Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Конструирование программ и языки программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

СИСТЕМА ПРОВЕРКИ ПАСПОРТОВ ГРАЖДАН

БГУИР КП 1-40 02 01 119 ПЗ

Студент: гр. 650501 Новицкий А.Д.

Руководитель: старший преподаватель Ковальчук А. М.

Минск 2017

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой

*––––––––––––––––––––––––*

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту    *Новицкому Андрею Дмитриевичу –––––––––––––––––––*

1. Тема проекта   Система проверки поспортов граждан –

2. Срок сдачи студентом законченного проекта–––––*21 декабря 2017 г.––––*

3. Исходные данные к проекту *текстовые файлы с областями, районами, городами и их кодами; бинарные файлы с закодированной информациях о гражданах; запись информации в файл; среда разработки – QT Creator(кросс-платформенная)–*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение. 1. Постановка задачи. 2. Структура входных и выходных данных. 3. Описание алгоритмов. 4.Разработка программного обеспечения. 5. Результаты работы программы. 6. Руководство пользователя. Заключение. Литература. Приложения.*

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*1. Диаграмма классов 2. Схема алгоритмов (2) шт – чертёж*

6. Консультант по проекту (с обозначением разделов проекта)   *Ковальчук А. М.*

7. Дата выдачи задания –––––*15 сентября 2017 г.–––––––––––––––––––––––*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

*разделы 1 к 10.10 – 15 %;–––––––––––––––––––––– –––––––––––––––––*

*раздел 2 к 01.11 – 15 %;–––––––––––––––––––––– –––––––––––––––––––*

*разделы 3,4 к 20.11 – 45 %;––––––––––––––––––––– –––––––––––––––––––*

*разделы 5, 6 к 01.12 – 10 %;––––––––––––––––––––– –––––––––––––––––––*

*оформление пояснительной записки и графического материала к 04.12 – 15 %*

*Защита курсового проекта с 21 по 29 декабря––––––––––––––––––––––––––*

РУКОВОДИТЕЛЬ*––––––––– А. М. Ковальчук*

(подпись)

Задание принял к исполнению *–––––– –\_\_ \_\_\_\_\_\_\_\_–– А.Д,Новицкий*

(дата и подпись студента)

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 5

1 ПОСТАНОВКА ЗАДАЧИ 6

2 СТРУКТУРА ВХОДНЫХ И ВЫХОДНЫХ ДАННЫХ 7

3 ОПИСАНИЕ АЛГОРИТМОВ 17

4 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 18

5 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ 19

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 20

ЗАКЛЮЧЕНИЕ 26

ЛИТЕРАТУРА 27

ПРИЛОЖЕНИЕ А 28

ПРИЛОЖЕНИЕ Б 29

ПРИЛОЖЕНИЕ В 30

ПРИЛОЖЕНИЕ Г 36

ПРИЛОЖЕНИЕ Д 85

ПРИЛОЖЕНИЕ Е 90

**ВВЕДЕНИЕ**

Язык C++ — [компилируемый](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), [статически типизированный](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения.

Поддерживает такие [парадигмы программирования](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D1%8B_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование.

Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как [высокоуровневых](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), так и [низкоуровневых языков](https://ru.wikipedia.org/wiki/%D0%9D%D0%B8%D0%B7%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). В сравнении с его предшественником — языком [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), — наибольшее внимание уделено поддержке [объектно-ориентированного](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [обобщённого программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

Язык C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования.

Область его применения включает создание [операционных систем](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), разнообразных прикладных программ, [драйверов](https://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%B0%D0%B9%D0%B2%D0%B5%D1%80) устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (игр).

Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, [GCC](https://ru.wikipedia.org/wiki/GNU_Compiler_Collection), [Visual C++](https://ru.wikipedia.org/wiki/Visual_C%2B%2B), [Intel C++ Compiler](https://ru.wikipedia.org/wiki/Intel_C%2B%2B_Compiler), [Embarcadero (Borland) C++ Builder](https://ru.wikipedia.org/wiki/Embarcadero_C%2B%2B_Builder) и другие.

Язык C++ оказал огромное влияние на другие языки программирования, в первую очередь на [Java](https://ru.wikipedia.org/wiki/Java) и [C#](https://ru.wikipedia.org/wiki/C_Sharp).

1. **ПОСТАНОВКА ЗАДАЧИ**

Темой данного курсового проекта является «Система проверки паспортов граждан».

Данная ситема содержит программное обеспечение для персонального компьютера.

Для демонстрации возможностей данной системы был проеведен сбор данных граждан, составлен перечень областей, районов и городов.

Тема является актуальной, так как внедрение электронных ситем проверки является первостепенной задачей нашего государсва.

*Цели* курсового проектирования: овладеть практическими навыками проектирования и разработки законченного, отлаженного и протестированного программного продукта с использованием языка высокого уровня С++;

*Задача* курсового проектирования – проектирование прикладного программного обеспечения.

