Министерство образования и науки Российской Федерации

Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №14**

Дисциплина: «Основы теории алгоритмов и структуры данных»

Тема: Методы поиска: метод Кнута-Мориса Пратта, метод Бойера-Мура, метод прямого поиска подстроки в строке

Вариант 18

Выполнил:

Студент группы РИС-20-1б

Тараканов Д. М.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

Пермь

2021 год

# **Цель работы**

Получить практические навыки работы с методами поисков подстроки в строке.

# **Постановка задачи**

1. Создать динамический массив из записей (в соответствии с вариантом), содержащий не менее 100 элементов.
2. Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для поиска использовать метод Кнута-Мориса Пратта, метод Бойера-Мура, метод прямого поиска подстроки в строке.

(18 вариант)

1. Данные: ФИО, дата\_рождения, №телефона.
2. Ключ: №телефона.

# **Анализ задачи**

1. Определить какие действия необходимо выполнить для решения задачи:

* Разработать структуру Directory которая хранит в себе данные (ФИО, дата\_рождения, №телефона).

struct Directory {

string data;

};

* Разработать функцию чтения данных из файла и занесения их в структуру.

int ReadFile(Directory book[], ifstream& file) {

string CurrentLine;

int index = 0;

while (!file.eof()) {

getline(file, CurrentLine);

book[index].data = CurrentLine;

index++;

}

return index;

};

* Разработать меню переключения между методами поиска.

void SwitchSearchMetod(int quantity, Directory book[]) {

int MethodNumber = -1, index = 0;

unsigned long long int key;

while ((MethodNumber < 0) || (MethodNumber > 1)) {

cout << "\nВведите номер метода поиска подстроки в строке\n\n0 - метод прямого поиска\n1 - метод Кнута-Мориса-Пратта\n2 - метод Бойера-Мура\n\nВаш выбор: ";

cin >> MethodNumber;

switch (MethodNumber) {

case (0):

cout << "\nВы выбрали метод прямого поиска\n";

cout << "\nВведите номер искомого телефона: ";

cin >> key;

while ((index < quantity) && (otvet == false)) {

otvet = DirectSearch(book[index].data, key);

if (otvet == true) {

cout << "\nРезультат поиска:\n" << book[index].data << endl;

}

index++;

}

if (otvet == false) {

cout << "\nПоиск не выдал результатов\n";

}

break;

case (1):

cout << "\nВы выбрали метод Кнута-Мориса-Прата\n";

cout << "\nВведите номер искомого телефона: ";

cin >> key;

while ((index < quantity) && (otvet == false)) {

otvet = KnuthMorrisPratt(book[index].data, key);

if (otvet == true) {

cout << "\nРезультат поиска:\n" << book[index].data << endl;

}

index++;

}

if (otvet == false) {

cout << "\nПоиск не принес результатов!\nТакого номера нет в массиве!\n";

}

break;

case (2):

cout << "\nВы выбрали метод Бойера-Мура\n";

cout << "\nВведите номер искомого телефона: ";

cin >> key;

while ((index < quantity) && (otvet == false)) {

otvet = BoyerMoore(book[index].data, key);

if (otvet == true) {

cout << "\nРезультат поиска:\n" << book[index].data << endl;

}

index++;

}

if (otvet == false) {

cout << "\nПоиск не принес результатов!\nТакого номера нет в массиве!\n";

}

break;

default: cout << "\nВведён не номер метода поиска!\n";

break;

}

}

};

* Разработать функции методов поиска подстроки в строке Прямого, Кнутта-Мориса-Пратта и Бойера-Мура.

bool DirectSearch(string row, string key) {

int LengthRow = row.length(), LengthKey = key.length();

for (int i = 0; i < LengthRow - LengthKey + 1; i++) {

for (int j = 0; j < LengthKey; j++) {

if (key[j] != row[i + j]) {

j = LengthKey;

}

else if (j == LengthKey - 1) {

return true;

}

}

}

return false;

};

bool KnuthMorrisPratt(string row, string key) {

int LengthRow = row.length(), LengthKey = key.length(), IndexRow = 0, IndexKey = 0, i = 1, j = 0;

int\* array = new int[LengthKey];

array[0] = 0;

for (int c = 0; c < LengthKey; c++)

{

if (key[i] == key[j]) {

array[i] = j + 1;

i++;

j++;

}

else if (j == 0) {

array[i] = 0;

i++;

}

else {

j = array[j - 1];

}

}

while (IndexRow < LengthRow)

{

if (row[IndexRow] == key[IndexKey])

{

IndexRow++;

IndexKey++;

if (IndexKey == LengthKey) {

return true;

}

}

if (row[IndexRow] != key[IndexKey])

{

if (IndexKey == 0) {

IndexRow++;

}

else {

IndexKey = array[IndexKey - 1];

}

}

}

return false;

};

bool BoyerMoore(string row, string key) {

int LengthRow = row.length(), LengthKey = key.length();

int array[256];

for (int i = 0; i < 256; i++) {

array[i] = LengthKey;

}

for (int i = LengthKey - 2; i >= 0; i--) {

if (array[int(unsigned char(key[i]))] == LengthKey) {

array[int(unsigned char(key[i]))] = LengthKey - i - 1;

}

}

int KeyPosition = LengthKey - 1;

while (KeyPosition < LengthRow) {

if (key[LengthKey - 1] != row[KeyPosition]) {

KeyPosition += array[int(unsigned char(row[KeyPosition]))];

}

else {

for (int i = LengthKey - 1; i >= 0; i--) {

if (key[i] != row[KeyPosition - LengthKey + i + 1]) {

KeyPosition += array[int(unsigned char(row[KeyPosition]))];

i = -1;

}

else if (i == 0) {

return true;

}

}

}

}

return false;

}

1. С какими типами данных действие надо сделать, в каком виде эти данные будут представлены:

* Для хранения ФИО, даты рождения, № телефона в структуре реализована переменная data типа string.

string data;

* Для хранения данных реализован массив book на 100 элементов типа Directory.

Directory book[100];

* Для хранения ключа поиска реализована переменная key типа string.

string key;

* Для чтения данных из файла реализована переменная inputfile типа ifstream

ifstream inputfile("F1.txt");

* Для подсчета кол-ва элементов в массиве реализована переменная quantity типа int.

int quantity = ReadFile(book, inputfile);

1. Реализация операций ввода и вывода данных:

* Ввод данных в структуру реализован через чтение строк из файла.

while (!file.eof()) {

getline(file, CurrentLine);

book[index].data = CurrentLine;

index++;

}

* Вывод данных в консоль реализован при помощи оператора cout.

for (int i = 0; i < N; i++) {

cout << book[i].data << endl;

}

1. Поставленные задачи будут решены при помощи следующих действий:

* Программа считывает из файла данные и подсчитывает кол-во введённых элементов в массив book.

int quantity = ReadFile(book, inputfile);

int ReadFile(Directory book[], ifstream& file) {

string CurrentLine;

int index = 0;

while (!file.eof()) {

getline(file, CurrentLine);

book[index].data = CurrentLine;

index++;;

}

return index;

};

* Программа запрашивает у пользователя номер метода поиска, а затем запрашивает на ввод ключ поиска. При выборе интерполяционного поиска происходит сортировка массива данных.

while ((MethodNumber < 0) || (MethodNumber > 4)) {

cout << "\nВведите номер метода поиска подстроки в строке\n\n0 - метод прямого поиска\n1 - метод Кнута-Мориса-Пратта\n2 - метод Бойера-Мура\n\nВаш выбор: ";

cin >> MethodNumber;

switch (MethodNumber) {

case (0):

cout << "\nВы выбрали метод прямого поиска\n";

cout << "\nВведите номер искомого телефона: ";

cin >> key;

while ((index < quantity) && (otvet == false)) {

otvet = DirectSearch(book[index].data, key);

if (otvet == true) {

cout << "\nРезультат поиска:\n" << book[index].data << endl;

}

index++;

}

if (otvet == false) {

cout << "\nПоиск не выдал результатов\n";

