LAB MANUAL

“One Dimensional Arrays”

Programming Fundamentals

BS(CS)



FAST National University of  
Computer and Emerging Sciences

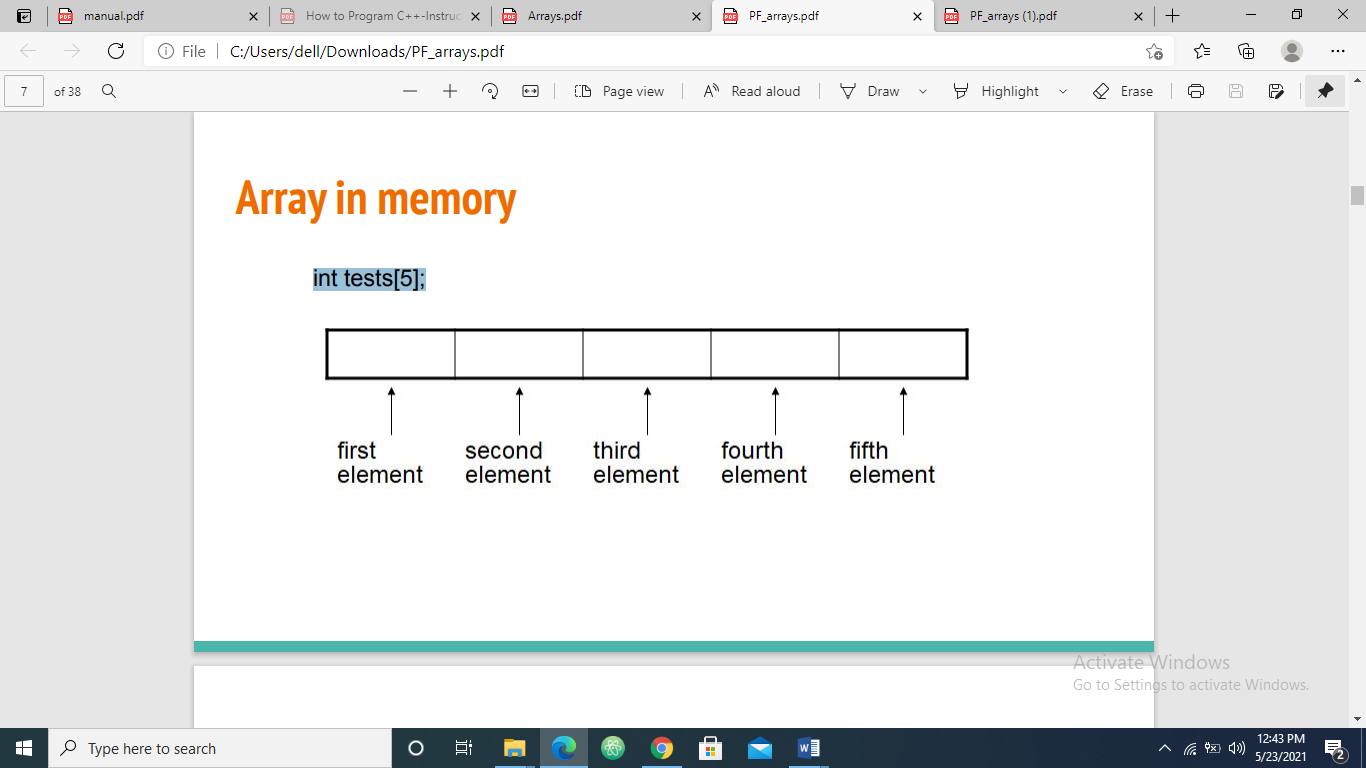
**Arrays:**

An array is a data structure for storing more than one data item that has a similar data type. Values are stored in adjacent memory locations.

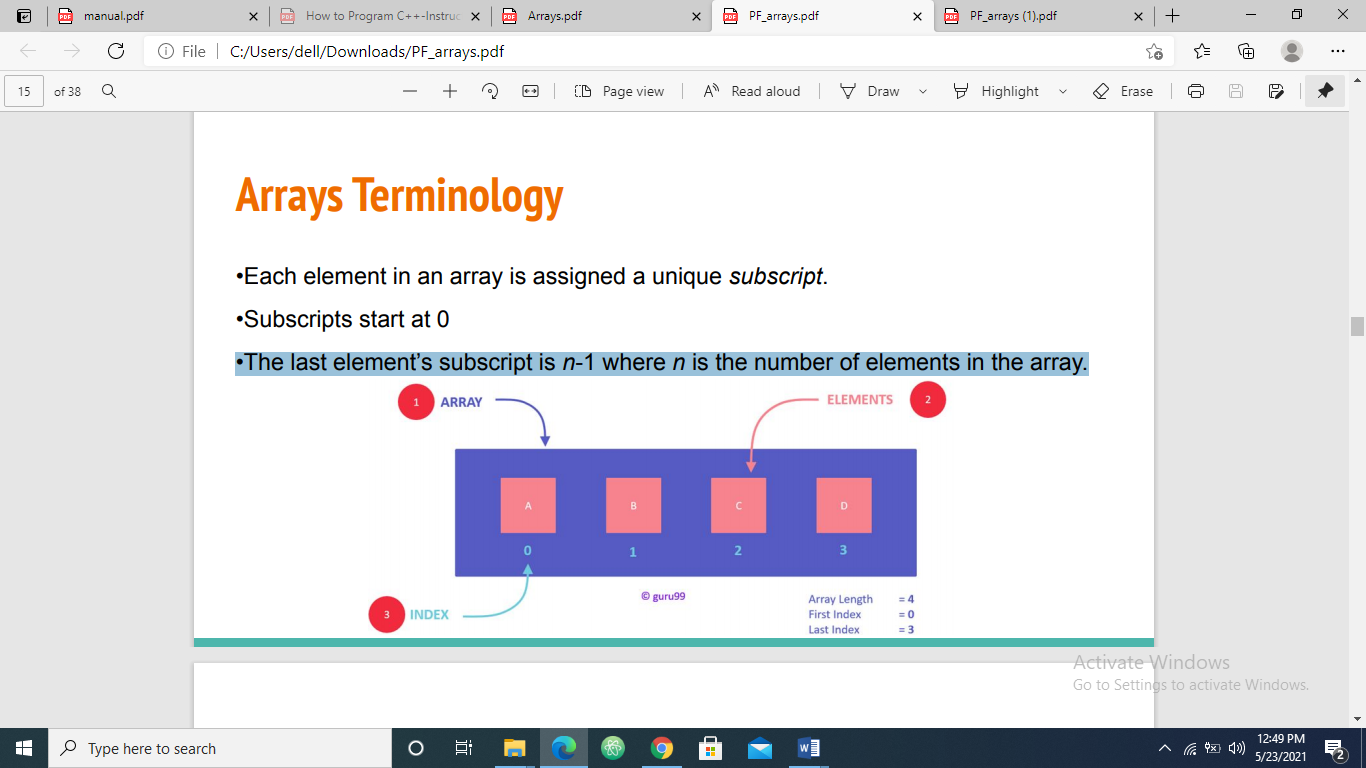
Declare using [] operator:

**int tests[5];**

The definition: int tests[5]; allocates the following memory:



**Arrays Terminology:**



**Size Declarators:**

• Named constants are commonly used as size declarators.

**const int SIZE = 5;**

**int tests[SIZE];**

**Array Initialization:**

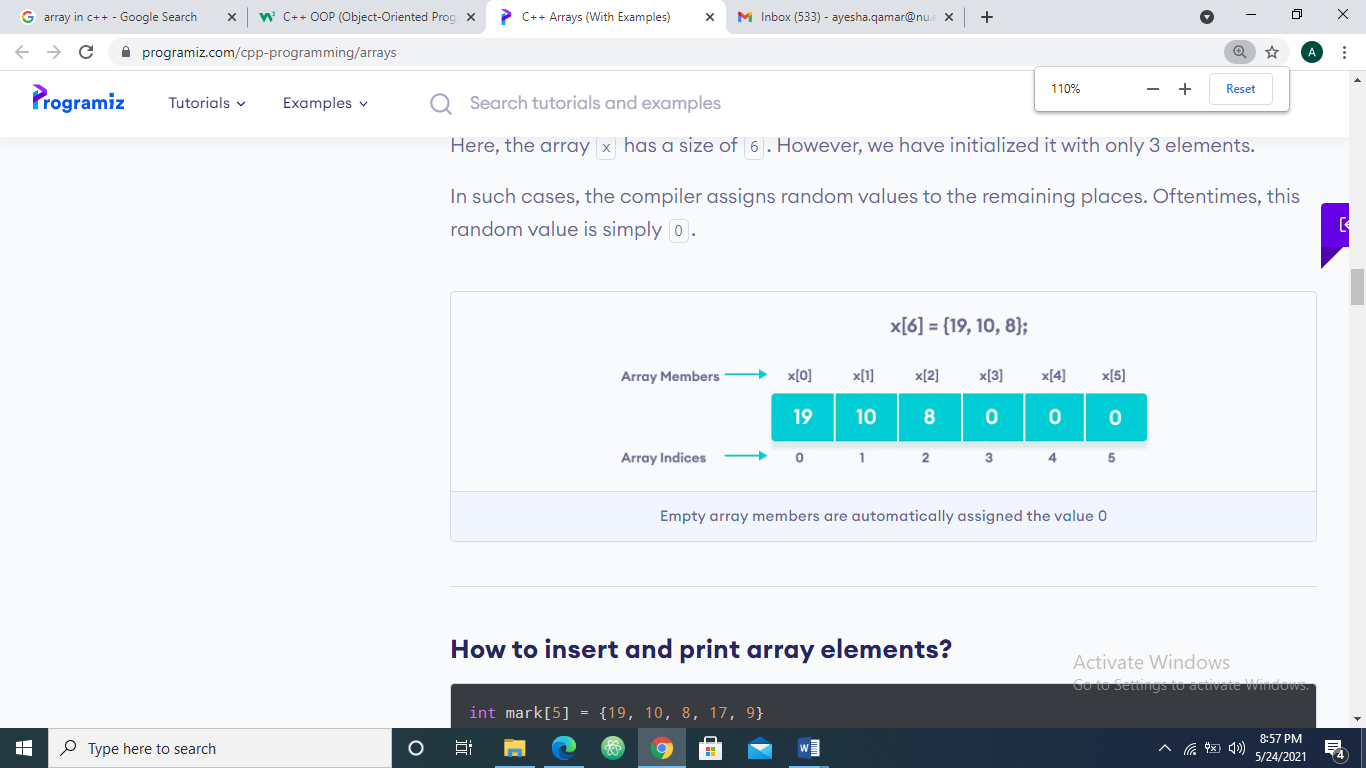
• Arrays can be initialized with an **initialization list**:

**const int SIZE = 5;**

**int tests[SIZE] = {79, 82, 91, 77, 84};**

• The values are stored in the array in the order in which they appear in the list.

