Thank you for printing our content at www.domain-name.com. Please check back soon for new contents. NEW

(https://programiz.pro/learn/master-c-programming?utr

Programiz PRO Learn to Code with 100+ Interactive Challenges and Quizzes. Start Learning C Programming bar&utm_campaign=programiz&utm_medium=referral) (https://programiz.pro/learn/master-c-programming?utm_source=top-Today! Enroll for FREE <u>Course</u> v <u>Tutorials</u> v <u>Examples</u> v Q Search tutorials and examples www.domain-name.com <u>utm_source=sidebar-</u> navigation&utm_campaign=programiz&utm_medium=referral)
Relationship Between Arrays and Pointers C Introduction In this tutorial, you'll learn about the relationship between arrays and pointers in C programming. You will also learn to access array elements using pointers. C Flow Control Before you learn about the relationship between arrays and pointers, be sure to check C Functions these two topics: C Programming Arrays <u>C Arrays (/c-programming/c-arrays)</u> <u>C Pointers (/c-programming/c-pointers)</u> C Programming Pointers <u>(/c-</u> Programming programming/c-Relationship Between Arrays and Pointers <u>Pointers</u> <u>pointers)</u> C Pointers (/c-programming/c-An array is a block of sequential data. Let's write a program to print addresses of <u>& Arrays</u> <u>pointers-arrays)</u> array elements. C Pointers (/c-<u>programming/c-</u> <u>Functions</u> <u>pointer-functions</u>) #include <stdio.h> int main() { C Memory(/c-programming/cint x[4];Allocationdynamic-memory-<u>allocation)</u> for(i = 0; i < 4; ++i) { Array & (/cprintf("&x[%d] = %p\n", i, &x[i]); <u>programming/c-</u> <u>Examples</u> <u>pointer-examples)</u> printf("Address of array x: %p", x); C Programming Strings Structure And Union Output C Programming Files &x[0] = 1450734448&x[1] = 1450734452 **Additional Topics** &x[2] = 1450734456 &x[3] = 1450734460 Address of array x: 1450734448 There is a difference of 4 bytes between two consecutive elements of array x. It is because the size of [int] is 4 bytes (on our compiler). Notice that, the address of [8x[0]] and [x] is the same. It's because the variable name \times points to the first element of the array. x[0] x[1] x[2] x[3]Relation between Arrays and Pointers From the above example, it is clear that [x] is equivalent to [x]. And, [x] is equivalent to [*x]. Similarly, • [&x[1]] is equivalent to [x+1] and [x[1]] is equivalent to [*(x+1)]. • [&x[2]] is equivalent to [x+2] and [x[2]] is equivalent to [*(x+2)]. • Basically, &x[i] is equivalent to x+i and x[i] is equivalent to *(x+i). **Example 1: Pointers and Arrays** int main() {

#include <stdio.h>
int main() {
 int i, x[6], sum = 0;

 printf("Enter 6 numbers: ");

 for(i = 0; i < 6; ++i) {
 // Equivalent to scanf("%d", &x[i]);
 scanf("%d", x+i);

 // Equivalent to sum += x[i]
 sum += *(x+i);
 }

 printf("Sum = %d", sum);

 return 0;
}</pre>

When you run the program, the output will be:

Related Topics

<u>array-pointer)</u>

<u>pointers)</u>

<u>examples)</u>

C Arrays

<u>functions)</u>

<u>Access Array Elements Using Pointer</u> (/c-programming/examples/access-

(<u>/c-programming/c-pointers</u>)

(/c-programming/c-structures-

C Array and Pointer Examples

<u>(/c-programming/c-pointer-</u>

<u>(/c-programming/c-arrays)</u>

C Pass Addresses and Pointers

<u>(/c-programming/c-pointer-</u>

C structs and Pointers

Enter 6 numbers: 2
3
4
4
5um = 29

Here, we have declared an array \times of 6 elements. To access elements of the array, we have used pointers.

In most contexts, array names decay to pointers. In simple words, array names are converted to pointers. That's the reason why you can use pointers to access elements of arrays. However, you should remember that **pointers and arrays are not the same**.

There are a few cases where array names don't decay to pointers. To learn more, visit:

When does array name doesn't decay into a pointer?

(https://stackoverflow.com/questions/17752978/exceptions-to-array-decaying-into-a-pointer)

Example 2: Arrays and Pointers

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

int x[5] = {1, 2, 3, 4, 5};
int* ptr;

// ptr is assigned the address of the third element
ptr = &x[2];

printf("*ptr = %d \n", *ptr); // 3
printf("*(ptr+1) = %d \n", *(ptr+1)); // 4
printf("*(ptr-1) = %d", *(ptr-1)); // 2

return 0;
}

When you run the program, the output will be:

Share on:

*ptr = 3
*(ptr+1) = 4
*(ptr-1) = 2

In this example, &x[2], the address of the third element, is assigned to the ptr pointer. Hence, 3 was displayed when we printed *ptr.

And, printing *(ptr+1) gives us the fourth element. Similarly, printing *(ptr-1) gives us the second element.

(https://www.facebook.com/sharer/sharer.php? u=https://www.programiz.com/c-programming/c-pointers-arrays)

Did you find this article helpful?

Lettle helpful?

<u>Learn C Interactively</u> × (https://www.programiz.com/learn-c?utm_campaign=programiz-homepage&utm_source=programiz-website-c-app-popup)

Thank you for printing our content at www.domain-name.com. Please check back soon for new contents. (https://programiz.pro/learn/master-c-programming?utm_source=top-Programiz PRO Learn to Code with 100+ Interactive Challenges and Quizzes. Start Learning C Programming bar&utm_campaign=programiz&utm_medium=referral)

Today! Enroll for FREE <u>Tutorials</u> <u>Examples</u> Search tutorials and examples www.domain-name.com **Related Tutorials** <u>C Tutorial</u> <u>C Tutorial</u> <u>C Tutorial</u> <u>C Tutorial</u> **C Pointers** C Array and Pointer **C** structs and Pointers <u>C Programming Strings</u> **Examples** (<u>/c-programming/c-pointer-examples)</u> (/c-programming/c-pointers) (/c-programming/c-structures-(/c-programming/c-strings) <u>pointers)</u>