|  |  |  |
| --- | --- | --- |
| **Раздел** | **Max**  **оценка** | **Итог.**  **оценка** |
| Постановка | 0,5 |  |
| Метод | 1 |  |
| Спецификация | 0,5 |  |
| Алгоритм | 1,5 |  |
| Работа программы | 1 |  |
| Листинг | 0,5 |  |
| Тесты | 1 |  |
| Вопросы | 2 |  |
| Доп.задание | 2 |  |

**Студент: Андреев Глеб Андреевич**

**Группа: \_\_\_\_\_\_\_\_\_БИВ254\_\_\_\_\_\_\_\_**

**Вариант: \_\_\_\_\_№252 (15, 3)\_\_\_\_**

**Руководитель: Альбатша Ахмад Мухаммад Хусайн**

**Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дата сдачи: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Оглавление**

[Задание 3](#_Toc209611387)

[Постановка задачи 4](#_Toc209611388)

[Метод решения задачи 5](#_Toc209611389)

[Внешняя спецификация 6](#_Toc209611390)

[Описание алгоритма на псевдокоде 7](#_Toc209611391)

[Листинг программы 9](#_Toc209611392)

[Распечатка тестов к программе и результатов 11](#_Toc209611393)

# **Задание**

**ЦЕЛЬ РАБОТЫ:** Алгоритмы с досрочным выходом из цикла. Алгоритмы обработки целых чисел.

*1.* (15) Даны целочисленная матрица Q[1:n, 1:m] и целочисленный массив Z[1:k]. Упорядочить по убыванию те строки матрицы Q, сумма элементов которых совпадает с одним из элементов массива Z.

**ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ РАБОТЫ.**

1. Необходима проверка допустимости исходных данных.

2. Необходимо использование алгоритмов с досрочным выходом из цикла. При этом используются либо цикл с предусловием, либо цикл с постусловием.

*2.* (3) Дан массив целых положительных чисел. Сформировать новый массив, содержащий произведения цифр каждого элемента исходного массива.

**ЗАМЕЧАНИЕ.** Каждую часть оформить как отдельную программу.

**ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ РАБОТЫ.**

1. Необходима проверка допустимости исходных данных, в том числе недопустим ввод строки вместо числа.

2. При вычислении результата необходимо использовать целый тип. Использование строк при решении данной задачи недопустимо.**Постановка задачи**

Дано:

1. Q[1:n, 1:m] – целочисленная матрица, Z[1:n] – целочисленный массив
2. A[1:n] – цел.

Результат:

1. Q[1:n][1:m] – целочисленная матрица
2. B[1:n] – целочисленный массив

При:

Связь:

# **Метод решения задачи**

1. j = 1

# **Внешняя спецификация**

{ }\* до и

|  |
| --- |
| Введите x, h, a:  < x > < h > < a >  Задание 1   1. Массив R[1, n]:   << R[1] >> << R[2] >> … << R[n] >> |

|  |
| --- |
| Задание 2 |

|  |
| --- |
| Задание 3 |

# **Описание алгоритма на псевдокоде**

алг «Лабораторная работа №1»

нач

вывод("Лабораторная работа №1")

цикл

вывод("Введите длину массива R от 1 до ", lmax)

ввод(n)

до n > 0 и n <= lmax

кц

вывод("Введены значения x, h, a: ")

ввод(x,h,a)

цикл от i:=1 до n

R[i] := 2.5 \* sin(a \* x + i \* i \* h)

кц

j := 1

цикл от i:=1 до n

если |R[i]| >= 0.7 то

R[j] := R[i]

j := j + 1

все

кц

k := j - 1

если k = 0 то

вывод("2. Все элементы удалены, массив пуст")

вывод("3. Нет среднего значения")

иначе

если k = n то

вывод(“2. Нет элементов, удовлетворяющих условию”)

иначе

вывод("2. Массив R после удаления (", k, " элементов):", R[1,k])

все

min := R[1]

minIndx := 1

цикл от i:= 1 до k

если R[i] <= min то

min := R[i]

minIndx := i

все

кц

sum := 0

цикл от I := 1 до minIndx

sum := sum + R[i]

кц

avg := sum / minIndx

вывод("3. Среднее арифметическое до последнего минимума включительно: ", avg)

все

кон

# **Листинг программы**

#include <stdio.h>

#include <math.h>

#define lmax 100

**int** **main**() {

**int** n, i, k = **0**, minIndx = **0**;

**double** x, h, a, R[lmax];

printf("Лабораторная работа №1**\n**");

**do** {

printf("Введите длину массива R от 1 до %d:**\n**", lmax);

scanf("%d", &n);

**if** (n < **1** || n > lmax)

printf("Введены некорректные значения**\n**");

} **while** (n < **1** || n > lmax);

printf("Введите x, h, a:**\n**");

scanf("%lf %lf %lf", &x, &h, &a);

// I. Вычисление массива

printf("1. Массив R[1,n]: ");

**for** (i = **1**; i <= n; i++) {

R[i] = **2.5** \* sin(a \* x + i \* i \* h);

printf("%8.4lf ", R[i]);

}

// II. Удаление элементов |R[i]| < 0.7 путем сдвига

**int** j = **1**;

**for** (i = **1**; i <= n; i++) {

**if** (fabs(R[i]) >= **0.7**) {

R[j] = R[i];

j++;

}

}

k = j - **1**;

**if** (k == **0**) {

printf("**\n**2. Все элементы удалены, массив пуст**\n**");

printf("3. Нет среднего значения**\n**");

**return** **0**;

}

**else** {

**if** (k == n) printf("2. Нет элементов, удовлетворяющих условию") ;

**else** {

printf("**\n**2. Массив R после удаления содержит %d элемент(-а/-ов): ", k);

**for** (i = **1**; i <= k; i++) {

printf("%8.4lf ", R[i]);

}

}

// III. Поиск последнего минимального элемента

**double** min = R[**1**];

**for** (i = **1**; i <= k; i++) {

**if** (R[i] <= min) {

min = R[i];

minIndx = i;

}

}

// Подсчет среднего арифметического до последнего минимального включительно

**double** sum = **0**;

**for** (i = **1**; i <= minIndx; i++) {

sum += R[i];

}

**double** avg = sum / minIndx;

printf("**\n**3. Среднее арифметическое до последнего минимума включительно: %8.4lf**\n**", avg);

**return** **0**;

}

}

# **Распечатка тестов к программе и результатов**

|  |  |  |
| --- | --- | --- |
| **№** | **Исходные данные** | **Результаты** |
| 1 | n=10  x=2; h=4; a=5 | R = {2.4765 1.9064 2.2545 -2.4629 -0.1106 -0.1548 -2.4366 2.1501 2.0917 2.4994}  R = {2.4765 1.9064 2.2545 -2.4629 -2.4366 2.1501 2.0917 2.4994}  avg = 1.0436 |
| 2 | n=1  x=0.00001; h=0.001; a=0.1 | R = {0.0025}  Все элементы удалены, массив пуст  Нет среднего значения |
| 3 | n=3  x=40; h=45; a=60 | R = { 1.8631 -1.7085 1.0699}  Нет элементов, удовлетворяющих условию  avg = 0.0773 |