}

break;

case (1):

cout << "\nВы выбрали метод Кнута-Мориса-Прата\n";

cout << "\nВведите номер искомого телефона: ";

cin >> key;

while ((index < quantity) && (otvet == false)) {

otvet = KnuthMorrisPratt(book[index].data, key);

if (otvet == true) {

cout << "\nРезультат поиска:\n" << book[index].data << endl;

}

index++;

}

if (otvet == false) {

cout << "\nПоиск не принес результатов!\nТакого номера нет в массиве!\n";

}

break;

case (2):

cout << "\nВы выбрали метод Бойера-Мура\n";

cout << "\nВведите номер искомого телефона: ";

cin >> key;

while ((index < quantity) && (otvet == false)) {

otvet = BoyerMoore(book[index].data, key);

if (otvet == true) {

cout << "\nРезультат поиска:\n" << book[index].data << endl;

}

index++;

}

if (otvet == false) {

cout << "\nПоиск не принес результатов!\nТакого номера нет в массиве!\n";

}

break;

default: cout << "\nВведён не номер метода поиска!\n";

break;

}

}

* В случаи выбора прямого поиска подстроки в строке происходит посимвольное сравнение ключа (подстроки) с элементом массива (строка) при помощи оператора for. При символьном совпадении (нахождении) подстроки со строкой программа меняет значение flag, а затем выводит в консоль элемент массива с найденным в ней ключом.

int LengthRow = row.length(), LengthKey = key.length();

for (int i = 0; i < LengthRow - LengthKey + 1; i++) {

for (int j = 0; j < LengthKey; j++) {

if (key[j] != row[i + j]) {

j = LengthKey;

}

else if (j == LengthKey - 1) {

return true;

}

}

}

return false;

* В случаи выбора алгоритма Кнута-Мориса Прата.

int LengthRow = row.length(), LengthKey = key.length(), IndexRow = 0, IndexKey = 0, i = 1, j = 0;

int\* array = new int[LengthKey];

array[0] = 0;

for (int c = 0; c < LengthKey; c++)

{

if (key[i] == key[j]) {

array[i] = j + 1;

i++;

j++;

}

else if (j == 0) {

array[i] = 0;

i++;

}

else {

j = array[j - 1];

}

}

while (IndexRow < LengthRow)

{

if (row[IndexRow] == key[IndexKey])

{

IndexRow++;

IndexKey++;

if (IndexKey == LengthKey) {

return true;

}

}

if (row[IndexRow] != key[IndexKey])

{

if (IndexKey == 0) {

IndexRow++;

}

else {

IndexKey = array[IndexKey - 1];

}

}

}

return false;

* В случаи выбора алгоритм Бойера-Мура

int LengthRow = row.length(), LengthKey = key.length();

int array[256];

for (int i = 0; i < 256; i++) {

array[i] = LengthKey;

}

for (int i = LengthKey - 2; i >= 0; i--) {

if (array[int(unsigned char(key[i]))] == LengthKey) {

array[int(unsigned char(key[i]))] = LengthKey - i - 1;

}

}

int KeyPosition = LengthKey - 1;

while (KeyPosition < LengthRow) {

if (key[LengthKey - 1] != row[KeyPosition]) {

KeyPosition += array[int(unsigned char(row[KeyPosition]))];

}

else {

for (int i = LengthKey - 1; i >= 0; i--) {

if (key[i] != row[KeyPosition - LengthKey + i + 1]) {

KeyPosition += array[int(unsigned char(row[KeyPosition]))];

i = -1;

}

else if (i == 0) {

return true;

