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»**

Выполнил: студент учебной группы

ИСПк-202-52-00

Князева Эвелина Евгеньевна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров,

2022

1. Цель домашней контрольной работы.

Получить базовые навыки работы с одномерными массивами, освоить принципы работы со строками как с частным случаем одномерных массивов.

1. Формулировка задания (с вариантом)

Вариант 4.

1. Написать программу, выполняющую поиск максимального и минимального элементов в массиве.
2. Написать программу, удаляющую из строки первое и последнее вхождение заданной подстроки.
3. Все данные вводятся с клавиатуры. При выполнении второго пункта запрещается использовать стандартные функции для работы со строками (за исключением функции определения длинны строки).

3. Схема алгоритма с комментариями

Программа 1:

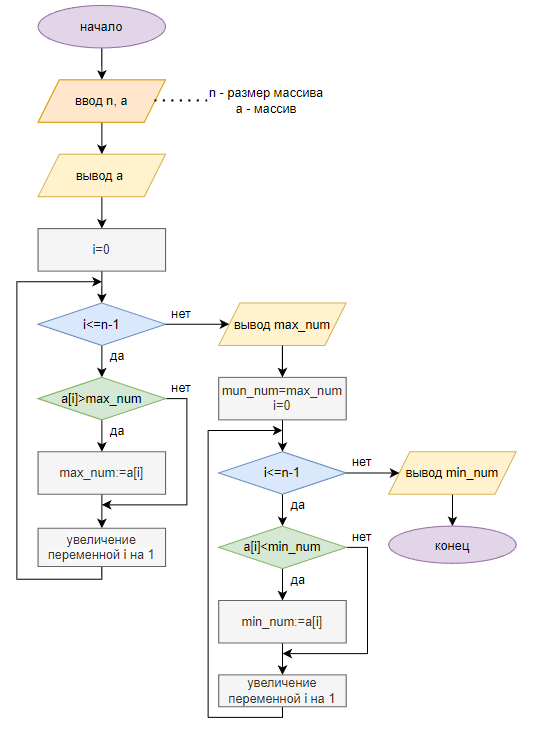


Рисунок 1 – Алгоритм работы программы №1

Программа 2:

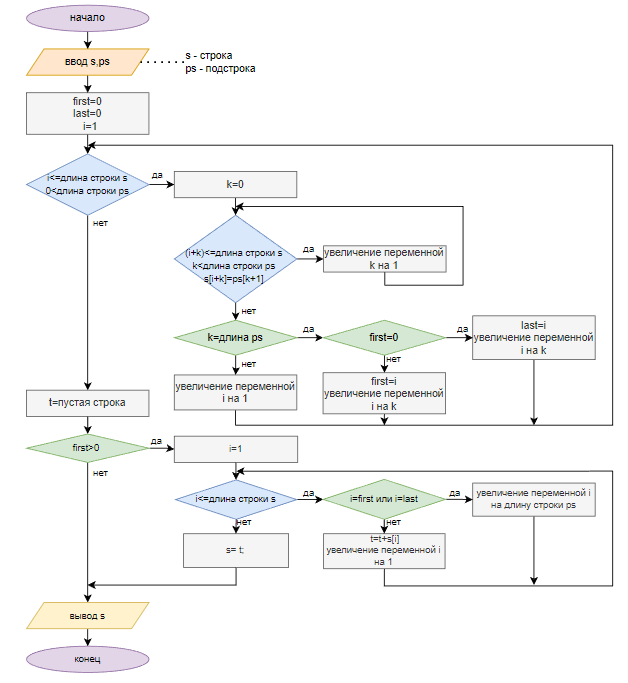


Рисунок 2 – Алгоритм работы программы №2

4. Код программы

Программа 1:

**Begin**

print('Введите размер массива');

**var** n:=readlninteger ();

print('Введите элементы массива');

**var** a := ReadArrInteger(n);

print ('Массив:');

a.Print;

writeln;

**var** max\_num: integer;

**for var** i := 0 **to** (n-1) **do**

**if** a[i] > max\_num **then**

max\_num := a[i];

print ('Максимальный элемент',max\_num);

writeln;