**2 СТРУКТУРА ВХОДНЫХ И ВЫХОДНЫХ ДАННЫХ**

Входными данным для программного обеспечения персонального компьютера являются 12 бинарных одинаковой структуры, которые содержат информацию гражданхт, 6 текстовых файлов одинаковой структуры, которые содержат назвния областей, районов, городов, имена, фамилии, отчества и их кодировку, 2 текстовых файла одинаковой структуры содержащие информацию о пользователях (рисунки 1, 2, 4, 5, 6, 7).

Файл *AdminAcc.txt*

|  |  |
| --- | --- |
| Логин: | Пароль: |
| AnjeyNov | 50bunila |

Файл *UserAcc.txt*

|  |  |
| --- | --- |
| Логин: | Пароль: |
| user | qwerty123 |

Файл *Districts.txt*

|  |  |
| --- | --- |
| Параметр: | Код: |
| Брестская | 1 |
| Витебская | 2 |
| Гомельская | 3 |
| Гродненская | 4 |
| Минская | 5 |
| Могилевская | 6 |

Рисунок 1 – Входные данные для ПО персонального компьютера

Файл *Regions.txt*

|  |  |
| --- | --- |
| Параметр: | Код: |
| Барановический | 1 |
| Березовский | 2 |
| Бобруйский | 3 |
| Браславский | 4 |
| Брестский | 5 |
| Витебский | 6 |

Файл *Towns.txt*

|  |  |
| --- | --- |
| Параметр: | Код: |
| Орша | 1 |
| Ивье | 2 |
| Бобруйск | 3 |
| Минск | 4 |
| Холхлово | 5 |
| Кличев | 6 |

Файл *Familis.txt*

|  |  |
| --- | --- |
| Параметр: | Код: |
| Шутов | 1 |
| Захаренко | 2 |
| Арбузов | 3 |
| Бородин | 4 |
| Войтка | 5 |
| Зинькевич | 6 |

Рисунок 2 – Входные данные для ПО персонального компьютера

Файл *Names.txt*

|  |  |
| --- | --- |
| Параметр: | Код: |
| Владимир | 1 |
| Артем | 2 |
| Тимофей | 3 |
| Дмитрий | 4 |
| Владислав | 5 |
| Алексей | 6 |

Файл *Surnames.txt*

|  |  |
| --- | --- |
| Параметр: | Код: |
| Александрович | 1 |
| Игоревич | 2 |
| Витальевич | 3 |
| Андреевич | 4 |
| Алексеевич | 5 |
| Вадимович | 6 |

Файл *year1900.dat*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кода фамилии | Код имени | Код отчества | Код области | Код района | Код города | День рождения | Месяц рождения | Год рождения | ID | Номер паспорта |
|  |  |  |  |  |  |  |  |  |  |  |

Рисунок 3– Входные данные для ПО персонального компьютера

Файл *year1910.dat*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кода фамилии | Код имени | Код отчества | Код области | Код района | Код города | День рождения | Месяц рождения | Год рождения | ID | Номер паспорта |
|  |  |  |  |  |  |  |  |  |  |  |

Файл *year1920.dat*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кода фамилии | Код имени | Код отчества | Код области | Код района | Код города | День рождения | Месяц рождения | Год рождения | ID | Номер паспорта |
|  |  |  |  |  |  |  |  |  |  |  |

Файл *year1930.dat*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кода фамилии | Код имени | Код отчества | Код области | Код района | Код города | День рождения | Месяц рождения | Год рождения | ID | Номер паспорта |
|  |  |  |  |  |  |  |  |  |  |  |

Рисунок 4 – Входные данные для ПО персонального компьютера

Файл *year1940.dat*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кода фамилии | Код имени | Код отчества | Код области | Код района | Код города | День рождения | Месяц рождения | Год рождения | ID | Номер паспорта |
|  |  |  |  |  |  |  |  |  |  |  |

Файл *year1950.dat*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кода фамилии | Код имени | Код отчества | Код области | Код района | Код города | День рождения | Месяц рождения | Год рождения | ID | Номер паспорта |
|  |  |  |  |  |  |  |  |  |  |  |

Файл *year1960.dat*

Рисунок 5 – Входные данные для ПО персонального компьютера

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кода фамилии | Код имени | Код отчества | Код области | Код района | Код города | День рождения | Месяц рождения | Год рождения | ID | Номер паспорта |
|  |  |  |  |  |  |  |  |  |  |  |

Файл *year1970.dat*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кода фамилии | Код имени | Код отчества | Код области | Код района | Код города | День рождения | Месяц рождения | Год рождения | ID | Номер паспорта |
|  |  |  |  |  |  |  |  |  |  |  |

Файл *year1980.dat*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кода фамилии | Код имени | Код отчества | Код области | Код района | Код города | День рождения | Месяц рождения | Год рождения | ID | Номер паспорта |
|  |  |  |  |  |  |  |  |  |  |  |

Файл *year1990.dat*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кода фамилии | Код имени | Код отчества | Код области | Код района | Код города | День рождения | Месяц рождения | Год рождения | ID | Номер паспорта |
| 1 | 1 | 1 | 3 | 36 | 38 | 2 | 5 | 1998 | 3020598E004PB9 | BM2179273 |

