**C Variables, Constants and Literals**

**Variables**

In programming, a variable is a container (storage area) to hold data.

To indicate the storage area, each variable should be given a unique name (identifier). Variable names are just the symbolic representation of a memory location. For example:

int playerScore = 95;

Here, playerScore is a variable of int type. Here, the variable is assigned an integer value 95.

The value of a variable can be changed, hence the name variable.

char ch = 'a';

// some code

ch = 'l';

**Rules for naming a variable**

1. A variable name can only have letters (both uppercase and lowercase letters), digits and underscore.
2. The first letter of a variable should be either a letter or an underscore.
3. There is no rule on how long a variable name (identifier) can be. However, you may run into problems in some compilers if the variable name is longer than 31 characters.

**Note:** You should always try to give meaningful names to variables. For example: firstName is a better variable name than fn.

C is a strongly typed language. This means that the variable type cannot be changed once it is declared. For example:

int number = 5; // integer variable

number = 5.5; // error

double number; // error

Here, the type of number variable is int. You cannot assign a floating-point (decimal) value 5.5 to this variable. Also, you cannot redefine the data type of the variable to double. By the way, to store the decimal values in C, you need to declare its type to either double or float.

**Literals**

Literals are data used for representing fixed values. They can be used directly in the code. For example: 1, 2.5, 'c' etc.

Here, 1, 2.5 and 'c' are literals. Why? You cannot assign different values to these terms.

**1. Integers**

An integer is a numeric literal(associated with numbers) without any fractional or exponential part. There are three types of integer literals in C programming:

* decimal (base 10)
* octal (base 8)
* hexadecimal (base 16)

For example:

Decimal: 0, -9, 22 etc

Octal: 021, 077, 033 etc

Hexadecimal: 0x7f, 0x2a, 0x521 etc

**2. Floating-point Literals**

A floating-point literal is a numeric literal that has either a fractional form or an exponent form. For example:

-2.0

0.0000234

-0.22E-5

**3. Characters**

A character literal is created by enclosing a single character inside single quotation marks. For example: 'a', 'm', 'F', '2', '}' etc.

**4. Escape Sequences**

Sometimes, it is necessary to use characters that cannot be typed or has special meaning in C programming. For example: newline(enter), tab, question mark etc.

In order to use these characters, escape sequences are used.

| 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 |

For example: \n is used for a newline. The backslash \ causes escape from the normal way the characters are handled by the compiler.

**5. String Literals**

A string literal is a sequence of characters enclosed in double-quote marks. For example:

"good" //string constant

"" //null string constant

" " //string constant of six white space

"x" //string constant having a single character.

"Earth is round\n" //prints string with a newline

**Constants**

If you want to define a variable whose value cannot be changed, you can use the const keyword. This will create a constant. For example,

const double PI = 3.14;

Notice, we have added keyword const.

Here, PI is a symbolic constant; its value cannot be changed.

const double PI = 3.14;

PI = 2.9; //Error

**C Data Types**

In C programming, data types are declarations for variables. This determines the type and size of data associated with variables. For example,

int myVar;

Here, myVar is a variable of int (integer) type. The size of int is 4 bytes.

**Basic types**

Here's a table containing commonly used types in C programming for quick access.

| **Type** | **Size (bytes)** | **Format Specifier** |
| --- | --- | --- |
| int | at least 2, usually 4 | %d, %i |
| char | 1 | %c |
| float | 4 | %f |
| double | 8 | %lf |
| short int | 2 usually | %hd |
| unsigned int | at least 2, usually 4 | %u |
| long int | at least 4, usually 8 | %ld, %li |
| long long int | at least 8 | %lld, %lli |
| unsigned long int | at least 4 | %lu |
| unsigned long long int | at least 8 | %llu |
| signed char | 1 | %c |
| unsigned char | 1 | %c |
| long double | at least 10, usually 12 or 16 | %Lf |

**int**

Integers are whole numbers that can have both zero, positive and negative values but no decimal values. For example, 0, -5, 10

We can use int for declaring an integer variable.

int id;

Here, id is a variable of type integer.

You can declare multiple variables at once in C programming. For example,

int id, age;

The size of int is usually 4 bytes (32 bits). And, it can take 232 distinct states from -2147483648 to 2147483647.

**float and double**

float and double are used to hold real numbers.

float salary;

double price;

In C, floating-point numbers can also be represented in exponential. For example,

float normalizationFactor = 22.442e2;

What's the difference between float and double?

The size of float (single precision float data type) is 4 bytes. And the size of double (double precision float data type) is 8 bytes.

**char**

Keyword char is used for declaring character type variables. For example,

char test = 'h';

The size of the character variable is 1 byte.

**void**

void is an incomplete type. It means "nothing" or "no type". You can think of void as **absent**.

For example, if a function is not returning anything, its return type should be void.

Note that, you cannot create variables of void type.

**short and long**

If you need to use a large number, you can use a type specifier long. Here's how:

long a;

long long b;

long double c;

Here variables a and b can store integer values. And, c can store a floating-point number.

If you are sure, only a small integer ([−32,767, +32,767] range) will be used, you can use short.

short d;

You can always check the size of a variable using the sizeof() operator.

