Introduction to Arrays in C

Arrays are a fundamental data structure in C that allow us to store multiple elements of the same data type in a contiguous block of memory. They provide an efficient way to organize and manipulate data, especially when working with collections of data that can be referenced through a single variable.

An array in C is declared by specifying its type, name, and the number of elements it can hold. For example:

int numbers[5];

Here, numbers is an array that can hold five integer values.

**Types of Arrays**

There are mainly two types of arrays in C:

* **One-dimensional Arrays**: Used to store a linear collection of elements.
* **Multi-dimensional Arrays**: Used to store tabular data. The most common is a two-dimensional array, which resembles a matrix.

**Declaring and Initializing Arrays**

**Declaration**

An array can be declared by specifying its type and size:

int arr[10];

**Initialization**

An array can be initialized at the time of declaration:

int arr[5] = {1, 2, 3, 4, 5};

If fewer values are provided, the remaining elements are set to zero by default.

**Accessing Array Elements**

Array elements are accessed using indices, with the first element at index 0. For example, to access the third element of an array arr, we use:

arr[2];

**Basic Operations on Arrays**

* **Inputting Data**: Reading elements into the array.
* **Displaying Elements**: Printing all elements of the array.
* **Modifying Elements**: Updating values of specific elements.
* **Searching**: Finding a specific element in the array.
* **Sorting**: Arranging elements in a particular order, like ascending or descending.

**Examples:**

1. **Write a C program that copies the elements from one array to another and prints the new array.**

*#include* <stdio.h>

*int* *main*() {

*int* *arr*[10] *=* { 1,2,3,4,5,6,7,8,9,10 };

*int* *new*[10];

*for* (*int* *i* *=* 0;*i* *<* 10;*i++*) {

*new*[*i*] *=* *arr*[*i*];

    }

*for* (*int* *i* *=* 0;*i* *<* 10;*i++*) {

*printf*("%d ", *new*[*i*]);

    }

*return* 0;

}

**Output:**

* 1. 2 3 4 5 6 7 8 9 10

1. **Write a C program that takes 10 integers as input and prints them in reverse order**

*#include*<stdio.h>

*int* *main*() {

*int* *arr*[10];

*int* *i* *=* 0;

*while* (*i++* *<* 10) {

*scanf*("%d", *&arr*[*i*]);

    }

*printf*("reverse order:: ");

*while* (*i--* *!=* *-*1) {

*printf*("%d ", *arr*[*i*]);

    }

*return* 0;

}

**Output:**

**Reverse order:: 10 9 8 7 6 5 4 3 2 1**

1. **Write a C program that continuously adds integers to an array, updates the sum of the integers entered, and displays the updated sum after each input. The program stops when 0 is entered.**

*#include* <stdio.h>

*int* *main*() {

*int* *arr*[50];

*int* *i* *=* 0,

*sum* *=* 0;

*printf*("ADD: ");

*while* ((*scanf*("%d", *&arr*[*i++*])) *&&* *arr*[*i* *-* 1] *!=* 0) {

*printf*("sum:: %d\n", *sum* *+=* *arr*[*i* *-* 1]);

*printf*("ADD: ");

    }

*return* 0;

}

**Output:**

**ADD: 5**

**sum:: 5**

**ADD: 10**

**sum:: 15**

**ADD: -3**

**sum:: 12**

**ADD: 0**

1. **Write a C program that takes 10 integers as input, calculates the sum, and then prints the average of the entered numbers.**

*#include* <stdio.h>

*int* *main*() {

*int* *arr*[10];

*int* *sum* *=* 0;

*for* (*int* *i* *=* 0;*i* *<* 10;*i++*) {

*scanf*("%d", *&arr*[*i*]);

*sum* *+=* *arr*[*i*];

    }

*printf*("avg is:: %.2f\n", (*float*)*sum/*10.0);

*return* 0;

}

**Output:**

**1 2 3 4 5 6 7 8 9 10**

**avg is:: 5.50**

1. **Write a C program that uses nested loops to print the values of i and j for each iteration where both i and j range from 0 to 2.**

*#include* <stdio.h>

*int* *main*() {

*// Your code goes here*

*for* (*int* *i* *=* 0;*i* *<* 3;*i++*) {

*for* (*int* *j* *=* 0;*j* *<* 3;*j++*) {

*printf*("%d %d\n", *i*, *j*);

        }

    }

*return* 0;

}

**Output:**

**0 0**

**0 1**

**0 2**

**1 0**

**1 1**

**1 2**

**2 0**

**2 1**

**2 2**

**Advantages and Limitations of Arrays**

**Advantages**

* Efficiently stores multiple elements of the same type.
* Allows easy access to elements using indexing.

**Limitations**

* Fixed size, so the number of elements cannot change after declaration.
* Only stores elements of the same data type.
* Lack of built-in functions for complex operations like insertion and deletion.

**Conclusion**

Through this assignment, you will gain hands-on experience with arrays in C, understanding their capabilities, limitations, and how they can be used in various applications.