# C Variables :-

Variables are containers for storing data values.

In C, there are different **types** of variables (defined with different keywords), for example:

- `int` - stores integers (whole numbers), without decimals, such as 123 or -123
- `float` - stores floating point numbers, with decimals, such as 19.99 or -19.99
- `char` - stores single characters, such as 'a' or 'B'. Char values are surrounded by **single quotes**

## Declaring (Creating) Variables

To create a variable, specify the **type** and assign it a **value**:

### Syntax

```
type variableName = value;
```

Where *type* is one of C types (such as `int`), and *variableName* is the name of the variable (such as **x** or **myName**). The **equal sign** is used to assign a value to the variable.

So, to create a variable that should **store a number**, look at the following example:

### Example

Create a variable called **myNum** of type `int` and assign the value **15** to it:

```
int myNum = 15;
```

You can also declare a variable without assigning the value, and assign the value later:

## Example

```
int myNum;
myNum = 15;
```

**Note:** If you assign a new value to an existing variable, it will overwrite the previous value:

## Example

```
int myNum = 15;  // myNum is 15
myNum = 10;  // Now myNum is 10
```

# Output Variables

You learned from the output chapter that you can output values/print text with the `printf()` function:

## Example

```
printf("Hello World!");
```

In many other programming languages (like Python, Java, and C++), you would normally use a **print function** to display the value of a variable. However, this is not possible in C:

## Example

```
int myNum = 15;
printf(myNum);  // Nothing happens
```

To output variables in C, you must get familiar with something called "format specifiers".

# Format Specifiers

Format specifiers are used together with the `printf()` function to tell the compiler what type of data the variable is storing. It is basically a placeholder for the variable value.

A format specifier starts with a percentage sign `%`, followed by a character.

For example, to output the value of an `int` variable, you must use the format specifier `%d` or `%i` surrounded by double quotes, inside the `printf()` function:

## Example

```
int myNum = 15;
printf("%d", myNum);  // Outputs 15
```

To print other types, use `%c` for `char` and `%f` for `float`:

## Example

```
// Create variables
int myNum = 5;               // Integer (whole number)
float myFloatNum = 5.99;     // Floating point number
char myLetter = 'D';         // Character

// Print variables
printf("%d\n", myNum);
printf("%f\n", myFloatNum);
printf("%c\n", myLetter);
```

To combine both text and a variable, separate them with a comma inside the `printf()` function:

## Example

```
int myNum = 5;
printf("My favorite number is: %d", myNum);
```

To print different types in a single printf() function, you can use the following:

## Example

```c
int myNum = 5;
char myLetter = 'D';
printf("My number is %d and my letter is %c", myNum, myLetter);
```

# Add Variables Together

To add a variable to another variable, you can use the `+` operator:

## Example

```c
int x = 5;
int y = 6;
int sum = x + y;
printf("%d", sum);
```

# Declare Multiple Variables

To declare more than one variable of the same type, use a **comma-separated** list:

## Example

```c
int x = 5, y = 6, z = 50;
printf("%d", x + y + z);
```

You can also assign the **same value** to multiple variables of the same type:

## Example

```c
int x, y, z;
x = y = z = 50;
printf("%d", x + y + z);
```

# C Variable Names

All C **variables** must be **identified** with **unique names**.

These unique names are called **identifiers**.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

**Note:** It is recommended to use descriptive names in order to create understandable and maintainable code:

## Example

```
// Good
int minutesPerHour = 60;

// OK, but not so easy to understand what m actually is
int m = 60;
```

The **general rules** for naming variables are:

- Names can contain letters, digits and underscores
- Names must begin with a letter or an underscore (_)
- Names are case sensitive (`myVar` and `myvar` are different variables)
- Names cannot contain whitespaces or special characters like !, #, %, etc.
- Reserved words (such as `int`) cannot be used as names