Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Иркутский национальный исследовательский технический университет»

|  |
| --- |
| Институт информационных технологий и анализа данных |
| (наименование) |

**О Т Ч Ё Т**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| о прохождении | | | | учебной практики |
|  | | | (вид практики: учебная/производственная) | |
| технологической (проектно-технологической) | | | | |
|  | | | | |
| (тип практики: технологическая/научно-исследовательская работа/преддипломная и др.) | | | | |
| в | | институте информационных технологий и анализа данных | | |
|  | (наименование профильной организации) | | | |

|  |  |  |
| --- | --- | --- |
| Обучающегося | Игнатьева Д.С, ИСТБ23-3 | |
|  | (ФИО, группа, подпись) | |
| Руководитель практики | | |
| Маланова Т.В, доцент | | |
| ( ФИО, должность, подпись) | | |
| Допущен к защите | | |
| Маланова Т.В, | | |
| (ФИО, подпись, дата) | | |
| Оценка по практике | |  |
| , | | |
| (ФИО, подпись, дата) | | |

|  |  |  |
| --- | --- | --- |
| Содержание отчета на |  | стр. |
|  |  |  |

Иркутск 2024

**Индивидуальное задание на прохождение**

**учебной практики: технологическая (проектно-технологическая) практика**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| для | Игнатьева Дениса Сергеевича | | | | |
|  | (ФИО обучающегося полностью) | | | | |
| обучающегося | | 1 | курса | группы | ИСТб-23-3 |

по направлению подготовки Информационные системы и технологии

профиль Информационные системы и технологии в административном управлении

Место прохождения практики: Институт ИТиАД

Сроки прохождения практики с «17» июня 2024 г. по « 30 » июня 2024 г.

Цели и задачи прохождения практики: приобретение опыта осуществления постановки задачи, алгоритмизации, тестирования и программирования на языке высокого уровня, применения инструментов для отладки, а также оформления отчётной документации.

Содержание практики, вопросы, подлежащие изучению: методы тестирования алгоритмов и программ (методы «черный ящик», «белый ящик»), использование распределенной системы контроля версий git, оценка вычислительной сложности алгоритмов.

Планируемые результаты практики: программа, реализующая решение задачи из индивидуального задания, репозиторий с исходным кодом, набор тестов и описание стратегии тестирования алгоритма решения задачи, результаты анализа вычислительной сложности реализованного алгоритма, отчёт по практике, оформленный в соответствии с требованиями СТО ИРНИТУ.

Руководитель практики от института ИТиАД

\_\_\_\_\_\_\_\_\_\_\_/ Иванов А.С. /

(подпись)

**Согласовано:**

Руководитель ООП

\_\_\_\_\_\_\_\_\_\_\_/Маланова Т.В./

(подпись)

«17» июня 2024 г.

С настоящим индивидуальным заданием и с программой практики ознакомлен(а), задание принято к исполнению

\_\_\_\_\_\_\_\_\_\_«17 » июня 2024 г.

(подпись)

Введение

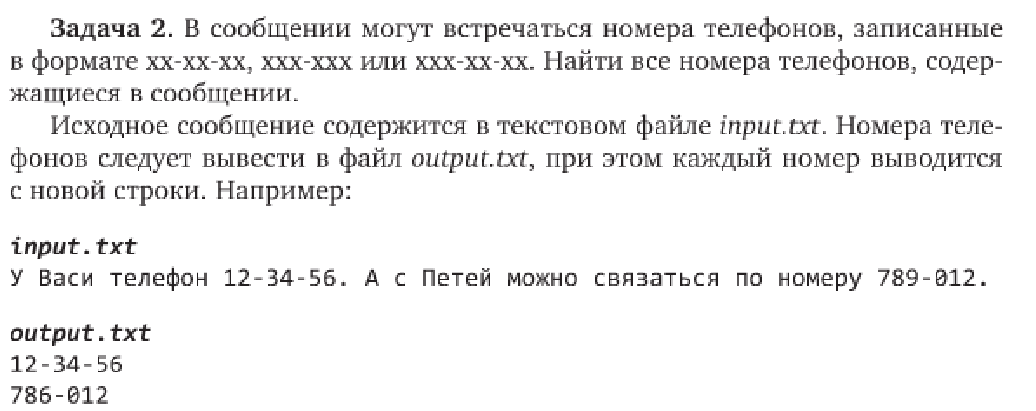
Целью практики является совершенствование навыков алгоритмизации, тестирования и программирования на языке высокого уровня, совершенствование навыков работы с текстовыми файлами в C++, а также закрепление знаний, полученных при изучении дисциплин «Информатика» и «Программирование».

Основными задачами учебной практики являются приобретение практических навыков:

* рационального использования рабочего времени;
* культуры труда;
* разработки алгоритмов решения задач;
* использования различных методов тестирования алгоритмов;
* применения языка программирования высокого уровня;
* качественного выполнения заданий;

межличностных отношений при коллективном выполнении задания.

1 Постановка задачи



Также нужно написать функцию для автоматического редактирования текста, исходящего от знаков препинания (точек, восклицательных и вопросительных знаков). Программа должна иметь функцию, которая различает и мобильные телефоны из разных стран, вывод страны, откуда этот номер.

Помимо решения задачи, программа должна иметь понятное меню, проверку ввода выбора в меню(только заданный диапазон чисел и запрет ввода букв), возможность сохранения и чтения файлов.

2.Таблица спецификаций

Таблица 1 – таблица спецификации глобальных переменных

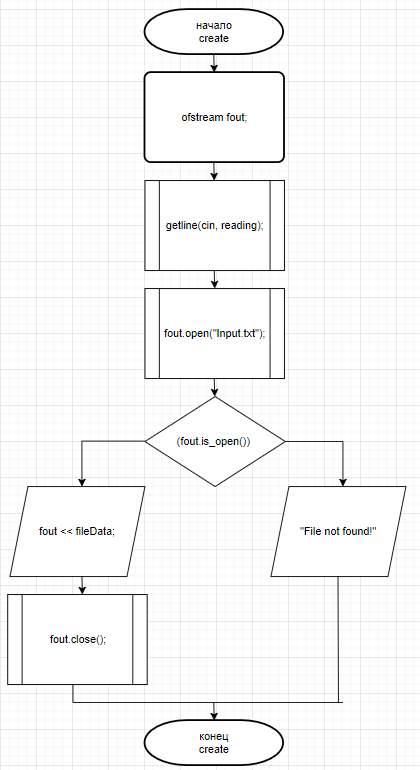
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Имя** | **Назначение** | **Тип** | **ОДЗ** |
| **1** | **filename** | **Имя файла при сохранении** | **string** | **-** |
| **2** | **numbers** | **Строка, в которую записываются номера телефонов из файла** | **string** | **-** |
| **4**  **3** | **inputFilename** | **Имя файла при** | **string** | **-** |
| **4** | **s** | **Строка для чтения файла, если он пересохранялся** | **string** | **-** |
| **5** | **output** | **Используется при чтении файла при получении номеров** | **string** | **-** |
| **6** | **reading** | **Строка в с помощью которой читается файл input.txt** | **String** | **-** |
| **7** | **renamed** | **Флаг переименования файла с номерами** | **Bool** | **-** |
| **8** | **renamedData** | **Флаг переименования файла input.txt** | **Bool** | **-** |
| **9** | **findNumbers** | **Флаг того, что в файле найдены номера телефонов** | **Bool** | **-** |
| **10** | **registr** | **Массив с позициями, в которых менялся регистр** | **Int** | **-** |

Таблица 2 – таблица спецификации функций.

