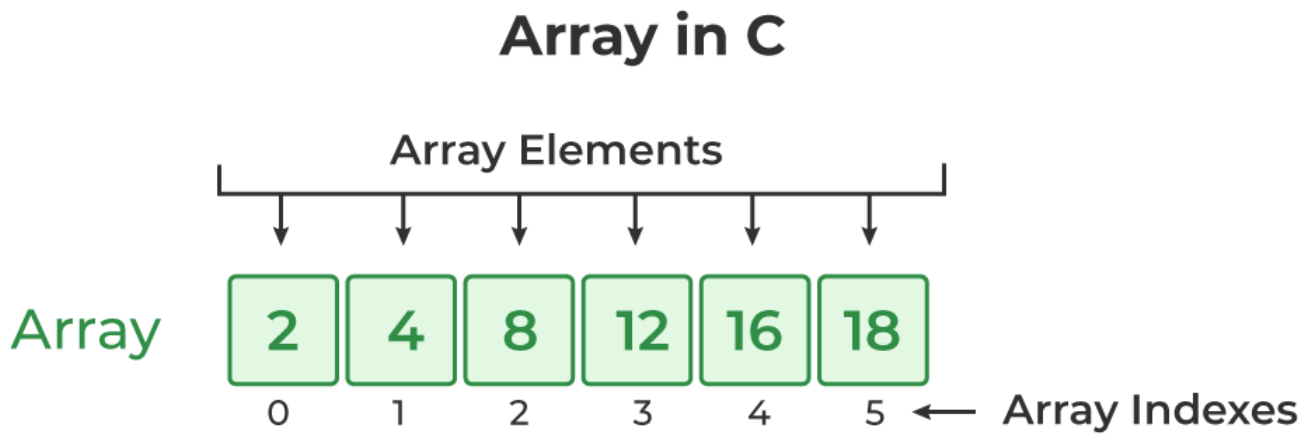
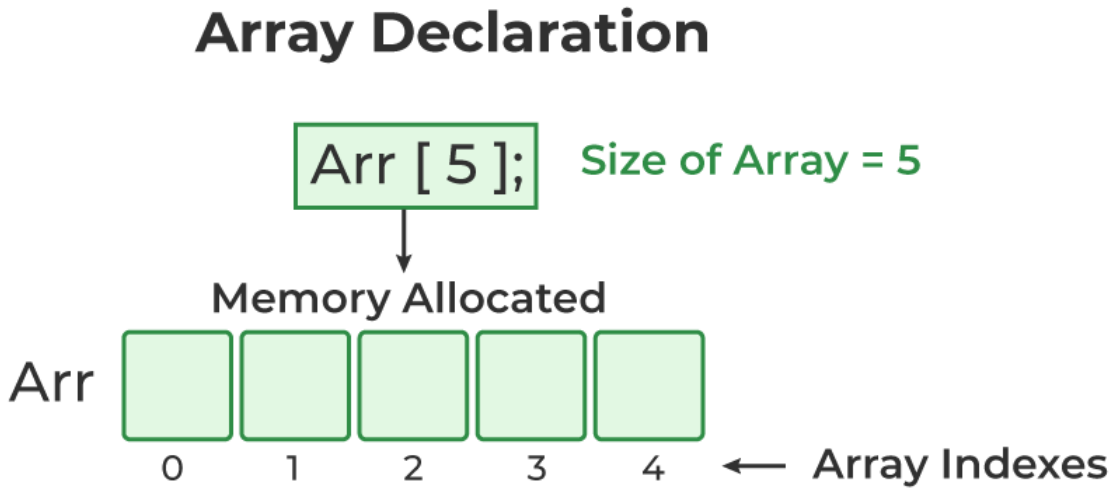
Arrays in C



In C programming, an array is a collection of elements of the same data type stored in contiguous memory locations. Arrays provide a convenient way to manage multiple related variables under a single name, allowing for efficient data manipulation and access.

