 **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ **БИОМЕДИЦИНСКАЯ ТЕХНИКА**

КАФЕДРА **БИОМЕДИЦИНСКИЕ ТЕХНИЧЕСКИЕ СИСТЕМЫ (БМТ-1)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика (Цифровые**  **биомедицинские системы)**

# **О Т Ч Е Т**

**по лабораторной работе №** 2

**Название:** Коллекции и строки

**Дисциплина:** Алгоритмизация и программирование

Студент БМТ1-13Б Н.А.Сухов

(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель  Т.А.Ким

(Подпись, дата) (И.О. Фамилия)

Москва, 2022

# Задание 1. Одномерные массивы

Дан вещественный массив из 45 элементов. Преобразовать массив следующим образом: сначала

расположить все положительные числа, затем все отрицательные и в конце – нули. Вывести на экран исходный и сформированный массив. Вспомогательного массива не использовать.

# Исходный код

println("Введите 45 вещественных числе друг за другом через пробел:")

nums = split(readline())

function special\_array\_sort(before\_sort)

  before\_sort = map((x) -> parse(Float64, x), before\_sort)

  null\_count = count(x->(x == 0.0), before\_sort)

  deleteat!(before\_sort, findall(x -> x == 0, before\_sort))

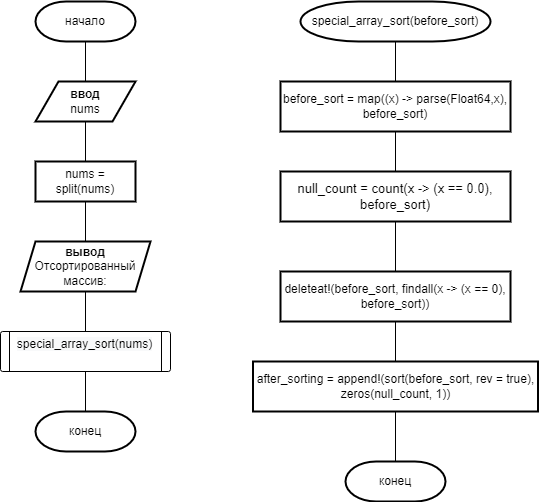
  after\_sorting = append!(sort(before\_sort, rev = true), zeros(null\_count, 1))

  return after\_sorting

end

print("Отсортированный массив: ", special\_array\_sort(nums))

# Схема алгоритма



# Тестирование алгоритма

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование проверки** | **Данные на вход** | **Ожидаемый результат** | **Полученный результат** | **Вывод** |
| Ввод 45 вещественных  чисел от -5.0 до 5.0 | -2.4 2.0 3.6  1.8 -2.1 -4.6  -4.0 -3.6 3.9 -1.0 -1.0  2.7 0.2 0.0  3.8 -2.0 1.8  0.6 2.2 3.1  0.6 -2.4 -1.1 -3.5 -1.1 4.7  2.8 0.1 -4.3 -4.3 1.1 2.1  -4.0 0.2 3.7  -0.2 3.3 -4.6  3.9 -0.4 3.0 -1.9 1.4 1.6  3.9 | Отсортированный массив, где сначала положительные, затем отрицательные и в конце нули | [4.7, 3.9, 3.9,  3.8, 3.7, 3.6, 3.3, 3.1, 3.0, 2.8, 2.7, 2.2, 2.1, 2.0, 1.8, 1.8, 1.6, 1.4,  1.1, 0.6, 0.6,  0.2, 0.2, 0.1, 0.2, -0.4, -1.0,  -1.0, -1.1, -1.1,  -1.9, -2.0, -2.1,  -2.4, -2.4, -3.5,  -3.6, -3.9, -4.0,  -4.0, -4.3, -4.3,  -4.6, -4.6, 0.0] | Все сработало в соответствии с ожиданиями |

# Задание 2. Матрицы

Решить поставленную задачу, используя средства управления вводом/выводом.

Дан массив размером N\*N (N<=10), каждый элемент которого – символ \*. Вывести сначала исходный массив, а затем вывести только главную и побочную диагонали массива.

# Исходный код

println("Введите число для задания квадратной матрицы: ")

n = parse(Int64, readline())

matrix = fill('\*', (n, n))

function diagonals\_output(n)

  println("Исходная матрица :")

  for i in 1:n

    for j in 1:n

      print(matrix[i, j])

    end

    println("")

  end

  println("Вывод главной и побочной диагонали :")

  for i in 1:n

    for j in 1:n

      if i == j || (i + j - 1) == n

        print("\*")

      else

        print(" ")

      end

    end

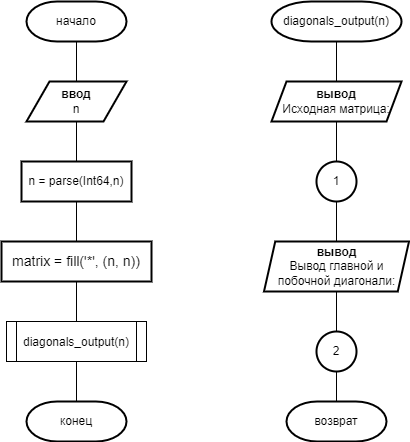
    print("\n")

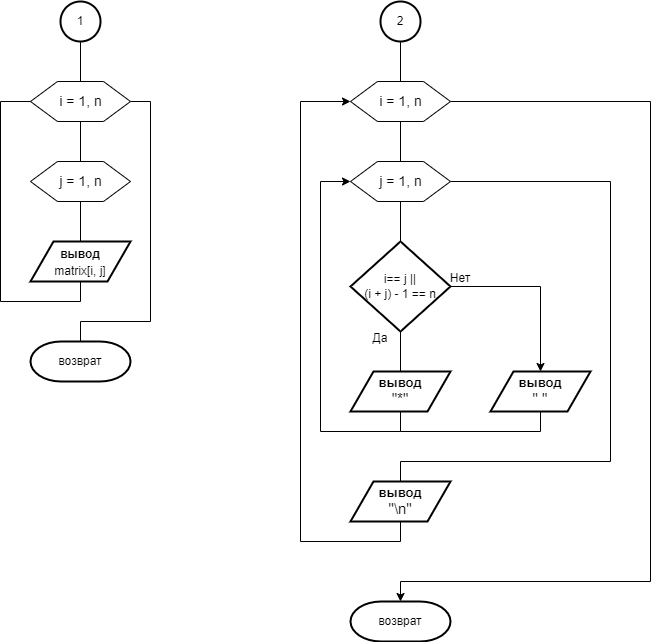
  end

end

diagonals\_output(n)

# Схема алгоритма





# Тестирование алгоритма

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование проверки** | **Ожидаемый результат** | **Полученный результат** | **Вывод** |
| Ввод нечетного n | Диагонали пересекаются в единой, понятной точке | Исходная матрица :  \*\*\*\*\*  \*\*\*\*\*  \*\*\*\*\*  \*\*\*\*\*  \*\*\*\*\*  Вывод главной и побочной диагонали :   * \* * \*   \*   * \* * \* | Вывод совпал с ожиданиями |
| Ввод четного n | Диагонали будут  “соприкосаться” | Исходная матрица :  \*\*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*  Вывод главной и побочной диагонали :   * \* * \*   \*\*  \*\*   * \* * \* | Вывод совпал с ожиданиями |

# Задание 3. Строки

Дана строка S длиной до 40 символов, содержащая разделенные запятыми слова. Дана вторая строка F длиной до 5 символов. Найти в строке S все слова, в которых встречается подстрока F. Вывести на экран исходные строки и результаты поиска.

# Исходный код

println("Введите в одну строку слова через запятую, без пробелов :")

str\_arr = split(readline(), ",")

println("Введите подстроку из 5 символов для поиска в словах прошлой строки:")

sub\_str = readline()

function find(str, substr)

  if length(substr) != 5

    println("Введеная подстрока содержит некорректное количество символов!")

    return 0

  else

    for i in str

      if occursin(substr, i)

        print(i, " ")

      end

    end

  end

end

println("Исходная строка: ")

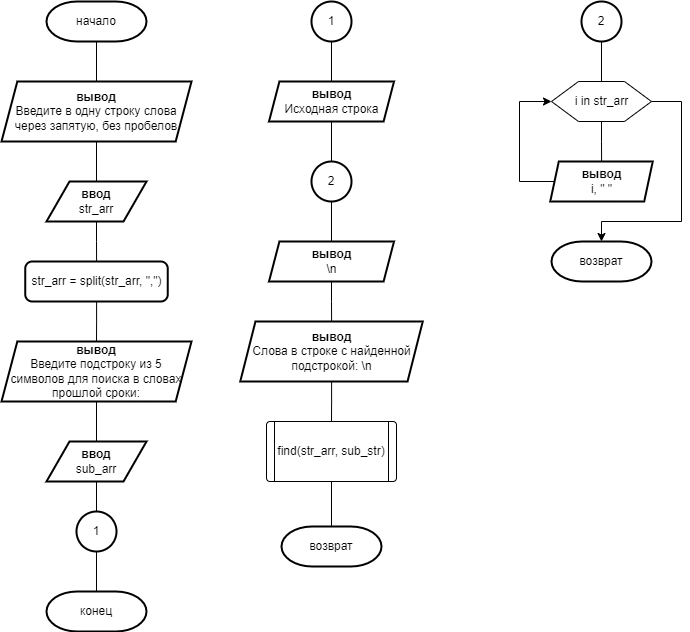
for i in str\_arr print(i, " ") end

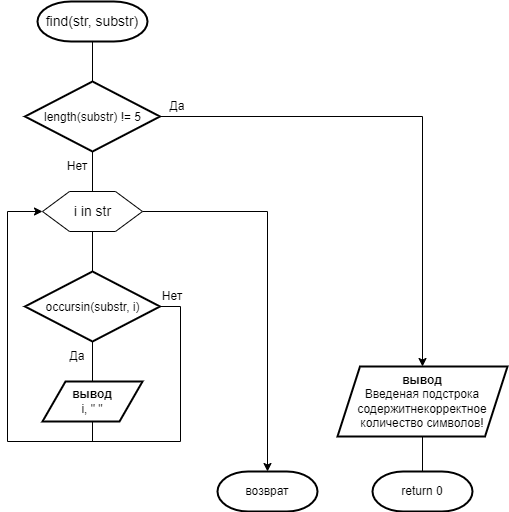
print("\n")

print("Слова в строке с найденной подстрокой:\n")

find(str\_arr, sub\_str)

# Схема алгоритма





# Тестирование алгоритма

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование проверки** | **Ожидаемый результат** | **Полученный результат** | **Вывод** |
| Ввод строки и подстроки | Количество подстрок в строке | Количество подстрок в строке | Полученный результат совпал с ожидаемым |

# Выводы

Я научился использовать массивы и представления строк, как символьных массивов, для решения практических задач. Научился обрабатывать массивы, делать форменные выводы, а также обрабатывать строки.

Также стоит отметить, что при выполнении заданий я научился использовать как внутренние функции языка Julia, так и задавать свои.