МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гомельский государственный университет

имени Франциска Скорины»

Факультет физики и информационных технологий

Кафедра общей физики

**Разработка программы учета сведений о пациентах медицинского центра**

Курсовая работа

По учебной дисциплине

''Технологии разработки программного обеспечения''

Исполнитель Бортник Н. А.

студент группы МС-12

Научный руководитель Ковалёв А. А.

Гомель 2022

**Реферат**

**Цель курсовой работы*:*** совершенствование навыков разработки программного обеспечения на высокоуровневом языке, получение и применение новых знаний в практике.

**Задача курсовой работы*:*** Освоение новой среды разработки, изучения основных принципов структурного программирования и технологии нисходящего программирования, конструирования алгоритмов и самой программы.

**Курсовая работа*:*** 17 страниц, 3 части, 24 рисунка, 4 источника, 1 приложение.

**Ключевые слова*:*** программирование, программа, программное обеспечение, проектирование, конструирование, реализация, среда разработки, базы данных.

**Методы исследования*:*** разработка программы, эксперимент, теоретический метод.

**Вывод*:*** по результату выполнения курсового проекта было написано приложение учета сведений о пациентах медицинского центра.

**Содержание**

[Введение 3](#_heading=h.30j0zll)

[1. Описание теоретических сведений 4](#_heading=h.1fob9te)

[1.1 Высокоуровневый язык программирования С 4](#_heading=h.3znysh7)

[1.2](#_heading=h.2et92p0) Технология нисходящего программирования 5

1.3 Структурное программирование 5

[2. Описание основных функций программы 6](#_heading=h.tyjcwt)

[2.1](#_heading=h.3dy6vkm) Процедура main 6

[2.2 Функция](#_heading=h.1t3h5sf) massiveNulling 6

[2.3 Функция](#_heading=h.4d34og8) logginAgain 6

[2.4 Функция](#_heading=h.2s8eyo1) authorization 6

[2.5 Функция](#_heading=h.17dp8vu) loginSystem 6

[2.6 Функция](#_heading=h.3rdcrjn) adminCreate 6

[2.7 Функция](#_heading=h.26in1rg) loadDatabase 6

[2.8 Функция save](#_heading=h.lnxbz9)Database 6

[2.9 Функция](#_heading=h.35nkun2) outputPatientInformation 7

[2.10 Функция](#_heading=h.1ksv4uv) addNewPatient 7

[2.11 Функция](#_heading=h.44sinio) editPatientInformation 7

[2.12 Функция](#_heading=h.2jxsxqh) deletePatient 7

[2.13 Функция](#_heading=h.z337ya) searchByAgeAndDiagnosis 7

[2.14 Функция](#_heading=h.3j2qqm3) searchNonresidentPatient 7

[2.15 Функция](#_heading=h.1y810tw) outputUsers 7

[2.16 Функция](#_heading=h.4i7ojhp) addNewUser 8

2.17 Функция editUsers 8

2.18 Функция deleteUsers 8

2.19 Функция userInteraction 8

2.20 Функция patientInteraction 8

2.21 Функция userMenu 8

2.22 Функция adminMenu 8

[3 Работоспособность программы](#_heading=h.2xcytpi) 9

[3.1 Вход в аккаунт](#_heading=h.1ci93xb) 9

[3.2](#_heading=h.3whwml4) Возможности администратора 10

[3.3.1](#_heading=h.2bn6wsx) Возможности панели работы с пользователями 11

[3.3.2](#_heading=h.qsh70q) Возможности панели работы с пациентами 13

[3.4](#_heading=h.3as4poj) Возможности пользователя 15

3.5 Алгоритм работы программы 16

[Заключение 1](#_heading=h.ihv636)7

[Список использованных источников](#_heading=h.32hioqz) 18

[Приложение](#_heading=h.1hmsyys) 19

**Введение**

Основное задание курсовой включает в себя разработку программы учета сведений о пациентах медицинского центра. Сведения о пациентах медицинского центра должны содержать: ФИО пациента, пол, дату рождения, место проживания (город), контактный телефон, диагноз.

Основными требованиями к курсовому заданию являются, разработка консольной программы, в которой есть: система авторизации, регистрация новых пользователей/добавление новых пациентов, режим редактирования данных, как пациентов так и пользователей, просмотр всех данных, работу с данными, сортировку.

Курсовая работа содержит содержит 3 главы. Первая глава описывает нам теоретические сведения. Вторая глава описывает основные функции программы. Третья глава демонстрирует работу программы, описывает принцип её работы.

1. **Описание теоретических сведений**
   1. **Высокоуровневый язык программирования С**

**Си (C)** является языком функций, типов данных, операторов присваивания и управления последовательностью вычислений. Программируя на Си (C), вы осуществляете обращение к функциям, и большинство функций возвращают некоторые значения. Значение, возвращаемое функцией, будь то значение переменной или константа, может использоваться в операторе присваивания, который изменяет значение другой переменной.

Дополненный операторами управления последовательностью вычислений (while, for, do, switch), Си (C) превращается в язык высокого уровня, способствующий хорошему стилю программирования.Си (C) имеет небольшой набор типов данных: целые числа, числа с плавающей запятой, битовые поля и перечислимый тип. В языке Си (C) вы можете описать переменную типа указатель, который связывается с объектом, принадлежащим к любому типу данных. Адресная арифметика языка Си (C) является чувствительной к типу данных того объекта, с которым связан используемый указатель.

Разрешены также указатели к функциям. Вы можете расширить список типов данных путем создания структур с иерархической зависимостью входящих в него типов данных. Каждый тип данных может принадлежать либо к основному типу, либо к ранее описанному структурному типу. Объединения напоминают структуры, но определяют различные виды иерархических зависимостей, в которых данные разных типов располагаются в памяти.

