**Лабораторная работа № 7**

Задача 1

Постановка задачи:

Вычислить минимальное значение (одно) из положительных элементов каждого из двух массивов чисел M[11] и N[11], в которых могут присутствовать отрицательные числа

Математическая модель:

Если (a[i] < min && a[i] > 0)

min = a[i]

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| n | const | Размер массивов (11) |
| searchMinimum | int | Функция поиска минимального положительного числа в массиве |
| i | int | Параметр цикла |
| size | int | Аргумент пользовательских функций (размер массива) |
| arr | int\* | Аргумент функции searchMinimum / возвращаемый массив функции fillAndCreateArray |
| fillAndCreateArray | int\* | Функция создания и заполнения массивов |
| M | int\* | Массив М |
| N | int\* | Массив N |
| minM | int | Минимальное положительное массива М |
| minN | int | Минимальное положительное массива N |

Код:

#include <stdio.h>

#include <math.h>

#include <limits.h>

#include <stdlib.h>

#define n 11

int searchMinimum(int size, int \*arr) {

    int min = INT\_MAX;

    for (int i = 0; i < size; i++) {

        if (arr[i] < min && arr[i] > 0)

            min = arr[i];

    }

    return min;

}

int\* fillAndCreateArray(int size) {

    int\* arr = (int\*)malloc(size \* sizeof(int));

    for (int i = 0; i < size; i++) {

        printf("[%d] = ", i);

        scanf("%d", &arr[i]);

    }

    return arr;

}

int main(void) {

    printf("Array M:\n");

    int \*M = fillAndCreateArray(n);

    printf("Array N:\n");

    int \*N = fillAndCreateArray(n);

    int minM = searchMinimum(n, M);

    int minN = searchMinimum(n, N);

    printf("Minimum of M-array = %d\nMinimum of N-array = %d", minM, minN);

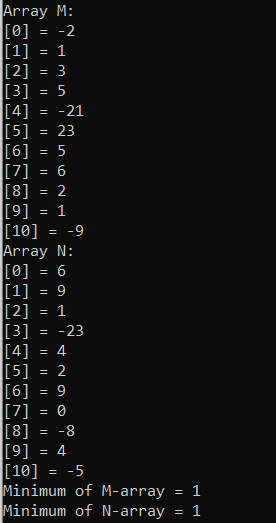
    return 0;

    free(M);

    free(N);

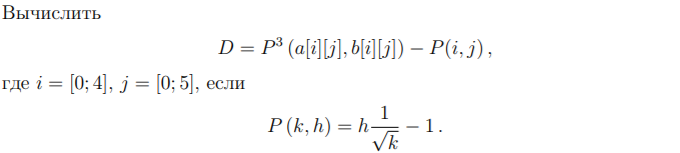
}

Результат:



**Задача 2**

Постановка задачи:



Математическая модель: представлена в условии задачи

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| n\_rows | const | Количество строк матриц |
| m\_cols | const | Количество столбцов матриц |
| freeArray | void | Функция освобождения памяти |
| arr | int\*\* | Аргумент пользовательских функций (принимаемая матрица) / возвращаемая матрица в функции fillAndCreateArray |
| fillAndCreateArray | int\*\* | Функция создания и заполнения двумерного массива |
| n | int | Аргумент пользовательских функций (количество строк) |
| m | int | Аргумент пользовательских функций (количество столбцов) |
| funcP | double | Функция вычисления P(k, h) |
| k | int | Аргумент функции funcP |
| h | int | Аргумент функции funcP |
| a | int\*\* | Двумерный массив a (матрица) |
| b | int\*\* | Двумерный массив b (матрица) |
| D | double | Вычисляемое значение D |
| i | int | Параметр цикла |
| j | int | Параметр цикла |

Код:

#include <stdio.h>

#include <math.h>

#include <limits.h>

#include <stdlib.h>

#define n\_rows 5

#define m\_cols 6

void freeArray(int n, int \*\*arr) {

    for (int i = 0; i < n; i++)

        free(arr[i]);

    free(arr);

}

int\*\* fillAndCreateArray(int n, int m) {

    int\*\* arr = (int\*\*)malloc(n \* sizeof(int\*));

    for (int i = 0; i < n; i++)

        arr[i] = (int\*)malloc(m \* sizeof(int));

    for (int i = 0; i < n; i++)

        for (int j = 0; j < m; j++) {

            printf("[%d][%d] = ", i, j);

            scanf("%d", &arr[i][j]);

        }

    return arr;

}

double funcP(int k, int h) {

        return (h \* 1. / sqrt(k) - 1);

}

int main(void) {

    printf("Array `a`:\n");

    int \*\*a = fillAndCreateArray(n\_rows, m\_cols);

    printf("Array `b`:\n");

    int \*\*b = fillAndCreateArray(n\_rows, m\_cols);

    double D;

    for (int i = 0; i < n\_rows; i++)

        for (int j = 0; j < m\_cols; j++) {

            if (i != 0 && a[i][j] != 0) {

                D = pow(funcP(a[i][j], b[i][j]), 3) - funcP(i, j);

                printf("D(%d, %d) = %lf\n", i, j, D);

            }

            else printf("Division by zero!\n");

        }

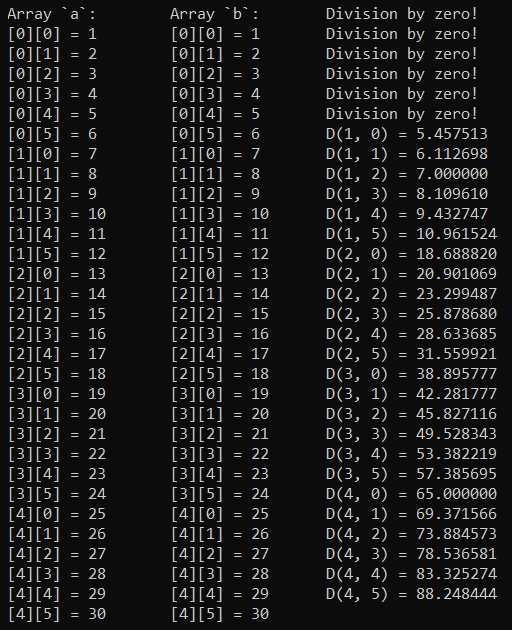
    freeArray(n\_rows, a);

    freeArray(n\_rows, b);

    return 0;

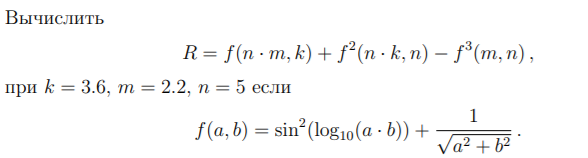
}

Результат:



**Задача 3**

Постановка задачи:



Математическая модель: представлена в условии задачи

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| k | const | Заданное число k |
| m | const | Заданное число m |
| n | const | Заданное число n |
| function | double | Пользовательская функция для вычисления f(a, b) |
| a | int | Аргумент функции function |
| b | int | Аргумент функции function |
| R | double | Результат заданного выражения |

Код:

#include <stdio.h>

#include <math.h>

#define k 3.6

#define m 2.2

#define n 5

double function(int a, int b) {

    return (pow(sin(log10(a \* b) \* M\_PI / 180), 2) + 1. / sqrt(a\*a + b\*b));

}

int main(void) {

    double R = function(n \* m, k) + pow(function(n \* k, n), 2) - pow(function(m, n), 3);

    printf("R = %lf\n", R);

    return 0;

}

Результат:



**Задача 4**

Постановка задачи:

Определить среднее значение для элементов, находящихся ниже побочной диагонали матрицы S[5][5].

Математическая модель:

for (int i = 1; i < n; i++)

for (int j = n-i; j < n; j++)

sum += a[i][j]

Среднее арифметическое будет равняться sum/k, где k - кол-во элементов (n \* (n - 1) / 2) , находящихся ниже побочной диагонали

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| n | const | Размер матрицы (5) |
| freeMatrix | void | Функция освобождения матрицы |
| size | int | Аргумент пользовательских функций (размер матрицы) |
| matrix | int\*\* | Аргумент пользовательских функций (принимаемая матрица) / в функции fillAndCreateMatrix - возвращаемая матрица |
| i | int | Параметр цикла |
| fillAndCreateMatrix | int\*\* | Функция создания и заполнения матрицы |
| searchAverage | double | Поиск среднего арифметического элементов, находящихся ниже побочной диагонали |
| sum | int | Накопительная переменная (сумма элементов, находящихся ниже побочной диагонали) |
| result | double | Результат функции searchAverage |
| S | int\*\* | Матрица S |
| averageS | double | Результат (среднее арифметическое элементов, находящихся ниже побочной диагонали) |

Код:

#include <stdio.h>

#include <stdlib.h>

#define n 5

void freeMatrix(int size, int \*\*matrix) {

    for (int i = 0; i < size; i++)

        free(matrix[i]);

    free(matrix);

}

int\*\* fillAndCreateMatrix(int size) {

    int\*\* matrix = (int\*\*)malloc(size \* sizeof(int\*));

    for (int i = 0; i < size; i++)

        matrix[i] = (int\*)malloc(size \* sizeof(int));

    for (int i = 0; i < size; i++)

        for (int j = 0; j < size; j++) {

            printf("[%d][%d] = ", i, j);

            scanf("%d", &matrix[i][j]);

        }

    return matrix;

}

double searchAverage(int size, int \*\*matrix) {

    int sum = 0;

    for (int i = 1; i < size; i++)

        for (int j = size - i; j < size; j++)

            sum += matrix[i][j];

    double result = sum / (size \* (size - 1) / 2);

    return result;

}

int main(void) {

    int \*\*S = fillAndCreateMatrix(n);

    double averageS = searchAverage(n, S);

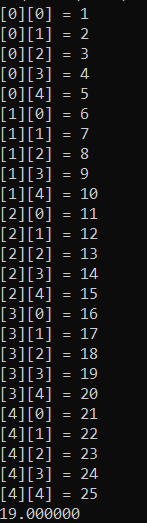
    printf("%lf", averageS);

    freeMatrix(n, S);

    return 0;

}

Результат:



**Задача 5**

Постановка задачи:

Даны две квадратные матрицы A и B размером N × N каждая. Напечатать ту из них, которая имеет минимальный «след» (т.е. сумму элементов главной диагонали). Математически это обозначается Tr(A) (англ. - trace) или Sp(A) (нем. - spur). При решении создать функцию для нахождения следа матрицы и функцию печати матрицы.

Математическая модель:

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| printMatrix | void | Функция распечатки матрицы на экран |
| freeMatrix | void | Функция освобождения памяти |
| fillAndCreateMatrix | int\*\* | Функция создания и заполнения матрицы |
|  |  |  |
| searchTrace | int | Функция нахождения «следа» матрицы |
| trace | int | Возвращаемое значение из функции SearchTrace (след) |
| size | int | Аргумент пользовательских функций (размер матрицы) |
| matrix | int\*\* | Аргумент пользовательских функций / возвращаемое значение функции fillAndCreateMatrix |
| n | const | Размер матриц |
| A | int\*\* | Матрица А |
| B | int\*\* | Матрица B |
| i | int | Параметр цикла |
| j | int | Параметр цикла |
| traceA | int | «След» матрицы А |
| traceB | int | «След» матрицы В |

Код:

#include <stdio.h>

#include <stdlib.h>

void printMatrix(int size, int \*\*matrix) {

    for (int i = 0; i < size; i++) {

        for (int j = 0; j < size; j++)

            printf("%d\t", matrix[i][j]);

        printf("\n");

    }

}

void freeMatrix(int size, int \*\*matrix) {

    for (int i = 0; i < size; i++)

        free(matrix[i]);

    free(matrix);

}

int\*\* fillAndCreateMatrix(int size) {

    int\*\* matrix = (int\*\*)malloc(size \* sizeof(int\*));

    for (int i = 0; i < size; i++)

        matrix[i] = (int\*)malloc(size \* sizeof(int));

    for (int i = 0; i < size; i++)

        for (int j = 0; j < size; j++) {

            printf("[%d][%d] = ", i, j);

            scanf("%d", &matrix[i][j]);

        }

    return matrix;

}

int searchTrace(int size, int \*\*matrix) {

    int trace = 0;

    for (int i = 0; i < size; i++) {

        int j = i;

        trace += matrix[i][j];

    }

    return trace;

}

int main(void) {

    int n;

    printf("Size of matrix (N) = ");

    scanf("%d", &n);

    printf("Matrix - A:\n");

    int \*\*A = fillAndCreateMatrix(n);

    printf("Matrix - B:\n");

    int \*\*B = fillAndCreateMatrix(n);

    int traceA = searchTrace(n, A);

    int traceB = searchTrace(n, B);

    if (traceA < traceB) {

        printf("The minimum trace its trace of the matrix A:\n");

        printMatrix(n, A);

    }

    else {

        printf("The minimum trace its trace of the matrix B:\n");

        printMatrix(n, B);

    }

    freeMatrix(n, A);

    freeMatrix(n, B);

    return 0;

}

Результат:

