C Language Tutorials

C Language Tutorials by GM dahar

while(true){}

by
GM DAHAR

www.fb.com/gmdahar

C Language by GM Dahar

contact link (follow me)

C Introduction

- C Keywords and Identifier
- C Variables and Constants
- C Programming Data Types
- C Programming input/Output
- C Programming Operators
- C Precedence and Associativity
- C Introduction Practice Programs

C Decisions And Loops

- C Programming if...else
- C Programming for Loops
- C do...while Loops
- C break and continue
- C Programming switch...case
- C Programming goto
- C Decision & Loop Examples

C Programming Functions

- C Functions Introduction
- C User-defined Functions
- C Function Types
- C Programming recursion
- C Storage Class
- C Function Examples

C Programming Arrays

- C Arrays Introduction
- C Multi-dimensional Arrays
- C Arrays & Functions
- C Arrays Examples

C Programming Pointers

- C Pointers Introduction
- C Pointers And Arrays
- C Pointers And Functions
- C Dynamic Memory Allocation
- C Pointer Examples

C Programming Strings

- C Programming Strings
- C String Functions
- C String Examples

C Structure And Unions

- C Structure Introduction
- C Structures and Pointers
- C Structure and Function
- C Programming Unions
- C Structure Examples

C Programming Files

- C Files Input/Output
- C Files Examples

More On C Programming

- C Programming Enumeration
- C Programming Preprocessors
- C Library Functions
- C Programming Examples

C Programming Keywords and Identifiers

Character set

Character set are the set of alphabets, letters and some special characters that are valid in C language.

Alphabets:

Uppercase: A B C ..... X Y Z

Lowercase: a b c ..... x y z

Digits:

0 1 2 3 4 5 6 8 9

Special Characters:

Special Characters in C language

,	<	>	.	_	(	)	;	$	:	%	[	]	#	?
'	&	{	}	"	^	!	*	/	|	-	\	~	+	

White space Characters:

blank space, new line, horizontal tab, carriage return and form feed

Keywords:

Keywords are the reserved words used in programming. Each keywords has fixed meaning and that cannot be changed by user. For example:

```
int money;
```

Here, int is a keyword that indicates, 'money' is of type integer.

As, C programming is case sensitive, all keywords must be written in lowercase.

Keywords in C Language

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while

default	goto	sizeof	volatile
const	float	short	unsigned

Besides these keywords, there are some additional keywords supported by Turbo C.

Additional Keywords for Borland C						
asm	far	interrupt	pascal	near	huge	cdecl

All these keywords, their syntax and application will be discussed in their respective topics.

Identifiers

In C programming, identifiers are names given to C entities, such as variables, functions, structures etc. Identifier are created to give unique name to C entities to identify it during the execution of program. For example:

```
int money;
int mango_tree;
```

Here, money is a identifier which denotes a variable of type integer. Similarly, mango_tree is another identifier, which

denotes another variable of type integer.

Rules for writing identifier

1. An identifier can be composed of letters (both uppercase and lowercase letters), digits and underscore '_' only.
2. The first letter of identifier should be either a letter or an underscore. But, it is discouraged to start an identifier name with an underscore though it is legal. It is because, identifier that starts with underscore can conflict with system names. In such cases, compiler will complain about it. Some system names that start with underscore are _fileno, _iob, _wfopen etc.
3. There is no rule for the length of an identifier. However, the first 31 characters of an identifier are discriminated by the compiler. So, the first 31 letters of two identifiers in a program should be different.

Tips for Good Programming Practice :

Programmer can choose the name of identifier whatever they want. However, if the programmer choose meaningful name for an identifier, it will be easy to understand and work on, particularly in case of large program.

C Programming Variables and Constants

Variables

Variables are memory location in computer's memory to store data. To indicate the memory location, each variable should be given a unique name called identifier. Variable names are just the symbolic representation of a memory location. Examples of variable name: sum, car_no, count etc.

```
int num;
```

Here, num is a variable of integer type.

