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

**О Т Ч Ё Т**

|  |  |
| --- | --- |
| о прохождении | учебной практики |
|  | (вид практики: учебная/производственная) |
| технологическая (проектно-технологическая) практика | |
| (тип практики: технологическая/научно-исследовательская работа/преддипломная и др.) | |
|  | |
| в | Институте ИТиАД |
|  | (наименование профильной организации) |

Обучающихся

Ананич А.П. гр. ИСТб-22-2

(ФИО, группа, подпись)

Руководитель практики от института ИТиАД

Осипова Е.А., доцент ЦПИ

(ФИО, должность, подпись)

Руководитель образовательной

программы

Аршинский В.Л., доцент ЦПИ

(ФИО, должность, подпись)

Оценка по практике

Осипова Е.А., доцент ЦПИ

(ФИО, подпись, дата)

Содержание отчета на 43 стр. Приложение к отчету на - стр.

Иркутск 2024

**Индивидуальное задание на прохождение**

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| для | Ананича Алексея Петровича | | | | |
|  | (ФИО обучающегося полностью) | | | | |
| обучающегося | | 1 | курса | группы | ИСТб-22-2 |

по направлению подготовки Информационные системы и технологии

профиль Информационные системы и технологии в административном управлении

Место прохождения практики: Институт ИТиАД

Сроки прохождения практики с «19» июня 2023 г. по «2» июля 2023 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

\_\_\_\_\_\_\_\_\_\_\_/ Осипова Е.А. /

(подпись)

**Согласовано:**

Руководитель ООП

\_\_\_\_\_\_\_\_\_\_\_/Аршинский В.Л./

(подпись)

«19» июня 2023 г.

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

\_\_\_\_\_\_\_\_\_\_ «19» июня 2023 г.

(подпись)

**Содержание**

[Введение 4](#_Toc139013352)

[Постановка задачи 5](#_Toc139013353)

[Таблица спецификаций 6](#_Toc139013354)

[Словесное описание основной программы 8](#_Toc139013355)

[Схема иерархии функций 9](#_Toc139013356)

[Блок схемы 10](#_Toc139013357)

[Листинг 21](#_Toc139013358)

[Таблица тестов 28](#_Toc139013359)

[Анализ вычислительной сложности алгоритма 31](#_Toc139013360)

[Результаты тестирования 32](#_Toc139013361)

[Организация работ над проектом 35](#_Toc139013362)

[Заключение 36](#_Toc139013363)

[Список литературы 37](#_Toc139013364)

Введение

Целью практики является совершенствование навыков алгоритмизации, тестирования и программирования на языке высокого уровня, а также закрепление знаний, полученных при изучении дисциплин «Информатика» и «Программирование на языке высокого уровня».

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

## Постановка задачи

Решение СЛАУ методом Камера

* Консольное приложение (интерфейс в виде текстового диалога)
* Работа с динамическими массивами
* При создании СЛАУ пользователь может записать его в ручную а, может создать случайную автоматически.
* Пользователь может запросить значение определителя основной матрицы.
* Программа завершает работу только если пользователь нажмёт выход.

## Таблица спецификаций

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Имя** | **Назначение** | **Тип** | **ОДЗ** |
| **PrintMatr – функция вывода СЛАУ** | | | | |
| 1 | i | Счётчик строк | int | Диапазон int |
| 2 | j | Счётчик столбцов | int | Диапазон int |
| **GetMatr – функция получение матрицы без i-й строки и j-го столбца** | | | | |
| 1 | ki | Счётчик пропускающий i строку | int | Диапазон int |
| 2 | kj | Счётчик пропускающий j столбец | int | Диапазон int |
| **Determinant – функция получения детерминанта** | | | | |
| 1 | det | Детерминант | double | Диапазон double |
| **void addRecord – функция добавления новой записи** | | | | |
| 1 | website | вебсайт | string | - |
| 2 | username | Имя пользователя | string | - |
| 3 | password | пароль | string | - |
| 4 | length | Длина пароля | int | Диапазон int |
| 5 | record | Вспомогательная переменная | string | - |
| 6 | temp | Добавляемая запись | string | - |
| **void viewRecords – функция просмотра всех записей** | | | | |
|  | | | | |
| **void editRecord – функция редактирования записи** | | | | |
| 1 | index | Номер записи | int | Диапазон int |
| 2 | website | вебсайт | int | Диапазон int |
| 3 | username | Имя пользователя | int | Диапазон int |
| 4 | password | пароль | int | Диапазон int |
| **void deleteRecord – функция удаления записи** | | | | |
| 1 | index | Номер записи | int | Диапазон int |
| 2 | j | счетчик | int | Диапазон int |
| 3 | temp | Удаляемая запись | string | - |
| **void viewPassword – функция просмотра пароля** | | | | |
| 1 | index | Номер записи | int | Диапазон int |
| **void saveToFile – функция сохранения записей в файл** | | | | |
|  | | | | |
| **void loadFromFile – функция загрузки записей из файла** | | | | |
| 1 | password | Мастер-пароль | string | - |
| 2 | n | Количество записей | int | Диапазон int |
| **Int main** | | | | |
| 1 | filename | Имя файла | string | - |
| 2 | masterPassword | Мастер-пароль | string | - |
| 3 | choise | Пункт меню | int | [1; 8] |

Таблица 1 – таблица спецификаций.

## Словесное описание основной программы

1. Генератор паролей:
   1. Вводим длину пароля length;
   2. Объявляем переменную password;
   3. Объявляем алфавит symbols;
   4. I = 0; пока I < длины length
      1. Прибавлять к password случайный элемент из алфавита symbols;
      2. I + 1;
   5. Выводим password.
2. Шифратор:
   1. Вводим слово data;
   2. Вводим мастер + пароль key
   3. I = 0; пока I < длины data
      1. К каждой букве data прибавляем key;
   4. Выводим data.
3. Дешифратор:
   1. Вводим слово data;
   2. Вводим мастер - пароль key
   3. I = 0; пока I < длины data
      1. От каждой буквы data отнимаем key;
4. Выводим data.

## Схема иерархии функций

На рисунке 1 представлена схема иерархии функций для первого способа решения задачи.



Рисунок 1 – схема иерархии функций.

## Блок схемы

**# generatePassword**

На рисунке 2 показана блок-схема функции генерации пароля.

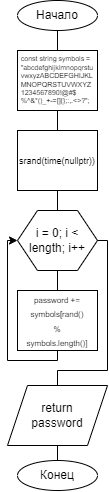


Рисунок 2 – Блок-схема функции генерации пароля.

**# viewRecords**

На рисунке 3 показана блок-схема функции показа всех записей.

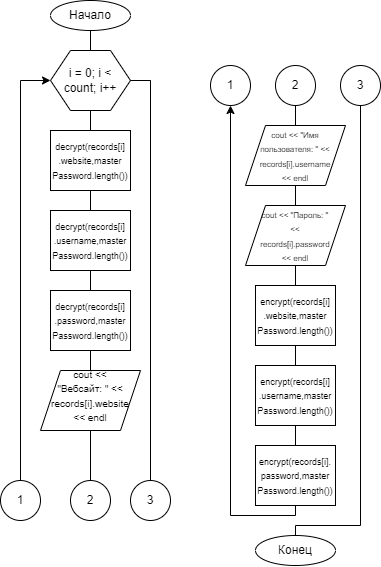


Рисунок 3 – Блок-схема функции показа всех записей.

