1. **Problem Statement**

This program takes an array filled by user’s choice of positive numbers and tells the user the total number of even numbers in that array. It will also print out the whole array to show what elements were stored.

1. **Requirements**
   1. **Assumptions**
      1. All user inputs are integer data type
      2. User will enter “y” or “n” to tell if they want to restart
      3. User enters exactly 6 numbers followed by spaces in middle.
   2. **Specifications**
      1. Program will make an empty array that would be filled up by the user
      2. #define MAX\_ROWS 3
      3. #define MAX\_COLUMN 2
      4. User will enter the numbers to be added in the two-dimensional array
         1. Integers have to be positive
            1. Invalid Integer

Display an error message “Error. You entered a negative number. Enter a positive number”

* + - * 1. Valid Integer

Add it in the array

* + 1. Ask the user if they would like to restart the program by typing y for yes and n for no.
       1. Loop the program until user says n for no.

1. **Decomposition Diagram**

|  |  |  |
| --- | --- | --- |
| **Main** | | |
| **Input** | **Process** | **Output** |
| Positive numbers to fill up the array | Store positive numbers given by the user in the array |  |
|  | Calculate the # of even numbers in the array | Print out # of even numbers in the array |
| “y” if user wants to enter new set of values | Reset the array | Ask the user for new set of data again |
| “n” if user wants to quit the program |  | Print out Thank You message |

1. **Test Strategy**
   1. Valid Data
   2. Invalid Data
2. **Test Plan Version 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid Data | 1 | User enters number < 0 |  |  |  |  |
| Valid Data | 1 | User enters number > -1 |  |  |  |  |
| Valid Data | 2 | User enters same even number every time |  |  |  |  |
| Valid Data | 2 | User enters same odd number every time |  |  |  |  |

1. **Initial Algorithm**
   1. Create two global integer variables
      1. One for MAX\_ROWS, assigned to 3
      2. One for MAX\_COLUMNS, assigned to 2
   2. Create a function called *howManyEven* to find the total number of even elements in the array.
      1. Returns number of even elements as an integer.
   3. Create a function called *printArray* to print out the elements in the array.
   4. In main function
      1. Create and initialize array *arrayIntValues* to default values.
         1. {{3, 2}, {4,5}, {2,2}}
      2. Prompt the user to enter number: “Please enter 6 numbers one-by-one followed by spaces in the middle”. Loop until these inputs pass the checks.
         1. Check that all inputs are positive
         2. Check that there is at least one even number
      3. Store these inputs in the array and call the function *howManyEven* to store the number of even numbers to a variable.
      4. Call the function *printArray* following it with the variable that holds number of even numbers in the array.
      5. Prompt the user if he or she wants to enter a new set of values: “Would you like to enter new set of values?”
      6. Empty the array and set default values if last user input passes the check:
         1. Check that user entered “y” for yes and “n” for no.
2. **Test Plan Version 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid Data | 1 | User enters number < 0 | -10 | Error. Loop until user gives all # >= 0 |  |  |
| Valid Data | 1 | User enters number > -1 | 0 | Program takes the number and puts it in the array. No output other than even numbers and elements in array |  |  |
| Valid Data | 2 | User enters same even number every time | 4, 4, 4, 4, 4, 4 | Program takes the number and puts it in array without any other checks. No output other than 6 even numbers and elements in array |  |  |
| Valid Data | 2 | User enters same odd number every time | 3, 3, 3, 3, 3, 3 | Program takes the number and puts it in array without any other checks. No output and no even is printed when user wants to know how many even. |  |  |

1. **Code**

/\*

Author: Nisarg Patel

CIS200 WINTER 2018

Creation Date: 1-29-2017

Modification Date: 2-01-2018

Purpose: This program takes an array filled by user’s choice of

positive numbers and tells the user the total number of even

numbers in that array. It will also print out the whole array

to show what elements were stored.

