# Лабораторная работа № 6

«Программирование операций над строками и текстовыми файлами»

# Цель работы

Изучение основных операций над множествами, строками и файлами, программирование операций обработки строк, текстовых файлов, исследование свойств файловых переменных.

# Постановка задачи

Научиться эффективно использовать множества, строки и файловый тип данных.

# Ход выполнения работы

# Описание алгоритма решения задачи

Вариант выполнения работы – 19.

# Определение способа представления исходных данных

Данные в файле представлены в виде текста, который записан в файл. Кодировка файла – ANSI. Текст состоит из неограниченного количества предложений. Для корректного отображения предложений рекомендуется размещать всё предложение на одной строке.

* + - 1. Определение входных и выходных данных

На вход поступает путь к файлу, из которого программа может взять текст.

На выходе программа печатает сначала предложения, начинающиеся с однобуквенных слов, а затем предложений, начинающихся с многобуквенных слов.

# Структурная схема алгоритма

Структурные схемы алгоритма представлены на рисунках 6.1 и 6.2.

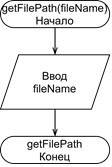


Рисунок 6.1 – Структурная схема процедуры getFilePath

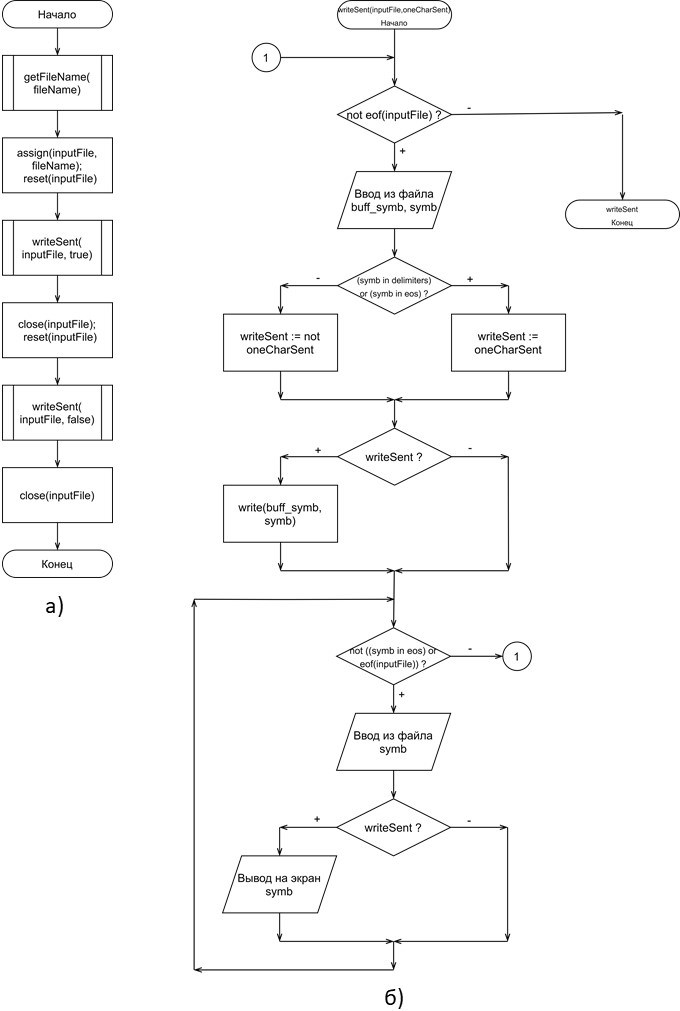


Рисунок 6.2 – Структурные схемы: а) основной программы; б) процедуры writeSent

# Текст программы на языке программирования Pascal

Program LW\_6\_19;

Uses Crt;

Type type\_fN = string[80];

Var fileName: type\_fN;

inputFile: text;

{процедура-для-получения-от-пользователя-пути-к-файлу}

{с-текстом-и-проверкой-на-существование-файла------------}

procedure getFilePath(var fileName: type\_fN);

Begin

write('Введите путь к файлу: '); readln(fileName);

while not fileExists(fileName) do begin

writeln('(x) Ошибка: файла "', fileName, '" не существует.');

write('Введите путь к файлу ещё раз: '); readln(fileName)

end

End;

{-Процедура-печати-предложений-из-файла.-Если-второй-параметр}

{-true,-то-выводятся-предложения,-начинающиеся-с-одной-буквы----}

procedure writeSent(var inputFile: text; const oneCharSent: boolean);

Const delimiters = [' ', ',', ':', ';', '-', '—', #10, #13];

eos = ['!', '?', '.'];

Var writeSent: boolean;

symb, buff\_symb: char;

begin

while not eof(inputFile) do

begin

buff\_symb := ' ';

//-Считываение-первой-буквы

while buff\_symb in delimiters do

read(inputFile, buff\_symb);

//-Считывание-следующего-символа

read(inputFile, symb);

//-Если-второй-символ-в-предложении-есть-буква,-то

//-предложение-начинается-с-многобуквенного-слова

if ( (symb in delimiters) or (symb in eos) ) then

writeSent := oneCharSent

else

writeSent := not oneCharSent;

if writeSent then write(buff\_symb, symb);

while not ( (symb in eos) or eof(inputFile) ) do

begin

read(inputFile, symb);

if writeSent then

begin

write(symb);

if symb in eos then writeln();

end;

end;

end;

end;

BEGIN

//-Получение-пути-к-файлу

getFilePath(fileName);

//-Открытие-файла-для-чтения

assign(inputFile, fileName);

reset(inputFile);

writeln('Предложения, начинающиеся с однобуквенных слов:');

writeSent(inputFile, true);

//-Переоткрытие-файла

close(inputFile);

reset(inputFile);

writeln();

writeln('Остальные предложения:');

writeSent(inputFile, false);

//-Конец-программы

close(inputFile);

writeln('Конец работы программы. Нажмите на любую кнопку.');

readkey();

halt(0)

END.