Рисунок 6 – Входные данные для ПО персонального компьютера

Файл *year2000.dat*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кода фамилии | Код имени | Код отчества | Код области | Код района | Код города | День рождения | Месяц рождения | Год рождения | ID | Номер паспорта |
|  |  |  |  |  |  |  |  |  |  |  |

Файл *year2010.dat*

Рисунок 7 – Входные данные для ПО персонального компьютера

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кода фамилии | Код имени | Код отчества | Код области | Код района | Код города | День рождения | Месяц рождения | Год рождения | ID | Номер паспорта |
|  |  |  |  |  |  |  |  |  |  |  |

Выходными данными программного обеспечения персонального компьютера являются текстовые файлы, которые содержат измененную информацию о пользователях, областях, районах, городах, фамилиях, именах, отчествах, бинарные файлы с измененной информацией о гражданах (рисунки 8, 9, 10, 11).

Файл *AdminAcc.txt*

|  |  |
| --- | --- |
| Логин: | Пароль: |
| AnjeyNov | 50bunila |
| admin | password |

Рисунок 8 – Выходные данные ПО персонального компьютера

Файл *UserAcc.txt*

|  |  |
| --- | --- |
| Логин: | Пароль: |
| user | qwerty123 |
| viza | qwerty123 |

Файл *Regions.txt*

|  |  |
| --- | --- |
| Параметр: | Код: |
| Барановический | 1 |
| Березовский | 2 |
| Бобруйский | 3 |
| Браславский | 4 |
| Брестский | 5 |
| Витебский | 6 |
| Волковысский | 7 |

Файл *Towns.txt*

|  |  |
| --- | --- |
| Параметр: | Код: |
| Орша | 1 |
| Ивье | 2 |
| Бобруйск | 3 |
| Минск | 4 |
| Холхлово | 5 |
| Кличев | 6 |
| Брест | 7 |

Рисунок 9 – Выходные данные ПО персонального компьютера

Файл *Familis.txt*

|  |  |
| --- | --- |
| Параметр: | Код: |
| Шутов | 1 |
| Захаренко | 2 |
| Арбузов | 3 |
| Бородин | 4 |
| Войтка | 5 |
| Зинькевич | 6 |
| Кравченя | 7 |

Файл *Names.txt*

|  |  |
| --- | --- |
| Параметр: | Код: |
| Владимир | 1 |
| Артем | 2 |
| Тимофей | 3 |
| Дмитрий | 4 |
| Владислав | 5 |
| Алексей | 6 |
| Евгений | 7 |

Рисунок 10 – Выходные данные ПО персонального компьютера

Файл *Surnames.txt*

|  |  |
| --- | --- |
| Параметр: | Код: |
| Александрович | 1 |
| Игоревич | 2 |
| Витальевич | 3 |
| Андреевич | 4 |
| Алексеевич | 5 |
| Вадимович | 6 |
| Михайлович | 7 |

Файл *year1990.dat*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кода фамилии | Код имени | Код отчества | Код области | Код района | Код города | День рождения | Месяц рождения | Год рождения | ID | Номер паспорта |
| 1 | 1 | 1 | 3 | 36 | 38 | 2 | 5 | 1998 | 3020598E004PB9 | BM2179273 |
| 2 | 2 | 2 | 3 | 25 | 1 | 4 | 2 | 1999 | 3130898Е010РВ7 | ВМ 2187271 |

Рисунок 11 – Выходные данные ПО персонального компьютера

**3 ОПИСАНИЕ АЛГОРИТМОВ**

Для построения диаграммы классов использовался язык UML.

UML— [язык](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) [графического](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) описания для [объектного моделирования](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) в области [разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), [моделирования бизнес-процессов](https://ru.wikipedia.org/w/index.php?title=%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1), [системного проектирования](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и отображения [организационных структур](https://ru.wikipedia.org/wiki/%D0%9E%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0).

UML является языком широкого профиля, это — [открытый стандарт](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82), использующий графические обозначения для создания [абстрактной модели](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) [системы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна [генерация кода](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%BE%D0%B3%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F).

Разработанная диаграмма классов находится в Приложении А.

Описание алгоритма работы функции отмены добавления (void insert\_in\_file\_int()) и функции поиска кода (int search\_code(QString parametr)) находятся в Приложении Б (чертёж 1 и чертёж 2 соответственно).

Описание алгоритмов по шагам находятся в Приложении В.

**4 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

Для разработки программного обеспечения для персонального компьютера воспользуемся средой разработки QT Creator (рисунок 15).

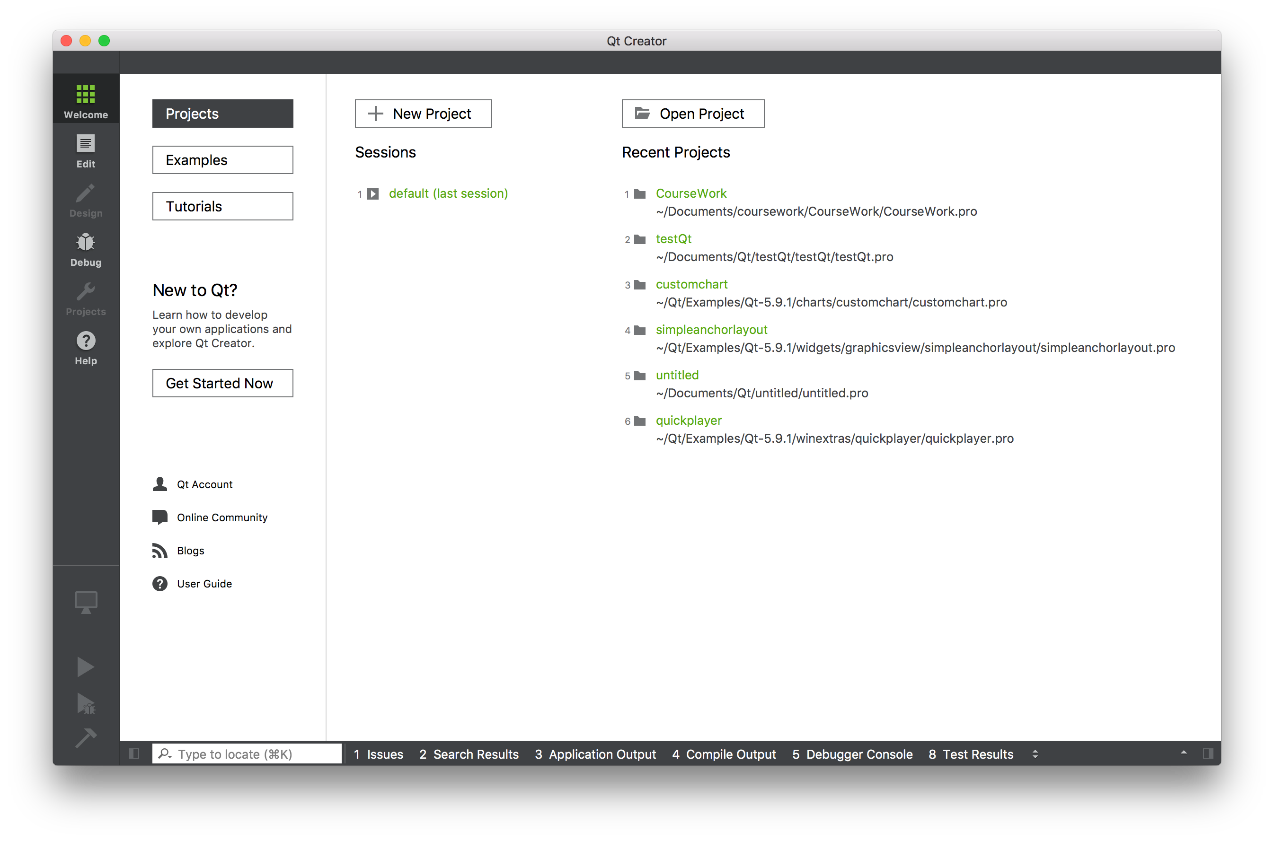


Рисунок 15 ‒ Среда разработки QT Creator

QT позволяет запускать написанное с его помощью [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) в большинстве современных [операционных систем](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) путём простой компиляции программы для каждой системы без изменения [исходного кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4).

Включает в себя все основные [классы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), которые могут потребоваться при разработке [прикладного программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Является полностью объектно-ориентированным.

Расширение возможностей обеспечивается системой плагинов, которые возможно размещать непосредственно в панели визуального редактора. Также существует возможность расширения привычной функциональности [виджетов](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B6%D0%B5%D1%82), связанной с размещением их на экране, отображением, перерисовкой при изменении размеров окна.

Листинг программного обеспечения для персонального компьютера находится в Приложении Г.

**5 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ**

В результате работы программы данные, введенные пользователем, обрабатываются и пользователь получает уведомление о их достоверности.

Также данные поступившые от администратора заносятся в файлы для дальнейшего использования.

Скриншоты работы программы находятся в Приложении Е.

**6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

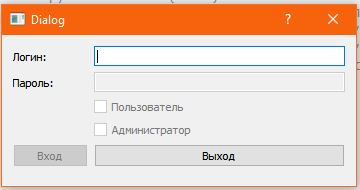
Для запуска программы необходимо открыть файл «Republican\_system\_of\_registration\_of\_citizens.exe». После этого откроется окно программы, где необходимо авторизоваться (рисунок 16).

Рисунок 16 ‒ Окно авторизации

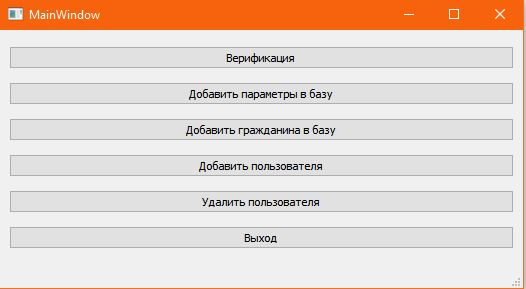
После успешной авторизации откроется окно с главным меню, где возможно выбрать необходимые действия в зависимости от прав доступа (рисунок 17).

Рисунок 17 – Главное меню

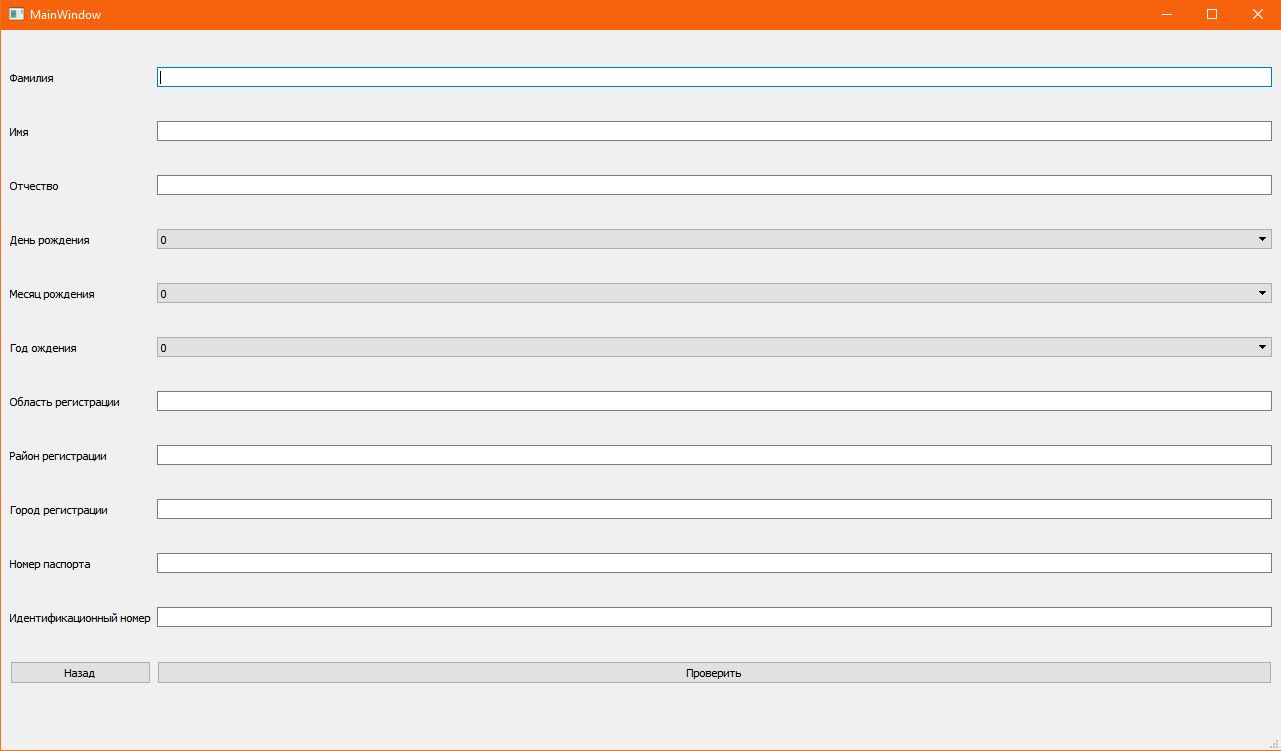
 Для того чтобы произвести верификацию, необходимо нажать кнопку «Верификация», после чего будет открыто окно с полями для ввода необходимой информации о гражданине (рисунок 18).

Рисунок 18 – Главное меню

После ввода информации и нажатия кнопки «Проверить», будет выведено уведомление о резултате верефикации (рисунок 19 а., б.).

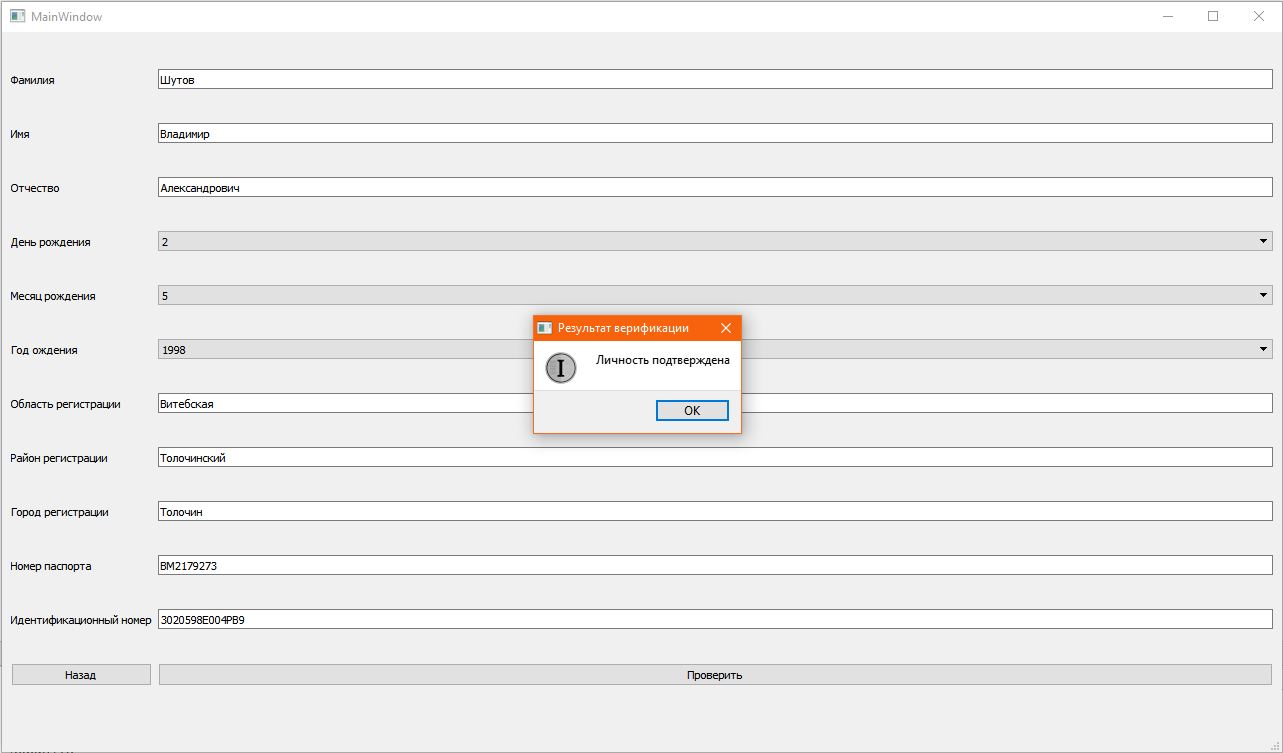


Рисунок 19 а. – Положительный результат

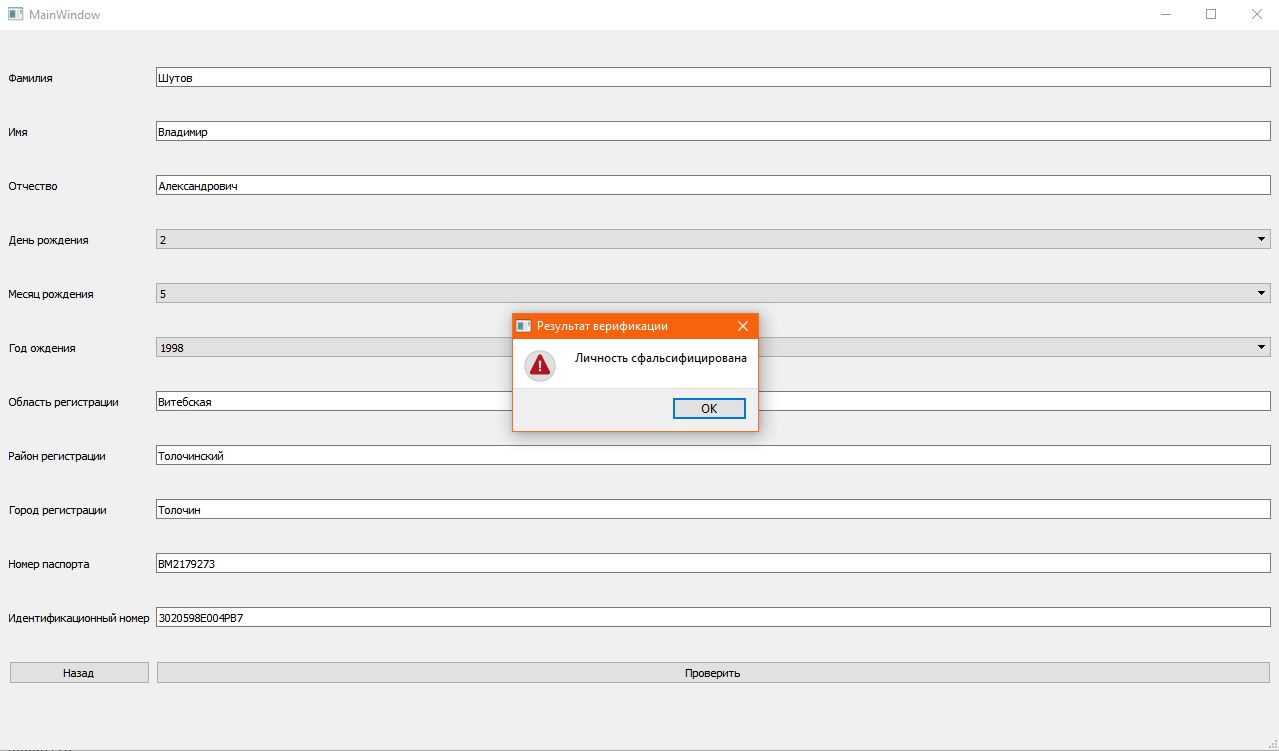


Рисунок 19 б. – Отрицательный результат

Для того чтобы добавить параметры в базу, небходимо нажать клавишу «Добавить параметры в базу» после чего откроется окно для добавления параметров, в котором необходимо следовать указанным инструкциям (рисунок 20).

