Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

# оТЧЕТ

по лабораторной работе

на тему:

Изучение строкового типа данных

|  |  |  |
| --- | --- | --- |
| Выполнил  Студент гр. 951002 |  | В. Н. Протасеня |
| Проверил |  | Асс. Е.Е. Фадеева |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Минск, 2019

1. Теоретические сведения по теме лабораторной работы

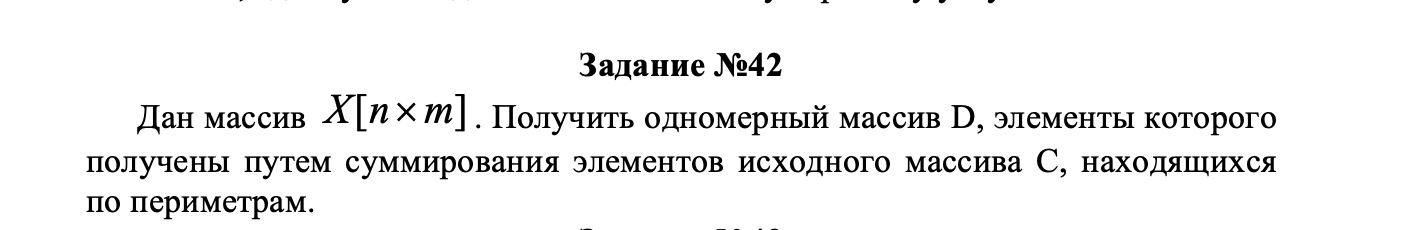
**Строка** – это последовательность любых символов кода обмена информацией (в персональных компьютерах – это код ASCII).

Строка наряду со знаками алфавита языка Паскаль может содержать русские буквы. Строка хранится в поле основной памяти, состоящем из смежных байтов. Каждый байт содержит один символ.

Максимальная длина строки, допускаемая компилятором Паскаль – 255 байт.

Различают строковые константы и строковые переменные.

1.Задание на лабораторную работу



* 1. Эскиз ожидаемого результата

initial massive:

1 9 8 3 1 7 6

2 5 2 8 2 1 3

5 2 4 4 7 1 2

9 3 4 9 8 3 4

5 0 4 2 0 8 3

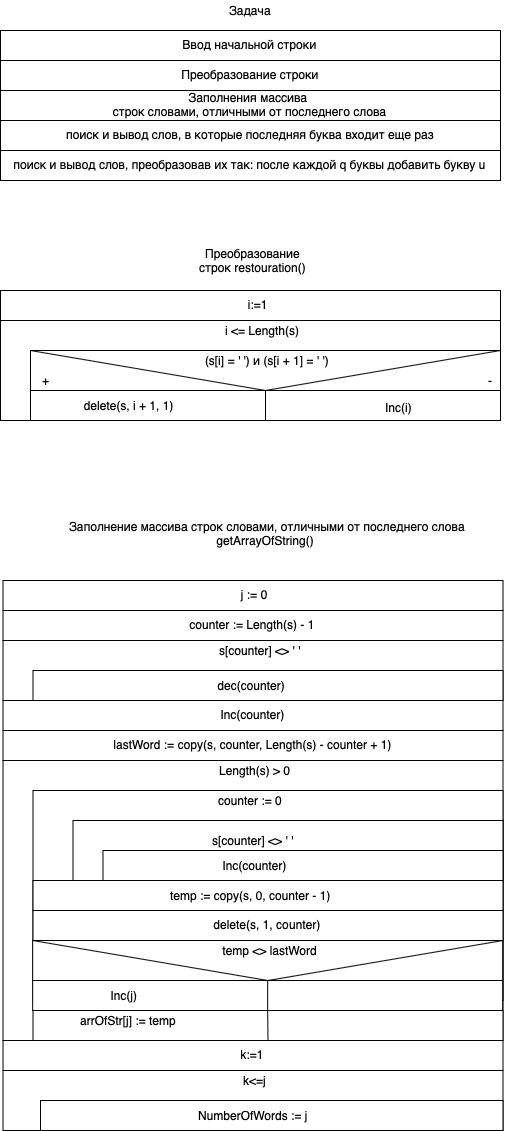
result massive: 82 48 15

1. Выполнение
   1. Разработка алгоритма

Таблица 3.1 используемые идентификаторы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя идентификатора | назначение | Тип идентификатора | Начальное значение | Закон изменения | Имя цикла, в котором происходит изменение переменной |
| C | Начальный двумерный массив | Matrix = array [1 .. n, 1 .. m] of integer | 20 | — | — |
| res | Одномерный массив для хранения сумм элементов, находящихся по периметру | string | 0 | res := res + ' ' + IntToStr(perimetr) | 108-128 |
| sum1 | Хранит значение суммы элементов в первой колонне | integer | 0 | sum1:=getSumInColumn(C, counter) | 108-128 |
| Sum2 | Хранит значение суммы элементов в последней колонне | integer | 0 | sum2:= getSumInColumn(C, m - counter + 1) | 108-128 |
| Sum3 | Хранит значение суммы элементов в первой строке | integer | 0 | sum3:=getSumInLine(C, counter) | 108-128 |
| Sum4 | Хранит значение суммы элементов в последней строке | integer | 0 | sum4:= getSumInLine(C, n - counter + 1 | 108-128 |
| perimetr | Хранит значение суммы элементов, находящихся по периметру | integer | 1 | perimetr := sum1+sum2+sum3+sum4 | 36-61 |
| i | Параметр цикла | integer | 1 | i:=i+1 | 72 |
| j | Параметр цикла | integer |  | j:=j+1 | 56-70 |
| counter | Параметр цикла | integer | 1 | Counter:=counter+1 | 108-128 |
| finish | Переменная, до которой идет цикл | integer | — | finish := (m div 2) + 1  finish := (m div 2)-1 | 100-103 |

