Пермский Национальный Исследовательский Политехнический Университет

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Лабораторная работа № 12

по дисциплине

«Основы алгоритмизации и программирования»

Тема: «Методы поиска. Прямой поиск подстроки в строке, линейный и интерполяционный методы»

Вариант № 17

Выполнил:

студент группы РИС-20-2б

Пономарев Егор Витальевич

Проверила:

доцент кафедры ИТАС

Полякова О. А.

Пермь, 2021

**Цель работы**

Получить практические навыки работы с очередями.

**Постановка задачи**

**Анализ задачи**

1. В первую очередь подключаем библиотеки <iostream>, <fstream> (для работы с файловым вводом-выводом), <cstdlib> (стандартная библиотека языка Си, которая содержит в себе функции преобразования типов и другие).
2. Данные о пользователях будем хранить в оперативной памяти в виде массива структур, информационными полями которых будут являться string name, string data, long long number.



1. Объявляем глобальную переменную int n (количество пользователей), string key (ключ, по которому будет осуществляться поиск) и указатель на начало массива User\* users\_arr.
2. Главную функцию необходимо организовать в виде меню. Для этого предоставляем пользователю выбор: загрузить список из файла “База данных.txt” или ввести вручную (используем условное ветвление if-else).



1. Если необходимо загрузить список из файла, открываем соответствующий файл и с помощью цикла while считаем количество строк int count, пока не достигнут конец (количество пользователей равно count / 4, так как один занимает 4 строки, включая отступ). Считывание строк осуществляется с помощью getline.



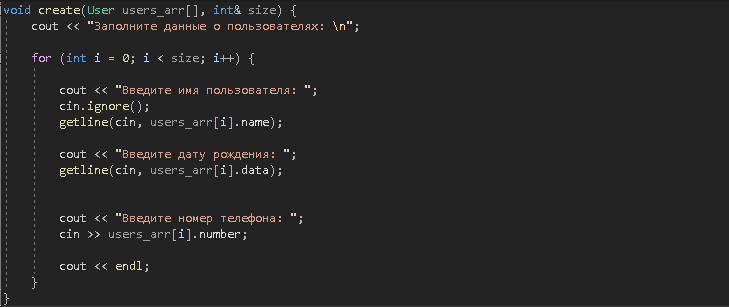
1. Закрываем файл и открываем заново, чтобы переместить указатель в начало. Объявляем динамический массив users\_arr и заполняем его данными из файла с помощью цикла, пока не будет достигнут конец. Обязательно закрываем файл после проделанных действий. (Так как строка с номером будет иметь тип string, а нам необходим long long int, воспользуемся преобразованием типов: istringstream(<row>) >> x, где row – это временная строка, а x – временная переменная для хранения номера)



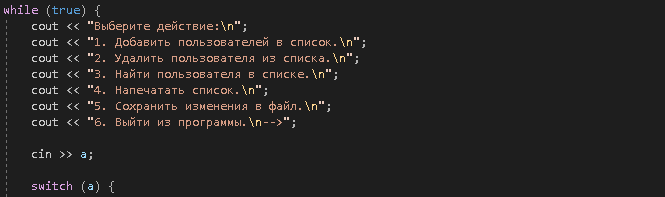
1. Если нужно ввести список вручную, просим ввести размер и так же объявляем динамический массив.



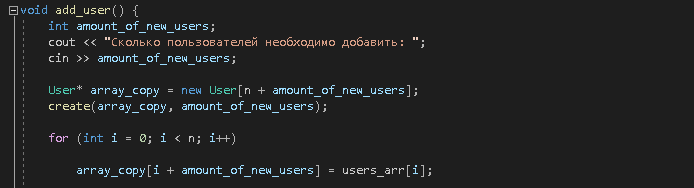
1. Вызываем функцию создания массива void create (User users\_arr[], int& size) и с помощью цикла for заполняем его.



1. Функцию печати void print () реализуем так же с помощью цикла for.
2. После того, как выполнена подготовительная часть, необходимо проработать меню выбора действий. Реализуем его с помощью конструкции switch и поместим в бесконечный цикл while (true), чтобы работа не прекращалась после выполнения одного действия. (с возможностью завершения работы - точкой выхода является exit (0)).



