**Variables:-**

* A **Variable** is a named storage location in memory that holds a value.
  + A variable name can only contain **letters** (a-z, A-Z), **numbers** (0-9), and the **underscore** (\_).
  + The first character **must be a letter or an underscore**. It cannot be a number.
  + Names are **case-sensitive**. For example, age and Age are two different variables.
  + You **cannot use C keywords** as variable names (e.g., int, if, return).

**Data Type:-**

* A **data type** is a classification that specifies the type of value a variable can hold, the amount of memory it occupies, and the set of operations that can be performed on it.
* **Types of Data Types in C:-**
  + Primary (or Basic) Data Types :-
    - **int**: Used for storing whole numbers (integers). **Example: 10, -5, 1000.**
    - **char**: Used for storing a single character. **Example: 'A', 'b', '5'.**
    - **float**: Used for storing single-precision floating-point numbers (numbers with decimal points). **Example: 3.14, -0.05.**
    - **double**: Used for storing double-precision floating-point numbers, which have a much larger range and more precision than float. **Example: 12345.6789.**
    - **void**: A special type that indicates the absence of a value. It's primarily used for functions that do not return anything.

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Size in Bytes** | **Format Specifier** |
| int | 4 | %d |
| char | 1 | %c |
| float | 4 | %f |
| double | 8 | %lf |

* + Derived Data Types :-

These types are derived from the primary data types.

* + - **Arrays**: A collection of a fixed number of elements of the same data type.
    - **Pointers**: A special variable that stores the memory address of another variable.
    - **Functions**: A block of code that performs a specific task and can be called from other parts of the program.
  + User-Defined Data Types :-

These types are defined by the programmer to suit their specific needs.

* + - **Structure (struct)**: A collection of variables of different data types grouped under a single name. This allows you to create complex data records.
    - **Union (union)**: Similar to a structure, but all its members share the same memory location. Only one member can hold a value at any given time.
    - **Enumeration (enum)**: A special type that consists of a set of named integer constants, making the code more readable.

**KeyWords:-**

* **Keyword** is a reserved word that has a predefined meaning and purpose for the compiler.

|  |  |  |  |
| --- | --- | --- | --- |
| auto | double | int | struct |
| break | else | long | switch |
| case | enum | register | typedef |
| char | extern | return | union |
| continue | for | signed | void |
| do | if | static | while |
| default | goto | sizeof | volatile |
| const | float | short | unsigned |

**Operators:-**

* An **operator** is a symbol that tells the compiler to perform a specific mathematical, relational, or logical operation.

Arithmetic Operators :- These are used for performing mathematical calculations.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| + | Addition | a + b |
| - | Substraction | a - b |
| \* | Multiplication | a \* b |
| / | Divison | a / b |
| % | Modulus | a % b |

Relational Operators :- These are used to compare two values and return a true (1) or false (0) result.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| = = | Equal to | a = = b |
| ! = | Not Equal to | a ! = b |
| > | Greater Than | a > b |
| < | Less Than | a < b |
| > = | Greater Than Equal to | a > = b |
| < = | Less Than Equal to | a < = b |

Logical Operators :- These are used to combine multiple conditions.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Description** |
| && | Logical AND | Returns true (1) if **both** conditions are true. |
| ! | Logical NOT | Reverses the logical state. |
| | | | Logical OR | Returns (1) if either conditions are true. |

Assignment Operators :- These are used to assign values to the variables.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Example** | **Equivalent to** |
| = | a=5 | a = 5 |
| + = | a+ =5 | a = a + 5 |
| - = | a- =5 | a = a - 5 |
| \* = | a\*=5 | a = a \* 5 |
| / = | a/=5 | a = a / 5 |