

National University of Computer and Emerging Sciences



Lab Manual 1 Object Oriented Programming – CL1004

Course Instructor	Dr. Saira Karim
Lab Instructor(s)	Mr. Muhammad Adeel Miss Amna Zulfiqar Ali
Section	BS CS B
Semester	Spring 2023

Department of Computer Science
FAST-NU, Lahore, Pakistan

Lab Manual 1 – Pointers and Arrays

Important Note:

- You may find the syntax to accomplish these exercises from lecture demo.
- Names of your submission files should start with your roll number throughout this semester.
- Make sure that the interface of your program is user friendly i.e. properly display information.
- Properly follow the coding standards.

1. Exercise – Debugging [30 Minutes]

See the following piece of code and write its output by debugging the code. Keys for debugging are listed below.

```
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();
}
```

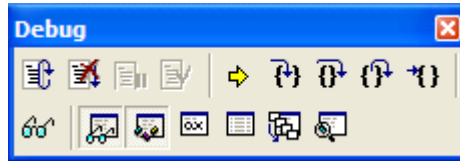
Write the address of array named 'numbers'










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

2. Exercise – Basic Pointers [30 Minutes]

Follow these steps:

1. Declare:
 - a. Int variables num1, num2 and sum
 - b. Int* pointer variables xPtr, yPtr and sumPtr
2. Set num1, num2 and sum to 5, 7 and 0 respectively
3. Initialize all pointers to 0 (nullptr)
4. Print values of variables num1, num2 with labels as shown in the required output below:

Required Output:

```
Num1 = 5
Num2 = 7
```

5. Print the addresses of Num1 and Num2 using Address Operator (&). Required Output (assuming addresses starting from 0x10 for Num1 and so on) is shown below. (Note that addresses will be different on different machines)

Required Output:

```
Num1 = 5
Num2 = 7
Address of Num1 = 0x10 //(This will be different on your machine)
Address of Num 2 = 0x14 //(This will be different on your machine)
```

6. Point xPtr to num1 and yPtr to num2
7. Print values of Num1 and Num2 by dereferencing xPtr and yPtr

Required Output:

```
Num1 = 5
Num2 = 7
Address of Num1 = 0x10 //(This will be different on your machine)
Address of Num 2 = 0x14 //(This will be different on your machine)
*xPtr = 5
*yPtr = 7
```

8. Point sumPtr to sum and print sum by dereferencing sumPtr.

Required Output:

```
Num1 = 5
Num2 = 7
Address of Num1 = 0x10 //(This will be different on your machine)
Address of Num 2 = 0x14 //(This will be different on your machine)
*xPtr = 5
*yPtr = 7
*sumPtr = 0
```

9. Add num1 and num2 using *xPtr and *yPtr and save the result in integer sum
10. Again Print sum using sumPtr

Required Output:

```
Num1 = 5
Num2 = 7
Address of Num1 = 0x10 //(This will be different on your machine)
Address of Num 2 = 0x14//(This will be different on your machine)
*xPtr = 5
*yPtr = 7
*sumPtr = 12
```

11. Print the values of xPtr and yPtr (cout<<"xPtr = "<<xPtr<<endl)

Required Output:

```
Num1 = 5
Num2 = 7
Address of Num1 = 0x10 //(This will be different on your machine)
Address of Num 2 = 0x14//(This will be different on your machine)
*xPtr = 5
*yPtr = 7
*sumPtr = 12
xPtr = 0x10 //This output should be same as address of num1 i.e. &num1
yPtr = 0x14 //This output should be same as address of num2 i.e. &num2
```

Help:

```
cout<<"Num1 = "<<num1<<endl; // Prints Num1 = 5
sum = *xPtr + *yPtr // Add num1 and num2 using *xPtr and *yPtr and save the result in integer sum
```

3. Exercise – Manipulating 2D array [30 Minutes]

Write a function in C++ which accepts a 2D array of integers and its size as arguments and displays the elements of middle row and the elements of middle column.

Note: Assume the 2D Array to be a square matrix with odd dimension i.e. 3x3 or 5x5

Declare and Initialize a 2D array (square) in main() and call the above created function by passing the created array and its size

Example, if the array contents are:

```
3 5 4
7 6 9
2 1 8
```

Required Output through the function should be:

```
Middle Row: 7 6 9
Middle column: 5 6 1
```

4. Exercise - Array Manipulation using pointers [30 Minutes]

Write a C++ program to accept five integer values from the keyboard and store them in an array using a pointer. Then print all the elements of an array in reverse order.

Note: use index number to return a value from an array

5. Exercise - Array Manipulation using pointers [30 Minutes]

Write a program to find the smallest and the largest value in an array. Read 5 values in an array in the main program and call the function "void func(int *a)" to access values of array. After finding the value place the smallest value at index 0 and the largest value at index 4 of the original array. To access the elements of array, use the notation " $*(a+i)$ ", where a is the reference of array and 'i' is for iteration. Print the smallest and the largest value in the main program.