

Riphah International University

Islamabad

**LAB MANUAL**

**Lab Manual for Data Structure & Algorithm**

**FACULTY OF COMPUTING (WISH)**

**Table of Contents**

Contents

Lab 1: Arrays............................................................................................................................................5

1. Installation ......................................................................................................................................5

2. Introduction ....................................................................................................................................5

3. Activity Time boxing........................................................................................................................6

4. Lab Manual Lecture [Expected time = 20 minutes]...........................................................6

5. Objective .........................................................................................................................................6

6. Concept Map...................................................................................................................................6

Arrays ..................................................................................................................................................6

7.1 Update Operation ...................................................................................................................7

7.2 Search Operation ....................................................................................................................7

9. Evaluation criteria ...........................................................................................................................9

10. Further Reading ..........................................................................................................................9

10.1 Books.......................................................................................................................................9

7.7 Out comes .....................................................................................................................................9

––

**Lab Manual for Data Structure & Algorithm**

**Lab 1: Arrays**

**Lab 1: Arrays**

**1. Installation**

The IDE that we will be using in this course is *Dev-C++*. It is a free IDE for Windows that uses either MinGW

or TDM-GCC as underlying compiler.

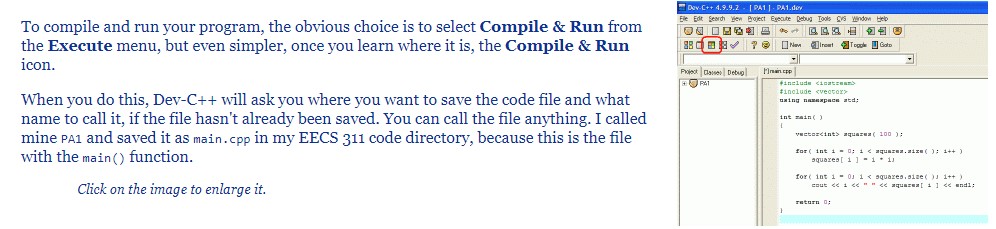
To compile and run simple console applications such as Lab Tasks, it is enough with opening the file with

Dev-C++ and hit F11. As an example, try:

**File -> New -> Source File (or Ctrl+N)** and write your code. Then:

**File -> Save As... (or Ctrl+Alt+S)**

And save it with some file name with a .cpp extension, such as example.cpp. Now, hitting F11 should compile and run the program.



**2. Introduction**

An array is used to process a collection of data all of which is of same type, such as a list of temperatures or list of names. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

For example, you want to store the score of 5 students, you can declare 5 variables, int student1, student2, student3, student4, student5;

Now what if the scenario changes and now we want to store the score of 100 students?

So instead of declaring individual variables, you declare one array variable such as **students** and use students [0], students [1], and ..., students [99] to represent individual variables.

**0 1 2 [n-1]**

**Index**

**Elements**

students [0] students [1] students [2] ………. students [n-1]

In this lab you will revise the basic working of arrays.

**3. Activity Time boxing**

Table 1: Activity Time Boxing

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Activity Name** | **Activity time** | **Total Time** |
|  | Lab Manual Lecture | 10 min |  |
|  | Examples | 5 min |  |
|  | Walkthrough Tasks | 10 min |  |
|  | Practice Tasks | 70 min |  |
|  | Tasks Evaluation | 10 min | 180 minutes |

4. Lab Manual Lecture [Expected time = 20 minutes]

**5. Objective**

1. Concepts of Data Structure

2. What is Array

3. CRUD operations on Arrays

**6. Concept Map**

## What is a data structure?

It is a data organization, management and storage format that

enables [efficient access and modification](https://en.wikipedia.org/wiki/Algorithmic_efficiency).

Examples are arrays, Lists, Binary trees, Heaps.

Today we will discuss and revise arrays.

## Arrays

An array is a collection of data that holds fixed number of values of same type. For example:

int age[100];

Here, the age array can hold maximum of 100 elements of integer type.

The size and type of arrays cannot be changed after its declaration.

### Declaring array:

type arrayName [ arraySize];

int students [5]; **Initializing array:** students = {10, 8, 9, 5, 7};

**Sample Program**

#include <iostream>

using namespace std;

int main () {

int students [ 5]; // n is an array of 5 integers

// initialize elements of array

for (int i = 0; i < 5; i++) {

cin>>students[i];

}

// output each array element's value

for (int j = 0; j < 10; j++) {

cout << “Student” <<j<< “scored” <<students[i] << “marks” <<endl;

}

return 0;

}

### 2.2. Array Operations

There are some basic operations that can be performed on array which are adding data element to array,

searching and updating particular data element in an array and deleting data element from array. Following are the detailed description of these operations

Following are the basic operations supported by an array.

* **Traverse** − print all the array elements one by one.
* **Insertion** − Adds an element at the given index.
* **Deletion** − Deletes an element at the given index.
* **Search** − Searches an element using the given index or by the value.
* **Update** − Updates an element at the given index.
  + 1. **Traverse Operation**

This operation is to traverse through the elements of an array.

1. Example

Following program traverses and prints the elements of an array:

#include <stdio.h>

main() {

int LA[] = {1,3,5,7,8};

int item = 10, k = 3, n = 5;

int i = 0, j = n;

printf("The original array elements are :\n");

for(i = 0; i<n; i++) {

cout<< LA[i];

}

}

* + 1. **Insertion Operation**

Insert operation is to insert one or more data elements into an array. Based on the requirement, a new element can be added at the beginning, end, or any given index of array.