**# addRecord**

На рисунке 4 показана блок-схема функции добавления новой записи.

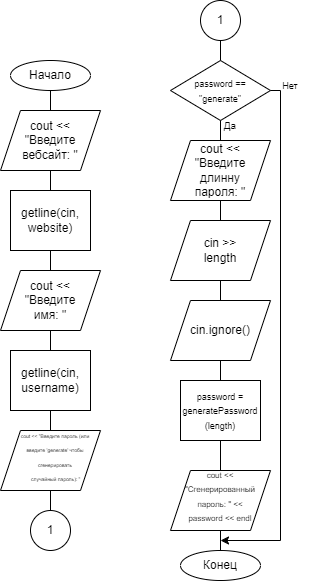


Рисунок 4 – блок-схема функции добавления новой записи.

**# editRecord**

На рисунке 5 показана блок-схема функции редактирования записи.

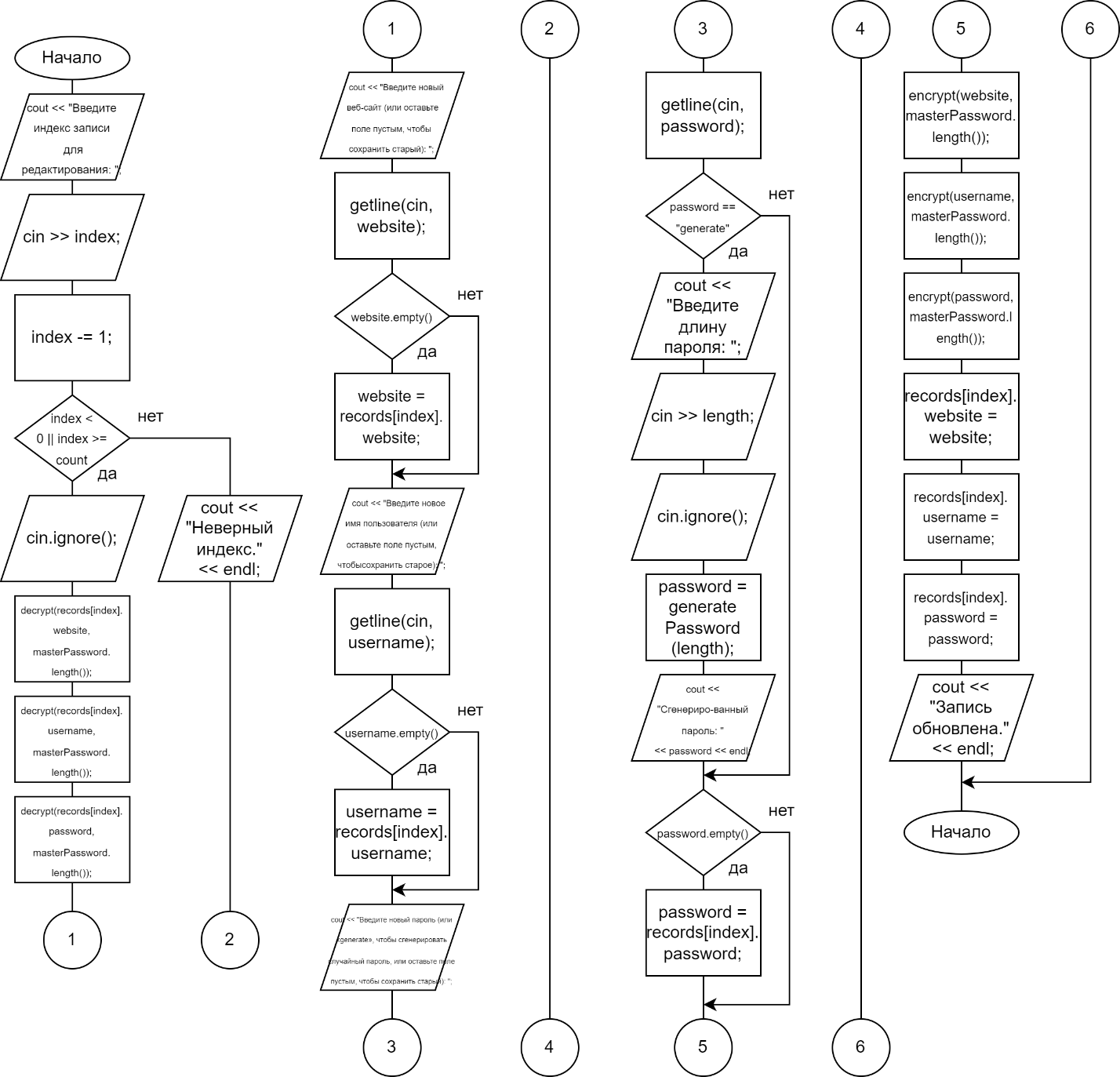


Рисунок 5 – Блок-схема функции редактирования записи.

**# deleteRecord**

На рисунке 6 представлена блок-схема функции удаления записи.

