Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №2**

**«ИЗУЧЕНИЕ ОДНОМЕРНЫХ МАССИВОВ И СТРОК»**

**ПО ДИСЦИПЛИНЕ**

**«ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»**

Выполнила: студентка учебной группы

ИСПк-202-52-00

Широнина Анна Андреевна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

**Цель работы**: получить базовые навыки работы с одномерными массивами, освоить принципы работы со строками как с частным случаем одномерных массивов.

1. **Формулировка заданий. Вариант 23**

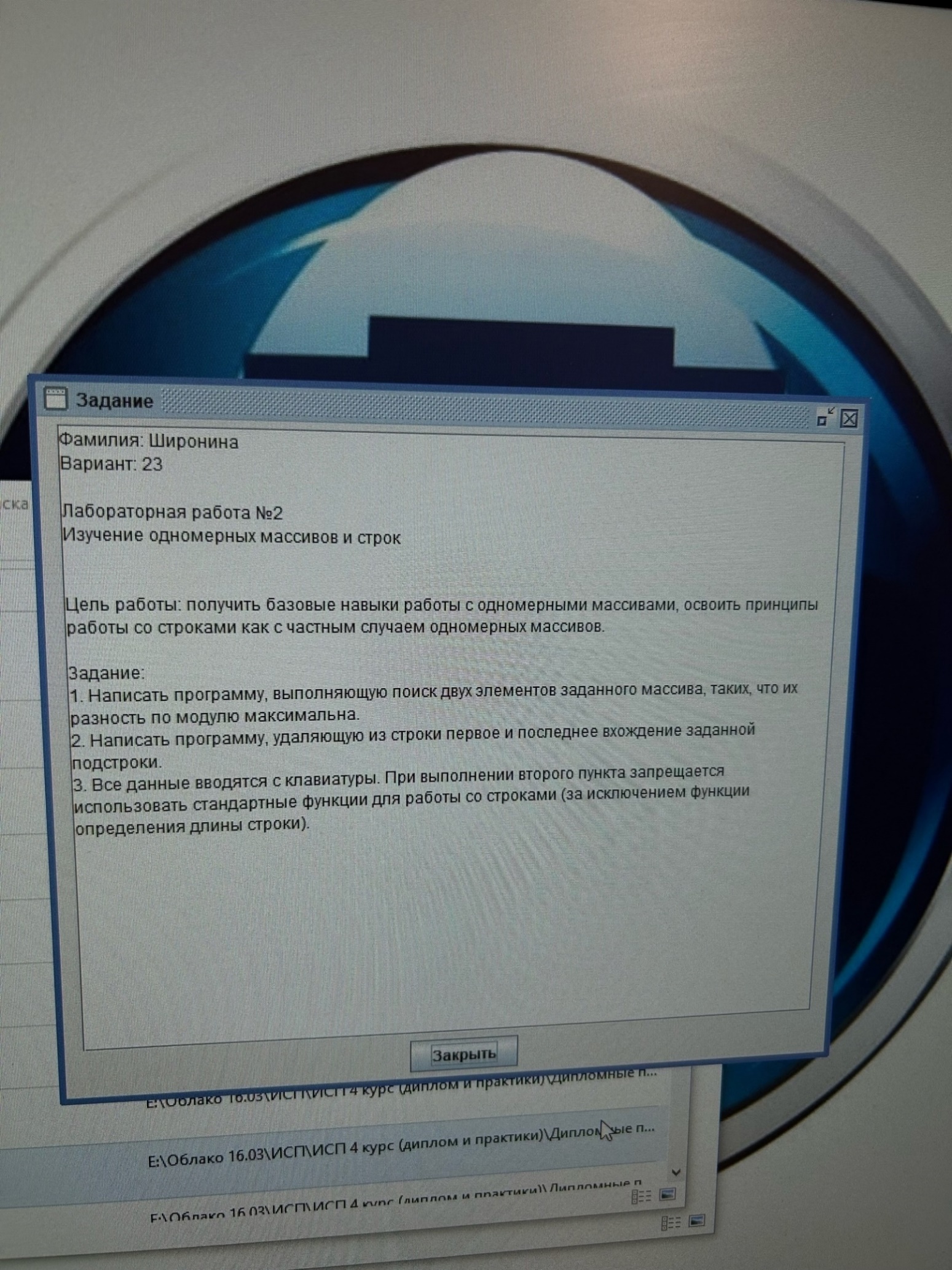
****

Рисунок 1 – Постановки задач

1. **Описание алгоритма**
2. **Задача 1**

Для начала мы запрашиваем количество элементов массива N. Устанавливаем длину массива. Выполняем проверку (нужно, чтобы N был больше нуля, иначе ошибка). Дальше для ввода запрашиваем элементы массива (количество элементов= N). Ищем максимальный и минимальный элемент способом сравнения элемента массива с следующим значением элемента массива (цикл от 0 до N). Находим значение разности по модулю максимального и минимального элемента. Выводим полученный массив, элементы и результат вычисления.