**var** min\_num: integer;

min\_num:=max\_num;

**for var** i := 0 **to** (n-1) **do**

**if** a[i] < min\_num **then**

min\_num := a[i];

print ('Минимальный элемент',min\_num);

**end**.

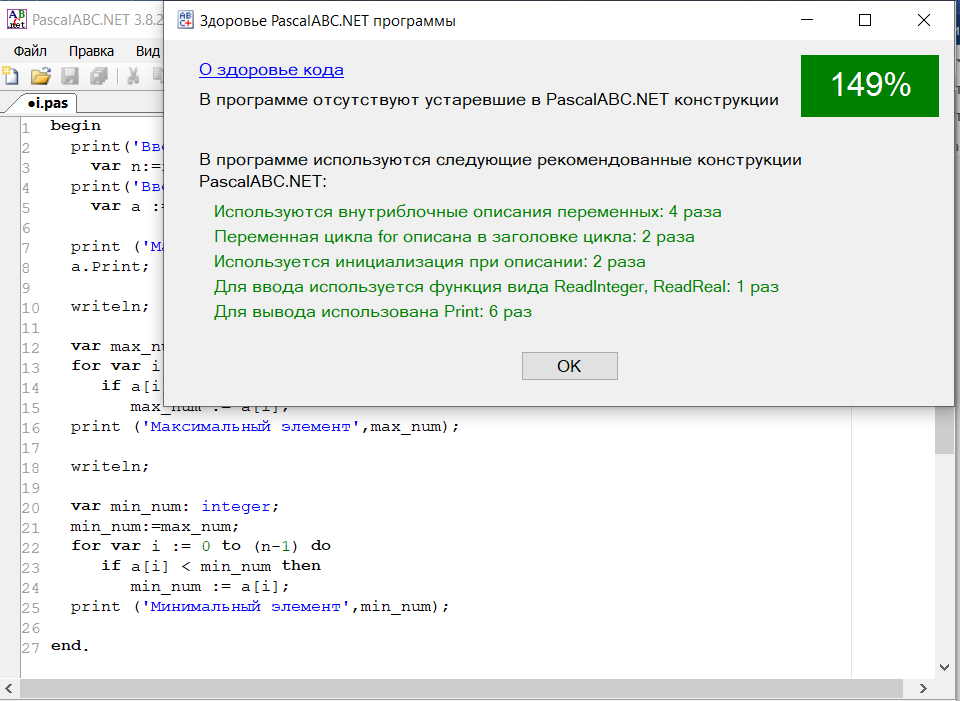


Рисунок 3 – Здоровье кода программы №1

Программа 2:

**begin**

**var** s:=ReadLnString('Введите строку: ');

**var** ps:=ReadLnString('Введите искомую подстроку: ');

**var** first:=0;

**var** last:=0;

**var** i:=1;

**while** (i<=Length(s)) **and** (0<Length(ps)) **do**

**begin**

**var** k:=0;

**while** ((i+k)<=Length(s)) **and** (k<Length(ps)) **and** (s[i+k]=ps[k+1])

**do** Inc(k);

**if** k=Length(ps)

**then begin**

**if** first=0

**then** first := i

**else** last := i;

Inc(i,k)

**end**

**else** inc(i)

**end**;

**var** t:='';

**if** first>0

**then begin**

i:=1;

**while** i<=Length(s) **do**

**begin**

**if** (i=first) **or** (i=last)

**then** Inc(i, Length(ps))

**else**

**begin**

t:=t+s[i];

inc(i)

**end**;

**end**;

s := t;

**end**;

print('Новая строка:');

s.Println;