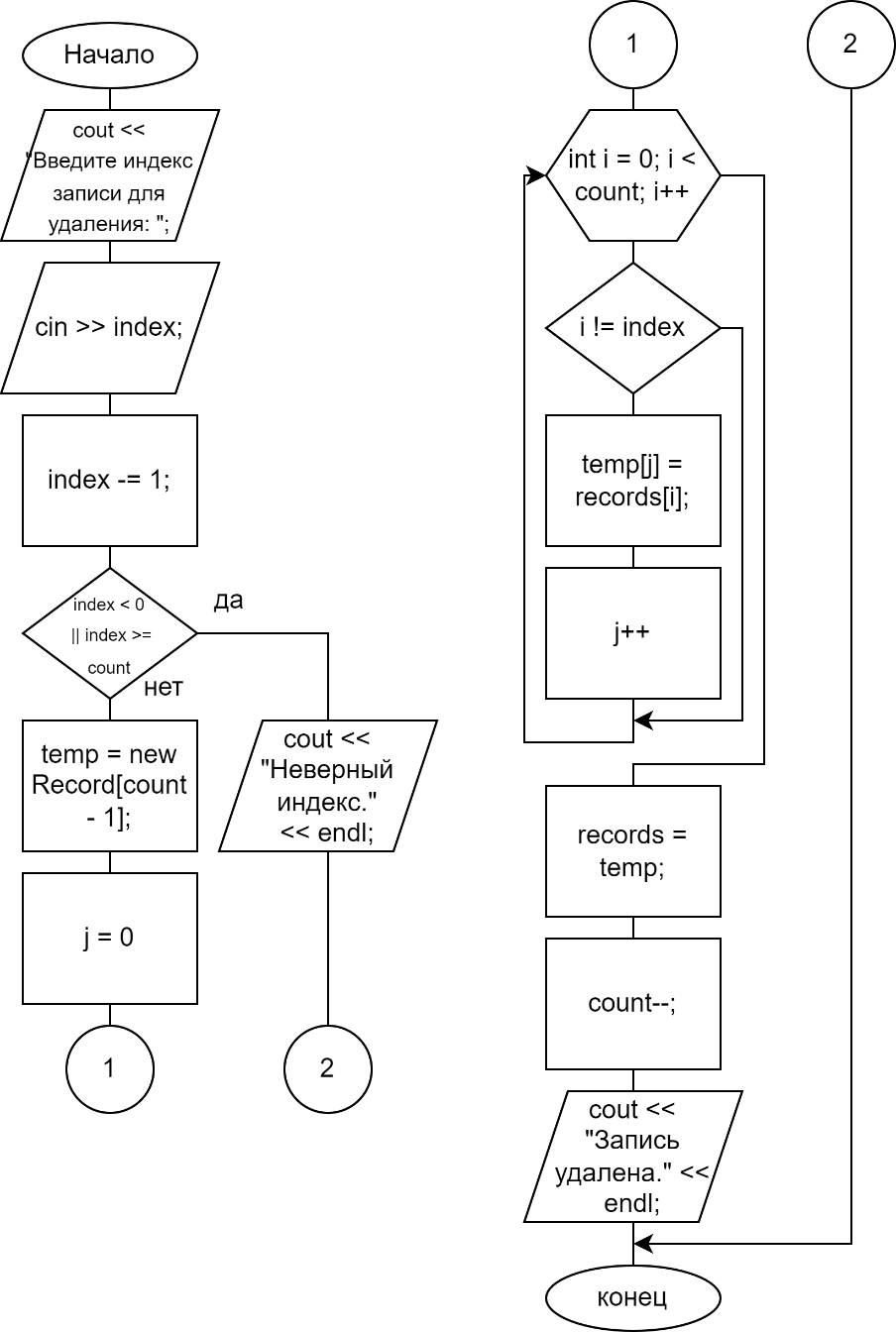


Рисунок 6 – Блок-схема функции удаления записи.

**# viewPassword**

На рисунке 7 представлена блок-схема функции просмотра пароля записи.

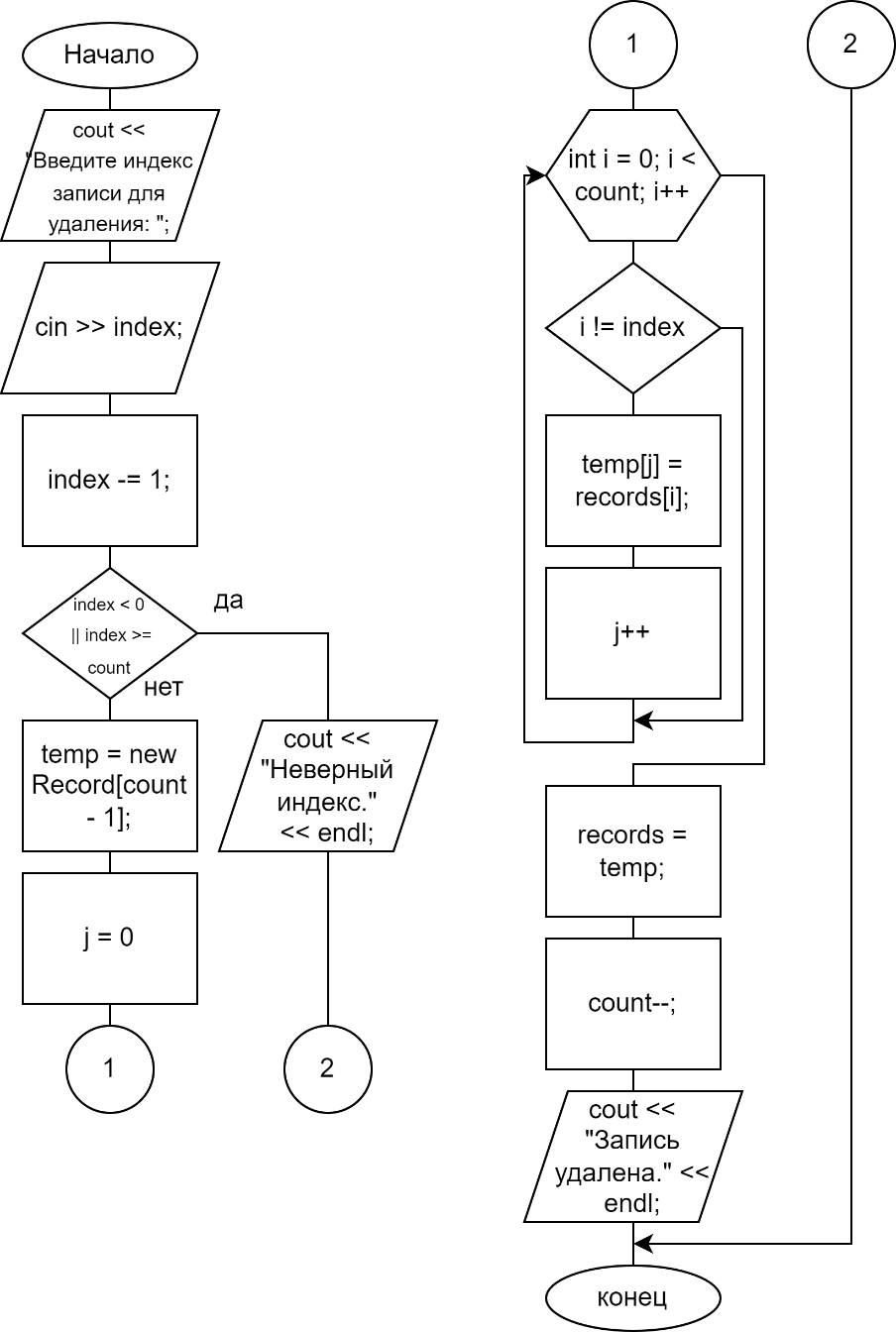


Рисунок 7 – Блок-схема функции просмотра пароля записи.

