**C character set**

C character set defines set of valid characters used to write a C program. The set consists of alphabets (both upper case and lower case), digits (0-9), various special characters symbols and white space characters.

* Alphabets: A-Z a-z
* Digits: 0-9
* Special character: ` ~ ! @ # $ % ^ & \* ( ) \_ – + = { } [ ] | \ : ; “ ‘ < , > . ? /
* White space characters: Blank space, new line, horizontal tab, vertical tab, formfeed and carriage return.

**Tokens in C programming**

*Tokens* are the smallest individual unit of a program. They are the building blocks of a program. For example, lets us consider an English paragraph:

Trending

[**Classification of programming languages**](https://codeforwin.org/fundamentals/programming-languages-classification)

*“I love programming in C. I love Codeforwin.”*

The above sentence is made with various basic elements such as – Alphabets (a-z, A-Z), punctuation marks and blank spaces. These are tokens of the given paragraph.  
Similarly, all basic elements of a program collectively known as Tokens. There are five types of tokens.

1. [Keywords](https://codeforwin.org/c-programming/keywords-identifiers-c#keywords)
2. [Identifiers](https://codeforwin.org/c-programming/keywords-identifiers-c#identifiers)
3. [Operators](https://codeforwin.org/2017/08/operators-separators-c-programming.html#operators)
4. [Separators](https://codeforwin.org/2017/08/operators-separators-c-programming.html#separators)
5. [Literals](https://codeforwin.org/2017/08/literals-c-programming.html)

Consider the below C statement –  
int num = 5 + 4;

Here,

* int is keyword.
* *num* is identifier.
* = and + are operators.
* 5 and 4 are literals.
* ; and white spaces are separators.

**What are Keywords?**

*Keywords* are the reserved words whose meaning is predefined by the programming language specification. They convey some special meaning in programming and we must not use them for other purposes.

For example – int is a C keyword that defines integer data type. You must not use int for other purposes.

There are total 32 keywords in C as per ANSI standards.

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

**Note:** Apart from the 32 standard C keyword, different compilers can have more number of keywords.

**What are Identifiers?**

*Identifiers* are the names given to various programming elements. Such as name given to a variable, function, user defined type etc.

In real life, you may think name given to a person or object as an identifier. In programming, we use identifiers to identify a programming element uniquely.

**Rules for naming an identifier**

1. Identifier must not be a keyword. For example – you cannot use int, return etc. as an identifier.
2. Identifiers must begin with an alphabet a-z A-Z or underscore \_ symbol.
3. Identifiers can contain digits 0-9.
4. Identifiers must not contain special character other than alphabets, digits and underscore symbol.
5. C is a case sensitive language hence you must be careful while naming an identifier. For example – *num*, *NUM*, *Num* all are different.

**Examples of some valid identifiers**

*num* *\_num* *num1* *\_num1* *\_1num1* *NUM* *\_NUM\_*

**Examples of some invalid identifiers**

*int* *int* is a keyword and must not be used as an identifier.  
*1num* Identifier should begin with an alphabet or underscore symbol.  
*num$* It must not contain any special symbol $ in this case.  
*first num* It must not contain spaces.  
*date-of-birth* It must not contain any special symbol – in this case.

**Note:** We must give a meaningful name to an identifier. However, it is not a naming convention but considered as a good programming practice. A meaningful identifier increases program readability and clarity.

*In C, every word is either a keyword or an identifier.*