**end**.

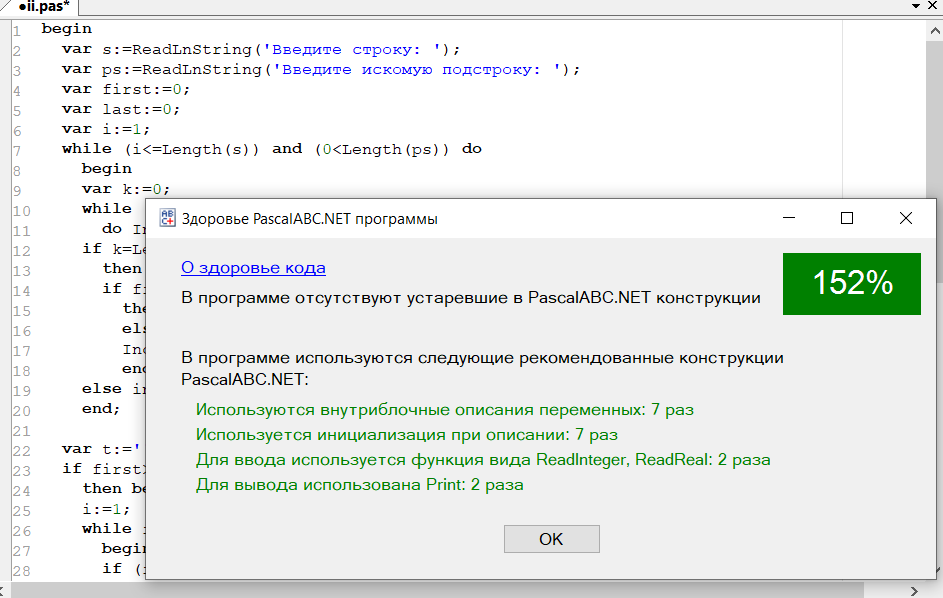


Рисунок 4 – Здоровье кода программы №2

5. Результат выполнения программы

Программа 1:

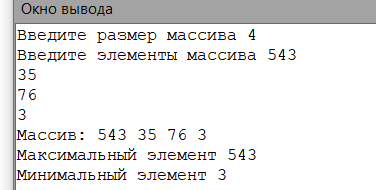


Рисунок 5 – Результат выполнения программы №1

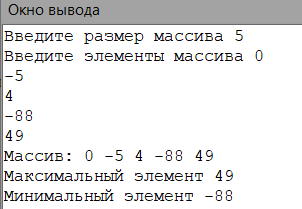


Рисунок 6 – Результат выполнения программы №1

Программа 2:

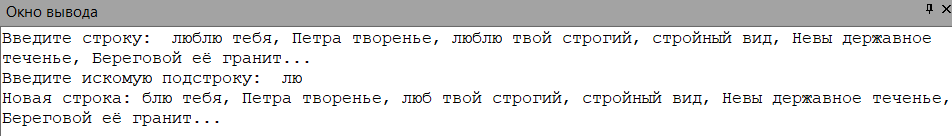


Рисунок 7 – Результат выполнения программы №2

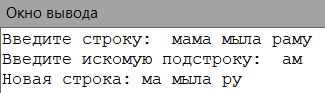


Рисунок 8 – Результат выполнения программы №2

6. Вывод

В ходе выполнения домашней контрольной работы были изучены динамические массивы, строки в языке программирования PascalABS.NET. При написании кода возникли трудности, связанные со здоровьем кода – нужно, чтобы оно было более 100%. В связи с этим пришлось полностью менять код программы и изучать статью «Здоровье кода PascalABC.NET». После изучения были сделаны выводы о том, что здоровье кода портили: количество var вне блоков (-12%), описание параметра цикла for вне заголовка (-25%), использование процедуры read для ввода (-15%), использование ключевого слова program (-10%), использование статических массивов (-10%). Данные замечания были проанализированы, и написаны показатели, дающие положительные проценты: использование внутриблочные описания переменных (+14%), переменная цикла for описана в заголовке цикла (+6%), использование инициализация переменной при описании (+8%), использование для ввода функция вида ReadInteger (+3%), использование для вывода Print (+16%). Таким образом, здоровье кода первой программы стало 149%. При написании программы №2 были проанализированы ошибки, связанные с уменьшением здоровья кода программы №1, поэтому код программы №2 писался уже по новым стандартам.