}

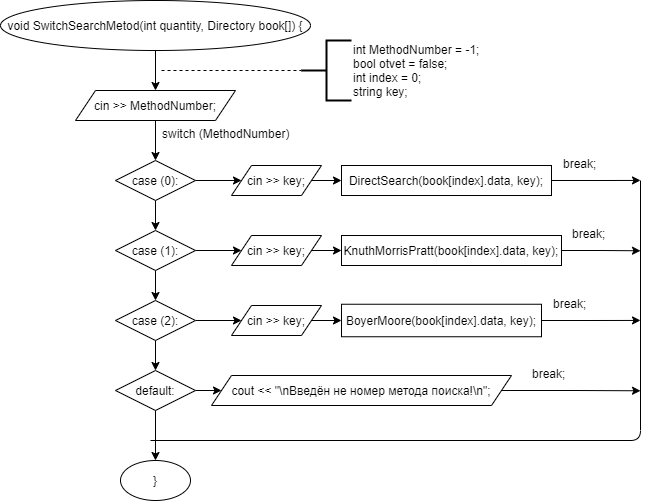
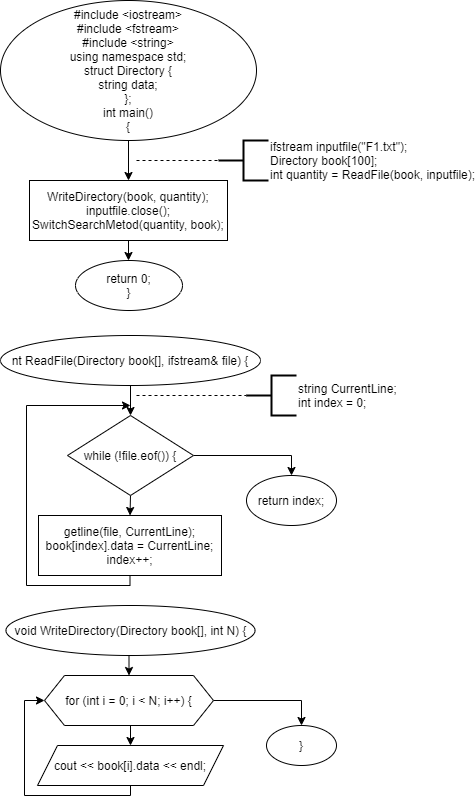
}

