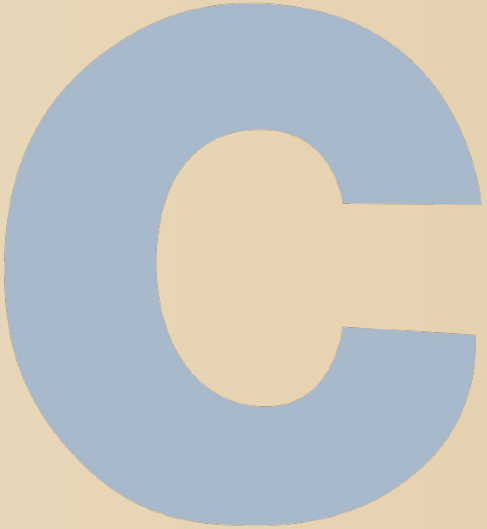
C Pointers

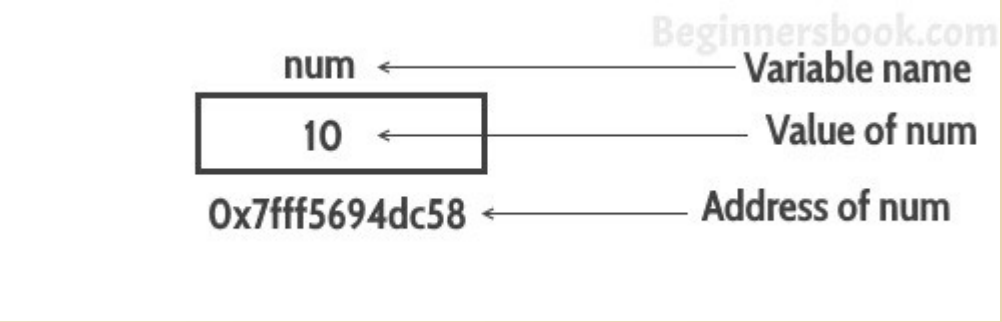
THE



PROGRAMMING LANGUAGE

C Pointers

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.



C Pointers

A simple example 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.

p1.c

```c
#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;
}
```

C Pointers

A simple example 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.

```
#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;
}
```

C Pointers

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.

p2.c

```c
#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;
}
```

C Pointers

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.

```
#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;
}
```

C Pointers

C Pointers – Operators that are used with Pointers

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

C Pointers

C Pointers – Operators that are used with Pointers

"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.

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 lesson, 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.

C Pointers

C Pointers – Operators that are used with Pointers

"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*/
```

C Pointers

Example of Pointer demonstrating the use of & and *

p3.c

```c
#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;
}
```

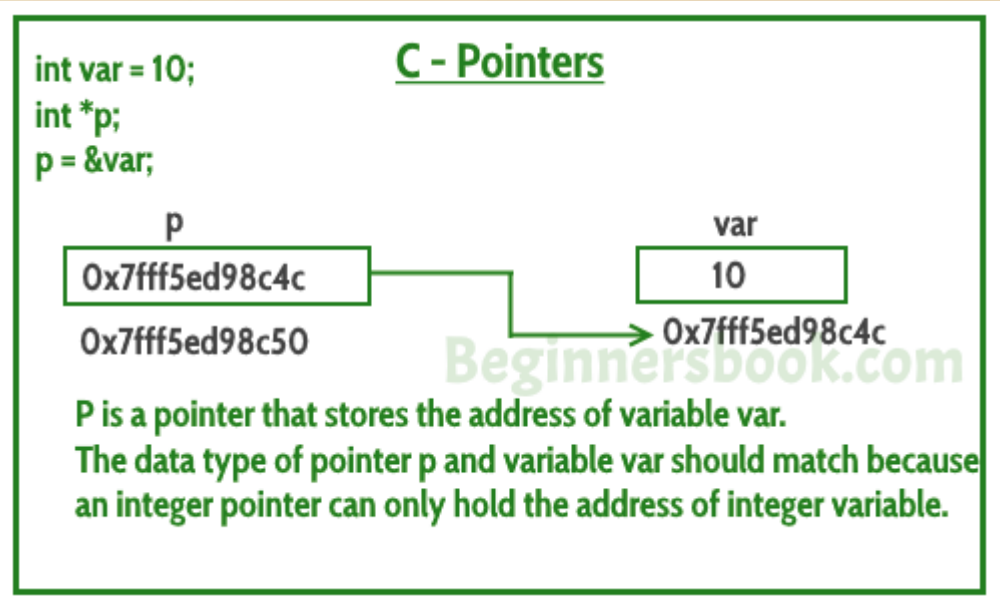
C Pointers

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;
}
```

C Pointers

Example of Pointer demonstrating the use of & and *



C Pointers

Few more examples to understand it better

```
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'.

C Pointers

Can you guess the output of following C program?

p4.c

```c
#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;
}
```

C Pointers

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;
}
```

C Pointers

Can you guess the output of following C program?

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
```

C Pointers

One more example

```
File  Edit  View  Search  Terminal  Help
(base) grivis@Grivis-Main:~/Documents/Active/TimS$ ./a.out
1 0x7fffb9b63d60
2 0x7fffb9b63d64 3
3 0x7fffb9b63d68 4
4 0x7fffb9b63d6c 5
5 0x7fffb9b63d70 6

(base) grivis@Grivis-Main:~/Documents/Active/TimS$
```

pointers.c

```c
#include <stdio.h>
#define NEWLINE putchar('\n');


int main()
{ int array[] = {1, 2, 3, 4, 5};
  int *ptr, *ptra, *ptrb;

  ptr = array;

  printf("%d ", *ptr);
  printf("%p \n", ptr);

  ptr++;
  printf("%d ", *ptr);
  printf("%p ", ptr);
  (*ptr)++;
  printf("%d \n", *ptr);

  ptr++;
  printf("%d ", *ptr);
  printf("%p ", ptr);
  (*ptr)++;
  printf("%d \n", *ptr);

  ptr++;
  printf("%d ", *ptr);
  printf("%p ", ptr);
  (*ptr)++;
  printf("%d \n", *ptr);

  ptr++;
  printf("%d ", *ptr);
  printf("%p ", ptr);
  (*ptr)++;
  printf("%d \n", *ptr);

//

NEWLINE
```

C Pointers

The End of the Lesson