# Описание тестовых примеров

Тестовые примеры представлены на Рисунках 6.2-6.4.

На Рисунке 6.2 показано содержимое каталога. Содержимое состоит из исполняемого файла основной программы app.new.exe и двух текстовых файлов file.txt и bp.txt.

На Рисунке 6.3а показано содержимое файла, а на Рисунке 6.3б – результат работы программы, если указано только имя файла, находящегося в этом же каталоге. Программа работает верно.

На Рисунке 6.4а показано содержимое файла, состоящее из предложений как с латинскими буквами, так и с кириллицей. На Рисунке 6.4б продемонстрирована работа программы. Сначала введён неверный путь, из-за чего программа показывает соответствующую ошибку и просит ввести верный путь. Затем введён полный путь к файлу. Программа работает верно, независимо от языка раскладки клавиатуры и полноты введённого пути.

Также было проведено дополнительное исследование, целью которого являлось выявление значения файловой переменной до и после выполнения процедуры Assign, а также до и после выполнения процедуры Reset. Результаты отображены на Рисунке 6.5.

Исследование показало, что до выполнения процедуры Assign файловая переменная объявлена, но не проинициализирована (Рисунок 6.5а). После выполнения Assign файловая переменная хранит основную информацию о файле (такую как Имя файла, Путь к файлу, Права доступа) (Рисунок 6.5б). После выполнения процедуры Reset в переменную также добавлена информация о том, как работать с файлом. В данном случае – чтение данных из файла (Рисунок 6.5в).

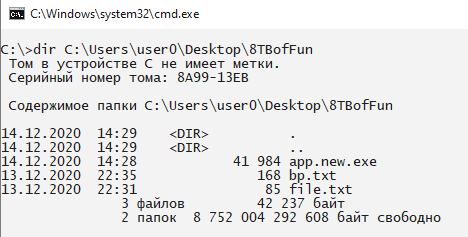
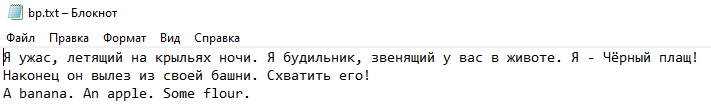


Рисунок 6.2 - Содержимое каталога

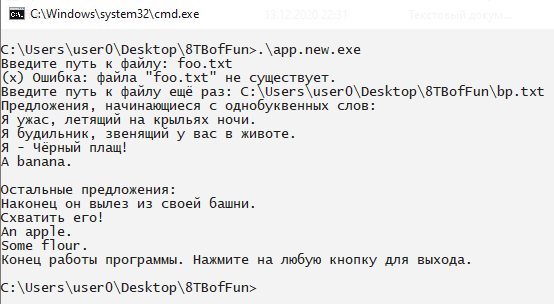
|  |  |
| --- | --- |
| а) | б) |

Рисунок 6.3 – Тестовый пример № 1. а) Содержимое текстового файла file.txt;

б) Результат работы программы



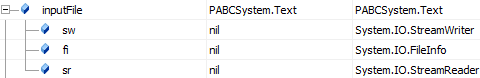
а)



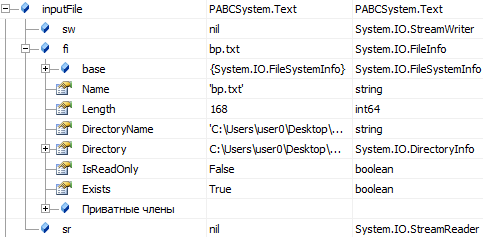
б)

Рисунок 6.4 – Тестовый пример № 2. а) Содержимое текстового файла bp.txt;

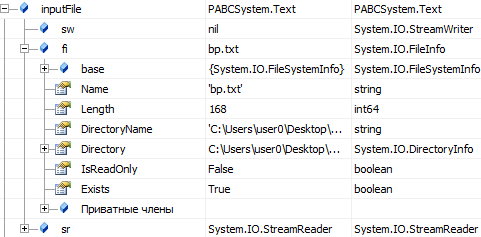
б) Результат работы программы



а)



б)



в)

Рисунок 6.5 – Результаты исследований значения файловой переменной:

а) до выполнения Assign; б) после выполнения Assign и до выполнения Reset;

в) после выполнения Reset

# Вывод

В ходе выполнения данной лабораторной работы были получены навыки разработки приложений, умеющих работать с пользовательскими файлами; были выявлены особенности работы с множествами, строками и файлами в языке программирования Pascal. Также было проведено исследование значения файловой переменной до и после выполнения процедуры Assign, а также до и после выполнения процедуры Reset. Установлено, что процедура Assign связывает файловую переменную с файлом, а Reset – сообщает программе, как работать с файлом: в режиме только чтения, только записи или чтения и записи.