**# saveToFile**

На рисунке 8 представлена блок-схема функции сохранения записей в файл

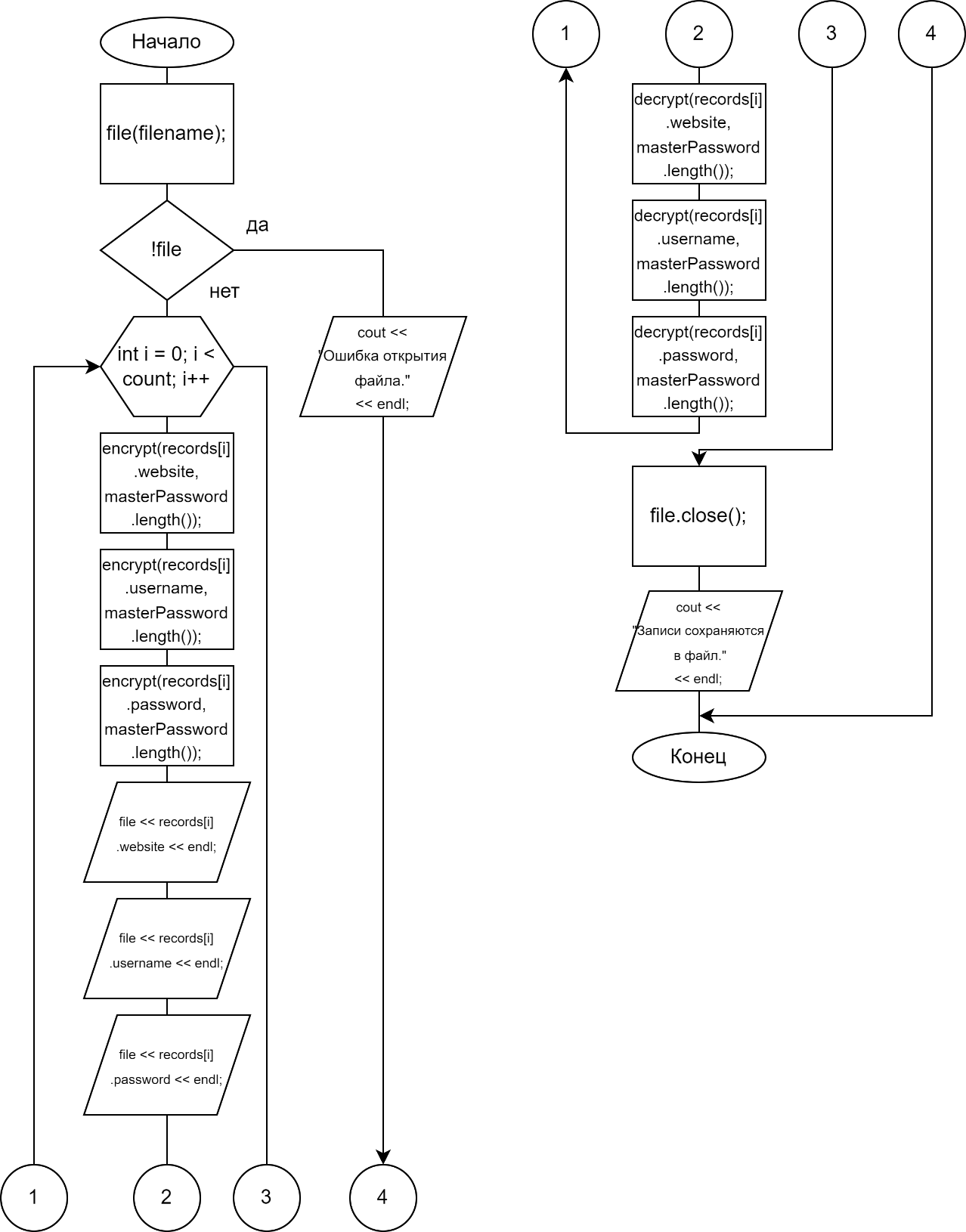


Рисунок 8 – Блок-схема функции сохранения записей в файл

**# loadFromFile**

На рисунке 9 представлена блок-схема функции загрузки записей из файла

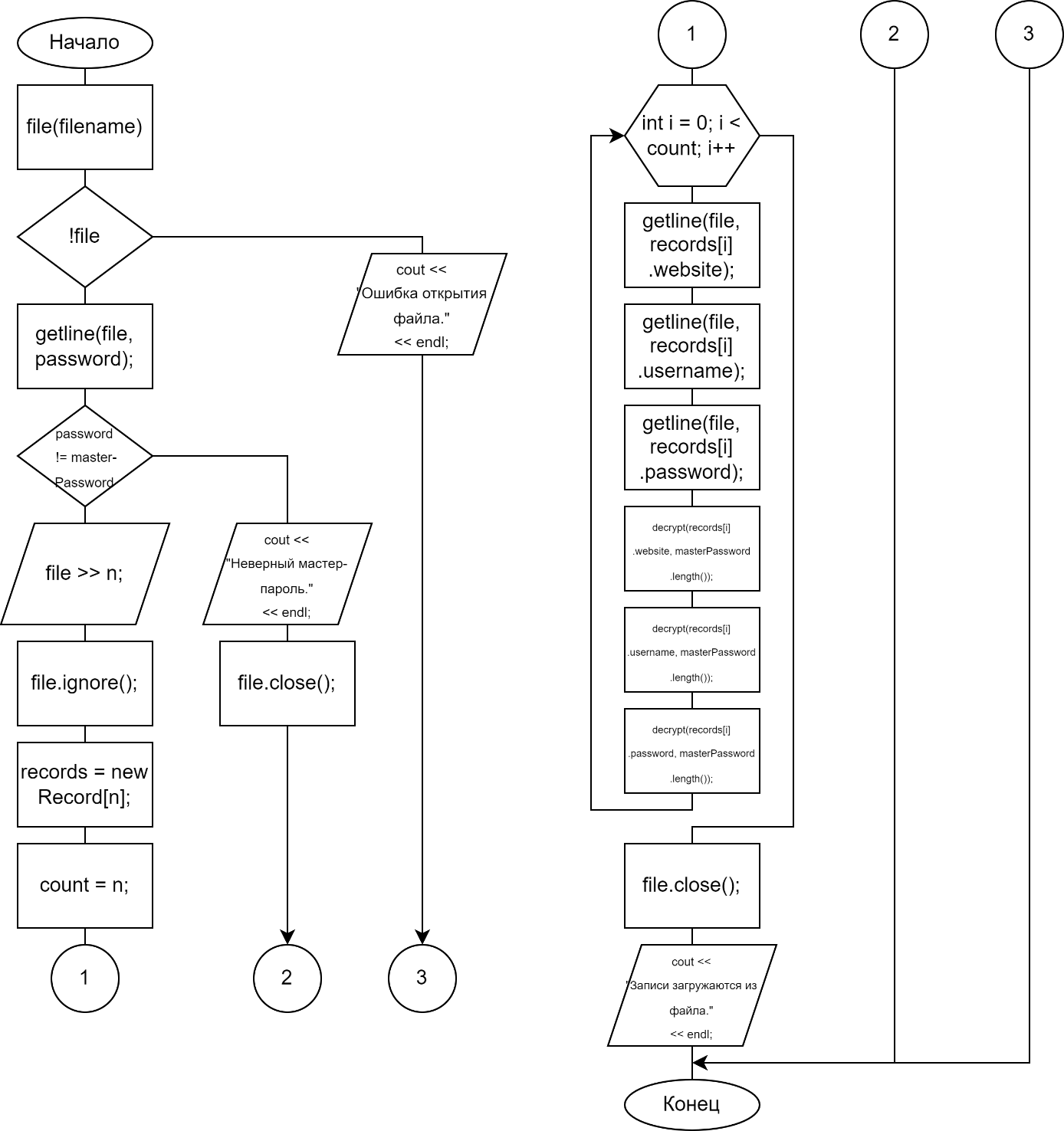


Рисунок 9 – Блок-схема функции загрузки записей из файла

**# encrypt**

На рисунке 10 представлена блок-схема функции шифрования

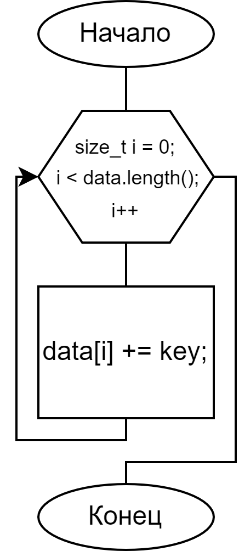


Рисунок 10 – Блок-схема функции шифрования

**# decrypt**

На рисунке 11 представлена блок-схема функции дешифрования

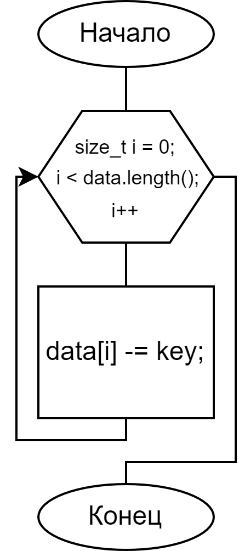
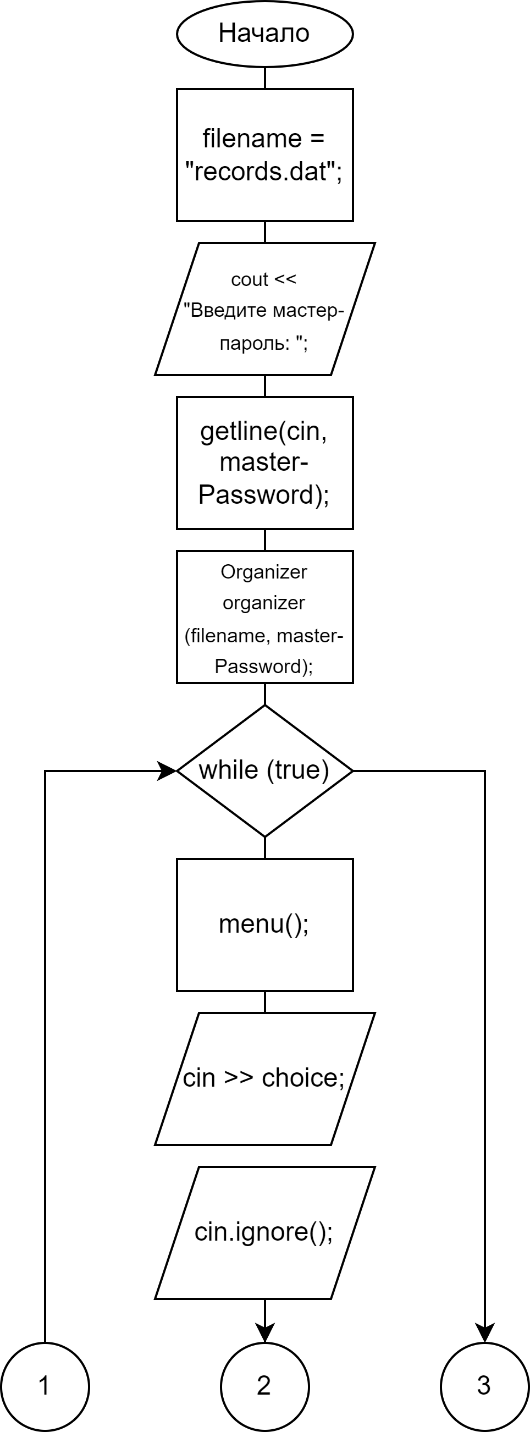


Рисунок 11 – Блок-схема функции дешифрования

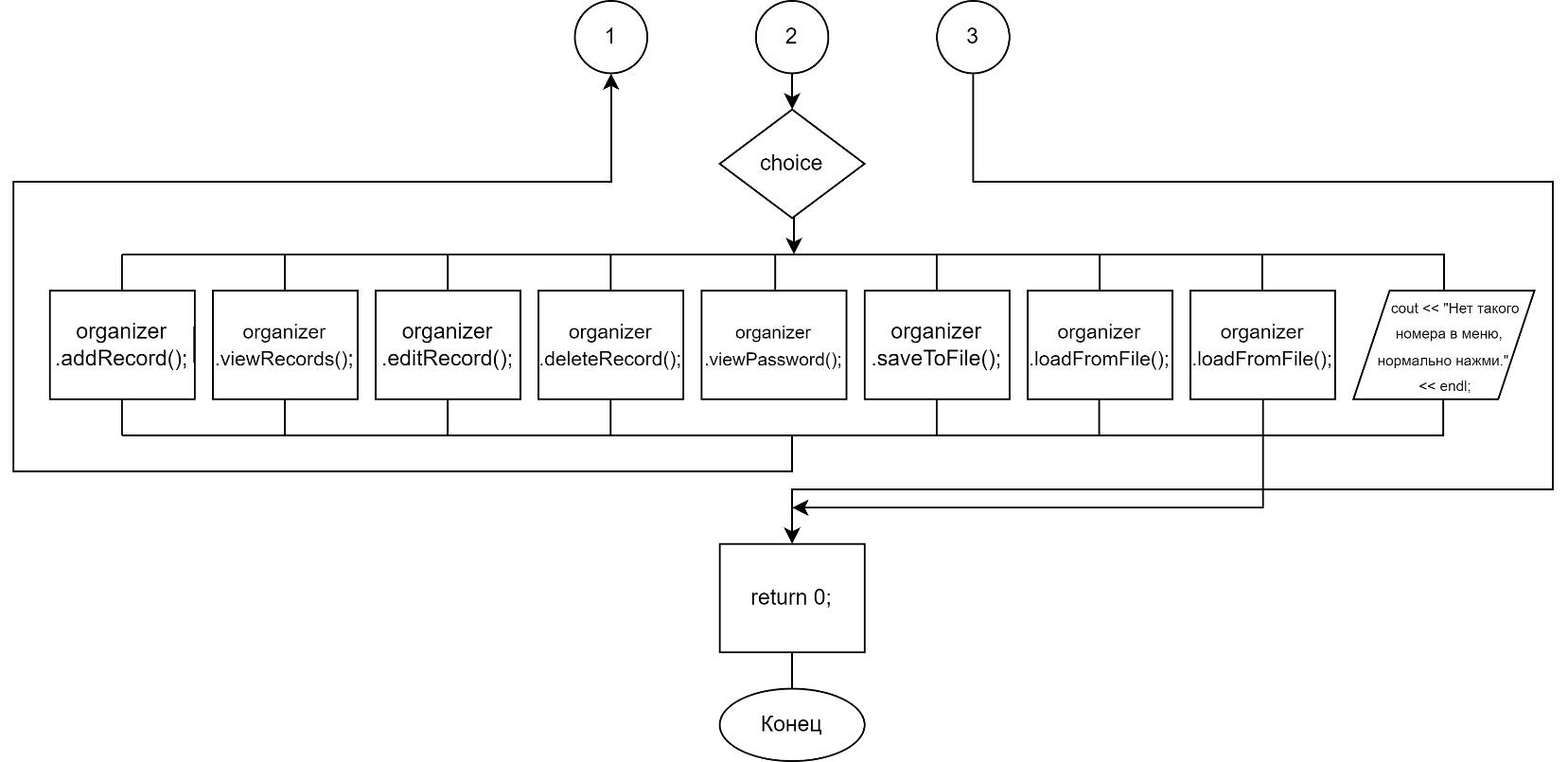
**# main**

На рисунке 12-13 представлена блок-схема главной функции.



Продолжение блоков 1, 2, 3 на странице 20

Рисунок 12 – Блок-схема главной функции.



Продолжение блоков 1, 2, 3 со страницы 19

Рисунок 13 – Блок-схема главной функции.

