Pointers in C Programming with examples

A **pointer** is a variable that stores the address of another variable. Unlike other variables that hold values of a certain type, pointer holds the address of a variable. For example, an integer variable holds (or you can say stores) an integer value, however an integer pointer holds the address of a integer variable. In this guide, we will discuss pointers in C programming with the help of examples.

Before we discuss about **pointers in C**, lets take a simple example to understand what do we mean by the address of a variable.

A simple example to understand how to access the address of a variable without pointers?

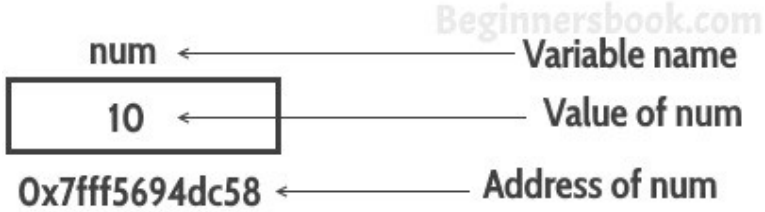
In this program, we have a variable num of int type. The value of num is 10 and this value must be stored somewhere in the memory, right? A memory space is allocated for each variable that holds the value of that variable, this memory space has an address. For example we live in a house and our house has an address, which helps other people to find our house. The same way the value of the variable is stored in a memory address, which helps the C program to find that value when it is needed.

So let's say the address assigned to variable num is 0x7fff5694dc58, which means whatever value we would be assigning to num should be stored at the location: 0x7fff5694dc58. See the diagram below.

```
#include <stdio.h>
int main()
{
   int num = 10;
   printf("Value of variable num is: %d", num);
   /* To print the address of a variable we use %p
    * format specifier and ampersand (&) sign just
    * before the variable name like &num.
    */
   printf("\nAddress of variable num is: %p", &num);
   return 0;
}
```

Output:

```
Value of variable num is: 10
Address of variable num is: 0x7fff5694dc58
```



A Simple Example of Pointers in C

This program shows how a pointer is declared and used. There are several other things that we can do with pointers, we have discussed them later in this guide. For now, we just need to know how to link a pointer to the address of a variable.

Important point to note is: The data type of pointer and the variable must match, an int pointer can hold the address of int variable, similarly a pointer declared with float data type can hold the address of a float variable. In the example below, the pointer and the variable both are of int type.

```
#include <stdio.h>
int main()
{
   //Variable declaration
   int num = 10;

   //Pointer declaration
   int *p;

   //Assigning address of num to the pointer p
   p = #

   printf("Address of variable num is: %p", p);
   return 0;
}
```

Output:

```
Address of variable num is: 0x7fff5694dc58
```

C Pointers - Operators that are used with Pointers

Lets discuss the operators & and * that are used with Pointers in C.

"Address of"(&) Operator

We have already seen in the first example that we can display the address of a variable using ampersand sign. I have used &num to access the address of variable num. The **& operator** is also known as "**Address of**" Operator.

```
printf("Address of var is: %p", &num);
```

Point to note: %p is a format specifier which is used for displaying the address in hex format.
Now that you know how to get the address of a variable but **how to store that address in some other variable?** That's where pointers comes into picture. As mentioned in the beginning of this guide, pointers in C programming are used for holding the address of another variables.

Pointer is just like another variable, the main difference is that it stores address of another variable rather than a value.

"Value at Address"(*) Operator

The * Operator is also known as **Value at address** operator.

How to declare a pointer?

```
int *p1  /*Pointer to an integer variable*/
double *p2 /*Pointer to a variable of data type double*/
char *p3   /*Pointer to a character variable*/
float *p4   /*pointer to a float variable*/
```

The above are the few examples of pointer declarations. **If you need a pointer to store the address of integer variable then the data type of the pointer should be int**. Same case is with the other data types.

By using * operator we can access the value of a variable through a pointer.
For example:

```
double a = 10;
double *p;
```

```
p = &a;
```
*p would give us the value of the variable a. The following statement would display 10 as output.

```
printf("%d", *p);
```
Similarly if we assign a value to *pointer like this:

```
*p = 200;
```
It would change the value of variable a. The statement above will change the value of a from 10 to 200.

