Variables in C Language

When we want to store any information(data) on our computer/laptop, we store it in the computer's memory space. Instead of remembering the complex address of that memory space where we have stored our data, our operating system provides us with an option to create folders, name them, so that it becomes easier for us to find it and access it.

Similarly, in C language, when we want to use some data value in our program, we can store it in a memory space and name the memory space so that it becomes easier to access it.

The naming of an address is known as **variable**. Variable is the name of memory location. Unlike constant, variables are changeable, we can change value of a variable during execution of a program. A programmer can choose a meaningful variable name.

Datatype of Variable

A **variable** in C language must be given a type, which defines what type of data the variable will hold.

It can be:

- `char`: Can hold/store a character in it.
- `int`: Used to hold an integer.
- `float`: Used to hold a float value.
- `double`: Used to hold a double value.
- `void`

Example :

Data_type variable_name;

Example: average, height, age, total etc.

For example:

```
int Score = 95;
```

Here, `Score` is a variable of `int` type. Here, the variable is assigned an integer value `95`.

The value of a variable can be changed, hence the name variable.

```
char ch = 'a';
// some code
ch = 'l';
```

Type	Syntax
Variable declaration	data_type variable_name; Example: int x, y, z; char flat, ch;
Variable initialization	data_type variable_name = value; Example: int x = 50, y = 30; char flag = 'x', ch='l';

Naming Conventions

Generally, C programmers maintain the following conventions for naming variables.

- Start a variable name with lowercase letters.
- Try to use meaningful identifiers
- Separate "words" within identifiers with mixed upper and lowercase (for example empCode) or underscores (for example emp_code).
- For symbolic constants use all uppercase letters (for example #define LENGTH 100, #define MRP 45).

Rules to name a Variable

1. Variable name must not start with a digit.
2. Variable name can consist of alphabets, digits and special symbols like underscore _.
3. Blank or spaces are not allowed in variable name.
4. The variable name may not be a C reserved word (keyword).
5. Upper- and lower-case names are treated as different, as C is case-sensitive, so it is suggested to keep the variable names in lower case. That is the variable Precat is not the same as precat and PRECAT.
6. There is no rule on how long a variable name (identifier) can be. However, you may run into problems in some compilers if the variable name is longer than 31 characters.
7. Memory space is not allocated for a variable while declaration. It happens only on variable definition.
8. Variable initialization means assigning a value to the variable.

Some valid & invalid variable names

int money$owed;	←	Incorrect
int total_count;	←	Correct
int score2 ;	←	Correct
int 2ndscore;	←	Incorrect
int long;	←	Incorrect
_Bool x;	←	Correct (B must be capital letter or “bool”)

Brief About Declaring, Defining and Initializing a variable

Declaration of variables must be done before they are used in the program. Declaration does the following things.

1. It tells the compiler what the variable name is.
2. It specifies what type of data the variable will hold.
3. Until the variable is defined the compiler doesn't have to worry about allocating memory space to the variable.
4. Declaration is more like informing the compiler that there exist a variable with following datatype which is used in the program.
5. A variable is declared using the `extern` keyword, outside the `main()` function.

```
extern int a;

extern float b;

extern double c, d;
```

Defining a variable means the compiler has to now assign a storage to the variable because it will be used in the program. It is not necessary to declare a variable using `extern` keyword, if you want to use it in your program. You can directly define a variable inside the `main()` function and use it.

To define a function we must provide the datatype and the variable name. We can even define multiple variables of same datatype in a single line by using comma to separate them.

```
int a;

float b, c;
```

Initializing a variable means to provide it with a value. A variable can be initialized and defined in a single statement, like:

```
int a = 10;
```

Let's write a program in which we will use some variables.

main.c Ctrl+S

```c
/*****************************************
Statement - Variable Declaration and initialization
Written For - PRE-CAT COURSE by CCATPREPARATION.COM
*****************************************/
#include <stdio.h>

int main() {
 /* variable definition: */
  // multiple declarations and definitions
    int price, salary;
  // declaration and definition of variable
  int a = 3;
  float b = 4.5;
  double c = 5.25;
  // This is also both declaration and definition as 'b' is allocated
   // memory and assigned some garbage value.
    float b;
   // Let us print a variable
    printf("%d \n", a);
  return 0;
}
```

Another great analogy to understand the difference between Identifier and Variable is:

Identifier	Variable
Identifier is the name given to a variable, function etc.	While, variable is used to name a memory location which stores data.
An identifier can be a variable, but not all indentifiers are variables.	All variable names are identifiers.

You can think of an Identifier `int x` to be a variable's name, but it can also be a function's name `int x() { }` and still be an identifier.

Types of Variables in C

1. Local Variable

A variable that is declared and used inside the function or block is called local variable. It's scope is limited to function or block.

It cannot be used outside the block. Local variables need to be initialized before us.

2. Global Variable

A variable that is declared outside the function or block is called a global variable. It is declared at the starting of program. It is available to all the functions.

```c
/*****************************************
Statement - Local & Glocal Variable
Written For - PRE-CAT COURSE by CCATPREPARATION.COM
*****************************************/

#include <stdio.h>
int x = 20; //global variable
void Somefunction() {
    // local variable
     /*x variables are having scope within this main function only.
        These are not visible to test funtion.*/
    int x = 12;
      printf("\n local values: x=%d",x);
}
void test()
{
   printf("\nGlobal values: x=%d",x);
}
void main()
{
    Somefunction();
    test();
```

input

```
 local values: x=12
Global values: x=20
```

3.Static Variable

A variable that retains its value between multiple function calls is known as static variable.
It is declared with the static keyword.

main.c Ctrl+S

```c
/*****************************************
Statement - Static Variable
Written For - PRE-CAT COURSE by CCATPREPARATION.COM
*****************************************/

#include <stdio.h>
int x = 20; //global variable
void Somefunction() {

    int x = 12; // local variable
    static int y = 30;//static variable
    x = x + 10;
    y = y + 10;
    printf("\n%d,%d",x,y);
}

void main()
{
    Somefunction();
    Somefunction();
    Somefunction();
}
```

input

```
22,40
22,50
22,60
```