#include <stdio.h>

main()

{

short a;

long b;

long long c;

long double d;

printf("size of short = %d bytes\n", sizeof(a));

printf("size of long = %d bytes\n", sizeof(b));

printf("size of long long = %d bytes\n", sizeof(c));

printf("size of long double= %d bytes\n", sizeof(d));

}

**signed and unsigned**

In C, signed and unsigned are type modifiers. You can alter the data storage of a data type by using them:

* signed - allows for storage of both positive and negative numbers
* unsigned - allows for storage of only positive numbers

For example,

// valid codes

unsigned int x = 35;

int y = -35; // signed int

int z = 36; // signed int

// invalid code: unsigned int cannot hold negative integers

unsigned int num = -35;

Here, the variables x and num can hold only zero and positive values because we have used the unsigned modifier.

Considering the size of int is 4 bytes, variable y can hold values from -231 to 231-1, whereas variable x can hold values from 0 to 232-1.

**Derived Data Types**

Data types that are derived from fundamental data types are derived types. For example: arrays, pointers, function types, structures, etc.

We will learn about these derived data types in later tutorials.

* bool type
* Enumerated type
* Complex types

**C Input Output (I/O)**

In this tutorial, you will learn to use scanf() function to take input from the user, and printf() function to display output to the user.

**C Output**

In C programming, printf() is one of the main output function. The function sends formatted output to the screen. For example,

**Example 1: C Output**

#include <stdio.h>

main()

{

// Displays the string inside quotations

printf("C Programming");

}

**Output**

C Programming

How does this program work?

* All valid C programs must contain the main() function. The code execution begins from the start of the main() function.
* The printf() is a library function to send formatted output to the screen. The function prints the string inside quotations.
* To use printf() in our program, we need to include stdio.h header file using the #include <stdio.h> statement.
* The return 0; statement inside the main() function is the "Exit status" of the program. It's optional.

**Example 2: Integer Output**

#include <stdio.h>

main()

{

int testInteger = 5;

printf("Number = %d", testInteger);

return 0;

}

**Output**

Number = 5

We use %d format specifier to print int types. Here, the %d inside the quotations will be replaced by the value of testInteger.

**Example 3: float and double Output**

#include <stdio.h>

main()

{

float number1 = 13.5;

double number2 = 12.4;

printf("number1 = %f\n", number1);

printf("number2 = %lf", number2);

return 0;

}

**Output**

number1 = 13.500000

number2 = 12.400000

To print float, we use %f format specifier. Similarly, we use %lf to print double values.

**Example 4: Print Characters**

#include <stdio.h>

main()

{

char chr = 'a';

printf("character = %c", chr);

}

**Output**

character = a

**C Input**

In C programming, scanf() is one of the commonly used function to take input from the user. The scanf() function reads formatted input from the standard input such as keyboards.

**Example 5: Integer Input/Output**

main()

{

int testInteger;

printf("Enter an integer: ");

scanf("%d", &testInteger);

printf("Number = %d",testInteger);

}

**Output**

Enter an integer: 4

Number = 4

Here, we have used %d format specifier inside the scanf() function to take int input from the user. When the user enters an integer, it is stored in the testInteger variable.

Notice, that we have used &testInteger inside scanf(). It is because &testInteger gets the address of testInteger, and the value entered by the user is stored in that address.

**Example 6: Float and Double Input/Output**

#include <stdio.h>

main()

{

float num1;

double num2;

printf("Enter a number: ");

scanf("%f", &num1);

printf("Enter another number: ");

scanf("%lf", &num2);

printf("num1 = %f\n", num1);

printf("num2 = %lf", num2);

}

**Output**

Enter a number: 12.523

Enter another number: 10.2

num1 = 12.523000

num2 = 10.200000

**Example 7: C Character I/O**

#include <stdio.h>

main()

{

char chr;

printf("Enter a character: ");

scanf("%c",&chr);

printf("You entered %c.", chr);

}

**Output**

Enter a character: g

You entered g

When a character is entered by the user in the above program, the character itself is not stored. Instead, an integer value (ASCII value) is stored.

And when we display that value using %c text format, the entered character is displayed. If we use %d to display the character, it's ASCII value is printed.

**Example 8: ASCII Value**

#include <stdio.h>

main()

{

char chr;

printf("Enter a character: ");

scanf("%c", &chr);

printf("You entered %c.\n",chr);

printf("ASCII value is %d.", chr);

}

Output

Enter a character: g

You entered g.

ASCII value is 103.

**I/O Multiple Values**

Here's how you can take multiple inputs from the user and display them.

#include <stdio.h>

main()

{

int a;

float b;

printf("Enter integer and then a float: ");

scanf("%d%f", &a, &b);

printf("You entered %d and %f", a, b);

}

**Output**

Enter integer and then a float: -3

3.4

You entered -3 and 3.400000

**Format Specifiers for I/O**

As you can see from the above examples, we use

* %d for int
* %f for float
* %lf for double
* %c for char

Here's a list of commonly used C data types and their format specifiers.

| **Data Type** | **Format Specifier** |
| --- | --- |
| int | %d |
| char | %c |
| float | %f |
| double | %lf |
| short int | %hd |
| unsigned int | %u |
| long int | %li |
| long long int | %lli |
| unsigned long int | %lu |
| unsigned long long int | %llu |
| signed char | %c |
| unsigned char | %c |
| long double | %Lf |