Pointers

Pointers are one of the core components of the C programming language. A pointer can be used to store the memory address of other variables, functions, or even other pointers. The use of pointers allows low-level memory access, dynamic memory allocation, and many other functionalities in C.

Syntax of C Pointers

The syntax of pointers is similar to the variable declaration in C, but we use the (*) dereferencing operator in the pointer declaration.

datatype * ptr;

I. Pointer Declaration

In pointer declaration, we only declare the pointer but do not initialize it. To declare a pointer, we use the (*) dereference operator before its name.

Example

int *ptr;

The pointer declared here will point to some random memory address as it is not initialized. Such pointers are called wild pointers.

2. Pointer Initialization

Pointer initialization is the process where we assign some initial value to the pointer variable. We generally use the

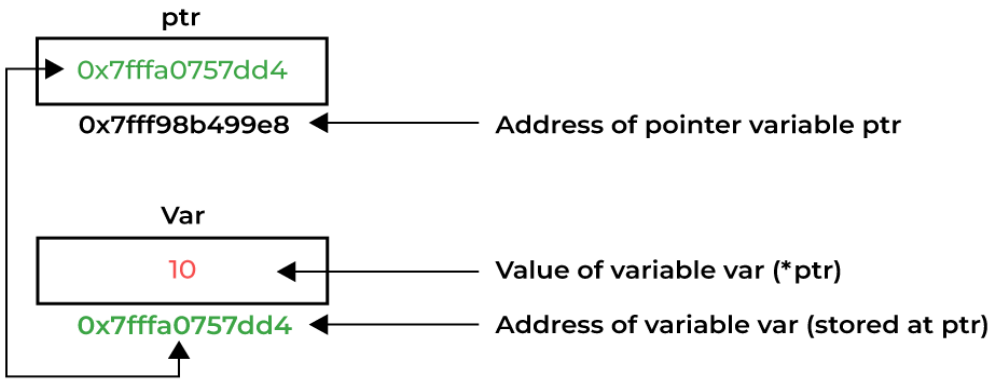
(&:) address of to get the memory address of a variable and then store it in the pointer variable.

Example

int var — 10;

int * ptr;

ptr = &var;



C Pointer Example

```
#include <stdio.h>

void geeks()
{
    int var = 10;

    // declare pointer variable
    int* ptr;

    ptr = &var;

    printf("Value at ptr = %p \n", ptr);
    printf("Value at var = %d \n", var);
    printf("Value at *ptr = %d \n", *ptr);
}

int main()
{
    geeks();
    return 0;
}
```

Dereference Pointer in C The dereference operator is used to access and manipulate the value stored in the variable pointed by the pointer. The dereference or indirection operator acts as a unary operator, and it needs a pointer variable as its operand.

```
#include <stdio.h>

int main() {

  int x = 10;

  int *ptr;

  ptr = &x;

  printf("Value of x = %d\n", *ptr);

  return 0;

}
```

Pointer Arithmetic's

Pointer Arithmetic is the set of valid arithmetic operations that can be performed on pointers. The pointer variables store the memory address of another variable. It doesn't store any value.

```
#include <stdio.h>

int main(){
```

```
  int x = 10;
  int *y = &x;
  printf("Value of y before increment: %d\n", y);
  y++;
  printf("Value of y after increment: %d", y);
}
```

Array of Pointers

In C, a pointer array is a homogeneous collection of indexed pointer variables that are references to a memory location. It is generally used in C Programming when we want to point at multiple memory locations of a similar data type in our C program. We can access the data by dereferencing the pointer pointing to it.

Syntax:

```
pointer_type *array_name [array_size];
```

```
#include <stdio.h>
int main()
{
   int var1 = 10;
   int var2 = 20;
   int var3 = 30;
   int* ptr_arr[3] = { &var1, &var2, &var3 };
   for (int i = 0; i < 3; i++) {
      printf("Value of var%d: %d\tAddress: %p\n", i + 1, *ptr_arr[i], ptr_arr[i]);
   }
   return 0;
}
```

Function Pointer

In C, like normal data pointers (int *, char *, etc), we can have pointers to functions. Following is a simple example that shows declaration and function call using function pointer.

```
#include <stdio.h>
void fun(int a) { printf("Value of a is %d\n", a); }
int main()
```

```
{
   void (*fun_ptr)(int) = &fun;
   void (*fun_ptr)(int);
   fun_ptr = &fun;
   (*fun_ptr)(10);
   return 0;
}
```

Pointer to Pointer (Double Pointer)

The pointer to a pointer in C is used when we want to store the address of another pointer. The first pointer is used to store the address of the variable. And the second pointer is used to store the address of the first pointer. That is why they are also known as double-pointers.

```
data_type_of_pointer **name_of_variable = & normal_pointer_variable;
```

```
#include <stdio.h>
int main()
{
        int var = 789;
        int* ptr2;
        int** ptr1;
        ptr2 = &var;
        ptr1 = &ptr2;
        printf("Value of var = %d\n", var);
        printf("Value of var using single pointer = %d\n", *ptr2);
        printf("Value of var using double pointer = %d\n", **ptr1);
        return 0;
}
```

void Pointer

A void pointer is a pointer that has no associated data type with it. A void pointer can hold an address of any type and can be type casted to any type.

```
#include <stdio.h>
int main()
```

```
{
        int a = 10;
        char b = 'x';
        void* p = &a;
        p = &b;
}
```