**Variable Definition in C**

Variable is basically the name of a memory location that one use for storing data. One can change the value of a variable in C or any other language, and one can also reuse it multiple times. One use symbols in variables for representing the memory location- so that it becomes easily identifiable by any user.

Use of the Variables in C

Variables are the storage areas in a code that the program can easily manipulate. Every variable in C language has some specific type- that determines the layout and the size of the memory of the variable, the range of values that the memory can hold, and the set of operations that one can perform on that variable.

The name of a variable can be a composition of digits, letters, and also underscore characters. The name of the character must begin with either an underscore or a letter. In the case of C, the lowercase and uppercase letters are distinct. It is because C is case-sensitive in nature. Let us look at some more ways in which one name a variable.

Rules for Naming a Variable in C

One can give a variable a meaningful name when one creates it. Here are the rules that one must follow when naming it:

1. The name of the variable must not begin with a digit.

2. A variable name can consist of digits, alphabets, and even special symbols such as an underscore ( \_ ).

3. A variable name must not have any keywords, for instance, float, int, etc.

4. There must be no spaces or blanks in the variable name.

5. The C language treats lowercase and uppercase very differently, as it is case sensitive. Usually, one keep the name of the variable in the lower case.

*Let us look at some of the examples,*

int var1; // it is correct

int 1var; // it is incorrect – the name of the variable should not start using a number

int my\_var1; // it is correct

int my$var // it is incorrect – no special characters should be in the name of the variable

char else; // there must be no keywords in the name of the variable

int my var; // it is incorrect – there must be no spaces in the name of the variable

int COUNT; // it is a new variable

int Count; // it is a new variable

int count; // it is a valid variable name

Data Type of the Variable

One must assign a data type to all the variables that are present in the C language. These define the type of data that one can store in any variable. If one do not provide the variable with a data type, the C compiler will ultimately generate a syntax error or a compile-time error.

The data types present in the C language are float, int, double, char, long int, short int, etc., along with other modifiers.

Types of Primary/ Primitive Data Types in C Language

The variables can be of the following basic types, based on the name and the type of the variable:

|  |  |  |  |
| --- | --- | --- | --- |
| **Type of Variable** | **Name** | **Description** | **Uses** |
| Char | Character | It is a type of integer. It is typically one byte (single octet). | One use them in the form of single alphabets, such as X, r, B, f, k, etc., or for the ASCII character sets. |
| Int | Integer | It is the most natural size of an integer used in the machine. | One use this for storing the whole numbers, such as 4, 300, 8000, etc. |
| Float | Floating- Point | It is a floating-point value that is single precision. | One use these for indicating the real number values or decimal points, such as 20.8, 18.56, etc. |
| Double | Double | It is a floating-point value that is double precision. | These are very large-sized numeric values that aren’t really allooned in any data type that is a floating-point or an integer. |
| Void | Void | It represents that there is an absence of type. | One use it to represent the absence of value. Thus, the use of this data type is to define various functions. |

*Let us look at a few examples,*

// int type variable in C

int marks = 45;

// char type variable in C

char status = ‘G’;

// double type variable in C

double long = 28.338612;

// float type variable in C

float percentage = 82.5;

If one tries to assign a variable with an incorrect value of datatype, then the compiler will (most probably) generate an error- the compile-time error. Or else, the compiler will convert this value automatically into the intended datatype of the available variable.

*Let us look at an example,*

#include <stdio.h>

int main() {

// assignment of the incorrect value to the variable

int a = 20.397;

printf(“Value is %d”, a);

return 0;

}

The output generated here will be:

20

As one can already look at this output- the compiler of C will remove the part that is present after the decimal. It is because the data types are capable of storing only the whole numbers.

One Cannot Change The Data Type

Once a user defines a given variable with any data type, then they will not be able to change the data type of that given variable in any case.