Рисунок 3.1 – Схема работы программы методом Насси-Шнейдермана



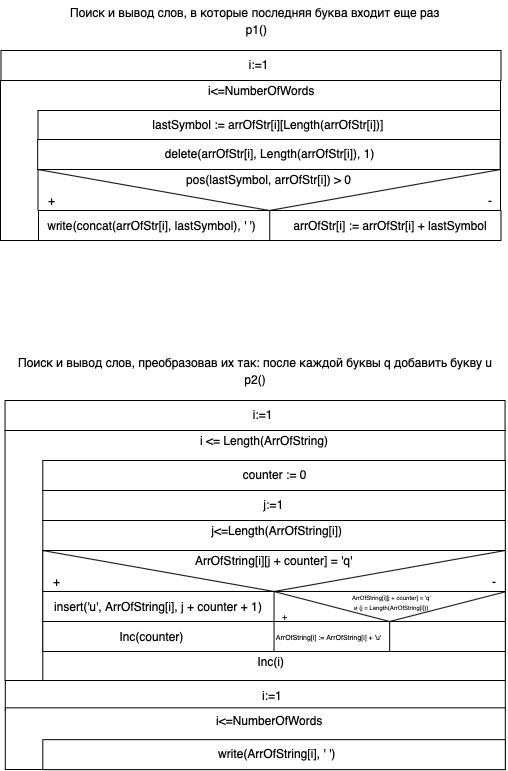
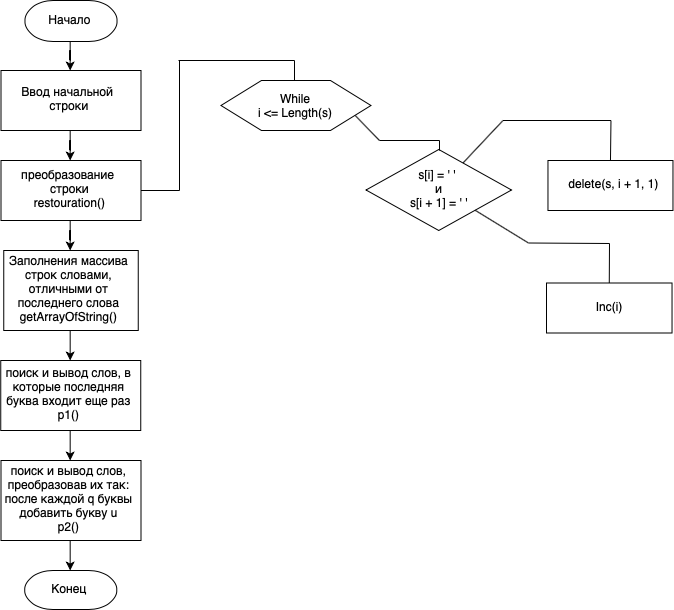
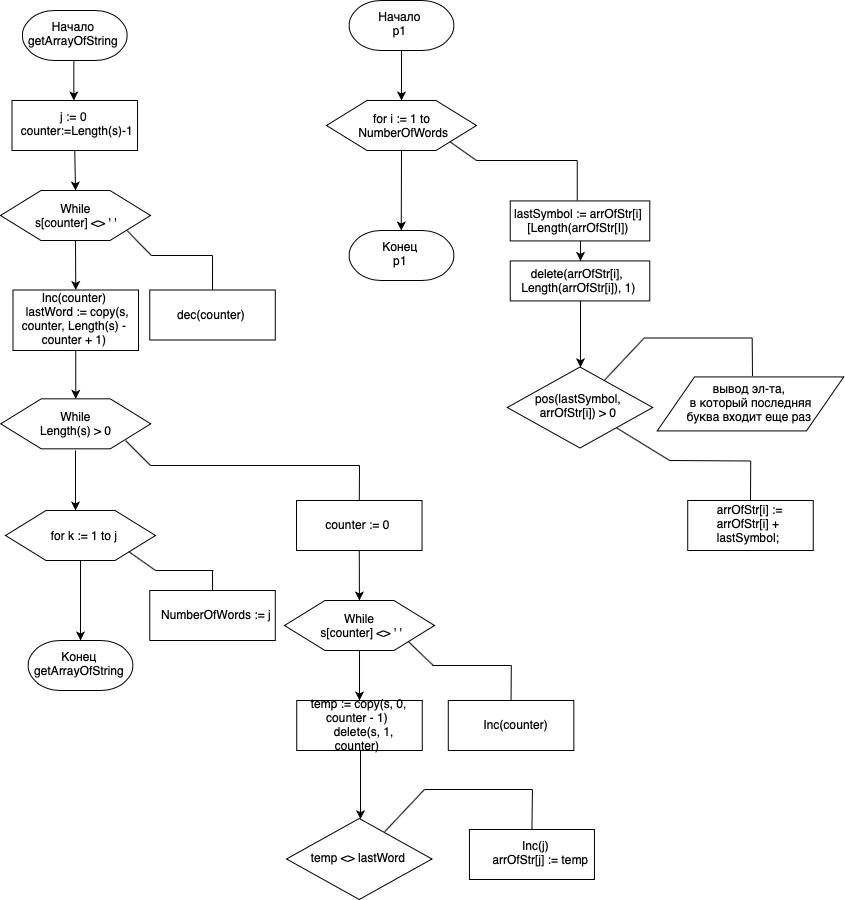
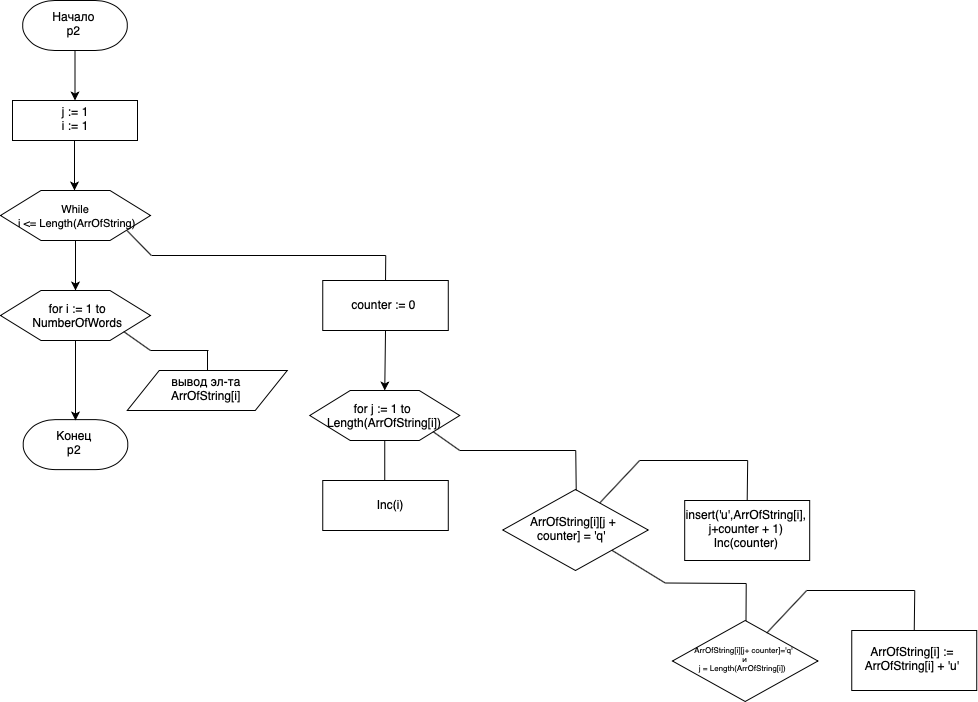


