Индивидуальное задание «Алгоритмы поиска»

Редакция 2015 г

Задачи

A) Частотный словарь. Дан текстовый файл, содержащий текст на русском или английском языке. Размер файла не ограничен. Получить частотный словарь этого текста, то есть перечень всех встречающихся в тексте слов (словоформ) с указанием, сколько раз встретилось в тексте данное слово.

a) В лексикографическом (алфавитном) порядке.
b) В порядке убывания частоты.

Словом *(словоформой) в рамках этой задачи называется последовательность русских (включая букву «ё») и латинских букв (строчных или заглавных), не содержащая внутри себя других символов.*

Программа в качестве исходных данных получает названия входного и выходного файлов. Читает входной файл(ы), результаты записывает в выходной текстовый файл. Рекомендуется для задания входного и выходного файлов использовать параметры командной строки. Запуск программы (название программы dictionary) из командной строки в этом случае может выглядеть так:

```
>dictionary <входной файл> <выходной файл>
```

Рекомендуется предусмотреть, чтобы программа могла обрабатывать больше одного входного файла за один раз. В этом случае запуск программы может выглядеть, например, так:

```
>dictionary *.txt dict.txt
```

В выходном файле в каждой строке приводятся данные об одном слове: вначале выводится слово, затем частота. Значения частоты должны быть выровнены. Можно считать, что длина слова не превышает 30 символов.

B) Измерение количества сравнений ключей. Таблица содержит в качестве ключей неотрицательные целые числа в диапазоне 0..32767. Данные отсутствуют.

Построить график зависимости среднего количества сравнений ключей при удачном и неудачном поиске от:

a) Количества элементов, занесенных в таблицу (m).
b) Степени заполнения таблицы (σ).

Опыты провести для m (или n) = 1000, 2000, 3000, 4000, 5000, 6000,
для σ = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 в случае таблицы с открытым перемешиванием и для σ = 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0 в случае таблицы с цепочками.
При изучении зависимости числа сравнений от числа элементов степень заполнения перемешанной таблицы должна оставаться постоянной: 0.7 для таблицы с открытым перемешиванием и 2.0 для таблицы с цепочками.

При изучении зависимости числа сравнений от степени заполнения перемешанной таблицы число элементов массива (n) должно оставаться постоянным (примерно 5000).

Порядок выполнения опытов может быть таким:

1) Заполнить таблицу m случайными числами (ключами) из диапазона 0..32767.
2) Выполнить поиск всех возможных ключей, то есть всех чисел от 0 до 32767, измеряя при каждом поиске количество сравнений ключа. Для измерения в программу поиска добавить счетчики числа сравнений ключа. Просуммировать количество сравнений выполненных при всех (m) удачных поисках и количество сравнений при всех неудачных (32767-m) поисках. Вычислить средние значения числа сравнений, разделив общее число сравнений при удачных поисках на количество удачных поисков, то есть на m, а общее число сравнений при всех неудачных поисках на (32767 - m). Среднее количество сравнений, разумеется, не обязано быть целым числом.
3) Построить графики (например, с помощью Excel).

C) Сравнение функций расстановки (перемешанные таблицы). Решить задачу **A)**, используя три различные функции расстановки, обеспечивающие малую длину поиска. Вывод словаря можно выполнять без упорядочения.

Измерить и сопоставить количество сравнений, выполняемых при обработке одного и того же файла при использовании разных функций расстановки. Выполнить не менее 5 серий измерений (то есть не меньше 15 опытов) для файлов существенного размера, содержащих тексты на естественном языке (русском, английском). Результаты свести в таблицу.

D) Статистика использования различных служебных (зарезервированных) слов языка Паскаль. Написать программу, которая позволяет подсчитать, сколько раз встретилось в программе(ах) не языке Паскаль каждое из служебных (зарезервированных) слов языка.
В качестве исходных данных программа получает названия входного и выходного файлов. Читает входной файл(ы), результаты записывает в выходной текстовый файл. Рекомендуется для задания входного и выходного файлов использовать параметры командной строки. Запуск программы (название программы statistics) из командной строки в этом случае может выглядеть так:

```
>statistics <входной файл> <выходной файл>
```

Рекомендуется предусмотреть, чтобы программа могла обрабатывать больше одного входного файла за один раз. В этом случае запуск программы может выглядеть, например, так:

```
>statistics *.pas stat.txt
```

В выходном файле в каждой строке приводятся данные об одном слове: вначале выводится слово, затем частота (абсолютная и в процентах). Значения частоты должны быть выровнены. Следует учесть, что заглавные и строчные буквы в языке Паскаль не различаются. Поэтому, например, слова **END** и **end** считаются одинаковыми. Перечень служебных слов оригинальной версии Паскаля приведен ниже.

