Apol Medrano

Heritage University 09 -25 -23

Mathematics and Computer Science Dept. Due: 10-01-23

C++

Homework 5

*(functions, file, arrays )*

*General:*

1**.** Find the number of zeros that are in two dimensional array using function

Data: {2,0,1},{3,4,3},{0,0,0},{7,0,3}

//Apol Medrano

//HW5 General Q1

//Find the zeros in 2-D array

#include <iostream>

using namespace std;

const int index1SIZE= 4, index2SIZE = 3;

//sets data indexes as global constants, so it can easily be used throughout program

int Search(int a[][index2SIZE]);

//takes 2-D array, searches for amount of zeros within it

//returns the zero count to main

int main()

{

int data[index1SIZE][index2SIZE] = {2,0,1,3,4,3,0,0,0,7,0,3}; //the 2-D array values

cout << endl << "The number of zeros that are in this two-dimensional array is "

<< Search(data) << "." << endl; //displays to the user the amount of zeros

return 0; //that are in the 2-D array

}

int Search(int a[][index2SIZE])

{

int ZeroCount = 0;

for (int index1 = 0; index1 < index1SIZE; index1++)//loops through index1

{

for (int index2 = 0; index2 < index2SIZE; index2++)//loops through index2

{

cout << a[index1][index2] << ", "; //displays sequence to user

if (0 == a[index1][index2])//counts the amount of zero that appear

ZeroCount++; //everytime the program loops to a new 2-D array index

}

}

return ZeroCount;

}

**GENERAL Q1 OUTPUT:**

**A screenshot of a computer

Description automatically generated**

2. Write a program that erase the sequence “rd” from a word like “border” using function.

//HW5 General Q2 Apol Medrano

#include <iostream>

#include <string>

using namespace std;

// Function to define a word the user inputs

string removeSequence(string word) {

string result = "";

int length = word.length();

// process to remove 'rd' sequence from a word

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

if (i < length - 1 && word[i] == 'r' && word[i + 1] == 'd') {

i++;

} else {

result += word[i];

}

}

return result;

}

// what the user will see on their end.

int main() {

string inputWord;

cout << "Enter a word: ";

cin >> inputWord;

string result = removeSequence(inputWord);

cout << "Word after removing 'rd': " << result << endl;

return 0;

}

**GENERAL Q2 OUTPUT:**

**A screenshot of a computer

Description automatically generated**

*Pattern Recognition:*

Prepare a program that finds four consecutive numbers with the same value in an array.

//Apol Medrano

//HW 5 Pattern Recognition

//Finds four consecutive numbers

#include <iostream>

const int arraySize = 10;

void Search(int array[], int numberUsed);

//takes array values and the length of array

//displays a message only if their is a number that shows up consecutively 4 times

using namespace std;

int main()

{

int next, numberUsed, index = 0, Array[arraySize];

cout << "Enter up to " << arraySize << " non-negative whole numbers." << endl

<< "Mark the end of the list with a negative number." << endl;

cin >> next;

while (next >= 0 && index < arraySize)

{

Array[index] = next;

index++;

if (index < 10)

cin >> next;

}

numberUsed = index;

Search (Array, numberUsed);

}

void Search(int array[], int numberUsed)

{

int ConsecutiveCount = 1, old = array[0]; //this is done so that there is already an old number

for (int index = 1; index < numberUsed; index++)//loops through index1

{

if (old == array[index])//old index number must be equal to new one

{

ConsecutiveCount++; //if so it causes the consecutive count to go up

if (ConsecutiveCount == 4)

{

cout << "Number that shows up 4 times consecutively is " << array[index] << endl;

ConsecutiveCount = 1; //shows user the number that was consecutively

}// then goes back to starting point so message wont get repeated

}

else

{

ConsecutiveCount = 1; //so if a streak ends, it goes back to zero

} //instead of stacking when a new streak begins

old = array[index]; //the current number becomes the old number and then the cycle is looped

}

}

**PATTERN RECOGNIZATION OUTPUT:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

*Searching*

1.Prepare a program that randomly filles in 0s and 1s into a 4x4 array, print it and find the first row and first col with the most 1s.

//Apol Medrano

//HW5 Searching Q1

//randomly puts 0s & 1s in array & finds first max of rows & collumns

#include <iostream>

#include <cstdlib> //random number generator

#include <time.h> //time

#include <iomanip> //setw

using namespace std;

const int index1SIZE = 4, index2SIZE = 4;

//declares the size of arrays as global variables

void fillArray (int Rand[][index2SIZE]);

//takes in array

//fills in array with randomly generated 1s and 0s

void findRowsSum (const int Rand[][index2SIZE], int Rows[]);

//takes in array searches each row of

//adds up each value within a row puts them in another array

void findCollumnsSum (const int Rand[][index2SIZE], int Collumns[]);

//takes in array searches each collumn of

//adds up each value within a collumn puts them in another array

void PrintResults (int Rand[][index2SIZE], int Rows[], int Collumns[]);