Rules for writing variable name in C

1. Variable name can be composed of letters (both uppercase and lowercase letters), digits and underscore '_' only.
2. The first letter of a variable should be either a letter or an underscore. But, it is discouraged to start variable name with an underscore though it is legal. It is because, variable name that starts with underscore can conflict with system names and compiler may complain.
3. There is no rule for the length of length of a variable. However, the first 31 characters of a variable are discriminated by the compiler. So, the first 31 letters of two variables in a program should be different.

In C programming, you have to declare variable before using it in the program.

Constants

Constants are the terms that can't be changed during the execution of a program. For example: 1, 2.5, "Programming is easy." etc. In C, constants can be classified as:

Integer constants

Integer constants are the numeric constants(constant associated with number) without any fractional part or exponential part. There are three types of integer constants in C language: decimal constant(base 10), octal constant(base 8) and hexadecimal constant(base 16) .

Decimal digits: 0 1 2 3 4 5 6 7 8 9

Octal digits: 0 1 2 3 4 5 6 7

Hexadecimal digits: 0 1 2 3 4 5 6 7 8 9 A B C D E F.

For example:

```
Decimal constants: 0, -9, 22 etc
Octal constants: 021, 077, 033 etc
Hexadecimal constants: 0x7f, 0x2a, 0x521 etc
```

Notes:

1. You can use small caps a, b, c, d, e, f instead of uppercase letters while writing a hexadecimal constant.
2. Every octal constant starts with 0 and hexadecimal constant starts with 0x in C programming.

Floating-point constants

Floating point constants are the numeric constants that has either fractional form or exponent form. For example:

```
-2.0
0.0000234
-0.22E-5
```

Note: Here, E-5 represents 10^{-5}. Thus, -0.22E-5 = -0.0000022.

Character constants

Character constants are the constant which use single quotation around characters. For example: 'a', 'l', 'm', 'F' etc.

Escape Sequences

Sometimes, it is necessary to use newline(enter), tab, quotation mark etc. in the program which either cannot be typed or has special meaning in C programming. In such cases, escape sequence are used. For example: \n is used for newline. The backslash(\ ) causes "escape" from the normal way the characters are interpreted by the compiler.

Escape Sequences

Escape Sequences	Character
\b	Backspace
\f	Form feed
\n	Newline
\r	Return
\t	Horizontal tab
\v	Vertical tab
\\	Backslash
\'	Single quotation mark
\"	Double quotation mark
\?	Question mark
\0	Null character

String constants

String constants are the constants which are enclosed in a pair of double-quote marks. For example:

```
"good"                    //string constant
""                       //null string constant
"      "                 //string constant of six white space
"x"                      //string constant having single character.
"Earth is round\n"          //prints string with newline
```

Enumeration constants

Keyword enum is used to declare enumeration types. For example:

```
enum color {yellow, green, black, white};
```

Here, the variable name is color and yellow, green, black and white are the enumeration constants having value 0, 1, 2 and 3 respectively by default.

C Programming Data Types

In C, variable(data) should be declared before it can be used in program. Data types are the keywords, which are used for assigning a type to a variable.

Data types in C

1. Fundamental Data Types
- Integer types
- Floating Type
- Character types
2. Derived Data Types
- Arrays
- Pointers
- Structures
- Enumeration

Syntax for declaration of a variable

```
data_type variable_name;
```

Integer data types

Keyword int is used for declaring the variable with integer type. For example:

```
int var1;
```

Here, var1 is a variable of type integer.

The size of int is either 2 bytes(In older PC's) or 4 bytes. If you consider an integer having size of 4 byte(equal to 32 bits), it can take 2^{32} distinct states as: -2^{31},$-2^{31}+1$, ...,-2, -1, 0, 1, 2, ..., $2^{31}-2$, $2^{31}-1$

Similarly, int of 2 bytes, it can take 2^{16} distinct states from -2^{15} to $2^{15}-1$. If you try to store larger number than $2^{31}-1$, i.e,+2147483647 and smaller number than -2^{31}, i.e, -2147483648, program will not run correctly.

