



**National University of Computer & Emerging Sciences, Karachi**  
**Computer Science Department**  
**Fall 2022, Lab Manual - 01**



<b>Course Code: CL-1004</b>	<b>Course : Object Oriented Programming Lab</b>
<b>Instructor(s) :</b>	<b>Shahroz Bakht, Ayesha Ali, Zumar Noor</b>

## Contents:

1. Introduction to C++
2. Arrays
3. Pointers
4. Structures
5. Lab Tasks

## INTRODUCTION TO C++

C++ is very similar to the C Language.

- For the input/output stream we use **<iostream>** library (in C it was <stdio>).
- For taking input and out we **cout** and **cin** (in C it was printf and scanf).
  - cout uses **insertion ( << ) operator**.
  - cin uses **extraction ( >> ) operator**.

### Sample C++ Code:

```
#include <iostream>
using namespace std;
int main()
{
    int var = 0;

    cout << "Enter an Integer value: ";
    cin >> var;
    cout << "Value of var is : " << var;
    return 0;
}
```

Sample Run: In this sample run, the user input is shaded.  
Enter an Integer value: 12  
Value of var is : 12

# ARRAYS:

- An Array is a collection of fixed number of elements of same data type.

## 1-D ARRAY:

- 1-D Array is a form of array in which elements are arranged in a form of List.
- To declare a 1D array you need to specify the data type, name and array size.

```
dataType arrayName [ arraySize ] ;
```

- Following is the declaration of a 1D array.

```
int numArray[5];
```

- Data Type: Integers
  - Name: numArray
  - Size: 5
- To access array element you use the array name along with the index in subscript operator “[ ]”.

```
numArray[0], numArray[1], numArray[2], numArray[3], numArray[4].
```

- Index of the array starts with **zero '0'**.
- Index of the last element is always '**size - 1**' (in this case it is 4).

## Example Code for 1-D Array:

```
//Program to read five numbers, find their sum, and
//print the numbers in reverse order.

#include <iostream>
using namespace std;
int main()
{
    int item[5]; //Declare an array item of five components
    int sum = 0;
    int counter;

    cout << "Enter five numbers: ";

    for (counter = 0; counter < 5; counter++)
    {
        cin >> item[counter];
        sum = sum + item[counter];
    }
```

```

    cout << endl;

    cout << "The sum of the numbers is: " << sum << endl;
    cout << "The numbers in reverse order are: ";

    //Print the numbers in reverse order.
    for (counter = 4; counter >= 0; counter--)
        cout << item[counter] << " ";

    cout << endl;
    return 0;
}

```

Sample Run: In this sample run, the user input is shaded.

Enter five numbers: 12 76 34 52 89

The sum of the numbers is: 263

The numbers in reverse order are: 89 52 34 76 12

## 2-D ARRAY:

- 2-D Array is a collection of fixed collection of elements arranged in **rows and columns**.
- To declare a 2D array you need to specify the data type, name and no. of rows and columns.

```
dataType arrayName [ rowSize ][ columnSize ] ;
```

- Following is the declaration of a 1D array.

```
int numArray[5][5];
```

- Data Type: Integers
  - Name: numArray
  - Rows: 5
  - Columns: 5
- To access array element you use the array name along with the rowIndex and columnIndex in subscript operator "[ ][ ]".

```
numArray[0][0], numArray[1][1], numArray[2][2], numArray[3][3],
numArray[4][4].
```

- Index for the rows and columns of the array starts with **zero '0'**.
  - Index of the last element in rows and columns is always '**sizeofRow - 1**' and '**sizeofColumn - 1**' respectively (in this case it is 4).

## Example Code for 2-D Array:

```
//Program to read a 2D array of size 3x3 find the sum for each row,  
//print the sum line by line.
```

```
#include <iostream>  
using namespace std;  
int main()  
{  
    int item[3][3]; //Declare an array of size 3x3  
    int sum = 0;  
    int row, col;  
  
    cout << "Enter array elements: " << endl;  
  
    for (row = 0; row < 3; row++)  
    {  
        for (col = 0; col < 3; col++)  
        {  
            cin >> item[row][col];  
            sum = sum + item[row][col];  
        }  
  
        cout << "The sum of row " << i << " : " << sum <<  
endl;  
    }  
  
    cout << endl;  
    return 0;  
}
```

Sample Run: In this sample run, the user input is shaded.