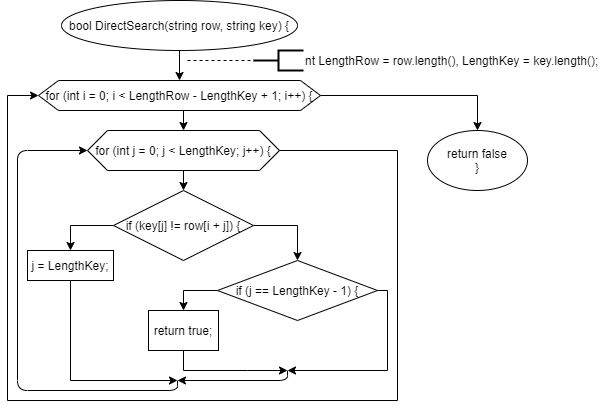
}

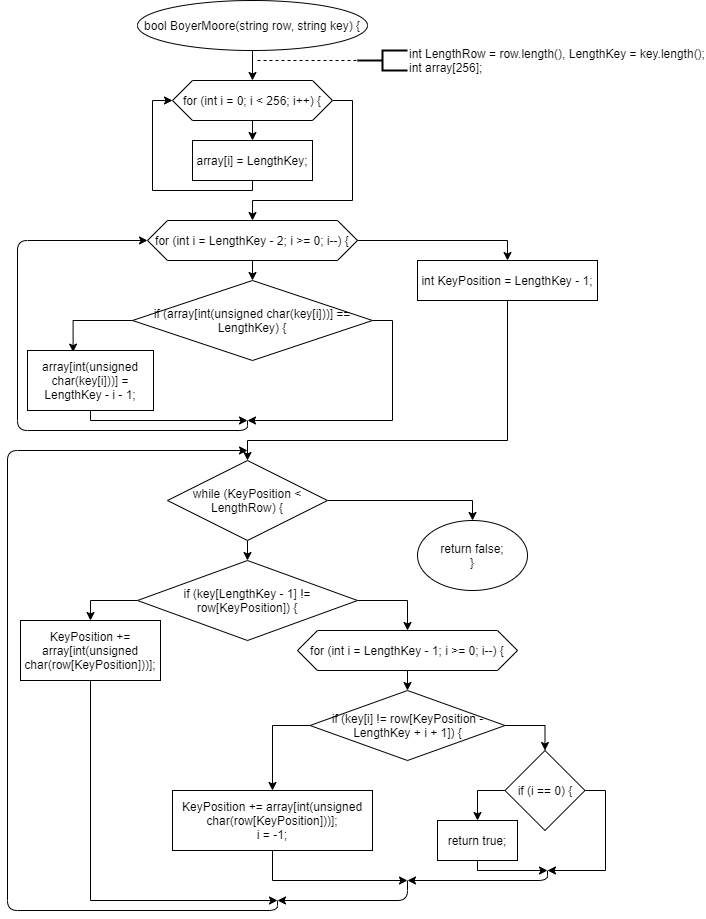
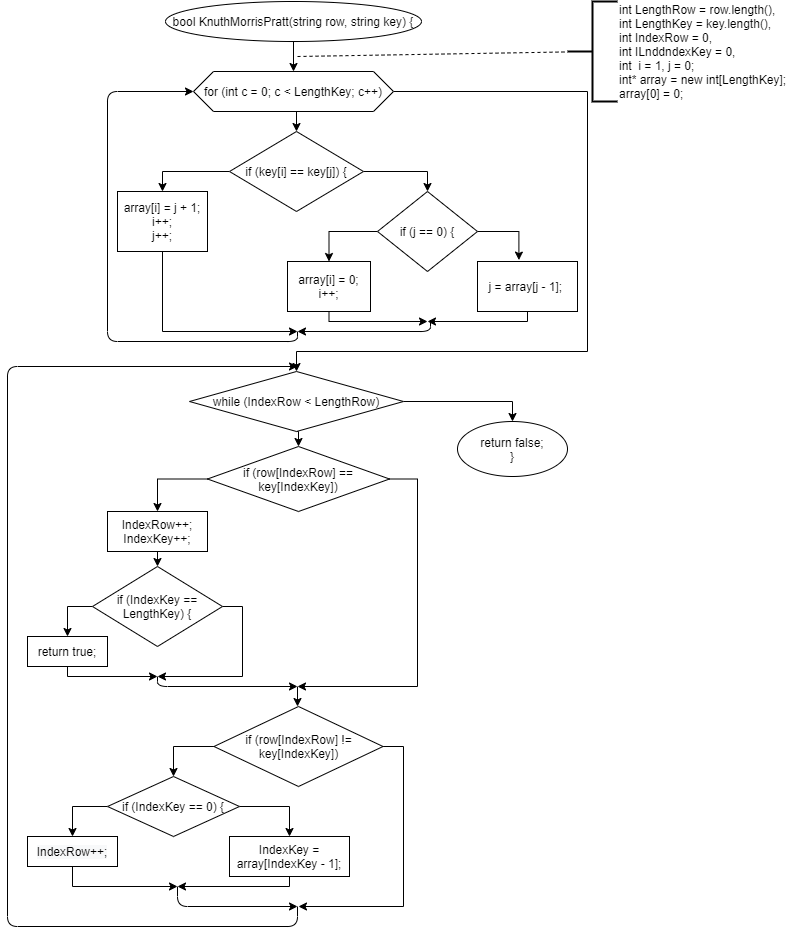
}

return false;

**Блок схема**







# **Код программы на языке C++**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct Directory {

string data;

};

int ReadFile(Directory book[], ifstream& file) {

string CurrentLine;

int index = 0;

while (!file.eof()) {

getline(file, CurrentLine);

book[index].data = CurrentLine;

index++;

}

return index;

};

void WriteDirectory(Directory book[], int N) {

for (int i = 0; i < N; i++) {

cout << book[i].data << endl;

}

};

bool DirectSearch(string row, string key) {

int LengthRow = row.length(), LengthKey = key.length();

for (int i = 0; i < LengthRow - LengthKey + 1; i++) {

for (int j = 0; j < LengthKey; j++) {

if (key[j] != row[i + j]) {

j = LengthKey;

}

else if (j == LengthKey - 1) {

return true;

}

}

}

return false;

};

bool KnuthMorrisPratt(string row, string key) {

int LengthRow = row.length(), LengthKey = key.length(), IndexRow = 0, IndexKey = 0, i = 1, j = 0;

int\* array = new int[LengthKey];

array[0] = 0;

for (int c = 0; c < LengthKey; c++)