Допустимо описание массивов данных различных типов, включая структуры и объединения. Массивы могут быть многомерными.Функции Си (C) являются рекурсивными по умолчанию. Вы можете, правда, создать функцию, которая не будет рекурсивной, но сам язык по своей природе стремится поддерживать рекурсивность и требует минимальных усилий при программировании рекурсий.

Программа функции на языке Си (C) разбивается на блоки, в каждом из которых могут быть определены свои собственные локальные переменные. Блоки могут выбираться для исполнения по результату выполнения оператора управления последовательностью вычислений. Блоки могут быть вложенными друг в друга.Переменные и функции могут быть глобальными для программы, глобальными для исходного модуля или локальными для блока, в котором они описаны. Локальные переменные могут быть описаны таким образом, что они будут сохранять свои значения при всех обращениях внутри данного блока (статические переменные) или же будут восприниматься как новые объекты при каждом обращении (автоматические переменные).

* 1. **Технология нисходящего программирования**

**Технология нисходящего программирования**— это метод после­довательной детализации, со­здание программы «сверху вниз». Сначала разрабатывается основная программа (ее общая структура) и в ней за­писываются обращения к пока еще не написанным вспомо­гательным подпрограммам; и так далее — до самых простых «неделимых» подпрограмм.

**1.3 Принципы методологии структурного программирования**

**Структурное программирование** — парадигма программирования, в основе которой лежит представление программы в виде иерархической структуры блоков. Концептуализирована в конце 1960-х — начале 1970-х годов на фундаменте теоремы Бёма — Якопини, математически обосновывающей возможность структурной организации программ, и работы Эдсгера Дейкстры «О вреде оператора goto».

В соответствии с парадигмой, любая программа, которая строится без использования оператора goto, состоит из трёх базовых управляющих конструкций: последовательность, ветвление, цикл; кроме того, используются подпрограммы. При этом разработка программы ведётся пошагово, методом «сверху вниз».

Методология структурного программирования появилась как следствие возрастания сложности решаемых на компьютерах задач, и соответственно, усложнения программного обеспечения. В 1970-е годы объёмы и сложность программ достигли такого уровня, что традиционная (неструктурированная) разработка программ перестала удовлетворять потребностям практики. Программы становились слишком сложными, чтобы их можно было нормально сопровождать. Поэтому потребовалась систематизация процесса разработки и структуры программ.

Методология структурной разработки программного обеспечения была признана «самой сильной формализацией 70-х годов».

По мнению Бертрана Мейера, «Революция во взглядах на программирование, начатая Дейкстрой, привела к движению, известному как структурное программирование, которое предложило систематический, рациональный подход к конструированию программ. Структурное программирование стало основой всего, что сделано в методологии программирования, включая и объектное программирование».

1. **Описание основных функций программы**
   1. **Процедура main**

Главная процедура запрашивает логин, пароль, вызывает функции проверки логина, создание базы данных о пользователях и пациентов, создание аккаунта с ролью админа, если данных о юзерах нет в базе данных, загрузки данных о пациентах и юзерах, реализует главное меню программы в цикле с предусловием, который будет остановлен, когда в меню будет выбран пункт выхода, после выхода сохраняет новые данные о пользователях и пациентах.

* 1. **Функция massiveNulling**

Функция предназначена для зануления массива перед тем как присваивать переменные.

* 1. **Функция logginAgain**

Функция предназначена для повторного входа в учетную запись, при выходе из неё, так же это функция помогает залогиниться на другой аккаунт.

* 1. **Функция authorization**

Функция предназначена сравнивает логин с паролем, которая в дальнейшем пригодится для входа в аккаунт.

* 1. **Функция loginSystem**

Функция предназначена для входа в аккаунт, она запрашивает логин и пароль, а дальше происходит авторизация в учетную запись.

* 1. **Функция adminCreate**

Функция предназначена для создания аккаунта администратора, если в базе аккаунтов, нет ни одного аккаунта.

* 1. **Функция loadDatabase**

Функция предназначена для загрузки данных о пользователях и пациентах из базы данных.

* 1. **Функция saveDatabase**

Функция предназначена для сохранения всех данных о пользователях и пациентах в базу данных.

* 1. **Функция outputPatientInformation**

Функция предназначена для вывода информации о всех пациентах, которые присутствуют в базе данных, она выводит основные данные о них, такие как ФИО, полный возраст, дата рождения, пол, место рождения и диагноз.

* 1. **Функция addNewPatient**

Функция предназначена для добавления нового пациента, в контекстном меню требуется ввести все данные о новом пациенте, далее он добавляется в базу данных.

* 1. **Функция editPatient information**

Функция предназначена для изменения данных о пациенте, если произошла какая-то ошибка, данные при изменении перезаписываются в базе данных.

* 1. **Функция deletePatient**

Функция предназначена для пациента из базы данных. Посредством смещения массива, данные пациента удаляются из базы данных.

* 1. **Функция searchByAgeAndDiagnosis**

Функция предназначена для поиска пациентов по возрасту и их диагнозу, если возраст пациента и его диагноз совпадают, информация о нём появляется в программе.

* 1. **Функция searchNonResidentPeople**

Функция предназначена для поиска иногородних пациентов, если введенное расположение госпиталя совпадает с местом проживания пациента, его данные выводятся в программе.

* 1. **Функция outputUsers**

Функция предназначена для вывода информации о юзере на экран, там показывается его логин, пароль и роль(администратор или пользователь). Все эти данные сохраняются в специальном файле, который подгружается при каждом входе в программу.

* 1. **Функция addNewUser**

Функция предназначена для cоздания нового пользователя, администратор задаёт новому пользователю логин, пароль и его роль(администратор или пользователь).

**2.17** **Функция editUsers**

Функция предназначена для редактирования данных о пользователе, администратор может поменять логин, пароль и роль пользователя.

**2.18 Функция deleteUsers**

Функция предназначена для удаления аккаунта пользователя, при удалении аккаунта, все данные о нём стираются из базы данных и аккаунт удаляется.

**2.19 Функция userInteraction**

Функция выводит возможности обычного пользователя, просмотр базы данных пациентов, поиск по диагнозу и возрасту, поиск иногородних пациентов.

**2.20 Функция patientInteraction**

Функция выводит возможности для изменения пациентов, редактирование данных пациентов, удаление пациентов, добавление пациетов, вывод информации о пациентах, поиск пациента по возрасту и диагнозу, поиск иногородних пациентов.

**2.21 Функция userPanel**

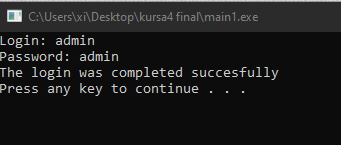
Функция представляет из себя возможности юзера, которые доступны ему в меню, основные возможности пользователя это: просмотр сведений о пациентах, поиск пациента по возрасту и диагнозу, поиск иногородних пациентов.

**2.22 Функция adminPanel**

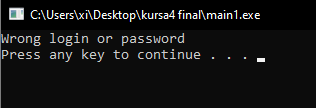
Функция представляет из себя возможности администратора, которые доступны ему в меню, основные возможности администратора это: работа с данными пользователей(удаление, создание, редактирование аккаунтов), работа с данными пациентов(удаление данных о пациентах, добавление новых пациентов, редактирование пациентов, поиск по диагнозу и возрасту, поиск иногородних пациентов.

1. **Работоспособность программы**
   1. **Вход в аккаунт**

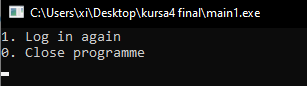
После запуска программы, нас встречает поле логин, в котором следует ввести логин от учетной записи, после ввода логина, программа запрашивает пароль, если логин и пароль совпадают, вход в аккаунт происходит успешно, если логин и пароль не совпадают, программа выводит надпись о том что данные не совпадают и предлагает ввести их ещё раз. Если при запуске программы в базе данных пользователей, нет ни одной учетной записи, запись с правами администратора создаётся автоматически.



*Рисунок 1 - Успешный вход в аккаунт*

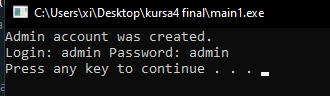


*Рисунок 2 - Неудачный вход в аккаунт*

**

*Рисунок 3 - Система повторного входа в аккаунт, при неправильно введенных*

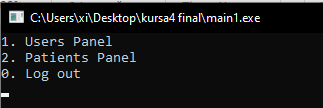
*данных*

**

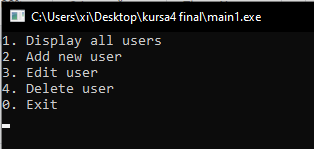
*Рисунок 4 - Создание системой аккаунта с правами администратора, если в базе данных нет ни одного пользователя*

* 1. **Возможности администратора**

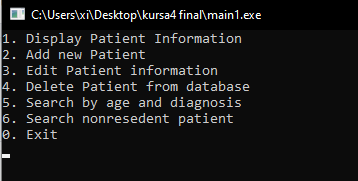
После входа в аккаунт администратора, появляется меню, в котором доступны всего два пункта, пользователи и пациенты. При выборе пользователей, администратору предоставляется выбор, на выбор даётся 4 функции, добавление пользователей, удаление пользователей, просмотр пользователей и редактирование пользователей. При выборе пациентов, администратору предоставляется 6 функций, такие как: просмотр информации о пациентах, редактирование данных пациентов, удаление пациентов из базы данных, добавление новых пациентов, поиск пациентов по возрасту и диагнозу, поиск иногородних пациентов.



*Рисунок 5 — Меню администратора*

**

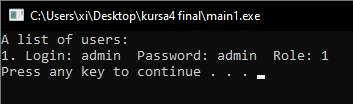
*Рисунок 6 - Панель для работы с пользователями*

**

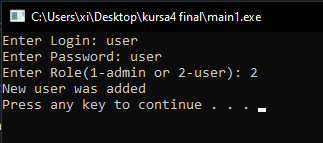
*Рисунок 7 - Панель для работы с пациентами*

**3.3.1 Возможности панели для работы с пользователями**

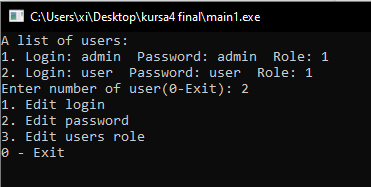
После захода в панель пользователей, администратор имеет доступ к следующим функциям: отобразить точное количество пользователей с их логинами и паролями, добавить нового пользователя и задать ему логин с паролем, редактировать уже существующего пользователя и удалить пользователя.

**

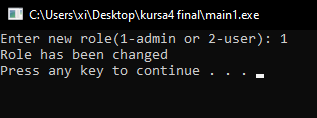
*Рисунок 8 - Отображение всех пользователей*

**

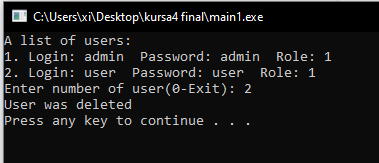
*Рисунок 9 - Создание нового пользователя*

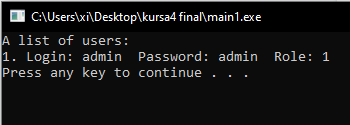
**

*Рисунок 10 - Меню редактирования данных пользователя*

**

*Рисунок 11 - Изменение роли пользователя*

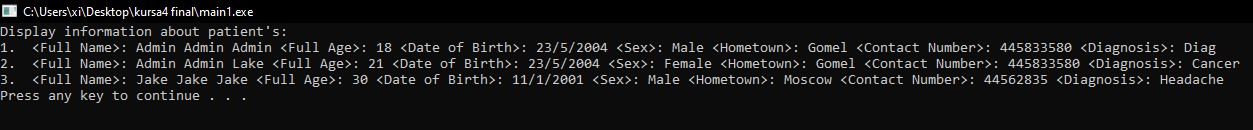
**

**

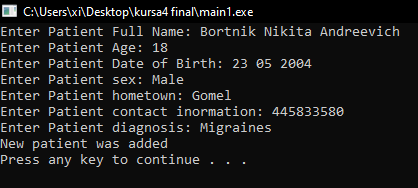
*Рисунок 12 - Система удаления пользователя*

**3.3.2 Возможности панели для работы с пациентами**

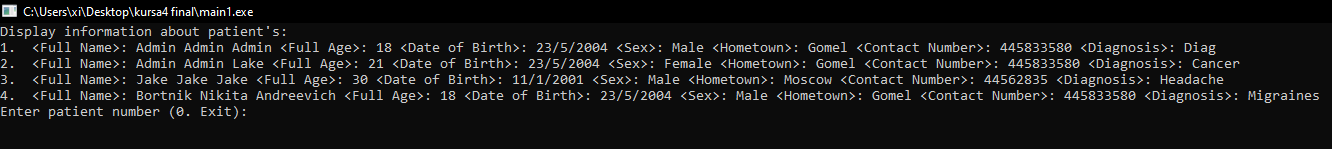
После захода в панель пациентов, администратору на выбор даются следующие функции: просмотр сведений о всех пациентах, редактирование данных пациента, добавление нового пациента, удаление пациента из базы данных, поиск пациента по возрасту и его диагнозу, поиск иногородних пациентов.

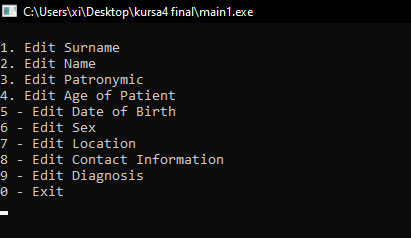


*Рисунок 13 — Отображение всех пациентов*

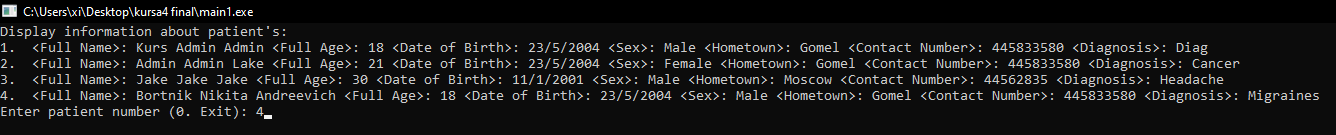


*Рисунок 14 — Cистема добавления нового пациента*

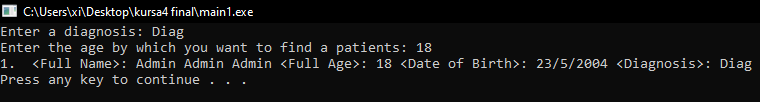
**

**

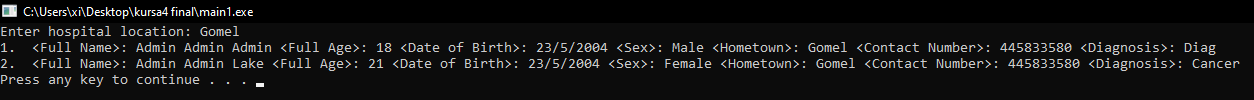
*Рисунок 15 - Система редактирования данных о пациентах*

**

*Рисунок 16 - Удаление пациента из базы данных*

**

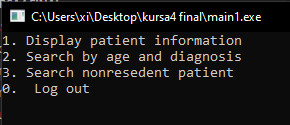
*Рисунок 17 - Поиск пациента по диагнозу и возрасту*

**

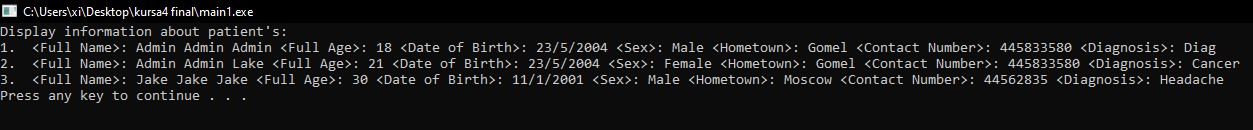
*Рисунок 18 - Поиск иногородних пациентов*

**3.4 Возможности пользователя**

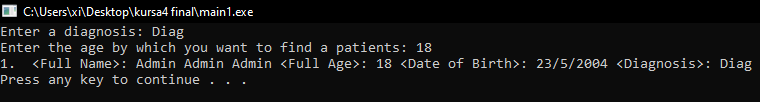
При входе в аккаунт с ролью пользователя, будет доступно всего три функции: просмотр данных о пациентах, поиск пациента по возрасту и диагнозу и поиск иногородних пациентов.



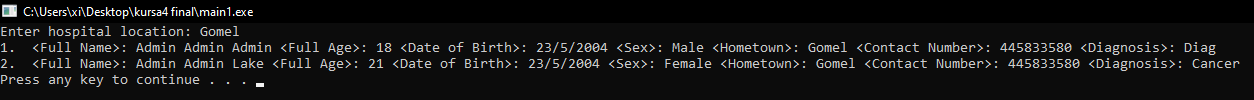
*Рисунок 19 - Меню пользователя*



*Рисунок 20 - Просмотр сведений о пациентах*

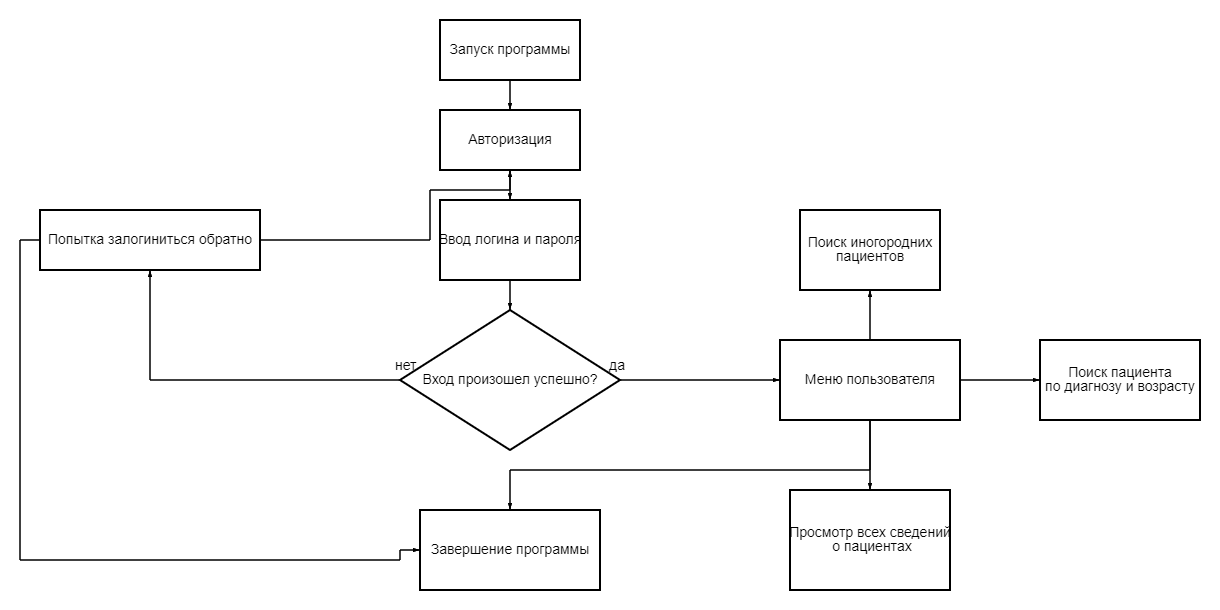
**

*Рисунок 21 - Поиск пациента по диагнозу и возрасту*

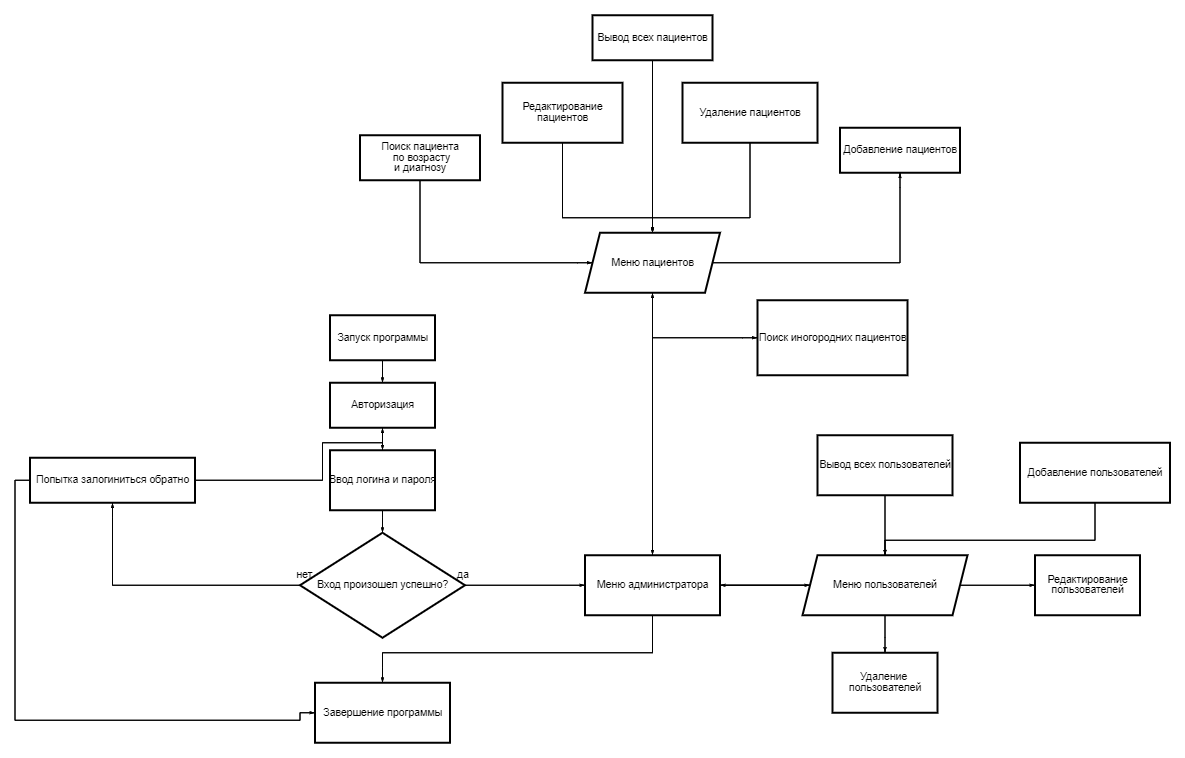
**

*Рисунок 22 - Поиск иногородних пациентов*

**3.5 Блок схемы**

****

*Рисунок 23 - Возможности пользователя*

**

*Рисунок 24 - Возможности администратора*

**Заключение**

По результату выполнения курсовой работы было разработано приложение для учета сведений о пациентах медицинского учреждения. Также были написаны дополнительные функции, которые облегчают жизнь пользователями. Была добавлена авторизация, вход в качестве администратора и пользователя. Администратор имеет функции для удобной работы с данными, он может добавлять/удалять/редактировать/смотреть пользователей и пациентов. Пользователь же имеет базовые функции такие как: просмотр данных всех пациентов, поиск по диагнозу и возрасту и поиск иногородних пациентов.

