1-D Array

Course Code: CSC 2107 Course Title: Data Structure (Lab)

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	**2**	**Week No:**	**2**	**Semester:**	**Spring 24-25**
Lecturer:	*Zerin Hasan Sahosh & sahosh@aiub.edu*				

Lecture Outline

1. Rules & Guidelines
2. Lab Tasks
3. Prerequisites
4. Objectives
5. Problem Descriptions
6. Books
7. References

Lab Tasks

1. Write C++ code to solve all the problems starting from slide 7 to 11.
2. Any remaining problem unsolved will be home task.

Prerequisites

- Have a clear and full understanding of 1-D Array.
- Theory Lectures 1.1 & 1.2

Objectives

- To know how to solve basic, moderate and complex programming problems using 1-Dimensional Array.
- To master array operations such as traversal, searching, insertion, and deletion.

Problem Descriptions

Problem 1

1. Initialize TWO integer arrays of different sizes. Merge the input arrays and create a new array. Then print the new array in reverse order.

For example,
Array_1 = `{10,20,30,40,50}`
Array_2 = `{1,2,3,4,5,6,7,8}`

Output: `8 7 6 5 4 3 2 1 50 40 30 20 10`

Problem Descriptions

Problem 2

3. Initialize an array. Size should be more than FIVE. Write you program to change the array in such a way so that there cannot be any duplicate element in the array anymore. Print the changed array. If the initialized array already had no duplicate elements from the beginning, output a message saying “Array already unique!”;

For example,

Scenario 1:

Array_1 = `{1,4,6,3,6,9,1}`

Output: `1  4  6  3  9`

Scenario 2:

Array_1 = `{1,4,5,3,6,9}`

Output: `Array already unique!`

Problem Descriptions

Problem 3

3. Initialize TWO integer arrays **A** and **B** of different sizes. Make a new array with the common elements between **A** and **B**. Print the new array element(s). If there is no common element, output “No common element!”.

For example,

Scenario 1:

Array_1 = `{1,4,6,3,6,9}`

Array_2 = `{5,3,7,1,2,6}`

Output: `1 6 3`

Scenario 2:

Array_1 = `{1,4,6,3,6,9}`

Array_2 = `{5,8,7,12,21,63}`

Output: `No common element!`

Problem Descriptions

Problem 4

4. Initialize an integer array **A** of size 10. Take an integer as input and print how many times that integer occurs in **A**.

For example,
Array_1 = `{8,4,6,1,6,9,6,1,9,8}`

Output:

```
Input a number to search: 6
The number occurs 3 times in the array
```

Problem Descriptions

Problem 5

5. Initialize an integer array of size 10. Print the number of time each element occurs in the array.

For example,
Array_1 = `{8,4,6,1,6,9,6,1,9,8}`

Output:

```
8 occurs = 2 times
4 occurs = 1 time
6 occurs = 3 times
1 occurs = 2 times
9 occurs = 2 times
```

Books

- **“Schaum's Outline of Data Structures with C++”**. By John R. Hubbard
- **“Data Structures and Program Design”,** Robert L. Kruse, 3rd Edition, 1996.
- **“Data structures, algorithms and performance”,** D. Wood, Addison-Wesley, 1993
- **“Advanced Data Structures”,** Peter Brass, Cambridge University Press, 2008
- **“Data Structures and Algorithm Analysis”,** Edition 3.2 (C++ Version), Clifford A. Shaffer, Virginia Tech, Blacksburg, VA 24061 January 2, 2012
- **“C++ Data Structures”,** Nell Dale and David Teague, Jones and Bartlett Publishers, 2001.
- **“Data Structures and Algorithms with Object-Oriented Design Patterns in C++”,** Bruno R. Preiss,

References

1. Theory Lecture 1.1 & 1.2 of this course
2. https://en.wikipedia.org/wiki/Array_data_structure