Programming Fundamentals with C++

Lecture 13 – Arrays

Dr. Muhammad Sajjad

RA: Asad Ullah

Overview

- **char Array**
  - What is a Char Array?
  - Declaring and Initializing Char Arrays
  - Why is Null Character Important?
  - Special Features of Char Arrays
  - Code Example
- **Two-Dimensional Arrays**
  - Introduction to 2D Arrays
  - Syntax of 2D Arrays
  - Declaration & Initialization
  - Accessing & Modifying Elements
  - Iterating Over a 2D Array
  - Code Example

char Array

What is a Char Array?

- A char array is an array where each element is of type char.
- It is commonly used to store strings, like words or sentences.

```
char name[5] = {'A', 'l', 'i', '\0'};
```

- Here:
  - name is a char array storing the characters **A**, **l**, **i**.
  - The **\0** is a null character marking the end of the string.

char Array

Declaring and Initializing Char Arrays

Two ways to declare and initialize a char array:

1. **Character by Character**:

```
char word[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

   Here, the \0 (null terminator) tells the compiler the end of the array.

2. **String Literal**:

```
char word[] = "Hello";
```

   The compiler automatically adds the null terminator when a string literal is used.

char Array

Why is Null Character Important?

- Without the \0, the program cannot determine the end of the string in the array.
- A practical example to highlight this:

```
#include <iostream>
using namespace std;

int main() {
    char name[5] = {'A', 'l', 'i'};
    cout << "Name: " << name << endl; // May print garbage
characters after Ali
    return 0;
}
```

Without \0, extra characters (garbage values) may be displayed.

char Array

Special Features of Char Arrays

1. **Input and Output with Char Arrays**:
   - How to take input using cin:

```
char name[20];
cout << "Enter your name: ";
cin >> name;
cout << "Your name is: " << name << endl;
```

2. **Using gets for Full Line Input:**
   - To read a full sentence (including spaces), use:

```
#include <iostream>
#include <cstring> // For string functions
using namespace std;

int main() {
    char sentence[100];
    cout << "Enter a sentence: ";
    cin.ignore(); // Ignore leftover input
    cin.getline(sentence, 100); // Reads a full
line
    cout << "You entered: " << sentence << endl;
    return 0;
}
```

3. **String Manipulations with Char Arrays**:
   - Introduce basic string manipulation functions:
     - **strlen():** Find length of a string.
     - **strcpy():** Copy one string to another.
     - **strcmp():** Compare two strings.

char Array

Code Example

```
#include <iostream>
#include <cstring>
using namespace std;

int main() {
    char username[20];
    char correctUsername[] = "student";

    cout << "Enter your username: ";
    cin >> username;

    if (strcmp(username, correctUsername) == 0) {
        cout << "Welcome, " << username << "!" << endl;
    } else {
        cout << "Incorrect username!" << endl;
    }

