Лабораторная работа №8

Задание 1

Постановка задачи: Даны три одномерных массива вещественных чисел

Найти общую сумму положительных элементов в массивах. Нахождение суммы элементов в массиве оформить функцией

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| **Идентификатор** | **Семантика** | **Тип** |
| n, k ,g | размеры массивов | int |
| min\_range, max\_range | границы диапазона случайных чисел | int |
| filling | Функция заполения массива случайными числами | - |
| sum\_of\_positive | Функция нахождение суммы положительных элементов | double |
| a[n], b[k], c[g] | Массивы | double |

Код программы:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*Размер первого массива\*/

#define n 6

/\*Размер второго массива\*/

#define k 8

/\*Размер третьго массива\*/

#define g 7

/\*Нижняя граница диапазона случайных чисел\*/

#define min\_range -1000

/\*Верхняя граница диапазона случайных чисел\*/

#define max\_range 1000

/\*Функция заполения массива случайными числами\*/

void filling(double \*array, int amount\_of\_elements) {

srand(time(NULL));

for (int i = 0; i < amount\_of\_elements; i++) {

array[i] = (double)(rand() % (max\_range - min\_range + 1)) / 100 - 10;

}

}

/\*Функция нахождение суммы положительных элементов\*/

double sum\_of\_positive(double \*array, int amount\_of\_elements) {

double sum = 0;

int i;

for (i = 0; i < amount\_of\_elements; i++) {

if (array[i] >= 0)

sum += array[i];

}

return sum;

}

int main(int argc, char const \*argv[]) {

double a[n], b[k], c[g];

int i;

/\*Заполнение массивов случайными вещественными числами\*/

filling(a, n);

filling(b, k);

filling(c, g);

/\*Нахождение сум всех положительных элементов\*/

double sums =

sum\_of\_positive(a, n) + sum\_of\_positive(b, k) + sum\_of\_positive(c, g);

printf("Сумма всех положительных элементов: %2.2f\n", sums);

return 0;

}

Результат выполненной работы:



Задание 2

Постановка задачи: Даны два одномерных массива целых чисел. Найти сумму их максимальных элементов. Для нахождения максимального элемента в массиве использовать функцию.

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| **Идентификатор** | **Семантика** | **Тип** |
| n | размер массива | int |
| max\_range | граница диапазона случайных чисел | int |
| filling | Функция заполения массива случайными числами | - |
| sum\_of\_max | Функция нахождения суммы максимума двух массивов одного размера | int |
| a[n], b[n] | Массивы | int |

Код программы:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*Размер первого и второго массива\*/

#define n 8

/\*Верхняя граница диапазона случайных чисел\*/

#define max\_range 100

/\*Функция заполения массива случайными числами\*/

void filling(int \*array, int amount\_of\_elements) {

for (int i = 0; i < amount\_of\_elements; i++) {

array[i] = rand() % max\_range;

}

}

/\*Функция нахождения суммы максимума двух массивов одного размера\*/

int sum\_of\_max(int \*array1, int \*array2, int amount\_of\_elements) {

int max1 = array1[0], max2 = array2[0];

for (int i = 0; i < amount\_of\_elements; i++) {

if (max1 < array1[i])

max1 = array1[i];

if (max2 < array2[i])

max2 = array2[i];

}

return max1 + max2;

}

int main(int argc, char const \*argv[]) {

srand(time(NULL));

int a[n], b[n];

/\*Заполнение массивов случайными числами\*/

filling(a, n);

filling(b, n);

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

printf("%d\t%d\n", a[i], b[i]);

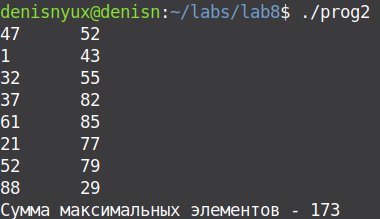
}

printf("Сумма максимальных элементов - %d\n", sum\_of\_max(a, b, n));

return 0;

}

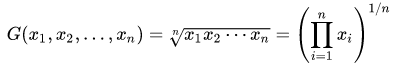
Результат выполненной работы:



Задание 3

Постановка задачи: Даны три одномерных массива вещественных чисел. Найти среднее геометрическое значение положительных элементов для каждого. Вычисление среднего геометрического оформить в виде функции.

