|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)**

**ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2**

**по дисциплине «Структуры и алгоритмы обработки данных»**

|  |  |  |  |
| --- | --- | --- | --- |
| Отчет представлен к  рассмотрению:  Студент группы ИКБО-36-22 | «19» сентября 2023 г. | (подпись) | Утенков Ю.Ю. |
|  |  |  |  |
| Преподаватель | «19» сентября 2023 г. | (подпись) | Красников С.А. |

Москва, 2023 г.

Оглавление

[Цель работы. 3](#_Toc146144973)

[Ход работы. 3](#_Toc146144974)

[Формулировка задачи. 3](#_Toc146144975)

[Математическая модель решения (описание алгоритма). 3](#_Toc146144976)

[Код программы с комментариями. 7](#_Toc146144977)

[Результаты тестирования. 10](#_Toc146144978)

[Выводы. 12](#_Toc146144979)

[Список информационных источников. 13](#_Toc146144980)

# Цель работы.

Освоить приёмы хеширования и эффективного поиска элементов множества. Освоить и понять метод цепного хеширования (по номеру задания).

# Ход работы.

## Формулировка задачи.

Необходимо разработать приложение, которое использует хеш-таблицу (пары «ключ – хеш») для организации прямого доступа к элементам динамического множества полезных данных. Множество реализовать на массиве, структура элементов (перечень полей) которого приведена в индивидуальном варианте (27 вариант). Приложение должно содержать класс с базовыми операциями: вставки, удаления, поиска по ключу, вывода. Включить в класс массив полезных данных и хеш-таблицу. Хеш-фунцию необходимо подобрать самостоятельно, используя правила выбора функции. Реализовать расширение размера таблицы и рехеширование, когда это требуется, в соответствии с типом разрешения коллизий. Предусмотреть автоматическое заполнение таблицы 5-7 записями. Реализовать текстовый командный интерфейс пользователя для возможности вызова методов в любой произвольной последовательности, сопроводите вывод достаточными для понимания происходящего сторонним пользователем подсказками. Провести полное тестирование программы (все базовые операции, изменение размера и рехеширование), тест-примеры определите самостоятельно. Результаты тестирования включить в отчет по выполненной работе.

## Математическая модель решения (описание алгоритма).

**Листинг 3.**

1. Алгоритм принимает входные параметры: ключ (key), номер группы (group\_number), и полное имя (fullName), которые требуется вставить в хеш-таблицу.
2. Вычисляется хеш-код (хеш) для заданного ключа. Этот хеш-код определяет индекс в массиве хеш-таблицы, в котором будет храниться элемент (группа элементов с одинаковым хеш-кодом будет представлена в виде односвязанного списка).
3. Проверяется, есть ли уже элемент с таким же ключом (key) в односвязанном списке, соответствующем данному хешу. Если элемент с таким ключом уже существует в списке, выводится сообщение о том, что вставка не удалась из-за одинаковых ключей, и возвращается значение false.
4. Если элемент с таким ключом не обнаружен, то новый элемент (с ключом, номером группы и полным именем) добавляется в начало односвязанного списка, соответствующего вычисленному хешу.
5. В случае успешной вставки элемента возвращается значение true для обозначения успешного выполнения операции.

Время выполнения данного алгоритма в среднем случае будет оценено как O(N+M), где N - это размер массива хеш-таблицы, а M - это максимальное количество элементов, которые могут быть хранимы в одной ячейке хеш-таблицы (длина односвязанного списка), так как он выполняет поиск по односвязанному списку (среднее время выполнения поиска в списке) и не требует перебора всех элементов хеш-таблицы в поиске коллизий.

**Листинг 4.**

1. Метод deleteItem принимает входной параметр key, который представляет собой ключ элемента, который требуется удалить из хеш-таблицы.
2. Сначала проверяется, есть ли элемент с заданным ключом в хеш-таблице, вызывая метод getListItem(key). Если элемент не найден (метод возвращает null), выводится сообщение о том, что false, что указывает на неуспешное удаление.
3. Если элемент с заданным ключом найден в хеш-таблице, то происходит удаление:

a. Вычисляется индекс ячейки хеш-таблицы, в которой находится элемент с заданным ключом, с помощью метода getIndex(key).

b. Создается временная переменная target, которая хранит ссылку на удаляемый элемент (это начальное значение элемента, который находится в данной ячейке хеш-таблицы).

c. Создается временная переменная temp, которая будет использоваться для замены связей внутри списка после удаления элемента.

d. Получается ссылка на первый элемент списка, находящегося в данной ячейке хеш-таблицы, и она сохраняется в переменной current.

e. Запускается цикл while, который перебирает элементы списка в данной ячейке хеш-таблицы.

f. Внутри цикла проверяется, является ли текущий элемент (current) удаляемым элементом (target). Если да, то элемент current удаляется из списка путем переключения ссылок. Если удаляемый элемент находится в начале списка (голове), то вместо него ставится следующий элемент в списке. Если удаляемый элемент находится в середине или в конце списка, то происходит замена связей между элементами.

g. После завершения цикла while весь процесс удаления завершается, и возвращается true, что указывает на успешное удаление элемента.

Этот алгоритм позволяет эффективно удалять элементы из хеш-таблицы, используя метод цепного хеширования для разрешения коллизий и сохраняя целостность списков внутри ячеек хеш-таблицы.

**Листинг 5.**

Метод print() перебирает все элементы в хеш-таблице и выводит на экран информацию о каждом элементе, включая номер группы учащегося. После вывода всех элементов в таблице выводится пустая строка для разделения вывода.

**Листинг 6.**

Метод resize() автоматически увеличивает размер исходной хеш-таблицы в два раза. Он выполняет следующие действия:

1. Выводит сообщение о начале процесса рехеширования таблицы.
2. Сохраняет текущий размер хеш-таблицы в переменной tempSizeTable.
3. Увеличивает размер хеш-таблицы в два раза (sizeTable \*= 2).
4. Создает новый пустой массив listes с новым размером.
5. Копирует элементы из старого массива listNodes в новый массив listes.
6. Обновляет старый массив listNodes с новым размером.
7. Затем метод выполняет рехеширование данных из старого массива listes в новый массив хеш-таблицы listNodes, используя метод insert() для каждого элемента списка.
8. После этого метод завершается, и размер хеш-таблицы увеличивается в два раза, а данные перераспределяются в соответствии с новой размерностью таблицы.

**Листинг 7.**

Метод getListItem() используется для поиска элемента в хеш-таблице по заданному ключу. Он выполняет следующие действия:

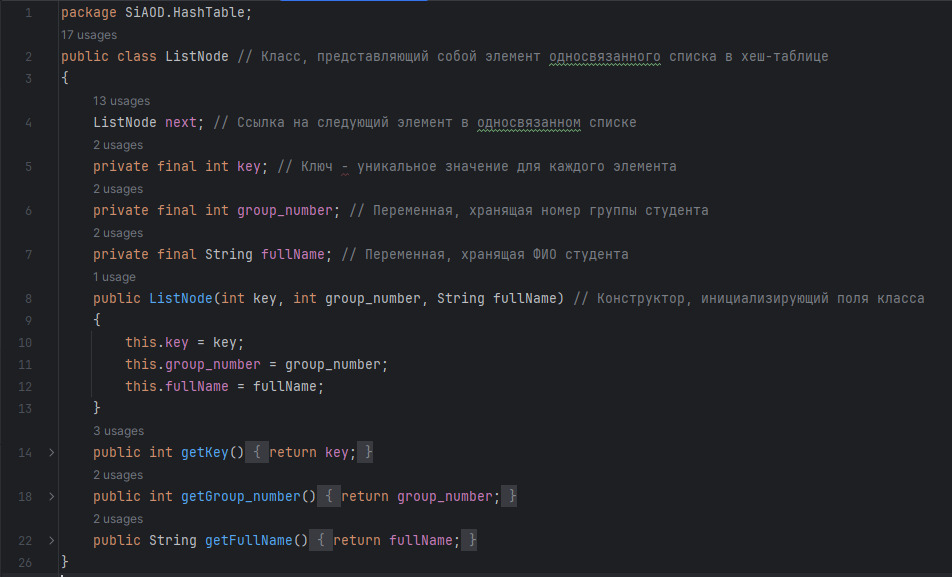
1. Получает индекс элемента в хеш-таблице для заданного ключа с помощью метода getIndex(key).
2. Получает текущий элемент (голову списка) с этим индексом из хеш-таблицы.
3. Проверяет, если текущий элемент равен null, это означает, что элемент с заданным ключом ещё не был добавлен в хеш-таблицу, и выводит сообщение о том, что элемент не найден, возвращая null.
4. Если текущий элемент не равен null, метод начинает перебирать элементы внутри односвязанного списка, начиная с текущего элемента.
5. Внутри цикла, если ключ текущего элемента совпадает с заданным ключом, метод возвращает текущий элемент, что указывает на успешное нахождение элемента в хеш-таблице.
6. Если текущий элемент не соответствует заданному ключу, метод двигается к следующему элементу списка (current.next).
7. Если после перебора всех элементов в списке не было найдено элемента с заданным ключом, метод выводит сообщение о том, что элемент не найден, и возвращает null.

Таким образом, метод getListItem() позволяет искать элемент в хеш-таблице по ключу и возвращать найденный элемент или null, если элемент не найден.

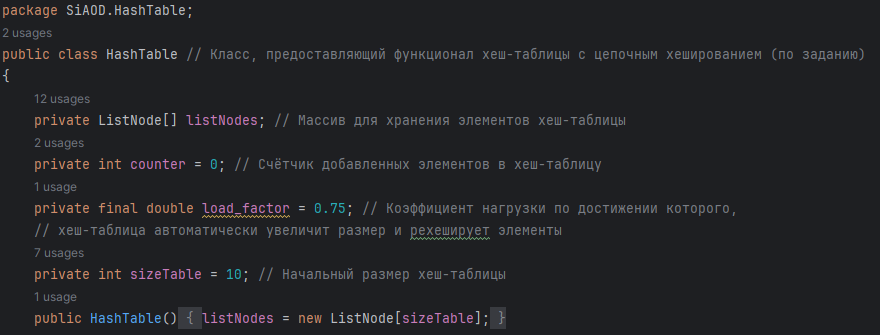
**Листинг 8.**

Метод getIndex(int key) генерирует индекс элемента в хеш-таблице на основе исходного ключа, путём вычисления остатка от деления значения ключа элемента на размер хеш-таблицы.

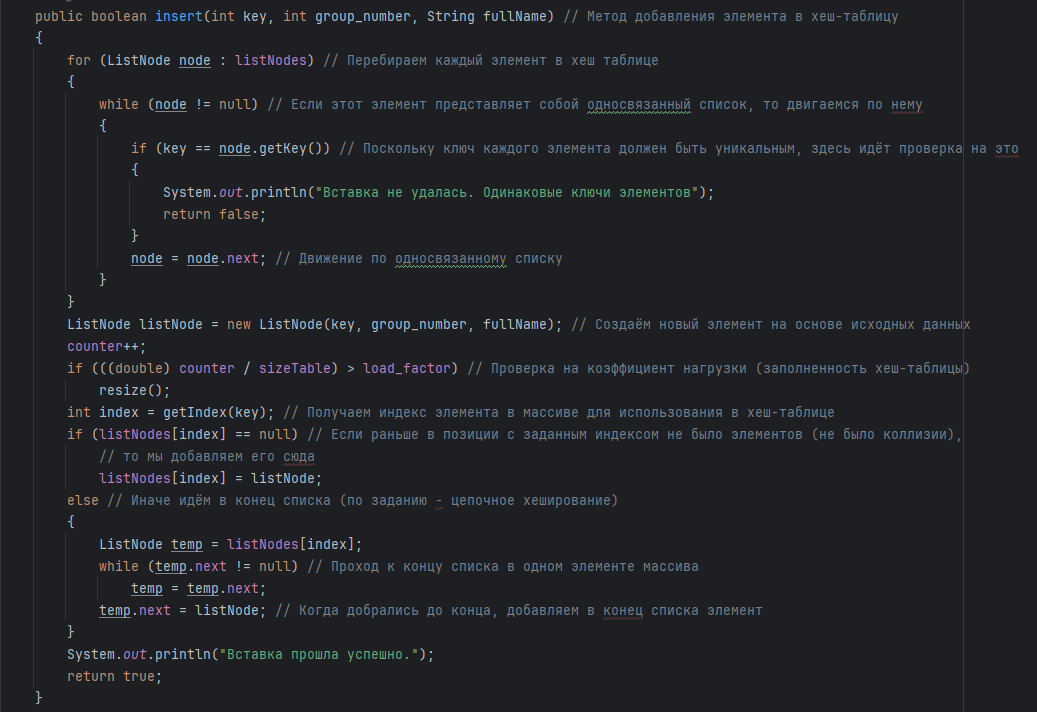
## Код программы с комментариями.



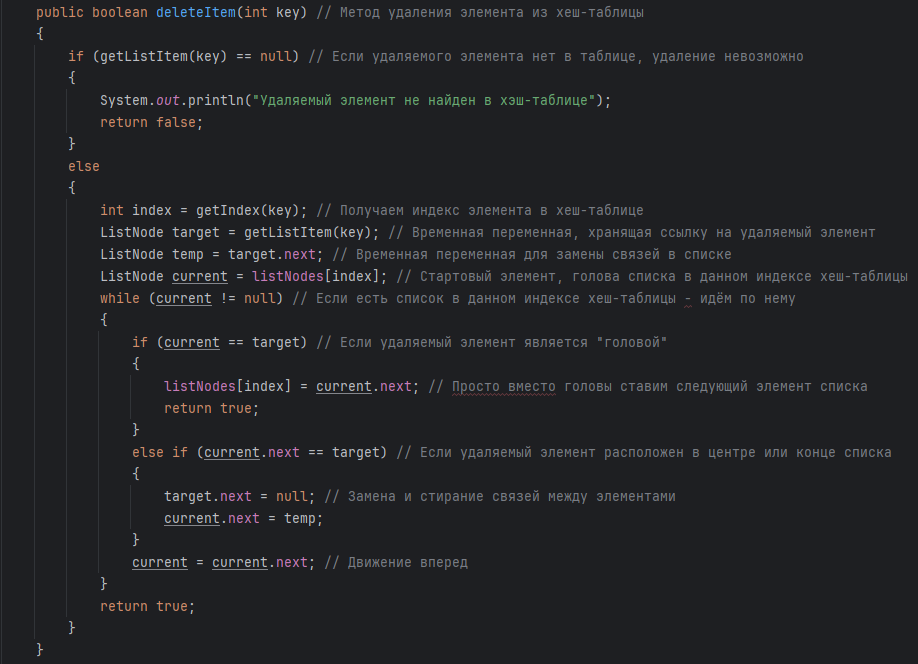
*Листинг 1. Структура элемента в хеш-таблице.*



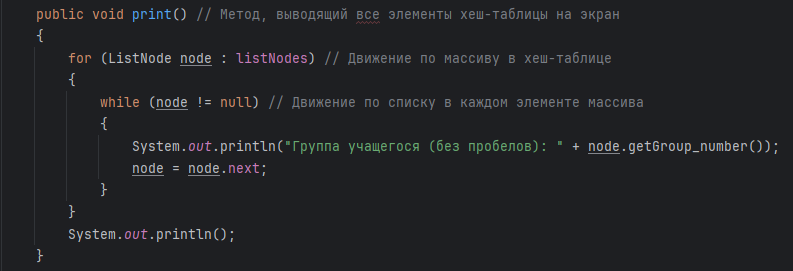
*Листинг 2. Поля и конструктор хеш-табл.*

**

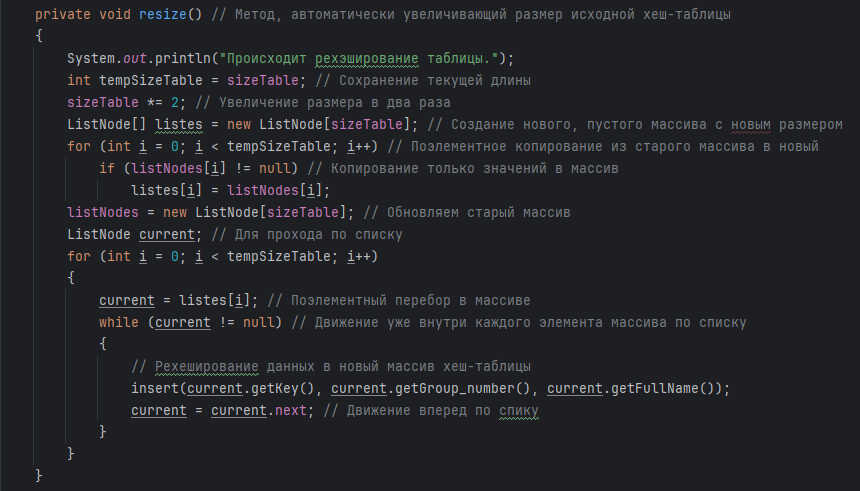
*Листинг 3. Метод, позволяющий добавлять элементы в хеш-таблицу с соблюдением условий.*



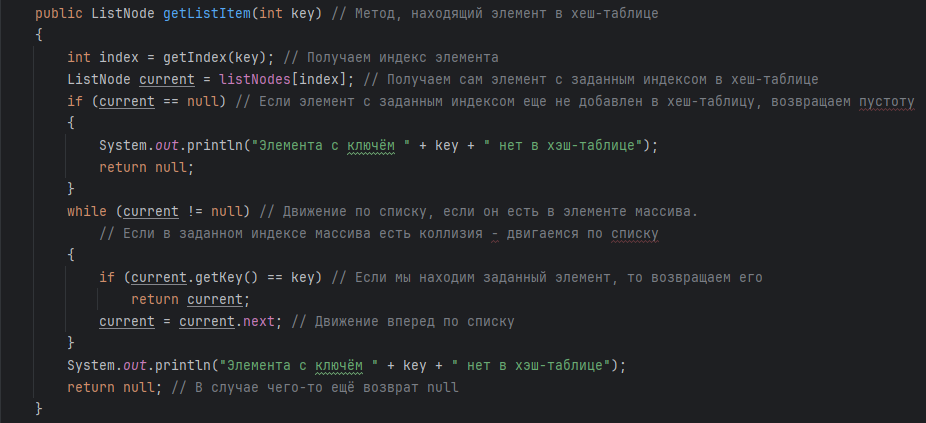
*Листинг 4. Метод, позволяющий удалить элемент из хеш-таблицы по ключу.*

**

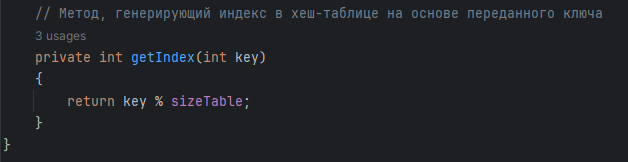
*Листинг 5. Метод, позволяющий вывести все элементы хеш-таблицы на экран.*



*Листинг 6. Метод, позволяющий увеличить размер хеш-таблицы в два раза с рехешированием элементов.*

**

*Листинг 7. Метод, позволяющий найти элемент в хеш-таблице, при условии что он существует.*

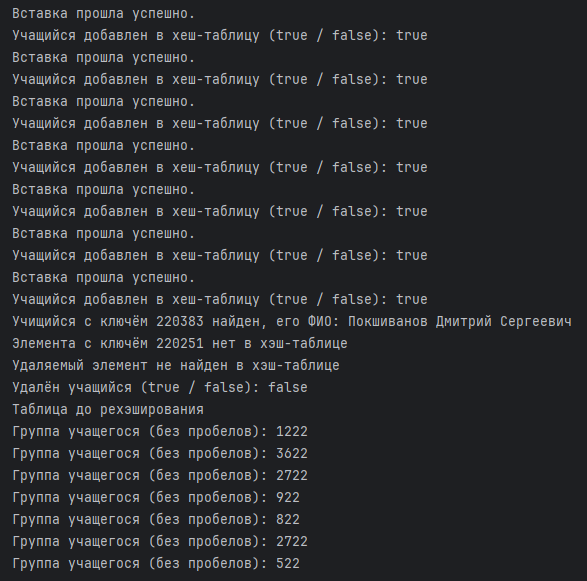


*Листинг 8. Метод, позволяющий сгенерировать индекс для элемента в хеш-таблице.*

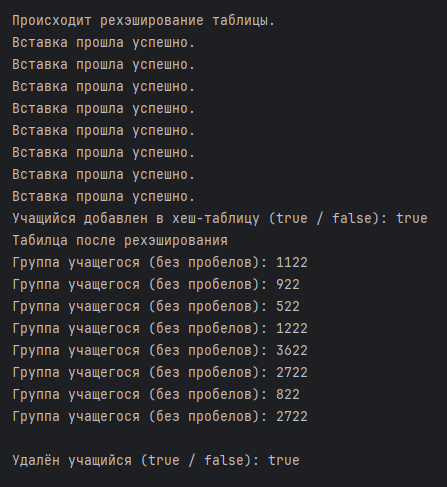
## Результаты тестирования.

****

*Листинг 9. Исходные данные для тестирования.*

**

*Рис.1. Пример работы программы.*

**

*Рис.2. Пример работы программы (продолжение).*

В данной части тестирования создаётся экземпляр класса HashTable и добавляются 8 узлов в таблицу, после добавления восьмого, запускается рехеширование так как коэффициент нагрузки превышен. Как видно из рисунка 1, 2, в процессе тестирования, функции добавления, отображения, рехеширования и удвоения работают без ошибок.

# Выводы.

В ходе выполнения данной практической работы я изучил и рассмотрелл принципы работы хеш-таблиц и их применение в различных областях информатики и программирования. Хеш-таблицы представляют собой мощный инструмент в мире информатики и программирования, который может значительно улучшить производительность и эффективность различных приложений. Понимание их принципов работы и методов реализации позволяет разработчикам более успешно решать задачи, связанные с управлением данными и поиском информации.

# Список информационных источников.

1. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих, 2017. – С. 100-126
2. Кораблин Ю.П. Структуры и алгоритмы обработки данных. Часть 1 [Электронный ресурс]: учебно методическое пособие / Ю.П.Кораблин, В.П.Сыромятников, Л.А. Скворцова – М.: РТУ МИРЭА, 2020. – 1 электрон. опт. диск (CD-ROM)
3. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. URL:https://online-edu.mirea.ru/course/view.php?id=4020 (дата обращения 20.09.2023).