    return 0;
}
```

Two-Dimensional Arrays

Introduction to 2D Arrays

- A **2D array** (two-dimensional array) can be thought of as an array of arrays. It stores data in rows and columns, similar to a table or matrix. This is useful when we need to represent data that has a grid-like structure.
- For example, if you wanted to store the marks of 5 students in 3 subjects, you can represent this data in a 2D array where each row represents a student, and each column represents a subject.

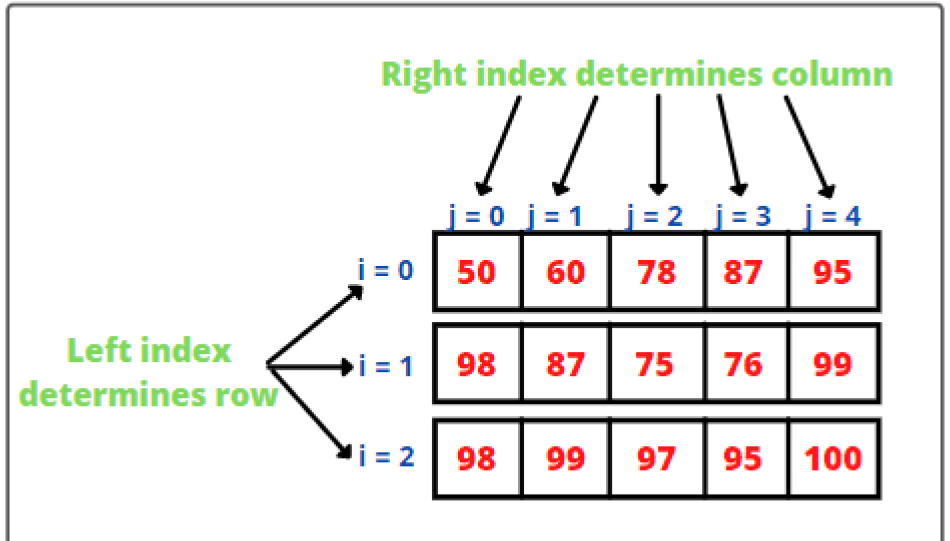


Fig: Arrangement of elements in two dimensional array

Two-Dimensional Arrays

Syntax of 2D Arrays

The syntax to declare a 2D array in C++ is:

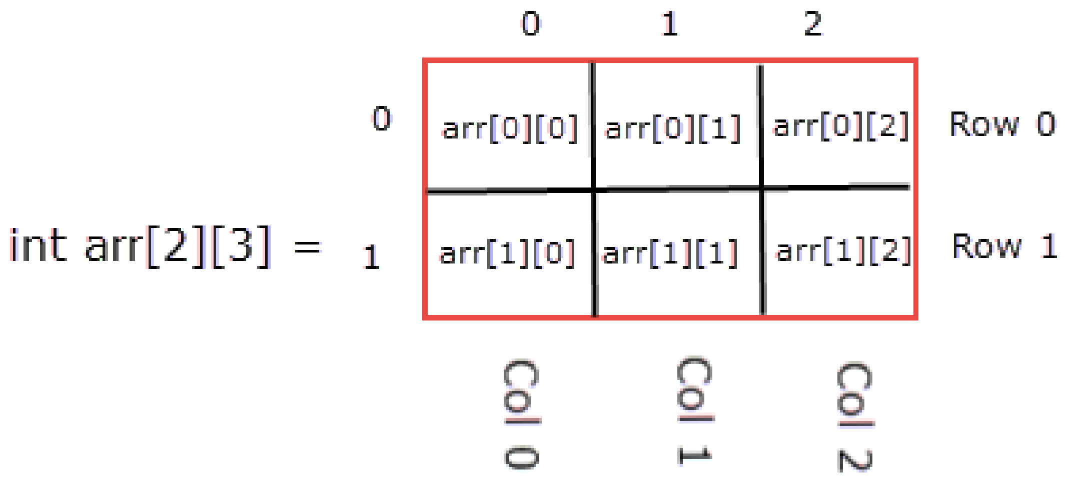
```
data_type  array_name [row_size] [column_size];
```

data_type: The type of data the array holds (e.g., int, float, char).

array_name: The name of the array.

row_size: The number of rows.

column_size: The number of columns.



Two-Dimensional Arrays

Declaration & Initialization

A 2D array can be initialized in several ways, either by specifying values directly or by using loops to assign values.

1. **Direct Initialization**

```
int marks[2][3] = {
    {90, 85, 88},  // First row
    {76, 82, 91}   // Second row
};
```

2. **Using a Loop for Initialization**

```
int marks[2][3];
for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 3; j++) {
        marks[i][j] = i * j;  // Example of assigning values
    }
}
```

Two-Dimensional Arrays

Accessing & Modifying Elements

To access or modify elements in a 2D array, you use the indices of the array. The first index refers to the row, and the second index refers to the column.

```
#include <iostream>
using namespace std;
int main() {
    int marks[2][3] = {
        {90, 85, 88},
        {76, 82, 91}
    };
    // Accessing an element
    cout << "Marks of first student in first subject: " << marks[0][0] << endl;  // Output: 90

    // Modifying an element
    marks[1][2] = 95;  // Change marks of second student in third subject
    cout << "Modified marks of second student in third subject: " << marks[1][2] << endl;  // Output: 95
    return 0;
}
```

Two-Dimensional Arrays

Iterating Over a 2D Array

To print all elements of a 2D array, you need two loops: one for rows and one for columns.

```
#include <iostream>
using namespace std;

int main() {
    int marks[2][3] = {
        {90, 85, 88},
        {76, 82, 91}
    };

    // Iterating over the 2D array
    for (int i = 0; i < 2; i++) {  // Loop over rows
        for (int j = 0; j < 3; j++) {  // Loop over columns
            cout << "Marks at [" << i << "][" << j << "] = " << marks[i][j] << endl;
        }
    }
    return 0;
}
```

Arrays in C++

Code Example

```
#include <iostream>
using namespace std;
int main()
{
		int count = 1;

		// Declaring 2D array
		int array1[3][4];

		// Initialize 2D array using loop
		for (int i = 0; i < 3; i++) {
			for (int j = 0; j < 4; j++) {
				array1[i][j] = count;
				count++;
			}
		}
		// Printing the element of 2D array
		for (int i = 0; i < 3; i++) {
			for (int j = 0; j < 4; j++) {
				cout << array1[i][j] << " ";
			}
			cout << endl;
		}
		return 0;
}
```

Thank You