Лабораторная работа 7.

«Функции. Введение в структуры.»

Комплект 8.

Задание 1.

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

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| n | Количество элементов массива | const |
| findMin(size, arr) | Функция нахождения минимального положительного числа в массиве | int |
| i | Параметр цикла | int |
| size | Локальная переменная функций createAndFillArr и findMin | int |
| arr | Аргумент функции findMin, возвращаемый массив функции createAndFillArr | int |
| fillAndCreateArray | Функция выделения памяти под динамический массив и заполнение его случайными числами | int |
| M | Массив М | int |
| N | Массив N | int |
| minM | Минимальный положительный элемент массива М | int |
| minN | Минимальный положительный элемент массива N | int |



#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#define n 11

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

int min = abs(arr[0]);

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

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

min = arr[i];

}

return min;

}

int\* createAndFillArr(int size) {

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

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

arr[i] = rand() % 101 - 50;

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

}

return arr;

}

int main() {

char \*locale=setlocale(LC\_ALL, "Russian");

srand(time(NULL));

printf("Массив M:\n");

int \*M = createAndFillArr(n);

printf("Массив N:\n");

int \*N = createAndFillArr(n);

int minM = findMin(n, M);

int minN = findMin(n, N);

printf("Минимальный положительный элемент массива M = %d\nМинимальный положительный элемент массива N = %d", minM, minN);

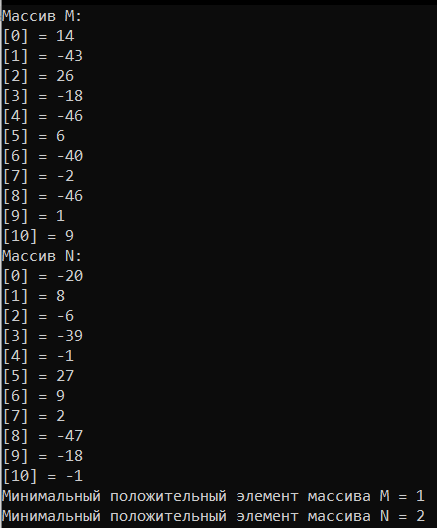
return 0;

free(M);

free(N);

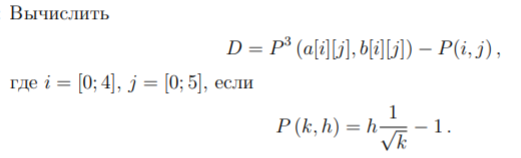
}





Задание 2.







|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| freeMatrix(n, matrix) | Функция освобождения памяти, выделенную под двумерный динамический массив | void |
| \*\*matrix | Принимаемая матрица, возвращаемая матрица в функции fillAndCreateArray | int |
| createAndFillMatrix(n, m) | Функция выделения памяти под динамический двумерный массив и заполнение его случайными числами | int\*\* |
| n | Аргумент функций freeMatrix и createAndFillMatrix, количество | int |
| m | Аргумент пользовательских функций (количество столбцов) | int |
| P(k, h) | Функция вычисления P(k, h) | double |
| k | Аргумент функции P | int |
| h | Аргумент функции P | int |
| \*\*a | Двумерный динамический массив a (матрица) | int |
| \*\*b | Двумерный динамический массив b (матрица) | int |
| D | Вычисляемое значение D | double |
| i | Параметр цикла | int |
| j | Параметр цикла | int |



#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

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

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

free(matrix[i]);

free(matrix);

}

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

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

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

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

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

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

matrix[i][j] = rand() % 101;

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

}

printf("\n");

}

return matrix;

}

double P(int k, int h) {

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

}

int main() {

char \*locale=setlocale(LC\_ALL, "Russian");

srand(time(NULL));

int n\_rows = 5;

int m\_cols = 6;

printf("Массив `a`:\n");

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

printf("Массив `b`:\n");

int \*\*b = createAndFillMatrix(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(P(a[i][j], b[i][j]), 3) - P(i, j);

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

}

else printf("Деление на ноль!\n");

}

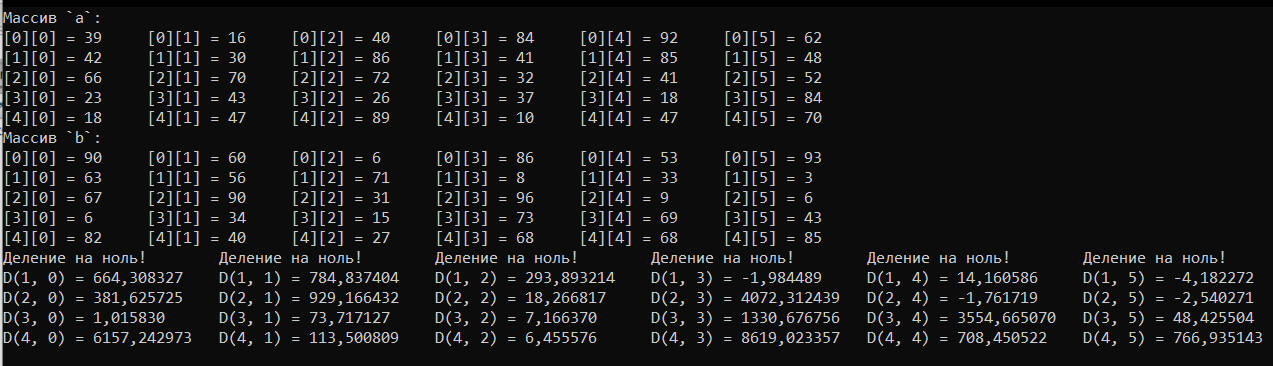
freeMatrix(n\_rows, a);

freeMatrix(n\_rows, b);

return 0;

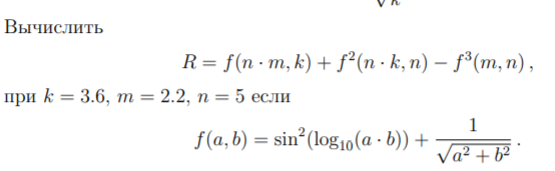
}





Задание 3.







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



#include <stdio.h>

#include <math.h>

#define k 3.6

#define m 2.2

#define n 5

double f(int a, int b) {

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

}

int main() {

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

printf("R = %lf\n", R);

return 0;

}





Задание 4.

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

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| n | Размер матрицы, равный пяти по условию | const |
| freeMatrix(size, matrix) | Функция освобождения памяти из-под динамического двумерного массива | void |
| size | Аргумент пользовательских функций, размер матрицы | int |
| \*\*matrix | Аргумент пользовательских функций, принимаемая матрица, в функции createAndFillMatrix - возвращаемая матрица | int |
| i | Параметр цикла | int |
| createAndFillMatrix(size) | Функция выделения памяти под динамический двумерный массив размера size | int\*\* |
| findAverage(size, matrix) | Функция поиска среднего арифметического элементов, находящихся ниже побочной диагонали | double |
| sum | Локальная переменная пользовательской функции findAverage, сумма элементов, находящихся ниже побочной диагонали) | int |
| result | Возвращаемое значение функции findAverage | double |
| S | Матрица S | int\*\* |
| av | Результат (среднее арифметическое элементов, находящихся ниже побочной диагонали) | double |



#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

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

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

free(matrix[i]);

free(matrix);

}

int\*\* createAndFillMatrix(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++) {

matrix[i][j] = rand() % 101 - 50;

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

}

printf("\n");

}

return matrix;

}

double findAverage(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() {

char \*locale=setlocale(LC\_ALL, "Russian");

int n = 5;

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

double av = findAverage(n, S);

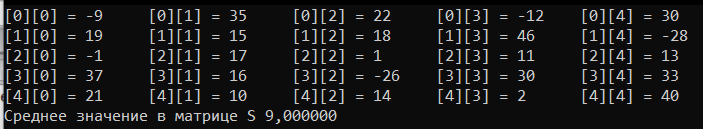
printf("Среднее значение в матрице S %lf", av);

freeMatrix(n, S);

return 0;

}





Задание 5.

1. 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.

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| freeMatrix | Функция освобождения памяти из-под двумерного динамического массива | void |
| createAndFillMatrix | Функция выделения памяти под динамический двумерный массив и заполнение его | int |
| findTrace | Функция нахождения «следа» матрицы | int |
| trace | Возвращаемое значение из функции findTrace (след) | int |
| size | Аргумент пользовательских функций, размер матрицы | int |
| matrix | Аргумент пользовательских функций, возвращаемое значение функции createAndFillMatrix | int |
| n | Размер квадратных матриц | const |
| A | Матрица А | int |
| B | Матрица B | int |
| i | Параметр цикла | int |
| j | Параметр цикла | int |
| traceA | «След» матрицы А | int |
| traceB | «След» матрицы В | int |



#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

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

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

free(matrix[i]);

free(matrix);

}

int\*\* createAndFillMatrix(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++) {

matrix[i][j] = rand() % 101 - 50;

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

}

printf("\n");

}

return matrix;

}

int findTrace(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() {

srand(time(NULL));

char \*locale=setlocale(LC\_ALL, "Russian");

int n;

printf("Размер квадратных матриц N = ");

scanf("%d", &n);

printf("Матрица `A`:\n");

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

printf("Матрица `B`:\n");

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

int traceA = findTrace(n, A);

int traceB = findTrace(n, B);

if (traceA < traceB) {

printf("Минимальный след матрицы `А`. Её след равен %d\n", traceA);

}

else {

printf("Минимальный след матрицы `B`. Её след равен %d\n", traceB);

}

freeMatrix(n, A);

freeMatrix(n, B);

return 0;

}