## Листинг

#include <iostream>

#include <fstream>

#include <string>

#include <cstdlib>

#include <ctime>

using namespace std;

// функция для шифрования данных

void encrypt(string & data, int key)

{

for (size\_t i = 0; i < data.length(); i++)

{

data[i] = data[i] + key;

}

}

// функция для дешифрования данных

void decrypt(string & data, int key)

{

for (size\_t i = 0; i < data.length(); i++)

{

data[i] = data[i] - key;

}

}

// класс записи логина и пароля

class Record

{

public:

string website;

string username;

string password;

Record(string website = "", string username = "", string password = "")

: website(website), username(username), password(password)

{

}

};

// класс органайзера логинов и паролей

class Organizer

{

private:

string filename; // имя файла для сохранения записей

string masterPassword; // мастер-пароль для доступа к файлу

Record\* records; // массив записей

int count; // количество записей

// функция для генерации случайного пароля

string generatePassword(int length)

{

string password;

const string symbols = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%^&\*()\_+-=[]{};:,.<>?";

srand(time(nullptr));

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

{

password += symbols[rand() % symbols.length()];

}

return password;

}

public:

Organizer(string filename, string masterPassword)

: filename(filename), masterPassword(masterPassword), records(nullptr), count(0)

{

}

~Organizer()

{

delete[] records;

}

// функция для добавления новой записи

void addRecord()

{

string website, username, password;

cout << "Введите вебсайт: ";

getline(cin, website);

cout << "Введите имя: ";

getline(cin, username);

cout << "Введите пароль (или введите 'generate' чтобы сгенерировать случайный пароль): ";

getline(cin, password);

if (password == "generate")

{

int length;

cout << "Введите длину пароля: ";

cin >> length;

cin.ignore();

password = generatePassword(length);

cout << "Сгенерированный пароль: " << password << endl;

}

// шифруем данные перед сохранением

encrypt(website, masterPassword.length());

encrypt(username, masterPassword.length());

encrypt(password, masterPassword.length());

// создаем новую запись и добавляем ее в массив

Record record(website, username, password);

Record\* temp = new Record[count + 1];

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

{

temp[i] = records[i];

}

temp[count] = record;

delete[] records;

records = temp;

count++;

cout << "Запись добавлена." << endl;

}

// функция для просмотра всех записей

void viewRecords()

{

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

{

// дешифруем данные перед выводом на экран

decrypt(records[i].website, masterPassword.length());

decrypt(records[i].username, masterPassword.length());

decrypt(records[i].password, masterPassword.length());

cout << "Вебсайт: " << records[i].website << endl;

cout << "Имя пользователя: " << records[i].username << endl;

cout << "Пароль: " << records[i].password << endl;

cout << endl;

// шифруем данные обратно после вывода на экран

encrypt(records[i].website, masterPassword.length());

encrypt(records[i].username, masterPassword.length());

encrypt(records[i].password, masterPassword.length());

}

}

// функция для редактирования записи

void editRecord()

{

int index;

cout << "Введите индекс записи для редактирования: ";

cin >> index;

index = index - 1;

if (index < 0 || index >= count)

{

cout << "Неверный индекс." << endl;

return;

}

cin.ignore();

// дешифруем данные для редактирования

decrypt(records[index].website, masterPassword.length());

decrypt(records[index].username, masterPassword.length());

decrypt(records[index].password, masterPassword.length());

string website, username, password;

cout << "Введите новый веб-сайт (или оставьте поле пустым, чтобы сохранить старый): ";

getline(cin, website);

if (website.empty())

{

website = records[index].website;

}

cout << "Введите новое имя пользователя (или оставьте поле пустым, чтобы сохранить старое): ";

getline(cin, username);

if (username.empty())

{

username = records[index].username;

}

cout << "Введите новый пароль (или «generate», чтобы сгенерировать случайный пароль, или оставьте поле пустым, чтобы сохранить старый): ";

getline(cin, password);

if (password == "generate")

{

int length;

cout << "Введите длину пароля: ";

cin >> length;

cin.ignore();

password = generatePassword(length);

cout << "Сгенерированный пароль: " << password << endl;

}

if (password.empty())

{

password = records[index].password;

}

// шифруем новые данные перед сохранением

encrypt(website, masterPassword.length());

encrypt(username, masterPassword.length());

encrypt(password, masterPassword.length());

// обновляем запись в массиве

records[index].website = website;

records[index].username = username;

records[index].password = password;

cout << "Запись обновлена." << endl;

}

// функция для удаления записи

void deleteRecord()

{

int index;

cout << "Введите индекс записи для удаления: ";

cin >> index;

index = index - 1;

if (index < 0 || index >= count)

{

cout << "Неверный индекс." << endl;

return;

}

// удаляем запись из массива

Record\* temp = new Record[count - 1];

int j = 0;

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

{

if (i != index)

{

temp[j] = records[i];

j++;

}

}

delete[] records;

records = temp;

count--;

cout << "Запись удалена." << endl;

}

// функция для вывода одного конкретного пароля для его копипаста из консоли

void viewPassword()

{

int index;

cout << "Введите индекс записи для просмотра пароля: ";

cin >> index;

index = index - 1;

if (index < 0 || index >= count)

{

cout << "Неверный индекс." << endl;

return;

}

// дешифруем пароль для вывода на экран

decrypt(records[index].password, masterPassword.length());

cout << "Пароль для " << records[index].website << ": " << records[index].password << endl;

encrypt(records[index].password, masterPassword.length()); // шифруем пароль обратно после вывода на экран

}

// функция для сохранения всех записей в файл

void saveToFile()

{

ofstream file(filename);

if (!file)

{

cout << "Ошибка открытия файла." << endl;

return;

}

// шифруем каждую запись перед сохранением в файл

file << masterPassword << endl;

file << count << endl;

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

{

encrypt(records[i].website, masterPassword.length());

encrypt(records[i].username, masterPassword.length());

encrypt(records[i].password, masterPassword.length());

file << records[i].website << endl;

file << records[i].username << endl;

file << records[i].password << endl;

decrypt(records[i].website, masterPassword.length());

decrypt(records[i].username, masterPassword.length());

decrypt(records[i].password, masterPassword.length());

}

file.close();

cout << "Записи сохраняются в файл." << endl;

}

// функция для загрузки всех записей из файла

void loadFromFile()

{

ifstream file(filename);

if (!file)

{

cout << "Ошибка открытия файла." << endl;

return;

}

// проверяем, что мастер-пароль совпадает с тем, что был введен при создании объекта Organizer

string password;

getline(file, password);

if (password != masterPassword)

{

cout << "Неверный мастер-пароль." << endl;

file.close();

return;

}

// читаем количество записей

int n;

file >> n;

file.ignore();

// создаем массив для записей и читаем записи из файла

delete[] records;

records = new Record[n];

count = n;

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

{

getline(file, records[i].website);

getline(file, records[i].username);

getline(file, records[i].password);

// дешифруем каждую запись после чтения из файла

decrypt(records[i].website, masterPassword.length());

decrypt(records[i].username, masterPassword.length());

decrypt(records[i].password, masterPassword.length());

}

file.close();

cout << "Записи загружаются из файла." << endl;

}

};

int main()

{

setlocale(LC\_ALL, "ru");

string filename = "records.dat";

string masterPassword;

int choice;

cout << "Введите мастер-пароль: ";

getline(cin, masterPassword);

Organizer organizer(filename, masterPassword);

