UNIT 1 BASICS OF C PROGRAMMING **6**

Problem Solving Techniques: Introduction to Algorithm, Pseudo code, Flow Chart, Structure of 'C' program. C Tokens: Keywords, Data Types, Constants, Variables - Declaration - Qualifiers – typedef

1.5 VARIABLE IN C

A variable is an Identifier used for naming entity in Programming. Variable C is a memory location with some name that helps store some form of data and retrieves it when required. They can be viewed as the names given to the memory location so that we can refer to it without having to memorize the memory address.

The size of the variable depends upon the data type it stores.

C Variable Syntax

The syntax to declare a variable in C specifies the name and the type of the variable.

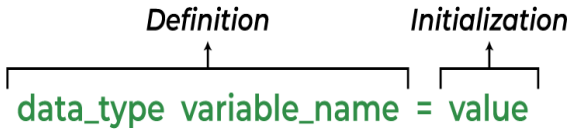
```
datatype variablename = value ;          // defining single variable
        or
datatype variable_name1, variable_name2 ;   // defining multiple variable
```

Here,

- **data_type:** Type of data that a variable can store.
- **variable_name:** Name of the variable given by the user.
- **value:** value assigned to the variable by the user.

Example

```
int var;          // integer variable
char a;           // character variable
float fff;        // float variables
```



There are 2 aspects of defining a variable:

1. Variable Declaration
2. Variable Initialization

1) Variable Declaration

Variable declaration in C tells the compiler about the existence of the variable with the given name and data type.

When the variable is declared, an entry in symbol table is created and memory will be

allocated at the time of initialization of the variable.

Example

```
int var;
char var2;
```

2) Variable Initialization

Initialization of a variable is the process where the user assigns some meaningful value to the variable when creating the variable.

Example

```
int var = 10;       // variable declaration and definition (i.e. Vairable Initialization)
```

EXAMPLE

```
// C program to demonstrate the declaration, initialization
#include <stdio.h>
int main()
{
    // declaration
    int defined_var;
        // assignment
    defined_var = 12;
    // initialization
    int ini_var = 25;
    printf("Value of defined_var after assignment: %d\n", defined_var);
    printf("Value of ini_var: %d", ini_var);
    return 0;
}
```

Output

```
Value of defined_var after assignment: 12

Value of ini_var: 25
```

Rules for Naming Variables In C

1. A variable name must only contain alphabets, digits, and underscore.
2. A variable name must start with an alphabet or an underscore only. It cannot start with a digit.

3. No white space is allowed within the variable name.
4. A variable name must not be any reserved word or keyword.

Example: valid variable names _arjun, arjun_hari, arjun123, arjun_1, room_105

Invalid variable names: arjun hari, 1arjun, 105room

C VARIABLE TYPES

The C variables can be classified into the following types:

1. Local Variables
2. Global Variables
3. Static Variables
4. Automatic Variables
5. Extern Variables
6. Register Variables

1) Local Variables in C

A **Local variable in C** is a variable that is declared inside a function or a block of code. Its scope is limited to the block or function in which it is declared.

Example of Local Variable in C

```
// C program to declare and print local variable inside a function.
#include <stdio.h>
void function()
{
   int x = 10;      // local variable
   printf("%d", x);
}

int main()
{
function();
}
```

Output

```
10
```

In the above code, x can be used only in the scope of function (). Using it in the main function

will give an error.

2. Global Variables in C

A **Global variable in C** is a variable that is declared outside the function or a block of code. Its scope is the whole program i.e. we can access the global variable anywhere in the C program after it is declared.

Example of Global Variable in C

```
// C program to demonstrate use of global variable
#include <stdio.h>
int x = 20; // global variable
void function1()
{
printf("Function 1: %d\n", x);
}
void function2()
{
printf("Function 2: %d\n", x);
}
int main()
{
   function1();
   function2();
   return 0;
}
```

Output

```
Function 1: 20

Function 2: 20
```

3. Static Variables in C

A **static variable in C** is a variable that is defined using the **static** keyword. It can be defined only once in a C program and its scope depends upon the region where it is declared (can be **global or local**).

The **default value** of static variables is **zero.**

Syntax of Static Variable in C

```
static data_type variable_name = initial_value;
```

As its lifetime is till the end of the program, it can retain its value for multiple function calls as shown in the example.

```
/ C program to demonstrate use of static variable
#include <stdio.h>
void function()
{
    int x = 20;                    // local variable
    static int y = 30;             // static variable
    x = x + 10;
    y = y + 10;
    printf("\tLocal: %d\n\t Static: %d\n", x, y);
}
int main()
{
    printf("First Call\n");
    function();
    printf("Second Call\n");
    function();
    printf("Third Call\n");
    function();
    return 0;
}
```

Output

```
First Call
    Local: 30
    Static: 40
Second Call
    Local: 30
    Static: 50
Third Call
    Local: 30
    Static: 60
```

4. Automatic Variable in C

All the **local** variables are **automatic** variables **by default**. They are also known as auto variables.

