Topic:- Constants in C

What are literals

Tokens in C

Constants in C

A constant is a value or variable that can't be changed in the program, for example: 10, 20, 'a', 3.4, "c programming" etc.

There are different types of constants in C programming.

List of Constants in C

|  |  |
| --- | --- |
| **Constant** | **Example** |
| Decimal Constant | 10, 20, 450 etc. |
| Real or Floating-point Constant | 10.3, 20.2, 450.6 etc. |
| Octal Constant | 021, 033, 046 etc. |
| Hexadecimal Constant | 0x2a, 0x7b, 0xaa etc. |
| Character Constant | 'a', 'b', 'x' etc. |
| String Constant | "c", "c program", "GLA university" etc. |

2 ways to define constant in C

There are two ways to define constant in [C programming](https://www.javatpoint.com/c-programming-language-tutorial).

1. const keyword
2. #define preprocessor

1) C const keyword

The const keyword is used to define constant in C programming.

1. **const** **float** PI=3.14;

Now, the value of PI variable can't be changed.

1. #include<stdio.h>
2. **int** main(){
3. **const** **float** PI=3.14;
4. printf("The value of PI is: %f",PI);
5. **return** 0;
6. }

**Output:**

The value of PI is: 3.140000

If you try to change the the value of PI, it will render compile time error.

1. #include<stdio.h>
2. **int** main(){
3. **const** **float** PI=3.14;
4. PI=4.5;
5. printf("The value of PI is: %f",PI);
6. **return** 0;
7. }

**Output:**

Compile Time Error: Cannot modify a const object

2) C #define preprocessor

The #define preprocessor is also used to define constant. We will learn about #define preprocessor directive later.

# What are literals?

Literals are the constant values assigned to the constant variables. We can say that the literals represent the fixed values that cannot be modified. It also contains memory but does not have references as variables. For example, const int =10; is a constant integer expression in which 10 is an integer literal.

## Types of literals

**There are four types of literals that exist in**[**C programming**](https://www.javatpoint.com/c-programming-language-tutorial)**:**

* **Integer literal**
* **Float literal**
* **Character literal**
* **String literal**

### **Integer literal**

It is a numeric literal that represents only integer type values. It represents the value neither in fractional nor exponential part.

### **Float literal**

It is a literal that contains only floating-point values or real numbers. These real numbers contain the number of parts such as integer part, real part, exponential part, and fractional part. The floating-point literal must be specified either in decimal or in exponential form. Let's understand these forms in brief.

### **Decimal form**

The decimal form must contain either decimal point, exponential part, or both. If it does not contain either of these, then the compiler will throw an error. The decimal notation can be prefixed either by '+' or '-' symbol that specifies the positive and negative numbers.

**Examples of float literal in decimal form are:**

1. 1.2, +9.0, -4.5

### **Character literal**

A character literal contains a single character enclosed within single quotes. If multiple characters are assigned to the variable, then we need to create a character array. If we try to store more than one character in a variable, then the warning of a **multi-character character constant** will be generated. Let's observe this scenario through an example.

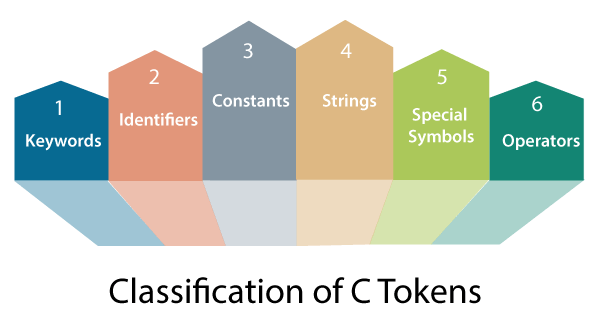
1. #include <stdio.h>
2. **int** main()
3. {
4. **const** **char** c='ak';
5. printf("%c",c);
6. **return** 0;
7. }

Tokens in C

Tokens in C is the most important element to be used in creating a program in C. We can define the token as the smallest individual element in C. For `example, we cannot create a sentence without using words; similarly, we cannot create a program in C without using tokens in C. Therefore, we can say that tokens in C is the building block or the basic component for creating a [program in C language](https://www.javatpoint.com/c-programs).

**Classification of tokens in C**

Tokens in [C language](https://www.javatpoint.com/c-programming-language-tutorial) can be divided into the following categories:



* Keywords in C
* Identifiers in C
* Strings in C
* Operators in C
* Constant in C
* Special Characters in C

Let's understand each token one by one.

46.6M

791

History of Java

**Keywords in C**

[Keywords in C](https://www.javatpoint.com/keywords-in-c) can be defined as the **pre-defined** or the **reserved words** having its own importance, and each keyword has its own functionality. Since keywords are the pre-defined words used by the compiler, so they cannot be used as the variable names. If the keywords are used as the variable names, it means that we are assigning a different meaning to the keyword, which is not allowed. C language supports 32 keywords given below:

|  |  |  |  |
| --- | --- | --- | --- |
| **auto** | **double** | **int** | **struct** |
| **break** | **else** | **long** | **switch** |
| **case** | **enum** | **register** | **typedef** |
| **char** | **extern** | **return** | **union** |
| **const** | **float** | **short** | **unsigned** |
| **continue** | **for** | **signed** | **void** |
| **default** | **goto** | **sizeof** | **volatile** |
| **do** | **if** | **static** | **while** |

**Identifiers in C**

[Identifiers in C](https://www.javatpoint.com/c-identifiers) are used for naming variables, functions, arrays, structures, etc. Identifiers in C are the user-defined words. It can be composed of uppercase letters, lowercase letters, underscore, or digits, but the starting letter should be either an underscore or an alphabet. Identifiers cannot be used as keywords. Rules for constructing identifiers in C are given below:

* The first character of an identifier should be either an alphabet or an underscore, and then it can be followed by any of the character, digit, or underscore.
* It should not begin with any numerical digit.
* In identifiers, both uppercase and lowercase letters are distinct. Therefore, we can say that identifiers are case sensitive.
* Commas or blank spaces cannot be specified within an identifier.
* Keywords cannot be represented as an identifier.
* The length of the identifiers should not be more than 31 characters.
* Identifiers should be written in such a way that it is meaningful, short, and easy to read.

**Strings in C**

[Strings in C](https://www.javatpoint.com/c-strings) are always represented as an array of characters having null character '\0' at the end of the string. This null character denotes the end of the string. Strings in C are enclosed within double quotes, while characters are enclosed within single characters. The size of a string is a number of characters that the string contains.

Now, we describe the strings in different ways:

char a[10] = "javatpoint"; // The compiler allocates the 10 bytes to the 'a' array.

char a[] = "javatpoint"; // The compiler allocates the memory at the run time.

char a[10] = {'j','a','v','a','t','p','o','i','n','t','\0'}; // String is represented in the form of characters.

**Operators in C**

[Operators in C](https://www.javatpoint.com/c-operators) is a special symbol used to perform the functions. The data items on which the operators are applied are known as operands. Operators are applied between the operands. Depending on the number of operands, operators are classified as follows:

**Unary Operator**

A unary operator is an operator applied to the single operand. For example: increment operator (++), decrement operator (--), sizeof, (type)\*.

**Binary Operator**

The binary operator is an operator applied between two operands. The following is the list of the binary operators:

* Arithmetic Operators
* Relational Operators
* Shift Operators
* Logical Operators
* Bitwise Operators
* Conditional Operators
* Assignment Operator
* Misc Operator

**Constants in C**

A constant is a value assigned to the variable which will remain the same throughout the program, i.e., the constant value cannot be changed.

There are two ways of declaring constant:

* Using const keyword
* Using #define pre-processor

**Types of**[**constants in C**](https://www.javatpoint.com/constants-in-c)

|  |  |
| --- | --- |
| **Constant** | **Example** |
| **Integer constant** | **10, 11, 34, etc.** |
| **Floating-point constant** | **45.6, 67.8, 11.2, etc.** |
| **Octal constant** | **011, 088, 022, etc.** |
| **Hexadecimal constant** | **0x1a, 0x4b, 0x6b, etc.** |
| **Character constant** | **'a', 'b', 'c', etc.** |
| **String constant** | **"java", "c++", ".net", etc.** |

**Special characters in C**

Some special characters are used in C, and they have a special meaning which cannot be used for another purpose.

* **Square brackets [ ]:** The opening and closing brackets represent the single and multidimensional subscripts.
* **Simple brackets ( ):** It is used in function declaration and function calling. For example, printf() is a pre-defined function.
* **Curly braces { }:** It is used in the opening and closing of the code. It is used in the opening and closing of the loops.
* **Comma (,):** It is used for separating for more than one statement and for example, separating function parameters in a function call, separating the variable when printing the value of more than one variable using a single printf statement.
* **Hash/pre-processor (#):** It is used for pre-processor directive. It basically denotes that we are using the header file.
* **Asterisk (\*):** This symbol is used to represent pointers and also used as an operator for multiplication.
* **Tilde (~):** It is used as a destructor to free memory.
* **Period (.):** It is used to access a member of a structure or a union.