**LAB: 3**

**Subject: Data Structures**

**Topic: Implementation of Multi-Dimensional Array**

**Definition:**

A **multidimensional array** is an array of arrays, where each element is itself an array. In a **2D array**, elements are arranged in rows and columns, forming a matrix-like structure. This allows you to store data in a tabular form, making it ideal for scenarios like storing matrices, tables, or grids.

**Example 1: 2D Array Declaration and Initialization**

#include <iostream>

using namespace std;

int main() {

// Declare and initialize a 2x3 array (2 rows, 3 columns)

int matrix[2][3] = {

{1, 2, 3}, // First row

{4, 5, 6} // Second row

};

// Print the 2D array

for (int i = 0; i < 2; i++) {

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

cout << matrix[i][j] << " "; // Access elements using row and column indices

}

cout << endl; // Newline after each row

}

return 0;

}

**Example 2: 2D Array for Storing Student Marks**

#include <iostream>

using namespace std;

int main() {

// Declare and initialize a 3x2 array to store marks for 3 students in 2 subjects

int marks[3][2] = {

{85, 90}, // Marks for Student 1 in Subject 1 and Subject 2

{78, 82}, // Marks for Student 2 in Subject 1 and Subject 2

{92, 88} // Marks for Student 3 in Subject 1 and Subject 2

};

// Display the marks

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

cout << "Student " << i+1 << " Marks: ";

for (int j = 0; j < 2; j++) {

cout << marks[i][j] << " "; // Print each student's marks

}

cout << endl;

}

return 0;

}

**Example: 3**

#include <iostream>

using namespace std;

int main() {

// Declare and initialize a 3x2 array to store marks for 3 students in 2 subjects

int marks[6][2] = {

{85, 90}, // Marks for Student 1 in Subject 1 and Subject 2

{78, 82}, // Marks for Student 2 in Subject 1 and Subject 2

{92, 88}, // Marks for Student 3 in Subject 1 and Subject 2

{56, 33},

{66, 99},

{89, 88},

};

// Display the marks

for (int i = 0; i < 6; i++) {

cout << "Student " << i+1 << " Marks: ";

for (int j = 0; j < 2; j++) {

cout << marks[i][j] << " "; // Print each student's marks

}

cout << endl;

}

return 0;

}

### Key Features of Vectors:

1. **Dynamic Sizing**: Unlike static arrays, vectors can automatically resize themselves when elements are added or removed.
2. **Random Access**: Elements in a vector can be accessed using an index, similar to arrays.
3. **Memory Management**: Vectors handle memory allocation and deallocation automatically, reducing the risk of memory leaks.
4. **Flexibility**: Vectors can store any data type, including user-defined types, and can be nested (i.e., vectors of vectors).

**Example: 4**

#include <iostream>

#include <vector> // Include the vector header

using namespace std;

int main() {

// Create a vector to store integers

vector<int> numbers;

// Add elements to the vector

numbers.push\_back(10); // Add 10

numbers.push\_back(20); // Add 20

numbers.push\_back(30); // Add 30

// Display the elements of the vector

cout << "Vector elements: ";

for (int i = 0; i < numbers.size(); i++) { // Use size() to get the number of elements

cout << numbers[i] << " "; // Access elements using the index

}

cout << endl;

// Remove the last element

numbers.pop\_back(); // Removes 30

// Display the updated vector

cout << "After pop\_back, elements: ";

for (int i = 0; i < numbers.size(); i++) {

cout << numbers[i] << " "; // Print updated vector

}

cout << endl;

return 0;

}

**Explanation:**

* The program begins by creating a vector named numbers to store integers.
* Elements are added to the vector using the push\_back() function.
* The size() function is used to determine the number of elements in the vector.
* The pop\_back() function removes the last element from the vector.
* Finally, the updated vector is displayed.

**Example: 5**

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<string> cars = {"Volvo", "BMW", "Ford", "Mazda"};

// Change the value of the first element

cars[0] = "Corolla";

cout << cars[0];

return 0;

}

**Example: 6**

In C++, the .at() function is a member function of the std::vector class that provides access to the elements of the vector. It allows you to retrieve or modify an element at a specific index while performing bounds checking.

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<string> cars = {"Volvo", "BMW", "Ford", "Mazda"};

// Change the value of the first element

cars.at(0) = "Opel";

cout << cars.at(0);

return 0;

}

## Add Vector Elements

The biggest difference between a vector and an array is that vectors can grow dynamically. That means you can add or remove elements from the vector.

To add an element to the vector, you can use the .push\_back() function, which will add an element at the end of the vector:

**Example: 7**

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<string> cars = {"Volvo", "BMW", "Ford", "Mazda"};

cars.push\_back("Tesla");

for (string car : cars) {

cout << car << "\n";

}

return 0;

}

**You can add as many elements as you want:**

**Example: 8**

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<string> cars = {"Volvo", "BMW", "Ford", "Mazda"};

cars.push\_back("Tesla");

cars.push\_back("VW");

cars.push\_back("Mitsubishi");

cars.push\_back("Mini");

for (string car : cars) {

cout << car << "\n";

}

return 0;

}

## Remove Vector Elements

To remove an element from the vector, you can use the .pop\_back() function, which removes an element from the end of the vector:

**Example: 9**

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<string> cars = {"Volvo", "BMW", "Ford", "Mazda"};

cars.pop\_back();

for (string car : cars) {

cout << car << "\n";

}

return 0;

}

## Vector Size

To find out how many elements a vector has, use the .size() function:

**Example: 10**

#include <iostream> // Include the standard input-output library

#include <vector> // Include the vector library to use the vector data structure

using namespace std; // Use the standard namespace for convenience

int main() {

// Declare and initialize a vector of strings with car brand names

vector<string> cars = {"Volvo", "BMW", "Ford", "Mazda"};

// Output the size of the vector using the .size() function

// This function returns the number of elements currently stored in the vector

cout << cars.size();

return 0; // Return 0 to indicate successful execution of the program

}

**Explanation:**

* **#include <vector>**: This includes the vector library, allowing you to use the std::vector class.
* **vector<string> cars = {...};**: Initializes a vector of strings called cars with four elements.
* **cars.size()**: The .size() function returns the number of elements in the vector, which in this case is 4.
* **cout << cars.size();**: Prints the size of the vector (4) to the console.