Enter array elements:

12 76 34

The sum of row 0 : 122

52 89 48

The sum of row 1 : 189

22 63 99

The sum of row 2 : 184

# POINTERS:

A Pointer is a variable whose content is a memory address.

## Single Pointers:

- To declare a single pointer variable you need to specify the data type, an asterisk symbol ( \* ) and the name of the pointer variable.

```
dataType *ptrName;
```

- Following is the declaration of a Pointer variable.

```
int *ptr;
```

- DataType: Integer
- Name: ptr
- Pointer variable holds the memory address of the variable which is of same data type (integer in this case).
- To assign the memory address of any variable to the pointer variable we use **Address of Operator ( & )**.

```
int intVar = 5;  
ptr = &intVar;
```

- In this statement **ptr** now holds the memory address of an integer variable '**intVar**'.
- To access the value at the memory address (currently stored) in the variable we use **Dereferencing Operator ( \* )**.
  - Do not confuse this with the symbol used for the declaration of a pointer.

```
int intVar2 = *ptr;
```

- In this statement another integer variable '**intVar2**' is now initialized with the value at the memory address which is stored in **ptr** (that is the value of intVar).

## Example Code for Single Pointers:

The following program illustrates how pointer variables work:

```
#include <iostream>
using namespace std;
int main()
{
    int *p;
    int x = 37;
    cout << "Line 1: x = " << x << endl; //Line 1
    p = &x; //Line 2
    //Line 3
    cout << "Line 3: *p = " << *p << ", x = " << x << endl;
    *p = 58; //Line 4
    //Line 5
    cout << "Line 5: *p = " << *p << ", x = " << x << endl;
    cout << "Line 6: Address of p = " << &p << endl; //Line 6
    cout << "Line 7: Value of p = " << p << endl; //Line 7
    cout << "Line 8: Value of the memory location " << "pointed to
    by *p = " << *p << endl; //Line 8
    cout << "Line 9: Address of x = " << &x << endl; //Line 9
    cout << "Line 10: Value of x = " << x << endl; //Line 10
    return 0;
}
```

### Sample Run:

```
Line 1: x = 37
Line 3: *p = 37, x = 37
Line 5: *p = 58, x = 58
Line 6: Address of p = 006BFDF4
Line 7: Value of p = 006BFDF0
Line 8: Value of the memory location pointed to by *p = 58
Line 9: Address of x = 006BFDF0
Line 10: Value of x = 58
```

## DYNAMIC VARIABLES:

Variables created during the program execution are called **dynamic variables**.

- To create a dynamic variable we use **new** operator.

```
new dataType;           // to allocate a single variable
new dataType [ size];    // to allocate an array of variables.
```

- The new operator allocates the memory of a designated type.
- It returns a pointer to the allocated memory.
- Following is the declaration of a dynamic variable.

```
int p = new int;
char cArray = new char[5];
```

- Line 01: creates a single variable of integer type.
- Line 02: Creates an array of 5 characters.
- To delete the dynamically allocated memory we use **delete** operator.

```
delete ptrVar;           //to deallocate single dynamic variable
delete [] ptrArray;      //to deallocate dynamically created array
```

- **delete** operator is used to free the memory which is dynamically allocated using **new** operator.

## Example Code for Dynamic Variables:

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

int main()
{
    int* intPtr;

    char* charArray;
    int arraySize;

    intPtr = new int; // allocating memory to single variable

    cout << "Enter an Integer Value: ";
    cin >> *intPtr;
    cout << "Enter the size of the Character Array : ";
    cin >> arraySize;
```

```
charArray = new char[arraySize]; // allocating memory to array

    for (int i = 0; i < arraySize; i++)
        cin >> charArray[i];

    for (int i = 0; i < arraySize; i++)
        cout << charArray[i];

    return 0;

}
```

Sample Run: In this sample run, the user input is shaded.

Enter an Integer Value: 2

Enter the size of the Character Array : 2

ab

ab



## STRUCTURES:

- A structure is a collection of fixed number of components in which the components are accessed by name. The components may be of different types.
- Components of a structure are called members of the structure.
- To declare a structure you need to use the “**struct**” keyword along with the structure name.
  - The **struct** block then contains all the members, which are variables of different/same type.

