**What is a Character set?**

Like every other language, 'C' also has its own character set. A program is a set of instructions that, when executed, generate an output. The data that is processed by a program consists of various characters and symbols. The output generated is also a combination of characters and symbols.

A character set in 'C' is divided into,

* Letters
* Numbers
* Special characters
* White spaces (blank spaces)

A compiler always ignores the use of characters, but it is widely used for formatting the data. Following is the character set in 'C' programming:

1. Letters
	* Uppercase characters (A-Z)
	* Lowercase characters (a-z)
2. Numbers
	* All the digits from 0 to 9
3. White spaces
	* Blank space
	* New line
	* Carriage return
	* Horizontal tab
4. Special characters
	* Special characters in 'C' are shown in the given table,

|  |  |
| --- | --- |
| , (comma) | { (opening curly bracket) |
| . (period) | } (closing curly bracket) |
| ; (semi-colon) | [ (left bracket) |
| : (colon) | ] (right bracket) |
| ? (question mark) | ( (opening left parenthesis) |
| ' (apostrophe) | ) (closing right parenthesis) |
| " (double quotation mark) | & (ampersand) |
| ! (exclamation mark) | ^ (caret) |
| |(vertical bar) | + (addition) |
| / (forward slash) | - (subtraction) |
| \ (backward slash) | \* (multiplication) |
| ~ (tilde) | / (division) |
| \_ (underscore) | > (greater than or closing angle bracket) |
| $ (dollar sign) | < (less than or opening angle bracket) |
| % (percentage sign) | # (hash sign) |

**What is Token in C?**

**TOKEN** is the smallest unit in a 'C' program. It is each and every word and punctuation that you come across in your C program. The compiler breaks a program into the smallest possible units (tokens) and proceeds to the various stages of the compilation.

In a C program ,tokens are building blocks. A token is divided into six different types, viz, **Keywords, Operators, Strings, Constants, Special Characters, and Identifiers.**

## Types of tokens Tokens used

## 1.Keywords int,void

## 2.Identifier x,main,printf

## 3.Constant 1,2,Incremented value:,Enter a number

## 4.Operators Addressof(&),Addition(+),Multiplication(\*)

## 5.Special Symbols (,),{,},%,&,:

## 1.KEYWORDS

In 'C' every word can be either a keyword or an identifier.

Keywords have fixed meanings, and the meaning cannot be changed. They act as a building block of a 'C' program. There are a total of 32 keywords in 'C'. Keywords are written in lowercase letters.

Following table represents the keywords in 'C'-

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

**2.IDENTIFIER**

An identifier is nothing but a name assigned to an element in a program. Example, name of a variable, function, etc. Identifiers are the user-defined names consisting of 'C' standard character set. As the name says, identifiers are used to identify a particular element in a program. Each identifier must have a unique name.

Following rules must be followed for identifiers:

1. The first character must always be an alphabet or an underscore.
2. It should be formed using only letters, numbers, or underscore.
3. A keyword cannot be used as an identifier.
4. It should not contain any whitespace character.
5. The name must be meaningful.

Thus just as persons,cities or streets have names ,the C entities such as variables,functions,files etc are given unique names(identifiers) for their identification in a C program.

### Following are some examples of identifiers:

**IDENTIFIER VALID? REMARK**

Sum valid

 Char invalid keywords are not allowed

Price# invalid special symbols are not allowed

Var 1 invalid blank space not allowed

avg\_num valid

## 3.Constants

Constants are the fixed values that never change during the execution of a program. Following are the various types of constants:

### Integer constants

An integer constant is nothing but a value consisting of digits or numbers. These values never change during the execution of a program. Integer constants can be octal, decimal and hexadecimal.

1. Decimal constant contains digits from 0-9 such as,

Example, 111, 1234

Above are the valid decimal constants.

1. Octal constant contains digits from 0-7, and these types of constants are always preceded by 0.

Example, 012, 065

Above are the valid decimal constants.

1. Hexadecimal constant contains a digit from 0-9 as well as characters from A-F. Hexadecimal constants are always preceded by 0X.

Example, 0X2, 0Xbcd

Above are the valid hexadecimal constants.

The octal and hexadecimal integer constants are very rarely used in programming with 'C'.

### Character constants

A character constant contains only a single character enclosed within a single quote (''). We can also represent character constant by providing ASCII value of it.

Example, 'A', '9'

Above are the examples of valid character constants.

### String constants

A string constant contains a sequence of characters enclosed within double quotes ("").

Example, "Hello", "Programming"

These are the examples of valid string constants.

### Real Constants

Like integer constants that always contains an integer value. 'C' also provides real constants that contain a decimal point or a fraction value. The real constants are also called as floating point constants. The real constant contains a decimal point and a fractional value.

Example, 202.15, 300.00

These are the valid real constants in 'C'.

A real constant can also be written as,

Mantissa e Exponent

For example, to declare a value that does not change like the classic circle constant PI, there are two ways to declare this constant

1. By using the **const** keyword in a variable declaration which will reserve a storage memory

 #include <stdio.h>

int main() {

const double PI = 3.14;

printf("%f", PI);

//PI++; // This will generate an error as constants cannot be changed

return 0;}

1. By using the **#define** pre-processor directive which doesn't use memory for storage and without putting a semicolon character at the end of that statement

#include <stdio.h>

#define PI 3.14

int main() {

printf("%f", PI);

return 0;}

### 4. 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**](https://data-flair.training/blogs/variables-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.

### Summary

* A token is the smallest unit in a program.
* A keyword is reserved words by language.
* There are total of 32 keywords.
* An identifier is used to identify elements of a program.
* A constant is a value that doesn't change throughout the execution of a program.
* A variable is an identifier which is used to store a value.
* There are four commonly used data types such as int, float, char and a void.
* Each data type differs in size and range from one another.