C++ Arrays

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To declare an array, define the variable type, specify the name of the array followed by **square brackets** and specify the number of elements it should store:

string cars[4];

We have now declared a variable that holds an array of four strings. To insert values to it, we can use an array literal - place the values in a comma- separated list, inside curly braces:

string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};

To create an array of three integers, you could write:

int myNum[3] = {10, 20, 30};

Access the Elements of an Array

You access an array element by referring to the index number. This statement accesses the value of the **first element** in **cars**:

Example

string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};

cout << cars[0];

// Outputs Volvo

**Note:** Array indexes start with 0: [0] is the first element. [1] is the second element, etc.

Change an Array Element

To change the value of a specific element, refer to the index number:

# Example

cars[0] = "Opel";

# Example

string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"}; cars[0] = "Opel";

cout << cars[0];

// Now outputs Opel instead of Volvo

Types of Array:-

1. single dimensional array:-

An array is a collection of data that holds homogeneous values. That means values should be in same type

Syntax:- type variable\_name[size] Ex:- int varName[10];

Array values stores like below structure,

int value[6] = {5,10,15,20,25,30};

Value[0] = 5

Value[1] : 10

Value[2] : 15

Value[3] : 20

Value[4] : 25

Value[5] : 30

*/\* Example Program For Single Dimensional Array In C++ Programming Language Array Example In C++\*/*

*// Header Files* **#include <iostream.h> #include<conio.h>**

int **main**()

{

int i;

*// declaring and Initialising array in C*

int value[6] = {5,10,15,20,25,30};

cout<<"Single Dimensional Array In C++ Example Program\n";

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

{

*// Accessing each variable using for loop*

cout<<"Position : "<<i<<" , Value : "<< value[i]<<" \n";

}

*// Wait For Output Screen*

getch();

*//Main Function return Statement*

return 0;

}

Output:-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Position : | [0] | , | Value : | 5 |
| Position : | [1] | , | Value : | 10 |
| Position : | [2] | , | Value : | 15 |
| Position : | [3] | , | Value : | 20 |
| Position : | [4] | , | Value : | 25 |
| Position : | [5] | , | Value : | 30 |

1. Multi dimensional array:-

|  |
| --- |
| C++ Multidimensional Arrays***In this article, you'll learn about multi-dimensional arrays in C++. More specifically, how to declare them, access them and use them efficiently in your program.***In C++, you can create an [array](https://www.programiz.com/cpp-programming/arrays) of an array known as multi-dimensional array. For example: |
| int x[3][4]; |

Here, is a two dimensional array. It can hold a maximum of 12 elements.

You can think this array as table with 3 rows and each row has 4 columns as shown below.

Three dimensional array also works in a similar way. For example:

x

|  |
| --- |
| float x[2][4][3]; |

This array can hold a maximum of 24 elements. You can think this example as: Each of the 2 elements can hold 4 elements, which makes 8 elements and each of

x

**Multidimensional Array Initialisation**

You can initialise a multidimensional array in more than one way.

**Initialisation of two dimensional array**

those 8 elements can hold 3 elements. Hence, total number of elements this array can hold is 24.

|  |
| --- |
| int test[2][3] = {2, 4, -5, 9, 0, 9}; |

Better way to initialise this array with same array elements as above.

|  |
| --- |
| int test[2][3] = { {2, 4, 5}, {9, 0 0}}; |

**Initialisation of three dimensional array**

|  |
| --- |
| int test[2][3][4] = {3, 4, 2, 3, 0, -3, 9, 11, 23, 12, 23,2, 13, 4, 56, 3, 5, 9, 3, 5, 5, 1, 4, 9}; |

Better way to initialise this array with same elements as above.

|  |
| --- |
| int test[2][3][4] = {{ {3, 4, 2, 3}, {0, -3, 9, 11}, {23, 12, 23, 2} },{ {13, 4, 56, 3}, {5, 9, 3, 5}, {3, 1, 4, 9} }}; |

|  |
| --- |
| **Example 1: Two Dimensional Array****C++ Program to display all elements of an initialised two dimensional array.** |
| #include <iostream> #include <conio.h>int main(){int test[3][2] ={{2, -5},{4, 0},{9, 1}};// Accessing two dimensional array using// nested for loops for(int i = 0; i < 3; ++i){for(int j = 0; j < 2; ++j){cout<< "test[" << i << "][" << j << "] = " << test[i][j] << endl;}}return 0;} |
| **Output** |
| test[0][0] = 2test[0][1] = -5test[1][0] = 4test[1][1] = 0test[2][0] = 9test[2][1] = 1 |
|  |