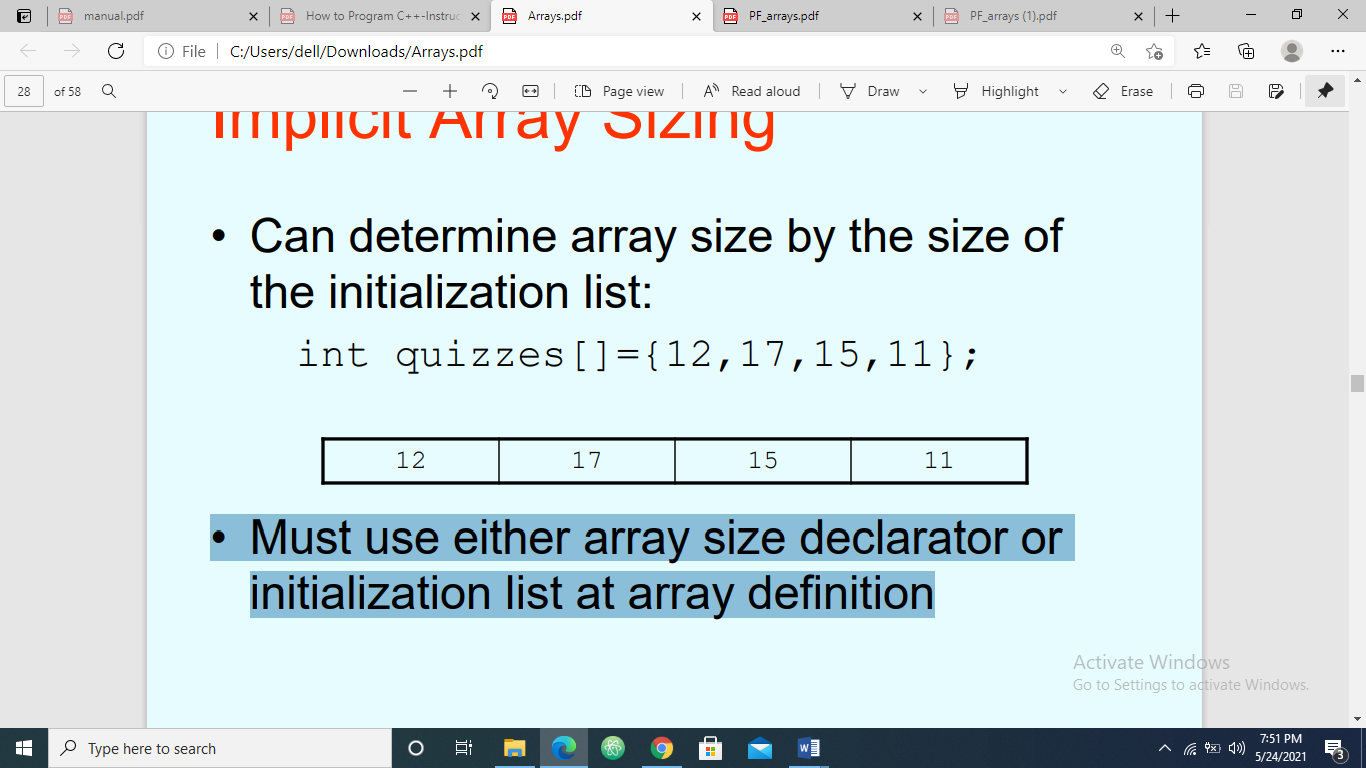
• The initialization list cannot exceed the array size.



**Implicit Array Sizing:**

• Can determine array size by the size of the initialization list:

**int quizzes[]={12, 17, 15, 11};**



• Must use either array size declarator or initialization list at array definition

* Once the array is declared (it’s size is set for fixed size arrays), you cannot change the size again. That is, you cannot add more elements to it

**Accessing Array Elements:**

• Array elements can be used as regular variables:

**tests[0] = 79;**

**cout << tests[0];**

**cin >> tests[1];**

**tests[4] = tests[0] + tests[1];**

• Arrays must be accessed via individual elements: **cout << tests; // not legal, it only works with character arrays**

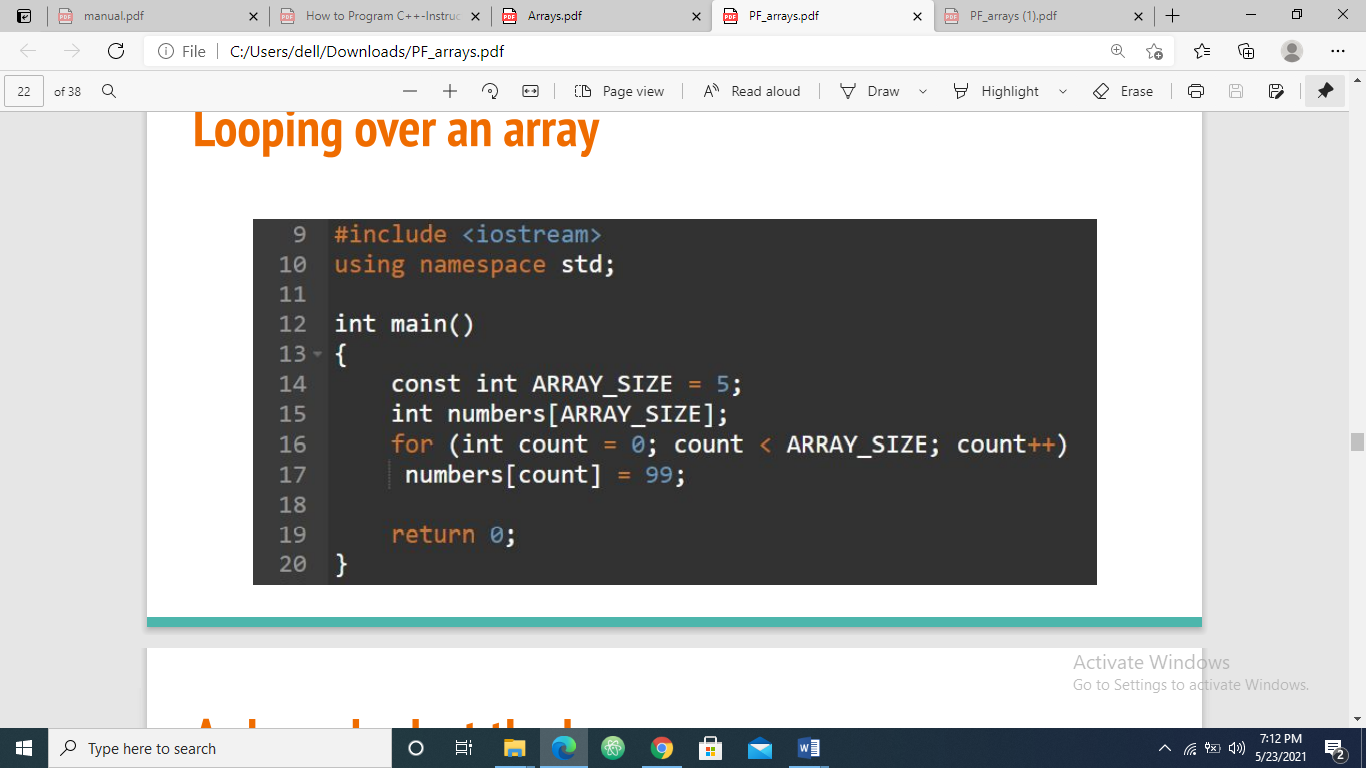
• Can access element with a constant or literal subscript: **cout << tests[3] << endl;**

