

**Lab 01**

**Static Arrays and File Streaming**

## **OBJECTIVE:**

Things that will be covered in today's lab:

- Introduction to Compiler
- Revision of Static Arrays

## **Introduction to Compiler (Visual Studio 2010)**

### 1. Creating a new project:

Go to the File menu. Select New. In the Project Tab, select Win32 Console Application. Write the Project Name. You can change the default location of the project from the location box. Check the Win32 Platform. Press OK. A new window appears. Select An Empty Project. Press Finish. If a new window appears showing project details, press OK.

### 2. Source File (.cpp):

Go to Project menu. Select Add to Project. Select New. From File Tab, select a C++ Source file. Write name of file and press ok.

### 3. Compiling the code:

After writing the code, press CTRL+ F7, or from the Build menu, select Compile (Building the executable file). After compiling, press F7.

### 4. Running the exe file:

Press CTRL+F5 for Debugging the code at any line of the code. Press F9 to insert a breakpoint.

Then do the following to start debugging.

Build>Start Debug>Go OR Press F5. Debugging is started now. Press F11 for step into, F10 for step over and shift+f11 for step out.

## **File Streaming:**

We have been dealing with input and output streams the whole time (cin and cout). The input/output stream is used with `<iostream>` library.

In C++ and any other programming language, you can read and write from a text file (.txt extension). We have to include a library in the same way as we include a library for cin and cout.

A five step process:

1. Include the header file <i>fstream</i>	<code>#include&lt;fstream&gt;</code>
2. Declare the file stream variables	<code>Ifstream inDATA; Ofstream outDATA;</code>
3. Open input output files	<code>inDATA.open ("Infile.txt"); outDATA.open ("Outfile.txt");</code>
4. Read or write data from files	<code>inDATA&gt;&gt;varName1; outDATA&lt;&lt;varName2;</code>
5. Close the file	<code>InDATA.close (); outDATA.close ();</code>

### Exercise 1:

A selection sort code searches an array looking for the smallest element in the array. Then, the smallest element is swapped with the first element of the array. The process is repeated for the sub-array beginning with the second element of the array. Each pass of the array results in one element being placed in its proper location. When the sub-array being processed contains one element, the array is sorted. Write C++ code for this selection sort and output must be stored in "output.txt" file (having all passes).

Example:

<b>Unsorted list</b>	<b>23 78 45 08 32 56</b>
First Pass	08 78 45 23 32 56
Second Pass	08 23 45 78 32 56
Third Pass	08 23 32 78 45 56
Fourth Pass	08 23 32 45 78 56
Fifth Pass	08 23 32 45 56 78
<b>Sorted List</b>	<b>08 23 32 45 56 78</b>

**Exercise 2:**

Write a program that reads students' names followed by their test scores. The program should output each student's name followed by the test scores and the relevant grade. It should also find and print the highest test score and the name of the students having the highest test score.

Student data is stored in a "student.txt" file and are arranged in following order.

- Student First Name of type string
- Student Last Name of type string
- Test Score of type int (testScore is between 0 and 100)
- Grade of type char.

Suppose that we have data of 20 students in a file. Your program must contain at least the following functions:

- A function to read the student's data into the array.
- A function to assign the relevant grade to each student.
- A function to find the highest test score.
- A function to print the names of the students having the highest test score.

Your program must output each student's name in this form: last name followed by a comma, followed by a space, followed by the first name; the name must be left justified. Moreover, other than declaring the variables and opening the input and output files, the function main should only be a collection of function calls.

**Post Lab:**

Write a program which takes 10 values as input and stores them in an array. Calculate the average of five values starting from index zero. Increment the index by one till the end of array. On every increment, calculate average of next five values starting from incremented index and store the calculated average in a separate array.

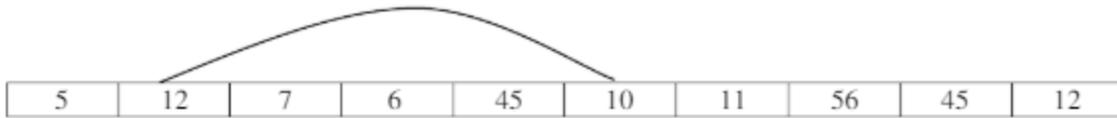
Example:

# Lab Manual of Programming Fundamentals



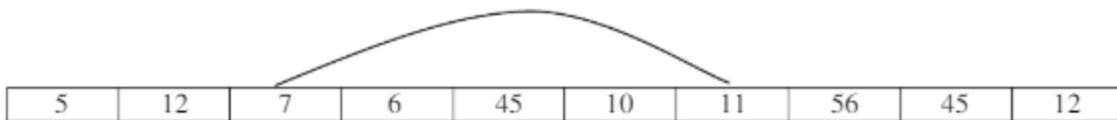
Average=  $(5+12+7+6+45)/5 = 15$

15					
----	--	--	--	--	--



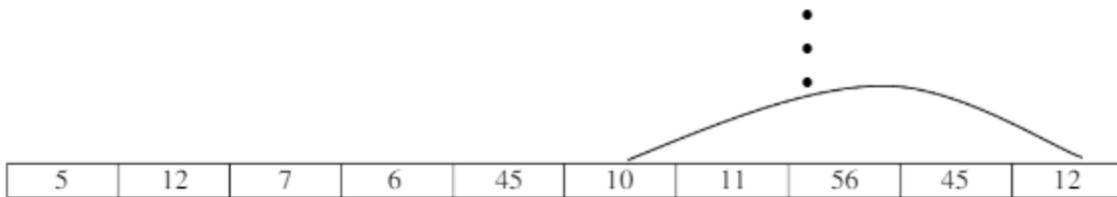
Average=  $(12+7+6+45+10)/5 = 16$

15	16				
----	----	--	--	--	--



Average=  $(7+6+45+10+11)/5 = 15.8$

15	16	15.8			
----	----	------	--	--	--



Average=  $(10+11+56+45+12)/5 = 13.8$

15	16	15.8	25.6	33.4	26.8
----	----	------	------	------	------