|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра инструментального и прикладного программного обеспечения | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 6.2** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Поиск образца в тексте»** | |
|  | |
| Выполнил студент группы ИКБО-10-23 | Харитонов А.Н. |
| Принял преподаватель | Макеева О.В. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_ \_ \_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2024

# **Цель работы**

Освоить приемы реализации алгоритмов поиска образца в тексте.

# **Ход работы**

Для реализации линейного поиска была написана функция linearSearch, которая принимает на вход строку, в которой происходит поиск, подстроку для поиска и счетчик для сравнений.

*int* linearSearch(const std::string& *text*, const std::string& *pattern*, *int*& *counter*) {

*int* n = *text*.size();

*int* m = *pattern*.size();

    if (m > n) return -1;

    for (*int* i = 0; i <= n - m; ++i) {

*int* j = 0;

        while (j < m && *text*[i + j] == *pattern*[j]) {

*counter*++;

            j++;

        }

*counter*++;

        if (j == m) {

            return i;

        }

    }

    return -1;

}

Приложение 1

Для реализации поиска Бойера-Мура-Хорспула были написаны функции buildShiftTable для генерации таблицы смещений и boyerMooreHorspoolSearch для самого поиска.

std::unordered\_map<*char*, *int*> buildShiftTable(const std::string& *pattern*) {

    std::unordered\_map<*char*, *int*> table;

*int* m = *pattern*.size();

    for (*int* i = 0; i < m - 1; ++i) {

        table[*pattern*[i]] = m - i - 1;

    }

    return table;

}

*int* boyerMooreHorspoolSearch(const std::string& *text*, const std::string& *pattern*, *int*& *counter*) {

*int* n = *text*.size();

*int* m = *pattern*.size();

    if (m > n) return -1;

    std::unordered\_map<*char*, *int*> shiftTable = buildShiftTable(*pattern*);

*int* i = 0;

*int* lastOccurrence = -1;

    while (i <= n - m) {

*int* j = m - 1;

        while (j >= 0) {

*counter*++;

            if (*text*[i + j] != *pattern*[j]) {

                break;

            }

            j--;

        }

        if (j < 0) {

            lastOccurrence = i;

            i++;

        } else {

            i += shiftTable.count(*text*[i + m - 1]) ? shiftTable[*text*[i + m - 1]] : m;

        }

    }

    return lastOccurrence;

}

Приложение 2.

|  |  |  |
| --- | --- | --- |
| "Тестовая строка" | Линейный поиск | Продвинутый поиск |
| Тест | 9 | 12 |
| ока | 37 | 11 |
| я | 19 | 16 |
| строка | 36 | 15 |
| Артем | 25 | 3 |

Таблица 1.

Сложность линейного поиска: O(n\*m)

Сложность продвинутого поиска: O(n) | O(n\*m)

N – длина строка

M – длина подстроки

1. **Вывод**

В ходе работы я освоил приемы реализации алгоритмов поиска образцов в тексте