

Pointer Arithmetics in C with Examples

Difficulty Level : Medium • Last Updated : 27 Jul, 2021

[Pointers](#) variables are also known as **address data types** because they are used to store the address of another variable. The address is the memory location that is assigned to the variable. It doesn't store any value.

Hence, there are only a few operations that are allowed to perform on [Pointers in C language](#). The operations are slightly different from the ones that we generally use for mathematical calculations. The operations are:

1. Increment/Decrement of a Pointer
2. Addition of integer to a pointer
3. Subtraction of integer to a pointer
4. Subtracting two pointers of the same type

Increment/Decrement of a Pointer

Increment: It is a condition that also comes under addition. When a pointer is incremented, it actually increments by the number equal to the size of the data type for which it is a pointer.

For Example:

If an integer pointer that stores **address 1000** is incremented, then it will increment by 2 (**size of an int**) and the new address it will point to **1002**. While if a float type pointer is incremented then it will increment by 4 (**size of a float**) and the new address will be **1004**.

Decrement: It is a condition that also comes under subtraction. When a pointer is decremented, it actually decrements by the number equal to the size of the data type.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge

Start Your Coding Journey Now!

[Login](#)[Register](#)

address will be **996**.

Below is the program to illustrate pointer increment/decrement:

C

```
// C program to illustrate  
// pointer increment/decrement
```

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

```
// Driver Code
```



[Data Structures](#) [Algorithms](#) [Interview Preparation](#) [Topic-wise Practice](#) [C++](#) [Java](#) [Python](#)

```
int *ptr1, *ptr2;  
  
// Pointer stores  
// the address of N  
ptr1 = &N;  
ptr2 = &N;  
  
printf("Pointer ptr1 "  
       "before Increment: ");  
printf("%p \n", ptr1);  
  
// Incrementing pointer ptr1;  
ptr1++;  
  
printf("Pointer ptr1 after"  
       " Increment: ");  
printf("%p \n\n", ptr1);  
  
printf("Pointer ptr1 before"  
       " Decrement: ");  
printf("%p \n", ptr1);  
  
// Decrementing pointer ptr1;
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge



Simpler tools lead to happier developers. And
happier developers lead to better results

GET \$100 FREE CREDIT

[HIDE AD](#) • [AD VIA BUYSPELLADS](#)

Start Your Coding Journey Now!

[Login](#)[Register](#)

Output:

Pointer ptr1 before Increment: 0x7ffcb19385e4

Pointer ptr1 after Increment: 0x7ffcb19385e8

Pointer ptr1 before Decrement: 0x7ffcb19385e8

Pointer ptr1 after Decrement: 0x7ffcb19385e4

Addition

When a pointer is added with a value, the value is first multiplied by the size of data type and then added to the pointer.

C

```
// C program to illustrate pointer Addition
#include <stdio.h>

// Driver Code
int main()
{
    // Integer variable
    int N = 4;

    // Pointer to an integer
    int *ptr1, *ptr2;

    // Pointer stores the address of N
    ptr1 = &N;
    ptr2 = &N;

    printf("Pointer ptr2 before Addition: ");
    printf("%p \n", ptr2);

    // Addition of 3 to ptr2
    ptr2 = ptr2 + 3;
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge

Start Your Coding Journey Now!

[Login](#)[Register](#)

Pointer ptr2 before Addition: 0x7fffffffcd984

Pointer ptr2 after Addition: 0x7fffffffcd990

Subtraction

When a pointer is subtracted with a value, the value is first multiplied by the size of the data type and then subtracted from the pointer.

Below is the program to illustrate pointer Subtraction:

C

```
// C program to illustrate pointer Subtraction
#include <stdio.h>

// Driver Code
int main()
{
    // Integer variable
    int N = 4;

    // Pointer to an integer
    int *ptr1, *ptr2;

    // Pointer stores the address of N
    ptr1 = &N;
    ptr2 = &N;

    printf("Pointer ptr2 before Subtraction: ");
    printf("%p \n", ptr2);

    // Subtraction of 3 to ptr2
    ptr2 = ptr2 - 3;
    printf("Pointer ptr2 after Subtraction: ");
    printf("%p \n", ptr2);

    return 0;
}
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge

Start Your Coding Journey Now!

[Login](#)[Register](#)

Subtraction of Two Pointers

The subtraction of two pointers is possible only when they have the same data type. The result is generated by calculating the difference between the addresses of the two pointers and calculating how many bits of data it is according to the pointer data type. The subtraction of two pointers gives the increments between the two pointers.

For Example:

Two integer pointers say **ptr1(address:1000)** and **ptr2(address:1016)** are subtracted. The difference between address is 16 bytes. Since the size of int is 2 bytes, therefore the **increment between ptr1 and ptr2** is given by **(16/2) = 8**.

Below is the implementation to illustrate the Subtraction of Two Pointers:

C

```
// C program to illustrate Subtraction
// of two pointers
#include <stdio.h>

// Driver Code
int main()
{
    int x;

    // Integer variable
    int N = 4;

    // Pointer to an integer
    int *ptr1, *ptr2;

    // Pointer stores the address of N
    ptr1 = &N;
    ptr2 = &N;

    // Incrementing ptr2 by 3
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
    return 0;  
}
```

Output:

Subtraction of ptr1 & ptr2 is 3

Pointer Arithmetic on Arrays:

Pointers contain addresses. Adding two addresses makes no sense because there is no idea what it would point to. Subtracting two addresses lets you compute the offset between the two addresses. An array name acts like a pointer constant. The value of this pointer constant is the address of the first element. **For Example:** if an array named arr then arr and &arr[0] can be used to reference array as a pointer.

Below is the program to illustrate the Pointer Arithmetic on arrays:

Program 1:

C

```
// C program to illustrate the array  
// traversal using pointers  
#include <stdio.h>  
  
// Driver Code  
int main()  
{  
  
    int N = 5;  
  
    // An array  
    int arr[] = { 1, 2, 3, 4, 5 };  
  
    // Declare pointer variable
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
        printf("%d ", ptr[0]);  
        ptr++;  
    }  
}
```

Output:

1 2 3 4 5

Program 2:

C

```
// C program to illustrate the array  
// traversal using pointers in 2D array  
#include <stdio.h>  
  
// Function to traverse 2D array  
// using pointers  
void traverseArr(int* arr,  
                int N, int M)  
{  
  
    int i, j;  
  
    // Traverse rows of 2D matrix  
    for (i = 0; i < N; i++) {  
  
        // Traverse columns of 2D matrix  
        for (j = 0; j < M; j++) {  
  
            // Print the element  
            printf("%d ", *((arr + i * M) + j));  
        }  
        printf("\n");  
    }  
}
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
        { 5, 6 } };
```

```
// Function Call  
traverseArr((int*)arr, N, M);  
return 0;  
}
```

Output:


```
1 2  
3 4  
5 6
```

If (Coding)

```
{  
  C foundation course = true;  
  Focus = 100;  
}  
cout << "Success";
```

Wait no more!

[Start Learning](#)



Like 21

[Previous](#)[Next](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge

Start Your Coding Journey Now!

[Login](#)[Register](#)

02 Difference between passing
pointer to pointer and address of
pointer to any function
25, Apr 21

Examples
27, Feb 20

03 What is a Pointer to a Null pointer
28, Aug 19

07 Difference between NULL pointer,
Null character ('\0') and '0' in C
with Examples
29, May 20

04 Difference between Dangling
pointer and Void pointer
12, Jul 21

08 How to declare a pointer to a
function?
01, Sep 09

Article Contributed By :



vishalraina
@vishalraina

Vote for difficulty

Current difficulty : [Medium](#)

[Easy](#)[Normal](#)[Medium](#)[Hard](#)[Expert](#)

Improved By : [surinderdawra388](#), [simmytarika5](#)

Article Tags : [C-Advanced Pointer](#), [C-Pointer Basics](#), [C-Pointers](#), [Pointers](#), [C Language](#),
[C Programs](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge



Simpler tools lead to happier developers. And
happier developers lead to better results

GET \$100 FREE CREDIT

[HIDE AD](#) • [AD VIA BUYSSELLADS](#)

Start Your Coding Journey Now!

[Login](#)[Register](#)

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

[Load Comments](#)

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

- [About Us](#)
- [Careers](#)
- [In Media](#)
- [Contact Us](#)
- [Privacy Policy](#)
- [Copyright Policy](#)

News

- [Top News](#)
- [Technology](#)
- [Work & Career](#)
- [Business](#)
- [Finance](#)
- [Lifestyle](#)

Learn

- [Algorithms](#)
- [Data Structures](#)
- [SDE Cheat Sheet](#)
- [Machine learning](#)
- [CS Subjects](#)
- [Video Tutorials](#)

Languages

- [Python](#)
- [Java](#)
- [CPP](#)
- [Golang](#)
- [C#](#)
- [SQL](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge



Simpler tools lead to happier developers. And
happier developers lead to better results

GET \$100 FREE CREDIT

[HIDE AD](#) • [AD VIA BUYSPELLADS](#)

Start Your Coding Journey Now!

Login

Register

javascript

internships

Bootstrap

Video Internship

@geeksforgeeks , Some rights reserved

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge



Simpler tools lead to happier developers. And happier developers lead to better results

GET \$100 FREE CREDIT

HIDE AD • AD VIA BUYSSELLADS