Arrays, String, Pointers & Reference

INDEX

Table of Contents

[Arrays in C++ 2](#_Toc142931921)

[Array declaration 2](#_Toc142931922)

[Multidimensional Array 3](#_Toc142931923)

[Memory Layout 4](#_Toc142931924)

[Strings in C++ 5](#_Toc142931925)

[C-style (character arrays and literals) 5](#_Toc142931926)

[String Class in C++ STL 6](#_Toc142931927)

[String manipulations 7](#_Toc142931928)

# Arrays in C++

Array is a collection of similar types of data elements, stored at continuous memory locations. It is a built-in data type.

A colorful rectangular object with numbers and symbols

Description automatically generated

## Array declaration

by specifying size: int arr[10]

by initializing elements: int arr[] = { 10, 20, 30, 40 }

## Multidimensional Array

A black background with white text

Description automatically generated

For example - A 2D array defined as follows can be viewed as a table of two rows and three columns.



An N-dimensional array is an array of arrays. It will be mapped to one-dimensional memory addresses.



## Memory Layout

A contiguous memory space is allocated for array elements and can be accessed via an array index.

A diagram of a number of numbers

Description automatically generated with medium confidence

# Strings in C++

There are mainly 2 ways of handling strings in C++.

1. **C-style** strings or character arrays.
2. string template class

## C-style (character arrays and literals)

A screenshot of a computer program

Description automatically generated

## String Class in C++ STL

A screen shot of a computer

Description automatically generated

### String manipulations

**Remember**

An **iterator** is an object (like a pointer) that points to an element inside the container. Containers are



Each container class in the C++ Standard Library provides its own specific type of iterator. Few Examples:

|  |  |
| --- | --- |
| **vector<int>::iterator** | **list<double>::iterator** |
| **set<std::string>::iterator** | **map<int, std::string>::iterator** |

Available functions in the std::string class which can be used for string manipulations.

**String Iteration**

****

OP



**String Length and Capacity**

Size: size refers to the number of characters in the string.

Capacity:

* The capacity of a container refers to the amount of memory that has been allocated for it, which determines how many elements it can hold efficiently.
* Capacity refers to the number of characters that the **string can hold without needing to reallocate memory**.
* When appending characters to a string, if the size exceeds the capacity, the string might need to be reallocated to accommodate the new characters.



OP

A black background with a white and blue light

Description automatically generated with medium confidence

**String Concatenation and Append**



OP

A black screen with white text

Description automatically generated