two-Dimensional Array: An array of arrays, often used to represent a matrix or table.

Syntax: dataType arrayName[rows][columns];

Sample Programs

1. Print N Natural Numbers

```
#include <iostream>

int main() {

   int n;
```

```
    std::cout << "Enter a number: ";
    std::cin >> n;
    for (int i = 1; i <= n; i++) {
        std::cout << i << " ";
    }
    return 0;
}
```

2. Sum of Digits

```
#include <iostream>
int main() {
    int n, sum = 0, digit;
    std::cout << "Enter a number: ";
    std::cin >> n;
    while (n != 0) {
        digit = n % 10;
        sum += digit;
        n /= 10;
    }
    std::cout << "Sum of digits: " << sum;
    return 0;
}
```

3. Palindrome or Not

```
#include <iostream>
int main() {
   int n, reversed = 0, remainder, original;
   std::cout << "Enter a number: ";
   std::cin >> n;
   original = n;
   while (n != 0) {
      remainder = n % 10;
      reversed = reversed * 10 + remainder;
      n /= 10;
   }
   if (original == reversed) {
      std::cout << original << " is a palindrome.";
   } else {
      std::cout << original << " is not a palindrome.";
   }
   return 0;
}
```

4. GCD of 2 Numbers

```
#include <iostream>
int gcd(int a, int b) {
   if (b == 0) return a;
   return gcd(b, a % b);
```

```
}
int main() {
   int num1, num2;
   std::cout << "Enter two numbers: ";
   std::cin >> num1 >> num2;
   std::cout << "GCD of " << num1 << " and " << num2 << " is " << gcd(num1, num2);
   return 0;
}
```

5. Matrix Addition (2D Array)

```
#include <iostream>
int main() {
   int A[2][2] = {{1, 2}, {3, 4}};
   int B[2][2] = {{5, 6}, {7, 8}};
   int C[2][2];
   for (int i = 0; i < 2; i++) {
      for (int j = 0; j < 2; j++) {
         C[i][j] = A[i][j] + B[i][j];
      }
   }
   std::cout << "Resultant Matrix: \n";
   for (int i = 0; i < 2; i++) {
      for (int j = 0; j < 2; j++) {
         std::cout << C[i][j] << " ";
```

```
        }
        std::cout << "\n";
    }
    return 0;
}
```

6. Find Maximum of an Array

```
#include <iostream>
int main() {
    int arr[] = {10, 5, 20, 15, 25};
    int n = sizeof(arr) / sizeof(arr[0]);
    int max = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    std::cout << "Maximum element is: " << max;
    return 0;
}
```

7. Switch Case Program (Day of the Week)

```
#include <iostream>
int main() {
```

```
    int day;
    std::cout << "Enter day number (1-7): ";
    std::cin >> day;
    switch (day) {
        case 1: std::cout << "Monday"; break;
        case 2: std::cout << "Tuesday"; break;
        case 3: std::cout << "Wednesday"; break;
        case 4: std::cout << "Thursday"; break;
        case 5: std::cout << "Friday"; break;
        case 6: std::cout << "Saturday"; break;
        case 7: std::cout << "Sunday"; break;
        default: std::cout << "Invalid day number";
    }
    return 0;
}
```

8. Leap Year or Not

```
#include <iostream>
int main() {
    int year;
    std::cout << "Enter a year: ";
    std::cin >> year;
    if ((year % 400 == 0) || (year % 4 == 0 && year % 100 != 0)) {
        std::cout << year << " is a leap year.";
```

```
    } else {
        std::cout << year << " is not a leap year.";
    }
    return 0;
}
```

9. Biggest among 2 Numbers

```
#include <iostream>
int main() {
    int num1, num2;
    std::cout << "Enter two numbers: ";
    std::cin >> num1 >> num2;
    if (num1 > num2) {
        std::cout << num1 << " is the biggest.";
    } else {
        std::cout << num2 << " is the biggest.";
    }
    return 0;
}
```

10. Biggest among 3 Numbers

```
#include <iostream>
int main() {
    int num1, num2, num3;
```

```
    std::cout << "Enter three numbers: ";
    std::cin >> num1 >> num2 >> num3;
    if (num1 >= num2 && num1 >= num3) {
        std::cout << num1 << " is the biggest.";
    } else if (num2 >= num1 && num2 >= num3) {
        std::cout << num2 << " is the biggest.";
    } else {
        std::cout << num3 << " is the biggest.";
    }
    return 0;
}
```

11. Print 2nd Tables

```
#include <iostream>
int main() {
    int num = 2;
    for (int i = 1; i <= 10; i++) {
        std::cout << num << " x " << i << " = " << num * i << "\n";
    }
    return 0;
}
```

12. Factorial

```
#include <iostream>
```

```
int main() {
    int n, factorial = 1;
    std::cout << "Enter a non-negative number: ";
    std::cin >> n;
    if (n < 0) {
        std::cout << "Factorial is not defined for negative numbers.";
    } else {
        for (int i = 1; i <= n; i++) {
            factorial *= i;
        }
        std::cout << "Factorial of " << n << " is " << factorial;
    }
    return 0;
}
```