Learn CODE: Start with C

Article is written by –

Sorowar Mahabub, Studies B.Sc in CSE,

Int' Islamic University Chittagong (IIUC)

Cell: 01834756433, 01521564157(*whasApp available*)

Email : sorowarmahabub1709vip@gmail.com

Article Reference : Google, DataFlair_blog

Tokens in C

(An Awesome Concept you can't Afford to Miss Out)

As we can't construct a sentence without the use of words, similarly, we can't construct a program without using building blocks. The smallest individual element in a program is a token. Without tokens, programming can't be done in C. Generally, tokens in C is the basic component of creating source code.

Here, you would be well acquainted with the –

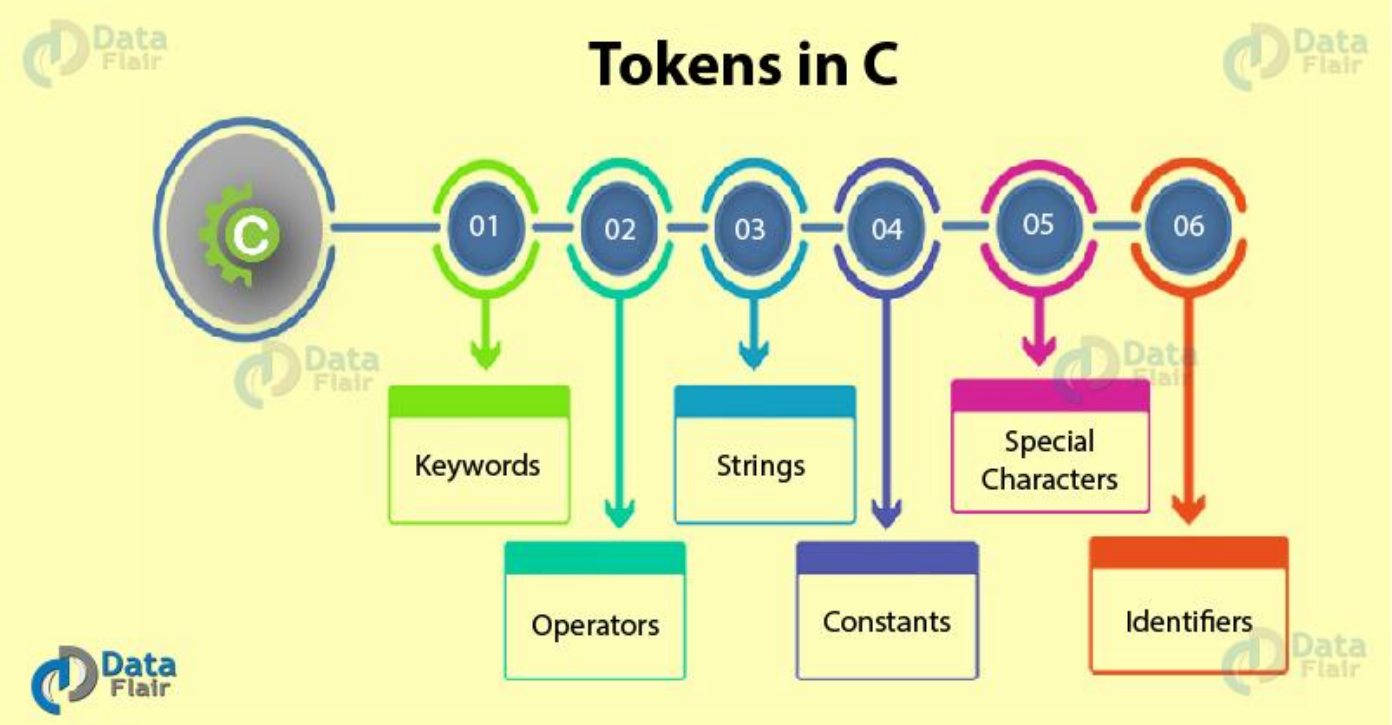
1. Keywords in C
2. Identifiers in C
3. Strings in C
4. Operators in C
5. Constant in C
6. Special Characters in C

What are Tokens in C?

Tokens in C language are the smallest possible unit of a program, that conveys a specific meaning to the compiler. It is the building blocks of a programming language.

Different Types of Tokens in C

There are 6 types of Tokens in C programming languages-



(This Picture Collected from DataFlair)

It is important to understand that these names can't be used interchangeably, hence, we will discuss each type of tokens in C in detail.

1. Keywords in C

Keywords in C language are the pre-defined & reserved words, each having its own significance and hence has a specific function associated with it. We can't simply use keywords for assigning variable names, as it would connote a totally different meaning altogether and would be erroneous. There are a total of 32 keywords offered in C.

Auto	Break	case	char
Continue	Do	default	const
Double	Else	enum	extern
For	If	goto	float
Int	Long	register	return
Signed	Static	sizeof	short
Struct	Switch	typedef	union
Void	While	volatile	unsigned

Key takeaway: All the keywords in C are lowercase in nature.

