C tokens - keywords, identifiers, constants, operators, special symbols, and strings

By

Prof. Muhammad Iqbal Bhat

Government Degree College Beerwah

Topics

C tokens

- keywords,
- identifiers,
- constants,
- operators,
- special symbols,
- strings

C tokens:

Tokens are the basic building blocks of a C program.

A token is a sequence of characters that represents a specific element in the program, such as a keyword, variable, operator, or punctuation symbol.

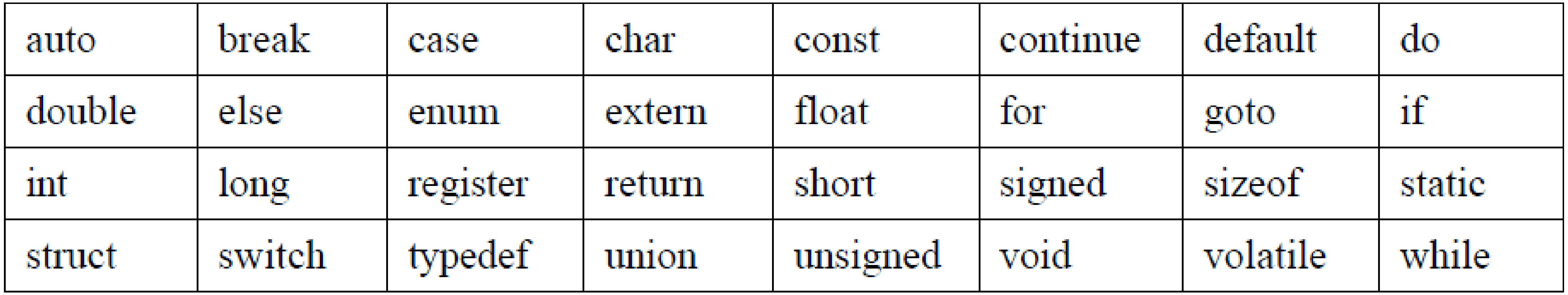
C language has six types of tokens: keywords, identifiers, constants, operators, special symbols, and strings.

Tokens are used to build the structure of a C program and to specify the actions that the program should take.

Keywords:

- In C programming, keywords are a set of reserved words that have a specific meaning and are used to build the structure of the language.
- Examples of keywords in C include: "if," "else," "for," "while," "return," and "void".
- Keywords cannot be used as variable names or identifiers, and it is important to avoid using them as such to prevent errors in your program.
- Using a keyword as a variable name will result in a compilation error because the compiler will confuse the keyword with a reserved word.

List of Keywords:



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

List of Keywords in C:

Keyword	Description
auto	Specifies automatic storage duration for a variable.
break	Terminates a loop or switch statement.
case	Defines a case in a switch statement.
char	Declares a character variable or data type.
const	Specifies that a variable's value cannot be changed.
continue	Skips the remaining statements in a loop.
default	Defines the default case in a switch statement.
do	Starts a do-while loop.
double	Declares a double-precision floating-point variable.
else	Specifies an alternative statement in an if-else block.
enum	Declares an enumeration type.
extern	Specifies that a variable is declared outside the program's scope.
float	Declares a floating-point variable or data type.
for	Starts a for loop.
goto	Transfers control to a labeled statement.
if	Starts an if statement.

Keyword	Description
int	Declares an integer variable or data type.
long	Declares a long integer variable or data type.
register	Specifies register storage class for a variable.
return	Returns a value from a function.
short	Declares a short integer variable or data type.
signed	Declares a signed variable or data type.
sizeof	Determines the size of a variable or data type.
static	Specifies static storage duration for a variable.
struct	Defines a structure type.
switch	Starts a switch statement.
typedef	Defines a new data type.
union	Defines a union type.
unsigned	Declares an unsigned variable or data type.
void	Declares a function returning no value or a pointer without a type.
volatile	Specifies that a variable may change unexpectedly.
while	Starts a while loop.