\*/

#include <iostream>

#include <string>

using namespace std;

#define MAX\_ROWS 3 //global variable for maximum rows in array

#define MAX\_COLUMNS 2 //global variable for maximum columns in array

int howManyEvens(int arrayIntValues[MAX\_ROWS][MAX\_COLUMNS]); //Funcion prototype to return the # of even numbers

void printArray(int arrayIntValues[MAX\_ROWS][MAX\_COLUMNS]); //Function prototype to print out the elements in the array

bool positiveInt(int tempUserInput[MAX\_ROWS][MAX\_COLUMNS]); //Function prototype to return true if all user input numbers are positive

int main()

{

cout << "Greetings, user. Welcome to an amazing program that tells you \nhow many even numbers are in a group of numbers" << endl << endl;

string restart = "y";

do {

int userInput;

bool status; //all positive = true; atleast 1 negative = false

int arrayIntValues[MAX\_ROWS][MAX\_COLUMNS] = { { 3,2 },{ 4,5 },{ 2,2 } }; //Default initialization of array

int tempUserInput[MAX\_ROWS][MAX\_COLUMNS]; //array to be used before checking if all values are positive or negative

cout << "Please enter 6 numbers one-by-one followed by spaces in the middle: " << endl;

do { //Loop until user wants to quit

for (int row = 0; row < MAX\_ROWS; row++)

{

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

{

cin >> tempUserInput[row][col]; //Stores all the numbers in a temperory array

}

}

status = positiveInt(tempUserInput); //status is either true or false; positive or negative

if (status) //If all positive...

{

for (int row = 0; row < MAX\_ROWS; row++)

{

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

{

arrayIntValues[row][col] = tempUserInput[row][col]; //All the data in temperory array are assigned to original array

}

}

cout << "Even numbers: " << howManyEvens(arrayIntValues) << endl; //Prints out total numbers of even numbers

printArray(arrayIntValues); //Prints all the elements in the array

break;

}

else {

cout << "You have entered at least 1 negative number." << endl << "Please try again: " << endl; //If all numbers are not positive, tells the user to try again

}

} while (status == false); //Keeps running until there is atleast one negative number in the array

cout << "Would you like to restart the program? Type y for yes and n for no: " << endl; //Asks user if they want to restart again

cin >> restart;

} while (restart == "y"); //Keeps running until user respond is "n"

cout << "Thanks for using this program." << endl;

system("pause");

return 0;

}

int howManyEvens(int arrayIntValues[MAX\_ROWS][MAX\_COLUMNS]) //Calculates total number of evens in the input array

{

int evenCounter = 0;

for (int row = 0; row < MAX\_ROWS; row++)

{

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

{

if ((arrayIntValues[row][col] % 2) == 0) //As long as each number in the array has a remainder of 0 when devided by 2...

{

evenCounter++; //...variable is incremented

}

}

}

return evenCounter; //variable is returned

}

void printArray(int arrayIntValues[MAX\_ROWS][MAX\_COLUMNS]) //Function prototype to print out the elements in the array

{

for (int row = 0; row < MAX\_ROWS; row++)

{

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

{

cout << arrayIntValues[row][col] << " ";

}

cout << endl;

}

}

bool positiveInt(int tempUserInput[MAX\_ROWS][MAX\_COLUMNS]) //Function prototype to return true if all user input numbers are positive

{

bool status = true;

for (int row = 0; row < MAX\_ROWS && status != false; row++)

{

for (int col = 0; col < MAX\_COLUMNS && status != false; col++)

{

if ((tempUserInput[row][col] >= 0)) //Each element is checked...if it is positive...

{

status = true; //variable stays true

}

else {

status = false; //If even 1 variable is negative, status changes to false and breaks out the loop

break;

}

}

}

return status;