1. **Задача 2**

Для начала мы запрашиваем строку s и подстроку для удаления pod\_s. Инициализируем переменные first и last для записи позиций первого и последнего вхождения подстроки. Придаем переменным len\_s и len\_pod\_s значения длин строк. Ищем позицию первого вхождения подстроки pod\_s в строке s. Ищем позицию последнего вхождения подстроки pod\_s в строке s. Создаем переменную res\_s для записи итоговой строки. Формируем результат, удаляя первое и последнее вхождения. Записываем в переменную res\_s результат. Выводим результирующую строку.

1. **Схема алгоритма**
2. **Задача 1**

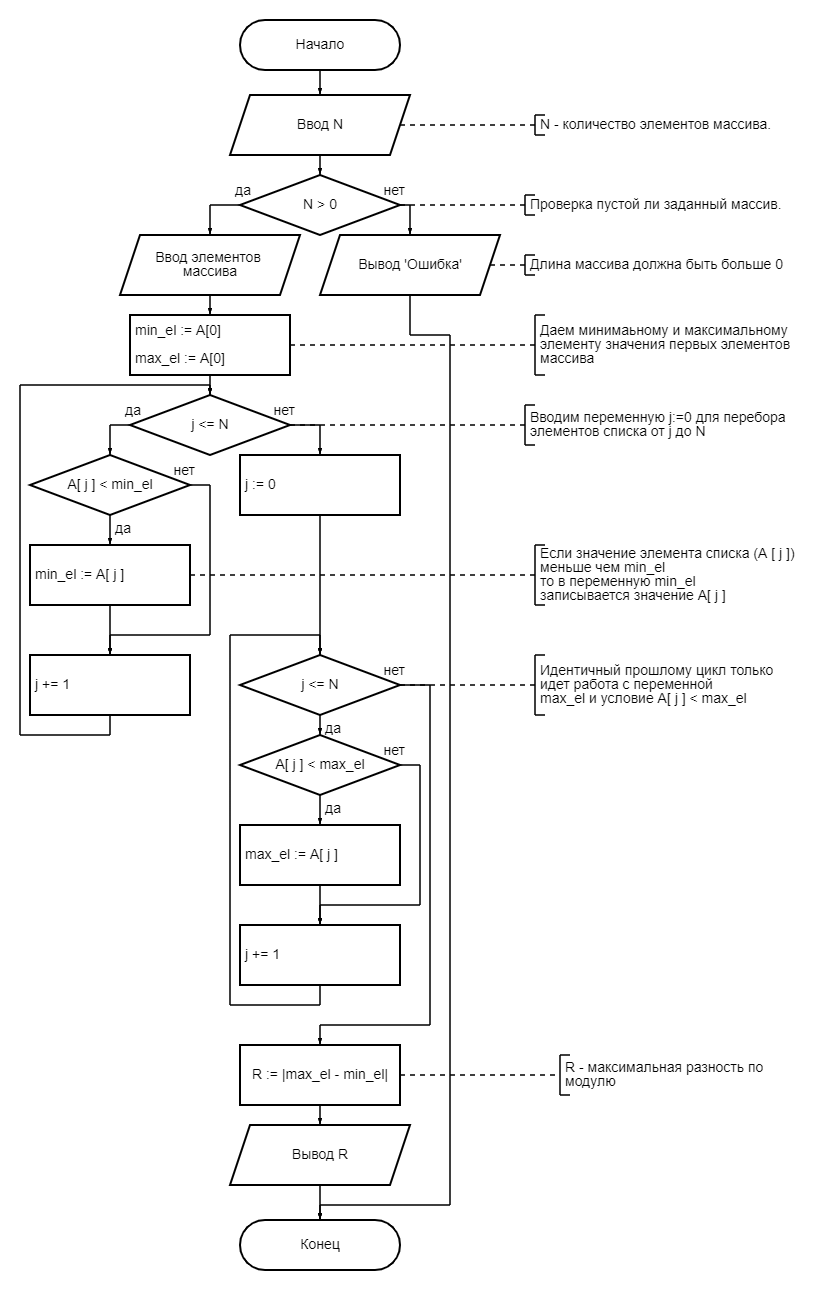
****

Рисунок 2 – Схема алгоритма задачи 1

1. **Задача 2**

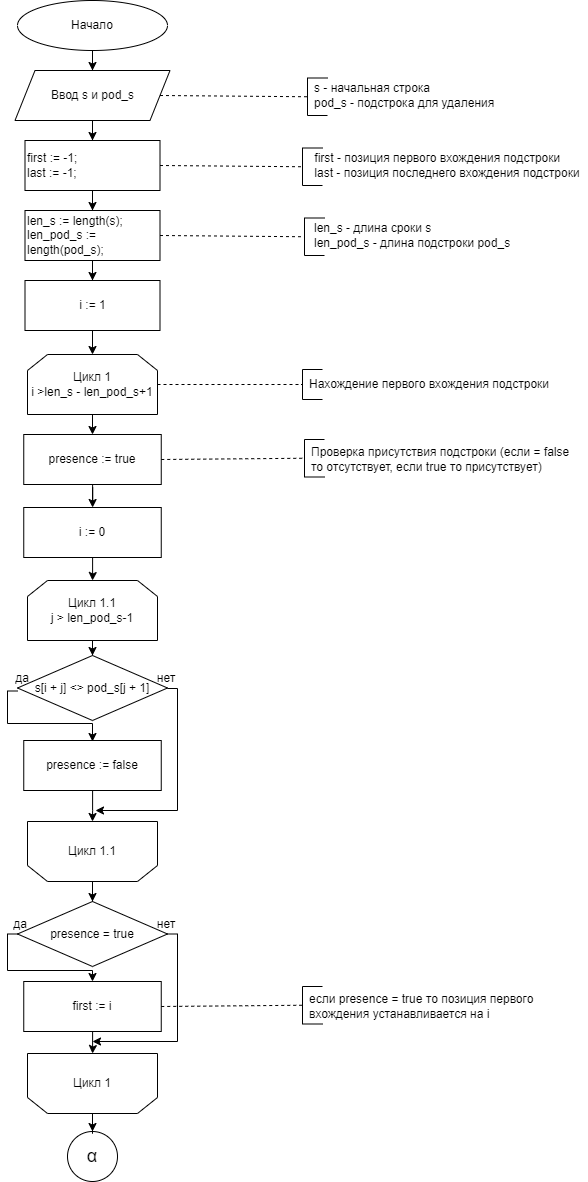
****

Рисунок 3 – Схема алгоритма задачи 2

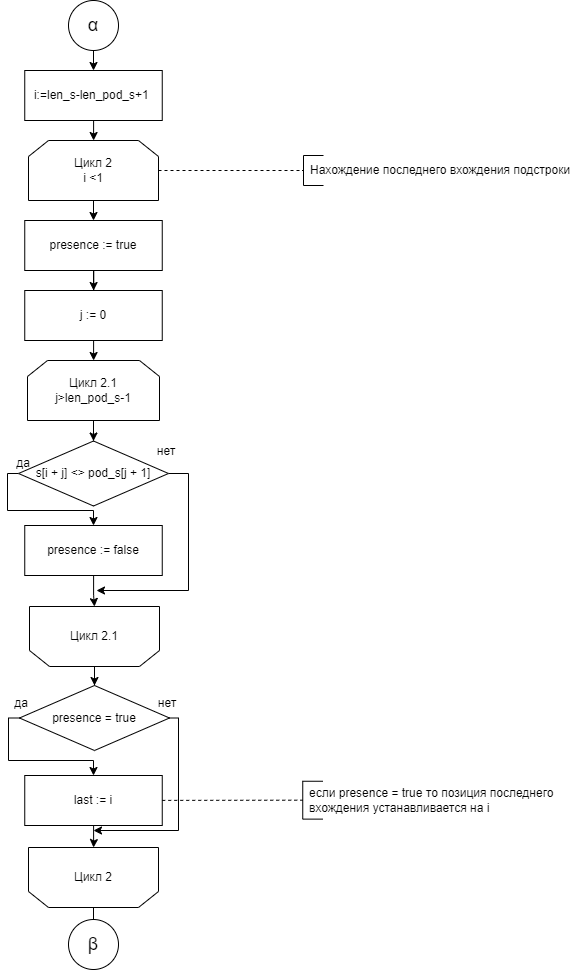


Рисунок 4 – Продолжение рисунка 3

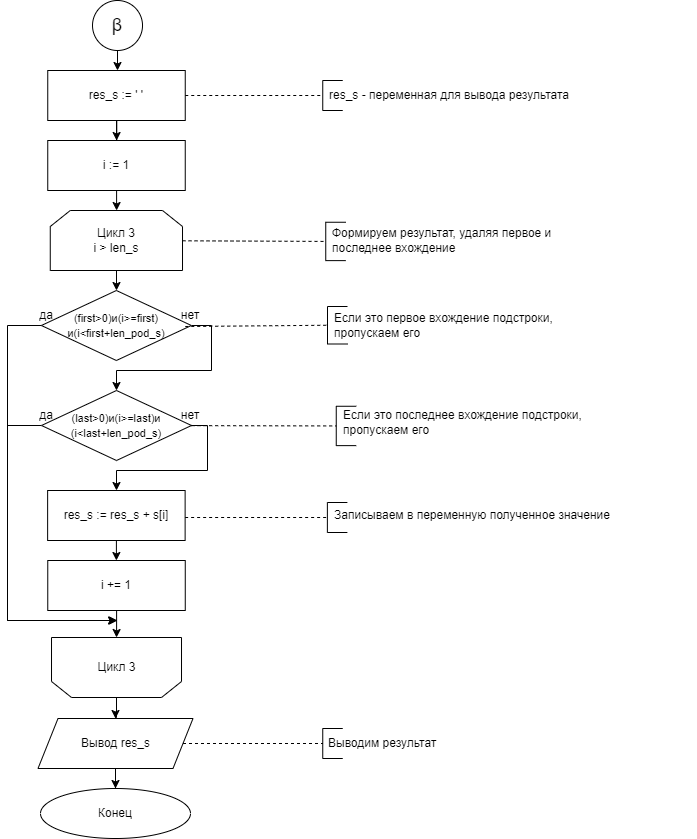


Рисунок 5 – Продолжение рисунка 4

1. **Код программы**
2. **Задача 1**

**var**

A: **array of** integer;

i, j, min\_el, max\_el, R, N: integer;

**begin**

write('Введите количество элементов массива: ');

readln(N);

setlength(A, N);

**if** N > 0 **then**

**begin**

writeln('Введите элементы массива:');

**for** i := 0 **to** N - 1 **do**

**begin**

read(A[i]);

**end**;

min\_el := A[0];

max\_el := A[0];

**for** j := 0 **to** N - 1 **do**

**begin**

**if** A[j] < min\_el **then**

**begin**

min\_el := A[j];

**end**;

**end**;

**for** j := 0 **to** N - 1 **do**

**begin**

**if** A[j] > max\_el **then**

**begin**

max\_el := A[j];

**end**;

**end**;

R := abs(max\_el - min\_el);

writeln('Массив: ', A);

writeln('Элементы: ', max\_el, ' и ', min\_el);

writeln('Максимальная разность по модулю: ', R);

**end**

**else**

**begin**

writeln('Ошибка. Длина массива должна быть больше 0.');

**end**;

**end**.

1. **Задача 2**

**var**

s, pod\_s, res\_s: string;

i, j, first, last, len\_s, len\_pod\_s: integer;

presence: boolean;

**begin**

write('Введите строку: ');

readln(s);

write('Введите подстроку для удаления: ');

readln(pod\_s);

first := -1;

last := -1;

len\_s := length(s);

len\_pod\_s := length(pod\_s);

// Находим первое вхождение подстроки

**for** i := 1 **to** len\_s - len\_pod\_s + 1 **do**

**begin**

presence := true;

**for** j := 0 **to** len\_pod\_s - 1 **do**

**if** s[i + j] <> pod\_s[j + 1] **then**

**begin**

presence := false;

**break**;

**end**;

**if** presence **then**

**begin**

first := i;

**break**;

**end**;

**end**;

// Находим последнее вхождение подстроки

**for** i := len\_s - len\_pod\_s + 1 **downto** 1 **do**

**begin**

presence := true;

**for** j := 0 **to** len\_pod\_s - 1 **do**

**if** s[i + j] <> pod\_s[j + 1] **then**

**begin**

presence := false;

**break**;

**end**;

**if** presence **then**

**begin**

last := i;

**break**;

**end**;

**end**;

// Формируем результат, удаляя первое и последнее вхождение

res\_s := '';

// Проходим по исходной строке

**for** i := 1 **to** len\_s **do**

**begin**

// Если это первое вхождение подстроки, пропускаем его

**if** (first > 0) **and** (i >= first) **and** (i < first + len\_pod\_s) **then**

**continue**;

// Если это последнее вхождение подстроки, пропускаем его

**if** (last > 0) **and** (i >= last) **and** (i < last + len\_pod\_s) **then**

**continue**;

// Добавляем символ в результирующую строку

res\_s := res\_s + s[i];

**end**;

writeln('Результат: ', res\_s);

**end**.

1. **Результат выполнения программы**
2. **Задача 1**

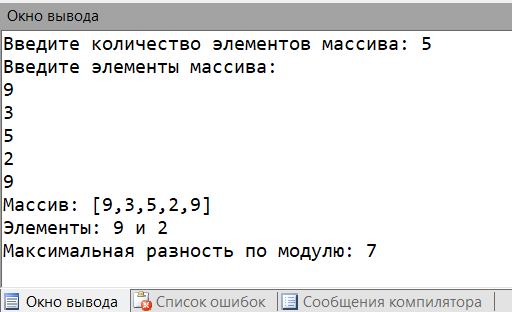
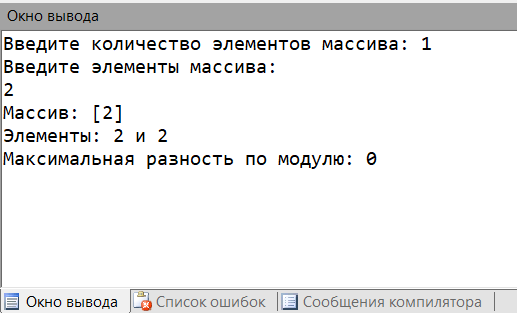
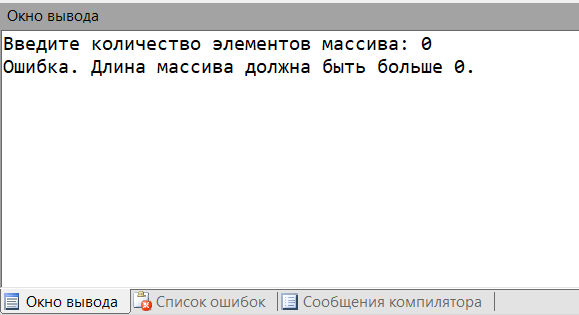
****

Рисунок 4 – Результат выполнения программы для задачки 1

1. **Задача 2**

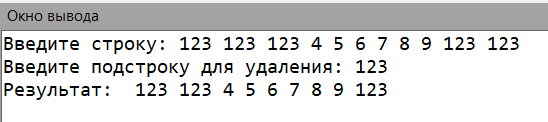
****

Рисунок 5 – Результат выполнения программы для задачки 2

1. **Вывод**

В данной работе мы сосредоточились на изучении одномерных массивов и строк в языке программирования Pascal. Одномерные массивы представляют собой упорядоченные наборы элементов одного типа, которые позволяют эффективно управлять и обрабатывать данные. Строки, в свою очередь, могут быть рассмотрены как массивы символов, что создает возможность работы с текстовыми данными, аналогично работе с числовыми массивами. Представление строк как одномерные массивы, значительно облегчает работу с текстовыми данными и позволяет применять аналогичные методы для их обработки.

Для работы над схемами алгоритма необходимо четко сформулировать проблему, которую необходимо решить, собрать входные данные, необходимые для алгоритма, описать последовательность шагов, которые необходимо выполнить для решения задачи, использовать стандартные символы для построения схемы алгоритма (овалы – начало и конец, прямоугольники – действия или процессы, ромбы – условия или проверки и т.д.), проверить алгоритм на разных входных данных для выявления возможных ошибок, проанализировать алгоритм и внести изменения для повышения его эффективности, записать и оформить полученные результаты. Работа над схемами алгоритма помогает визуализировать процесс и обеспечить более легкое понимание и оптимизацию решения задачи.

В ходе выполнения данной работы были достигнуты цели, изложенные в начале. Получив базовые навыки работы с одномерными массивами и строками, мы смогли освоить основные принципы обработки данных в языке Pascal.