Declaration and Initialization

To declare an array in C, specify the data type of its elements, the array name, and the number of elements (size) it will hold.

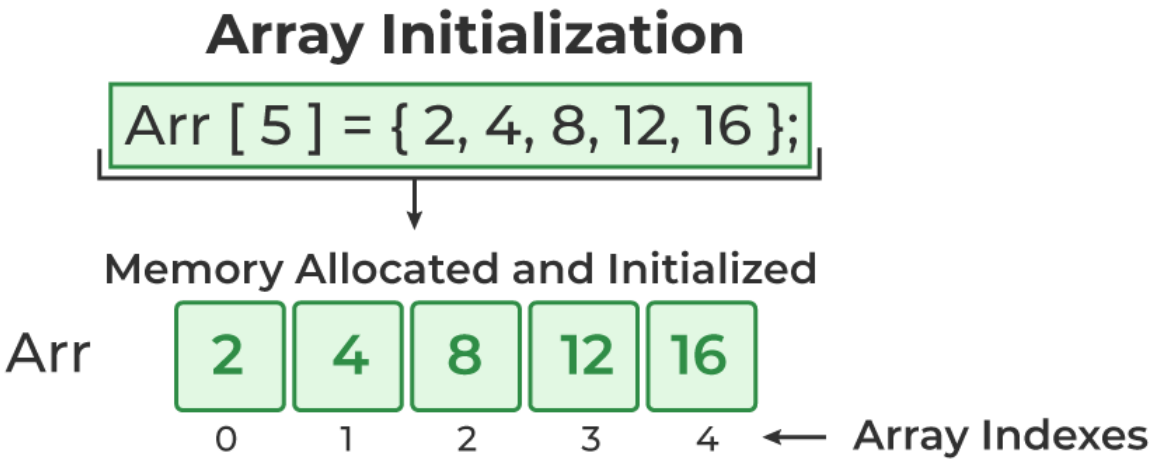


Syntax:

```
data_type array_name[array_size];
```

Example:

```
int numbers[5]; // Declares an array of 5 integers
```



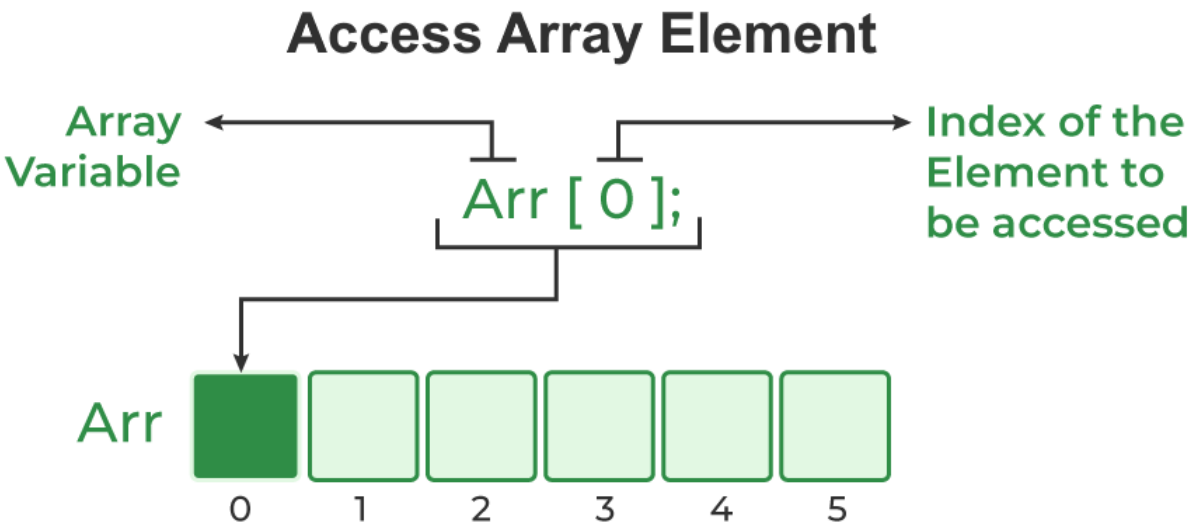
Arrays can be initialized at the time of declaration:

```
int numbers[5] = {1, 2, 3, 4, 5};
int cars[5];
```

If the array size is omitted, the compiler determines it based on the number of initializers:

```
int numbers[] = {1, 2, 3, 4, 5}; // Compiler sets array size to 5
```

Accessing Array Elements



Array elements are accessed using their index, starting from 0 up to `array_size - 1`.

