EDUCBA
(https://www.educba.com/software-development/)

← (https://www.educba.com/dangling-pointers-in-c/)

→ (https://www.educba.com/c-operators/)

Introduction to Pointer Arithmetic in C

The following article provides an outline for Pointer Arithmetic in C. As we are well awa
Pointers are one of the most interesting topics in C. Pointers are basically the variables which hold the address that points to a specific memory location accessed using '&' operator. Pointers

(https://www.educba.com/software-development/)

Both the unary and binary operations can be performed on pointers like:

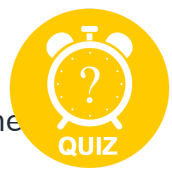
- Increment
- Decrement
- Addition (addition of any integer value to a pointer)
- Subtraction (either any integer value or the subtraction of 2 pointers)
- Comparison

All the above mentioned arithmetic operations can be performed on pointers as they are integers and nothing else. But some operations seem to be useless while performing as there is no idea of what they would result.

Arithmetic Operations along with their Examples in C

Given below are the pointer arithmetic operations and their implementation in C code:

1. Increment



By incrementing the value to a pointer to 1, it will start pointing to the next address/ me location. Incrementing the value of pointer is very useful while traversing the array in C. In order to access the next element of the array, we can simply use ptr++. Value is incremented

(https://www.educba.com/software-development/)

```
#include<stdio.h>
int main(){
int num =50;
char a = 'x';
// pointer 'ptr' to point the above 'num' and 'ptr1' to point 'a'
int *ptr;
char *ptr1;
// pointer 'ptr' holding the address of 'num' location and 'ptr1'
to hold the address of character 'a'
ptr = &num;
ptr1 = &a;
printf("\n The address which the pointer holds is %u",ptr);
printf("\n The address which the pointer holds is %u",ptr1);
// incrementing the value of pointer by 1
ptr++;
ptr1++;
// Pointer address will now gets incremented by 4 bytes as it
holds the address of integer value
printf("\n Now the address which the pointer holds is %u",ptr);
// Pointer address will now gets incremented by 1 byte as it holds
the address of character value

printf("\n Now the address which the pointer holds is %u",ptr1
return 0;
}
```

Output:

```
The address which the pointer holds is 1378350060
The address which the pointer holds is 1378350059
Now the address which the pointer holds is 1378350064
Now the address which the pointer holds is 1378350060
```

2. Decrement

Decrement operation works similar to the increment operation in case of pointers.

Decrementing a pointer value using ‘ptr–’ will decrease its value by 1 resulting in the previous

(https://www.educba.com/software-development/)

```
#include<stdio.h>
int main(){
float num = 50.3;
char a = 'x';
// pointer 'ptr' to point the above 'num' and 'ptr1' to point 'a'
float *ptr;
char *ptr1;
// pointer 'ptr' holding the address of 'num' location and 'ptr1'
to hold the address of character 'a'
ptr = &num;
ptr1 = &a;
printf("\n The address which the pointer holds is %u",ptr);
printf("\n The address which the pointer holds is %u",ptr1);
// decrementing the value of pointer by 1
ptr--;
ptr1--;
// Pointer address will now gets decremented by 4 bytes as it
holds the address of float value
printf("\n Now the address which the pointer holds is %u",ptr);
// Pointer address will now gets decremented by 1 byte as it holds
the address of character value


printf("\n Now the address which the pointer holds is %u",ptr1
return 0;
}
```

(https://www.educba.com/software-development/)

```
Now the address which the pointer holds is 1106958152
Now the address which the pointer holds is 1106958154
```

3. Addition

We cannot add the two pointers, as it would also result in the address of an unknown memory location. So there is no use of it. But we can add any integer value to the pointer in order to point to that memory location. Addition of integer value works according to the datatype of the value pointer pointing to using ptr+x. For example, if the pointer holds the address of any integer value (64 – bit integer system having 4 bytes integer), on adding +2 in it, it will increment the value by 8 bytes.

Code:

```
#include<stdio.h>
int main(){
double num = 50.3;
char a = 'u';
// pointer 'ptr' to point the above 'num' and 'ptr1' to point 'a'
double *ptr;
char *ptr1;
// pointer 'ptr' holding the address of 'num' location and 'ptr1'
to hold the address of character 'a'

ptr = &num;
ptr1 = &a;
printf("\n The address which the pointer holds is %u",ptr);
```

(https://www.educba.com/software-development/)

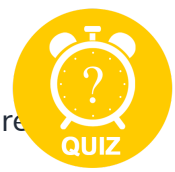
```
// Pointer address will now gets incremented by 4*8 bytes as it
holds the address of double value
printf("\n Now the address which the pointer holds is %u",ptr);
// Pointer address will now gets incremented by 4*1 bytes as it
holds the address of character value
printf("\n Now the address which the pointer holds is %u",ptr1);
return 0;
}
```

Output:

```
The address which the pointer holds is 3223767000
The address which the pointer holds is 3223766999
Now the address which the pointer holds is 3223767032
Now the address which the pointer holds is 3223767003
```

4. Subtraction

Subtraction in case of pointers is possible with two addresses (i.e. with 2 pointer values) as well as subtraction of an integer value from the pointer. Subtraction of an integer value from the pointer works similar to the addition of integer value as discussed above, i.e. any integer value can be subtracted from the pointer using ptr-x. And it will result the difference of pointer by x * bytes of value datatype hold by pointer.