```
struct structName
{
    dataType1 varName1;
    dataType2 varName2;
    . . .
    dataTypeN varNameN;
}
```

- To declare an object of the structure you need to use the **name of structure** and then the name of **structure object**.

```
struct studentStruct
{
    string firstName;
    string lastName;
    char courseGrade;
    int testScore;
    double GPA;
};
```

```
studentStruct Obj;
```

## Example Code for Structure:

```
#include<iostream>

using namespace std;

struct studentType
{
    string firstName;
    string lastName;
    char courseGrade;
    int courseScore;
    double GPA;
};

int main()
{
    studentType newStudent;

    cout << "Enter Details for the Student";

    cout << "Enter Student's First Name : ";
    cin >> newStudent.firstName;

    cout << "Enter Student's Last Name : ";
    cin >> newStudent.lastName;

    cout << "Enter Student's Course Grade : ";
    cin >> newStudent.courseGrade;

    cout << "Enter Student's Course Score : ";
    cin >> newStudent.courseScore;

    cout << "Enter Student's GPA : ";
    cin >> newStudent.GPA;

    cout << newStudent.firstName << endl;
    cout << newStudent.lastName << endl;
    cout << newStudent.courseGrade << endl;
    cout << newStudent.courseScore << endl;
    cout << newStudent.GPA << endl;

}
```

Sample Run: In this sample run, the user input is shaded.

```
Enter Details for the Student
Enter Student's First Name : First_Name
Enter Student's Last Name : Last_Name
Enter Student's Course Grade : A
Enter Student's Course Score : 84
Enter Student's GPA : 2.0
```

First\_Name  
Last\_Name  
A  
84  
2

### Lab Activities:

- 1) Write a C++ program from the user, which takes 3 floating point numbers from the user and prints the largest and smallest number of them.
  - 2) Write a C++ program to find the sum of individual digits of your roll number. (You may Use Loop here)
  - 3) Write a C++ program to perform the transpose of the 2-D array.
  - 4) Write a C++ program which stores your basic bio-data like name(use string data type for this), roll number, first semester GPA. The program should input number of students, scan the bio-data of all students and print the bio-data by the end.
  - 5) Write a program having following functionality using functions.
    - a. • Addition
    - b. • Subtraction
    - c. • Multiplication
    - d. • Division
- Hint. You must use different function for each functionality.

# Object Oriented Programing (CL-1004)

## Exercises

**Task\_01:**

Write a C++ program to input the marks of Chemistry, Mathematics and Physics (each out of 100) for atleast 4 students. Now you must use a structure named struct\_Marks having variables name, roll no., marks\_chem, marks\_maths and marks\_phy and then display the percentage of each student.

**Task\_02:**

Write a C++ program to find the maximum number between any 2 numbers using pointers.

**Task\_03:**

Write a C++ program that swaps the value of two variables using pointers.

**Task\_04:**

Write a C++ program that creates a function which returns the highest number of an array.

**Task\_05:**

Write a C++ program that creates a function which returns the count of duplicate elements in an array.

**Task\_06:**

Write a C++ program that finds the frequency of the letter "a" entered in a string.  
Example: Today is a great day!  
Frequency of "a" in the string is: 4

**Task\_07:**

Write a C++ program to create a structure which stores the names, salary and hours of work per day of 10 employees in a company. Write a program to increase the salary depending on the number of hours of work per day as follows and then print the name of all the employees along with their final salaries.

Hours of work per day	5	7	>=9
Increase in salary	3500	7000	9000

**Task\_08:**

Write a C++ program that includes a string and a pointer and check if the string is a mirror to itself or not using that pointer.

**Task\_09:**

Write a C++ program that takes a 4x4 matrix (2D – Array) as input and finds what is the highest row in terms of sum from all of the rows.

Example:

Input:

2 3 4 5

1 1 1 1

2 2 2 2

9 8 3 1

The highest row is of the 4th row: Sum is 21