C - Pointers

Pointers in C are easy and fun to learn. Some C programming tasks are performed more easily with pointers, and other tasks, such as dynamic memory allocation, cannot be performed without using pointers. So it becomes necessary to learn pointers to become a perfect C programmer. Let's start learning them in simple and easy steps.

As you know, every variable is a memory location and every memory location has its address defined which can be accessed using ampersand (&) operator, which denotes an address in memory. Consider the following example, which will print the address of the variables defined:

```
#include <stdio.h>

int main ()
{
   int  var1;
   char var2[10];

   printf("Address of var1 variable: %x\n", &var1  );
   printf("Address of var2 variable: %x\n", &var2  );

   return 0;
}
```

When the above code is compiled and executed, it produces result something as follows:

```
Address of var1 variable: bff5a400
Address of var2 variable: bff5a3f6
```

So you understood what is memory address and how to access it, so base of the concept is over. Now let us see what is a pointer.

What Are Pointers?

A **pointer** is a variable whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before you can use it to store any variable address. The general form of a pointer variable declaration is:

```
type *var-name;
```

Here, **type** is the pointer's base type; it must be a valid C data type and **var-name** is the name of the pointer variable. The asterisk * you used to declare a pointer is the same asterisk that you use for multiplication. However, in this statement the asterisk is being used to designate a variable as a pointer. Following are the valid pointer declaration:

```
int    *ip;    /* pointer to an integer */
double *dp;    /* pointer to a double */
float  *fp;    /* pointer to a float */
char   *ch     /* pointer to a character */
```

The actual data type of the value of all pointers, whether integer, float, character, or otherwise, is the same, a long hexadecimal number that represents a memory address. The only difference between pointers of different data types is the data type of the variable or constant that the pointer points to.

How to use Pointers?

There are few important operations, which we will do with the help of pointers very frequently. **(a)** we define a pointer variable **(b)** assign the address of a variable to a pointer and **(c)** finally access the value at the address available in the pointer variable. This is done by using unary operator * that returns the value of the variable located at the address specified by its operand. Following example makes use of these operations:

```
#include <stdio.h>
```

```
int main ()
{
   int  var = 20;   /* actual variable declaration */
   int  *ip;        /* pointer variable declaration */

   ip = &var;  /* store address of var in pointer variable*/

   printf("Address of var variable: %x\n", &var  );

   /* address stored in pointer variable */
   printf("Address stored in ip variable: %x\n", ip );

   /* access the value using the pointer */
   printf("Value of *ip variable: %d\n", *ip );

   return 0;
}
```

When the above code is compiled and executed, it produces result something as follows:

```
Address of var variable: bffd8b3c
Address stored in ip variable: bffd8b3c
Value of *ip variable: 20
```

NULL Pointers in C

It is always a good practice to assign a NULL value to a pointer variable in case you do not have exact address to be assigned. This is done at the time of variable declaration. A pointer that is assigned NULL is called a **null** pointer.

The NULL pointer is a constant with a value of zero defined in several standard libraries. Consider the following program:

```
#include <stdio.h>

int main ()
{
   int  *ptr = NULL;

   printf("The value of ptr is : %x\n", ptr  );

   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
The value of ptr is 0
```

On most of the operating systems, programs are not permitted to access memory at address 0 because that memory is reserved by the operating system. However, the memory address 0 has special significance; it signals that the pointer is not intended to point to an accessible memory location. But by convention, if a pointer contains the null (zero) value, it is assumed to point to nothing.

---

## C Pointers in Detail.

Pointers have many but easy concepts and they are very important to C programming. There are following few important pointer concepts which should be clear to a C programmer:

| Concept | Description |
| --- | --- |
| **C - Pointer arithmetic** | There are four arithmetic operators that can be used on pointers: ++, --, +, - |
| **C - Array of pointers** | You can define arrays to hold a number of pointers. |
| **C - Pointer to pointer** | C allows you to have pointer on a pointer and so on. |
| **Passing pointers to functions in C** | Passing an argument by reference or by address both enable the passed argument to be changed in the calling function by the called function. |
| **Return pointer from functions in C** | C allows a function to return a pointer to local variable, static variable and dynamically allocated memory as well. |
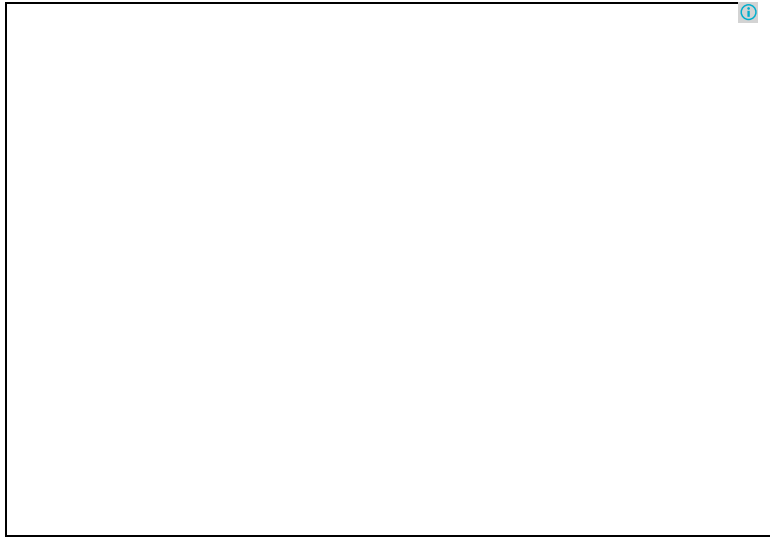
◢ *Extras*

## Mobile First

## About us

▸ Company

▸ Team

▸ Careers

▸ Privacy Policy

▸ Terms of use

## Extra Links

▸ Forums

▸ Articles

▸ Shared

▸ Seo Tools

▸ Contact

## Contact Us

📍 **Address:** 388-A , Road no 22, Jubilee Hills, Hyderabad Telangana, INDIA-500033

✉ **Email: Click Here**

⊛ **Website: www.tutorialspoint.com**