

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №5
по дисциплине «Алгоритмы и структуры данных»
Тема: БДП и Хеш-таблицы

Студент гр. 9381

Камакин Д.В.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2020

Цель работы

Познакомиться с хеш-таблицей методом цепочек, представить реализацию на языке программирования C++.

Постановка задачи

Вариант 23.

Хеш-таблица: с цепочками;

действие: $1+2a$

1) По заданной последовательности элементов Elem построить структуру данных определённого типа – БДП или хеш-таблицу;

2) Выполнить одно из следующих действий:

а) Для построенной структуры данных проверить, входит ли в неё элемент e типа Elem, и если входит, то в скольких экземплярах. Добавить элемент e в структуру данных. Предусмотреть возможность повторного выполнения с другим элементом.

Основные теоретические положения.

Хеш-таблица (hash table) — это специальная структура данных для хранения пар ключей и их значений. По сути это ассоциативный массив, в котором ключ представлен в виде хеш-функции.

Пожалуй, главное свойство hash-таблиц — все три операции вставка, поиск и удаление в среднем выполняются за время $O(1)$, среднее время поиска по ней также равно $O(1)$ и $O(n)$ в худшем случае.

Описание алгоритма

Был реализован класс HashTable, таблица в котором представляет собой вектор списков (цепочек), хеш элемента высчитывается путём взятия остатка от его длины на размер таблицы. Для поиска элемента высчитываем хеш, берём

нужный список и делаем по нему обход. Для добавления высчитываем хеш, после чего вставляем в нужный список элемент.

Описание классов:

HashTable — шаблонный класс хеш-таблицы. Приватное поле `table_` содержит саму таблицу, `size_` - её размер.

Методы класса:

1. `void add(T value)` — добавление элемента в хеш-таблицу. При помощи вызова функции `hash()` высчитываем хеш, после чего добавляем по нему наш элемент.

2. `int count(T value)` — поиск элемента в хеш-таблице. Получаем хеш функцией `hash()`, после чего считаем и возвращаем количество элементов по данному хешу. Используется цикл `foreach()`.

3. `int hash(T value)` — возвращает хеш элемента. Считается на основе длины элемента и размера таблицы (остаток от деления первого на второе).

4. `friend std::ostream& operator<<(std::ostream &out, const HashTable<T> &table)` — оператор вывода в поток. Возвращает поток, в который требуется вывести таблицу.

Описание функций:

1. `void outputHelp(std::ostream &output)` — выводит в `output` справку по использованию программы.

2. `int getAction(std::istream &input)` — считывает из `input` выбранное пользователем действие и возвращает его.

3. `std::vector<std::string> split(const std::string& str, char delim)` — разбиение строки `str` по разделителю `delim`, возвращает вектор строк.

4. `void readString(std::istream &input, std::string &string)` — считывает строку из `input` в `string`, разделитель — символ переноса строки.

Тестирование

Входные данные	Исходные данные
2 add test one two three	Your string: add test one two three The table is: Table[0] = Table[1] = Table[2] = Table[3] = add->one->two-> Table[4] = test-> Table[5] = three-> Table[6] = Table[7] = Table[8] = Table[9] =
2 this is a simple test don't mind if i leave it here ok	Your string: this is a simple test don't mind if i leave it here ok The table is: Table[0] = Table[1] = a->i-> Table[2] = is->if->it->ok-> Table[3] = add->one->two-> Table[4] = test->this->test->mind->here-> Table[5] = three->don't->leave-> Table[6] = simple-> Table[7] = Table[8] = Table[9] =
1 this is a simple test don't mind if i leave it here ok	Your string: this is a simple test don't mind if i leave it here ok this contains 1 times in the table is contains 1 times in the table a contains 1 times in the table simple contains 1 times in the table test contains 2 times in the table don't contains 1 times in the table mind contains 1 times in the table if contains 1 times in the table i contains 1 times in the table leave contains 1 times in the table it contains 1 times in the table

	<p>here contains 1 times in the table</p> <p>ok contains 1 times in the table</p>
<p>2</p> <p>adding as much as possible wow this is so funny i love writiing reports</p>	<p>Your string: adding as much as possible wow this is so funny i love writiing reports</p> <p>The table is:</p> <p>Table[0] =</p> <p>Table[1] = a->i->i-></p> <p>Table[2] = is->if->it->ok->as->as->is->so-></p> <p>Table[3] = add->one->two->wow-></p> <p>Table[4] = test->this->test->mind->here->much->this->love-></p> <p>Table[5] = three->don't->leave->funny-></p> <p>Table[6] = simple->adding-></p> <p>Table[7] = reports-></p> <p>Table[8] = possible->writiing-></p> <p>Table[9] =</p>
<p>1</p> <p>elephant</p>	<p>Your string: elephant</p> <p>elephant contains 0 times in the table</p> <p>The table is:</p> <p>Table[0] =</p> <p>Table[1] = a->i->i-></p> <p>Table[2] = is->if->it->ok->as->as->is->so-></p> <p>Table[3] = add->one->two->wow-></p> <p>Table[4] = test->this->test->mind->here->much->this->love-></p> <p>Table[5] = three->don't->leave->funny-></p> <p>Table[6] = simple->adding-></p> <p>Table[7] = reports-></p> <p>Table[8] = possible->writiing-></p> <p>Table[9] =</p>

Выводы

Была изучена и реализована на языке программирования C++ хеш-таблица методом цепочек.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <vector>
#include <string>
#include <list>
#include <fstream>

template <typename T>
class HashTable {
    std::vector < std::list<T> > table_;
    int size_;

public:
    explicit HashTable(int size) : size_(size) {
        table_.resize(size_);
    }

    void add(T value) {
        table_[hash(value)].push_back(value);
    }

    int count(T value) {
        auto key = hash(value), count = 0;

        for (const auto &elem : table_[key])
            if (value == elem)
                count++;

        return count;
    }

    friend std::ostream& operator<<(std::ostream &out, const
    HashTable<T> &table) {
        for (auto i = 0; i < table.size_; i++) {
            out << "Table[" << i << "] = ";
            for (const auto &elem : table.table_[i])
```

```

        out << elem << "->";
        out << "\n";
    }

    return out;
}

private:
    int hash(T value) {
        return value.size() % size_;
    }

};

void outputHelp(std::ostream &output) {
    output << "Choose one of the following actions: " << '\n';
    output << "1. Find the elements" << '\n';
    output << "2. Add the elements" << '\n';
    output << "3. Open a file" << '\n';
    output << "4. Close the file and read from std::cin" << '\n';
    output << "5. Exit" << '\n';
    output << "Your action: ";
}

int getAction(std::istream &input) {
    int action;
    outputHelp(std::cout);
    input >> action;
    input.ignore();
    return action;
}

std::vector<std::string> split(const std::string& str, char delim)
{
    std::vector<std::string> strings;
    size_t start;
    size_t end = 0;

```

```

        while ((start = str.find_first_not_of(delim, end)) !=
std::string::npos) {
            end = str.find(delim, start);
            strings.push_back(str.substr(start, end - start));
        }

        return strings;
    }

void readString(std::istream &stream, std::string &string) {
    std::cout << "Input: ";
    getline(stream, string, '\n');
    std::cout << "Your string: " << string << '\n';
}

int main() {
    HashTable<std::string> table(10);
    int action;

    std::ifstream file;
    std::string filePath;
    std::string string;
    std::vector<std::string> elements;
    std::istream *input = &std::cin;

    while ((action = getAction(std::cin)) != 5) {

        switch (action) {
            case 1:
                readString(*input, string);
                elements = split(string, ' ');

                for (auto &i : elements)
                    std::cout << i << " contains " <<
table.count(i) << " times in the table" << '\n';

                break;
            case 2:
                readString(*input, string);

```



```

        elements = split(string, ' ');

        for (auto &i : elements)
            table.add(i);

        break;
    case 3:
        std::cout << "Path to the file: ";
        std::cin >> filePath;
        file.open(filePath);

        if (!file.is_open()) {
            std::cout << "Couldn't open the file, please
try again" << '\n';
            continue;
        }

        input = &file;
        break;
    case 4:
        if (file.is_open())
            file.close();
        input = &std::cin;
        break;
    case 5:
    default:
        std::cout << "Exiting the program" << '\n';
        return 0;
    }

    std::cout << '\n' << "The table is: " << '\n';
    std::cout << table << '\n';
}

return 0;
}

```