Математическая модель:



Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| **Идентификатор** | **Семантика** | **Тип** |
| n, k ,g | размеры массивов | int |
| min\_range, max\_range | границы диапазона случайных чисел | int |
| filling | Функция заполения массива случайными числами | - |
| geometric\_mean | Функция нахождения среднего геометрического | double |
| a[n], b[k], c[g] | Массивы | double |

Код программы:

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*Размер первого массива\*/

#define n 6

/\*Размер второго массива\*/

#define k 8

/\*Размер третьго массива\*/

#define g 7

/\*Нижняя граница диапазона случайных чисел\*/

#define min\_range -1000

/\*Верхняя граница диапазона случайных чисел\*/

#define max\_range 1000

/\*Функция заполения массива случайными числами\*/

void filling(double \*array, int amount\_of\_elements) {

for (int i = 0; i < amount\_of\_elements; i++) {

array[i] = (double)(rand() % (max\_range - min\_range + 1)) / 100;

}

}

/\*Функция нахождения среднего геометрического\*/

double geometric\_mean(double \*array, int amount\_of\_elements) {

double product = 1, mean;

for (int i = 0; i < amount\_of\_elements; i++)

product \*= array[i];

mean = pow(product, 1.0 / amount\_of\_elements);

return mean;

}

int main(int argc, char const \*argv[]) {

srand(time(NULL));

double a[n], b[k], c[g];

int i;

/\*Заполнение массивов случайными вещественными числами\*/

filling(a, n);

filling(b, k);

filling(c, g);

printf("Массивы:\nA:\tB:\tC:\n");

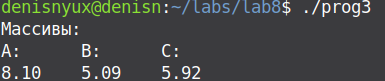
printf("%2.2f\t%2.2f\t%2.2f\n", geometric\_mean(a, n), geometric\_mean(b, k),

geometric\_mean(c, g));

return 0;

}

Результат выполненной работы:



Задание 4

Постановка задачи: Даны две матрицы целых чисел в каждой из которых имеется по два одинаковых числа. Распечатать их значения.

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| **Идентификатор** | **Семантика** | **Тип** |
| n | размер массива | int |
| printing | Функция, распечатывающая значения массивов | int |
| s[n][n], k[n][n] | Матрицы | int |

Код программы:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*Размеры первого и второго массива\*/

#define n 3

void printing(int array[][n]) {

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

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

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

}

printf("\n");

}

}

int main(int argc, char const \*argv[]) {

int s[n][n]={1,2,3,4,5,6,7,8,9};

int k[n][n]={-1,-2,-3,4,-5,6,-7,-8,-9};

/\*Вывод матрицы s\*/

printf("Матрица S:\n" );

printing(s);

/\*Вывод матрицы k\*/

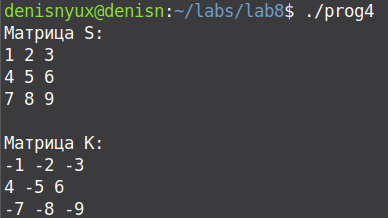
printf("\nМатрица K:\n" );

printing(k);

return 0;

}

Результат выполненной работы:



Задание 5

Постановка задачи: Даны матрицы целых чисел. Найти наименьшую из сумм неотрицательных элементов строк матрицы. Для вычисления суммы использовать подпрограмму (функцию)

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| **Идентификатор** | **Семантика** | **Тип** |
| n, m | размер массива | int |
| max\_range | граница диапазона случайных чисел | int |
| filling | Функция заполения массива случайными числами | - |
| sum\_of\_strings | Функция нахождения суммы положительных элементов в строке | int |
| d[n] | Массив | int |
| minimal\_sum | Минимальная сумма элементов строки | int |

Код программы:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*Количество строк\*/

#define n 6

/\*Количество столбцов\*/

#define m 5

/\*Верхняя граница диапазона случайных чисел\*/

#define max\_range 100

/\*Функция заполения массива случайными числами\*/

void filling(int array[n][m]) {

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

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

array[i][j] = rand() % max\_range - max\_range / 2;

}

}

}

/\*Функция вывода матрицы\*/

void printing(int array[n][m]) {

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

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

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

}

printf("\n");

}

}

/\*Функция нахождения суммы минимальных неотрицательных элементов\*/

int sum\_of\_strings(int array[n][m], int i) {

int sum = 0;

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

if (array[i][j] > 0)

sum += array[i][j];

return sum;

}

int main(int argc, char const \*argv[]) {

srand(time(NULL));

int d[n][m];

filling(d);

printf("D:\n");

printing(d);

int minimal\_sum = sum\_of\_strings(d, 0);

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

if (minimal\_sum > sum\_of\_strings(d, i))

minimal\_sum = sum\_of\_strings(d, i);

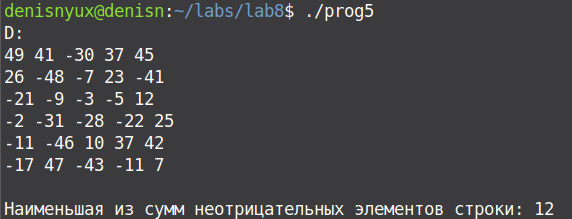
printf("\nНаименьшая из сумм неотрицательных элементов строки: %d\n",

minimal\_sum);

return 0;

}

Результат выполненной работы:



Задание 6

Постановка задачи: Дана матрица целых чисел. Используя функцию, найти среднее геометрическое значение для каждого столбца матрицы.