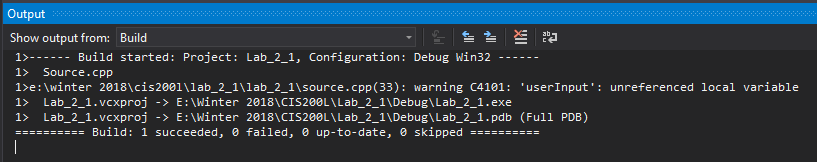
}

1. **Updated Algorithm**
   1. Create two global integer variables
      1. One for MAX\_ROWS, assigned to 3
      2. One for MAX\_COLUMNS, assigned to 2
   2. Create a function called *howManyEven* to find the total number of even elements in the array.
      1. Returns number of even elements as an integer.
   3. Create a function called *printArray* to print out the elements in the array.
   4. Create a function called *positiveInt* to return true if all user input numbers are positive
   5. In main function
      1. Greetings
      2. Create and initialize array *arrayIntValues* to default values.
         1. {{3, 2}, {4,5}, {2,2}}
      3. Prompt the user to enter number: “Please enter 6 numbers one-by-one followed by spaces in the middle”. Store values in temporary array. Loop until these inputs pass the checks.
         1. Check that all inputs are positive by calling the function *positiveInt*
         2. ~~Check that there is at least one even number~~
      4. Store these inputs in the array and call the function *howManyEven* to store the number of even numbers to a variable if
         1. All elements are positive
      5. Call the function *printArray* following it with the variable that holds number of even numbers in the array.
      6. Prompt the user if he or she wants to enter a new set of values: “Would you like to enter new set of values?”
         1. Loop until user wants to quit.
      7. Thank you message
2. **Test Plan Version 3**

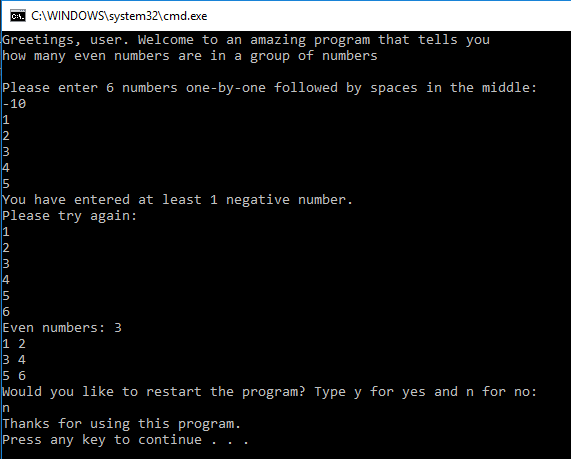
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid Data | 1 | User enters numbers < 0 | -10, 1, 2, 3, 4, 5 | Error. Loop until user gives all # >= 0 | Error. Loop until user gives all # >= 0 | Pass |
| Valid Data | 1 | User enters numbers > -1 | 0, 1, 2, 3, 4, 5 | Program takes the number and puts it in the array. No output other than even numbers and elements in array | Program takes the number and puts it in the array. No output other than even numbers and elements in array | Pass |
| Valid Data | 2 | User enters same even number every time | 4, 4, 4, 4, 4, 4 | Program takes the number and puts it in array without any other checks. No output other than 6 even numbers and elements in array | Program takes the number and puts it in array without any other checks. No output other than 6 even numbers and elements in array | Pass |
| Valid Data | 3 | User enters same odd number every time | 3, 3, 3, 3, 3, 3 | Program takes the number and puts it in array without any other checks. No output and no even is printed when user wants to know how many even. | Program takes the number and puts it in array without any other checks. No output and no even is printed when user wants to know how many even. | Pass |

1. **Screenshots**

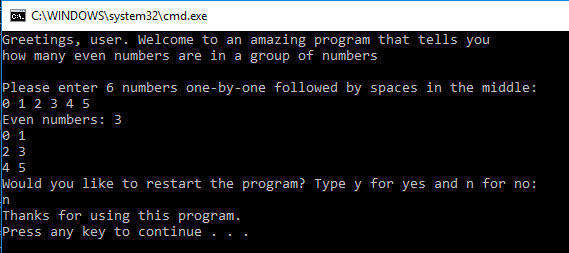
Build (Windows):



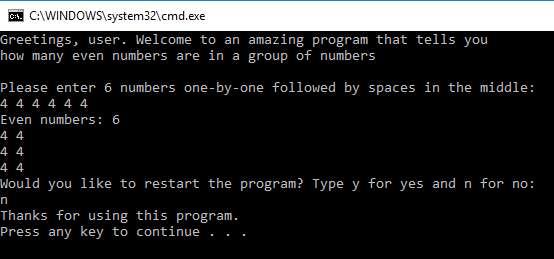
Invalid Test Case 1 (Windows):



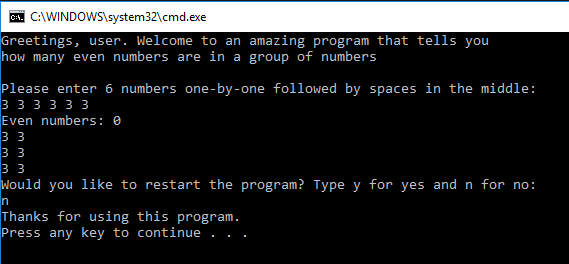
Valid Test Case 1 (Windows):



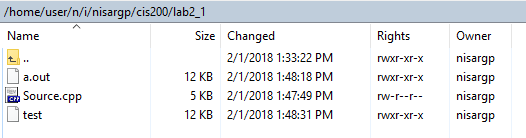
Valid Test Case 2 (Windows):



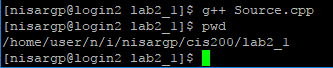
Valid Test Case 3 (Windows):



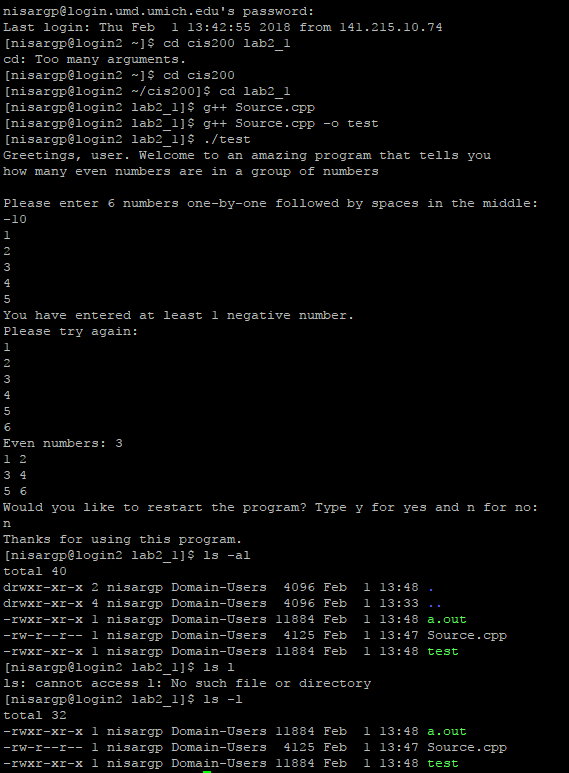
Directory (Linux):



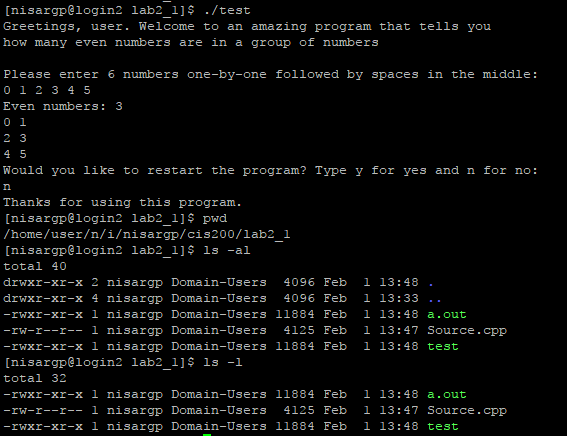
Compile using *g++* (Linux):



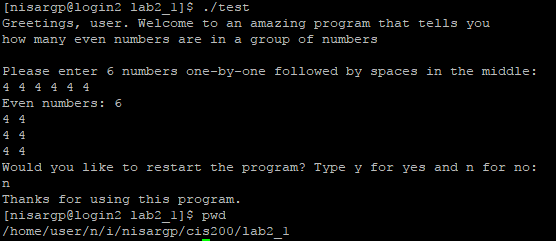
Invalid Test Case 1 (Linux):



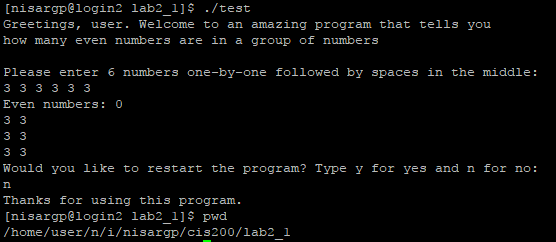
Valid Test Case 1 (Linux):



Valid Test Case 2 (Linux):



Valid Test Case 3 (Linux):



1. **Status**

It works with assumption in mind.