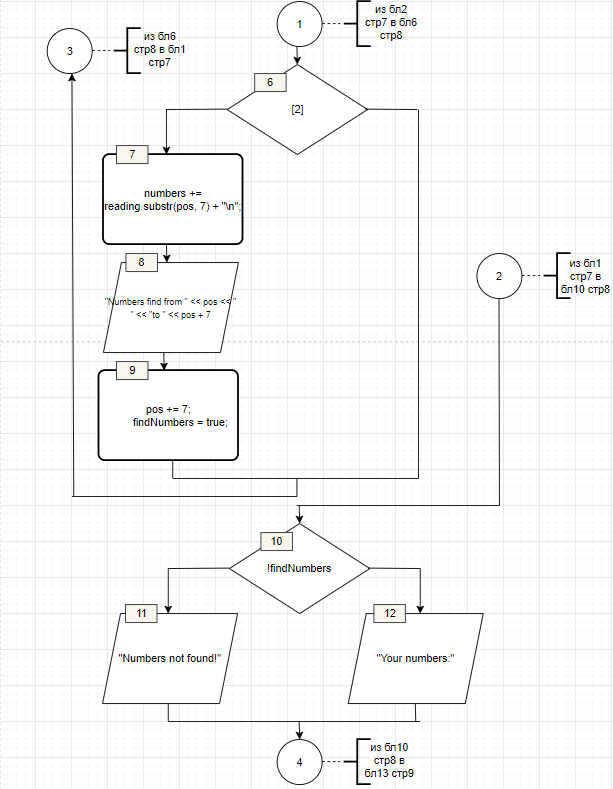
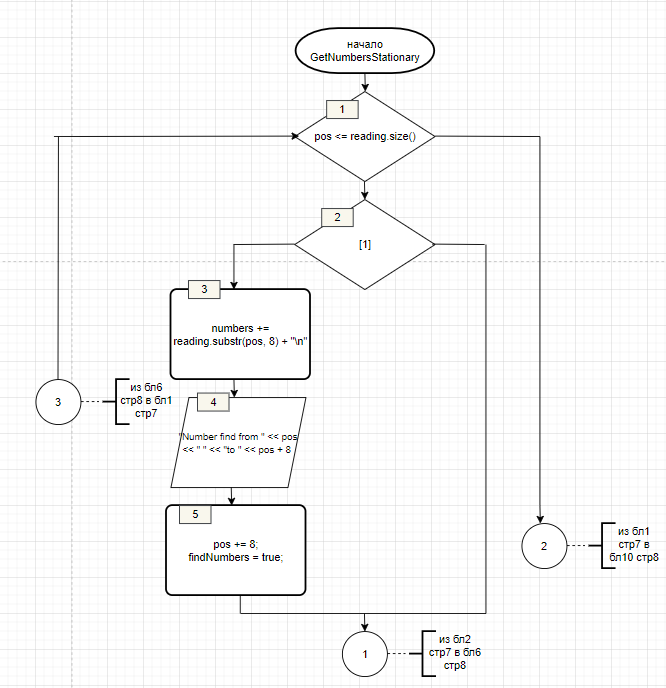
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | тип | Параметры |
| 1 | read | Чтение файла | void | - |
| 2 | create | Создание текстового файла | void | - |
| 3 | GetNumbersStationary | Создание файла с номерами телефонов | void | - |
| 4 | SaveNumbersAs | Сохранение файла с номерами как | void | - |
| 5 | SaveInputAs | Сохранение текстового файла | void | - |
| 6 | getValidatedChoose | Проверка ввод в меню | int | - |
| 7 | registrChecking | Проверка текста | void | - |
| 8 | mobilePhone | Запись в файл мобильных телефонов | void | - |

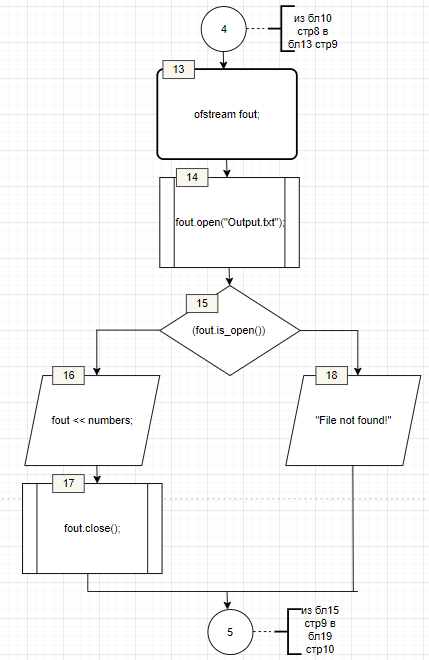
3. Блок схемы

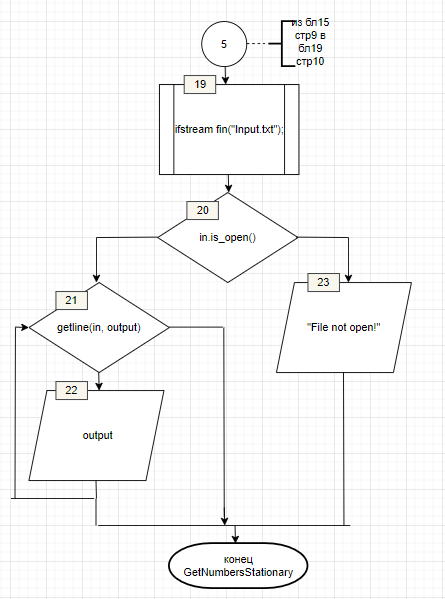
Блок схема 1 – блок схема функции create



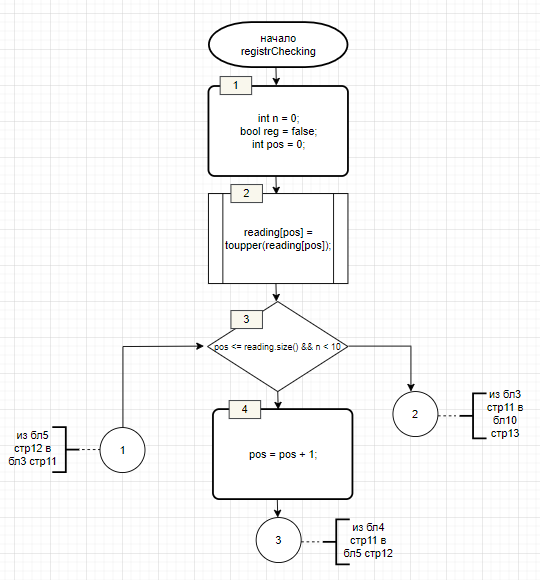
Блок схема 2 – блок схема функции GetNumbers

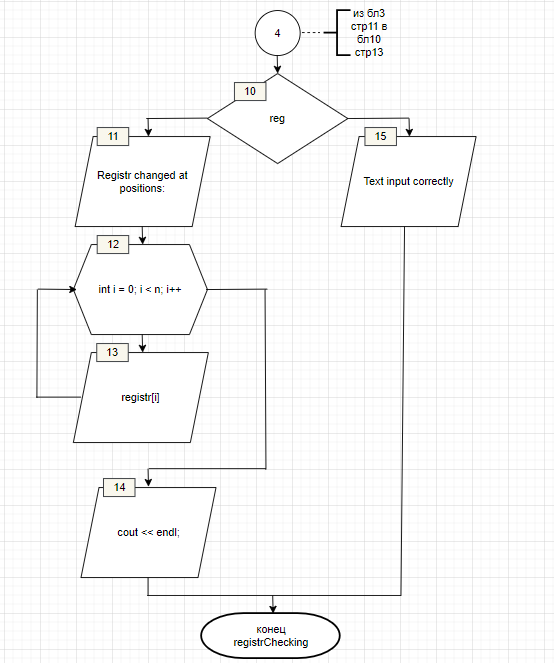
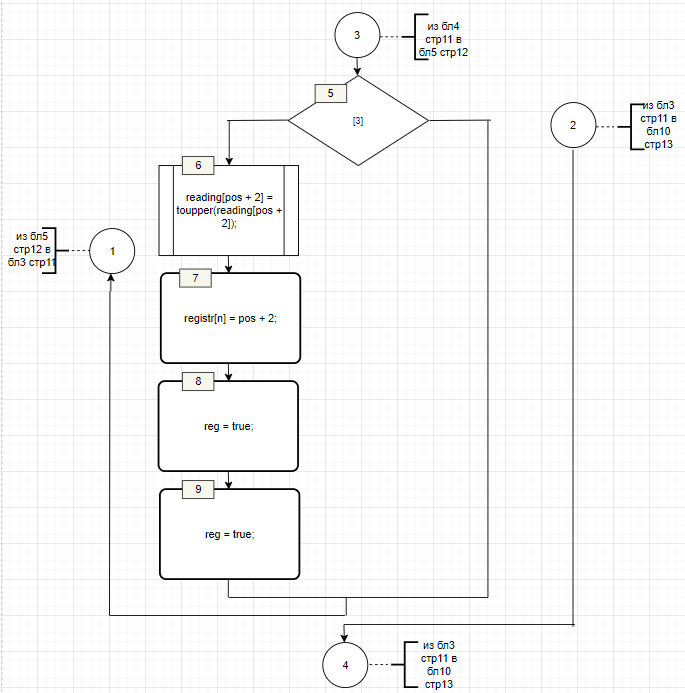






Блок схема 3 – функция изменения регистра





Блок схема 4 – функция получения мобильных телефонов

4.Таблицы тестов

В соответствии с рекомендованной стратегией тестирования были определены правильные и неправильные классы эквивалентности для входных данных на основе анализа постановки задачи (см. таблицу 2).

Таблица 2 – Выделенные классы эквивалентности

|  |  |  |
| --- | --- | --- |
| № | Правильные классы эквивалентности | Неправильные классы эквивалентности |
| 1 | y=int | y=char |
| 2 | 1<=y<=7 | y>7 |
| 3 | Строка имеет стационарный номер | Строка не имеет стационарного номера |
| 4 | Строка имеет мобильный телефон | Строка не имеет мобильный телефон |
| 5 | Строка с неправильными регистрами | Строка с правильными регистрами |

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Выходные данные | Результат |
| 1 | y=1 | Ввод данных в файл | Тест пройден |
| 2 | y=ch | Неправильный ввод | Тест пройден |
| 3 | y=2 | Вывод данных из файла | Тест пройден |
| 4 | y=9 | Неправильный ввод | Тест пройден |
| 5 | Reading =”hospital has 12-34-56 number” | 12-34-56 | Тест пройден |
| 6 | Reading =”hospital has number” | Номера не найдены | Тест пройден |
| 7 | Reading=”8(950)3877885 – number of vova” | 8(950)3877885 –number from Russian | Тест пройден |
| 8 | Reading=”number of vova” | Номера не найдены | Тест пройден |
| 9 | Reading =”privet. kak dela” | Privet. Kak dela | Тест пройден |

5.Исходный код

#include <string>

#include <fstream>

#include <iostream>

using namespace std;

string filename;

string numbers;

string inputFilename;

string s;

string output;

string reading;

bool renamed = false;

bool renamedData = false;

bool findNumbers = false;

bool edit = false;

int registr[10] = { 0 };

void Read() {

if (!renamedData) {

ifstream fin("Input.txt");

if (fin.is\_open()) {

while (getline(fin, reading)) {

cout << reading << endl;

}

}

else {

cout << "File not open!" << endl;

}

}

else {

ifstream fin(inputFilename + ".txt");

if (fin.is\_open()) {

while (getline(fin, s)) {

cout << s << endl;

}

}

else {

cout << "File not open!" << endl;

}

}

}

void create() {

ofstream fout;

getline(cin, reading);

fout.open("Input.txt");

if (fout.is\_open()) {

fout << reading;

fout.close();

}

else {

cout << "File not found!" << endl;

}

}

void GetNumbersStationary() {

int pos = 0;

while (pos <= reading.size()) {

if ((reading[pos] >= '0' && reading[pos] <= '9') && (reading[pos + 1] >= '0' && reading[pos + 1] <= '9')

&& (reading[pos + 2] == '-') && (reading[pos + 3] >= '0' && reading[pos + 3] <= '9') && (reading[pos + 4] >= '0' && reading[pos + 4] <= '9') && (reading[pos + 5] == '-')

&& (reading[pos + 6] >= '0' && reading[pos + 6] <= '9') && (reading[pos + 7] >= '0' && reading[pos + 7] <= '9')) {

numbers += reading.substr(pos, 8) + "\n";

cout << "Number find from " << pos << " " << "to " << pos + 8 << endl;

pos += 8;

findNumbers = true;

}

else if ((reading[pos] >= '0' && reading[pos] <= '9') && (reading[pos + 1] >= '0' && reading[pos + 1] <= '9') &&

(reading[pos + 2] >= '0' && reading[pos + 2] <= '9') && (reading[pos + 3] == '-') &&

(reading[pos + 4] >= '0' && reading[pos + 4] <= '9') && (reading[pos + 5] >= '0' && reading[pos + 5] <= '9') &&

(reading[pos + 6] >= '0' && reading[pos + 6] <= '9')) {

numbers += reading.substr(pos, 7) + "\n";

cout << "Numbers find from " << pos << " " << "to " << pos + 7 << endl;

pos += 7;

findNumbers = true;

}

pos = pos + 1;

}

if (!findNumbers) {

cout << "Stationary numbers not found!" << endl;

}

else {

cout << "Your stationary numbers:" << endl;

}

ofstream fout;

fout.open("Output.txt");

if (fout.is\_open()) {

fout << numbers;

fout.close();

}

else {

cout << "File is not created!" << endl;

}

ifstream in("Output.txt");

if (in.is\_open()) {

while (getline(in, output)) {

cout << output << endl;

}

}

else {

cout << "File is not open!" << endl;

}

}

void SaveNumbersAs() {

cout << "Enter new name of file: ";

cin >> filename;

ofstream fout;

fout.open(filename + ".txt");

if (fout.is\_open()) {

fout << numbers;

fout.close();

cout << "File resaved" << endl;

}

else {

cout << "File not open!" << endl;

}

}

void saveInputAs() {

cout << "Enter new name of file: ";

cin >> inputFilename;

ofstream fout;

fout.open(inputFilename + ".txt");

if (fout.is\_open()) {

fout << reading;

fout.close();

}

else {

cout << "File not open!" << endl;

}

renamedData = true;

}

int getValidatedChoose() {

string input;

while (true) {

cout << "Input choose: ";

getline(cin, input);

bool isValid = true;

for (char c : input) {

if (!isdigit(c)) {

isValid = false;

break;

}

}

if (isValid && !input.empty()) {

return stoi(input);

}

else {

cout << "Invalid input. Please enter a valid choose" << endl;

}

}

}

void registrChecking() {

int n = 0;

bool reg = false;

int pos = 0;

reading[pos] = toupper(reading[pos]);

while (pos <= reading.size() && n < 10) {

if (((reading[pos] == '.') || (reading[pos] == '!') || (reading[pos] == '?')) && (pos + 2 <= reading.size())) {

reading[pos + 2] = toupper(reading[pos + 2]);

registr[n] = pos + 2;

reg = true;

n = n + 1;

}

pos = pos + 1;

}

if (reg) {

cout << "Registr changed at positions: ";

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

cout << registr[i] << ", ";

}

cout << endl;

}

else {

cout << "Text input correctly" << endl;

}

ofstream fout;

fout.open("Input.txt");

if (fout.is\_open()) {

fout << reading;

fout.close();

}

else {

cout << "File not found!" << endl;

}

}

void mobilePhone() {

string mobile;

int i = 0;

int a = 0;

int b = 0;

int pos = 0;

while (pos <= reading.size()) {

if ((reading[pos] == '8') && (reading[pos + 1] == '(') && (reading[pos + 2] >= '0' && reading[pos + 2] <= '9')

&& (reading[pos + 3] >= '0' && reading[pos + 3] <= '9') && (reading[pos + 4] >= '0' && reading[pos + 4] <= '9')

&& (reading[pos + 5] == ')') && (reading[pos + 6] >= '0' && reading[pos + 6] <= '9') && (reading[pos + 7] >= '0' && reading[pos + 7] <= '9')

&& (reading[pos + 8] >= '0' && reading[pos + 8] <= '9') && (reading[pos + 9] >= '0' && reading[pos + 9] <= '9')

&& (reading[pos + 10] >= '0' && reading[pos + 10] <= '9') && (reading[pos + 11] >= '0' && reading[pos + 11] <= '9')

&& (reading[pos + 12] >= '0' && reading[pos + 12] <= '9')) {

mobile += reading.substr(pos, 13) + " - Number from Russia" + "\n";

cout << "Number find from " << pos << " " << "to " << pos + 13 << endl;

pos += 13;

findNumbers = true;

}

if ((reading[pos] == '+') && (reading[pos + 1] == '7') && (reading[pos + 2] == '(')

&& (reading[pos + 3] >= '0' && reading[pos + 3] <= '9') && (reading[pos + 4] >= '0' && reading[pos + 4] <= '9')

&& (reading[pos + 5] >= '0' && reading[pos + 5] <= '9') && (reading[pos + 6] == ')') && (reading[pos + 7] >= '0' && reading[pos + 7] <= '9')

&& (reading[pos + 8] >= '0' && reading[pos + 8] <= '9') && (reading[pos + 9] >= '0' && reading[pos + 9] <= '9')

&& (reading[pos + 10] >= '0' && reading[pos + 10] <= '9') && (reading[pos + 11] >= '0' && reading[pos + 11] <= '9')

&& (reading[pos + 12] >= '0' && reading[pos + 12] <= '9') && (reading[pos + 13] >= '0' && reading[pos + 13] <= '9')) {

mobile += reading.substr(pos, 14) +" - Number from Russia" + "\n";

cout << "Numbers find from " << pos << " " << "to " << pos + 14 << endl;

pos += 14;

findNumbers = true;

}

if ((reading[pos] == '+') && (reading[pos + 1] == '0') && (reading[pos + 2] == '(')

&& (reading[pos + 3] >= '0' && reading[pos + 3] <= '9') && (reading[pos + 4] >= '0' && reading[pos + 4] <= '9')

&& (reading[pos + 5] >= '0' && reading[pos + 5] <= '9') && (reading[pos + 6] == ')') && (reading[pos + 7] >= '0' && reading[pos + 7] <= '9')

&& (reading[pos + 8] >= '0' && reading[pos + 8] <= '9') && (reading[pos + 9] >= '0' && reading[pos + 9] <= '9')

&& (reading[pos + 10] >= '0' && reading[pos + 10] <= '9') && (reading[pos + 11] >= '0' && reading[pos + 11] <= '9')

&& (reading[pos + 12] >= '0' && reading[pos + 12] <= '9') && (reading[pos + 13] >= '0' && reading[pos + 13] <= '9')) {

mobile += reading.substr(pos, 14) + " - Number from Birma"+"\n";

cout << "Numbers find from " << pos << " " << "to " << pos + 14 << endl;

pos += 14;

findNumbers = true;

}

if ((reading[pos] == '+') && (reading[pos + 1] == '1') && (reading[pos + 2] == '(')

&& (reading[pos + 3] >= '0' && reading[pos + 3] <= '9') && (reading[pos + 4] >= '0' && reading[pos + 4] <= '9')

&& (reading[pos + 5] >= '0' && reading[pos + 5] <= '9') && (reading[pos + 6] == ')') && (reading[pos + 7] >= '0' && reading[pos + 7] <= '9')

&& (reading[pos + 8] >= '0' && reading[pos + 8] <= '9') && (reading[pos + 9] >= '0' && reading[pos + 9] <= '9')

&& (reading[pos + 10] >= '0' && reading[pos + 10] <= '9') && (reading[pos + 11] >= '0' && reading[pos + 11] <= '9')

&& (reading[pos + 12] >= '0' && reading[pos + 12] <= '9') && (reading[pos + 13] >= '0' && reading[pos + 13] <= '9')) {

mobile += reading.substr(pos, 14) +" - Number from USA" +"\n";

cout << "Numbers find from " << pos << " " << "to " << pos + 14 << endl;

pos += 14;

findNumbers = true;

}

pos = pos + 1;

}

if (!findNumbers) {

cout << "Mobile numbers not found!" << endl;

}

else {

cout << "Your mobile numbers:" << endl;

}

ofstream fout;

fout.open("Output.txt");

if (fout.is\_open()) {

fout << mobile;

fout.close();

}

else {

cout << "File is not created!" << endl;

}

ifstream in("Output.txt");

if (in.is\_open()) {

while (getline(in, output)) {

cout << output << endl;

}

}

else {

cout << "File is not open!" << endl;

}

}

int main()

{

int y;

do {

cout << "Create a file(1)" << endl;

cout << "Read file(2)" << endl;

cout << "Get stationary numbers(3)" << endl;

cout << "Save numbers as(4)" << endl;

cout << "Save input as(5)" << endl;

cout << "Check text(6)" << endl;

cout << "Get mobile phones(7)" << endl;

cout << "Exit(7) " << endl;

y = getValidatedChoose();

switch (y) {

case 1:

cout << "Your text: ";

create();

break;

case 2:

cout << "TEXT FROM FILE:" << endl;

Read();

break;

case 3:

GetNumbersStationary();

mobilePhone();

break;

case 4:

SaveNumbersAs();

break;

case 5:

saveInputAs();

break;

case 6:

registrChecking();

break;

}

} while (y != 7);

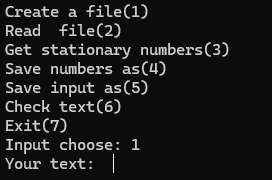
return 0;

}

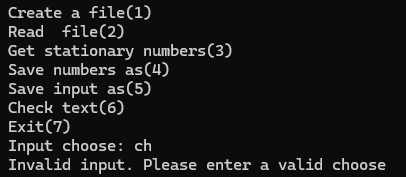
6.Анализ вычислительной сложности алгоритма

7. Результаты тестирования

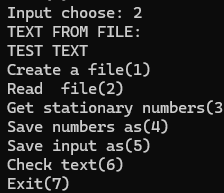
Тест 1.



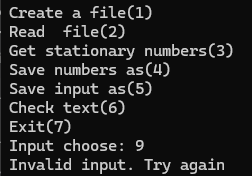
Тест 2



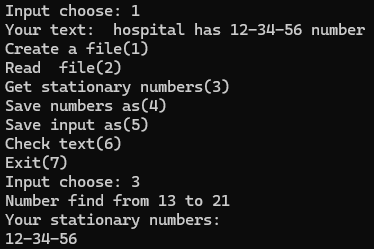
Тест 3



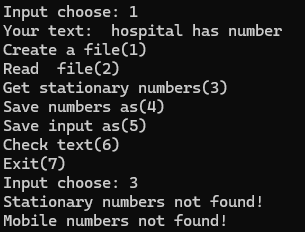
Тест 4



Тест 5

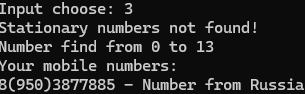


Тест 6



Тест 7





Тест 8





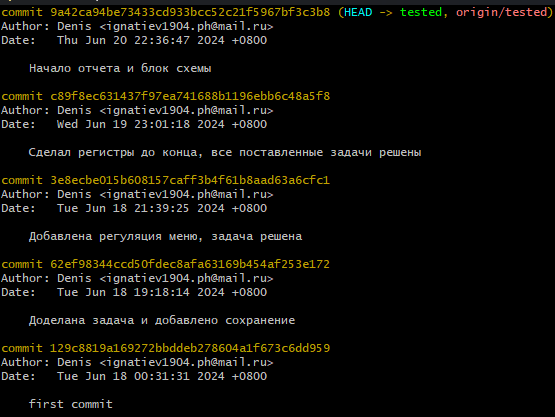
Тест 9





8.Организация работы над проектом

Вся работы над проектом организована одним студентом группы ИСТБ23-3 – Игнатьевым Денисом. В работе использовались многие инструменты: IDE – mvs, блок схемы создавались с помощью сервиса draw.io, контроль версий проводился с помощью локального репозитория на git, удаленного на github.

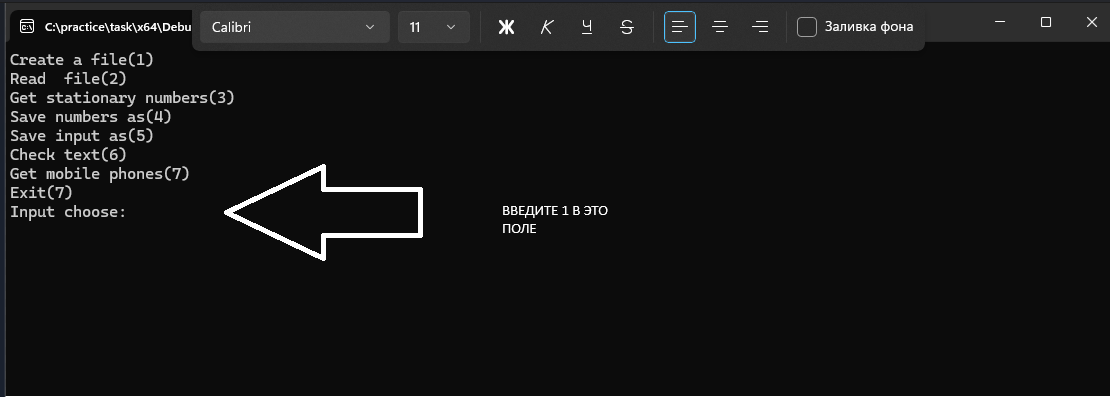


Приложение выполняет свои функции, имеет понятный консольный интерфейс.

9.Руководство пользователя

1.Создание файла input.txt.

Ввести в консоль 1, далее ввести текст в файлы, для выхода из функции нажмите enter.



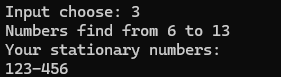
2.Чтение файла

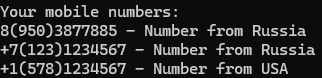
Введите в консоль 2, чтобы считать текст из файла.



3. Получение стационарных номеров

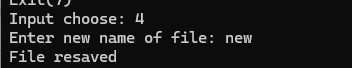
После ввода текста введите в консоль 3, тогда в консоль выведутся стационарные и мобильные номера и позиции в тексте, где эти номера находились





4.Сохранение номеров

Вы можете сохранить номера в другой файл, после ввода в консоль. Введите новое имя файла и номера телефона сохранятся в новый файл.



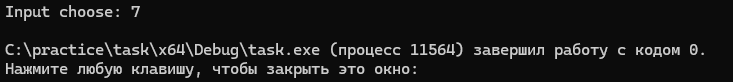
5.Проверка текста

После ввода в консоль цифры 6 вы исправите в исходном тексте регистр букв, если вы где-то упустили большую букву после знака препинания.



6.Выход

Для выхода введите в консоль 7.



Заключение