



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

---

Отчет по выполнению практического задания №6  
**Тема: «АЛГОРИТМЫ ПОИСКА»**  
Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Петров А.  
Фамилия И.О.

Группа: ИКБО 22-23  
Номер группы

**Москва 2024**

## Содержание

Содержание .....	2
Часть 6.1. Быстрый доступ к данным с помощью хеш-таблиц .....	3
Цель работы.....	3
Ход Работы .....	3
Формулировка задачи .....	3
Математическая модель решения.....	3
Описание подхода к решению .....	4
Код программы с комментариями.....	5
Результаты тестирования.....	8
Часть 6.2. Поиск образца в тексте .....	9
Цель работы.....	9
Ход Работы .....	9
Формулировка задач .....	9
Задание 1. ....	9
Математическая модель решения.....	9
Код программы с комментариями.....	9
Результаты тестирования.....	10
Задание 2. ....	10
Математическая модель решения.....	10
Код программы с комментариями .....	10
Результаты тестирования.....	12
Вывод.....	13
Список литературы .....	14

## **Часть 6.1. Быстрый доступ к данным с помощью хеш-таблиц**

### **Цель работы**

Освоить приёмы хеширования и эффективного поиска элементов множества.

### **Ход Работы**

#### **Формулировка задачи**

Разработайте приложение, которое использует хеш-таблицу (пары «ключ – хеш») для организации прямого доступа к элементам динамического множества полезных данных. Множество реализуйте на массиве, структура элементов (перечень полей) которого приведена в индивидуальном варианте.

Вариант № 17.

*Метод хеширования:*

Цепное хеширование.

*Структура записи файла (ключ – подчеркнутое поле):*

Специализация вуза: код специальности – (прим.: "09.03.01"), название вуза

#### **Математическая модель решения**

##### **Хеш-функция**

Хеш-функция используется для преобразования ключей элементов в индексы хеш-таблицы. В данной реализации используется простая хеш-функция, которая вычисляет сумму кодов символов ключа и берет ее по модулю размера хеш-таблицы.

##### **Вставка**

При вставке элемента в хеш-таблицу вычисляется его хеш-индекс. Если хеш-позиция пуста, создается новый список для хранения элементов с тем же хеш-индексом. Если хеш-позиция занята, элемент добавляется в существующий список.

##### **Удаление**

При удалении элемента из хеш-таблицы вычисляется его хеш-индекс. Затем в списке элементов с тем же хеш-индексом находится элемент с соответствующим ключом и удаляется из списка.

## **Поиск**

При поиске элемента в хеш-таблице вычисляется его хеш-индекс. Затем в списке элементов с тем же хеш-индексом находится элемент с соответствующим ключом.

## **Расширение и рехеширование**

Если коэффициент заполнения хеш-таблицы (отношение количества элементов в таблице к размеру таблицы) превышает заданный порог (обычно 0,75), необходимо расширить таблицу и перехешировать все элементы.

Расширение таблицы заключается в увеличении ее размера в два раза. Перехеширование заключается в повторном вычислении хеш-индексов для всех элементов и их размещении в соответствующих позициях новой таблицы.

## **Цепное разрешение коллизий**

В данной реализации используется цепное разрешение коллизий. Это означает, что элементы с одинаковыми хеш-индексами хранятся в связанных списках. При возникновении коллизии новый элемент добавляется в начало соответствующего списка.

## **Описание подхода к решению**

Описание класса `HashTable`:

Поля:

`table`: вектор списков пар (ключ, значение), представляющий хеш-таблицу. Размер вектора равен размеру таблицы.

`size`: размер хеш-таблицы.

`num_elements`: количество элементов, хранящихся в хеш-таблице.

Методы:

Конструктор:

`HashTable(int size = 10)` создает хеш-таблицу заданного размера.

Основные операции:

`insert(const string& key, const string& value)` вставляет элемент в хеш-таблицу.

`remove(const string& key)` удаляет элемент из хеш-таблицы.

search(const string& key): ищет элемент в хеш-таблице и возвращает его значение.

Вспомогательные методы:

hash(const string& key) вычисляет хеш-индекс для данного ключа.

resize() расширяет хеш-таблицу в два раза и перехэширует все элементы.

print() выводит содержимое хеш-таблицы в консоль.

Описание массива DataSet:

### Код программы с комментариями

```
#include <iostream>
#include <vector>
#include <list>
#include <Windows.h>
using namespace std;

class HashTable {
private:
    vector<list<pair<string, string>>> table;
    int size;
    int num_elements;

public:
    HashTable(int size = 10) : size(size), num_elements(0) {
        table.resize(size);
    }

    unsigned int hash(const string& key) {
        unsigned int hash_value = 0;
        for (int i = 0; i < key.length(); i++) {
            hash_value += (unsigned int)key[i];
        }
        return hash_value % size;
    }

    void insert(const string& key, const string& value) {
        int index = hash(key);
        table[index].push_back(make_pair(key, value));
        num_elements++;
        if (num_elements / size > 0.75) {
            resize();
        }
    }

    void remove(const string& key) {
        int index = hash(key);
        for (auto it = table[index].begin(); it != table[index].end(); it++) {
            if (it->first == key) {
                table[index].erase(it);
                num_elements--;
                break;
            }
        }
    }
}
```

```

string search(const string& key) {
    int index = hash(key);
    for (auto it = table[index].begin(); it != table[index].end(); it++) {
        if (it->first == key) {
            return it->second;
        }
    }
    return "";
}

void resize() {
    vector<list<pair<string, string>>> old_table = table;
    size *= 2;
    table.resize(size);
    num_elements = 0;
    for (auto& list : old_table) {
        for (auto& pair : list) {
            insert(pair.first, pair.second);
        }
    }
}

void print() {
    for (int i = 0; i < size; i++) {
        cout << "Index " << i << ": ";
        if (table[i].empty()) {
            cout << "Empty";
        }
        else {
            for (auto& pair : table[i]) {
                cout << "(" << pair.first << ", " << pair.second << ") ";
            }
        }
        cout << endl;
    }
}

};

class DataSet {
private:
    vector<pair<string, string>> data;
    HashTable hash_table;

public:
    DataSet() : hash_table(10) {}

    void insert(const string& key, const string& value) {
        data.push_back(make_pair(key, value));
        hash_table.insert(key, value);
    }

    void remove(const string& key) {
        hash_table.remove(key);
        data.erase(remove_if(data.begin(), data.end(), [&](const pair<string,
string>& p) { return p.first == key; }));
    }

    string search(const string& key) {
        return hash_table.search(key);
    }

    void print() {
        cout << "Содержимое хеш-таблицы:" << endl;
    }
}

```

```

        hash_table.print();
        /*cout << "Содержимое вектора данных:" << endl;
        for (const auto& pair : data) {
            cout << "(" << pair.first << ", " << pair.second << ") " << "\n";
        }*/
        cout << endl;
    }
};

void menu(DataSet& data)
{
    string command;
    while (true) {
        cout << "Введите команду (insert, remove, search, print, exit): ";
        cin >> command;

        if (command == "insert") {
            string key, value;
            cout << "Введите ключ: ";
            cin >> key;
            cout << "Введите значение: ";
            cin >> value;
            data.insert(key, value);
            cout << "Элемент успешно вставлен." << endl;
        }
        else if (command == "remove") {
            string key;
            cout << "Введите ключ для удаления: ";
            cin >> key;
            data.remove(key);
            cout << "Элемент успешно удален." << endl;
        }
        else if (command == "search") {
            string key;
            cout << "Введите ключ для поиска: ";
            cin >> key;
            string value = data.search(key);
            if (value.empty()) {
                cout << "Элемент с данным ключом не найден." << endl;
            }
            else {
                cout << "Значение элемента: " << value << endl;
            }
        }
        else if (command == "print") {
            data.print();
        }
        else if (command == "exit") {
            break;
        }
        else {
            cout << "Неизвестная команда." << endl;
        }
    }
}

int main() {
    setlocale(LC_ALL, "RU");
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    DataSet data;

    data.insert("09.03.01", "МГУ");
    data.insert("10.05.01", "СПбГУ");
}

```

```

data.insert("11.03.03", "МГТУ");
data.insert("12.04.01", "МФТИ");
data.insert("31.23.12", "ИИТ");
data.insert("09.09.09", "МОС");
data.insert("01.05.04", "ТХТ");
data.insert("04.06.05", "ИКБ");
data.insert("09.04.06", "РТУ");
data.print();
menu(data);
cout << "\n";
return 0;
}

```

## Результаты тестирования

```

C:\Users\artem\source\repos\ <+>
(09.03.01, МГУ)
(10.05.01, СПбГУ)
(11.03.03, МГТУ)
(12.04.01, МФТИ)
(13.03.02, НИУ ВШЭ)

Введите команду (insert, remove, search, print, exit): insert
Введите ключ: 09.03.02
Введите значение: МИРЭА
Элемент успешно вставлен.
Введите команду (insert, remove, search, print, exit): print
(09.03.01, МГУ)
(10.05.01, СПбГУ)
(11.03.03, МГТУ)
(12.04.01, МФТИ)
(13.03.02, НИУ ВШЭ)
(09.03.02, МИРЭА)

Введите команду (insert, remove, search, print, exit):

```

```

C:\Users\artem\source\repos\ <+>
(09.03.01, МГУ)
(10.05.01, СПбГУ)
(11.03.03, МГТУ)
(12.04.01, МФТИ)
(13.03.02, НИУ ВШЭ)

Введите команду (insert, remove, search, print, exit): remove
Введите ключ для удаления: 13.03.02
Элемент успешно удален.
Введите команду (insert, remove, search, print, exit): print
(09.03.01, МГУ)
(10.05.01, СПбГУ)
(11.03.03, МГТУ)
(12.04.01, МФТИ)

Введите команду (insert, remove, search, print, exit): |

```

```

Консоль отладки Microsoft V <+>
(09.03.01, МГУ)
(10.05.01, СПбГУ)
(11.03.03, МГТУ)
(12.04.01, МФТИ)
(13.03.02, НИУ ВШЭ)

Введите команду (insert, remove, search, print, exit): search
Введите ключ для поиска: 10.05.01
Значение элемента: СПбГУ
Введите команду (insert, remove, search, print, exit): exit

```

## Анализ сложности

Вставка, удаление, поиск:  $O(1)$

Расширение и рехеширование:



## Часть 6.2. Поиск образца в тексте

### Цель работы

Освоить приёмы реализации алгоритмов поиска образца в тексте.

### Ход Работы

#### Формулировка задач

Разработайте приложения в соответствии с заданиями в индивидуальном варианте.

Вариант № 2.

1. Дано предложение, состоящее из слов. Сформировать массив слов – целых чисел.

Словом считаем подстроку, ограниченную с двух сторон пробелами.

2. Найти все вхождения подстроки в строку, используя алгоритм Бойера-Мура (только эвристика хорошего суффикса)

#### Задание 1.

#### Математическая модель решения

1. Пробелы служат разделителями слов.
2. Каждое слово должно быть проверено, является ли оно целым числом.
3. Используя стандартные функции для работы со строками в C++, разделим предложение на слова и преобразуем их в числа.

#### Код программы с комментариями

```
#include <iostream>
#include <Windows.h>
#include <vector>
#include <sstream>
#include <string>
using namespace std;
// Функция для проверки, является ли строка целым числом
bool isNumber(const string& str) {
    for (char const& c : str) {
        if (isdigit(c) == 0) return false;
    }
    return true;
}

// Функция для разделения строки на слова и возврата массива целых чисел
vector<int> extractNumbers(const string& sentence) {
    vector<int> numbers;
    stringstream ss(sentence);
    string word;

    while (ss >> word) {
        if (isNumber(word)) {
```

```

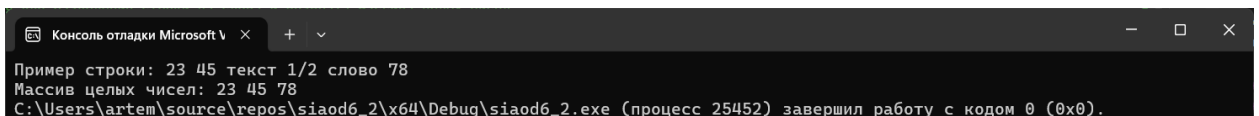
        numbers.push_back( stoi(word));
    }
}
return numbers;
}

int main() {
    setlocale(LC_ALL, "RU");
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    string sentence = "23 45 текст 1/2 слово 78";
    vector<int> numbers = extractNumbers(sentence);
    cout << "Пример строки: " << sentence << endl;
    cout << "Массив целых чисел: ";
    for (int num : numbers) {
        cout << num << " ";
    }
    return 0;
}

```

## Результаты тестирования



Консоль отладки Microsoft V

```

Пример строки: 23 45 текст 1/2 слово 78
Массив целых чисел: 23 45 78
C:\Users\artem\source\repos\siaod6_2\x64\Debug\siaod6_2.exe (процесс 25452) завершил работу с кодом 0 (0x0).

```

## Задание 2.

### Математическая модель решения

1. Предварительная обработка строки образца для вычисления таблицы смещений на основе суффиксов.
2. Сравнение подстроки с текстом с конца образца, начиная с конца рассматриваемого окна.
3. При несовпадении происходит сдвиг на основе длины совпадающего суффикса.

### Код программы с комментариями

```

#include <iostream>
#include <vector>
#include <string>
#include <Windows.h>
using namespace std;
// Функция для вычисления таблицы сдвигов по эвристике хорошего суффикса
vector<int> goodSuffix(const string& pattern) {
    int m = pattern.size();
    vector<int> shift(m + 1, m); // Инициализация смещений размером m+1
    vector<int> border(m + 1, 0);

    int i = m, j = m + 1;
    border[i] = j;

    while (i > 0) {
        while (j <= m && pattern[i - 1] != pattern[j - 1]) {
            if (shift[j] == m) {
                shift[j] = j - i;
            }
        }
    }
}

```

```

        j = border[j];
    }
    i--;
    j--;
    border[i] = j;
}

for (int i = 0; i <= m; i++) {
    if (shift[i] == m) {
        shift[i] = j;
    }
    if (i == j) {
        j = border[j];
    }
}
return shift;
}

// Функция для поиска подстроки в строке с использованием алгоритма Бойера-Мура
vector<int> boyerMooreSearch(const string& text, const string& pattern) {
    int n = text.size();
    int m = pattern.size();

    if (m == 0) return {}; // Если подстрока пустая, возвращаем пустой результат

    vector<int> result; // Массив для хранения индексов вхождений
    vector<int> shift = goodSuffix(pattern);

    int s = 0; // Смещение относительно строки
    while (s <= n - m) {
        int j = m - 1;

        // Сравнение с конца образца
        while (j >= 0 && pattern[j] == text[s + j]) {
            j--;
        }

        // Если образец полностью совпал с подстрокой в тексте
        if (j < 0) {
            result.push_back(s);
            s += shift[0];
        }
        else {
            // Иначе сдвигаем строку по таблице сдвигов
            s += shift[j + 1];
        }
    }

    return result;
}

int main() {
    string text = "abracadabra";
    string pattern = "abra";

    vector<int> positions = boyerMooreSearch(text, pattern);

    cout << "Позиции вхождений: ";
    for (int pos : positions) {
        cout << pos << " ";
    }
    return 0;
}

```

## Результаты тестирования

```
Консоль отладки Microsoft Visual Studio
строка - ABOBQWERTBOBQWERTBOB
подстрока - BOB
Позиции вхождения: 1 9 17
C:\Users\artem\source\repos\siaod6_2_2\x64\Debug\siaod6_2_2.exe (процесс 27276) завершил работу с кодом 0 (0x0).
```

```
Консоль отладки Microsoft Visual Studio
строка - ABOBQWERTBOBQWERTBOB
подстрока - C
Позиции вхождения:
C:\Users\artem\source\repos\siaod6_2_2\x64\Debug\siaod6_2_2.exe (процесс 19220) завершил работу с кодом 0 (0x0).
```

## **Вывод**

В ходе выполнения практической работы были реализованы два задания, направленные на освоение алгоритмов работы с текстом и поиска подстроки в строке.

В первом задании успешно реализована функция для разбиения строки на слова и формирования массива целых чисел. Программа корректно обрабатывает строку, извлекая только числовые значения, что продемонстрировано на примерах тестирования.

Во втором задании реализован алгоритм Бойера-Мура с эвристикой хорошего суффикса. В процессе реализации была выполнена предварительная обработка подстроки для построения таблицы сдвигов, что позволяет эффективно находить вхождения подстроки в строку. Алгоритм был протестирован на различных примерах, как для успешного, так и для неуспешного поиска,

## Список литературы

1. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. – СПб: Питер, 2017. – 288 с.
2. Кормен Т.Х. и др. Алгоритмы. Построение и анализ, 2013. – С. 285-318.
3. Страуструп Б. Программирование. Принципы и практика с использованием C++. 2-е изд., 2016.
4. Документация по языку C++ [Электронный ресурс]. URL: <https://docs.microsoft.com/ruru/cpp/cpp>.
5. Алгоритмы – всё об алгоритмах / Хабр [Электронный ресурс]. URL: <https://habr.com/ru/hub/algorithms/>.