Array of Pointers in C

Last Updated : 30 Jul, 2024

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];
```

Here,

- **pointer_type:** Type of data the pointer is pointing to.
- **array_name:** Name of the array of pointers.
- **array_size:** Size of the array of pointers.

> ***Note:*** *It is important to keep in mind the operator precedence and associativity in the array of pointers declarations of different type as a single change will mean the whole different thing. For example, enclosing *array_name in the parenthesis will mean that array_name is a pointer to an array.*

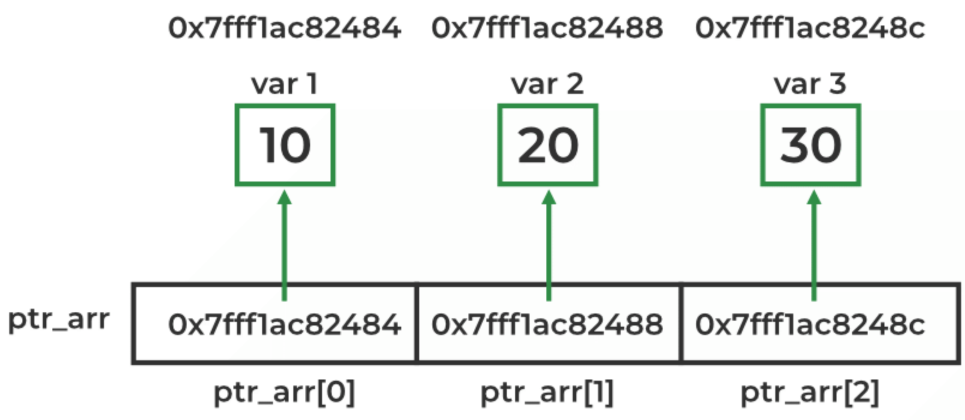
Example:

```c
// C program to demonstrate the use of array of pointer
#include <stdio.h>

int main()
{
    // declaring some temp variables
    int var1 = 10;
    int var2 = 20;
    int var3 = 30;

    // array of pointers to integers
    int* ptr_arr[3] = { &var1, &var2, &var3 };

    // traversing using loop
    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;
}
```

Output

```
Value of var1: 10    Address: 0x7fff1ac82484
Value of var2: 20    Address: 0x7fff1ac82488
Value of var3: 30    Address: 0x7fff1ac8248c
```

Explanation:



As shown in the above example, each element of the array is a pointer pointing to an integer. We can access the value of these integers by first selecting the array element and then dereferencing it to get the value.

Array of Pointers to Character

One of the main applications of the array of pointers is to store multiple strings as an array of pointers to characters. Here, each pointer in the array is a character pointer that points to the first character of the string.

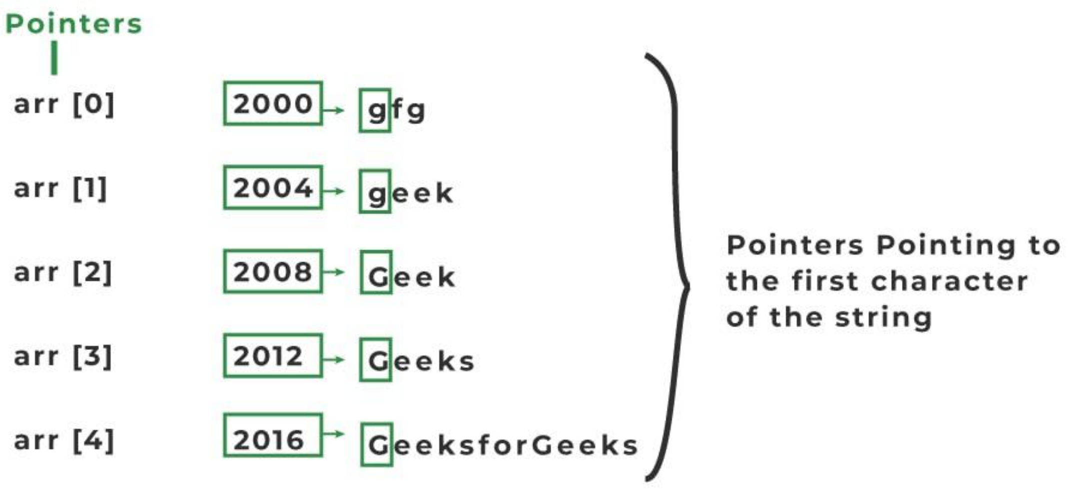
Syntax:

```
char *array_name [array_size];
```

After that, we can assign a string of any length to these pointers.

Example:

```
char* arr[5]
    = { "gfg", "geek", "Geek", "Geeks", "GeeksforGeeks" }
```



***Note:** Here, each string will take different amount of space so offset will not be the same and does not follow any particular order.*

This method of storing strings has the advantage of the traditional array of strings. Consider the following two examples:

Example 1:

```
// C Program to print Array of strings without array of pointers
#include <stdio.h>
int main()
{
    char str[3][10] = { "Geek", "Geeks", "Geekfor" };

    printf("String array Elements are:\n");

    for (int i = 0; i < 3; i++) {
        printf("%s\n", str[i]);
```

```c
    }

    return 0;
}
```

Output

```
String array Elements are:
Geek
Geeks
Geekfor
```

In the above program, we have declared the 3 rows and 10 columns of our array of strings. But because of predefining the size of the array of strings the space consumption of the program increases if the memory is not utilized properly or left unused. Now let's try to store the same strings in an array of pointers.

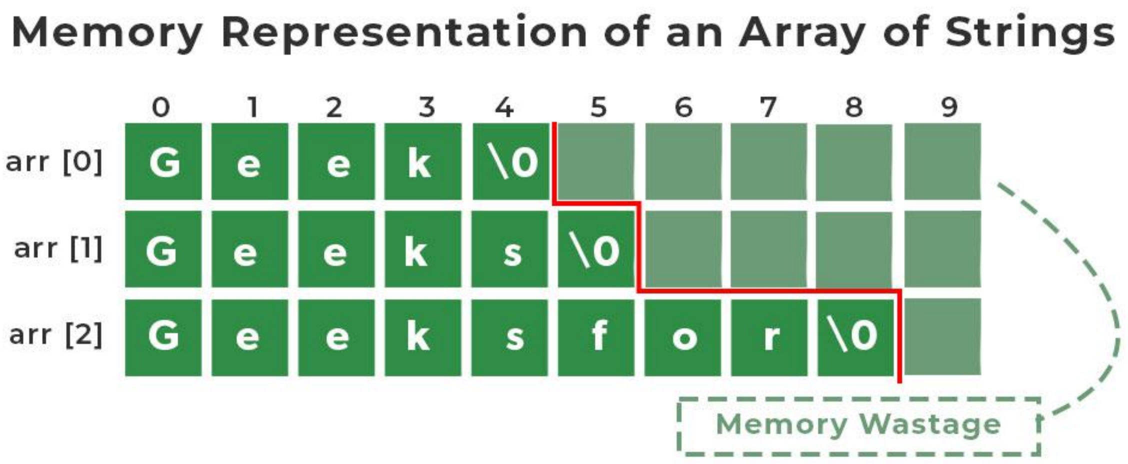
Example 2:

```c
// C Program to print Array of strings with array of po e
#include <stdio.h>

int main() {
    // Declare an array of pointers to characters
    char* arr[] = { "geek", "Geeks", "Geeksfor" };

    // Print each string and its address
    for (int i = 0; i < 3; i++) {
        printf("%s\n", arr[i]);
    }

    // Print addresses of each string
    for (int i = 0; i < 3; i++) {
        printf("Address of arr[%d]: %p\n", i, (void*)arr[i]);
    }

    return 0;
}
```

Output

```
geek
Geeks
Geeksfor
Address of arr[0]: 0x400634
Address of arr[1]: 0x400639
Address of arr[2]: 0x40063f
```

Here, the total memory used is the memory required for storing the strings and pointers without leaving any empty space hence, saving a lot of wasted space. We can understand this using the image shown below.



The space occupied by the array of pointers to characters is shown by solid green blocks excluding the memory required for storing the pointer while the space occupied by the array of strings includes both solid and light green blocks.

Array of Pointers to Different Types

Not only we can define the array of pointers for basic data types like int, char, float, etc. but we can also define them for derived and user-defined data types such as arrays, structures, etc. Let's consider the below example where we create an array of pointers pointing to a function for performing the different operations.

Example:

```
// C program to illustrate the use of array of pointers
// function
#include <stdio.h>

```

```c
// some basic arithmetic operations
void add(int a, int b) {
  printf("Sum : %d\n", a + b);
}

void subtract(int a, int b) {
    printf("Difference : %d\n", a - b);
}

void multiply(int a, int b) {
    printf("Product : %d\n", a * b);
}

void divide(int a, int b) {
    printf("Quotient : %d", a / b);
}

int main() {

    int x = 50, y = 5;

    // array of pointers to function of return type int
    void (*arr[4])(int, int)
        = { &add, &subtract, &multiply, &divide };
    for (int i = 0; i < 4; i++) {
        arr[i](x, y);
    }
    return 0;
}
```

Output

```
Sum : 55
Difference : 45
Product : 250
Quotient : 10
```

Application of Array of Pointers

An array of pointers is useful in a wide range of cases. Some of these applications are listed below:

- It is most commonly used to store multiple strings.
- It is also used to implement LinkedHashMap in C and also in the Chaining technique of collision resolving in Hashing.
- It is used in sorting algorithms like bucket sort.
- It can be used with any pointer type so it is useful when we have separate declarations of multiple entities and we want to store them in a single place.

Disadvantages of Array of Pointers

The array of pointers also has its fair share of disadvantages and should be used when the advantages outweigh the disadvantages. Some of the disadvantages of the array of pointers are:

- **Higher Memory Consumption:** An array of pointers requires more memory as compared to plain arrays because of the additional space required to store pointers.
- **Complexity:** An array of pointers might be complex to use as compared to a simple array.
- **Prone to Bugs:** As we use pointers, all the bugs associated with pointers come with it so we need to handle them carefully.

Related Article: Pointer to an Array | Array Pointer

Comment More info Advertise with us

Next Article
Applications of Pointers in C

Similar Reads

Pointer vs Array in C

Most of the time, pointer and array accesses can be treated as acting the same, the major exceptions being:Â Â 1. the sizeof operator sizeof(array) returns the amount of memory used by all elements in the...

1 min read

Pointers vs Array in C++

Arrays and pointers are two derived data types in C++ that have a lot in common. In some cases, we can even use pointers in place of arrays. But even though they are so closely related, they are still different...

3 min read

Properties of Array in C

An array in C is a fixed-size homogeneous collection of elements stored at a contiguous memory location. It is a derived data type in C that can store elements of different data types such as int, char, struct, etc. I...

8 min read

Array of Strings in C

In C, an array of strings is a 2D array where each row contains a sequence of characters terminated by a '\0' NULL character (strings). It is used to store multiple strings in a single array. Let's take a look at an...

3 min read

Applications of Pointers in C

Pointers in C are variables that are used to store the memory address of another variable. Pointers allow us to efficiently manage the memory and hence optimize our program. In this article, we will discuss...

4 min read

Size of Pointers in C

As pointers in C store the memory addresses, their size is independent of the type of data they are pointing to. This size of pointers in C only depends on the system architecture. That is why it is equal for...

3 min read

Is an Array Name a Pointer?

In C, arrays and pointers are closely related and are often considered same by many people but this a common misconception. Array names are not a pointer. It is actually a label that refers to a contiguous...

4 min read

Maximum Size of an Array in C

Array in C is a collection of elements of the same data type that are stored in contiguous memory locations. The size of an array is determined by the number of elements it can store and there is a limit o...

4 min read

Features and Use of Pointers in C/C++

Pointers store the address of variables or a memory location. Syntax: datatype *var_name; Example: pointer "ptr" holds the address of an integer variable or holds the address of memory whose value(s) ca...

2 min read

Pointer to an Array | Array Pointer

A pointer to an array is a pointer that points to the whole array instead of the first element of the array. It considers the whole array as a single unit instead of it being a collection of given elements. Example:...

5 min read