Identifiers:

- Identifiers are names given to variables, functions, arrays, and other user-defined items in a C program.
- An identifier in C must start with a letter (a to z or A to Z) or underscore (_), and it can be followed by any combination of letters, digits (0 to 9), and underscores.
- C language is case-sensitive, so upper and lowercase letters are treated as distinct characters in identifiers.
- Identifiers in C cannot be the same as C keywords, such as if, else, for, while, etc.

Rules for creating Identifiers in C:

- The first character must be an alphabet (either uppercase or lowercase) or an underscore (_).
- The identifier can be a combination of letters, digits, or underscore characters.
- The identifier must not be a keyword in C.
- The identifier must not contain spaces or special characters (such as @, #, $, %, ^, etc.).
- The identifier must be unique within the scope of the program.

Examples of Identifiers:

Identifier	**Correct/Incorrect**	**Remarks**
age	Correct	Starts with a letter, contains only letters
_count	Correct	Starts with an underscore, contains letters and _
myArray	Correct	Starts with a letter, contains letters and uppercase
num_1	Correct	Contains letters, underscore and digit
2var	Incorrect	Starts with a digit
my-name	Incorrect	Contains a hyphen
if	Incorrect	Same as C keyword (reserved word)
my array	Incorrect	Contains a space
my@var	Incorrect	Contains a special character

Variables:

- Variables are used to store data in a C program.
- A variable is a named location in the computer's memory that can hold a value of a specific data type.
- Variables are declared using a data type and a name, and they can be assigned a value and used in calculations or other operations throughout the program.
- C language supports several data types for variables, including integer, character, floating-point, and pointer types.
- Variables are an essential part of C programming, as they allow programs to store and manipulate data dynamically.

Naming Variables:

Must be a valid identifier.

Must not be a keyword

Names are case sensitive.

Variables are identified by only first 32 characters.

Library commonly uses names beginning with _

Examples

Naming conventions for variables in C language:

- Use meaningful names: Variable names should be descriptive and convey the purpose of the variable. Avoid using vague or generic names such as "x" or "temp".
- Use camel case: In camel case, the first letter of each word is capitalized except for the first word. For example, "firstName" or "accountBalance".
- Use underscores: Alternatively, some developers prefer to use underscores to separate words in variable names. For example, "first_name" or "account_balance".
- Use lowercase letters: Variable names should be written in lowercase letters to differentiate them from constants, which are typically written in uppercase.
- Be consistent: It is important to be consistent in variable naming conventions throughout a program or project. This helps ensure that the code is easy to read and maintain.
- Use plural for arrays: When naming arrays, it is common to use a plural form of the name. For example, "students" instead of "student".
- Use meaningful prefixes: In some cases, it can be helpful to use a prefix to indicate the type of variable or its scope. For example, "g_" for global variables or "p_" for pointers.

Data types in C language:

Data types are an important concept in C programming as they define the type of data that a variable can hold and the operations that can be performed on that data.

C language has a variety of data types, including

1. Basic data types (such as integers, floating-point numbers, and characters),

2. Derived data types (such as arrays and pointers)

3. User-defined data types (such as structures and unions).

Basic Data types:

Basic data types in C language are used to represent fundamental types of data that are used in programming.

C language provides four basic integer data types: char, short, int, and long.

Range of Data types:

Data Type	Size (in bytes)	Range
char	1	-128 to 127 or 0 to 255 (if unsigned)
short	2	-32,768 to 32,767
int	2 or 4	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
long	4 or 8	-2,147,483,648 to 2,147,483,647 or -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4	3.4e-38 to 3.4e+38
double	8	1.7e-308 to 1.7e+308
long double	12 or 16	3.4e-4932 to 1.1e+4932
void	0	No range (represents the absence of a value)

Examples of Basic Data types:

Data Type	Examples
char	char ch = 'a';
short	short s = 12345;
int	int i = 123;
long	long l = 1234567890L;
float	float f = 3.14f;
double	double d = 3.14159;
void	void function();

```c
/*
      * Program to demonstrate the use of basic data types in C language
*/
#include <stdio.h>

int main() {
    // Declaring and initializing variables of various data types
    char c = 'A';
    short s = 10;
    int i = 100;
    long l = 10000L;
    float f = 3.14f;
    double d = 3.14159;

    // Printing the values of the variables to the console
    printf("Value of char: %c\n", c);
    printf("Value of short: %hd\n", s);
    printf("Value of int: %d\n", i);
    printf("Value of long: %ld\n", l);
    printf("Value of float: %f\n", f);
    printf("Value of double: %lf\n", d);

    return 0;
}
```

Constants or Literals

Constants or Literals

- Constants refer to fixed values that the program may not alter during its execution.
- These fixed values are also called literals.
- In C, there are different types of constants, including integer, floating-point, character, and string constants.

Integer Literals:

Integer constants are whole numbers without decimal points, and can be expressed in decimal (base 10), octal (base 8), or hexadecimal (base 16) formats.

A prefix specifies the base or radix: 0x or 0X for hexadecimal, 0 for octal, and nothing for decimal.

An integer literal can also have a suffix that is a combination of U and L, for unsigned and long, respectively. The suffix can be uppercase or lowercase and can be in any order.

Integer Literals (Examples)

- `212                /* Legal */`
- `215u               /* Legal */`
- `0xFeeL             /* Legal */`
- `078                /* Illegal: 8 is not an octal digit */`
- `032UU              /* Illegal: cannot repeat a suffix */`
- `85                 /* decimal */`
- `0213               /* octal */`
- `0x4b               /* hexadecimal */`
- `30                 /* int */`
- `30u                /* unsigned int */`
- `30l                /* long */`
- `30ul               /* unsigned long */`

Floating Point Literals:

A floating-point literal has an integer part, a decimal point, a fractional part, and an exponent part.

You can represent floating point literals either in decimal form or exponential form.

While representing decimal form, you must include the decimal point, the exponent, or both; and while representing exponential form, you must include the integer part, the fractional part, or both.

The signed exponent is introduced by e or E.

Floating Point Literals (Examples)

• Standard notation:	`float pi = 3.14159;`
• Exponential notation:	`float x = 1.23e-4; (which means 1.23 x 10^-4)`
• `3.14159`	`/* Legal */`
• `314159E-5L`	`/* Legal */`
• `510E`	`/* Illegal: incomplete exponent */`
• `210f`	`/* Illegal: no decimal or exponent */`
• `.e55`	`/* Illegal: missing integer or fraction */`

Character Literals:

Character literals are enclosed in single quotes, e.g., 'x' can be stored in a simple variable of char type.

A character literal can be a plain character (e.g., 'x'), an escape sequence (e.g., '\t'), or a universal character (e.g., '\u02C0’).

There are certain characters in C that represent special meaning when preceded by a backslash, for example, newline (\n) or tab (\t).

Character Literals (Examples):

Escape sequence	Meaning
\\	\ character
\'	' character
\"	" character
\?	? character
\a	Alert or bell
\b	Backspace
\f	Form feed
\n	Newline
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\ooo	Octal number of one to three digits
\xhh . . .	Hexadecimal number of one or more digits

String Literals:

- String literals or constants are enclosed in double quotes "". A string contains characters that are similar to character literals: plain characters, escape sequences, and universal characters.
- You can break a long line into multiple lines using string literals and separating them using whitespaces.

```
char str[] = "Hello, world!";
"hello, dear"
"hello, \
dear"
"hello, " "d" "ear"
```

How to Define Constants:

Preprocessor	const
Using #define preprocessor	Using const keyword

The #define Preprocessor

- The #define directive defines a symbolic constant that can be used throughout the program.
- The syntax for #define is `#define constant_name value`, where constant_name is the name of the constant, and value is the value of the constant.

```
  #define LENGTH 10
  #define WIDTH 5
  #define NEWLINE '\n'
```

The const Keyword

- Another way to define constants in C is by using the const keyword.
- The syntax for const is `const data_type constant_name = value;`
- It is a good programming practice to define constants in CAPITALS

```
const int MAX = 100;
const int LENGTH = 10;
const int WIDTH = 5;
const char NEWLINE = '\n';
```

Questions?