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:

Строки

|  |  |  |
| --- | --- | --- |
| Выполнил  Студент гр. 451001 |  | Д. Д. Хренков |
| Проверил |  | Асс. Е.Е. Фадеева |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Минск, 2024

Теоретические сведения по теме лабораторной работы

Строковый тип — тип данных, значениями которого является произвольная последовательность символов.

Строковые данные могут быть:

• константами

• переменными

Строковый литерал – последовательность символов, заключенная в апострофы. Если в строке должен быть символ апострофа - он удваивается.

Существует несколько видов строковых переменных:

• строки постоянной длины – такая строка обладает всеми свойствами массива. Можно присваивать строковые литералы и можно сравнивать строки одинаковой длины

• строки переменной длины

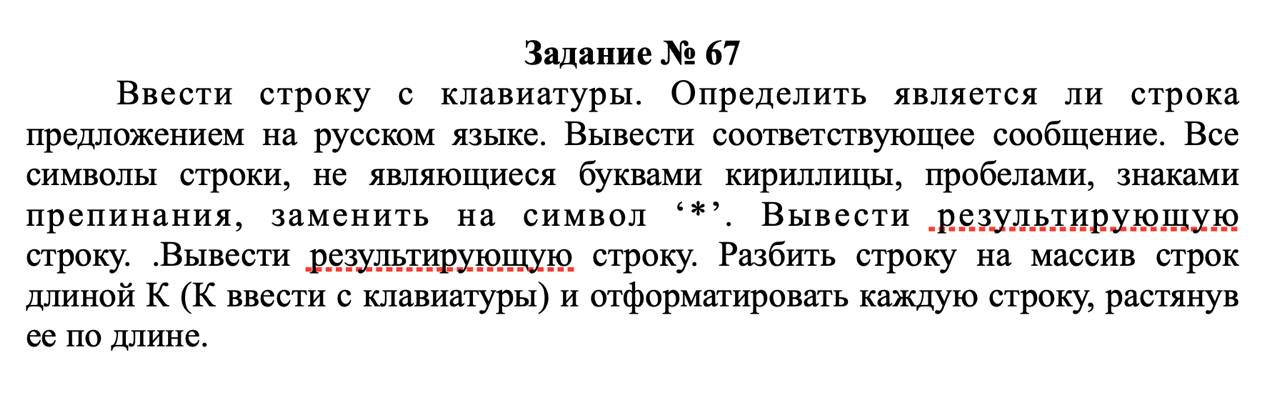
• Delphi строки

Доступ к символам строки по индексам.

• нумерация начинается с 1

• обращения к символу с индексом не из диапазона 1..L аналогично выходу за границы массива (L - длина строки)

1. Задание на лабораторную работу

Постановка задачи

Система правил:

1. Длина строки больше одного символа;
2. Первая буква – заглавная;
3. Последний символ строки либо точка, либо восклицательный знак, либо вопросительный знак;
4. Перед знаками препинания не может быть пробела;
5. В строке не может быть несколько пробелов подряд;
6. В строке не встречаются символы, не являющиеся буквами кириллицы, знаками препинания и пробелами.
7. Если в предложении встречается точка, то она должна быть последним символом, в противном случае эта точка формирует троеточие с последними тремя символами строки и при этом длина строки обязательно более трёх;
8. Все слова строки содержатся в словаре русских слов.

Схема алгоритма по методу Насси-Шнейдермана:

(показать)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя подпрограммы | Назначение подпрограммы | Заголовок подпрограммы | Имя параметра | | Назначение параметра |
| LoadDict | Заносит в массив слова из текстового файла словаря | procedure LoadDict(fname: TString; var isCorrect: boolean; var dict: TDict); | fname | | Имя файла |
| isCorrect | | Флаг для проверки правильности открытия файла |
| dict | | Массив для слов из словаря |
| getWords | Записывает в массив слова из строки по допустимым символам | procedure getWords(const s, letters: TString; var words: TMas;  var len: integer); | s | | Массив для сортироки |
| letters | | Допустимые символы |
| words | | Массив слов для записи |
| len | | Последний индекс, по которому лежит слово |
| CheckInDict | Проверяет, есть ли слова из массива строк в словаре | function CheckInDict(const dict: TDict; var words: TMas;  const wordsLen: integer): boolean; | words | | Массив слов |
| wordsLen | | Последний индекс, по которому лежит слово |
| dict | | Словарь слов |
| CheckString | Проверяет, является ли строка предложением на русском языке | function CheckString(const s, UpperLetters, approvedSymbols: TString;  const dict: TDict): boolean; | s | | Строка для проверки |
| UpperLetters | | Буквы верхнего регистра |
| approvedSymbols | | Допустимые символы |
| dict | | Словарь слов |
| formatString | Форматирует строку | function formatString(const s, approvedSymbols: TString): TString; | s | | Строка для форматирования |
| approvedSymbols | | Допустимые символы |
| stretchline | Растягивает строку по ширине | function stretchline(line: TString; k: integer): string; | line | Срока для растягивания | |
| k | Ограничитель длины | |
| stretchWords | Возвращает массив растянутых по ширине строк | procedure stretchWords(const words: TMas; const len, k: integer;  var lines: TMas; var count: integer; var available: boolean); | words | Массив слов | |
| len | Последний индекс, по которому лежит слово | |
| k | Ограничитель длины | |
| lines | Массив растянутых по ширине строк | |
| count | Последний индекс, по которому лежит растянутая в ширину строка | |
| available | Флаг, возможно ли растянуть строки | |

Выделение основных структур данных

|  |  |  |
| --- | --- | --- |
| Имя идентификатора структуры | Назначение структуры | Тип структуры |
| TString | Типизированная строка | String[255] |
| TMAS | Массив типизированных строк | array [1 .. N] of TString |
| TDict | Структура для словаря слов | array [1 .. wordsCount] of TString |

Тестирование и отладка программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Специфика тестирования | Номер теста | Вводимые данные | Ожидаемый результат | Полученный результат |
| Строка, являющаяся предложением на русском языке | 1 | Обычное предложение.  15 | Строка является предложением на русском языке  Отформатированная строка: Обычное предложение.  Растянутая в ширину по k:  Обычное  предложение. | Как и ожидалось |
| Строка, без завершающего символа | 2 | Без точки в конце  5 | Строка не является предложением на русском языке  Отформатированная строка: Без точки в конце  Растянутая в ширину по k:  Без  точки  в  конце | Как и ожидалось |
| Строка с числом в цифровой записи | 3 | Около 15000 людей.  15 | Строка не является предложением на русском языке  Отформатированная строка: Около \*\*\*\*\* людей.  Растянутая в ширину по k:  Около \*\*\*\*\*  людей. | Как и ожидалось |
| Строка, со случайным набором букв вместо слов | 4 | Жэхзл в Юбль лд?  8 | Строка не является предложением на русском языке  Отформатированная строка: Жэхзл в Юбль лд?  Растянутая в ширину по k:  Жэхзл в  Юбль  лд? | Как и ожидалось |
| Строка, начинающаяся со строчной буквы | 5 | с маленькой буквы.  10 | Строка не является предложением на русском языке  Отформатированная строка: с маленькой буквы.  Растянутая в ширину по k:  с  маленькой  буквы. | Как и ожидалось |
| Строка, являющаяся предложением на русском языке, но невозможно растянуть в ширину из-за k | 6 | Огромное слово!  6 | Строка является предложением на русском языке  Отформатированная строка: Огромное слово!  Невозможно растянуть строку | Как и ожидалось |
| Строка, содержащая слово с дефисом | 7 | Когда-нибудь мы встретимся снова.  15 | Строка является предложением на русском языке  Отформатированная строка: Когда-нибудь мы встретимся снова.  Растянутая в ширину по k:  Когда-нибудь  мы встретимся  снова. | Как и ожидалось |
| Строка с плохо расставленными пробелами | 8 | Я пьяный помогите , пожалуйста.  20 | Строка не является предложением на русском языке  Отформатированная строка: Я пьяный помогите , пожалуйста.  Растянутая в ширину по k:  Я пьяный помогите ,  пожалуйста. | Как и ожидалось |
| Строка с троеточием | 9 | Не так я себе представлял выходные...  15 | Строка является предложением на русском языке  Отформатированная строка: Не так я себе представлял выходные...  Растянутая в ширину по k:  Не так я себе  представлял  выходные... | Как и ожидалось |
| Строка с точкой в середине | 10 | Кто-то случайно. тыкнул точку.  10 | Строка не является предложением на русском языке  Отформатированная строка: Кто-то случайно. тыкнул точку.  Растянутая в ширину по k:  Кто-то  случайно.  тыкнул  точку. | Как и ожидалось |

program lab7\_2;

{$APPTYPE CONSOLE}

uses

SysUtils;

const

N = 255;

wordsCount = 1532629;

UpperLetters = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЭЮЯ';

approvedSymbols =

' ,-:;АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчшщъыьэюя.?!';

type

TString = String;

TMas = array [1 .. N] of TString;

TDict = array [1 .. wordsCount] of TString;

procedure LoadDict(fname: TString; var isCorrect: boolean; var dict: TDict);

var

f: textfile;

s: TString;

i: integer;

begin

i := 0;

if FileExists(fname) then

begin

AssignFile(f, fname);

Reset(f);

while (not EOF(f)) do

begin

Inc(i);

Readln(f, s);

dict[i] := s;

end;

CloseFile(f);

end

else

isCorrect := false;

end;

procedure getWords(const s, letters: TString; var words: TMas;

var len: integer);

var

i: integer;

currentWord: TString;

begin

len := 0;

currentWord := '';

for i := 1 to Length(s) do

begin

if Pos(s[i], letters) <> 0 then

currentWord := currentWord + s[i]

else if Length(currentWord) > 0 then

begin

Inc(len);

words[len] := currentWord;

currentWord := '';

end;

end;

if Length(currentWord) > 0 then

begin

Inc(len);

words[len] := currentWord;

end;

end;

function CheckInDict(const dict: TDict; var words: TMas;

const wordsLen: integer): boolean;

var

low, high, index, i: integer;

notFound, inSearch, extra: boolean;

begin

i := 1;

notFound := false;

extra := false;

while (i <= wordsLen) and (not notFound) do

begin

low := 1;

high := wordsCount;

inSearch := True;

while inSearch do

begin

index := ((high + low) div 2);

if (high = low) and (dict[high] <> words[i]) then

begin

if (i = 1) and not(extra) then

begin

words[i] := ansilowercase(words[i]);

extra := True;

inSearch := false;

end

else

begin

writeln('(debug) Not found: ', words[i]);

extra := False;

notFound := True;

inSearch := false;

end;

end

else if dict[index] = words[i] then

begin

inSearch := False;

extra := False;

end

else if dict[index] < words[i] then

low := index + 1

else if dict[index] > words[i] then

high := index;

end;

if not extra then

Inc(i);

end;

CheckInDict := not notFound;

end;

function CheckString(const s, UpperLetters, approvedSymbols: TString;

const dict: TDict): boolean;

var

i, len, spaces, wordsLen: integer;

spacesFlag: boolean;

words: TMas;

begin

CheckString := True;

len := Length(s);

if (len = 0) or (len = 1) then

CheckString := false

else if Pos(s[1], UpperLetters) = 0 then

CheckString := false

else if not((s[len] = '.') or (s[len] = '!') or (s[len] = '?')) then

CheckString := false

else if s[len - 1] = ' ' then

CheckString := false;

spaces := 0;

spacesFlag := True;

i := 2;

while (i <= len - 1) and (spacesFlag) do

begin

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

begin

spacesFlag := false;

CheckString := false;

end

else if (s[i] = ' ') and (Pos(s[i + 1], ',.:;') <> 0) then

CheckString := false

else if s[i] = ' ' then

Inc(spaces)

else

spaces := 0;

if Pos(s[i], Copy(approvedSymbols, 1, 72)) = 0 then

CheckString := false

else if (s[i] = '.') and not((len - 2 <= i) and (i <= len)) then

CheckString := false

else if (s[i] = '.') and not((s[len] = '.') and (s[len - 1] = '.') and

(s[len - 2] = '.')) then

CheckString := false;

