Keywords in C

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 total 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

Identifiers in C

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

What is a Variable?

A variable is an identifier which is used to store some value. Constants can never change at the time of execution. Variables can change during the execution of a program and update the value stored inside it.

A single variable can be used at multiple locations in a program. A variable name must be meaningful. It should represent the purpose of the variable.

```
Example: Height, age, are the meaningful variables that represent the purpose
it is being used for. Height variable can be used to store a height value.
Age variable can be used to store the age of a person
```

A variable must be declared first before it is used somewhere inside the program. A variable name is formed using characters, digits and an underscore.

Following are the rules that must be followed while creating a variable:

1. A variable name should consist of only characters, digits and an underscore.
2. A variable name should not begin with a number.
3. A variable name should not consist of whitespace.
4. A variable name should not consist of a keyword.
5. 'C' is a case sensitive language that means a variable named 'age' and 'AGE' are different.

Following are the examples of valid variable names in a 'C' program:

```
height or HEIGHT
_height
_height1
My_name
```

Following are the examples of invalid variable names in a 'C' program:

```
1height
Hei$ght
My name
```

For example, we declare an integer variable **my_variable** and assign it the value 48:

```
int my_variable;
my_variable = 48;
```

By the way, we can both declare and initialize (assign an initial value) a variable in a single statement:

```
int my_variable = 48;
```

Data types in C

A data-type in C programming is a set of values and is determined to act on those values. C provides various types of data-types which allow the programmer to select the appropriate type for the variable to set its value. ANSI C provides three types of data types:

1. Primitive data types

2. Derived data types
3. User-defined data types

Primitive data types

Every C compiler supports five primary data types:

void	As the name suggests, it holds no value and is generally used for specifying the type of function or what it returns. If the function has a void type, it means that the function will not return any value.
int	Used to denote an integer type.
char	Used to denote a character type.
float, double	Used to denote a floating point type.
int *, float *, char *	Used to denote a pointer type.

Data type	Size in bytes	Range
Char or signed char	1	-128 to 127
Unsigned char	1	0 to 255
int or signed int	2	-32768 to 32767
Unsigned int	2	0 to 65535
Short int or Unsigned short int	2	0 to 255
Signed short int	2	-128 to 127
Long int or Signed long int	4	-2147483648 to 2147483647
Unsigned long int	4	0 to 4294967295
float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
Long double	10	3.4E-4932 to 1.1E+4932

Integer data type

Integer is nothing but a whole number. The range for an integer data type varies from machine to machine. The standard range for an integer data type is -32768 to 32767.

An integer typically is of 2 bytes which means it consumes a total of 16 bits in memory. A single integer value takes 2 bytes of memory. An integer data type is further divided into other data types such as short int, int, and long int.

Each data type differs in range even though it belongs to the integer data type family. The size may not change for each data type of integer family.

The short int is mostly used for storing small numbers, int is used for storing averagely sized integer values, and long int is used for storing large integer values.

Whenever we want to use an integer data type, we have place int before the identifier such as,

```
int age;
```

Here, age is a variable of an integer data type which can be used to store integer values.

Floating point data type

Like integers, in 'C' program we can also make use of floating point data types. The 'float' keyword is used to represent the floating point data type. It can hold a floating point value which means a number is having a fraction and a decimal part. A floating point value is a real number that contains a decimal point. Integer data type doesn't store the decimal part hence we can use floats to store decimal part of a value.

Generally, a float can hold up to 6 precision values. If the float is not sufficient, then we can make use of other data types that can hold large floating point values. The data type double and long double are used to store real numbers with precision up to 14 and 80 bits respectively.

While using a floating point number a keyword float/double/long double must be placed before an identifier. The valid examples are,

```
float division;
double BankBalance;
```

Character data type

Character data types are used to store a single character value enclosed in single quotes.

A character data type takes up-to 1 byte of memory space.

Example,

```
Char letter;
```

Void data type

A void data type doesn't contain or return any value. It is mostly used for defining functions in 'C'.

Example,

```
void displayData()
```

Type declaration of a variable

```
int main() {
int x, y;
float salary = 13.48;
char letter = 'K';
x = 25;
y = 34;
int z = x+y;
printf("%d \n", z);
printf("%f \n", salary);
printf("%c \n", letter);
return 0;}
```

Output:

```
59
13.480000
K
```

We can declare multiple variables with the same data type on a single line by separating them with a comma. Also, notice the use of format specifiers in **printf** output function float (%f) and char (%c) and int (%d).

Derived data types

C supports three derived data types:

Data Types	Description
Arrays	Arrays are sequences of data items having homogeneous values. They have adjacent memory locations to store values.
References	Function pointers allow referencing functions with a particular signature.
Pointers	These are powerful C features which are used to access the memory and deal with their addresses.

User defined data types

C allows the feature called *type definition* which allows programmers to define their identifier that would represent an existing data type. There are three such types:

Data Types	Description
Structure	It is a package of variables of different types under a single name. This is done to handle data efficiently. "struct" keyword is used to define a structure.
Union	These allow storing various data types in the same memory location. Programmers can define a union with different members, but only a single member can contain a value at a given time. It is used for

Enum	Enumeration is a special data type that consists of integral constants, and each of them is assigned with a specific name. "enum" keyword is used to define the enumerated data type.

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.

2. 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.

3. 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;}
```

2. 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;}
```