|  |
| --- |
| **Example 2: Two Dimensional Array****C++ Program to store temperature of two different cities for a week and display it.** |
| #include <iostream> #include <conio.h>const int CITY = 2; const int WEEK = 7;int main(){int temperature[CITY][WEEK];cout << "Enter all temperature for a week of first city and then second city. \n";// Inserting the values into the temperature array for (int i = 0; i < CITY; ++i){for(int j = 0; j < WEEK; ++j){cout << "City " << i + 1 << ", Day " << j + 1 << " : "; cin >> temperature[i][j];}}cout << "\n\nDisplaying Values:\n";// Accessing the values from the temperature array for (int i = 0; i < CITY; ++i){for(int j = 0; j < WEEK; ++j){cout << "City " << i + 1 << ", Day " << j + 1 << " = " <<temperature[i][j] << endl;}}return 0;} |

|  |
| --- |
| **Output** |
| Enter all temperature | for | a | week | of | first | city | and | then | second | city. |
| City 1, Day 1 : 32 |  |  |  |  |  |  |  |  |  |  |
| City 1, Day 2 : 33 |  |  |  |  |  |  |  |  |  |  |
| City 1, Day 3 : 32 |  |  |  |  |  |  |  |  |  |  |
| City 1, Day 4 : 34 |  |  |  |  |  |  |  |  |  |  |
| City 1, Day 5 : 35 |  |  |  |  |  |  |  |  |  |  |
| City 1, Day 6 : 36 |  |  |  |  |  |  |  |  |  |  |
| City 1, Day 7 : 38 |  |  |  |  |  |  |  |  |  |  |
| City 2, Day 1 : 23 |  |  |  |  |  |  |  |  |  |  |
| City 2, Day 2 : 24 |  |  |  |  |  |  |  |  |  |  |
| City 2, Day 3 : 26 |  |  |  |  |  |  |  |  |  |  |
| City 2, Day 4 : 22 |  |  |  |  |  |  |  |  |  |  |
| City 2, Day 5 : 29 |  |  |  |  |  |  |  |  |  |  |
| City 2, Day 6 : 27 |  |  |  |  |  |  |  |  |  |  |
| City 2, Day 7 : 23 |  |  |  |  |  |  |  |  |  |  |
| Displaying Values: |  |  |  |  |  |  |  |  |  |  |
| City 1, Day 1 = 32 |  |  |  |  |  |  |  |  |  |  |
| City 1, Day 2 = 33 |  |  |  |  |  |  |  |  |  |  |
| City 1, Day 3 = 32 |  |  |  |  |  |  |  |  |  |  |
| City 1, Day 4 = 34 |  |  |  |  |  |  |  |  |  |  |
| City 1, Day 5 = 35 |  |  |  |  |  |  |  |  |  |  |
| City 1, Day 6 = 36 |  |  |  |  |  |  |  |  |  |  |
| City 1, Day 7 = 38 |  |  |  |  |  |  |  |  |  |  |
| City 2, Day 1 = 23 |  |  |  |  |  |  |  |  |  |  |
| City 2, Day 2 = 24 |  |  |  |  |  |  |  |  |  |  |
| City 2, Day 3 = 26 |  |  |  |  |  |  |  |  |  |  |
| City 2, Day 4 = 22 |  |  |  |  |  |  |  |  |  |  |
| City 2, Day 5 = 29 |  |  |  |  |  |  |  |  |  |  |
| City 2, Day 6 = 27 |  |  |  |  |  |  |  |  |  |  |
| City 2, Day 7 = 23 |  |  |  |  |  |  |  |  |  |  |

**Example 3: Three Dimensional Array**

|  |
| --- |
| **C++ Program to Store value entered by user in three dimensional array and display it.** |
| #include <iostream.h> #include <conio.h>int main(){// This array can store upto 12 elements (2x3x2) int test[2][3][2];cout << "Enter 12 values: \n";// Inserting the values into the test array// using 3 nested for loops. for(int i = 0; i < 2; ++i){for (int j = 0; j < 3; ++j){for(int k = 0; k < 2; ++k ){cin >> test[i][j][k];}}}cout<<"\nDisplaying Value stored:"<<endl;// Displaying the values with proper index. for(int i = 0; i < 2; ++i){for (int j = 0; j < 3; ++j){for(int k = 0; k < 2; ++k){cout << "test[" << i << "][" << j << "][" << k << "] = " <<test[i][j][k] << endl;}}}return 0;} |

|  |
| --- |
| **Output** |
| Enter 12 values:123456789101112Displaying Value stored: test[0][0][0] = 1test[0][0][1] = 2test[0][1][0] = 3test[0][1][1] = 4test[0][2][0] = 5test[0][2][1] = 6test[1][0][0] = 7test[1][0][1] = 8test[1][1][0] = 9test[1][1][1] = 10test[1][2][0] = 11test[1][2][1] = 12 |
| As the number of dimension increases, the complexity also increases tremendously although the concept is quite similar. |