Floating types

Variables of floating types can hold real values(numbers) such as: 2.34, -9.382 etc. Keywords either float or double is used for declaring floating type variable. For example:

```
float var2;
double var3;
```

Here, both var2 and var3 are floating type variables.

In C, floating values can be represented in exponential form as well. For example:

```
float var3=22.442e2
```

Difference between float and double

Generally the size of float(Single precision float data type) is 4 bytes and that of double(Double precision float data type) is 8 bytes. Floating point variables has a precision of 6 digits whereas the the precision of double is 14 digits.

Note: Precision describes the number of significant decimal places that a floating values carries.

Character types

Keyword char is used for declaring the variable of character type. For example:

```
char var4='h';
```

Here, var4 is a variable of type character which is storing a character 'h'.

The size of char is 1 byte. The character data type consists of ASCII characters. Each character is given a specific value. For example:

```
For, 'a', value =97
For, 'b', value=98
For, 'A', value=65
For, '&', value=33
For, '2', value=49
```

Qualifiers

Qualifiers alters the meaning of base data types to yield a new data type.

Size qualifiers:

Size qualifiers alters the size of basic data type. The keywords long and short are two size qualifiers. For example:

```
long int i;
```

The size of int is either 2 bytes or 4 bytes but, when long keyword is used, that variable will be either 4 bytes of 8 bytes. Learn more about long keyword in C programming. If the larger size of variable is not needed then, short keyword can be used in similar manner as long keyword.

Sign qualifiers:

Whether a variable can hold only positive value or both values is specified by sign qualifiers. Keywords signed and unsigned are used for sign qualifiers.

```
unsigned int a;
// unsigned variable can hold zero and positive values only
```

It is not necessary to define variable using keyword signed because, a variable is signed by default. Sign qualifiers can be applied to only int and char data types. For a int variable of size 4 bytes it can hold data from -2^{31} to $2^{31}-1$ but, if that variable is defined unsigned, it can hold data from 0 to 2^{32} -1.

Constant qualifiers

Constant qualifiers can be declared with keyword const. An object declared by const cannot be modified.

```
const int p=20;
```

The value of p cannot be changed in the program.

Volatile qualifiers:

A variable should be declared volatile whenever its value can be changed by some external sources outside program. Keyword volatile is used to indicate volatile variable.

C Programming Input Output (I/O)

ANSI standard has defined many library functions for input and output in C language. Functions printf() and scanf() are the most commonly used to display out and take input respectively. Let us consider an example:

```
#include <stdio.h>      //This is needed to run printf() function.
int main()
{
    printf("C Programming");  //displays the content inside quotation
    return 0;
}
```

Output

```
C Programming
```

Explanation of How this program works

1. Every program starts from main() function.
2. printf() is a library function to display output which only works if #include<stdio.h>is included at the beginning.
3. Here, stdio.h is a header file (standard input output header file) and #include is command to paste the code from the header file when necessary. When compiler encounters printf() function and doesn't find stdio.h header file, compiler shows error.
4. Code return 0; indicates the end of program. You can ignore this statement but, it is good programming practice to use return 0;.

I/O of integers in C

```
#include<stdio.h>
int main()
{
    int c=5;
    printf("Number=%d",c);
```

```
    return 0;
}
```

Output

```
Number=5
```

Inside quotation of printf() there, is a conversion format string "%d" (for integer). If this conversion format string matches with remaining argument,i.e, c in this case, value of c is displayed.

```
#include<stdio.h>
int main()
{
    int c;
    printf("Enter a number\n");
    scanf("%d",&c);
    printf("Number=%d",c);
    return 0;
}
```

Output

```
Enter a number
4
Number=4
```

The scanf() function is used to take input from user. In this program, the user is asked a input and value is stored in variable c. Note the '&' sign before c. &c denotes the address of c and value is stored in that address.