{

if (key[i] == key[j]) {

array[i] = j + 1;

i++;

j++;

}

else if (j == 0) {

array[i] = 0;

i++;

}

else {

j = array[j - 1];

}

}

while (IndexRow < LengthRow)

{

if (row[IndexRow] == key[IndexKey])

{

IndexRow++;

IndexKey++;

if (IndexKey == LengthKey) {

return true;

}

}

if (row[IndexRow] != key[IndexKey])

{

if (IndexKey == 0) {

IndexRow++;

}

else {

IndexKey = array[IndexKey - 1];

}

}

}

return false;

};

bool BoyerMoore(string row, string key) {

int LengthRow = row.length(), LengthKey = key.length();

int array[256];

for (int i = 0; i < 256; i++) {

array[i] = LengthKey;

}

for (int i = LengthKey - 2; i >= 0; i--) {

if (array[int(unsigned char(key[i]))] == LengthKey) {

array[int(unsigned char(key[i]))] = LengthKey - i - 1;

}

}

int KeyPosition = LengthKey - 1;

while (KeyPosition < LengthRow) {

if (key[LengthKey - 1] != row[KeyPosition]) {

KeyPosition += array[int(unsigned char(row[KeyPosition]))];

}

else {

for (int i = LengthKey - 1; i >= 0; i--) {

if (key[i] != row[KeyPosition - LengthKey + i + 1]) {

KeyPosition += array[int(unsigned char(row[KeyPosition]))];

i = -1;

}

else if (i == 0) {

return true;

}

}

}

}

return false;

}

void SwitchSearchMetod(int quantity, Directory book[]) {

int MethodNumber = -1;

bool otvet = false;

int index = 0;

string key;

while ((MethodNumber < 0) || (MethodNumber > 4)) {

cout << "\nВведите номер метода поиска подстроки в строке\n\n0 - метод прямого поиска\n1 - метод Кнута-Мориса-Пратта\n2 - метод Бойера-Мура\n\nВаш выбор: ";

cin >> MethodNumber;

switch (MethodNumber) {

case (0):

cout << "\nВы выбрали метод прямого поиска\n";

cout << "\nВведите номер искомого телефона: ";

cin >> key;

while ((index < quantity) && (otvet == false)) {

otvet = DirectSearch(book[index].data, key);

if (otvet == true) {

cout << "\nРезультат поиска:\n" << book[index].data << endl;

}

index++;

}

if (otvet == false) {

cout << "\nПоиск не выдал результатов\n";

}

break;

case (1):

cout << "\nВы выбрали метод Кнута-Мориса-Прата\n";

cout << "\nВведите номер искомого телефона: ";

cin >> key;

while ((index < quantity) && (otvet == false)) {

otvet = KnuthMorrisPratt(book[index].data, key);

if (otvet == true) {

cout << "\nРезультат поиска:\n" << book[index].data << endl;

}

index++;

}

if (otvet == false) {

cout << "\nПоиск не принес результатов!\nТакого номера нет в массиве!\n";

}

break;

case (2):

cout << "\nВы выбрали метод Бойера-Мура\n";

cout << "\nВведите номер искомого телефона: ";

cin >> key;

while ((index < quantity) && (otvet == false)) {

otvet = BoyerMoore(book[index].data, key);

if (otvet == true) {

cout << "\nРезультат поиска:\n" << book[index].data << endl;

}

index++;

}

if (otvet == false) {

cout << "\nПоиск не принес результатов!\nТакого номера нет в массиве!\n";

}

break;

default: cout << "\nВведён не номер метода поиска!\n";

break;

}

}

};

int main()

{

system("chcp 1251");

ifstream inputfile("F1.txt");

if (inputfile.is\_open()) {

Directory book[100];

int quantity = ReadFile(book, inputfile);

cout << "\nМассив данных:\n\n";

WriteDirectory(book, quantity);

inputfile.close();

SwitchSearchMetod(quantity, book);

}

else {

cout << "Ошибка! Не удалось открыть файл F1.txt.\n";

}

system("pause");

return 0;

}

# **Скриншоты тестов**