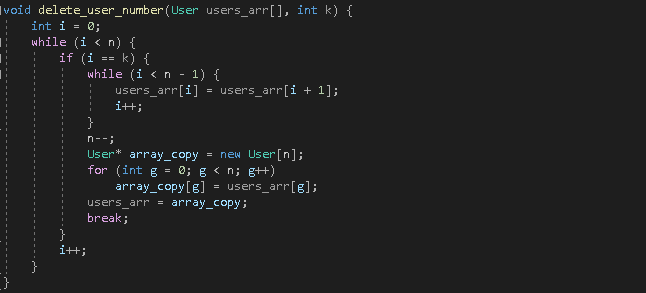
1. Функция добавления пользователя void add\_user ():
2. Чтобы ввести новых пользователей, объявляем новый массив размера n + amount\_of\_new\_users (количество новых пользователей). В начало нового массива вводим новые данные, а в остальную часть копируем данные из старого массива.



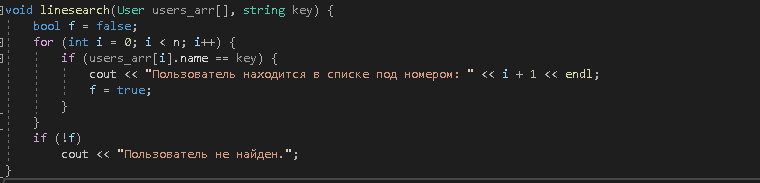
1. К количеству пользователей прибавляем количество новых и настраиваем указатель users\_arr на новый массив.



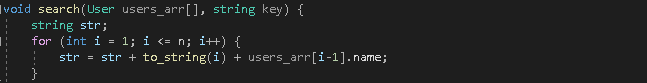
1. Для удаления пользователя по номеру в списке, создаем функцию, которая смещает элементы влево, начиная со следующего от того, который необходимо удалить. Так же создаем новый динамический массив размера n – 1, куда скопируем данные.



1. Для реализации линейного поиска просто воспользуемся циклом for, который пройдет по всем элементам массива, и сравнит ФИО с ключом. Выведем номер пользователя, если он найдется.



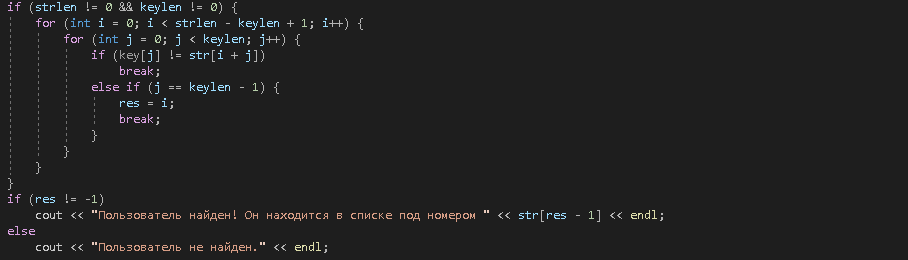
1. Для реализации прямого поиска подстроки (шаблона) в строке, создаем функцию void search (User users\_arr[], string key). Объявляем строку string str, в которую поместим все ФИО с помощью цикла вместе с номером, чтобы потом извлечь его.



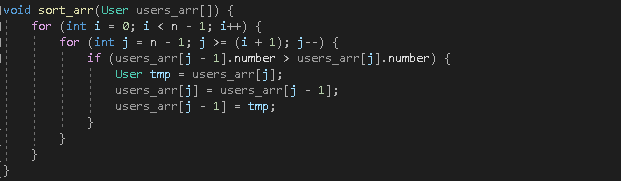
1. Объявляем длину получившейся строки int strlen и длину ключа int keylen, а также int res = -1, куда поместим номер пользователя. Длину будем считать с помощью метода size ().



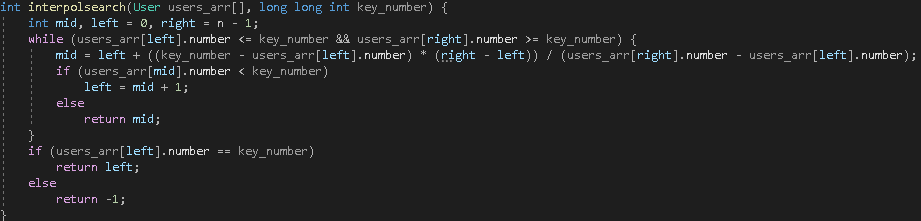
1. Далее сравниваем посимвольно ключ и строку, и если произошло несовпадение, то начинаем сравнивать со следующим символом строки, до тех пор, пока не найдется пользователь или не кончится строка.



