**University of Michigan – Dearborn**

**CIS 200 – Computer Science 2**

**Lab# 02**

Nahrin Sharna

[nsharna@umich.edu](mailto:nsharna@umich.edu)

01/27/2019

# Windows Source Courses:

**Source file(Source.cpp) for main function:**

/\*Author: Nahrin Sharna

Creation Date: 01/25/2019

Modification Date: 01/25/2019

Purpose: To display an array and the total numbers of positive even numbers

\*/

#include<iostream>

using namespace std;

const int MAX\_ROWS = 3;

const int MAX\_COLUMNS = 2;

//funtion prototypes

int howManyEven(int [][MAX\_COLUMNS], int);

void printArray(int [][MAX\_COLUMNS], int);

void newPrintArray(int[][MAX\_COLUMNS], int);

int main() {

int arrayIntValues[MAX\_ROWS][MAX\_COLUMNS] = { {3 , 2}, {4, 5}, {2, 2} };

printArray(arrayIntValues, MAX\_ROWS);

cout << "Total even numbers: " << howManyEven(arrayIntValues, MAX\_ROWS) << endl;

newPrintArray(arrayIntValues, MAX\_ROWS);

system("pause");

return 0;

}

**Source(Source1.cpp) File for Other Functions:**

//Author: Nahrin Sharna

//Creation Date: 01/25/2019

//Modification Date: 01/26/2019

//Purpose: To print the current array values and total even numbers within the array

#include<iostream>

#include<iomanip>

using namespace std;

const int MAX\_ROWS = 3;

const int MAX\_COLUMNS = 2;

//Author:Nahrin Sharna

//Input: array a, ROWS: rows of array a

//Output: Total even numbers of array a

//Purpose: To return the total even numbers

int howManyEven(int a[][MAX\_COLUMNS], int ROWS) {

int count = 0;

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

for (int j = 0; j < MAX\_COLUMNS; j++)

{

if (a[i][j] % 2 == 0)

count++;

}

return count;

}

//Author: Nahrin Sharna

//Input: 2d array a, ROWS - rows of array a

//Output: Print the elements of array a with proper formatting

//Purpose: To display the elements of array a with proper formatting

void printArray(int a[][MAX\_COLUMNS], int ROWS) {

cout << setw(11) << "Columns" << setw(11) << "Columns"<< endl;

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

for (int j = 0; j < MAX\_COLUMNS; j++)

{

cout << setw(10) << a[i][j];

}

cout << endl;

}

}

/\*Author: Nahrin Sharna

Input: array a, ROWS - rows of array a

Output: Display the array with new elements and the total even numbers

Purpose: To ask user if they want to change the elements of the array

If yes, input the new values and print the array with new elements

If no, show a greeting message

\*/

void newPrintArray(int a[][MAX\_COLUMNS], int ROWS)

{

int value;

char choice;

do {

cout << "Would you like to enter a new set of values(Y or N): ";

cin >> choice;

switch (toupper(choice))

{

case 'Y':

{

cout << "Please enter a positive integer value: ";

cin >> value;

if (value >= 0) {

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

for (int j = 0; j < MAX\_COLUMNS; j++)

{

a[i][j] = value;

cout << "Please enter another positive value: ";

cin >> value;

}

}

}

if (value < 0)

{

cout << "Values can not be negative." << endl;

}

}

printArray(a, 3);

cout << "Total even numbers: " << howManyEven(a, 3) << endl;

break;

case 'N':

cout << "Thank You for using our software!" << endl;

break;

default:

cout << "Invalid choice." << endl;

}

} while (toupper(choice) != 'N');

}

**Windows Test Plans:**

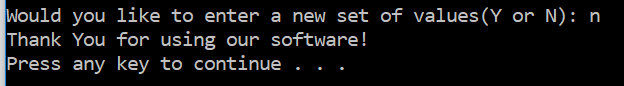
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1 | Valid | Ask User whether to input new sets of value | Y | Ask for the new numbers |  | Pass |
| 2 | Valid | Ask User whether to input new sets of value | n | Greeting message, program quits |  | Pass |
| 3 | Invalid | Ask User whether to input new sets of value | g | Show an error message |  | Pass |
| 4 | Invalid | When the new  integer is negative | -9 | Show an error message |  | Pass |
| 5 | Valid | When the user enters a positive value | 3,4,6,2,7,9,8 | Store the new value, keep asking for positive values until the condition met |  | Pass |

**Windows Test Screenshot:**

1. Choice ‘Y’:



1. Choice ‘N’:



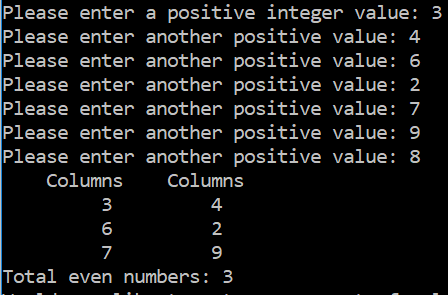
1. Invalid choice:



1. Negative Integer:



1. Positive Integer:



**Unix Test Plans:**

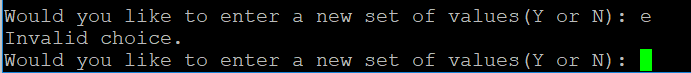
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1 | Valid | User like to change the values | y | Asking for new integer values | Asked for positive integer values | Pass |
| 2 | Invalid | User input invalid option after asking whether to change array elements | e | Error message | “Invalid choice” | Pass |
| 3 | Valid | User doesn’t want to change values | n | Show a message | “Thank you for using our software!” | Pass |
| 4 | Invalid | User input negative integer as a new element of array | -7 | An error message | “Values cannot be negative” | Pass |
| 5 | Valid | When the user enters positive value | 7,3,8,12,9,4,8 | Store all new values to the array | New values are assigned to the array | Pass |

**Unix Screen Shots:**

1. Choice ‘Y’:



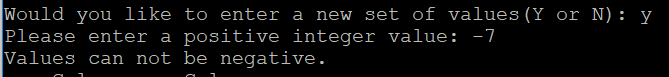
1. Invalid choice:



1. Choice ‘N’:



1. Negative Integer:



1. Positive Integer:

