**National University of Computer and Emerging Sciences**

# Lab Manual 01
# Object Oriented Programming

| | |
|---|---|
| Course Instructor | Ms. Noshaba Nasir |
| Lab Instructor (s) | Saif Ali<br>Siddiqua Nayyer |
| Section | BDS-2A |
| Semester | Spring 2021 |

Department of Computer Science
FAST-NU, Lahore, Pakistan

## 1.1 Objectives

After performing this lab, students shall be able to:

- ✔ Have an improved understanding of pointers.
- ✔ Access and modify arrays via pointers.
- ✔ Debugging.

-

## Debugging

See the following piece of code and write its output by debugging the code

```
int myFunction ()
{
        int numbers[5];
        int * p;
        p = numbers;
        *p = 10;
        p++;
        *p = 20;
        p = &numbers[2];
        *p = 30;
        p = numbers + 3;
        *p = 40;
        p = numbers;
        *(p+4) = 50;

         for (int n=0; n<5; n++)
           cout << numbers[n] << ", ";

         return 0;
}
Void main()
{
        myFunction();
}
```

Write the address of array named ‘numbers’

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| | | | | |

| Sr. No | code | Value of p | Address of p | Value of array ‘numbers’ | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | [0] | [1] | [2] | [3] | [4] |
| 1 | int numbers[5]; | | | | | | | |
| 2 | int * p=numbers; | | | | | | | |
| 3 | *p = 10; | | | | | | | |
| 4 | p++; | | | | | | | |
| 5 | *p = 20; | | | | | | | |
| 6 | p = &numbers[2]; | | | | | | | |
| 7 | *p = 30; | | | | | | | |
| 8 | p = numbers + 3; | | | | | | | |
| 9 | *p = 40; | | | | | | | |
| 10 | p = numbers; | | | | | | | |
| 11 | *(p+4) = 50; | | | | | | | |

**Help:** Debugging commands:


| Short cut key | Icon | Menu | Explanation |
|---|---|---|---|
| F-9 | | | Insert/Remove breakpoint |
| F-5 | | Debug-Go | Execute a program until the next breakpoint |
| Shift F-5 | | Debug-Stop debugging | To stop debugging a program. It will stop executing the program |
| F-10 | | Debug-StepOver | Go to the next statement |
| F-11 | | Debug-Step Into | Go inside a function |
| Shift F-11 | | Debug – Step Out | Come out of the function |
| | | Debug - Run to cursor | Execute all statements till the statement on which the cursor is placed or until the next breakpoint |

| | | | |
|---|---|---|---|
| Alt -3 | | Debug-Windows-Watch | Show the window where only the variables in scope are shown |
| Alt-4 | | Debug-Windows-Variables | Show the window in which you can type a variable name to see its value |
| Alt-7 | | debug-windows-call stack | You can see the activation of stack of functions here |

**Note: For all the integer or float variables in all the programs, take input value from user.**

**TASK 1:**

Write the following code and observe the output:

```
int x,y;
x=3, y=4;
int * p;
int * q;
p=& x;
q=&y;
cout<< x<<'\t'<<p<<'\t'<<*p<<'\t'<<&p<<'\t'<<&x<<endl;
cout<<y<<'\t'<<q<<'\t'<<*q<<'\t'<<&q<<'\t'<<&y<<endl;
```

**TASK 2:**

Given two integers x and y, find and print their sum using pointers.

**TASK 3:**

Write a C++ program that creates a pointer to an integer and print the following:
Square of the integer, cube of the integer, half of the integer

**TASK 4:**

Write a C++ program that finds and prints the median of following three integers using their pointers.

```
int a=5;
int b=10;
int c=12;
```

**TASK 5:**

A C++ program where you create an integer array of **size 10**. Your program will add 3 to each element of the array. You have to add to the elements using pointer only. Array subscript notation cannot be used (neither in addition nor while printing resultant array).

Note  *(ptr+i) is same as ptr[i]

## Exercise – Expand Array

Write a program that keeps taking integer input from the user until user enters -1 and displays the data in reverse order.

Your program should save the input in a dynamically allocated array. Initially create a dynamic array of five integers. Each time the array gets filled your program should double the size of array (i.e. create a new array of double size, copy previous data in new array, delete previous array) and continue taking the input. After receiving -1 (i.e. end of data input) your program should print the numbers in the reverse order as entered by the user.

**Important Note:** subscript operator [] is not allowed to traverse the array. Use only offset notation. i.e instead of using myArray[i] use *(myArray+i) to read/write an element. **Do not consume extra space. There shouldn't be any memory leakage or dangling pointers in your code.**