Рисунок 3.2 – Схема работы программы методом Дамке







Описание алгоритма: Тело программы состоит из 1 цикла с предусловием с параметром. Так же в программе используеется 4 процедуры — vvod(рандомно заполняет двумерный массив), print(выводит двумерный массив),setZeroInColumn(обнуление элементов в колонне), setZeroInLine(обнуление элементов в строке), и 2 функции — getSumInColumn(подсчитывет сумму эдементов в колонне), getSumInLine(подсчитывет сумму эдементов в строке).

3.2 Текст программы и его описание

program laba6;

{$APPTYPE CONSOLE}

{$R \*.res}

uses

System.SysUtils, windows;

TYPE

Arr = array [1 .. 255] of string;

var

s, temp: string;

ArrOfString: Arr;

i: integer;

lastWord, letter: string;

flag: boolean;

j, k, h: integer;

NumberOfWords: integer;

procedure restouration(var s: string);

begin

i := 1;

while i <= Length(s) do

begin

if (s[i] = ' ') and (s[i + 1] = ' ') then

delete(s, i + 1, 1)

else

Inc(i);

end;

end;

procedure getArrayOfString(var s: string; var arrOfStr: Arr);

var

counter, j, k: integer;

temp: string;

lastWord: string;

begin

j := 0;

counter := Length(s) - 1;

while s[counter] <> ' ' do

dec(counter);

Inc(counter);

lastWord := copy(s, counter, Length(s) - counter + 1);

while Length(s) > 0 do

begin

counter := 0;

while s[counter] <> ' ' do

Inc(counter);

temp := copy(s, 0, counter - 1);

delete(s, 1, counter);

if temp <> lastWord then

begin

Inc(j);

arrOfStr[j] := temp;

end;

end;

for k := 1 to j do

NumberOfWords := j;

end;

procedure p1(var arrOfStr: Arr);

var

i: integer;

lastSymbol: string;

word: string;

begin

for i := 1 to NumberOfWords do

begin

lastSymbol := arrOfStr[i][Length(arrOfStr[i])];

delete(arrOfStr[i], Length(arrOfStr[i]), 1);

if pos(lastSymbol, arrOfStr[i]) > 0 then

write(concat(arrOfStr[i], lastSymbol), ' ')

else

arrOfStr[i] := arrOfStr[i] + lastSymbol;

end;

end;

procedure p2(var ArrOfString: Arr);

var

counter: integer;

begin

j := 1;

i := 1;

while i <= Length(ArrOfString) do

begin

counter := 0;

for j := 1 to Length(ArrOfString[i]) do

begin

if ArrOfString[i][j + counter] = 'q' then

begin

insert('u', ArrOfString[i], j + counter + 1);

Inc(counter);

end

else if (ArrOfString[i][j + counter] = 'q') and

(j = Length(ArrOfString[i])) then

ArrOfString[i] := ArrOfString[i] + 'u';

end;

Inc(i);

end;

for i := 1 to NumberOfWords do

write(ArrOfString[i], ' ');

end;

begin

SetConsoleCP(1251);

SetConsoleOutPutCP(1251);

writeln('Введиту строку на латинском:');

readln(s);

if s = '' then

writeln('Строка оказалась пустой, попробуйте еще раз!', s);

s := trim(s);

restouration(s);

getArrayOfString(s, ArrOfString);

writeln;

writeln('строка, отвечающая требованиям первого условия:');

p1(ArrOfString);

if ArrOfString[1] = '' then

writeln('Строка оказалась пустой, попробуйте еще раз!', ArrOfString[1]);

writeln;

writeln;

writeln('строка, отвечающая требованиям второго условия:');

p2(ArrOfString);

if ArrOfString[1] = '' then

writeln('Строка оказалась пустой, попробуйте еще раз!', ArrOfString[1]);

readln;

end.

* 1. Тестирование и отладка программы

Таблица 3.2 Прохождение тестов программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Специфика тестирования | Номер теста | Вводимые данные | Ожидаемый результат | Полученный результат |
| n = 10  m = 7 | 1 | 9 0 2 6 5 5 7  4 3 6 8 1 7 7  8 1 0 7 8 0 6  6 7 4 2 6 7 1  2 4 9 2 6 6 3  6 5 5 8 2 7 3  4 6 3 1 1 8 3  1 6 0 1 4 4 3  4 8 1 5 1 2 6  2 5 8 4 8 5 1 | 134 103 56 13 | Тест пройден |
| n = 7  m = 10 | 2 | 3 6 5 6 7 1 0  6 2 8 7 2 4 0  8 2 1 8 9 7 0  3 0 1 1 9 8 9  8 3 9 5 4 8 5  9 5 9 9 6 6 0  5 0 7 5 4 7 4  1 1 2 4 6 4 9  8 7 9 5 9 6 9  7 9 9 3 6 0 0 | 146 110 79 20 | Тест пройден |
| n = 2  m = 4 | 3 | 1 7 4 4  9 8 5 8 | 46 | Тест пройден |
| n = 8  m = 7 | 4 | 5 5 3 0 0 4 2  6 4 3 6 6 3 3  2 1 4 4 2 6 5  7 0 9 8 3 6 1  8 5 8 9 3 3 1  5 6 1 5 9 4 2  2 9 3 1 0 3 2  7 4 6 8 2 4 8  6 6 9 7 2 6 2  8 3 3 3 7 3 9 | 122 103 62 23 | Тест пройден |

* 1. Анализ прохождения тестов и полученных результатов

Программа прошла тесты успешно и ошибок не обнаружено.

Итоговый текст программы

program laba6;

{$APPTYPE CONSOLE}

{$R \*.res}

uses

System.SysUtils;

const

n = 5;

m = 7;

f = 10;

type

matrix = array [1 .. n, 1 .. m] of integer;

var

C: matrix;

i, j, sum, perimetrInColumn, perimetrInLine, perimetr, step, counter, finish,

q: integer;

res: string;

sum1,sum2,sum3, sum4: integer;

procedure vvod(var X: matrix);

var

i, j, sum: integer;

Var

k, h, step: byte;

begin

randomize;

for i := 1 to n do

for j := 1 to m do

X[i, j] := random(10);

end;

procedure print(var X: matrix);

var

i, j: integer;

begin

for i := 1 to n do

begin

for j := 1 to m do

write(X[i, j]:4);

writeln;

end;

end;

function getSumInColumn(C: matrix; number: integer): integer;

begin

result := 0;

for i := 1 to n do

begin

result := result + C[i, number];

end;

end;

procedure setZeroInColumn(var C: matrix; number: integer);

begin

for i := 1 to n do

begin

C[i, number] := 0;

end;

end;

function getSumInLine(C: matrix; number: integer): integer;

begin

result := 0;

for j := 1 to m do

begin

result := result + C[number, j];

end;

end;

procedure setZeroInLine(var C: matrix; number: integer);

begin

for j := 1 to m do

begin

C[number, j] := 0;

end;

end;

begin

writeln(' initial massive:');

writeln;

vvod(C);

print(C);

writeln;

if m mod 2 = 1 then

finish := (m div 2) + 1

else

finish := m div 2;

counter := 1;

for counter := 1 to finish do

begin

sum1:=getSumInColumn(C, counter);

setZeroInColumn(C, counter);

sum2:= getSumInColumn(C, m - counter + 1);

SetZeroInColumn(C, m - counter + 1);

sum3:=getSumInLine(C, counter) ;

setZeroInLine(C, counter);

sum4:= getSumInLine(C, n - counter + 1);

setZeroInLine(C, n - counter + 1);

perimetr := sum1+sum2+sum3+sum4;

res := res + ' ' + IntToStr(perimetr);

end;

writeln('result massive: ', res);

readln;

end.