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| **Идентификатор** | **Семантика** | **Тип** |
| n, m | размер массива | int |
| max\_range, min\_range | граница диапазона случайных чисел | int |
| filling | Функция заполения массива случайными числами | - |
| geometric mean | Функция нахождения среднего геометрического для каждой строки | double |
| printing | функция вывода матрицы | - |
| d[n][m] | Массив | double |

Код программы:

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*Количество строк\*/

#define n 3

/\*Количество столбцов\*/

#define m 5

/\*Нижняя граница диапазона случайных чисел\*/

#define min\_range -1000

/\*Верхняя граница диапазона случайных чисел\*/

#define max\_range 1000

/\*Функция заполения массива случайными числами\*/

void filling(double array[n][m]) {

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

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

array[i][j] = (double)(rand() % (max\_range - min\_range + 1)) / 100;

}

}

}

/\*Функция вывода матрицы\*/

void printing(double array[n][m]) {

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

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

printf("%2.2f ", array[i][j]);

}

printf("\n");

}

}

/\*Функция нахождения среднего геометрического для каждой строки\*/

double geometric\_mean(double array[n][m], int i) {

double product = 1, mean;

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

product \*= array[i][j];

mean = pow(product, 1.0 / n);

return mean;

}

int main(int argc, char const \*argv[]) {

srand(time(NULL));

double d[n][m];

filling(d);

printf("D:\n");

printing(d);

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

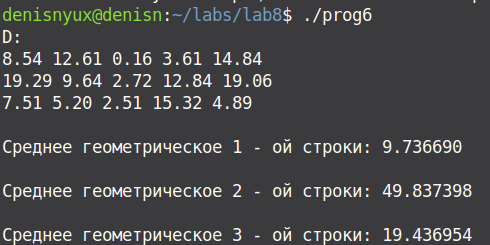
printf("\nСреднее геометрическое %d - ой строки: %f\n", i + 1,

geometric\_mean(d, i));

return 0;

}

Результат выполненной работы:



Задание 7

Постановка задачи: Дана матрица целых чисел. Найти наименьшие значения элементов в каждой из строк матрицы с помощью функции.

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| **Идентификатор** | **Семантика** | **Тип** |
| n, m | размер массива | int |
| max\_range | граница диапазона случайных чисел | int |
| filling | Функция заполения массива случайными числами | - |
| min\_of\_string | Функция нахождения минимального элемента в строке | int |
| printing | функция вывода матрицы | - |
| f[n][m] | Массив | int |

Код программы:

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*Количество строк\*/

#define n 4

/\*Количество столбцов\*/

#define m 5

/\*Верхняя граница диапазона случайных чисел\*/

#define max\_range 100

/\*Функция заполения массива случайными числами\*/

void filling(int array[n][m]) {

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

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

array[i][j] = rand() % max\_range;

}

/\*Функция вывода матрицы\*/

void printing(int array[n][m]) {

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

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

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

}

printf("\n");

}

}

/\*Функция нахождения минимального элемента в строке\*/

int min\_of\_string(int array[n][m], int i) {

int min = array[i][0];

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

if (min > array[i][j])

min = array[i][j];

return min;

}

int main(int argc, char const \*argv[]) {

srand(time(NULL));

int f[n][m];

filling(f);

printf("F:\n");

printing(f);

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

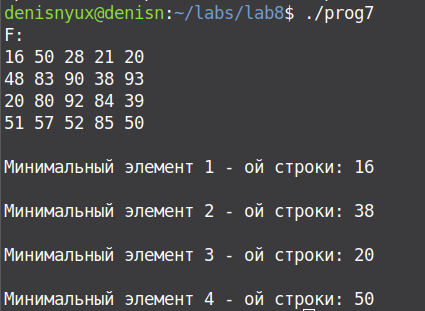
printf("\nМинимальный элемент %d - ой строки: %d\n", i + 1,

min\_of\_string(f, i));

return 0;

}

Результат выполненной работы:



Задание 8

Постановка задачи: вести число N и определить, простое оно или нет. Использовать функцию, которая отвечает на этот вопрос.

Математическая модель: простое число это число которое не делится ни на какие числа кроме самого себя и единицы.

