Basic Programming Terminology

Before we begin coding, it's important to understand some key terms that are frequently used in programming. Let’s go through them:

1. Keywords:

Keywords are reserved words in a programming language that have predefined meanings and cannot be used as variable or function names. For example:

- `for`: Used to create loops that repeat actions.
- `while`: Used to execute a block of code repeatedly.
- `true` and `false`: Boolean values that represent truth and falsity.

2. Variables:

Variables are named storage locations in a program that can hold a value of a particular data type. In C++, variables must be declared with a specific data type before they can be used. Common data types in C++ include "int" for integers, "float" for decimal numbers, and "char" for characters.

3. Functions:

A function is a block of code that performs a specific task. For example, max(a, b) is a function that takes two inputs and returns the larger value. Functions like printf() in C/C++ take inputs (like a string) and output them to the screen. You can define your own functions for repeated tasks in your code.

4. Object-Oriented Programming (OOP):

There are two main programming paradigms:

- **Functional Programming**: Everything is done via functions calling other functions.
- **Object-Oriented Programming (OOP)**: Instead of functions calling each other, you create **objects** that interact with each other.

OOP is ideal for larger software projects as it organizes code into classes and objects. In OOP:

- A **class** is a blueprint that defines the properties and behaviors of objects. For example, a `Student` class might have properties like `name` and `enrollment number`, and behaviors like `enroll()` or `graduate()`.
- An **object** is an instance of a class. For example, `student1` can be an object created from the `Student` class.

C++ supports both **functional programming** (with functions interacting) and **object-oriented programming** (with objects interacting).

5. Primitive and Non-Primitive Types

C++ and Java support both:

- **Primitive types**: Variables like `int`, `float`, and `char` that store basic data values.
- **Non-Primitive types**: Objects that are instances of classes.

In **Python**, everything is an object, so there are no primitive types like in C++ or Java.

6. Statically Typed vs Dynamically Typed:

- **Statically Typed Languages** (C++ and Java): Variables must be declared with a specific type before use. For example, you must specify `int x = 5;` before using `x`.
- **Dynamically Typed Languages** (Python): Variables don't need an explicit type declaration. You can assign a value to a variable without

specifying its type, like `x = 5`, and later change `x` to store a string.

7. Header Files:

In C++, header files are files that contain declarations for functions, variables, and other constructs. These files are included in C++ programs using the preprocessor directive #include. Header files allow the program to use functions and variables without having to know the implementation details. For example, #include <iostream> allows the use of input/output operations like cout and cin.

8. Namespace:

A **namespace** is a container for organizing code into logical groups. For example, `std` is a namespace that contains standard libraries and functions in C++. The advantage of namespaces is that they prevent **name collisions**. If two different libraries or parts of your code define a variable with the same name, namespaces allow them to coexist by distinguishing them. For example, `std::cout` refers to the `cout` function in the `std` namespace.

Mark as Read

Report An Issue