// даем пользователю выбрать действие из списка

while (true)

{

cout << endl;

cout << "1. Добавить запись" << endl;

cout << "2. Посмотреть все записи" << endl;

cout << "3. Изменить запись" << endl;

cout << "4. Удалить запись" << endl;

cout << "5. Посмотреть пароль" << endl;

cout << "6. Сохранить в файл" << endl;

cout << "7. Загрузить из файла" << endl;

cout << "8. Выход" << endl;

cout << "Введите номер действия: ";

cin >> choice;

cout << endl;

cin.ignore();

switch (choice)

{

case 1:

organizer.addRecord();

break;

case 2:

organizer.viewRecords();

break;

case 3:

organizer.editRecord();

break;

case 4:

organizer.deleteRecord();

break;

case 5:

organizer.viewPassword();

break;

case 6:

organizer.saveToFile();

break;

case 7:

organizer.loadFromFile();

break;

case 8:

return 0;

default:

cout << "Нет такого номера в меню, нормально нажми." << endl;

break;

}

}

return 0;

}

## Таблица тестов

Чтобы провести тестирование по модели **белого ящика** поставим задачу, которая задействует все возможности программы. Построим граф потока управления на основе блок-схемы алгоритма. Для выполнения поставленной задачи необходимо разбить ее на подзадачи. Для каждой подзадачи используется определенный набор функций, представляющий собой последовательность пунктов меню. Каждая инструкция представлена отдельным узлом, а ребра показывают направление выполнения алгоритма. Граф потока управления приведен на рисунке 14.

Задание для покрытия всех функций программы:

Создать 2 записи, одну со случайным паролем, другую с заданным паролем, удалить первую запись, изменить имя второй записи, сохранить запись в файл, загрузить запись в файл. Проверить как шифруются записи. Проверить работу мастер-пароля, ввести неправильный и подтвердить ошибку.

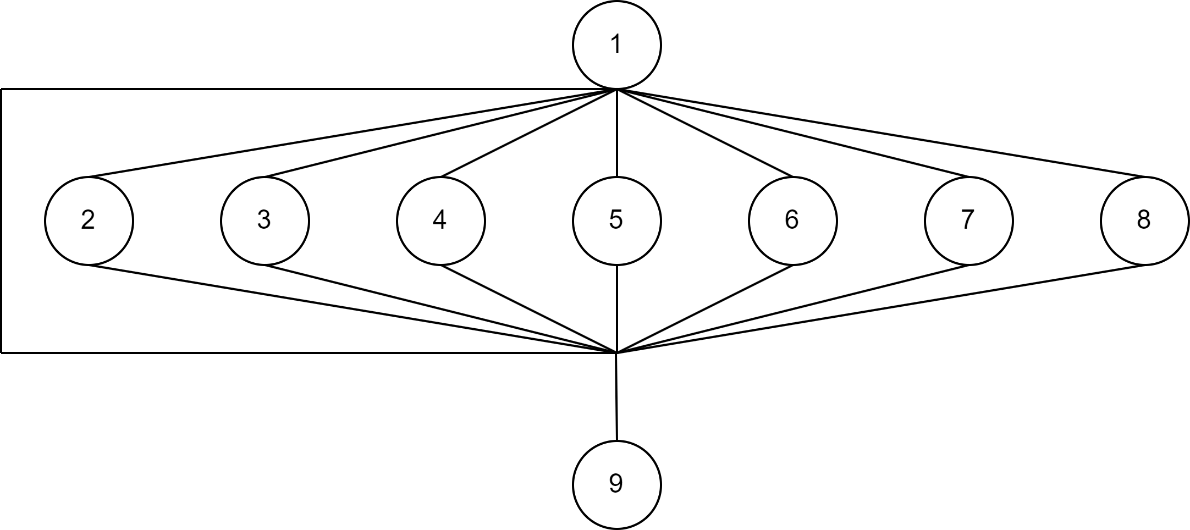


Рисунок 14 — Граф потока управления для алгоритма

Чтобы провести тесты из таблицы 2, нужен определенный набор функций. Комбинации определенных функций образуют проход.

В результате выполнения теста 1 выполняется проход по пути *1-2-1-3-9*. В результате выполнения теста 2 - путь *1-2-1-3-9*. Теста 3 - *1-5-1-3-9*. Теста 4 – 1-4-1-3-9. Тест 5 – 1-7-9. Тест 6 – 1-8-1-3-9.

Тестирование методом **черного ящика**

Таблица 3 – Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Входные данные | Выходные данные |
| 1 | Создание записи с генерацией пароля | Вебсайт: youtube  Имя: roma  Пароль: generate  Длина: 8 | Вебсайт: youtube  Имя: roma  Пароль: случайный пароль |
| 2 | Создание записи с заданным паролем | Вебсайт: twitch  Имя: igor  Пароль: aboba123 | Вебсайт: twitch  Имя: igor  Пароль: aboba123 |
| 3 | Удаление первой записи | Index: 1 |  |
| 4 | Изменение второй записи | Index: 1  Имя: igoretto | Вебсайт: twitch  Имя: igoretto  Пароль: aboba123 |
| 5 | Сохранение записей в файл |  | Записи сохраняются в файл |
| 6 | Загрузка записей из файла |  | Записи загружаются из файла |
| 7 | Проверка зашифрованного файла | Зайти в папку с записями и убедиться в том, что они зашифрованы | qwerty  1  ЂѓuЂot  us{~qЂЂ{  mn{nm=>? |
| 8 | Ввести не правильный мастер-пароль | Мастер-пароль: notqwerty | Неверный мастер пароль |

На основе этой таблицы будем проводить тестирование, охватывающие все возможные варианты.

## Анализ вычислительной сложности алгоритма

// функция для шифрования данных

void encrypt(string & data, int key)

{

for (size\_t i = 0; i < data.length(); i++)

{

data[i] = data[i] + key;

}

}

// функция для дешифрования данных

void decrypt(string & data, int key)

{

for (size\_t i = 0; i < data.length(); i++)

{

data[i] = data[i] - key;

}

}

// функция для генерации случайного пароля

string generatePassword(int length)

{

string password;

const string symbols = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%^&\*()\_+-=[]{};:,.<>?";

srand(time(nullptr));

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

{

password += symbols[rand() % symbols.length()];

}

return password;

}

Проанализировав код основных вычислительных функций, видим, что размер входных данных для алгоритма является не постоянным, следовательно, алгоритм зависит от объема входных данных. В результате выполнения алгоритма не все из операторов выполняется не более одного раза, но все используемые циклы содержат константные значения. Исходя из вышесказанного, можно заключить, что алгоритм не имеет постоянное время выполнения. Используя О-нотацию, верхняя асимптотическая граница функции сложности исследуемого алгоритма записывается как **O(length)**.

