Лабораторная №2: коллекции и ввод-вывод

1. Построение частотного словаря

Реализуйте приложение, которое для заданных текстов на русском языке вычисляет частоту используемых слов. Обрабатываемые тексты должны загружаться из заданных текстовых файлов, имена которых передаются через аргументы командной строки.

Для вычисления частоты используемых слов тексты должны быть разделены на слова. Словом является последовательность букв, записанная кириллицей. Слова отделяются друг от друга пробелами и знаками препинания, которые должны игнорироваться. При обработке знаков препинания нужно учитывать, что есть слова, включающие в себя дефис, например «северо-запад», «какой-нибудь», «серо-буро-малиновый» и т. п., которые так же должны учитываться. При вычислении частоты регистр букв должен игнорироваться.

В результате вычислений должно быть сформировано несколько отчётов, содержащих одни и те же данные — слова, их абсолютную частоту (то, сколько раз слово встретилось в текстах) и относительную частоту (абсолютная частота, делённая на общее количество слов в обработанных текстах). Отчёты с данными, упорядоченными различными способами, должны быть сохранены в файлы:

- `report-by-alph.txt` должен содержать отчёт со списком слов, отсортированных по алфавиту;
- `report-by-alph-rev.txt` должен содержать отчёт со списком слов, отсортированных по принципу обратного словаря — словаря, в котором слова отсортированы не по начальным, а по конечным буквам; другими словами, в обратном словаре сначала перечисляются слова, заканчивающиеся на «а», потом на «б» и так далее; среди слов, заканчивающихся на одну и ту же букву, сначала идут слова, у которых вторая буква с конца — это буква «а», потом «б» и т. д.;
- `report-by-freq.txt` должен содержать отчёт со списком слов, отсортированных по убыванию частоты; при совпадении частот слова должны быть упорядочены по алфавиту.

Если при чтении любого из исходных файлов возникает ошибка (файл не существует, ошибка чтения и т. п.), то пользователю должно быть выведено сообщение с описанием проблемы, после чего работа программы должна быть завершена.

Если при запуске приложения файлы для обработки не указаны, то программа должна вывести сообщение, описывающее способ её использования.

*В классе, вычисляющим количество слов, реализуйте метод, возвращающий словарь найденных слов. Словарь должен быть возвращён так, чтобы его изменение вне класса, вычисляющего частоту, было невозможно.

2. Реализация скриптового языка

Напишите программу, реализующую интерпретатор скриптового языка, который позволяет:

- складывать и вычитать числа друг с другом;
- сохранять результат вычисления в переменных;
- использовать переменные в вычислениях;
- печатать строки и переменные на экран.

Вычисления производятся в целых числах.

Скрипт загружается из текстового файла и выполняется построчно. Строка может содержать операторы `set` или `print`, комментарии, пустые строки:

- оператор `set` вычисляет значение заданного математического выражения и присваивает результат указанной переменной; выражение может содержать числа, другие переменные, операции сложения и вычитания;
- оператор `print` печатает на экран заданный список строк и переменных; строки и переменные в списке разделяются запятыми; строки заключаются в двойные кавычки;
- строки, начинающиеся с хеш-символа (символа решётки, «#»), должны игнорироваться;
- пустые строки также должны игнорироваться.

Имя переменной предваряется знаком доллара («$») и состоит из английских букв, цифр и знака подчёркивания. Например: `$n`, `$var_name`, `$longVarName`, `$42`.

Если в выражении оператора `set` или в операторе `print` используются неизвестные переменные, то выполнение скрипта должно быть прервано, а пользователю должно быть показано соответствующее сообщение об ошибке.

Имя файла с текстом скрипта указывается при запуске приложения через аргумент командной строки. Если при запуске аргумент приложения не задан, то программа должна вывести сообщение, описывающее способ её использования.

Если при чтении файла возникают ошибки (файл не существует, ошибка чтения и т.п.), то пользователю должно быть выведено текстовое сообщение с описанием проблемы, после чего работа программы должна быть завершена.

Пример использования

Например, при запуске программы, которой передано имя файла со следующим содержимым

```
# Printing "Hello, world!!!" phrase
print "Hello, world!!!"

# Printing "21+121-42 = 100"
set $n1 = 21
set $n2 = 121
set $sum = $n1 + $n2 - 42
print "$sum = ", $n1, "+", $n2, "-42 = ", $sum
```

должно печататься

```
Hello, world!!!
$sum = 21+121-42 = 100
```

Предполагается, что скрипт записан корректно и не содержит ошибок, кроме ошибок в именах переменных.

*Дополнительные задания (при отправке на проверку укажите в письме, какие пункты вы выполнили):

1) реализуйте оператор `input`, при помощи которого скрипт может вывести заданную строку приглашения к вводу, запросить у пользователя числовое значение и сохранить его в заданную переменную (например, «`input "What is your age: ", $age`»); если переменной не существует, то оператор должен её создать;
2) реализуйте обработку строковых литералов в операторах `print` и `input` (если реализовали) так, чтобы они могли содержать символ двойной кавычки;

3) реализуйте интерполяцию строк: при выводе строкового литерала на экран, если он содержит имя переменной, то вместо переменной выводится её значение (например, если есть переменная `$n1`, и её значение равно «42», то оператор «`print "num = $n1"`» должен вывести «`num = 42`»); если переменной нет, то должна выводиться ошибка, а работа скрипта завершаться; реализуйте интерполяцию так, чтобы осталась возможность печати символа доллара («$»);
4) реализуйте оператор `params`, который определяет имена переменных, значения которых должны быть заданы пользователем скрипта через параметры командной строки (например, в виде «`$foo=42`»); переменные в операторе перечисляются через запятую; если при запуске значение переменной не задано, то выполнение скрипта должно быть прервано с ошибкой;
5) реализуйте в математических выражениях не только сложение и вычитание, но и умножение и деление, которые должны выполняться с более высоким приоритетом, по сравнению со сложением и вычитанием; реализуйте возможность использования скобок для изменения приоритета выполнения операций;
6) реализуйте вычисления на базе типа double; реализуйте поддержку основных математических операций и функций (возведение в степень, sin, cos, tg, ctg, ln, exp);
7) реализуйте корректную обработку всех возможных ошибок, которые могут встречаться в скрипте (неверный оператор, отсутствие числа после знаков сложения и вычитания и т.п.); в случае ошибки пользователю должно быть выведено сообщение с описанием проблемы, после чего обработка скрипта прекращается, и приложение завершает работу.