//takes all the arrays in the program

//prints the results to user in the output of the program

void firstMax (int Rows[], int Collumns[]);

//takes in the row and collumn arrays

//finds the first largest collumn/row in the array

int main ()

{

int Rand[index1SIZE][index2SIZE], Rows[index1SIZE], Collumns[index2SIZE];

fillArray (Rand); //rand array filled

findRowsSum(Rand, Rows); //rows array filled

findCollumnsSum(Rand, Collumns); //collum array filled

PrintResults(Rand, Rows, Collumns); //results coded to print to user

}

void fillArray (int Rand[][index2SIZE])

{ //makes sure that there a is a new random number generated

srand(time(0));

for (int index1 = 0; index1 < index1SIZE; index1++)

{

for (int index2 = 0; index2 < index2SIZE; index2++)

{ //generates random number between 0 and 1

Rand[index1][index2] = rand() % 2;

}

}

}

void findRowsSum (const int Rand[][index2SIZE], int Rows[])

{ //index1 has index2 nested, so a row is being seached for every value within it

for (int index1 = 0; index1 < index1SIZE; index1++)

{

int sum = 0; //restarts sum value after values within row end

for (int index2 = 0; index2 < index2SIZE; index2++)

{ //adds up the values with in a row

sum += Rand[index1][index2];

Rows[index1] = sum;

//saves sum value of row to Rows array

}

}

}

void findCollumnsSum (const int Rand[][index2SIZE], int Collumns[])

{//index2 has index2 nested, so a collumn is being seached for every value within it

for (int index2 = 0; index2 < index2SIZE; index2++)

{

int sum = 0; //restarts sum value aftere

for (int index1 = 0; index1 < index1SIZE; index1++)

{ //adds up the values in a collumn

sum += Rand[index1][index2];

Collumns[index2] = sum;

} //saves sum value of row to Collumns array

}

}

void PrintResults (int Rand[][index2SIZE], int Rows[], int Collumns[])

{

cout << "Randomly Generated 0s & 1s" << endl;

for (int index1 = 0; index1 < index1SIZE; index1++)

{

cout << setw(10);

for (int index2 = 0; index2 < index2SIZE; index2++)

{

cout << Rand[index1][index2] << setw(2);

}

cout << endl;

}

cout << endl;

firstMax (Rows,Collumns); //prints first max of each

}

void firstMax (int Rows[],int Collumns[])

{

int max = Rows[0], maxLocation = 0;

//starts loop off with a max so it can work properly

for (int index1 = 1; index1 < index1SIZE; index1++)

{

if (max < Rows[index1]) //if another row larger

{ //it will become new max and mark a new maxLocation

max = Rows[index1];

maxLocation = index1;

}

}

cout << "The largest row that shows up first is Row " << (maxLocation + 1)

<< "." << endl; //must raise location by so it makes sense to user

//ex: Row 0 wouldn't make sense to every user

max = Collumns[0];

maxLocation = 0;

//same process as before but for collumn array

for (int index2 = 1; index2 < index2SIZE; index2++)

{

if (max < Collumns[index2])

{

max = Collumns[index2];

maxLocation = index2;

}

}

cout << "The largest collumn that shows up first is Collumn " << (maxLocation + 1)

<< "." << endl;

}

**Down below is output**

**SEARCHING Q1 OUTPUT:**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

2.Find the location of the largest element of a 3x4 array.

//Apol Medrano

//HW5 Searching Q2

//finds largest element in a 3x4 array

#include <iostream>

#include <cstdlib> //random number generator

#include <time.h> //time

#include <iomanip> //setw

using namespace std;

const int index1SIZE = 3, index2SIZE = 4;

//declares the size of arrays as global variables

void fillArray (int Rand[][index2SIZE]);

//takes in array

//fills in array with randomly generated number between 10 and 99

void PrintResults (int Rand[][index2SIZE]);

//takes in the array

//prints the results to user in the output of the program

void locationOfLarge(int Rand[][index2SIZE]);

//takes in array

//finds the exact location of the largest element

int main ()

{

int Array [index1SIZE][index2SIZE], Rows[index1SIZE], Collumns[index2SIZE];

fillArray (Array); //array filled

PrintResults(Array); //results coded to print to user

}

void fillArray (int Array[][index2SIZE])

{ //makes sure that there a is a new random number generated

srand(time(0));

for (int index1 = 0; index1 < index1SIZE; index1++)

{

for (int index2 = 0; index2 < index2SIZE; index2++)

{ //generates random number between 0 and 99

Array[index1][index2] = rand() % 100;

}

}

}

void findLargeLocation (const int Array[][index2SIZE])

{

int max = Array[0][0], maxRowLocation = 0, maxCollumnLocation = 0;

for (int index1 = 0; index1 < index1SIZE; index1++)

{

for (int index2 = 0; index2 < index2SIZE; index2++)

{

if (max < Array[index1][index2])

{

max = Array[index1][index2];

maxRowLocation = index1;

maxCollumnLocation = index2;

}

}

}

cout << "The largest element is at Row " << (maxRowLocation + 1)

<< " and at Collumn " << (maxCollumnLocation + 1) << ".";

}

void PrintResults (int Array[][index2SIZE])

{

cout << "Randomly Generated Numbers" << endl;

for (int index1 = 0; index1 < index1SIZE; index1++)

{

cout << setw(10);

for (int index2 = 0; index2 < index2SIZE; index2++)

{

cout << Array[index1][index2] << " ";

}

cout << endl;

}

cout << endl;

findLargeLocation (Array);

}

**SEARCHING Q2 OUTPUT:**

**A screenshot of a computer

Description automatically generated**

*A screenshot of a computer

Description automatically generated*

*Cryptography*

*Monoalphabetic Ciphers*

For the character ciphers a message (plaintext) is encrypted into cyphertext by translating letters into numbers. The English alphabet contains 26 letters and translate them into integers from 0 to

25. Each character is changed individually to another letter by substitution. There are 26! possible ways to produce a monographic transformation. On the shift transformation *(Caeser cipher)* the k is the key representing the size of shift of letters in the alphabet and C≡P(mod26).(p.37)

More generally , we will consider the affine transformations :

C≡ aP + b(mod26), 0≤C≤25 where a and b are integers with (a,26)=1

To encrypt a message using transformation, we first change it to numerical equivalent, grouping letters in blocks of five. Then we transform each number. The grouping of letters into blocks helps to prevent successful cryptoanalysis based on recognizing particular words.

Using the frequency of letters in an encrypted message we can transform it to the corresponding plaintext. Counting of the letters of the cyphertext is necessary for the decryption of the message.

*Input:* several (≤3) lines of text and prints a table indicating the number of occurrences of each different letter in the text.

The message :

IVQLM IQATO SMIKP QTLVW

VMQAJ MBBMZ BPIVG WCZWE

VNZWU KPQVM AMNWZ BCVMK WWSQM

was encrypted using a shift transformation C≡P +b (mod 26)

*Output:* Write a program that reads several (≤3) lines of text and prints:

a. a table indicating the number of occurrences of each different letter in the text. you can have the format: Letter, counts.

b. what is the plaintext message?

*Hint* : for (b) the equation P=C-8(mod26) can give direct the answer. (C: the encrypted letters , P: the plaintext letters)

//Apol Medrano

//HW5 Cryptography

//takes ciphered text and prints

//prints amount letters in ciphered text

//prints out deciphered text

#include <iostream>

#include <string>

#include <cctype> //toupper

#include <fstream> //file opening

using namespace std;

void SortLetters (string& Decrypted);

//takes in string and sorts them alphabetic order

//using bubblesort method

void NumberOfLetters (const string Decrypted);

//counts the number of letters

int main ()

{ //open files that contains encrypted text

ifstream InStream("Encrypted.txt");

string textFromFile, PlainText, DeLetterCount, Encrypted;

do { //gets text from file turns it into string

getline(InStream,textFromFile);

for (int index = 0; index < textFromFile.length(); index++)

{

char letter = toupper(textFromFile[index]);

if (letter != ' ')

{

Encrypted += letter;

DeLetterCount += letter;

PlainText += char(int(letter - 8 + 65 )% 26 + 65);

} //formula to turn encrypted to decrypted text

else

{

Encrypted += letter;

DeLetterCount += ""; //every space in text is ommited to make sorting later easier

PlainText += " "; //every space in text appears in final text

}

}

} while (!InStream.eof());

cout << " Encrypted Text: \n" << Encrypted << endl << endl;

SortLetters(DeLetterCount);

NumberOfLetters(DeLetterCount);

cout << endl << " Plaintext Message: \n" << PlainText;

}

void SortLetters (string& DeLetterCount)

{ //sorts letters using bubble sort method

for (int i = DeLetterCount.length()-1; i > 0; i--)

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

if (DeLetterCount[index] > DeLetterCount[index+1])

{

// Swap the numbers

int temp = DeLetterCount[index+1];

DeLetterCount[index+1] = DeLetterCount[index];

DeLetterCount[index] = temp;

}

}

void NumberOfLetters (const string DeLetterCount)

{

cout << " Letter Count\n";

int WordCount = 1;

for (int index = 1; index < DeLetterCount.length()+1; index++)

{

char letter = DeLetterCount[index];

if (DeLetterCount[index-1] == letter)//old index number must be equal to new one

{

WordCount++; //if so it causes the consecutive count to go up

}

else

{

cout << " [ " << DeLetterCount[index-1]

<< " ][ " << WordCount << " ]\n";

WordCount = 1; //so if a streak ends, it goes back to zero

} //instead of stacking when a new streak begins

}

}

**CRYPTOGRAPHY OUTPUT:**

**A screenshot of a computer

Description automatically generated**

**TEXT FILE USED IN PROGRAM “Encrypted.txt”:**

**A screenshot of a computer

Description automatically generated**