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| **Идентификатор** | **Семантика** | **Тип** |
| number | число | int |
| prime\_number | функция, определяющее простое ли число или нет | int |

Код программы:

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int prime\_number(int number) {

int primeness = 0;

for (int i = 2; i < number; i++)

if (number % i == 0)

primeness = 1;

return primeness;

}

int main(int argc, char const \*argv[]) {

int number;

printf("Введите чило: ");

scanf("%d", &number);

if (prime\_number(number) == 0)

printf("\nЧисло %d - простое\n", number);

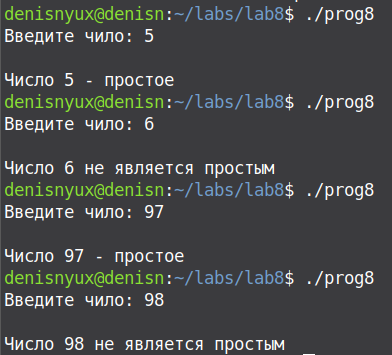
else

printf("\nЧисло %d не является простым\n", number);

return 0;

}

Результат выполненной работы:



Задание 9

Постановка задачи: Реализовать выделение и освобождение памяти для произвольной матрицы с помощью функций malloc и free в своих собственных отдельных функциях. С помощью этих новых функций реализовать код в функции main, выполняющий выделение и освобождение памяти под матрицу с заполнением значениями элементов матрицы и распечаткой этой матрицы на экране терминала.

Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| **Идентификатор** | **Семантика** | **Тип** |
| n, m | размеры массивов | int |
| generation\_of\_matrix | Функция, создающая массив с динамическим выделением памяти | - |

Код программы:

#include <malloc.h>

#include <stdio.h>

#include <stdlib.h>

/\*Верхняя граница диапазона случайных чисел\*/

#define max\_range 100

void generation\_of\_matrix(int n, int m) {

int \*a;

int i, j;

a = (int \*)malloc(n \* m \* sizeof(int));

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

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

\*(a + i \* m + j) = rand() % max\_range;

}

}

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

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

printf("%5d ", \*(a + i \* m + j));

}

printf("\n");

}

free(a);

}

int main(int argc, char const \*argv[]) {

int n, m;

printf("Введите количество строк: ");

scanf("%d", &n);

printf("Введите количество столбцов: ");

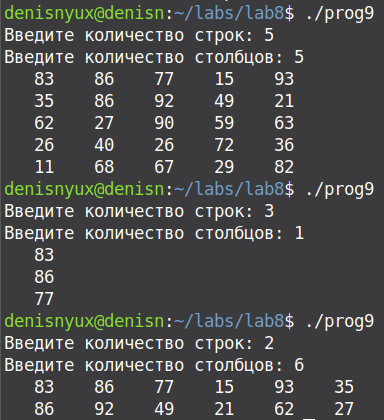
scanf("%d", &m);

generation\_of\_matrix(n, m);

return 0;

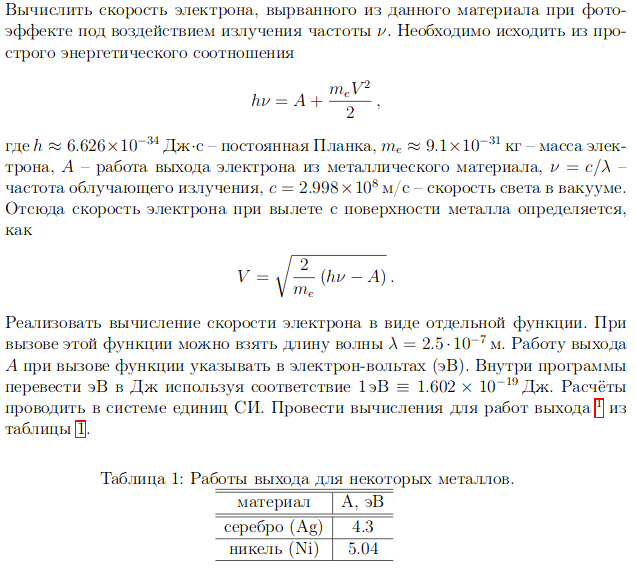
}

Результат выполненной работы:



Задание 10

Постановка задачи:



Код программы:

#include "math.h"

#include "stdio.h"

/\*Постоянная Планка\*/

const double h = 6.626E-34;

/\*Масса электрона\*/

const double me = 9.1E-31;

/\*Скорость света в вакууме\*/

const double c = 2.998E8;

/\*Длина волны\*/

const double l = 2.5E-7;

/\*Частота\*/

const double fr = c / l;

/\*1 эВ в джоулях\*/

const double eV = 1.602E-19;

/\*Работа для серебра в Эв\*/

const double a\_ag = 4.3;

/\*Работа для никеля\*/

const double a\_ni = 5.04;

/\*Перевод Эв в джоули\*/

double invertion(double a) { return a \* eV; }

/\*Cкорость электрона\*/

double speed(double a) { return sqrt(2 / me \* (h \* fr - invertion(a))); }

int main(int argc, char const \*argv[]) {

printf("Ag: %1.5f\n", speed(a\_ag));

printf("Ni: %1.5f\n", speed(a\_ni));

return 0;

}

Результат выполненной работы:

