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

ОТЧЕТ

по лабораторная работа №5

по дисциплине «Алгоритмы и структуры данных»

ТЕМА: ХЭШ-ТАБЛИЦА: С ЦЕПОЧКАМИ

Студент гр. 9303 _____ Павлов Д.Р.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург

2020

Цель работы.

Реализовать хэш-таблицу и решить коллизию методом цепочек.

Задание.

Вариант 24.

Хеш-таблица: с цепочками; действие: 1+2б

1) По заданной последовательности элементов *Elem* построить структуру данных определённого типа – БДП или хеш-таблицу;
2) б) Для построенной структуры данных проверить, входит ли в неё элемент *e* типа *Elem*, и если входит, то удалить элемент *e* из структуры данных (первое обнаруженное вхождение). Предусмотреть возможность повторного выполнения с другим элементом.

Основные теоретические положения.

Хеш-таблица — это структура данных, реализующая интерфейс ассоциативного массива, а именно, она позволяет хранить пары (ключ, значение) и выполнять три операции: операцию добавления новой пары, операцию поиска и операцию удаления пары по ключу.

Существуют два основных варианта хеш-таблиц: с цепочками и открытой адресацией. Хеш-таблица содержит некоторый массив , элементы которого есть пары (хеш-таблица с открытой адресацией) или списки пар (хеш-таблица с цепочками).

Выполнение работы.

В ходе выполнения работы был реализован класс `HashTable`, который содержит поля `list<int> *table` и `total_elements`, также существует метод `getHash` который получает ключ и возвращает хэш.

Затем были созданы публичные методы `explicit HashTable(int n)`, конструктор, который задает размерность хэш-таблицы. Также был написан метод `insertElement`, который получает хэш ключа и заполняет хэш-таблицу. Еще был написан метод `removeElement`, который получает хэш ключа и удаляет из таблицы элемент. И были написаны два метода для распечатывания всей хэш-таблицы и по ключу.

Исходную последовательность элементов программа считает из файла.

Разработанный программный код см. в приложении А.

Результаты тестирование см. в приложении В.

Вывод.

В ходе выполнения лабораторной работы была изучена структура данных типа Хэш-таблица с методом цепочек. Также была написана программа, в которой реализован класс данной структуры и методы для работы с ней.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.cpp

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

using namespace std;

class HashTable{
private:
    list<int> *table;
    int total_elements;

    [[nodiscard]] int getHash(int key) const{
        return key % total_elements;
    }

public:
    explicit HashTable(int n){
        total_elements = n;
        table = new list<int>[total_elements];
    }

    void insertElement(int key){
        table[getHash(key)].push_back(key);
    }

    void removeElement(int key, ofstream &output){
        int x = getHash(key);

        list<int>::iterator i;
        for (i = table[x].begin(); i != table[x].end(); i++) {
            // Check if the iterator points to the required item:
            if (*i == key)
                break;
        }

        if (i != table[x].end())
            table[x].erase(i);
        else{
            output<<"[WARNING] Key not found!\n";
            cout << "[WARNING] Key not found!\n";
        }

    }

    void printAll(ofstream &output){
        for(int i = 0; i < total_elements; i++){
            output << "Index " << i << ": ";
            cout << "Index " << i << ": ";
            for(int j : table[i]) {
                output << j << " => ";
                cout << j << " => ";
```

```
            }

            output << endl;
            cout << endl;
        }
    }

    void printSearch(int key, ofstream &output){
        for(int i = 0; i < total_elements; i++){
            if (i == key){
                for(int j : table[i]) {
                    output << j << " => ";
                    cout << j << " => ";
                }
            }

        }
    }
};


void checkStr(std::string& str) {
    for (int i = 0; i < str.size(); i++) {
        if (!isdigit(str[i])&&str[i]!=' '&&str[i] != '-') {
            str.erase(i, 1);
            i -= 1;
        }
    }
}

int main() {
    vector<int> nums;
    HashTable ht(5);

    ifstream input("./tests/input.txt");
    ofstream output("./tests/result.txt");
    std::string numbers;
    int value;
    input>>noskipws;
    if(!input){
        output<<"Can't open this file!";
        return 0;
    }
    std::getline(input, numbers);
    checkStr(numbers);
    std::stringstream ss(numbers);
    while (ss >> value) {
        nums.push_back(value);
        if (ss.peek() == ' ') {
            ss.ignore();
        }
    }
    for(int i : nums) {
        ht.insertElement(i);
    }
    while(true){
        output << "\nMENU:\n"
                  "1: Print HashTable\n"
                  "2: Add element\n"
                  "3: Delete element\n"
                  "4: Print elements by key\n"
                  "Another key: Quit\n";
        cout << "\nMENU:\n"
                  "1: Print HashTable\n"
```

```
                "2: Add element\n"
                "3: Delete element\n"
                "4: Print elements by key\n"
                "Another key: Quit\n";
        int choose;
        std::cin >> choose;
        output << choose << endl;
        switch (choose) {
            case 1:
                output << "[INFO]Hash Table" << endl;
                cout << "[INFO]Hash Table" << endl;
                ht.printAll((ofstream &) output);
                break;
            case 2:
                int addValue;
                output << "[QUESTION]Which value you want to insert?\n";
                cout << "[QUESTION]Which value you want to insert?\n";
                std::cin >> addValue;
                output << addValue << endl;
                cout << addValue << endl;
                ht.insertElement(addValue);
                output << endl << "[INFO]After adding " << addValue << '\n';
                cout << endl << "[INFO]After adding " << addValue << '\n';
                ht.printAll((ofstream &) output);
                break;
            case 3:
                int rmValue;
                output << "[QUESTION]Which value you want to delete?\n";
                cout << "[QUESTION]Which value you want to delete?\n";
                std::cin >> rmValue;
                output << rmValue << endl;
                cout << rmValue << endl;
                ht.removeElement(rmValue, (ofstream &) output);
                output << endl << "[INFO]After removal " << rmValue << '\n';
                cout << endl << "[INFO]After removal " << rmValue << '\n';
                ht.printAll((ofstream &) output);
                break;
            case 4:
                int lsValue;
                output << "[QUESTION]Which key you want to look?\n";
                cout << "[QUESTION]Which key you want to look?\n";
                std::cin >> lsValue;
                output << lsValue << endl << endl;
                cout << lsValue << endl << endl;
                ht.printSearch(lsValue, output);
                break;
            default:
                return 0;
        }

    }
}
```

ПРИЛОЖЕНИЕ В

ТЕСТИРОВАНИЕ

Входные данные из файла input.txt : 2, 6, 1, 104, 201412, 42, 14, 4, 0

```
MENU:
1: Print HashTable
2: Add element
3: Delete element
4: Print elements by key
Another key: Quit
2
[QUESTION]Which value you want to insert?
3

[INFO]After adding 3
Index 0: 0 =>
Index 1: 6 => 1 =>
Index 2: 2 => 201412 => 42 =>
Index 3: 3 =>
Index 4: 104 => 14 => 4 =>

MENU:
1: Print HashTable
2: Add element
3: Delete element
4: Print elements by key
Another key: Quit
3
[QUESTION]Which value you want to delete?
104

[INFO]After removal 104
Index 0: 0 =>
Index 1: 6 => 1 =>
Index 2: 2 => 201412 => 42 =>
Index 3: 3 =>
Index 4: 14 => 4 =>

MENU:
1: Print HashTable
2: Add element
3: Delete element
4: Print elements by key
Another key: Quit
4
[QUESTION]Which key you want to look?
2

2 => 201412 => 42 =>
MENU:
1: Print HashTable
2: Add element
3: Delete element
4: Print elements by key
Another key: Quit
1
[INFO]Hash Table
Index 0: 0 =>
Index 1: 6 => 1 =>
Index 2: 2 => 201412 => 42 =>
Index 3: 3 =>
Index 4: 14 => 4 =>

MENU:
1: Print HashTable
2: Add element
3: Delete element
4: Print elements by key
Another key: Quit
0
```