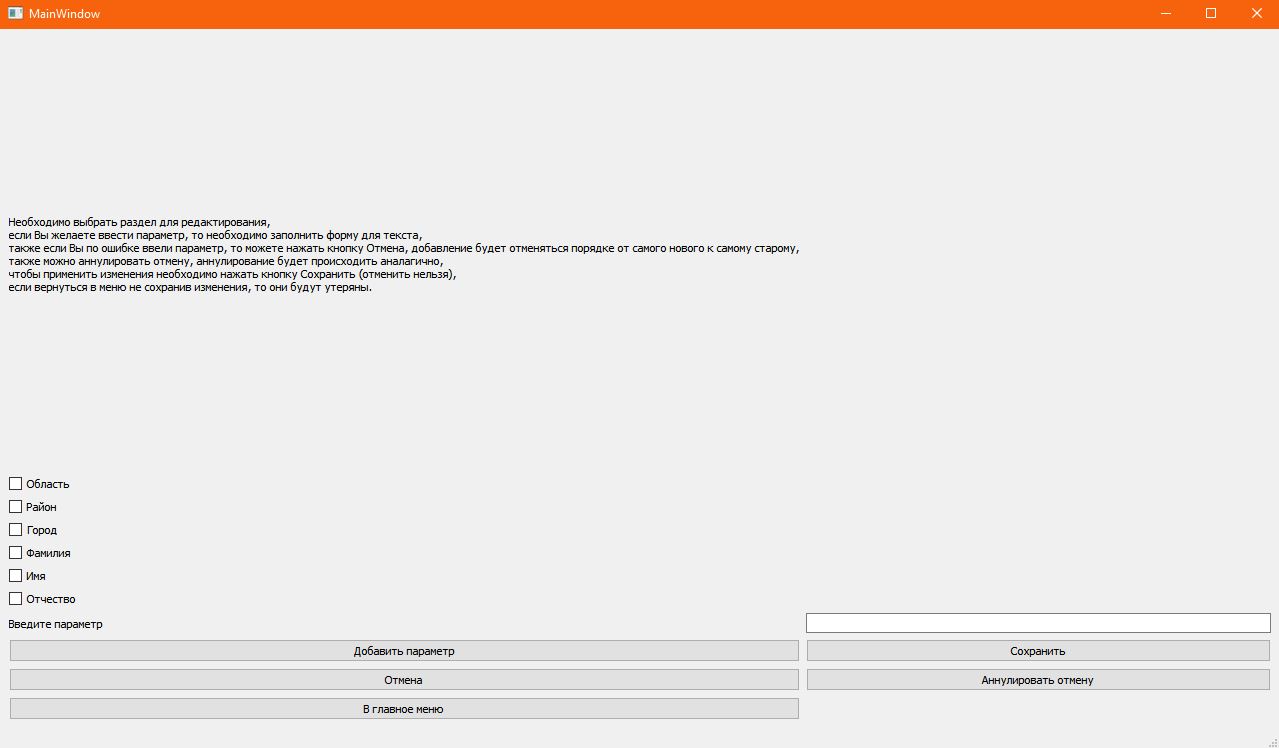


Рисунок 20 – Окно добавления параметра

Для того чтобы добавить гражданина в базу, необходимо нажать кнопку «Добавить гражданина в базу» после чего будет открыто окно для ввода информации о гражданине (рисунок 21).