*Let us look at an example,*

// the int variable in C

int marks = 20;

float marks; // it generates an error

Variable Definition in C

The variable definition in C language tells the compiler about how much storage it should be creating for the given variable and where it should create the storage. Basically, the variable definition helps in specifying the data type. It contains a list of one variable or multiple ones as follows:

*type variable\_list;*

In the given syntax, *type* must be a valid data type of C that includes w\_char, char, float, int, bool, double, or any object that is user-defined. The *variable\_list*, on the other hand, may contain one or more names of identifiers that are separated by commas. Here one have shown some of the valid declarations:

char c, ch;

int p, q, r;

double d;

float f, salary;

Here, the line int p, q, r; defines as onell as declares the variables p, q, and r. It instructs the compiler to create three variables- named p, q, and r- of the type int.

One can initialize the variables in their declaration (assigned an initial value). The initializer of a variable may contain an equal sign- that gets follooned by a constant expression. It goes like this:

*type variable\_name = value;*

A few examples are −

extern int p = 3, q = 5; // for the declaration of p and q.

int p = 3, q = 5; // for the definition and initialization of p and q.

byte x = 22; // for the definition and initialization of x.

char a = ‘a’; // the variable x contains the value ‘a’.

In case of definition without the initializer: The variables with a static duration of storage are initialized implicitly with NULL (here, all bytes have a 0 value), while the initial values of all the other variables are not defined.

Declaration of Variable in C

Declaring a variable provides the compiler with an assurance that there is a variable that exists with that very given name. This way, the compiler will get a signal to proceed with the further compilation without needing the complete details regarding that variable.

The variable definition only has a meaning of its own during the time of compilation. The compiler would require an actual variable definition during the time of program linking.

The declaration of variables is useful when one use multiple numbers of files and one define the variables in one of the files that might be available during the time of program linking. One use the *extern* keyword for declaring a variable at any given place. Though one can declare one variable various times in a C program, one can only define it once in a function, a file, or any block of code.

Example

Let us look at the given example where one have declared the variable at the top and initialized and defined them inside the main function:

#include <stdio.h>

// Declaration of Variable

extern int p, q;

extern int c;

extern float f;

int main () {

/\* variable definition: \*/

int p, q;

int r;

float i;

/\* actual initialization \*/

p = 10;

q = 20;

r = p + q;

printf(“the value of r : %d \n”, r);

i = 70.0/3.0;

printf(“the value of i : %f \n”, i);

return 0;

}

The compilation and execution of the code mentioned above will produce the result as follows:

the value of r : 30

the value of i : 23.333334

Classification of Variables in C

The variables can be of the following basic types, based on the name and the type of the variable:

* **Global Variable**: A variable that gets declared outside a block or a function is known as a global variable. Any function in a program is capable of changing the value of a global variable. It means that the global variable will be available to all the functions in the code. Because the global variable in c is available to all the functions, one have to declare it at the beginning of a block. Explore, [Global Variable in C](https://byjus.com/gate/global-variable-in-c/) to know more.

*Example,*   
int value=30; // a global variable  
void function1(){  
int a=20; // a local variable  
}

* **Local Variable:**A local variable is a type of variable that one declare inside a block or a function, unlike the global variable. Thus, one also have to declare a local variable in c at the beginning of a given block.

*Example,*  
void function1(){  
int x=10; // a local variable  
}  
A user also has to initialize this local variable in a code before one use it in the program.

* **Automatic Variable:**Every variable that gets declared inside a block (in the C language) is by default automatic in nature. One can declare any given automatic variable explicitly using the keyword *auto*.

*Example,*  
void main(){  
int a=80; // a local variable (it is also automatic variable)  
auto int b=50; // an automatic variable  
}

* **Static Variable:** The [static variable in c](https://byjus.com/gate/static-variable-in-c/) is a variable that a user declares using the *static* keyword. This variable retains the given value between various function calls.

*Example,*void function1(){

int a=10; // A local variable

static int b=10; // A static variable

a=a+1;

b=b+1;

printf(“%d,%d”,a,b);

}

If one call this given function multiple times, then the local variable will print this very same value for every function call. For example, 11, 11, 11, and so on after this. The static variable, on the other hand, will print the value that is incremented in each and every function call. For example, 11, 12, 13, and so on.

* **External Variable:** A user will be capable of sharing a variable in multiple numbers of source files in C if they use an external variable. If one want to declare an external variable, then one need to use the keyword *extern*.

*Syntax,*extern int a=10;// external variable (also a global variable)