Выполнил

студент КТбо1-4 \_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Воронов

Принял

ст. преподаватель каф. САПР \_\_\_\_\_\_\_\_\_\_\_\_\_ В. И. Данильченко

Таганрог 2023

ИНДИВИДУАЛЬНАЯ РАБОТА №1 (Вариант 3)

по дисциплине «Алгоритмизация и программирование»

**«Программирование на языке Си/С++»**

Министерство науки и высшего образования Российской Федерации

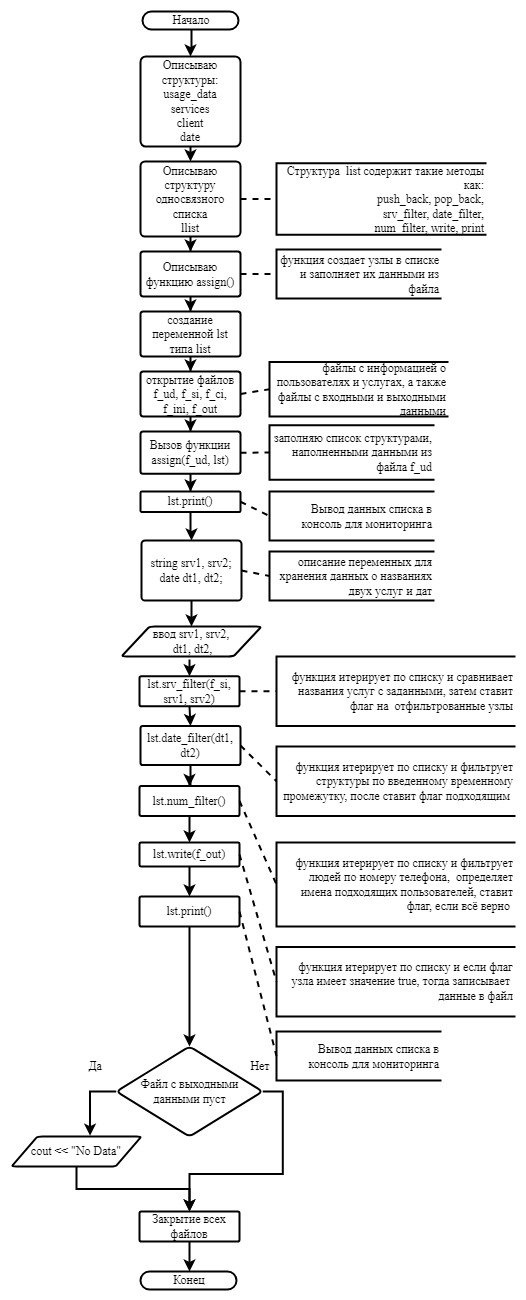
Федеральное государственное автономное образовательное учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

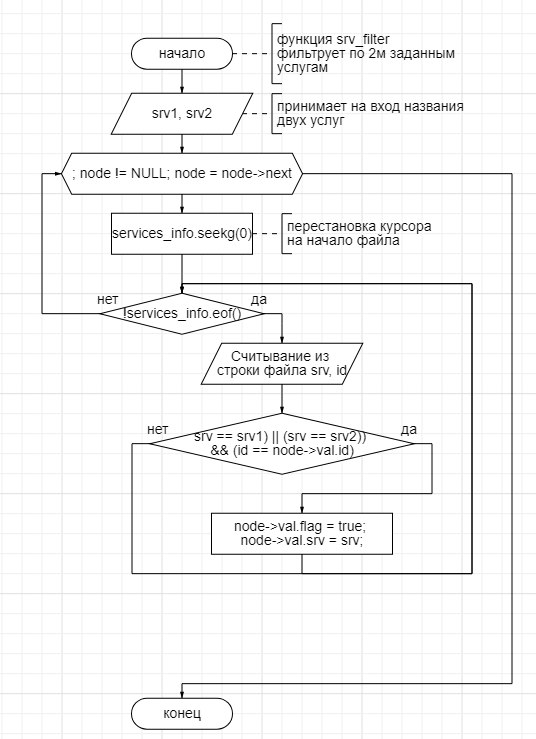
(ФГАОУ ВО «ЮФУ»)

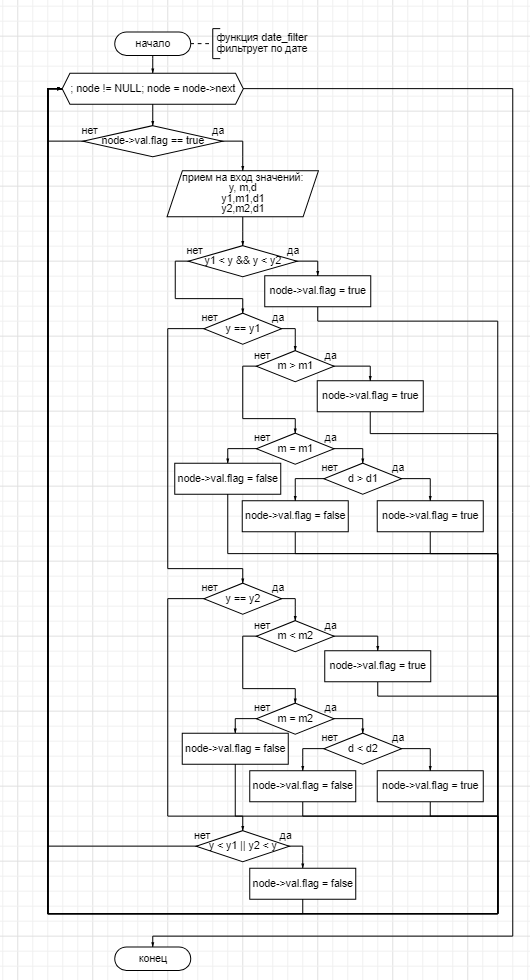
1. **Формулировка индивидуального задания:** задание состоит в разработке программы, которая считывает настроечные параметры и формирует отчет по имеющимся данным в файлах. Отчет записывается в текстовый файл. Имеется информация о клиентах телефонной компании и предоставляемых им услугах. Каждая услуга имеет собственный тариф, а каждый клиент может пользоваться произвольным набором услуг в течение ограниченного интервала времени (соответственно срокам договора). Имеются данные о фактическом использовании услуг. Вся указанная информация представлена текстовыми файлами. В варианте 3 требуется написать программу на языке Си / С++, которая будет составлять список клиентов, использовавших услуги двух наименований в указанный временной промежуток (с… по…).
2. **Описание алгоритма в виде блок-схемы:**

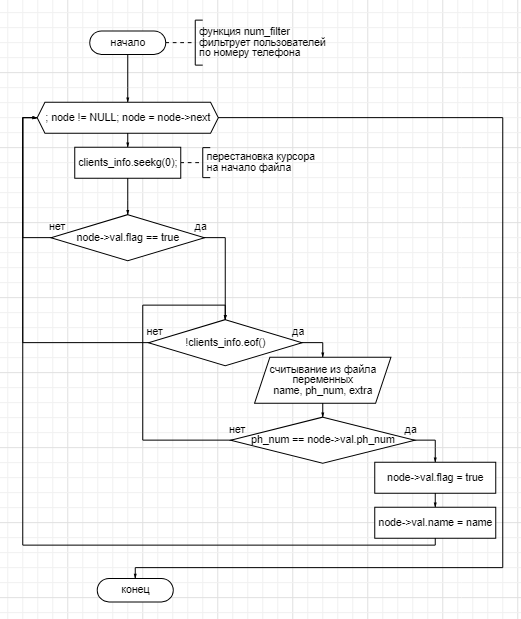
**Общая блок-схема алгоритма:**

****

**Блок-схемы ключевых функций:**

****

****

****

**3. Листинг программы**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct usage\_data {

string ph\_num, id, d, m, y, extra, name, srv;

bool flag = false;

void ud\_print() {

cout

<< name << "\t"

<< ph\_num << "\t"

<< id <<"\t"

<< flag <<"\t"

<< srv << "\n";

}

void ud\_write(fstream& file) {

file

<< name << "\t"

<< ph\_num << "\t"

<< srv << "\t"

<< d <<"."

<< m <<"."

<< y <<"\n";

}

};

struct services {string srv, id, extra; };

struct client {string name, ph\_num, extra;};

struct date {string d, m, y;};

struct Node {

usage\_data val;

Node\* next;

Node(usage\_data val) {

this->val = val;

this->next = NULL;

}

};

struct list {

Node\* first, \* last;

list() : first(nullptr), last(nullptr) {}

void push\_back(usage\_data data) { //добавление узла

Node\* node = new Node(data);

if (first == NULL) first = node;

if (last != NULL) last->next = node;

last = node;

}

void pop\_back() { //удаление узла

if (last == NULL) return;

if (first == last) {

delete last;

first = last = NULL;

return;

}

Node\* node = first;

for (; node->next != last; node = node->next);

node->next = NULL;

delete last;

last = node;

}

void srv\_filter(ifstream& services\_info, string srv1, string srv2) { //фильтр по названию услуг

Node\* node = first;

for (; node != NULL; node = node->next) {

services\_info.seekg(0);

while (!services\_info.eof()) {

services serv;

getline(services\_info, serv.srv, ',');

getline(services\_info, serv.id, ',');

getline(services\_info, serv.extra);

if (((serv.srv == srv1) || (serv.srv == srv2)) && (serv.id == node->val.id)) {

node->val.flag = true;

node->val.srv = serv.srv;

}

}

//node->val.ud\_print();

}

}

void date\_filter(date dt1, date dt2) { //фильтр по временному промежутку

Node\* node = first;

for (; node != NULL; node = node->next) {

if (node->val.flag == true) {

int y = stoi(node->val.y);

int m = stoi(node->val.m);

int d = stoi(node->val.d);

int y1 = stoi(dt1.y);

int m1 = stoi(dt1.m);

int d1 = stoi(dt1.d);

int y2 = stoi(dt2.y);

int m2 = stoi(dt2.m);

int d2 = stoi(dt2.d);

if(y1 < y && y < y2) node->val.flag = true;

if (y == y1) {

if (m > m1) node->val.flag = true;

if (m = m1) {

if (d>d1) node->val.flag = true;

else node->val.flag = false;

}

else node->val.flag = false;

}

if (y == y2) {

if (m < m2) node->val.flag = true;

if (m = m2) {

if (d < d2) node->val.flag = true;

else node->val.flag = false;

}

else node->val.flag = false;

}

if (y<y1 || y2<y) node->val.flag = false;

}

//node->val.ud\_print();

}

}

void num\_filter(ifstream& clients\_info) { //фильтр по номеру телефона, поиск фамилий

Node\* node = first;

client clnt;

for (; node != NULL; node = node->next) {

clients\_info.seekg(0);

if (node->val.flag == true) {

while (!clients\_info.eof()) {

getline(clients\_info, clnt.name, ',');

getline(clients\_info, clnt.ph\_num, ',');

getline(clients\_info, clnt.extra);

if (clnt.ph\_num == node->val.ph\_num) {

node->val.flag = true;

node->val.name = clnt.name;

}

}

}

}

}

void write(fstream& file) { //запись в файл

Node\* node = first;

for (; node != NULL; node = node->next)

if (node->val.flag == true)

node->val.ud\_write(file);

}

void print() { //вывод в консоль

Node\* node = first;

for (; node != NULL; node = node->next)

if (node->val.flag == true)

node->val.ud\_print();

}

};

void assign(ifstream& f\_ud, list& lst) { //f\_ud == file "usage\_data.txt"

while (!f\_ud.eof()) {

usage\_data x;

getline(f\_ud, x.ph\_num, ',');

getline(f\_ud, x.id, ',');

getline(f\_ud, x.d, '.');

getline(f\_ud, x.m, '.');

getline(f\_ud, x.y, ' ');

getline(f\_ud, x.extra);

lst.push\_back(x);

}

}

//-------------------------------------------------------------------------------

int main() {

list lst;

ifstream f\_ud, f\_si, f\_ci, f\_ini;

fstream f\_out;

f\_ud.open("usage\_data.txt");

f\_si.open("services\_info.txt");

f\_ci.open("clients\_info.txt");

f\_out.open("Report.txt", ios::out);

f\_ini.open("Param.ini");

assign(f\_ud, lst); //считал структуру и записал в список

string srv1, srv2;

date dt1, dt2;

getline(f\_ini, srv1,' ');

getline(f\_ini, srv2, ' ');

getline(f\_ini, dt1.d, '.');

getline(f\_ini, dt1.m, '.');

getline(f\_ini, dt1.y, ' ');

getline(f\_ini, dt2.d, '.');

getline(f\_ini, dt2.m, '.');

getline(f\_ini, dt2.y);

lst.srv\_filter(f\_si, srv1, srv2);

lst.date\_filter(dt1, dt2);

lst.num\_filter(f\_ci);

lst.write(f\_out);

lst.print();

if (f\_out.peek() == EOF) cout << "No Data\n"; //проверка на пустоту файла

f\_ud.close();

f\_si.close();

