Home Work No.13(Arrays)

Review of last lecture:

Arrays:

- Are ***Grouped*** data. It have more than one data cells or ***elements***.
- All elements have of ***similar data*** type i.e., we can create an array of multiple elements with similar data type.
- Hold ***Contiguous/continuous/adjacent*** Memory locations
- Are **static** and ***fixed*** sized. These occupy memory at compile time. Always constants or literals are used to create an array.
- Each array maintains an internal addressing called ***indexing***. Each element of an array is assigned an index or subscript number. In C++ indices are started from zero to size-1. Do not confuse indexing with physical memory addresses these both are entirely different. Physical memory address where an array occupy memory.
- Indexing is one the ***power*** of any array which makes a programmer to access and use and array very easily and conveniently.
- ***Subscript operator []*** is used to access each or any element of an array.

Things to Remember:

- ✓ Array **size** is always be **constant** and of **integer** type.
- ✓ Each element/cell size of an array depends upon type of an array.
- ✓ In C++, indices are started from **zero** to **size-1**.
- ✓ Array name is **Constant Pointer** [i.e., it always point to the first element of an array]. We will discuss later on what is meant by constant pointer.
- ✓ When we print name of array its will print starting bytes physical/memory address of array (except character array where it prints content/data or character of array until null character]
- ✓ In C++ no bound checks so be careful during writing you code. Always write a loop from starting index i.e. zero to ending index i.e. size-1.
- ✓ Be careful of ***off-by-one*** error.

///...

/////// Questions from Some Previous Exams

Understand and then explain what is happening in all codes below.

1) What will be output of following code?

```
int main()
{

	int count = 0;
	int array1[] = { 119,101,108,108,32,100,111,110,101,33 };
	char array2[11];

	cout << "Size of integer array is " << sizeof(array1) << "    bytes." << endl;

	cout << "Size of Character array is " << sizeof(array2) << "  bytes." << endl;

	while (count <= 9)
	{
		array2[count] = array1[count];
		count++;
	}
	array2[count] = '\0';


	cout << array2 << endl;


}
```

2) What will be output of following code?

```
	int number = 3;
	int array1[20];
	for (int i = 0; i < 20; i++)
	{
		array1[i] = number;
		number += 3;
	}

	for (int i = 0; i < 20; i++)
	{
		if (array1[i] % 2 == 0 && array1[i] % 3 == 0 && array1[i] % 5 == 0)
		cout << array1[i] << " : This number is divisible by 2, 3 and 5. " << endl;
	}
```

3) What will be output of following code?

```
int main() {
	int nrows = 3, ncols = 4;
	int A[2][3][4] = { {

		{ 1, 3, 2 },
		{ 4, 5 },
		{ 7, 8, 9 } },
		{ { 4 },
		{ 5, 5, 7 },
		{ -2, 3, 4 } }
	};
	int b[2][4] = { { 0 } };
	for (int i = 0; i < 2; ++i) {

		for (int j = 0; j < ncols; ++j)
			for (int k = 0; k < nrows; ++k)
				b[i][j] += A[i][k][j];
	}
	for (int i = 0; i < 2; ++i) {
		for (int j = 0; j < nrows; ++j) {
			for (int k = 0; k < ncols; ++k)
				cout << A[i][j][k] << " ";
			cout << endl;
		}
		for (int j = 0; j < ncols; ++j)
			cout << b[i][j] << " ";
		cout << endl;
	}
	return 0;
}
```

What will be the output of following codes?

/////code-1//////////

```
int main() {
	int arr1[] = { 10,3,45,22,24,4 };

	cout << "\nBase address of array is  : " << arr1;
	cout << "\nBase address of array is  : " << &arr1[0];
	cout << "\nValue of First element of array is  : " << arr1[0];

	cout << "\nRemaining address of array are  : " << endl;

	for (int i = 1; i < 6; i++)
	{
		cout << "\nAddress of No. " <<i+1<<" Element is  :\""<<&arr1[i]<<"\" and " ;
		cout << "Value of  No. " << i+1 << " Element is  : " << arr1[i] << endl;
	}

	return 0;
}
```

/////code-2//////////

```
int main() {
	double darray[5] = { 10.5,3.2,4.5,2.2,2.4 };

	cout << "\nBase address of array is  : " << darray;
	cout << "\nBase address of array is  : " << &darray[0];
	cout << "\nValue of First element of array is  : " << darray[0];

	cout << "\nRemaining address of array are  : " << endl;

	for (int i = 1; i < 5; i++)
	{
		cout << "\nAddress of No. " <<i+1<<" Element is  :\""<<&darray[i]<<"\" and
" ;
		cout << "Value of  No. " << i+1 << " Element is  : " << darray[i] << endl;
	}

	return 0;
}
```

/////code-3/////////

///Code below demonstrate cube function, which I taught you people in lecture. Do what id //demanded in the comments.

```
void cubeArray(double[], int);//prototype for cube function for any double type array

int main() {
	double darray1[5] = { 10.5,3.2,4.5,2.2,2.4 }; // double array of size 5
	double darray2[7] = { 10.5,3.2,4.5,2.2,2.4, 1.1, 3.9 };// double array of size 7
	double darray3[] = { 2.4, 1.1, 3.9 };// double array of size 3

	cout << "\n Elements of darray1 before cubing \n";
	for (int i = 0; i < 5; i++)
		cout<<darray1[i]<<"  ";

	cubeArray(darray1, 5);

	cout << "\n Elements of darray1 after cubing \n";
	for (int i = 0; i < 5; i++)
		cout << darray1[i] << "  ";

	//////Write here valid calls for darray2 and darray3 for cubeArray function


	return 0;
}

void cubeArray(double arr[], int s)//Defination for cube function for any double type 
array
{

	for (int i = 0; i < s; i++)
		arr[i] = arr[i] * arr[i] * arr[i];

}
```

//

Some problems related arrays:

1. Write a C++ program that inputs a N size array from user. Sorts it using **bubble sort** and searches a value from it using **binary search**.
2. Suppose X, Y, Z are arrays of integers of size M, N, and M + N respectively. The elements in array X and Y appear in ascending order. Write a C++ program to produce third array Z by merging elements arrays X and Y in descending order.

   Example:

   Array X is {1,3,5,6}

   Array Y is {2,4,8}

   Array Z should be {8, 6, 5, 4, 3, 2, 1}

   Note : Do not use Merge then sort strategy. Array Z should be merged in sorted order.

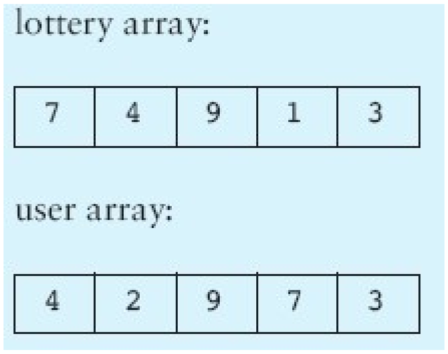
3. Write a program that uses the following arrays:
   - **empId** : an array of seven long integers to hold employee identification numbers. The array should be initialized with the following numbers:

     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 double s to hold each employee s hourly pay rate
   - **wages** : an array of seven double s to hold each employee s gross wages

   The 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 employees 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 6.00 for pay rate.

4. Write a program that simulates a lottery. The program should have an array of five integers named lottery , and should generate a random number in the range of 0 through 9 for each element in the array. The user should enter five digits which should be stored in an integer array named user . The program is to compare the corresponding elements in the two arrays and keep a count of the digits that match. For example, the following shows the lottery array and the user array with sample numbers stored in each. There are two matching digits (elements 2 and 4).



lottery array:

7	4	9	1	3

user array:

4	2	9	7	3

The program should display the random numbers stored in the lottery array and the number of matching digits. If all of the digits match, display a message proclaiming the user as a grand prize winner.

5. Write a C++ function that returns true if two-integer arrays passed into it have the same entries (even if they are not in the same order) otherwise false.

For example the following two arrays:

A = {1, 2, 3, 4, 5, 6} and B = {6, 3, 1, 2, 4, 5}

Your prints true because the two arrays have same entries (although the order is not same).

6. Suppose A and B are arrays of size M and N respectively. Write a C++ program that creates a third array C whose size is M + N and populates it such that the following sequence is followed.
   - All even number of A from left to right are copied into C from left to right
   - All odd numbers of A from left to right are copied into C from right to left.
   - All even numbers of B from left to right are copied into C from left to right.
   - All odd numbers of B from left to right are copied into C from right to left.

<u>Sample Input:</u>

A = {3, 2, 1, 7, 6, 3}

B = {9, 3, 5, 6, 2, 8, 10}

<u>Expected Outcome:</u>

C= {2, 6, 6, 2, 8, 10, 5, 3, 9, 3, 7, 1, 3}

7. Write a Menu Driven C++ program that creates a two-dimensional array/Matrix of size **3 X 3** and initialize it with user. The program should do following Tasks using Menu:

- **Total** : Calculate the total/sum of all the values in the array.
- **Average:** Calculates average of all the values in the array.
- **RowTotal**: Calculates total/sum of the values in the specified row.
- **ColumnTotal**: Calculates total/sum of the values in the specified column.
- **HighestInRow**: Finds highest value in the specified row of the array.
- **LowestInRow:** Finds lowest value in the specified row of the array.
- **Transpose:** Find Transpose of array.
- **LeftDiagonalTotal:** Calculates total/sum of the values in the left Diagonal of array.
- **RightDiagonalTotal: :** Calculates total/sum of the values in the right Diagonal of array.
- **Multiply:** Take another 3 X 3 array as input from user and Multiply both.

Note: Make all code separately and then merge them all in a menu. Use switch statement for menu.

8. Mean of matrix is the average of all the elements present in matrix. Consider a modified mean as floor of mean row-wise minimum and column-wise maximum. Row-wise minimum is the sum of minimum element from each row of given matrix and column-wise maximum is the sum of maximum element from each column. Given a matrix of order n *m, write a C++ program to find the number of elements greater than new mean obtained.

mean = floor((sum(row-wise Min) + sum(col-wise Max)) / (row_no. + col_no.)

Input:

int arr[3][3] = {{1,5,6},{2,3,0},{5,2,8}}

Expected Output:

Number of Elements: 4

Challenged question:

Write a function named ***reverseChunks*** that accepts three parameters, an array of integers a, its size and an integer "chunk" size s, and reverses every s elements of a. For example, if s is 2 and array a stores {1, 2, 3, 4, 5, 6}, a is rearranged to store {2, 1, 4, 3, 6, 5}. With an s of 3 and the same elements {1, 2, 3, 4, 5, 6}, array a is rearranged to store {3, 2, 1, 6, 5, 4}. The chunks on this page are underlined for convenience.

If a's length is not evenly divisible by s, the remaining elements are untouched. For example, if s is 4 and array a stores {5, 4, 9, 2, 1, 7, 8, 6, 2, 10}, a is rearranged to store {2, 9, 4, 5, 6, 8, 7, 1, 2, 10}.

It is also possible that s is larger than a's entire length, in which case the array is not modified at all. You may assume that s is 1 or greater (an s of 1 would not modify the array). If array a is empty, its contents should remain unchanged.

The following table shows some calls to your method and their expected results:

Arrays and Call	**Array Contents After Call**

`int a1[] = {20, 10, 30, 60, 50, 40};`
`reverseChunks(a1,6, 2);`
	`{10, 20, 60, 30, 40, 50}`

`int a2[] = {2, 4, 6, 8, 10, 12, 14, 16};`
`reverseChunks(a2,8, 3);`
	`{6, 4, 2, 12, 10, 8, 14, 16}`

`Int a3[] = {7, 1, 3, 5, 9, 8, 2, 6, 4, 10, 0, 12};`
`reverseChunks(a3,12, 5);`
	`{9, 5, 3, 1, 7, 10, 4, 6, 2, 8, 0, 12}`

`int a4[] = {1, 2, 3, 4, 5, 6};`
`reverseChunks(a4,6, 8);`
	`{1, 2, 3, 4, 5, 6}`

`int a5[]= {};`
`reverseChunks(a5,0, 2);`
	`{}`