Their scope is **local** and their lifetime is till the end of the **block.** If we need, we can use the **auto** keyword to define the auto variables.

The default value of the auto variables is a garbage value.

Syntax of Auto Variable in C

```
auto data_type variable_name;
        or
data_type variable_name;     (in local scope)
```

Example of auto Variable in C

```
// C program to demonstrate use of automatic variable
#include <stdio.h>
void function()
{
    int x = 10; // local variable (also automatic)
    auto int y = 20; // automatic variable
    printf("Auto Variable: %d", y);
}
int main()
{
    function();
    return 0;
}
```

Output

```
Auto Variable: 20
```

In the above example, both x and y are automatic variables. The only difference is that variable y is explicitly declared with the **auto** keyword.

5) External Variables in C

External variables in C can be **shared** between **multiple C files**. We can declare an external variable using the **extern** keyword.

Their scope is **global** and they exist between multiple C files.

Syntax of Extern Variables in C

```
extern data_type variable_name;
```

Example of Extern Variable in C

```
-------myfile.h--------
 extern int x=10;          //external variable (also global)


-------program1.c----------
#include "myfile.h"
#include <stdio.h>
void printValue(){
printf("Global variable: %d", x);
}
```

6) Register Variables in C

Register variables in C are those variables that are stored in the **CPU register** instead of the conventional storage place like RAM. Their scope is **local** and exists till the **end** of the **block** or a function.

- These variables are declared using the **register** keyword.
- The default value of register variables is a **garbage value**.

Syntax of Register Variables in C

```
register data_type variable_name = initial_value;
```

Example of Register Variables in C

```
//C program to demonstrate the definition of register variable
#include <stdio.h>
int main()
{
   register int var = 22;
   printf("Value of Register Variable: %d\n", var);
   return 0;
}
```

Output

```
Value of Register Variable: 22
```

1.1 CONSTANTS IN C

The constants in C are the read-only variables whose values cannot be modified once they are declared in the C program.

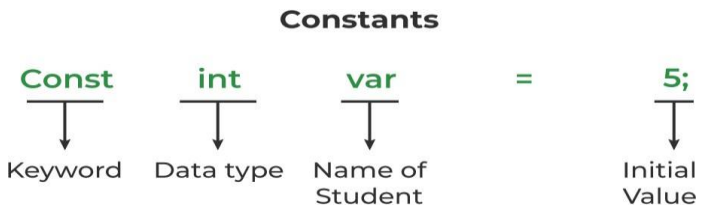
- The type of constant can be an integer constant, a floating pointer constant, a string constant, or a character constant.
- In C language, the **const** keyword is used to define the constants.
- As the name suggests, a constant in C is a variable that cannot be modified once it is declared in the program.
- We cannot make any change in the value of the constant variables after they are defined.

How to Define Constant in C?

We define a constant in C language using the **const** keyword. Also known as a const type qualifier, the const keyword is placed at the start of the variable declaration to declare that variable as a constant.

Syntax to Define Constant

```
const data_type var_name = value;
```

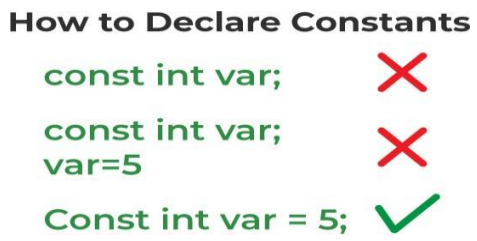


Example of Constants in C

```
// C program to illustrate consta
#include <stdio.h>
int main()
{
    // defining integer constant using const keyword
    const int int_const = 25;
    // defining character constant using const keyword
    const char char_const = 'A';

    // defining float constant using const keyword
    const float float_const = 15.66;
    printf("Printing value of Integer Constant: %d\n", int_const);
    printf("Printing value of Character Constant: %c\n",char_const);
    printf("Printing value of Float Constant: %f", float_const);
    return 0;
}
```

Output

```
Printing value of Integer Constant: 25

Printing value of Character Constant: A

Printing value of Float Constant: 15.660000
```



Types of Constants in C

The type of the constant is the same as the data type of the variables. Following is the list of the types of constants

- Integer Constant
- Character Constant
- Floating Point Constant
- Double Precision Floating Point Constant
- Array Constant
- Structure Constant

PROPERTIES OF CONSTANT IN C

The important properties of constant variables in C defined using the const keyword are as follows:

1. Initialization with Declaration

We can only initialize the constant variable in C at the time of its declaration. Otherwise, it will store the garbage value.

2. Immutability

The constant variables in c are immutable after its definition, i.e., they can be initialized only once in the whole program. After that, we cannot modify the value stored inside that variable.

```
// C Program to demonstrate the behaviour of constant
// variable
#include <stdio.h>
int main()
{
   // declaring a constant variable
```

```
    const int var;
    // initializing constant variable var after declaration
    var = 20;
    printf("Value of var: %d", var);
    return 0;
}
```

Output

```
In function 'main':
10:9: error: assignment of read-only variable 'var'
10 |     var = 20;
   |         ^
```