I/O of floats in C

```
#include <stdio.h>
int main(){
    float a;
    printf("Enter value: ");
    scanf("%f",&a);
    printf("Value=%f",a);    //%f is used for floats instead of %d
    return 0;
}
```

Output

```
Enter value: 23.45
Value=23.450000
```

Conversion format string "%f" is used for floats to take input and to display floating value of a variable.

I/O of characters and ASCII code

```
#include <stdio.h>
int main(){
    char var1;
    printf("Enter character: ");
    scanf("%c",&var1);
    printf("You entered %c.",var1);
```

```
    return 0;
}
```

Output

```
Enter character: g
You entered g.
```

Conversion format string "%c" is used in case of characters.

ASCII code

When character is typed in the above program, the character itself is not recorded a numeric value(ASCII value) is stored. And when we displayed that value by using "%c", that character is displayed.

```
#include <stdio.h>
int main(){
    char var1;
    printf("Enter character: ");
    scanf("%c",&var1);
    printf("You entered %c.\n",var1);
/* \n prints the next line(performs work of enter). */
    printf("ASCII value of %d",var1);
    return 0;
}
```

Output

```
Enter character:
g
103
```

When, 'g' is entered, ASCII value 103 is stored instead of g.

You can display character if you know ASCII code only. This is shown by following example.

```
#include <stdio.h>
int main(){
    int var1=69;
    printf("Character of ASCII value 69: %c",var1);
    return 0;
}
```

Output

```
Character of ASCII value 69: E
```

The ASCII value of 'A' is 65, 'B' is 66 and so on to 'Z' is 90. Similarly ASCII value of 'a' is 97, 'b' is 98 and so on to 'z' is 122.

More about Input/Output of floats and Integer

Variations in Output for integer an floats

Integer and floating-points can be displayed in different formats in C programming as:

```
#include<stdio.h>
int main(){
    printf("Case 1:%6d\n",9876);
/*  Prints the number right justified within 6 columns  */
    printf("Case 2:%3d\n",9876);
/* Prints the number to be right justified to 3 columns but, there are 4 digits so number
is not right justified  */
    printf("Case 3:%.2f\n",987.6543);
/* Prints the number rounded to two decimal places */
    printf("Case 4:%.f\n",987.6543);
/* Prints the number rounded to 0 decimal place, i.e, rounded to integer */
    printf("Case 5:%e\n",987.6543);
/* Prints the number in exponential notation(scientific notation) */
    return 0;
}
```

Output

```
Case 1:  9876
Case 2:9876
Case 3:987.65
Case 4:988
Case 5:9.876543e+002
```

Variations in Input for integer and floats

```
#include <stdio.h>
int main(){
    int a,b;
    float c,d;
    printf("Enter two intgers: ");
/*Two integers can be taken from user at once as below*/
    scanf("%d%d",&a,&b);
    printf("Enter intger and floating point numbers: ");
/*Integer and floating point number can be taken at once from user as below*/
    scanf("%d%f",&a,&c);
    return 0;
}
```

Similarly, any number of input can be taken at once from user.

C Programming Operators

Operators are the symbol which operates on value or a variable. For example: + is a operator to perform addition.

C programming language has wide range of operators to perform various operations. For better understanding of operators, these operators can be classified as:

Operators In C Programming
Arithmetic Operators

Increment and Decrement Operators
Assignment Operators
Relational Operators
Logical Operators
Conditional Operators
Bitwise Operators
Special Operators

Arithmetic Operators

Operator	Meaning Of Operator
+	addition or unary plus
-	subtraction or unary minus
*	multiplication
/	division
%	remainder after division(modulo division)

Example of working of arithmetic operators

```
/* Program to demonstrate the working of arithmetic operators in C.  */
#include <stdio.h>
int main(){
    int a=9,b=4,c;
    c=a+b;
    printf("a+b=%d\n",c);
    c=a-b;
    printf("a-b=%d\n",c);
    c=a*b;
    printf("a*b=%d\n",c);
    c=a/b;
```

```
    printf("a/b=%d\n",c);
    c=a%b;
    printf("Remainder when a divided by b=%d\n",c);
    return 0;
}
}
```

```
a+b=13
a-b=5
a*b=36
a/b=2
Remainder when a divided by b=1
```

