Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ОТЧЕТ

по лабораторной работе №8

Тема работы: Строки

Выполнил

студент: гр. 251003 Панкратьев Е.С.

Проверил: Фадеева Е.П.

Минск 2022

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_Toc124097769)

[2 Методика решения 4](#_Toc124097770)

[2.1 Условия ввода 4](#_Toc124097771)

[2.2 Проверка введенных данных 4](#_Toc124097772)

[2.3 Нахождение слов, удовлетворяющие условиям 4](#_Toc124097773)

[3 Текстовый алгоритм решения задачи 5](#_Toc124097774)

[4 Структура данных 7](#_Toc124097775)

[5 Схема алгоритма решения задачи по Насси-Шнейдерману 8](#_Toc124097776)

[Приложение А 10](#_Toc124097777)

[Приложение Б 15](#_Toc124097778)

# Постановка задачи

Дана последовательность, содержашая от 2 до 50 слов, в каждом из которых от 1 до 8 строчных латинских букв; между соседними словами – запятая или не менее одного пробела, за последним словом – точка.

Напечатать те слова последовательности, которые отличны от последнего слова и удовлетворяют следующему свойству:

- слово начинается с гласной буквы и заканчивается согласной буквой.

# [Методика решения](#_Toc83996305)

## Условия ввода

Условия ввода:

* Последовательность должна содержать от 2 до 50 слов.;
* В словах должно быть от 1 до 8 строчных латинский букв;
* Между словами запятая или не менее одного пробела;
* После последнего слова должна стоять точка;

## Проверка введенных данных

Проверка введенных данных происходит с помощью цикла с предусловием repeat..until, чтобы при вводе некорректных данных пользователь заново заполнял их. В теле цикла проверяем корректность ввода. Также в теле цикла идет сразу упрощение строки: между соседними словами один пробел, в конце убирается точка.

## Нахождение слов, удовлетворяющие условиям

После проверки и упрощения строки находится последнее слово в последовательности. Потом проверяются все слова в последовательности (за исключением последнего) на соблюдения условий: проверяется первая буква, она должна быть гласной, последняя – согласной; далее сравнивается текущее слово с последним – они должны быть не равны.

# Текстовый алгоритм решения задачи

Таблица – Алгоритм решения

|  |  |
| --- | --- |
| Номер  шага | Назначение шага |
|  | Начало цикла А1 |
|  | isCorrect:= True |
|  | Ввод InitialStr |
|  | ProcessedStr:= Copy(InitialStr, Low(InitialStr), Length(InitialStr)) |
|  | Если условие ProcessedStr[High(ProcessedStr)] <> '.' истинно, перейти к шагу 6. Иначе перейти к шагу 8 |
|  | isCorrect:= False |
|  | Перейти к шагу 38 |
|  | WordCount:= 0 |
|  | i:= Low(ProcessedStr) |
|  | Начало цикла А1.1. Проверка выполнения условий (i < High (ProcessedStr)) and isCorrect. Если условия истинны, перейти к шагу 11, иначе – к шагу 36 |
|  | LetterCount:= 0 |
|  | Начало цикла А1.1.1. Проверка выполнения условий (ProcessedStr [i] <> ',') and (ProcessedStr[i] <> ' ') and (ProcessedStr[i] <> '.') and isCorrect. Если условия истинны, перейти к шагу 13, иначе – к шагу 18 |
|  | Если условие not (ProcessedStr[i] in ['a'..'z']) истинно, перейти к шагу 14. Иначе перейти к шагу 15 |
|  | isCorrect:= False |
|  | Inc(LetterCount) |
|  | Inc(i) |
|  | Конец цикла А1.1.1. Вернуться к шагу 12 |
|  | Inc(WordCount) |
|  | Если условие not (LetterCount in [MinLetters..MaxLetters]) and isCorrect истинно, перейти к шагу 20. Иначе перейти к шагу 22 |
|  | isCorrect:= False |
|  | Перейти к шагу 34 |
|  | Если условие ProcessedStr[i] = ',' истинно, перейти к шагу 23; если истинно условие ProcessedStr[i] = ' ', перейти к шагу 26; если истинно условие ProcessedStr[i] = '.', перейти к шагу 33. Иначе перейти к шагу 34 |
|  | Delete(ProcessedStr, i, 1) |
|  | Insert(' ', ProcessedStr, i) |
|  | Перейти к шагу 34 |
|  | j:= i |