Here, we see a practical implementation of insertion operation, where we add data at the end of the array Example

Following is the implementation of the above algorithm −

#include <stdio.h>

main() {

int LA[] = {1,3,5,7,8};

int item = 10, k = 3, n = 5;

int i = 0, j = n;

cout<<"The original array elements are :\n");

for(i = 0; i<n; i++) {

cout<<LA[i];

}

n = n + 1;

while( j >= k) {

LA[j+1] = LA[j];

j = j - 1;

}

LA[k] = item;

Cout<<”The array elements after insertion :\n";

for(i = 0; i<n; i++) {

cout<<LA[%d] = %d \n", i, LA[i];

}

* + 1. **Update Operation**

void Update Values ()

{

cout<<"Enter Index Number to Update Value:";

int index;

cin>>index;

cout<<"Enter the New Value For Index array [ "<<index<<" ] = ";

cin>>array[index];

cout<<"Array Updated... Successfully "<<endl;

}

* + 1. **Search Operation**

1. First take number of elements in array as input from user and store it in a variable **size**.

2. Using a loop, take input from user and store it in array (Let the name of the array be **inputArray**).

3. Ask user to enter element to be searched. Let it be num.

4. Now, using a for loop, traverse **inputArray** from index 0 to size-1 and compare **num** with every array element. If **num** is equal to any array element then print a message saying "Element found at index 4" otherwise print "Element Not Present".

* + 1. **Delete Operation**

Deletion refers to removing an existing element from the array and re-organizing all elements of an array.

### Algorithm

Consider **LA** is a linear array with **N** elements and **K** is a positive integer such that **K<=N**. Following is the algorithm to delete an element available at the Kth position of LA.

1. Start

2. Set J = K

3. Repeat steps 4 and 5 while J < N

4. Set LA[J] = LA[J + 1]

5. Set J = J+1

6. Set N = N-1

7. Stop

#include <iostream>

#include <stdio.h>

using namespace std;

int main()

{

int del; int count;

int array[10],m;

cout<<"Enter the size of your array:";

cin>>m;

for(int i=0;i<m; i++)

{

cin>>array[i];}

cout<<"Enter the number to delete"<<endl;

cin>>del;

for (int i=0; i<m; i++)

{

if(array[i]==del)

{

for (int j=i; j<(m-1); j++)

{

array[j]=array[j+1];

}

count++;

break;

} }

if(count==0) {

cout<<"Element not found...!!";

}

else {

cout<<"Element deleted successfully...!!\n"; cout<<"Now the new array is :\n";

for(int i=0; i<(m-1); i++)

{

cout<<array[i]<<" ";

}

}

return 0;}

**Practice Tasks**

**Task 1: [Time required: 20 minutes]**

Write a program that lets a maker of chips and salsa keep track of sales for five different types of salsa: mild, medium, sweet, hot, and zesty.

The program should use two parallel 5-element arrays: an array of strings that holds the five salsa names and an array of integers that holds the number of jars sold during the past month for each salsa type. The salsa names should be stored using an initialization list at the time the name array is created.

The program should prompt the user to enter the number of jars sold for each type. Once this sales data has been entered, the program should produce a report that displays sales for each salsa type, total sales, and the names of the highest selling and lowest selling products.

**Input Validation**: Do not accept negative values for number of jars sold.

**Task 2: [Time required: 25 minutes]**

Write a program that uses the following arrays:

• empId: an array of seven long integers to hold employee identification numbers. The array should be initialized with the following numbers:

5658845 4520125 7895122 8777541 8451277 1302850 7580489

• hours: an array of seven integers to hold the number of hours worked by each employee

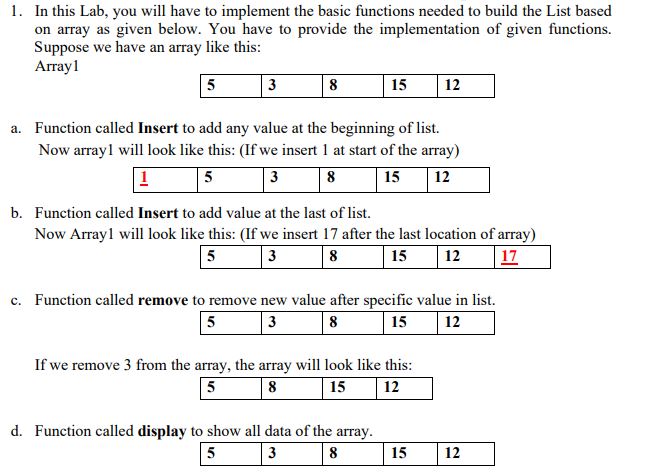
• payRate: an array of seven doubles to hold each employee’s hourly pay rate

• wages: an array of seven doubles to hold each employee’s gross wages

Program should relate the data in each array through the subscripts. For example, the number in element 0 of the hours array should be the number of hours worked by the employee whose identification number is stored in element 0 of the empId array. That same employee’s pay rate should be stored in element 0 of the payRate array.

The program should display each employee number and ask the user to enter that employee’s hours and pay rate. It should then calculate the gross wages for that employee (hours times pay rate) and store them in the wages array. After the data has been entered for all the employees, the program should display each employee’s identification number and gross wages. Input Validation: Do not accept negative values for hours or numbers less than 15.00 for pay rate.

**Task 3: [Time required: 25 minutes]**



1. Evaluation Criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

**Table 2: Evaluation of Lab**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Task No** | **Marks** | Total |
| 1 | Task 1 | 10 |  |
| 2 | Task 2 | 10 |  |
| 3 | Task 3 | 10 |  |

## **Further Reading**

### Books

Data Structures and Algorithms by Granville Barnett, and Luca Del

Tongo 2008

**Slides**

The slides and reading material can be accessed from the folder of the class instructor available at Vle.

### Outcomes

The outcomes of this lab were:

* 1. Students learn arrays data structures in C++
  2. Learn and implement Insert, Update & Delete operations on array.