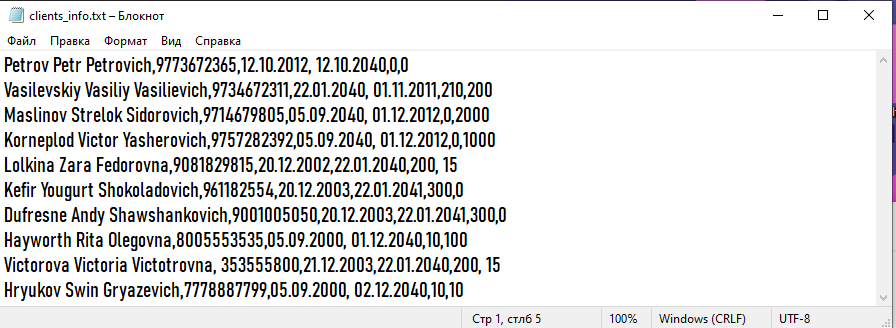
f\_ci.close();

f\_out.close();

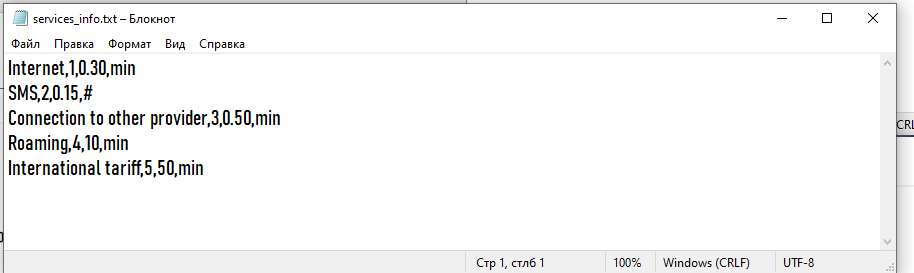
f\_ini.close();

}**4. Тексты файлов данных**

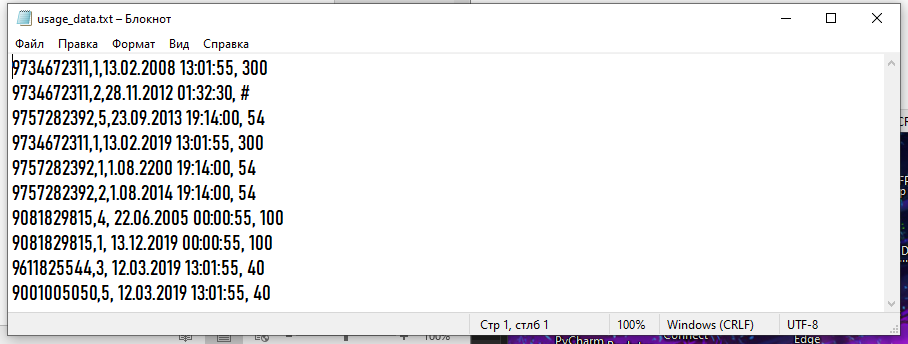
**Файл 1:**

****

**Файл 2:**

****

**Файл 3:**

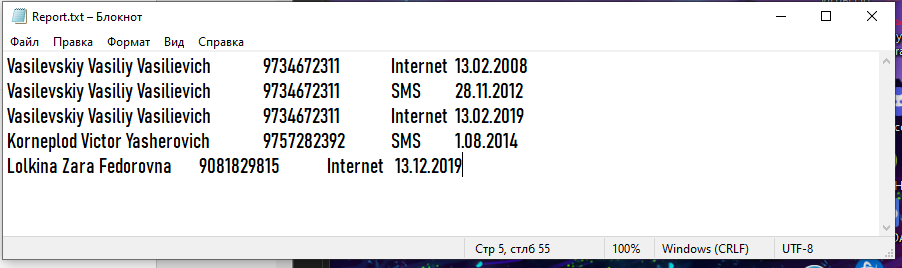
****

**5. Контрольный пример**

**Файл ввода:**

****

**Файл вывода:**

****

**Вывод консоли:**