Продолжение таблицы 1

|  |  |
| --- | --- |
|  | Начало цикла А1.1.2. Проверка выполнения условия not (ProcessedStr[j+1] in ['a'..'z']). Если условие истинно, перейти к шагу 28, иначе – к шагу 30 |
|  | Inc(j) |
|  | Конец цикла А1.1.2. Вернуться к шагу 27 |
|  | Если условие j <> i истинно, перейти к шагу 31. Иначе перейти к шагу 32 |
|  | Delete(ProcessedStr, i+1, j-i) |
|  | Перейти к шагу 34 |
|  | Delete(ProcessedStr, i, 1) |
|  | Inc(i) |
|  | Конец цикла А1.1. Вернуться к шагу 10 |
|  | Если условия not (WordCount in [MinWords..MaxWords]) and isCorrect истинны, перейти к шагу 37. Иначе перейти к шагу 38 |
|  | isCorrect:= False |
|  | Конец цикла А1. Проверка выполнения условия isCorrect. Если условие истинно, идти к шагу 39, иначе – к шагу 1 |
|  | LastWord:= Copy(ProcessedStr, High(ProcessedStr) - LetterCount + 1, LetterCount) |
|  | j:= Low(ProcessedStr) |
|  | LengthWithoutLastWord:= Length(ProcessedStr) - Length(LastWord) |
|  | Начало цикла А2. Проверка выполнения условия j < LengthWithoutLastWord. Если условие истинно, перейти к шагу 43, иначе – к шагу 54 |
|  | i:= j |
|  | FirstLastLetters[1]:= ProcessedStr[i] |
|  | Начало цикла А2.1. Проверка выполнения условия ProcessedStr [j+1] <> ' '. Если условие истинно, перейти к шагу 46, иначе – к шагу 48 |
|  | Inc(j) |
|  | Конец цикла А.2.1. Вернуться к шагу 45 |
|  | FirstLastLetters[2]:= ProcessedStr[j] |
|  | CurrWord:= Copy(ProcessedStr, i, j-i+1) |
|  | Если условия (Pos(FirstLastLetters[1], Vowels) <> 0) and (Pos(FirstLastLetters[2], Consonants) <> 0) and (CurrWord <> LastWord) истинны, перейти к шагу 51. Иначе перейти к шагу 52 |
|  | Вывод CurrWord |
|  | Inc(j, 2) |
|  | Конец цикла А2. Вернуться к шагу 42 |
|  | Останов. |

# Структура данных

Таблица 2 – Данные

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| Vowels | String | Гласные буквы |
| Consonants | String | Согласные буквы |
| MaxLetters | Integer | Максимальное количество букв в слове |
| MinLetters | Integer | Минимальное количество букв в слове |
| MaxWords | Integer | Максимальное количество слов в последовательности |
| MinWords | Integer | Минимальное количество слов в последовательности |
| InitialStr | String | Исходная строка |
| ProcessedStr | String | Упрощенная строка |
| LastWord | String | Последнее слово в последовательности |
| CurrWord | String | Текущее слово |
| FirstLastLetters | Array[1..2] of Char | Первая и последняя буква в текущем слове |
| i | Integer | Счетчик цикла |
| j | Integer | Счетчик цикла |
| WordCount | Integer | Счетчик количества слов |
| LetterCount | Integer | Счетчик количества букв в слове |
| LengthWithoutLastWord | Integer | Длина последовательности без последнего слова |
| isCorrect | Integer | Идентификатор правильного ввода |

# Схема алгоритма решения задачи по Насси-Шнейдерману

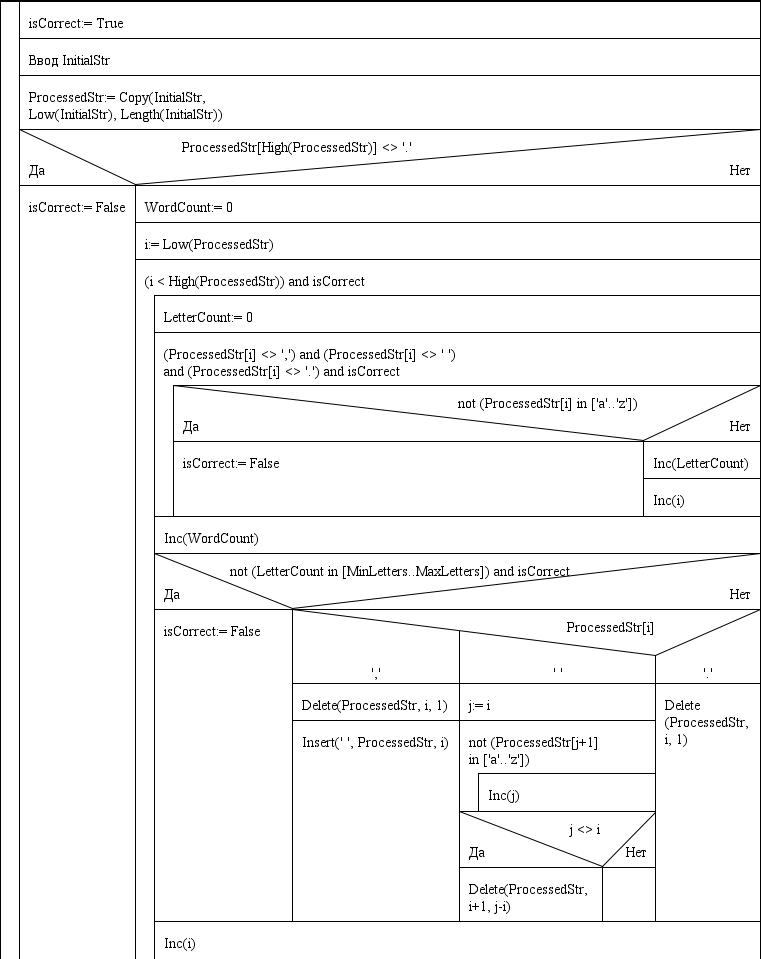
****

Рисунок – Схема алгоритма решения задачи по Насси-Шнейдерману (часть 1)

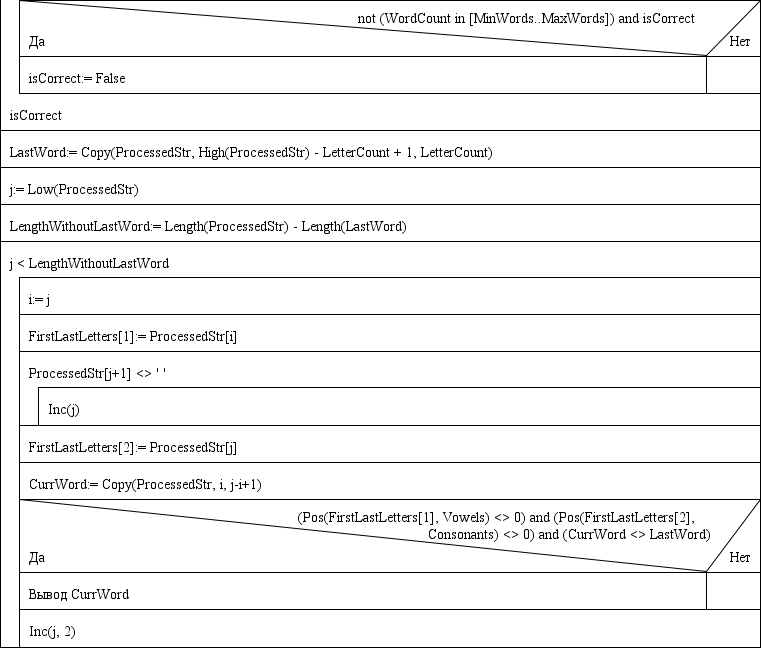


Рисунок – Схема алгоритма решения задачи по Насси-Шнейдерману (часть 2)

Приложение А

(обязательное)

Исходный код программы

Program Lab8ProcessedString;

{

Given a sequence of 2 to 50 words, each with 1 to 8 lowercase

letters. Between adjacent words a comma or at least one

space, after the last word a dot. Output words that are

different from the last word, the word starts with a vowel

and ends with a consonant

}

{$APPTYPE CONSOLE}

uses

System.SysUtils;

Const

Vowels = 'aeiouy';

Consonants = 'bcdfghjklmnpqrstvwxz';

MaxLetters = 8;

MinLetters = 1;

MaxWords = 50;

MinWords = 2;

//Vowels - vowel letters

//Consonants - consonant letters

//MaxLetters - maximum amount of letters in a word

//MinLetters - minimum amount of letters in a word

//MaxWords - maximum amount of words in a sequence

//MinWords - minimum amount of words in a sequence

Var

InitialStr, ProcessedStr, LastWord, CurrWord: string;

FirstLastLetters: array[1..2] of char;

i,j,WordCount,LetterCount,LengthWithoutLastWord: integer;

isCorrect : boolean;

//InitialStr - initial string

//ProcessedStr - processed string

//LastWord - last word in the sequence

//CurrWord - current check word

//FirstLastLetters - the first and last letter in the

//current check word

//i, j - cycle counter

//WordCount - word count

//LetterCount - letter count

//LengthWithoutLastWord - sequence length without last word

//isCorrect - flag to confirm the correctness of entering

//numbers

Begin

Writeln('Enter a sequence. Words should be from 2 to 50,

each with 1 to 8 lowercase letters.');

Writeln('A virgule or at least one space between words.

There should be a dot at the end.');

Writeln('Output words that are different from the last

word, the word starts with a vowel and ends with

a consonant');

//Cycle with postcondition for entering correct data

repeat

//Initialize the flag

isCorrect:= True;

Readln(InitialStr);

//Сopy the initial string into a processed string

ProcessedStr:= Copy(InitialStr, Low(InitialStr),

Length(InitialStr));

//The sequence must end with a dot

if ProcessedStr[High(ProcessedStr)] <> '.' then

begin

Writeln('Should be a dot at the end');

isCorrect:= False;

end

else

begin

//Reset counter

WordCount:= 0;

//Checking and processing the sequence

i:= Low(ProcessedStr);

while (i < High(ProcessedStr)) and isCorrect do

begin

//Сheck the words separately.

LetterCount:= 0;

while (ProcessedStr[i] <> ',') and (ProcessedStr[i]

<> ' ') and (ProcessedStr[i] <> '.') and

isCorrect do

begin

//Сhecking for valid symbols

if not (ProcessedStr[i] in ['a'..'z']) then

begin

Writeln('Found Invalid character! Enter the

sequence again');

isCorrect:= False;

end;

//Increasing the amount of letters

Inc(LetterCount);

//Modernize i

Inc(i);

end;

//Increasing the amount of words

Inc(WordCount);

//Validity check of the amount of letters

if not (LetterCount in [MinLetters..MaxLetters]) and

isCorrect then

begin

Writeln('The word must be from ',MinLetters,' to

',MaxLetters,' letters');

isCorrect:= False;

end

//Removing unnecessary symbols: punctuation marks and

//extra spaces

else

case ProcessedStr[i] of

',':

begin

Delete(ProcessedStr, i, 1);

Insert(' ', ProcessedStr, i);

end;

' ':

begin

j:= i;

while not (ProcessedStr[j+1] in ['a'..'z']) do

Inc(j);

if j <> i then

Delete(ProcessedStr, i+1, j-i);

end;

'.': Delete(ProcessedStr, i, 1);

end;

Inc(i);

end;

//Validity check of the amount of words

if not (WordCount in [MinWords..MaxWords]) and

isCorrect then

begin

Writeln('The sequence must contain from ',MinWords,'

to ',MaxWords,' words');

isCorrect:= False;

end;

end;

until isCorrect;

//Finding the last word

LastWord:= Copy(ProcessedStr, High(ProcessedStr) –

LetterCount + 1, LetterCount);

Writeln('Words satisfying the requirement:');

//Checking words (except the last one)

j:= Low(ProcessedStr);

LengthWithoutLastWord:= Length(ProcessedStr) –

Length(LastWord);

while j < LengthWithoutLastWord do

begin

//Find the first letter

i:= j;

FirstLastLetters[1]:= ProcessedStr[i];

//Find the last letter

while ProcessedStr[j+1] <> ' ' do

Inc(j);

FirstLastLetters[2]:= ProcessedStr[j];

//Find the current word

CurrWord:= Copy(ProcessedStr, i, j-i+1);

//Condition check: first letter is a vowel, second is a

//consonant, not equal to the last word

if (Pos(FirstLastLetters[1], Vowels) <> 0) and

(Pos(FirstLastLetters[2], Consonants) <> 0) and

(CurrWord <> LastWord) then

Write(CurrWord,' ');

//Modernize j

Inc(j, 2);

end;

Readln;

End.

Приложение Б

(обязательное)

Тестовые наборы

**Тестовая ситуация: некорректный ввод данных**

**Тест 1**

Исходные данные: Некорректный ввод дат

Ожидаемый результат: Повторная попытка

Полученный результат:

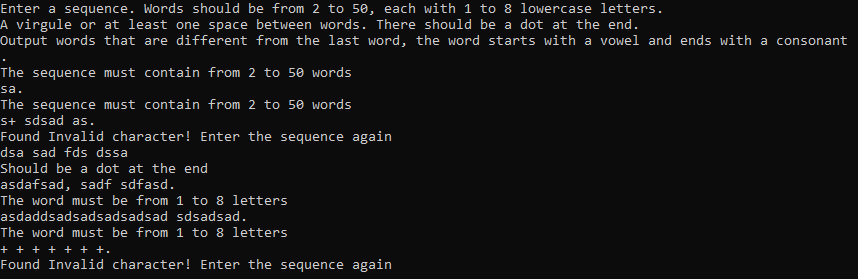
****

Рисунок 3 – Результаты расчетов

**Тестовая ситуация: корректный ввод данных**

**Тест 2**

Исходные данные:

asd esd,sad ass ass ass ass ass sas ass.

Ожидаемый результат:

asd esd

Полученный результат:

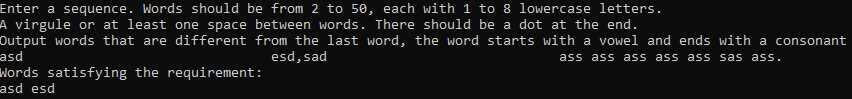


Рисунок 4 – Результаты расчетов

**Тест 3**

Исходные данные:

d a s a d asd dads adsa sdasd sdad sads sdad sadsa sdadas,sdas sdad,asd,

dasdaa,ass asd,dasss,sas,ass.

Ожидаемый результат:

asd asd asd

Полученный результат:

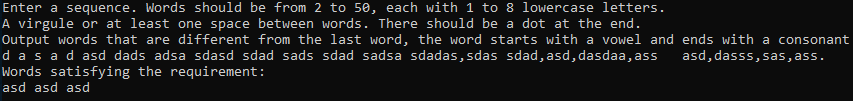


Рисунок 5 – Результаты расчетов

**Тест 4**

Исходные данные:

aeeed aeeed aeeed aeeed aeeed aeeed aeeed aeeed eeeea eedd aeeed aeeed.

Ожидаемый результат:

eedd

Полученный результат:

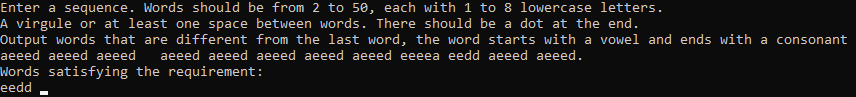
****

Рисунок 6 – Результаты расчетов

**Тест 5**

Исходные данные:

asasasss sas sas asd asd asd asd asd.

Ожидаемый результат:

asasasss

Полученный результат:

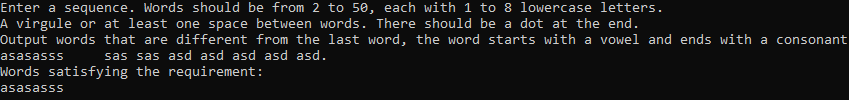


Рисунок 7 – Результаты расчетов

**Тест 6**

Исходные данные:

iis iis isss sis asd iis.

Ожидаемый результат:

isss asd

Полученный результат:

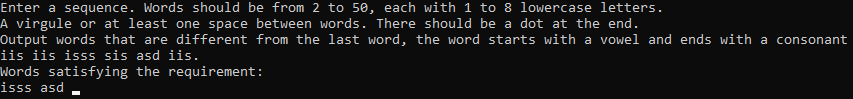


Рисунок 8 – Результаты расчетов

**Тест 7**

Исходные данные:

oop oop opp opooo opopp pop pop pop pop.

Ожидаемый результат:

oop oop opp opopp

Полученный результат:

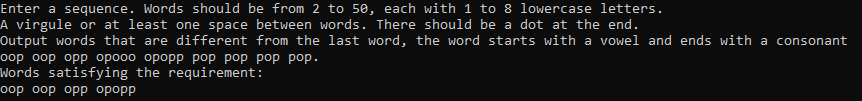


Рисунок 9 – Результаты расчетов