In subtraction of 2 pointers, both the pointers need to be of the same data type and it re
an integer value which is useful in case of arrays when we want to find the number of elements in between them using the 2 addresses.

(https://www.educba.com/software-development/)

```
#include<stdio.h>
int main(){
double num = 50.3;
char a = 'u';
// pointer 'ptr' to point the above 'num' and 'ptr1' to point 'a'
double *ptr;
char *ptr1;
// pointer 'ptr' holding the address of 'num' location and 'ptr1'
to hold the address of character 'a'
ptr = &num;
ptr1 = &a;
printf("\n The address which the pointer holds is %u",ptr);
printf("\n The address which the pointer holds is %u",ptr1);
// subtracting the integer value 4 to the pointer value
ptr = ptr - 4;
ptr1 = ptr1 - 4;
// Pointer address will now gets decreased by 4*8 bytes as it
holds the address of double value
printf("\n Now the address which the pointer holds is %u",ptr);
// Pointer address will now gets decreased by 4*1 bytes as it
holds the address of character value

printf("\n Now the address which the pointer holds is %u",ptr1
return 0;
}
```

(https://www.educba.com/software-development/)

```
Now the address which the pointer holds is 2459298296
Now the address which the pointer holds is 2459298323
```

5. Comparison

C provides a wonderful feature of comparing the 2 pointers with only the condition being that both the pointers are of the same type and pointing to the same array. All the comparison operations like (>, <, <=, >=, ==, !=) can be performed on them. Infact, C does not throw an error on the console in comparison of 2 pointers pointing to different datatype.

Code:

```
#include <stdio.h>
int main()
{
int arr1[6] = {100, 200, 300, 400, 500, 600};
// pointer 'ptr1' pointing to the address of 1st array element
int *ptr1 = &arr1[0];
printf("\n Array elements are given below:");
while(ptr1 < &arr1[6])
{
printf("\n array value is %d ", *ptr1);
//Incrementing the pointer to move to the address of next element

ptr1++;
}
return 0;
```

(https://www.educba.com/software-development/)

```
array value is 100
array value is 200
array value is 300
array value is 400
array value is 500
array value is 600
```

Conclusion

Above description clearly explains what are pointers and the various arithmetic operations that can be performed on them in C. Pointers are very useful when performing operations in arrays or in other data structures like linked list, stack, etc. One needs to understand them thoroughly before implementing them in the code as sometimes they return very absurd results. Like addition of 2 pointers is possible but there is no use of it as it would result in the address of some memory location which we don't know.

Recommended Articles

This is a guide to Pointer Arithmetic in C. Here we discuss the introduction to Pointer Arithmetic in C with 5 arithmetic operations along with their examples. You may also have a look at the following articles to learn more –

1. Tokens in C (https://www.educba.com/tokens-in-c/)
2. Address Operator in C (https://www.educba.com/address-operator-in-c/)
3. Dangling Pointers in C (https://www.educba.com/dangling-pointers-in-c/)
4. Void Pointer in C (https://www.educba.com/void-pointer-in-c/)

ALL IN ONE SOFTWARE DEVELOPMENT BUNDLE

(https://www.educba.com/software-development/)

Apps

iPhone & iPad (https://itunes.apple.com/in/app/educba-learning-app/id1341654580?mt=8)

Android (https://play.google.com/store/apps/details?id=com.educba.www)

Resources

Free Courses (https://www.educba.com/software-development/free-courses/?source=footer)

Java Tutorials (https://www.educba.com/software-development/software-development-tutorials/java-tutorial/?source=footer)

Python Tutorials (https://www.educba.com/software-development/software-development-tutorials/python-tutorial/?source=footer)

All Tutorials (https://www.educba.com/software-development/software-development-tutorials/?source=footer)

Certification Courses

All Courses (https://www.educba.com/software-development/courses/?source=footer)

Software Development Course - All in One Bundle (https://www.educba.com/software-development/courses/software-development-course/?source=footer)

Become a Python Developer (https://www.educba.com/software-development/courses/python-certification-course/?source=footer)

Java Course (https://www.educba.com/software-development/courses/jav course/?source=footer)

Become a Selenium Automation Tester (https://www.educba.com/software-development/courses/selenium-training-certification/?source=footer)

(https://www.educba.com/software-development/)

PHP Course (https://www.educba.com/software-development/courses/php-course/?source=footer)