# Data Structures Lab 01

**Course:** Data Structures (CS2001) **Semester:** Spring 2024

**Instructor:** Bushra Sattar **T.A:** N/A

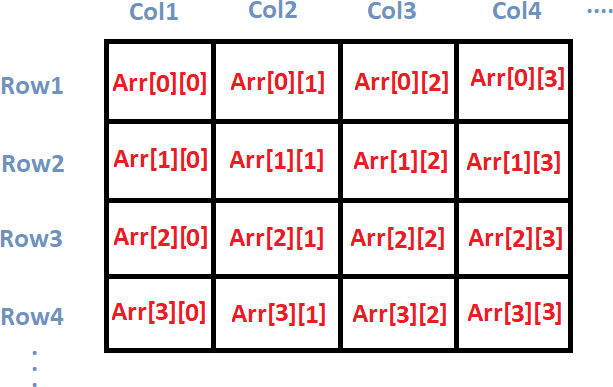
**Note:**

* Maintain discipline during the lab.
* Listen and follow the instructions as they are given.
* Just raise your hand if you have any problem.
* Completing all tasks of each lab is compulsory.
* Get your lab checked at the end of the session.

# 2D Arrays, Pointers and Double Pointers

**2D Arrays**

A two-dimensional array in C++ is the simplest form of a multi-dimensional array. It can be visualized as an array of arrays. The image below depicts a two-dimensional array.



A two-dimensional array is also called a matrix. It can be of any type like integer, character, float, etc. depending on the initialization.

# [Initializing a 2D array in C++](https://www.digitalocean.com/community/tutorials/two-dimensional-array-in-c-plus-plus#initializing-a-2d-array-in-c)

we initialize a 2D array arr, with 4 rows and 2 columns as an array of arrays. Each element of the array is yet again an array of integers.

int arr[4][2] = {

{12, 56},

{2, 33},

{3, 80},

{3, 78}

# [Printing a 2D Array in C++](https://www.digitalocean.com/community/tutorials/two-dimensional-array-in-c-plus-plus#printing-a-2d-array-in-c)

#include<iostream> using namespace std; main( )

{

int arr[4][2] = {

{ 10, 11 },

{ 20, 21 },

{ 30, 31 },

{ 40, 41 }

} ;

int i,j;

cout<<"Printing a 2D Array:\n"; for(i=0;i<4;i++)

{

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

{

cout<<"\t"<<arr[i][j];

}

cout<<endl;

}

}

**Output**

|  |  |
| --- | --- |
| 10 | 11 |
| 20 | 21 |
| 30 | 31 |
| 40 | 41 |

**What are Pointers?**

A pointer is a variable whose value is the address of another variable. Like any variable or constant, you must declare a pointer before you can work with it. The general form of a pointer variable declaration is: **Syntax: type \*var-name;**

Here, type is the pointer's base type; it must be a valid C++ type and var-name is the name of the pointer variable. The asterisk you used to declare a pointer is the same asterisk that you use for multiplication. However, in this statement the asterisk is being used to designate a variable as a pointer.

# Following are the valid pointer declaration

int \*ip; // pointer to an integer double \*dp; // pointer to a double float \*fp; // pointer to a float char \*ch // pointer to character

The actual data type of the value of all pointers, whether integer, float, character, or otherwise, is the same, a long hexadecimal number that represents a memory address. The only difference between pointers of different data types is the data type of the variable or constant that the pointer points to.

Using Pointers in C++.

# There are few important operations, which we will do with the pointers very frequently.

1. We define a pointer variable.
2. Assign the address of a variable to a pointer.
3. Finally access the value at the address available in the pointer variable. This is done by using unary operator \* that returns the value of the variable located at the address specified by its operand.

# Following example makes use of these operations

#Include <iostream> Using namespace std; Int main () {

int var = 20; // actual variable declaration. int \*ip; // pointer variable

ip = &var; // store address of var in pointer variable cout << "value of var variable: ";

cout << var << endl;

// print the address stored in ip pointer variable cout << "address stored in ip variable: ";

cout << ip << endl;

// access the value at the address available in pointer cout << "value of \*ip variable: ";

cout << \*ip << endl; return 0;

}

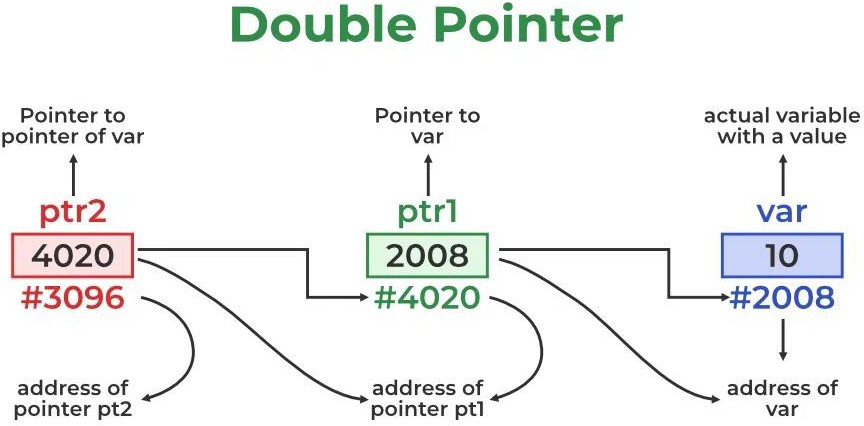
**Output**

Value of var variable: 20

Address stored in ip variable: 0xbfc601ac Value of \*ip variable: 20

**Double Pointers**

The pointer to a pointer in C is used when we want to store the address of another pointer. The first pointer is used to store the address of the variable. And the second pointer is used to store the address of the first pointer. That is why they are also known as *double-pointers*. We can use a pointer to a pointer to change the values of normal pointers or create a variable-sized 2-D array. A double pointer occupies the same amount of space in the memory stack as a normal pointer.



The above diagram shows the memory representation of a pointer to a pointer. The first pointer ptr1 stores the address of the variable and the second pointer ptr2 stores the address of the first pointer.

## Example:

**int var = 10;**

**int \*ptr1 = &var; // storing address of val to pointer ptr.**

**int \*\*ptr2 = &ptr1; // pointer to a pointer declared which is pointing to an integer.**

#include <iostream> using namespace std; int main()

{

int variable = 10;

// Pointer to store the address of variable int\* ptr1;

// double pointer to store the address of pointer1 int\*\* ptr2;

// Storing address of variable in pointer1 ptr1 = &var;

// Storing address of pointer1 in pointer2 ptr2 = &ptr1;

// Displaying the value of variable by using both single and double pointers. cout << "Value of variable :- " << var<< "\n";

cout << "Value of variable using single pointer :- " << \*ptr1 << "\n"; cout << "Value of variable using double pointer :- " << \*\*ptr2 << "\n"; return 0;

}

**Output**

Value of variable :- 10

Value of variable using single pointer :- 10 Value of variable using double pointer :- 10

# How to Declare a Pointer to a Pointer in C ++?

Declaring a Pointer to Pointer is similar to declaring a pointer in C++. The difference is we have to use an additional \* operator before the name of a Pointer in C++.

**Syntax of a Pointer to Pointer (Double Pointer) in C++:**

**data\_type\_of\_pointer \*\*name\_of\_variable = & normal\_pointer\_variable;**

# What will be the size of a pointer to a pointer in C++?

In the C++ programming language double pointer behave similarly to a normal pointer. So, the size of the variable of the double-pointer and the size of the normal pointer variable is always equal.

## Below is a C++ program to check the size of a double pointer:

#include <iostream> using namespace std;

// Driver code int main()

{

int val = 169; int\* ptr = &val;

int\*\* double\_ptr = &ptr;

cout << " Size of normal Pointer: " << sizeof(ptr) << "\n";

cout << " Size of double Pointer: " << sizeof(double\_ptr) << "\n"; return 0;

}

**Output**

Size of normal Pointer: 8 Size of double Pointer: 8

**Note:** The output of the above code also depends on the type of machine which is being used. The size of a pointer is not fixed in the C++ programming language and it totally depends on other factors like CPU architecture and OS used. Usually, for a 64-bit Operating System, a size of 8 bytes memory and for a 32-bit Operating system, a size of 4 bytes memory is assigned.

# Lab Tasks

1. Write a C++ program to perform a matrix addition, subtraction, multiplication and division by using two-dimensional array.

Print Matrix A and Matrix B by taking user input and then the resultant matrices for the following addition subtraction multiplication and division.

1. Write a C++ program that takes user input for an array of integers and performs the following tasks

* **Finding Positive and Negative Elements**

Write a recursive function to find and count the number of positive and negative elements in the given array. The program should display the count of positive and negative elements.

* **Finding the Maximum Element**

Implement a recursive function to find and return the maximum element in the array. Display the maximum element after the analysis is complete.

1. Write a C++ program that initializes an array by taking user input. Write a recursive function to determine if the given array of integers is sorted in ascending order. The program should display whether the array is sorted or not.
2. Developing a C++ program that takes user input for an array of non-negative integers and calculates the factorial of each element. For this Implement a function to take user input for the size of the array and its non-negative integer elements. Ensure that the array contains valid non-negative integer values. Write a recursive function named **calculateFactorials** to calculate the factorial of each element in the given array. The function should modify the array in-place to store the calculated factorials.

The **calculateFactorials** function should have parameters for the array, its size, and the current index being processed. Use appropriate base cases in your recursive function to terminate the recursion. Calculate the factorial of the current element and update the array with the result.

1. Write a program that swaps the values of the first and last elements in an array of integers using pointers.
2. Write a program that reverses the elements of an array of integers using pointers
3. Write a recursive function **stringReverse** that takes a string and starting substring as an argument, prints the string backward return nothing. The function should stop processing and return when the end of string is encountered. Note that like an array the square brackets ([]) operator can be used to iterate through the characters in a string