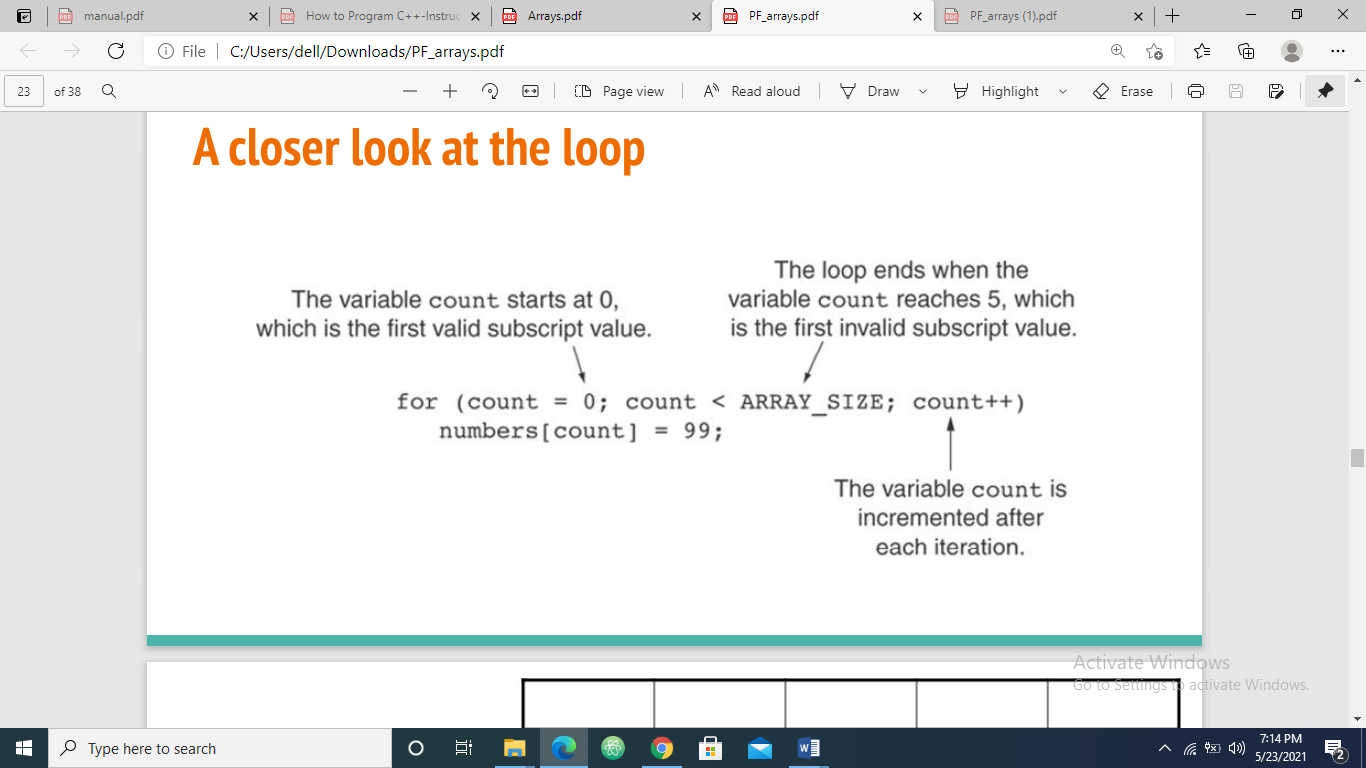
• Can use integer expression as subscript:

**int i = 5;**

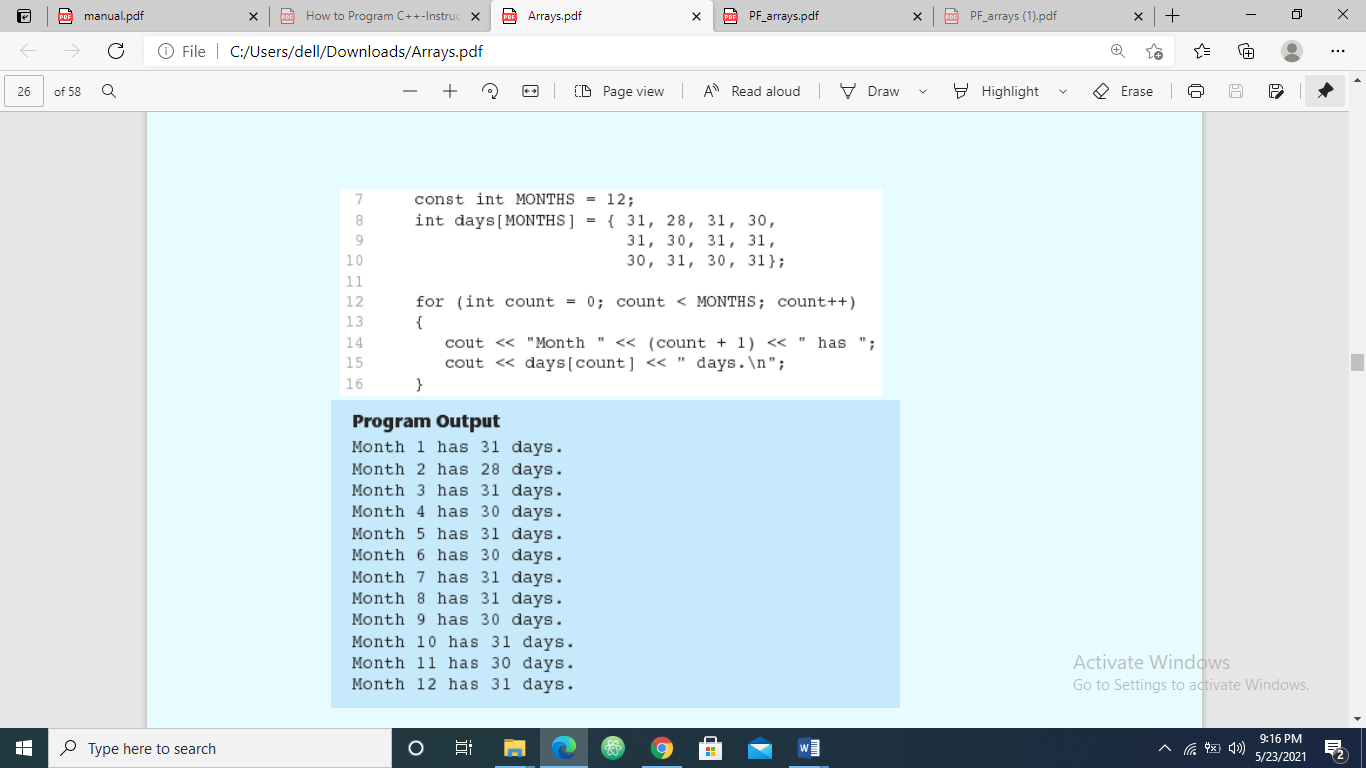
**cout << tests[i] << endl;**

**Looping over an array:**





**Printing out array using loops:**



**Processing Array Contents:**

• Array elements can be treated as ordinary variables of the same type as the array

• When using ++, -- operators, don’t confuse the element with the subscript:

**tests[i]++; // add 1 to tests[i]**

**tests[i++]; // increment i, no effect on tests**

• To copy one array to another, don’t try to assign one array to the other:

**newTests = tests; // Won't work**

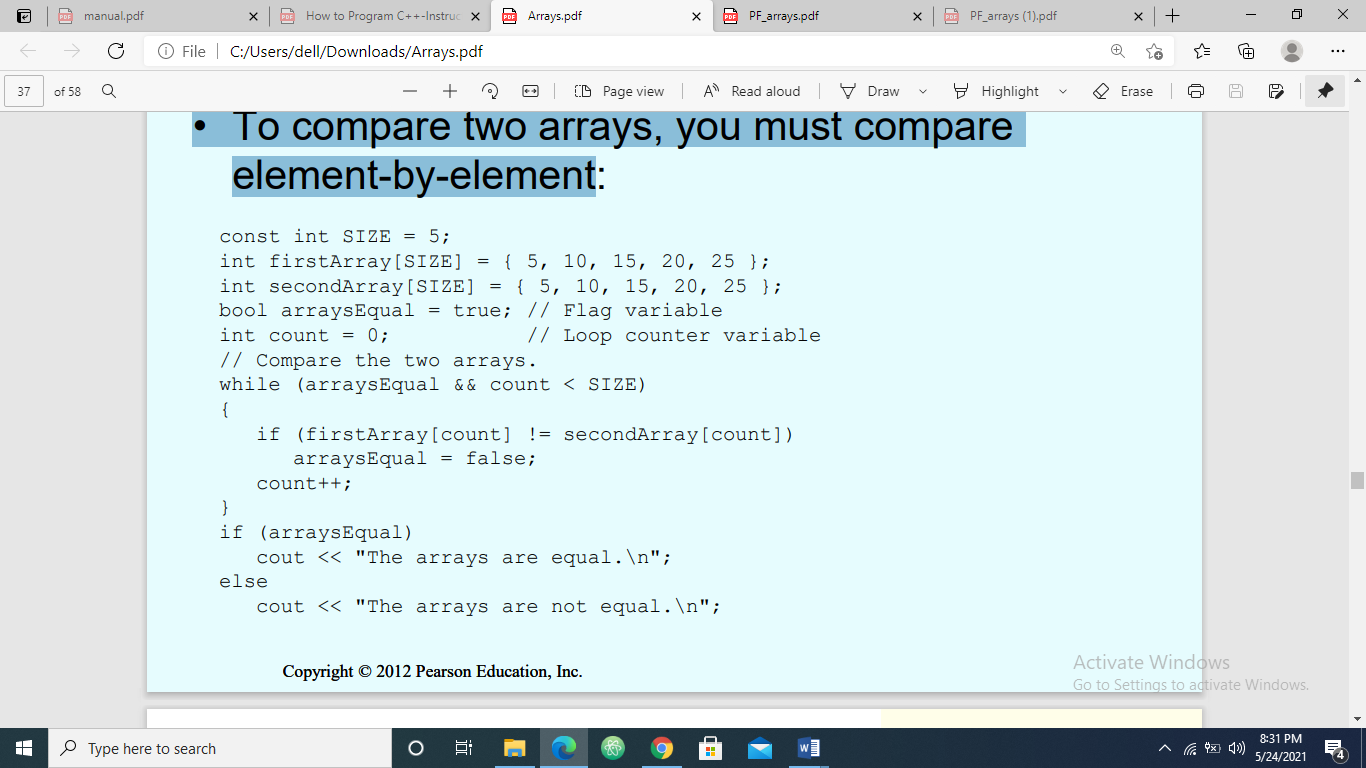
• Instead, assign element-by-element:

**for (i = 0; i < ARRAY\_SIZE; i++)**

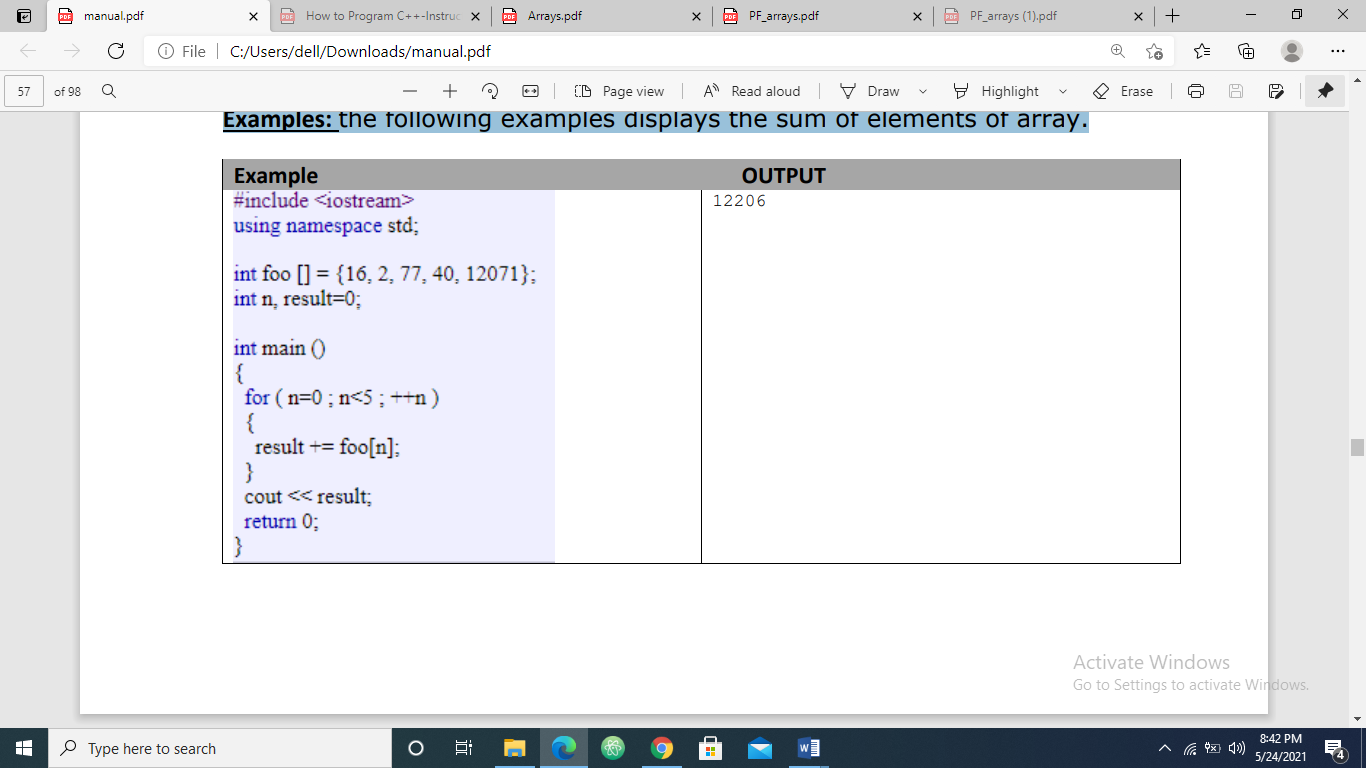
**newTests[i] = tests[i];**

**Comparing Arrays:**

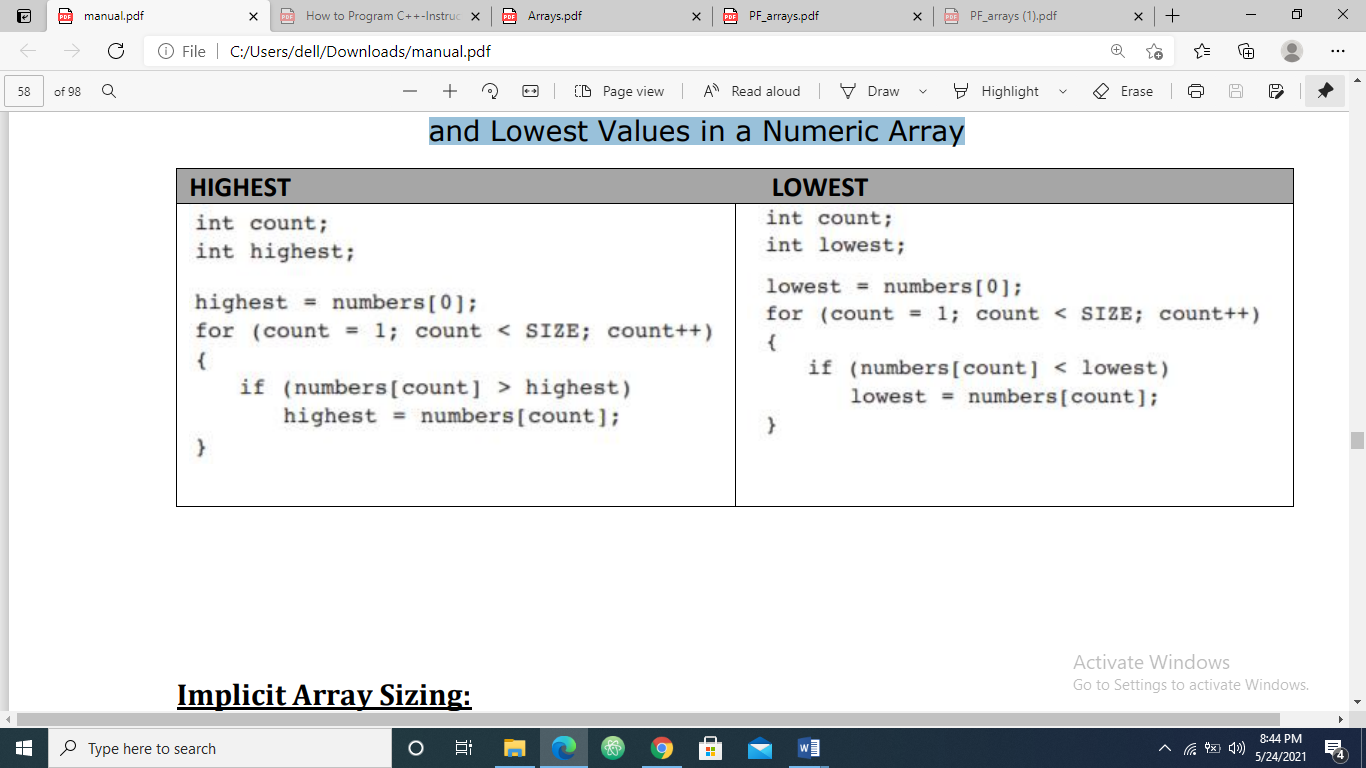
• To compare two arrays, you must compare element-by-element