Example:

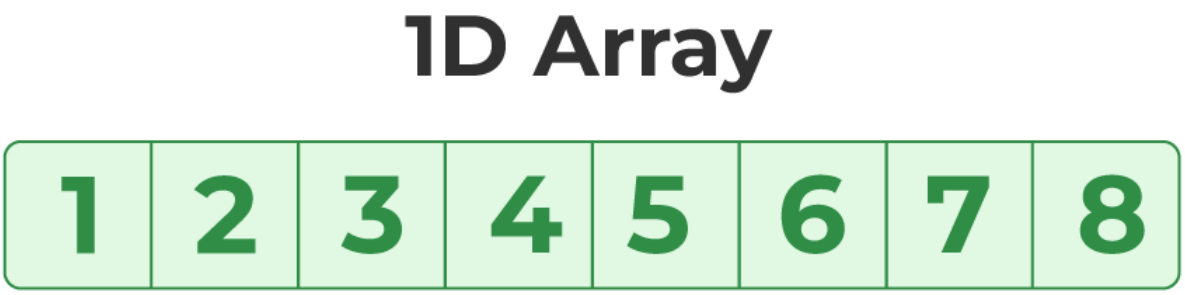
```
#include <stdio.h>

int main() {
    int numbers[] = {10, 20, 30, 40, 50};
    printf("%d\n", numbers[2]); // Outputs: 30
    return 0;
}
```

In this example, `numbers[2]` accesses the third element of the array, which is `30`.

Types of Arrays

One-Dimensional Arrays



A one-dimensional array is a linear collection of elements.

Declaration:

```
data_type array_name[size];
```

Example:

```
float temperatures[7]; // Array to store temperatures for a week
```

Multidimensional Arrays

C supports multidimensional arrays, commonly used for matrices or tables. The most common is the two-dimensional array.

Declaration of a 2D Array:

```
data_type array_name[rows][columns];
```

Example:

```
int matrix[3][4]; // 2D array with 3 rows and 4 columns
```

Initialization:

```
int matrix[3][4] = {
   {1, 2, 3, 4},
   {5, 6, 7, 8},
   {9, 10, 11, 12}
};
```

Accessing Elements:

```
int value = matrix[1][2]; // Accesses element at second row, third column (value is 7)
```

Advantages of Arrays

- **Efficient Data Management** --> Arrays allow for efficient storage and retrieval of multiple elements using a single identifier.
- **Random Access** --> Elements can be accessed directly using their index, enabling quick data retrieval.
- **Memory Efficiency** --> Storing elements in contiguous memory locations reduces memory overhead.

Limitations of Arrays

- **Fixed Size** --> Once declared, the size of an array cannot be changed during runtime.
- **Homogeneous Elements** --> Arrays can only store elements of the same data type.
- **Lack of Boundary Checking** --> C does not perform automatic bounds checking, which can lead to undefined behavior if indices are accessed out of range.

Relationship Between Arrays and Pointers

In C, the name of an array acts as a pointer to its first element. This means that `array_name` is equivalent to `&array_name[0]`. However, there are differences between arrays and pointers, especially in terms of memory allocation and how they are used in expressions.

Example:

```
int numbers[] = {10, 20, 30};
int *ptr = numbers; // ptr now points to the first element of numbers
```

Here, `ptr` is a pointer to an integer, and it points to the first element of the `numbers` array.

Revision #2
Created 11 February 2025 23:41:48 by YP
Updated 12 February 2025 00:07:04 by YP