Inc(i);

end;

getWords(s, Copy(approvedSymbols, 6, 66), words, wordsLen);

if not CheckInDict(dict, words, wordsLen) then

CheckString := false;

end;

function formatString(const s, approvedSymbols: TString): TString;

var

i: integer;

formatted: TString;

begin

formatted := '';

for i := 1 to Length(s) do

begin

if Pos(s[i], approvedSymbols) = 0 then

formatted := formatted + '\*'

else

formatted := formatted + s[i];

end;

formatString := formatted;

end;

function stretchline(line: TString; k: integer): string;

var

words: array of TString;

i, start, wordcount, totalspaces, spaceslots, extraspaces, spaceperslot: integer;

newline: TString;

begin

line := trim(line);

wordcount := 0;

i := 1;

while i <= Length(line) do

begin

while (i <= Length(line)) and (line[i] = ' ') do

Inc(i);

if i <= Length(line) then

begin

start := i;

while (i <= Length(line)) and (line[i] <> ' ') do

Inc(i);

Inc(wordcount);

setlength(words, wordcount);

words[wordcount - 1] := Copy(line, start, i - start);

end;

end;

if wordcount > 1 then

begin

totalspaces := k - Length(line) + (wordcount - 1);

spaceslots := wordcount - 1;

spaceperslot := totalspaces div spaceslots;

extraspaces := totalspaces mod spaceslots;

end

else

begin

spaceperslot := 0;

extraspaces := 0;

end;

newline := '';

for i := 0 to wordcount - 2 do

begin

newline := newline + words[i] + stringofchar(' ', spaceperslot);

if extraspaces > 0 then

begin

newline := newline + ' ';

dec(extraspaces);

end;

end;

newline := newline + words[wordcount - 1];

stretchline := newline;

end;

procedure stretchWords(const words: TMas; const len, k: integer;

var lines: TMas; var count: integer; var available: boolean);

var

i: integer;

line, word: TString;

begin

available := True;

count := 1;

line := '';

word := '';

for i := 1 to len do

begin

if Length(words[i]) > k then

available := false;

if Length(words[i]) = k then

begin

if Length(line) > 0 then

begin

lines[count] := stretchline(line, k);

Inc(count);

line := '';

end;

lines[count] := stretchline(words[i], k);

Inc(count);

end

else if (Length(line) + Length(words[i]) + 1 > k) and available then

begin

lines[count] := stretchline(line, k);

Inc(count);

line := words[i] + ' ';

end

else

begin

line := line + words[i] + ' ';

end;

if (i = len) and (Length(line) > 0) and available then

lines[count] := stretchline(line, k);

end;

end;

var

s, formattedStr: TString;

dict: TDict;

words, lines: TMas;

k, wordsLen, linesCount, i: integer;

isCorrect: boolean;

begin

isCorrect := True;

LoadDict('../russian\_sorted.txt', isCorrect, dict);

if not isCorrect then

begin

writeln('Файл словаря недоступен или повреждён');

Readln;

exit;

end;

Write('Введите строку: ');

Readln(s);

Write('Введите k: ');

Readln(k);

writeln;

if CheckString(s, UpperLetters, approvedSymbols, dict) then

writeln('Строка является предложением на русском языке')

else

writeln('Строка не является предложением на русском языке');

formattedStr := formatString(s, approvedSymbols);

writeln;

writeln('Отформатированная строка: ', formattedStr);

getWords(formattedStr, Copy(approvedSymbols, 2, 73), words, wordsLen);

stretchWords(words, wordsLen, k, lines, linesCount, isCorrect);

writeln;

if isCorrect then

begin

writeln('Растянутая в ширину по k:');

for i := 1 to linesCount do

writeln(lines[i]);

end

else

writeln('Невозможно растянуть строку');

Readln;

end.