AND, FILE, OF, TYPE, ARRAY, FOR, OR, UNTIL, BEGIN, FUNCTION, PACKED, VAR, CASE, GOTO, PROCEDURE, WHILE, CONST, IF, PROGRAM, WITH, DIV, IN, RECORD, DO, LABEL, REPEAT, DOWNTO, MOD, SET, ELSE, NIL, THEN, END, NOT, TO

E) Статистика использования слов из заданного словаря. Написать программу, которая позволяет подсчитать, сколько раз в текстовом файле(ах) встретилось каждое из слов, имеющихся в словаре.

Словом *(словоформой) в рамках этой задачи называется последовательность русских букв (строчных или заглавных, включая букву «ё»), латинских букв (строчных или заглавных) и цифр, не содержащая внутри себя других символов.*

В качестве исходных данных программа получает названия файла словаря, входного и выходного файлов. Читает файл словаря и входной файл(ы), результаты записывает в выходной текстовый файл. Рекомендуется для задания словаря, входного и выходного файлов использовать параметры командной строки. Запуск программы (название программы для примера dictstat) из командной строки в этом случае может выглядеть так:

```
>dictstat <файл словаря> <входной файл> <выходной файл>
```

Рекомендуется предусмотреть, чтобы программа могла обрабатывать больше одного входного файла за один раз. В этом случае запуск программы может выглядеть, например, так:

```
>statistics words.dic *.txt stat.out
```

В выходном файле в каждой строке приводятся данные об одном слове: вначале выводится слово, затем частота (абсолютная и в процентах). Значения частоты должны быть выровнены.

F) Количество различных цветов на фотографии. Написать программу, которая определяет количество различных цветов на фотографии.

В подавляющем большинстве случаев цвет каждой точки (пикселя) на цифровых фотографиях задается тремя значениями R, G и B, каждое из которых представляет собой целое 8-разрядное неотрицательное число (соответствует типу byte). Таким образом, можно задать 256x256x256 = 16 777 216 цветов. Но обычно количество разных цветов на фотографии существенно меньше количества пикселей. Например, снимок, размером 10 мегапикселей может содержать лишь около миллиона различных цветов. Число различных цветов определяет (формально) количество информации, содержащейся на фотографии.

Программа получает в качестве исходных данных название файла, содержащего данные фотографии, и выводит количество различных цветов.

Чтобы упростить задачу можно воспользоваться программой JPEG2RGB.EXE (см. Набор материалов к индивидуальному заданию), которая преобразует JPG-файл в двоичный (типизированный) файл простого формата, в котором последовательно записаны все тройки RGB, составляющие фотографию. Тип файла, создаваемый программой JPEG2RGB.EXE, можно определить так:

```
type
  tRGB = record
   R, G, B: byte;
  end;

  tRGBFile = file of tRGB;
```

Вызов программы может выглядеть, например, так:

```
>JPEG2RGB.exe DSC08478.jpg DSC08478.RGB
```

G) Определение простоты числа по таблице простых чисел. Если нужно определить, является ли не слишком большое натуральное число простым, можно воспользоваться таблицей простых чисел, включающей все простые числа от 2 до некоторого числа, больше или равного заданного. Если воспользоваться эффективным алгоритмом поиска, то время, затраченное на определение простоты числа, может оказаться меньше, чем непосредственная проверка, требующая, как правило, $O(\sqrt{n})$ действий.

Написать программы и выполнить измерения в следующем порядке:

1) Заполнить таблицу простых чисел всеми простыми числами от 2 до 32767. Всего в этом диапазоне имеется 3512 простых чисел.
2) Измерить время, затраченное на последовательное определение простоты чисел от 2 до 32767, с использованием обычного алгоритма с временной сложностью не хуже $O(\sqrt{n})$. Алгоритм «Решето Эратосфена», позволяющий определить все простые до заданного n, не использовать, поскольку основная задача состоит в определении простоты отдельного числа.
3) Измерить время, затраченное на последовательное определение простоты чисел от 2 до 32767, используя поиск в таблице.

Для определения времени работы программ можно использовать обычные часы с секундомером или, например, секундомер смартфона. Чтобы точность измерений была достаточной, необходимо измерять интервалы времени не короче 30 секунд. Если время работы программы оказывается меньше, можно поместить участок программы, время работы которого измеряется, в цикл, который выполняется столько раз, сколько требуется для того, чтобы время его многократного выполнения было не меньше 30 секунд.

Для измерения времени также можно использовать модуль `Measure.pas` или `MeasureWin.pas` (см. Набор материалов к индивидуальному заданию).

Вывод результатов (простых чисел) при измерении времени должен быть отключен.

H) Проверка орфографии. Определить все слова (словоформы) текста, содержащегося в текстовом файле, отсутствующие в перечне словоформ.

Список всех словоформ русского языка содержится в текстовых файлах AllRussanWordForm.txt (кодировка MS Windows) и AllRussanWordForm_DOS.txt (кодировка MS DOS) (см. Набор материалов к индивидуальному заданию). Всего файлы содержат 4 058 759 словоформ. Словоформы состоят из букв русского алфавита и могут включать дефис (символ чёрточка «-»). Наибольшая длина словоформы 28 символов. В перечне словоформ все слова записаны строчными буквами.

В качестве исходных данных программа получает название обрабатываемого файла(ов) и файла результатов. Читает входной файл, результаты записывает в выходной текстовый файл. Рекомендуется для задания входного и выходного файлов использовать параметры командной строки. Запуск программы (название программы spelling) из командной строки в этом случае может выглядеть так:

```
>spelling <входной файл> <выходной файл>
```

Рекомендуется предусмотреть, чтобы программа могла обрабатывать больше одного входного файла за один раз. В этом случае запуск программы может выглядеть, например, так:

```
>spelling *.txt res.dat
```

В выходном файле в каждой строке записывается одно слово. Слова могут повторяться.

I) **Обратный словарь.** Обратный словарь — словарь, в котором слова отсортированы в алфавитном порядке с учётом обратного чтения, т.е. не по начальным, а по конечным буквам. Для удобства поиска выравнивание списка слов в таком словаре идет не по левому, а по правому краю (для языков с письмом слева направо). Обратный словарь помогает изучать языки, имеющие суффиксальную агглютинацию, в которых конец слова несёт бóльшую грамматическую нагрузку, чем начало. Такие словари полезны при изучении словообразования (суффиксального, постфиксального), особенностей строения конца слов. В компьютерной лингвистике они используются как основа для составления и проверки словарей словоформ. Обратный словарь также может быть использован в качестве словаря рифм (Википедия).

Написать программу, которая строит обратный словарь заданного текста.

Словом *(словоформой) в рамках этой задачи называется последовательность русских (включая букву «ё») и латинских букв (строчных или заглавных), не содержащая внутри себя других символов.*

Программа в качестве исходных данных получает названия входного и выходного файлов. Читает входной файл(ы), результаты записывает в выходной текстовый файл. Рекомендуется для задания входного и выходного файлов использовать параметры командной строки. Запуск программы (название программы dictionary) из командной строки в этом случае может выглядеть так:

```
>revdict <входной файл> <выходной файл>
```

Рекомендуется предусмотреть, чтобы программа могла обрабатывать больше одного входного файла за один раз. В этом случае запуск программы может выглядеть, например, так:

```
>revdict *.txt dict.txt
```

В выходном файле в каждой строке помещается одно слово. Можно считать, что длина слова не превышает 30 символов.

Приложения

1. Программа JPEG2RGB.EXE

2. Файл *AllRussanWordForm.txt (кодировка MS Windows)*

3. Файл AllRussanWordForm_DOS.txt (кодировка MS DOS)

4. Текст всех известных произведений Н. А. Клюева (файлы KLUEV_DOS.txt и KLUEV_WIN.txt) .

Варианты заданий

	Линейный поиск в неупоря-доченной таблице без барьера	**Линейный поиск в неупоря-доченной таблице с барьером**	**Линейный поиск в упорядо-ченной таблице**	**Двоичный поиск**	**Поиск в двоичном дереве**	**Переме-шанная таблица с открытым перемеши-ванием**	**Переме-шанная таблица с цепочками**
1	Aa	Ab		D			
2		Aa			D, I		
3			E		Ab	D	
4	Ab				G		D
5			D		Aa		Ab
6		D	G			Aa	
7	D			G			Aa
8	E			E	E		
9		E				Bb	Ba
10						G, C	Bb
11	Ba			E			G
12		Ba			E	F	
13			Ba	H		E	
14				Ba	H		E
15					Ba	H	F
16		D				Ba	H
17					F	Ab	C
18				H		F	Ab
19							Ba, Bb, C
20						Ba, Bb, C	
21					Aa, Ab, Ba		
22	G			G		I	
23		G			I		I
24			H		H	H	
25	I	I					F

Программы могут быть написаны на любом диалекте языка Паскаль. Следует учитывать, что некоторые варианты не могут быть реализованы на Турбо-Паскале в силу ограниченности объема предоставляемой программе памяти. Некоторые задания могут предполагать использование для решения задачи также и алгоритмов сортировки.

При выполнении заданий (решении задач) процедура, реализующая алгоритм поиска, должна быть оформлена отдельно и снабжена необходимыми параметрами.