UNIT-1

Variables in C

A **variable** is a name of the memory location. It is used to store data. Its value can be changed, and it can be reused many times.

It is a way to represent memory location through symbol so that it can be easily identified.

Let's see the syntax to declare a variable:

1. type variable_list;

The example of declaring the variable is given below:

1. **int** a;
2. **float** b;
3. **char** c;

Here, a, b, c are variables. The int, float, char are the data types.

We can also provide values while declaring the variables as given below:

1. **int** a=10,b=20;//declaring 2 variable of integer type
2. **float** f=20.8;
3. **char** c='A';

Rules for defining variables

- A variable can have alphabets, digits, and underscore.
- A variable name can start with the alphabet, and underscore only. It can't start with a digit.
- No whitespace is allowed within the variable name.
- A variable name must not be any reserved word or keyword, e.g. int, float, etc.

Valid variable names:

1. **int** a;
2. **int** _ab;

3. **int** a30;

Invalid variable names:

1. **int** 2;
2. **int** a b;
3. **int long**;

Types of Variables in C

There are many types of variables in c:

1. **local variable**
2. **global variable**
3. **static variable**
4. **automatic variable**
5. **external variable**

---

1. Local Variable

A variable that is declared inside the function or block is called a local variable.

It must be declared at the start of the block.

```
1. void function1(){
2. int x=10;//local variable
3. }
```

You must have to initialize the local variable before it is used.

2. Global Variable

A variable that is declared outside the function or block is called a global variable. Any function can change the value of the global variable. It is available to all the functions.

It must be declared at the start of the block.

```
1. int value=20;//global variable
2. void function1(){
3. int x=10;//local variable
4. }
```

3. Static Variable

A variable that is declared with the static keyword is called static variable.

It retains its value between multiple function calls.

```
1. void function1(){
2. int x=10;//local variable
3. static int y=10;//static variable
4. x=x+1;
5. y=y+1;
6. printf("%d,%d",x,y);
7. }
```


If you call this function many times, the **local variable will print the same value** for each function call, e.g, 11,11,11 and so on. But the **static variable will print the incremented value** in each function call, e.g. 11, 12, 13 and so on.

4. Automatic Variable

All variables in C that are declared inside the block, are automatic variables by default. We can explicitly declare an automatic variable using **auto keyword**.

```
1. void main(){
2. int x=10;//local variable (also automatic)
3. auto int y=20;//automatic variable
```

```
4. }
```

5. External Variable

We can share a variable in multiple C source files by using an external variable. To declare an external variable, you need to use **extern keyword**.

myfile.h

```
1. extern int x=10;//external variable (also global)
```

program1.c

```
1. #include "myfile.h"
2. #include <stdio.h>
3. void printValue(){
4.     printf("Global variable: %d", global_variable);
5. }
```