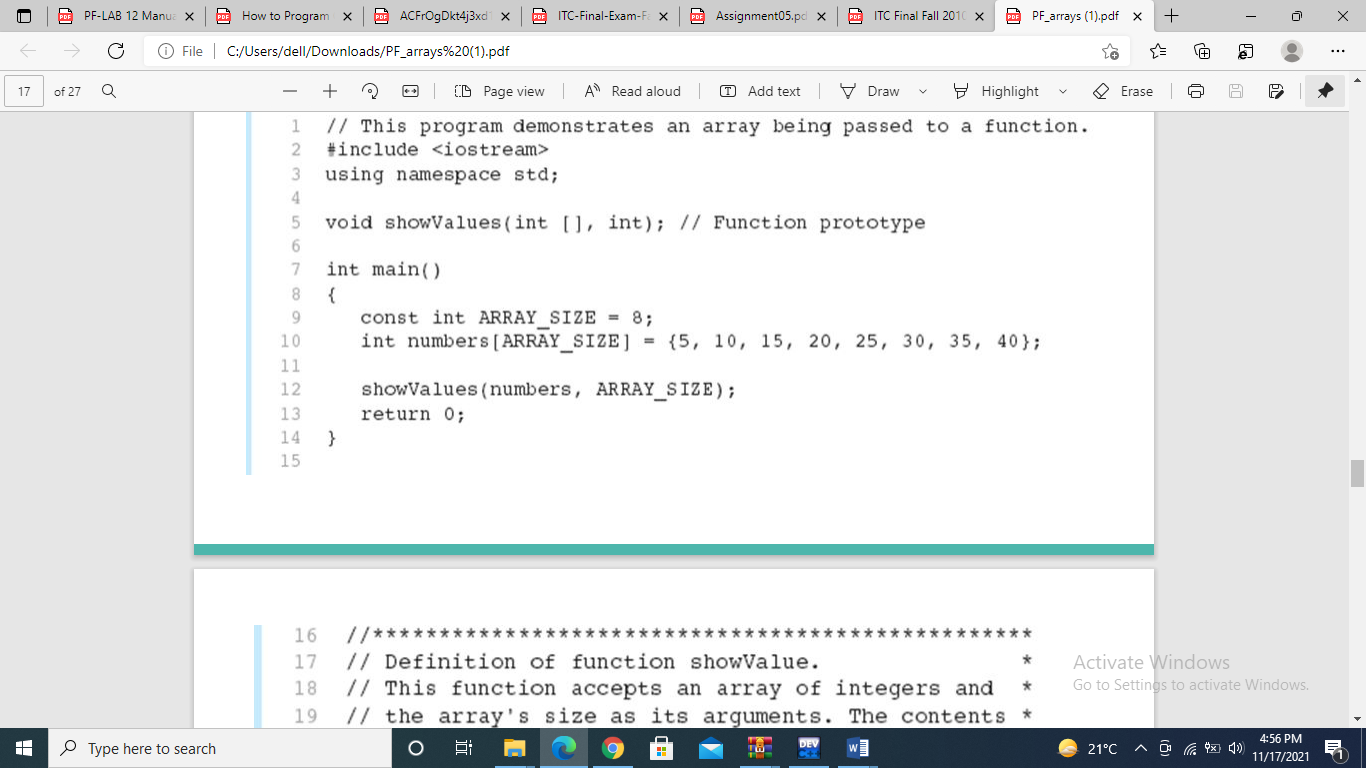
**Example:** The following example displays the sum of elements of array.

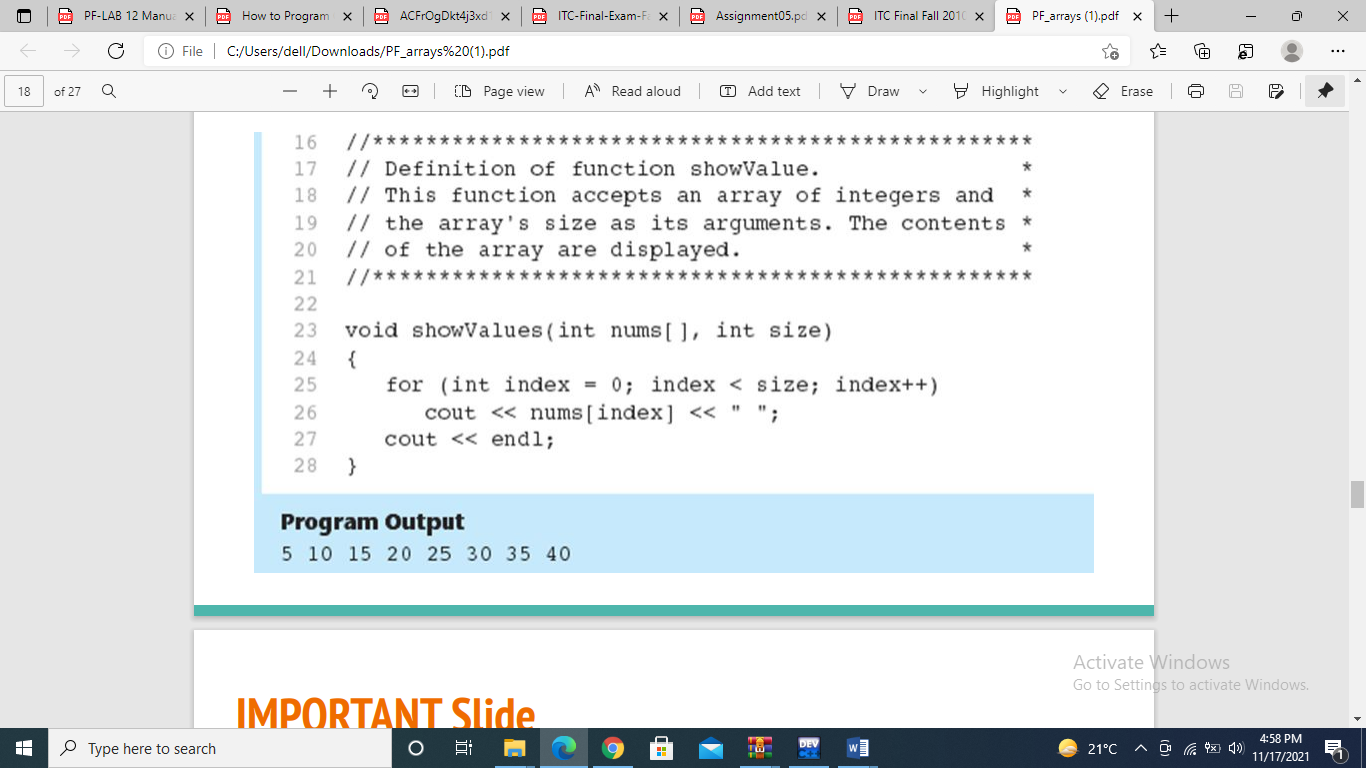


**Example:** The following example shows how to Find the Highest and Lowest Values in a Numeric Array



**Arrays as function argument:**





**Lab Tasks:**

**Submission instructions:**

* Submit **single .cpp** file containing code for all patterns.
* Rename your file as **i24XXXX\_LAB07.cpp** and then submit it on GCR.

**Problem#01:**

A traveler wants to travel from point A to point B and he has two path options (path1, path2). Find a new path with minimum cost from the beginning of any path to the end of any of the two paths. We can switch from one path to another path only at the common elements.  
The path costs are given:

Output: 20

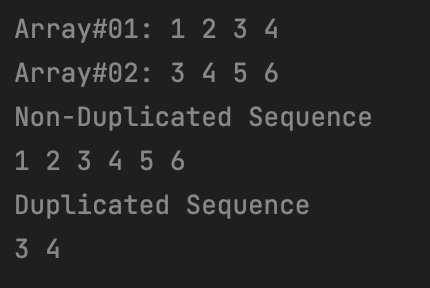
Explanation: The path will be 2+3+7+8= 20

**Problem#02:**

You need to declare two arrays. Input data into these two arrays from the users – both arrays must contain some similar items and some distinct items. Calculate and store the sizes of both the arrays in two different variables – again, avoid hardcoded values. Now, you need to define the following functions, consider these prototypes,

1. you should display only the duplicated values present in both the arrays.
2. You should display all the elements of both the arrays – duplicated values must be printed once.

Output:



**Problem#03:**

Define a floating array of length n (n must be greater than 10). The array would be passed to a function along with its calculated size and the function would traverse the entire array from starting item to the last item and check whether the item being traversed is prime or not.

1. If the item is prime, then item must be halved (divide by 2). And the halved item should be passed to another function where it would modify the passed value as per the following formula,

**float formula1 (float);**

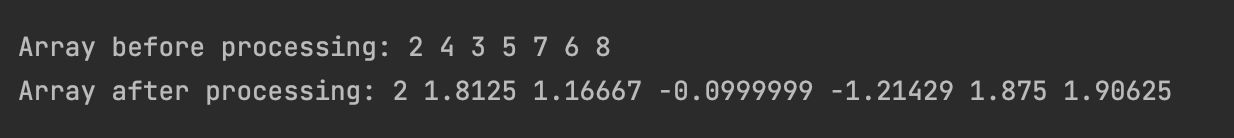
1. If the item is not prime, the item would not be halved in this case. The item would then be passed to another function where it would modify the passed value as per the following formula,

**float formula1 (float);**

1. A function that should check whether a received item is prime or not.

**bool isPrime (int);**

Output:



**Problem#04:**

Declared a character array of size n (n>10) and store the read input in it.

1. You have to create a function, if we pass the character array to it, the function will return another character array having all the vowels present in the passed array.

**string checkVowels (char []);**

1. Another function is needed to be created to which we will pass a single character and our character array, it will display the count of the character repeated in the array and will return true. If the character doesn’t exist, the function should return false.

**bool countChar (char [], char);**

Output:

