Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра интеллектуальных информационных технологий

**Отчет по лабораторной работе №6**

**по курсу «АОИС»**

|  |  |
| --- | --- |
| Выполнил студент группы 721701: | Козицкий А.С |
| Проверил: | Захаров В.В. |

**МИНСК**

**2018**

**Тема:** Моделирование хеш-таблиц;

**Цель работы:** освоение навыков построения и проверки таблиц хеширования.

**Индивидуальные задания:**

1. Разработать и проверить программу, обеспечивающую формирование хеш-таблицы, по ключевым словам и выполнение различных операций с этой таблицей – включение в таблицу новых строк, поиск информации в таблице по ключевым словам, удаление строк из таблицы.
2. Хеш-таблица должна быть разработана по образцу примера, рассмотренного выше.
3. Хеш-таблица должна быть тематической, т.е. все ключевые слова и данные должны быть из определенной отрасли науки, техники или деятельности людей. Перечень вариантов тематической направленности хеш-таблицы приведен в таблице 2.  
   *Таблица 2.*

|  |  |  |  |
| --- | --- | --- | --- |
| № варианта | Тематическая направленность (ТХ) | Ключевые слова (ID) | Данные () |
| 1 | Математика | Названия, термины, понятия из соответствующей отрасли науки, техники, деятельности. | Определение, что это или из какого раздела данной науки, техники и т.д. это ключевое слово. |
| 2 | Физика |
| 3 | География |
| 4 | Биология |
| 5 | Грамматика |
| 6 | Литература |
| 7 | Медицина |
| 8 | Транспорт |
| 9 | Спорт |
| 0 | Армия, вооружение |

1. Каждый студент выполняет вариант хеш-таблицы, номер которого совпадает с последней цифрой номера записи его фамилии в списке группы.  
    *Примечание:* если у студента доставшийся ему вариант хеш-таблицы вызывает затруднее, например, из-за недостатка знаний в данной тематике, он может предложить и по согласованию с преподавателем выполнить свой другой вариант, отсутствующий в таблице 2 с соблюдением всех требований к построению хеш-таблиц, приведенных в данном методическом материале.

При выполнение любого варианта ЛР №6 должны быть выполнены слудующие требовония:  
 - размер таблицы не менее 20 строк;  
 - количество записанных в таблице строк – не менее 10-ти;  
 - количество коллизий не менее 2-х;  
 - количество строк, объединенных в цепочки резервных ячеек, сформированных из-за коллизий – не менее 3-х;  
 - размер и структура каждой строки хеш-таблицы должны быть одинаковыми и постоянными.  
 *Примечание:* если размер поля в хеш-таблице не позволяет записать все данные по конкретному ключевому слову, может быть использована отдельная область памяти для записи этих данных, при этом в поле записывается указатель, где в памяти находятся данные, и флажок «L» (флажок связи) устанавливается в «1».

**Требовония к программе:**

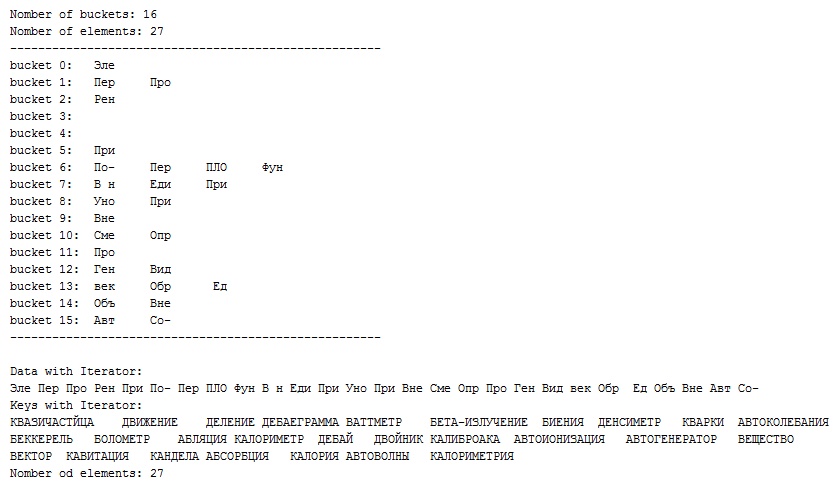
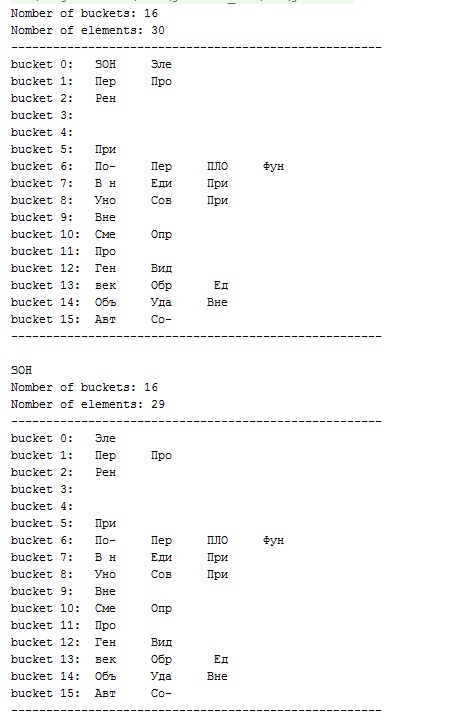
1. Разработаная программа должна уметь выполнять следующие функции:  
    - определение числовых значений ключевых слов (V);  
    - формирование хеш-адресов [h(v)];  
    - формирование самой хеш-таблицы, в соответствии со структурой, приведенной на рисунке 1;  
    - обработку коллизии (т.е. обеспечение нахождения резервных ячеек и запись в них);  
    - выборку из хеш-таблицы необходимой информации по соответствующему ключевому слову;  
    - занесение в хеш-таблицу новых новых записей по новым ключевым словам;  
    - удаление из хеш-таблицы отдельных записей;  
    - контроль за потыткой записи в хеш-таблицу информации по ключевому слову, уже имеющемуся в таблице;  
    - расчет коэффициентов заполнения хеш-таблицы.
2. Программа должна обеспечивать индикацию (вывод на экран) содержимого всех полей и указателей всех строк хеш-таблицы, вычисленных значений V(K) и h(V) для каждой записи, коэффициента заполнения хеш-таблицы.

**Вариант 12**

Реализовать хеш-таблицу, заполнить её значениями из области физики.

Порядок вызова методов: Создание кастомной хеш-таблицы, заполнение её значениями, вывод количество вложенных списков (баскетов), вывод кол-ва элементов в таблице, общий вывод(все значения, в соответсвии с их баскетом), поиск эелемента по ключу, вывод кол-ва элементов и баскетов, общий вывод, удаление 2-ух существующих элементов из таблицы, общий вывод, вывод с помощью итератора сначала всех значений, а потом всех ключей, вывод кол-ва элементов в таблице.

**Результат работы:**

Ниже приведён скриншот с результатами работы программы.

**Вывод**

Хэш-таблица (далее просто таблица) на мой взгляд на равнее с сбалансированным и деревьями является одним из лучших решений хранения данных. Деревья уступают таблице в скорости поиска для них это Log2(N), для таблицы же скорость поиска константная. Однако деревья в свою очередь представляют данные в упорядоченном виде, в отличие от таблице.

В моей реализации хеш-таблица представлена массивом, содержащим в каждой своей ячейке двусвязный список. Прокомментирую выше мной указанною скорость поиска элемента в таблице. Константная скорость достигается только при грамотной реализации хеш-кода, в противном же случае если оставить корректный (с точки зрения компилятора хеш-код (return 0)), вся таблица будет сведена к связном списку( одна огромная коллизия), что влечёт с собой линейную скорость поиска. Также в моей реализации не предусмотрено перестроение таблицы (добавление новых баскетов), так например в реализации HashMap в jdk, предусмотрена возможность изменения кол-ва баскетов, насколько я знаю оптимальное соотношение заполненных баскетов к пустым 75% | 25%, после чего неплохо бы перестроить таблицу с добавлением новых басветов (например увеличение в 1,5 раза как в HashMap).

Теперь касаемо коллизий и метода hashCode(). Количество коллизий напрямую зависит от его реализации. Данный метод является легковесным подспорьем, позволяющим быстро определить различны ли объекты. Важно отметить что он не дает точной гарантии касаемо сходства объектов, а именно:

1)hash1 != hash2 🡺 объекты точно различны

2)hash1 == hash2 🡺 мы не смогли быстро сравнить объекты, они возможно равны, возможно не равны 🡺 воспользуйся методом equals().

Метод equals() в свою очередь даёт однозначно понять равны объекты или нет и применяется он только в случае коллизии.

Также хотелось бы отметить что коллизия не является чем-то “ПЛОХИМ” (как мне показалось - это общее заблуждение), напротив при полном отсутствии коллизий мы получаем массив списков единичной величины, что опять же ведёт к линейно скорости поиска. Таким образ ещё раз хочу сакцентировать внимание на разумном балансе (цифры указаны выше).

Также работу выполнял на Java, аргументы всё те же что и в предыдущих работах + хочу отметить то что стандарты коллекций jdk, а именно хеш-таблиц реализованы на чистой джаве без использования нативных методов.