Результаты тестирования

Таблица 4 – Таблица результатов тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Выходные данные | Результат |
| 1 | Вебсайт: youtube  Имя: roma  Пароль: generate  Длина: 8 | Вебсайт: youtube  Имя: roma  Пароль: случайный пароль | Тест пройден |
| 2 | Вебсайт: twitch  Имя: igor  Пароль: aboba123 | Вебсайт: twitch  Имя: igor  Пароль: aboba123 | Тест пройден |
| 3 | Index: 1 |  | Тест пройден |
| 4 | Index: 1  Имя: igoretto | Вебсайт: twitch  Имя: igoretto  Пароль: aboba123 | Тест пройден |
| 5 |  | Записи сохраняются в файл | Тест пройден |
| 6 |  | Записи загружаются из файла | Тест пройден |
| 7 | Зайти в папку с записями и убедиться в том, что они зашифрованы | qwerty  1  ЂѓuЂot  us{~qЂЂ{  mn{nm=>? | Тест пройден |
| 8 | Мастер-пароль: notqwerty | Неверный мастер пароль | Тест пройден |

Результаты выполнения тестов с 1 по 8 для всех способов приведены на рисунках с 17 по 49 соответственно.

**Тесты**

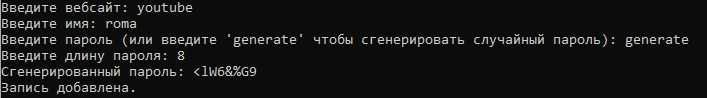




Рисунок 15 – Результат выполнения теста 1

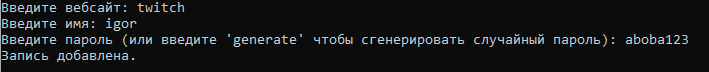




Рисунок 16 – Результат выполнения теста 2



Рисунок 17 – Результат выполнения теста 3



Рисунок 18 – Результат выполнения теста 4



Рисунок 19 – Результат выполнения теста 5



Рисунок 20 – Результат выполнения теста 6

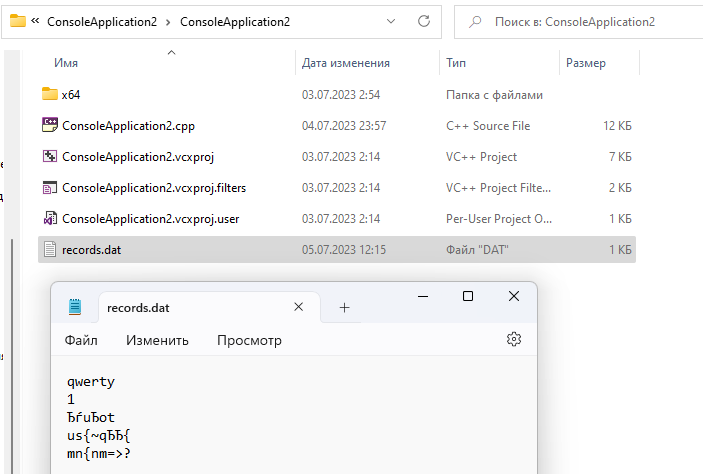


Рисунок 21 – Результат выполнения теста 7





Рисунок 22 – Результат выполнения теста 8

Организация работ над проектом

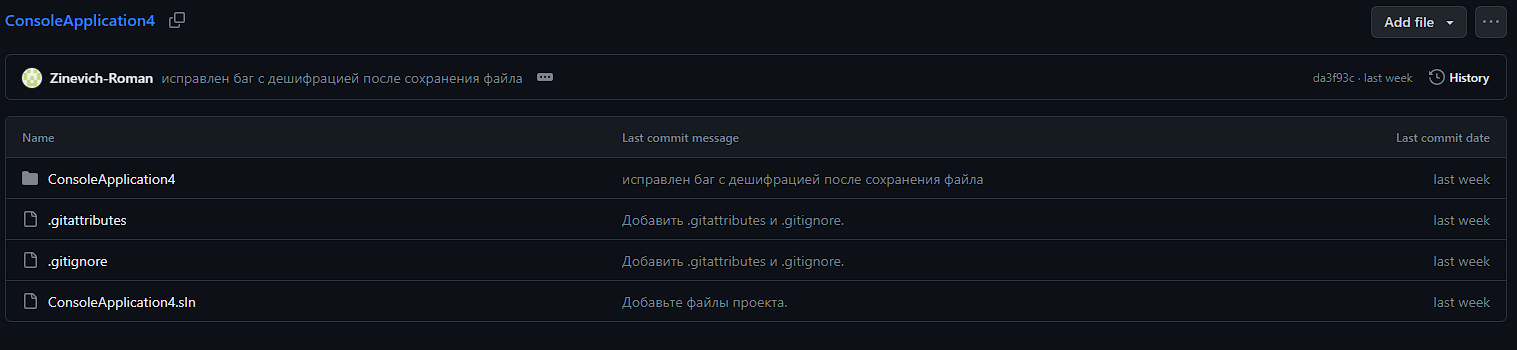


Рисунок 23 – система управления версиями

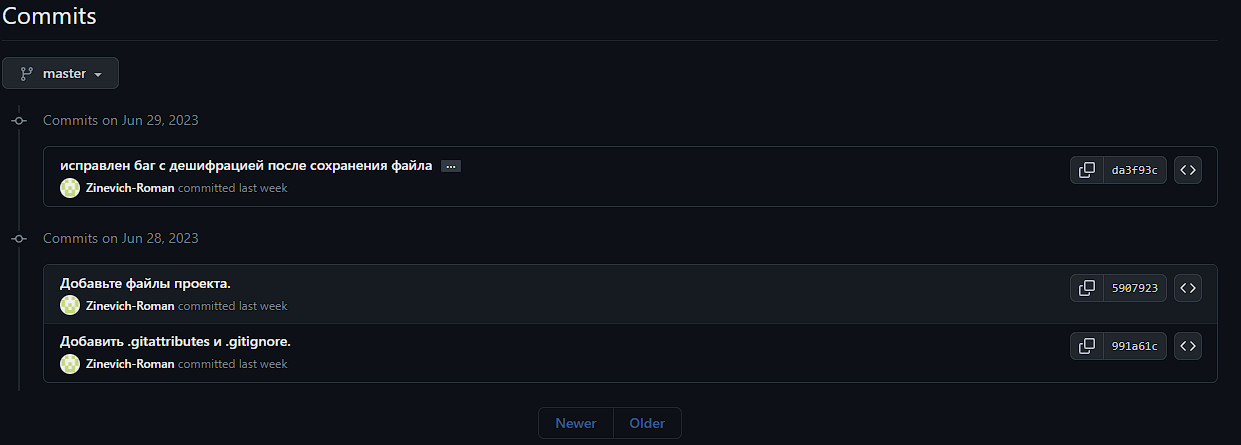


Рисунок 24 – система управления версиями

Заключение

В процессе практики был разработан и реализован алгоритм решения индивидуального задания. С использованием методов классов эквивалентности и предположения об ошибке сформировано 8 тестов, покрывающих все пути. Составленная на языке С++ программа, реализующая разработанный алгоритм, прошла все 8 тестов успешно.

В ходе учебной практики были получены навыки составления блок-схем алгоритма решения задачи, разработки тестов с использованием различных методов, а также применения языка программирования С++ и для реализации алгоритма решения задачи.

Список литературы

1. Павловская Т.А. С и С++. Программирование на языке высокого уровня. СПб.: Питер, 2003. – 461с.
2. Программирование на языке Си [Электронный ресурс] URL: http://www.codenet.ru/progr/cpp/1/ (дата обращения: 29.06.20)
3. Меньшиков Ф. В. Олимпиадные задачи по программированию. - СПб.: Питер,2006