Example of Pointer demonstrating the use of & and *

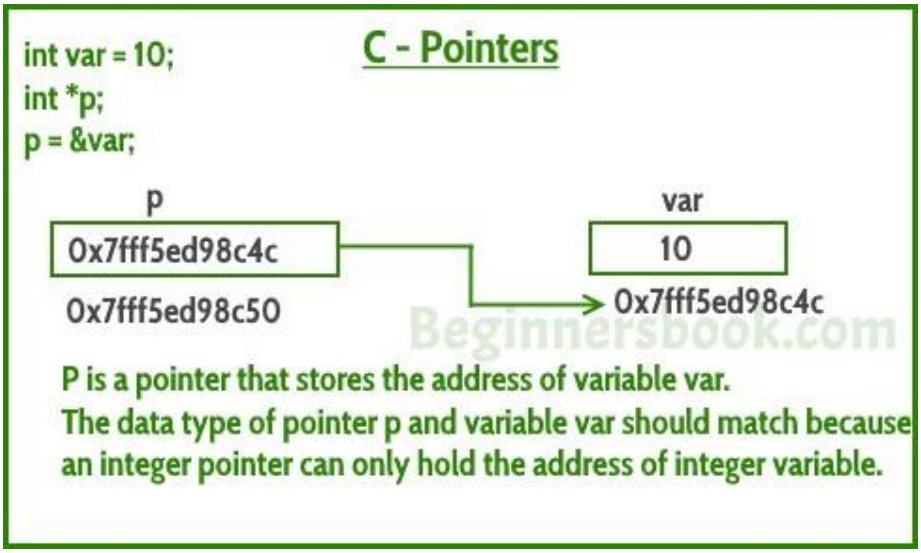
```
#include <stdio.h>
int main()
{
   /* Pointer of integer type, this can hold the
    * address of a integer type variable.
    */
   int *p;

   int var = 10;

   /* Assigning the address of variable var to the pointer
    * p. The p can hold the address of var because var is
    * an integer type variable.
    */
   p= &var;

   printf("Value of variable var is: %d", var);
   printf("\nValue of variable var is: %d", *p);
   printf("\nAddress of variable var is: %p", &var);
   printf("\nAddress of variable var is: %p", p);
   printf("\nAddress of pointer p is: %p", &p);
   return 0;
}
```
Output:

```
Value of variable var is: 10
Value of variable var is: 10
Address of variable var is: 0x7fff5ed98c4c
Address of variable var is: 0x7fff5ed98c4c
Address of pointer p is: 0x7fff5ed98c50
```



Lets take few more examples to understand it better –
Lets say we have a char variable ch and a pointer ptr that holds the address of ch.

```
char ch='a';
char *ptr;
```
Read the value of ch

```
printf("Value of ch: %c", ch);
or
printf("Value of ch: %c", *ptr);
```
Change the value of ch

```
ch = 'b';
or
*ptr = 'b';
```
The above code would replace the value ‘a’ with ‘b’.

Can you guess the output of following C program?

```
#include <stdio.h>
int main()
{
    int var =10;
    int *p;
    p= &var;

    printf ( "Address of var is: %p", &var);
    printf ( "\nAddress of var is: %p", p);

    printf ( "\nValue of var is: %d", var);
    printf ( "\nValue of var is: %d", *p);
    printf ( "\nValue of var is: %d", *( &var));

    /* Note I have used %p for p's value as it represents an address*/
    printf( "\nValue of pointer p is: %p", p);
    printf ( "\nAddress of pointer p is: %p", &p);

    return 0;
}
```
Output:

```
Address of var is: 0x7fff5d027c58
Address of var is: 0x7fff5d027c58
Value of var is: 10
Value of var is: 10
Value of var is: 10
Value of pointer p is: 0x7fff5d027c58
Address of pointer p is: 0x7fff5d027c50
```
More Topics on Pointers

1) **Pointer to Pointer** – A pointer can point to another pointer (which means it can store the address of another pointer), such pointers are known as double pointer OR pointer to pointer.

2) **Passing pointers to function** – Pointers can also be passed as an argument to a function, using this feature a function can be called by reference as well as an array can be passed to a function while calling.

3) **Function pointers** – A function pointer is just like another pointer, it is used for storing the address of a function. Function pointer can also be used for calling a function in C program.