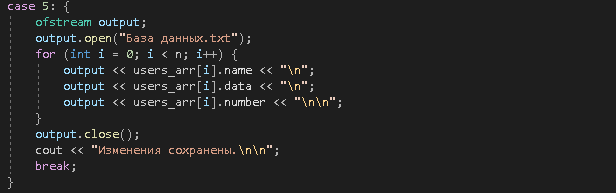
1. Для реализации интерполяционного поиска необходимо сначала отсортировать массив по номеру телефона (так как этот поиск работает только с целочисленными значениями). Сортировка будет осуществляться с помощью метода пузырька.



1. Вместо сравнения каждого элемента с искомым, как при линейном поиске, данный алгоритм производит предсказание местонахождения элемента: поиск происходит подобно двоичному поиску, но вместо деления области поиска на две части, интерполирующий поиск производит оценку новой области поиска по расстоянию между ключом и текущим значением элемента с помощью специальной формулы.



1. Для сохранения изменений в файл объявляем объект ofstream output, открываем файл для записи и с помощью цикла for и оператора вывода output заполняем его. В конце файл закрываем.



**Блок-схема**

**Код С++**

#include <iostream>

#include <string>

#include <cstdlib>

#include <locale.h>

#include <fstream>

#include <stdlib.h>

#include <sstream>

using namespace std;

int n;

string key;

struct User {

string name;

string data;

long long number;

};

User\* users\_arr;

void correct\_input(int& num) {

cin >> num;

while (num < 0) {

cout << "Некорректный ввод. Введите заново: ";

cin >> num;

}

}

void create(User users\_arr[], int& size) {

cout << "Заполните данные о пользователях: \n";

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

cout << "Введите имя пользователя: ";

cin.ignore();

getline(cin, users\_arr[i].name);

cout << "Введите дату рождения: ";

getline(cin, users\_arr[i].data);

cout << "Введите номер телефона: ";

cin >> users\_arr[i].number;

cout << endl;

}

}

void print() {

if (n == 0)

cout << "Пусто.\n";

else {

cout << endl;

cout << "Информация о пользователях:\n";

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

cout << "ФИО: " << users\_arr[i].name << endl;

cout << "Дата рождения: " << users\_arr[i].data << endl;

cout << "Номер телефона: " << users\_arr[i].number << endl;

cout << endl;

}

}

}

void add\_user() {

int amount\_of\_new\_users;

cout << "Сколько пользователей необходимо добавить: ";

cin >> amount\_of\_new\_users;

User\* array\_copy = new User[n + amount\_of\_new\_users];

create(array\_copy, amount\_of\_new\_users);

for (int i = 0; i < n; i++)

array\_copy[i + amount\_of\_new\_users] = users\_arr[i];

n += amount\_of\_new\_users;

users\_arr = array\_copy;

}

void delete\_user\_number(User users\_arr[], int k) {

int i = 0;

while (i < n) {

if (i == k) {

while (i < n - 1) {

users\_arr[i] = users\_arr[i + 1];

i++;

}

n--;

User\* array\_copy = new User[n];

for (int g = 0; g < n; g++)

array\_copy[g] = users\_arr[g];

users\_arr = array\_copy;

break;

}

i++;

}

}

void linesearch(User users\_arr[], string key) {

bool f = false;

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

if (users\_arr[i].name == key) {

cout << "Пользователь находится в списке под номером: " << i + 1 << endl;

f = true;

}

}

if (!f)

cout << "Пользователь не найден.";

}

void search(User users\_arr[], string key) {

string str;

for (int i = 1; i <= n; i++) {

str = str + to\_string(i) + users\_arr[i-1].name;

}

int strlen, keylen, res = -1;

strlen = str.size();

keylen = key.size();

if (strlen != 0 && keylen != 0) {

for (int i = 0; i < strlen - keylen + 1; i++) {

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

if (key[j] != str[i + j])

break;

else if (j == keylen - 1) {

res = i;

break;

}

}

}

}

if (res != -1)

cout << "Пользователь найден! Он находится в списке под номером " << str[res - 1] << endl;

else

cout << "Пользователь не найден." << endl;

}

void sort\_arr(User users\_arr[]) {

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

for (int j = n - 1; j >= (i + 1); j--) {

if (users\_arr[j - 1].number > users\_arr[j].number) {

User tmp = users\_arr[j];

users\_arr[j] = users\_arr[j - 1];

users\_arr[j - 1] = tmp;

}

}

}

}

int interpolsearch(User users\_arr[], long long int key\_number) {

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

while (users\_arr[left].number <= key\_number && users\_arr[right].number >= key\_number) {

mid = left + ((key\_number - users\_arr[left].number) \* (right - left)) / (users\_arr[right].number - users\_arr[left].number);

if (users\_arr[mid].number < key\_number)

left = mid + 1;

else

return mid;

}

if (users\_arr[left].number == key\_number)

return left;

else

return -1;

}

int main() {

setlocale(LC\_ALL, "rus");

system("chcp 1251");

int a;

cout << "Выберите действие:\n";

cout << "1. Открыть файл.\n";

cout << "2. Ввести список вручную.\n";

cin >> a;

if (a == 1) {

int count = 0;

string row;

ifstream input;

input.open("База данных.txt");

while (!input.eof()) {

count++;

getline(input, row);

}

n = count / 4;

users\_arr = new User[n];

input.close();

input.open("База данных.txt");

int i = 0;

while (!input.eof() && n - i > 0) {

getline(input, users\_arr[i].name);

getline(input, users\_arr[i].data);

getline(input, row);

long long int x;

istringstream(row) >> x;

users\_arr[i].number = x;

getline(input, row);

i++;

}

input.close();

cout << "Список загружен.\n\n";

}

else {

cout << "Введите количество пользователей: ";

correct\_input(n);

users\_arr = new User[n];

create(users\_arr, n);

print();

}

while (true) {

cout << "Выберите действие:\n";

cout << "1. Добавить пользователей в список.\n";

cout << "2. Удалить пользователя из списка.\n";

cout << "3. Найти пользователя в списке.\n";

cout << "4. Напечатать список.\n";

cout << "5. Сохранить изменения в файл.\n";

cout << "6. Выйти из программы.\n-->";

cin >> a;

switch (a) {

case 1: {

add\_user();

cout << "Пользователь добавлен.\n";

break;

}

case 2: {

int k;

cout << "Введите номер пользователя в списке:\n";

correct\_input(k);

delete\_user\_number(users\_arr, k);

cout << "Пользователь удален.\n";

break;

}

case 3: {

cout << "Каким методом искать:\n";

cout << "1. Линейный метод.\n";

cout << "2. Интерполяционный метод.\n";

cout << "3. Прямой поиск подстроки в строке.\n-->";

cin >> a;

if (a != 2) {

cout << "Введите ФИО пользователя: ";

cin.ignore();

getline(cin, key);

if (a == 1)

linesearch(users\_arr, key);

else

search(users\_arr, key);

}

else {

cout << "Введите номер пользователя:\n";

long long key\_number;

cin >> key\_number;

cout << "Сортируем список...\n";

sort\_arr(users\_arr);

int b = interpolsearch(users\_arr, key\_number);

if (b != -1)

cout << "Пользователь находится в списке под номером: " << b + 1 << endl;

else

cout << "Пользователь не найден." << endl;

}

break;

}

case 4:

print();

break;

case 5: {

ofstream output;

output.open("База данных.txt");

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

output << users\_arr[i].name << "\n";

output << users\_arr[i].data << "\n";

output << users\_arr[i].number << "\n\n";

}

output.close();

cout << "Изменения сохранены.\n\n";

break;

}

case 6:

exit(0);

default: {

cout << "Неверный номер действия.";

break;

}

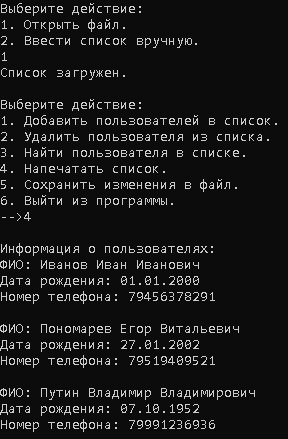
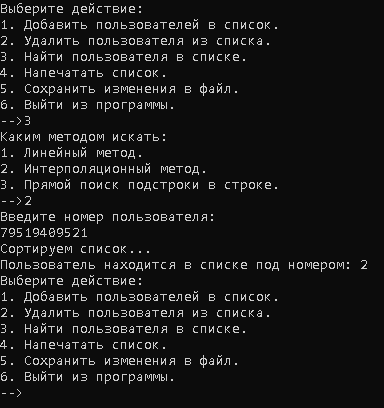
}

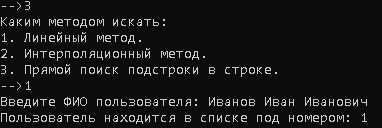
}

return 0;

}

**Результаты выполнения**

**** ****

****