**Список использованных источников**

1. https://en.wikipedia.org/wiki/Structured\_programming
2. https://en.wikipedia.org/wiki/C\_(programming\_language)
3. https://habr.com/ru/post/464075
4. https://semenalidery.com/yazyki-programmirovaniya/nishodyashhee-i-voshodyashhee-programmirovanie.html

**Приложение**

**Листинг программы**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct patient

{

char surname[20];

char name[20];

char patronymic[20];

int age;

int dayOfBirth;

int monthOfBirth;

int yearOfBirth;

char sex[10];

char location[10];

int contactInformation;

char diagnosis[15];

} patients[100];

struct user

{

char login[15];

char password[15];

int role;

} users[50];

int sizeUsers = 0, sizePatients = 0, userNum = 0, role = 0;

void massiveNulling(char \*array, int size)

{

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

{

\*(array + i) = '\000';

}

}

void logginAgain(int \*stopmain)

{

int n, stop = 0;

while (stop == 0)

{

system("cls");

printf("1. Log in again\n0. Close programme\n");

scanf("%d", &n);

switch (n)

{

case 1:

userNum = 0;

stop = 1;

break;

case 0:

userNum = -1;

stop = 1;

\*stopmain = 1;

break;

default:

system("cls");

printf("Wrong number, please try again\n");

system("pause");

break;

}

}

}

void authorization(char \*login, char \*password)

{

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

{

if (strcmp(users[i].login, login) == 0 && strcmp(users[i].password, password) == 0)

{

userNum = i + 1;

role = users[i].role;

printf("The login was completed succesfully\n");

}

}

}

void loginSystem(int \*stopmain)

{

int n, stop;

char login[15], password[15];

while (userNum == 0)

{

system("cls");

stop = 0;

role = 0;

massiveNulling(login, 15);

massiveNulling(password, 15);

printf("Login: ");

scanf("%s", &login);

printf("Password: ");

scanf("%s", &password);

authorization(login, password);

if (userNum == 0)

{

system("cls");

printf("Wrong login or password\n");

system("pause");

logginAgain(stopmain);

}

}

if (userNum != -1)

system("pause");

}

void adminCreate()

{

sizeUsers++;

strcpy(users[0].login, "admin");

strcpy(users[0].password, "admin");

users[0].role = 1;

printf("Admin account was created.\nLogin: admin Password: admin\n");

system("pause");

}

int loadDatabase()

{

FILE \*fPatients = fopen("database.txt", "rb");

FILE \*fUsers = fopen("users.txt", "r");

if (fUsers)

{

fscanf(fUsers, "%d", &sizeUsers);

fread(&sizePatients, sizeof(int), 1, fPatients);

if (sizeUsers > 50)

{

printf("The users database is damaged\n");

system("pause");

return 1;

}

if (sizePatients > 100)

{

printf("The patients database is damaged\n");

system("pause");

return 1;

}

else

{

if (sizeUsers > 0)

{

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

fscanf(fUsers, "%s %s %d\n", &users[i].login, &users[i].password, &users[i].role);

}

if (sizePatients > 0)

fread(patients, sizeof(struct patient), sizePatients, fPatients);

}

fclose(fPatients);

fclose(fUsers);

}

else

adminCreate();

return 0;

}

void saveDatabase()

{

FILE \*fPatients = fopen("database.txt", "wb");

FILE \*fUsers = fopen("users.txt", "w");

fprintf(fUsers, "%d\n", sizeUsers);

fwrite(&sizePatients, sizeof(int), 1, fPatients);

if (sizeUsers > 0)

{

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

{

fprintf(fUsers, "%s %s %d\n", users[i].login, users[i].password, users[i].role);

}

}

if (sizePatients > 0)

{

fwrite(patients, sizeof(struct patient), sizePatients, fPatients);

}

fclose(fPatients);

fclose(fUsers);

}

void outputPatientInformation()

{

system("cls");

fflush(stdin);

if (sizePatients > 0)

{

printf("Display information about patient's: \n");

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

printf("%d. <Full Name>: %s %s %s <Full Age>: %d <Date of Birth>: %d/%d/%d <Sex>: %s <Hometown>: %s <Contact Number>: %d <Diagnosis>: %s\n", i + 1, (patients + i)->surname, (patients + i)->name, (patients + i)->patronymic, (patients + i)->age, (patients + i)->dayOfBirth, (patients + i)->monthOfBirth, (patients + i)->yearOfBirth, (patients + i)->sex, (patients + i)->location, (patients + i)->contactInformation, (patients + i)->diagnosis);

}

else

printf("There are no patients in the database\n");

}

void addNewPatient()

{

system("cls");

fflush(stdin);

printf("Enter Patient Full Name: ");

scanf("%s %s %s", &(patients + sizePatients)->surname, &(patients + sizePatients)->name, &(patients + sizePatients)->patronymic);

printf("Enter Patient Age: ");

scanf("%d", &(patients + sizePatients)->age);

printf("Enter Patient Date of Birth: ");

scanf("%d %d %d", &(patients + sizePatients)->dayOfBirth, &(patients + sizePatients)->monthOfBirth, &(patients + sizePatients)->yearOfBirth);

printf("Enter Patient sex: ");

scanf("%s", &((patients + sizePatients)->sex));

printf("Enter Patient hometown: ");

scanf("%s", &((patients + sizePatients)->location));

printf("Enter Patient contact inormation: ");

scanf("%d", &((patients + sizePatients)->contactInformation));

printf("Enter Patient diagnosis: ");

scanf("%s", &((patients + sizePatients)->diagnosis));

printf("New patient was added\n");

sizePatients++;

system("pause");