Explanation

Here, the operators +, - and * performed normally as you expected. In normal calculation, 9/4 equals to 2.25. But, the output is 2 in this program. It is because, a and b are both integers. So, the output is also integer and the compiler neglects the term after decimal point and shows answer 2 instead of 2.25. And, finally a%b is 1,i.e. ,when a=9 is divided by b=4, remainder is 1.

```
Suppose a=5.0, b=2.0, c=5 and d=2
In C programming,
a/b=2.5
a/d=2.5
c/b=2.5
c/d=2
```

Note: % operator can only be used with integers.

Increment and decrement operators

In C, ++ and -- are called increment and decrement operators respectively. Both of these operators are unary operators, i.e, used on single operand. ++ adds 1 to operand and -- subtracts 1 to operand respectively. For example:

```
Let a=5 and b=10
a++;  //a becomes 6
a--;  //a becomes 5
++a;  //a becomes 6
--a;  //a becomes 5
```

Difference between ++ and -- operator as postfix and prefix

When i++ is used as prefix(like: ++var), ++var will increment the value of var and then return it but, if ++ is used as postfix(like: var++), operator will return the value of operand first and then only increment it. This can be demonstrated by an example:

```
#include <stdio.h>
int main(){
    int c=2,d=2;
    printf("%d\n",c++); //this statement displays 2 then, only c incremented by 1 to 3.
    printf("%d",++c);   //this statement increments 1 to c then, only c is displayed.
    return 0;
}
```

Output

```
2
4
```

Assignment Operators

The most common assignment operator is =. This operator assigns the value in right side to the left side. For example:

```
var=5  //5 is assigned to var
a=c;   //value of c is assigned to a
5=c;   // Error! 5 is a constant.
```

Operator	Example	Same As
=	a=b	a=b
+=	a+=b	a=a+b
-=	a-=b	a=a-b
*=	a*=b	a=a*b
/=	a/=b	a=a/b
%=	a%=b	a=a%b

Relational Operator

Relational operators checks relationship between two operands. If the relation is true, it returns value 1 and if the relation is false, it returns value 0. For example:

```
a>b
```

Here, > is a relational operator. If a is greater than b, a>b returns 1 if not then, it returns 0.

Relational operators are used in decision making and loops in C programming.

Operator	Meaning Of Operator	Example
==	Equal to	5==3 returns false (0)
>	Greater than	5>3 returns true (1)
<	Less than	5<3 returns false (0)
!=	Not equal to	5!=3 returns true(1)

Operator	Meaning Of Operator	Example
>=	Greater than or equal to	5>=3 returns true (1)
<=	Less than or equal to	5<=3 return false (0)

Logical Operators

Logical operators are used to combine expressions containing relation operators. In C, there are 3 logical operators:

Operator	Meaning Of Operator	Example
&&	Logial AND	If c=5 and d=2 then,((c==5) && (d>5)) returns false.
||	Logical OR	If c=5 and d=2 then, ((c==5) || (d>5)) returns true.
!	Logical NOT	If c=5 then, !(c==5) returns false.

Explanation

For expression, ((c==5) && (d>5)) to be true, both c==5 and d>5 should be true but, (d>5) is false in the given example. So, the expression is false. For expression ((c==5) || (d>5)) to be true, either the expression should be true. Since, (c==5) is true. So, the expression is true. Since, expression (c==5) is true, !(c==5) is false.

Conditional Operator

Conditional operator takes three operands and consists of two symbols ? and : . Conditional operators are used for decision making in C. For example:

```
c=(c>0)?10:-10;
```

If c is greater than 0, value of c will be 10 but, if c is less than 0, value of c will be -10.

Bitwise Operators

A bitwise operator works on each bit of data. Bitwise operators are used in bit level programming.

Operators	Meaning Of Operators
&	Bitwise AND