Apart from the above-listed keywords, there are certain supplementary words that cannot be used as variable names. They are listed in alphabetical order as follows:

1. asm
2. bool
3. catch
4. class
5. const_cast
6. delete
7. dynamic_cast
8. explicit
9. export
10. false
11. friend
12. inline
13. mutable
14. namespace
15. new
16. operator
17. private
18. protected
19. public
20. reinterpret_cast
21. static_cast
22. template
23. this
24. throw
25. true
26. try
27. typeid
28. typename
29. using
30. virtual
31. wchar_t

2. Identifiers in C

The C programmer has the provision to give names of his own choice to variables, arrays, and functions. These are called identifiers in C. The user may use the combination of different character sets available in the C language to name an identifier but, there are certain rules to be abided by the user on his part when naming identifiers, otherwise the situation would prove to be dicey.

Rules for Identifiers in C –

1. **First character:** The first character of the identifier should necessarily begin with either an alphabet or an underscore. It cannot begin with a digit.
2. **No special characters:** C does not support the use of special characters while naming an identifier. For instance, special characters like comma or punctuation marks can’t be used.
3. **No keywords:** The use of keywords as identifiers is strictly prohibited, as they are reserved words which we have already discussed.
4. **No white space:** White spaces include blank spaces, newline, carriage return, and horizontal tab, which can’t be used.
5. **Word limit:** The identifier name can have an arbitrarily long sequence that should not exceed 31 characters, otherwise, it would be insignificant.
6. **Case sensitive:** Uppercase and lowercase characters are treated differently.

Here is a table which illustrates the valid use of Identifiers in C:

Identifiers Name	Valid or Invalid	Correction or Alternative, If Invalid	Elucidation, If Invalid
20th_name	Invalid	name_20	It violates Rule 1 as it begins with a digit
_age	Valid	–	–
market.bill	Invalid	market_bill	It violates Rule 2 as it contains a special character '.'
delete[5]	Invalid	delet[5]	It violates Rule 3 as it contains a keyword
employee[10]	Valid	–	–
customer name	Invalid	Customername	It violates Rule 4 as it contains a blank space
area()	Valid	–	–

3. Constants in C

Often referred to as literals, constants, as the name itself suggests, are fixed values i.e. they cannot change their value during program run once they are defined.

Syntax of Constant in C Programming-

const data_type variable_name = value;

Various Types of Constants in C Language-

1. **Integer constants:** These are of the integer data type. For example, *const int value = 400;*
2. **Floating constants:** These are of the float data type. For example, *const float pi = 3.14;*
3. **Character constants:** These are of the character data type. For example, *const char gender = 'f';*
4. **String constants:** These are also of the character data type, but differ in the declaration. For example, *const char name[] = ''DataFlair'';*
5. **Octal constants:** The number system which consists only 8 digits, from 0 to 7 is called the octal number system. The constant octal values can be declared as, *const int oct = 040;* (It is the octal equivalent of the digit "32" in the decimal number system.)
6. **Hexadecimal constants:** The number system which consists of 16 digits, from 0 to 9 and alphabets 'a' to 'f' is called hexadecimal number system. The constant

hexadecimal values can be declared as, *const int hex = 0x40;* (It is the hexadecimal equivalent of the digit 64 in the decimal number system.)

4. Strings in C

Just like characters, strings are used to store letters and digits. *Strings in C are referred to as an array of characters. It is enclosed within double quotes, unlike characters which are stored within single quotes.* The termination of a string is represented by the null character that is '**\0**'. The size of a string is the number of individual characters it has. In C, a string can be declared in the following ways:

```
1. char name[30] = ''LearnCODE"; // The compiler reserves 30 bytes of memory
1. char name[] = "LearnCODE"; // The compiler reserves the required amount of memory
1. char name[30] = { 'L' , 'e' , 'a' , 'r' , 'n' , 'C' , 'O' , 'D' , 'E' , '\0' }; // How a string is represented as a set of characters.
```

5. Special Symbols of C

Apart from letters and digits, there are some special characters in C, which will help you to manipulate or perform data operations. Each special symbol has a specific meaning to the C compiler.

1. **[ ]** – ***Square brackets*** – The opening and closing brackets of an array indicate single and multidimensional subscripts.
2. **()** – ***Simple brackets*** – Used to represent function declaration and calls, used in print statements.
3. **{ }** – ***Curly braces*** – Denote the start and end of a particular fragment of code which may be functions or loops or conditional statements.
4. **,** – ***Comma*** – Separate more than one statements, like in the declaration of different variable names in C.
5. **#** – ***Hash / Pound / Preprocessor*** – A preprocessor directive, utilize for denoting the use of a header file.
6. ***** – ***Asterisk*** – To declare pointers, used as an operand for multiplication.
7. **~** – ***Tilde*** – As a destructor to free memory.
8. **.** – ***Period/dot*** – To access a member of a structure.

6. Operators in C

Operators in C are tools or symbols, which are used to perform a specific operation on data. Operations are performed on operands. Operators can be classified into three broad categories, according to the number of operands used. Which are as follows:

1. Unary: It involves the use of one a single operand. For instance, '!' is a unary operator which operates on a single variable, say 'c' as !c which denotes its negation or complement.

2. Binary: It involves the use of 2 operands. They are further classified as:

- Arithmetic
- Logical
- Bitwise
- Relational
- Assignment
- Conditional

3. Ternary: It involves the use of 3 operands. For instance, **?**: Is used in place of if-else conditions.

Summary

Tokens in C are the basic building blocks of a program. A person who has mastered these concepts is a valuable entity in the market. Different types of tokens in C allow us to have numerous functionalities to various fields. In short, we can say that every C aspirant should know the concept of tokens in the C programming language.

Thanks...!