}

void editPatientInformation()

{

int n, stop = 0, n1, stop1;

while (stop == 0)

{

outputPatientInformation();

if (sizePatients > 0)

{

printf("Enter patient number (0. Exit): ");

scanf("%d", &n1);

stop1 = 0;

if ((n1 < 0) || (n1 > sizePatients))

printf("Wrong number, please try again\n");

else if (n1 == 0)

stop = 1;

else

while (stop1 == 0)

{

system("cls");

printf("\n1. Edit Surname\n2. Edit Name\n3. Edit Patronymic\n4. Edit Age of Patient\n5 - Edit Date of Birth");

printf("\n6 - Edit Sex\n7 - Edit Location\n8 - Edit Contact Information\n9 - Edit Diagnosis\n0 - Exit\n");

scanf("%d", &n);

switch (n)

{

case 1:

printf("\nEnter new surname: ");

scanf("%s", &patients[n1 - 1].surname);

printf("Patient surname has been changed\n");

break;

case 2:

printf("\nEnter new name: ");

scanf("%s", &patients[n1 - 1].name);

printf("Patient name has been changed\n");

break;

case 3:

printf("\nEnter new patronymic: ");

scanf("%s", &patients[n1 - 1].patronymic);

printf("Patient patronymic has been changed\n");

break;

case 4:

printf("\nEnter new age: ");

scanf("%s", &patients[n-1].age);

break;

case 5:

printf("\nEnter new Date of Birth: ");

scanf("%d.%d.%d", &patients[n1 - 1].dayOfBirth, &patients[n1 - 1].monthOfBirth, &patients[n1 - 1].yearOfBirth);

printf("Patient Date of Birh has been changed\n");

break;

case 6:

printf("\nEnter new sex: ");

scanf("%s", &patients[n1 - 1].sex);

printf("Patient sex has been changed\n");

break;

case 7:

printf("\nEnter new location: ");

scanf("%s", &patients[n1 - 1].location);

printf("Arrival time has been changed\n");

break;

case 8:

printf("\nEnter new contact information: ");

scanf("%d", &patients[n1 - 1].contactInformation);

printf("Contact Information has been changed\n");

break;

case 9:

printf("\nEnter new diagnosis: ");

scanf("%s", &patients[n1 - 1].diagnosis);

printf("Diagnosis has been changed\n");

break;

case 0:

stop1 = 1;

break;

default:

printf("Wrong number, please try again\n");

break;

}

}

}

else

{

stop = 1;

system("pause");

}

}

}

void deletePatient()

{

int stop = 0, n, i;

while (stop == 0)

{

outputPatientInformation();

if (sizePatients > 0)

{

printf("Enter patient number (0. Exit): ");

scanf("%d", &n);

if ((n < 0) || (n > sizePatients))

printf("Wrong number, please try again\n");

else if (n == 0)

stop = 1;

else

{

for (i = n - 1; i < sizePatients - 1; i++)

{

strcpy(patients[i].surname, patients[i+1].surname);

strcpy(patients[i].name, patients[i+1].name);

strcpy(patients[i].patronymic, patients[i+1].patronymic);

patients[i].age = patients[i + 1].age;

patients[i].dayOfBirth = patients[i + 1].dayOfBirth;

patients[i].monthOfBirth = patients[i + 1].monthOfBirth;

patients[i].yearOfBirth = patients[i + 1].yearOfBirth;

strcpy(patients[i].sex, patients[i+1].sex);

strcpy(patients[i].location, patients[i+1].location);

patients[i].contactInformation = patients[i + 1].contactInformation;

strcpy(patients[i].diagnosis, patients[i+1].diagnosis);

}

sizeUsers--;

printf("Patient deleted successfully\n");

}

}

else

{

stop = 1;

system("pause");

}

}

}

void searchByAgeAndDiagnosis()

{

char diagnosis[30];

int i, age;

system("cls");

if (sizePatients == 0)

printf("There are no patients in the database\n");

else

{

printf("Enter a diagnosis: ");

scanf("%s", &diagnosis);

printf("Enter the age by which you want to find a patients: ");

scanf("%d", &age);

for (i = 0; i < sizePatients; i++)

{

if (strcmp((patients + i)->diagnosis, diagnosis) == 0 && patients[i].age == age)

{

printf("%d. <Full Name>: %s %s %s <Full Age>: %d <Date of Birth>: %d/%d/%d <Diagnosis>: %s\n", i + 1, (patients + i)->surname, (patients + i)->name, (patients + i)->patronymic, (patients + i)->age, (patients + i)->dayOfBirth, (patients + i)->monthOfBirth, (patients + i)->yearOfBirth, (patients + i)->diagnosis);

}

}

system("pause");

}

}

void searchNonResedentPatient()

{

char location[20];

system("cls");

if (sizePatients == 0)

printf("There are no patients in the database\n");

else

{

printf("Enter hospital location: ");

scanf("%s", &location);

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

{

if (strcmp(patients[i].location, location) == 0)

{

printf("%d. <Full Name>: %s %s %s <Full Age>: %d <Date of Birth>: %d/%d/%d <Sex>: %s <Hometown>: %s <Contact Number>: %d <Diagnosis>: %s\n", i + 1, (patients + i)->surname, (patients + i)->name, (patients + i)->patronymic, (patients + i)->age, (patients + i)->dayOfBirth, (patients + i)->monthOfBirth, (patients + i)->yearOfBirth, (patients + i)->sex, (patients + i)->location, (patients + i)->contactInformation, (patients + i)->diagnosis);

}

}

}

}

void outputUsers()

{

system("cls");

printf("A list of users:\n");

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

printf("%d. Login: %s Password: %s Role: %d\n", i + 1, (users + i)->login, (users + i)->password, (users + i)->role);

}

void addNewUser()

{

int i, stop = 0;

while (stop == 0)

{

system("cls");

stop = 1;

printf("Enter Login: ");

scanf("%s", &((users + sizeUsers)->login));

for (i = 0; i < sizeUsers; i++)

{

if (strcmp((users + sizeUsers)->login, (users + i)->login) == 0)

{

printf("This login already exists\n");

stop = 0;

system("pause");

break;

}

}

}

if (stop == 1)

{

printf("Enter Password: ");

scanf("%s", &((users + sizeUsers)->password));

while (stop == 1)

{

printf("Enter Role(1-admin or 2-user): ");

scanf("%d", &((users + sizeUsers)->role));

if ((users + sizeUsers)->role == 1 || (users + sizeUsers)->role == 2)

{

printf("New user was added\n");

sizeUsers++;

stop = 0;

system("pause");

}

else

{

printf("Wrong number of role, please try again\n");

system("pause");

}

}

}

}

void editUsers()

{

int stop = 0, n, stop1, n1, stop2;

while (stop == 0)

{

outputUsers();

stop1 = 0;

stop2 = 0;

printf("Enter number of user(0-Exit): ");

scanf("%d", &n);

if ((n < 0) || (n > sizeUsers))

printf("Wrong number, please try again\n");

else if (n == 0)

stop = 1;

else

while (stop1 == 0)

{

printf("1. Edit login\n2. Edit password\n3. Edit users role\n0 - Exit\n");

scanf("%d", &n1);

switch (n1)

{

case 1:

system("cls");

printf("Enter new login: ");

scanf("%s", &(users + n - 1)->login);

printf("Login has been changed\n");

system("pause");

break;

case 2:

system("cls");

printf("Enter new password: ");

scanf("%s", &(users + n - 1)->password);

printf("Password has been changed\n");

system("pause");

break;

case 3:

while (stop2 == 0)

{

system("cls");

printf("Enter new role(1-admin or 2-user): ");

scanf("%d", &(users + n - 1)->role);

if ((users + n - 1)->role == 1 || (users + n - 1)->role == 2)

stop2 = 1;

else

printf("Wrong number, please try again\n");

}

printf("Role has been changed\n");

system("pause");

break;

case 0:

stop1 = 1;

break;

default:

system("cls");

printf("Wrong number, please try again\n");

system("pause");

break;

}

}

}

}

void deleteUsers()

{

int stop = 0, n;

while (stop == 0)

{

system("cls");

outputUsers();

printf("Enter number of user(0-Exit): ");

scanf("%d", &n);

if ((n < 0) || (n > sizeUsers))

printf("Wrong number, please try again\n");

else if (n == 0)

stop = 1;

else if (n == userNum)

{

printf("You can't delete an active user\n");

system("pause");

}

else

{

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

{

strcpy((users + i)->login, (users + i + 1)->login);

strcpy((users + i)->password, (users + i + 1)->password);

(users + i)->role = (users + i + 1)->role;

}

sizeUsers--;

if (userNum > n)

userNum--;

printf("User was deleted\n");

system("pause");

}

}

}

void userInteraction()

{

int stop = 0, n;

while (stop == 0)

{

system("cls");

printf("1. Display all users\n2. Add new user\n3. Edit user\n4. Delete user\n0. Exit\n");

scanf("%d", &n);

switch (n)

{

case 1:

outputUsers();

system("pause");

break;

case 2:

addNewUser();

break;

case 3:

editUsers();

break;

case 4:

deleteUsers();

break;

case 0:

stop = 1;

break;

default:

printf("Wrong number, please try again\n");

system("pause");

break;

}

}

}

void patientInteraction()

{

int stop = 0, n;

while (stop == 0)

{

system("cls");

printf("1. Display Patient Information\n2. Add new Patient\n3. Edit Patient information\n4. Delete Patient from database\n5. Search by age and diagnosis\n6. Search nonresedent patient\n0. Exit\n");

scanf("%d", &n);

switch (n)

{

case 1:

outputPatientInformation();

system("pause");

break;

case 2:

addNewPatient();

break;

case 3:

editPatientInformation();

system("pause");

case 4:

deletePatient();

break;

case 5:

searchByAgeAndDiagnosis();

break;

case 6:

searchNonResedentPatient();

system("pause");

case 0:

stop = 1;

break;

default:

printf("Wrong number, please try again\n");

system("pause");

break;

}

}

}

void userMenu() // Меню возможностей обычного пользователя

{

int stop = 0, n;

while (stop == 0)

{

system("cls");

printf("1. Display patient information\n2. Search by age and diagnosis\n3. Search nonresedent patient\n0. Log out\n");

scanf("%d", &n);

switch (n)

{

case 1:

outputPatientInformation();

system("pause");

break;

case 2:

searchByAgeAndDiagnosis();

break;

case 3:

searchNonResedentPatient();

system("pause");

case 0:

stop = 1;

userNum = 0;

break;

default:

printf("Wrong number, please try again\n");

break;

}

}

}

void adminMenu() // Меню возможностей админа

{

int stop = 0, n;

while (stop == 0)

{

system("cls");

printf("1. Users Panel\n2. Patients Panel\n0. Log out\n");

scanf("%d", &n);

switch (n)

{

case 1:

userInteraction();

break;

case 2:

patientInteraction();

break;

case 0:

stop = 1;

userNum = 0;

break;

default:

printf("Wrong number, please try again\n");

system("pause");

break;

}

}

}

int main()

{

int stopmain = 0;

if (loadDatabase() == 0)

{

while (stopmain == 0)

{

system("cls");

loginSystem(&stopmain);

if (role == 2)

userMenu();

else if (role == 1)

adminMenu();

if (userNum == 0)

logginAgain(&stopmain);

}

saveDatabase();

}

return 0;

}