***אביחי חדאד 209286665***

***אלעד פישר 318882800***

תרגיל 1 – אורך המופע השלילי הארוך ביותר

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

int getLongestNegative(int[], int);

int isArrEmpty(int[], int);

#define N 8

#define TRUE 1 // defining true/false for readability

#define FALSE 0

int main()

{

int arr[N], isInputValid, i;

int tmpCounter, maxCounter = 0;

do { // running until array is full of zeros.

isInputValid = FALSE; // a flag to exit inside loop if user try to enter char to array.

while (!isInputValid) { // looping if user enters a char to array.

isInputValid = TRUE;

printf("Enter the array: ");

for (i = 0; i < N; i++)

if (scanf("%d", &arr[i]) != 1) // checking if invalid input.

isInputValid = FALSE;

if (isInputValid == FALSE) {

rewind(stdin); // cleaning buffer of keyboard after user inserts a char.

printf("InputError: please enter %d INTEGERS seperated by spaces!\n\n", N);

}

}

tmpCounter = getLongestNegative(arr, N);

// "maxCounter" holds always the biggest negative sequence possible.

maxCounter = (maxCounter < tmpCounter) ? tmpCounter : maxCounter;

} while (!isArrEmpty(arr, N));

printf("Maximal negative sequence length is: %d\n", maxCounter);

return 0;

}

/\* Function gets a pointer to an array and it's size

and returns wethere the array is full of zeros or not. \*/

int isArrEmpty(int arr[], int size)

{

int i;

for (i = 0; i < size; i++) // looping the array.

if (arr[i] != 0)

return FALSE;

return TRUE; // logical else.

}

/\* Function gets a pointer to an array and it's size

and returns the size of the longest negative sequence of that array. \*/

int getLongestNegative(int arr[], int size)

{

int i, tmpCounter = 0, counter = 0;

for (i = 0; i < size; i++) { // looping the array.

if (arr[i] < 0)

tmpCounter++;

else {

// inserting counter the max("counter", "tmpCounter")

counter = (counter < tmpCounter) ? tmpCounter : counter;

tmpCounter = 0;

}

}

// re-assigning counter to the last sequence size is in the end of the array if it's bigger.

counter = (counter < tmpCounter) ? tmpCounter : counter;

return counter;

}

פלטים

|  |  |
| --- | --- |
| הרצה | פלט |
| א |  |

תרגיל 2 – הזזה מעגלית שמאלה

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <math.h>

void printArray(long arr[], int size);

int getFirstDigit(long num);

void buildNum(long\* src, int firstdigit);

void leftCircularShift(long arr[], int size);

#define N 4

#define TRUE 1 // defining true/false for readability

#define FALSE 0

int main()

{

int i, isInputValid;

long arr[N];

isInputValid = FALSE; // a flag to exit inside loop if user try to enter char to array.

while (!isInputValid) { // looping if user enters a char to array.

isInputValid = TRUE;

printf("Please enter an array of %d long values: ", N);

for (i = 0; i < N; i++)

if (scanf("%d", &arr[i]) != 1) // checking if invalid input.

isInputValid = FALSE;

if (isInputValid == FALSE) {

rewind(stdin); // cleaning buffer of keyboard after user inserts a char.

printf("InputError: please enter %d LONGS seperated by spaces!\n\n", N);

}

}

printf("\nArray before shift: ");

printArray(arr, N);

leftCircularShift(arr, N);

printf("\nArray after shift: ");

printArray(arr, N);

printf("\n");

return 0;

}

/\* Function gets a pointer to array and it's size

and prints all elements of array in one line. \*/

void printArray(long arr[], int size)

{

int i;

for (i = 0; i < size; i++)

printf("%d ", arr[i]);

}

// Function gets a long number and returns the first digit(leftest)

int getFirstDigit(long num)

{

/\* (int)log10(number) + 1 is equal to numberOfDigits.

calculating by using pow with log10,

(e.x: for num = 234: divder = 10 ^ 2 = 100 )) \*/

long divider = (long) pow(10, (int) log10(num));

// returning one digit(e.x: (int) 234 / 100 = 2).

return (int) num / divider;

}

/\* Function get a pointer to a long number and a digit

and adds the "firstdigit" to the right of the long number. \*/

void buildNum(long \*src, int firstdigit)

{

// casting to long because the modulo operand afterwards.

long divider = (long) pow(10, (int) log10(\*src));

// deleting first digit.

\*src = \*src % divider;

// adding digit to the right.

\*src \*= 10;

\*src += firstdigit;

}

/\* Function gets a pointer to an array and it's size,

and doing the leftCircularShift operation on the array. \*/

void leftCircularShift(long arr[], int size)

{

int i;

/\* declaring an int that will hold the leftest digit

of the first element in the array. \*/

long firstDigitOfFirst;

for (i = 0; i < size - 1; i++) { // iterating the array.

if (i == 0)

firstDigitOfFirst = getFirstDigit(arr[i]);

// building number.

buildNum(&arr[i], getFirstDigit(arr[i + 1]));

}

// building last number with "firstDigitOfFirst" variable.

if (size > 1)

buildNum(&arr[size - 1], firstDigitOfFirst);

}

פלטים

|  |  |
| --- | --- |
| הרצה | פלטים |
| א |  |
|
|
|
|

תרגיל 3– בניית היסטוגרמה

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

void createHistogram(char\* str);

void printHistogram(int[], int);

#define N 4

#define STR\_SIZE 40

int main()

{

char str[STR\_SIZE], chr;

int i;

printf("Enter your string: ");

// iterating until user enter a '!' and there is still free space in string.

for (i = 0; i < STR\_SIZE && str[i - 1] != '!'; i++) {

scanf("%c", &str[i]);

}

if (str[i - 1] != '!') { // if last char in string is not '!'.

printf("InputError: Histogram could not be created.\n");

printf("Invalid ending of string with max size of %d, should be ended with '!'\n", STR\_SIZE);

return 1;

}

createHistogram(str);

return 0;

}

/\* Function gets a pointer to a string

and builds a counter array of size N of occurrences of chars in string,

and print the result by calling anther function. \*/

void createHistogram(char \*str)

{

int cntArray[N] = { 0 }, indexInAZ;

// iterating over the string until \*str == '!'.

for (; \*str != '!'; str++) {

indexInAZ = \*str - 'a'; // calculating index of \*str in a-z.

if (indexInAZ >= 0 && indexInAZ < N) // checking if valid to insert into histogram.

cntArray[indexInAZ]++;

}

printHistogram(cntArray, N);

}

/\* Function gets a pointer to array and it's size,

and prints the array according to the requirements. \*/

void printHistogram(int arr[], int size)

{

int i, j;

for (i = 0; i < size; i++) { // itereting over elements in arr.

printf("%c: ", i + 'a'); // prints the char of that row.

for (j = 0; j < arr[i]; j++) // itereting arr[i] times and adding \* each iteration.

printf("\*");

printf("\n"); // space between rows.

}

}

פלטים

|  |  |
| --- | --- |
| הרצה | פלטים |
| א |  |
|
|
|
|