Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

ПНИПУ

**Лабораторная работа**

**«Сортировки»**

Выполнил:

студент группы РИС-23-2б

Ившин Максим Сергеевич

Проверила:

доцент кафедры ИТАС

О.А. Полякова

2024 г.

**Разработка алгоритма**

**Постановка задачи:**

Реализовать поиски: линейный, бинарный, интерполяционный, линейный поиск строки, Бойера-Мура, Кнута-Морриса-Пратта.

**Анализ задачи:**

1. Создадим главный файл для основного кода**.**
2. Для удобства все поиски будут находиться в отдельных файлах.
3. Для использования поисков в главном файле будем подключать файлы с кодом поисков для дальнейшего использования.

Бинарный поиск

#pragma once

#include <iostream>

using namespace std;

int binary\_search(int \*mas, int &key, int size)

{

    int left = 0, right = size - 1, mid;

    while (left <= right)

    {

        mid = (right + left) / 2;

        if (mas[mid] == key)

        {

            return mid;

        }

        else

        {

            if (mas[mid] < key)

                left = mid + 1;

            else

                right = mid - 1;

        }

    }

    return -1;

}

Интерполяционный поиск

#pragma once

#include <iostream>

using namespace std;

int interpol\_search(int \*mas, int &key, int size)

{

    int left = 0, right = size - 1, mid;

    while (left <= right)

    {

        mid = left + (((key - mas[left]) \* (right - left)) / (mas[right] - mas[left]));

        if (mas[mid] == key)

        {

            return mid;

        }

        else

        {

            if (mas[mid] < key)

                left = mid + 1;

            else

                right = mid - 1;

        }

    }

    return -1;

}

Линейный поиск

#pragma once

#include <iostream>

using namespace std;

int Line\_search(int \*mas, int &key, int size)

{

    for (int i = 0; i < size; i++)

    {

        if (mas[i] == key)

        {

            return i;

        }

    }

    return -1;

}

Поиск Бойера-Мура

#pragma once

#include <iostream>

using namespace std;

void table\_formation(int \*shift\_table, string &substr)

{

    for (int i = 0; i < 256; i++)

    {

        shift\_table[i] = substr.length();

    }

    for (int i = substr.length() - 2; i >= 0; i--)

    {

        if (shift\_table[int((unsigned char)substr[i])] == substr.length())

        {

            shift\_table[int((unsigned char)substr[i])] = substr.length() - i - 1;

        }

    }

}

int Boiere\_Moore(string &str, string &substr, int \*shift\_table)

{

    int found\_index = -1;

    int current\_char = substr.length() - 1;

    while (current\_char < str.length() && found\_index == -1)

    {

        bool flag = true;

        for (int i = substr.length() - 1; i >= 0 && found\_index == -1 && flag; i--)

        {

            if (substr[i] != str[current\_char - substr.length() + i + 1])

            {

                current\_char += shift\_table[int((unsigned char)str[current\_char - substr.length() + i + 1])];

                flag = false;

            }

            else if (i == 0)

            {

                found\_index = current\_char - substr.length() + 1;

            }

        }

    }

    return found\_index;

}

Поиск Кнута-Морриса-Пратта

#pragma once

#include <iostream>

#include <E:\\source\\repos\\TestCpp\\usefulFuncs.h>

using namespace std;

void prefix\_function(string sub\_str, int \*mas\_prefix)

{

    int i = 1, j = 0;

    while (i < sub\_str.size())

    {

        if (sub\_str[i] == sub\_str[j])

        {

            mas\_prefix[i] = j + 1;

            i++;

            j++;

        }

        else

        {

            if (j == 0)

            {

                mas\_prefix[i] = 0;

                i++;

            }

            else

            {

                j = mas\_prefix[j - 1];

            }

        }

    }

}

int KMP\_search(string all\_str, string sub\_str)

{

    int \*mas\_prefix = new int[sub\_str.size()];

    mas\_prefix[0] = 0;

    int str\_index = 0, substr\_index = 0, found\_index = -1;

    prefix\_function(sub\_str, mas\_prefix);

    while (str\_index < all\_str.size() && found\_index == -1)

    {

        if (sub\_str[substr\_index] == all\_str[str\_index])

        {

            str\_index++;

            substr\_index++;

        }

        if (substr\_index == sub\_str.size())

        {

            found\_index = str\_index - sub\_str.size();

        }

        else if (str\_index < all\_str.size() && sub\_str[substr\_index] != all\_str[str\_index]) // если не прошли строку и элементы не совпадают

        {

            if (substr\_index == 0)

            {

                str\_index++;

            }

            else

            {

                substr\_index = mas\_prefix[substr\_index - 1];

            }

        }

    }

    print(mas\_prefix, sub\_str.size());

    cout << endl;

    delete[] mas\_prefix;

    return found\_index;

}

Линейный поиск подстроки в строке

#pragma once

#include <iostream>

using namespace std;

int line\_str\_search(string &str, string &substr)

{

    bool is\_found;

    for (int i = 0; i <= str.length() - substr.length(); i++)

    {

        is\_found = true;

        if (str[i] == substr[0])

        {

            for (int j = 1; j < substr.length() && is\_found; j++)

            {

                is\_found = (str[i + j] == substr[j]);

            }

            if (is\_found)

            {

                return i;

            }

        }

    }

    return -1;

}

**GitHub:** *https://github.com/geroineee/PNRPU*