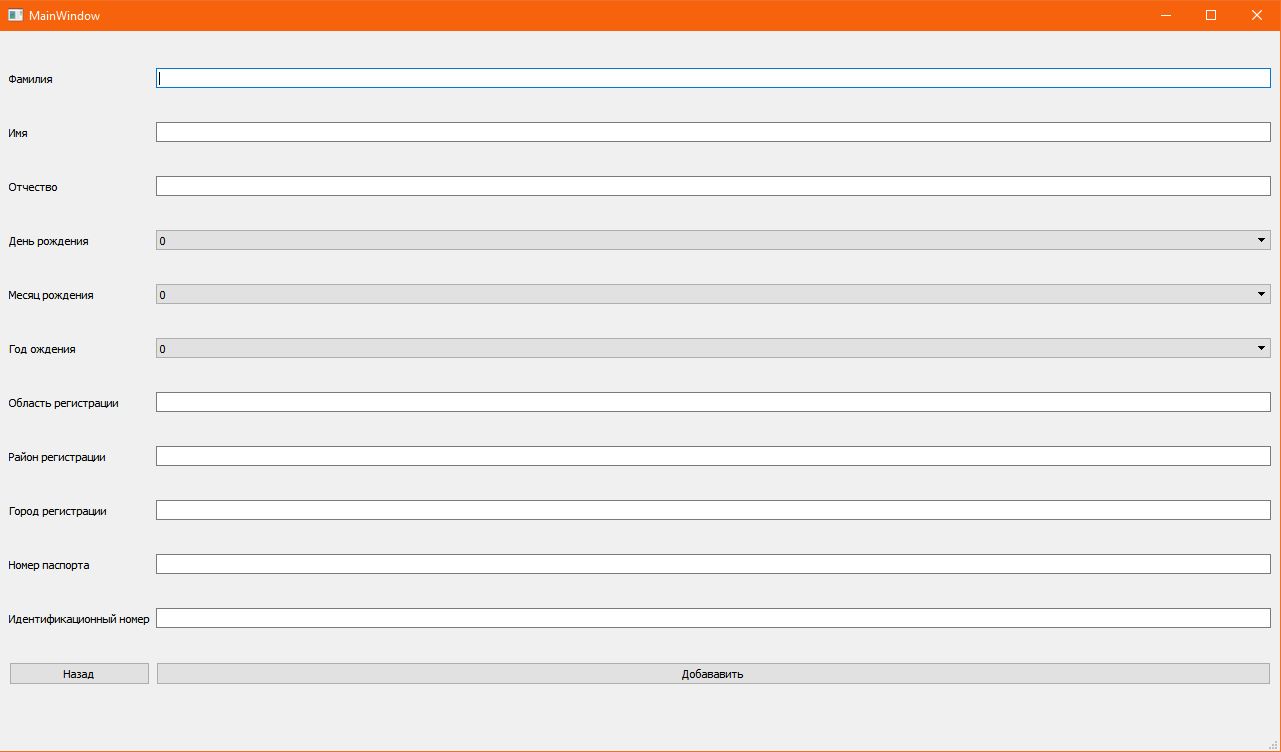


Рисунок 21 – Окно добавления гражданина

Для того чтобы добавить пользователя, необходимо нажать кнопку «Добавить пользователя» после чего откроется окно для ввода информации о пользователе и выбора типа доступа, где необходимо следовать инструкциям (рисунок 22).

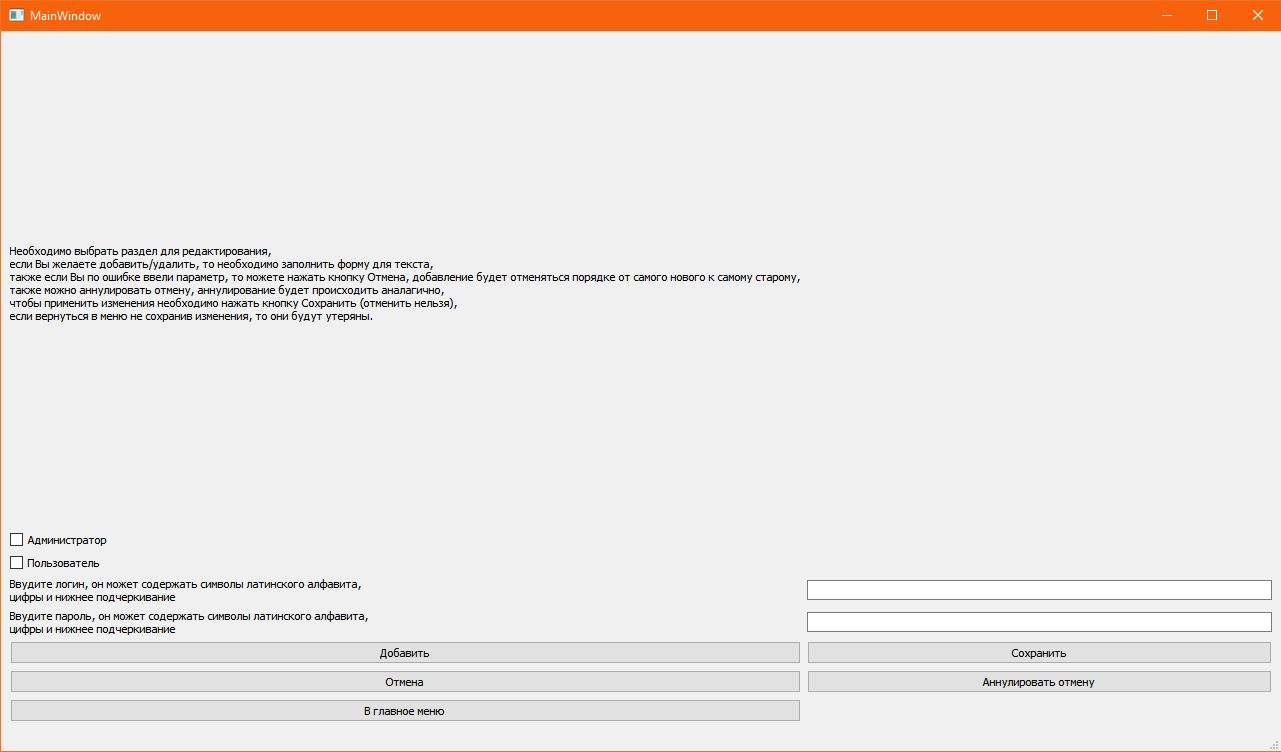


Рисунок 22 – Окно добавления пользователя

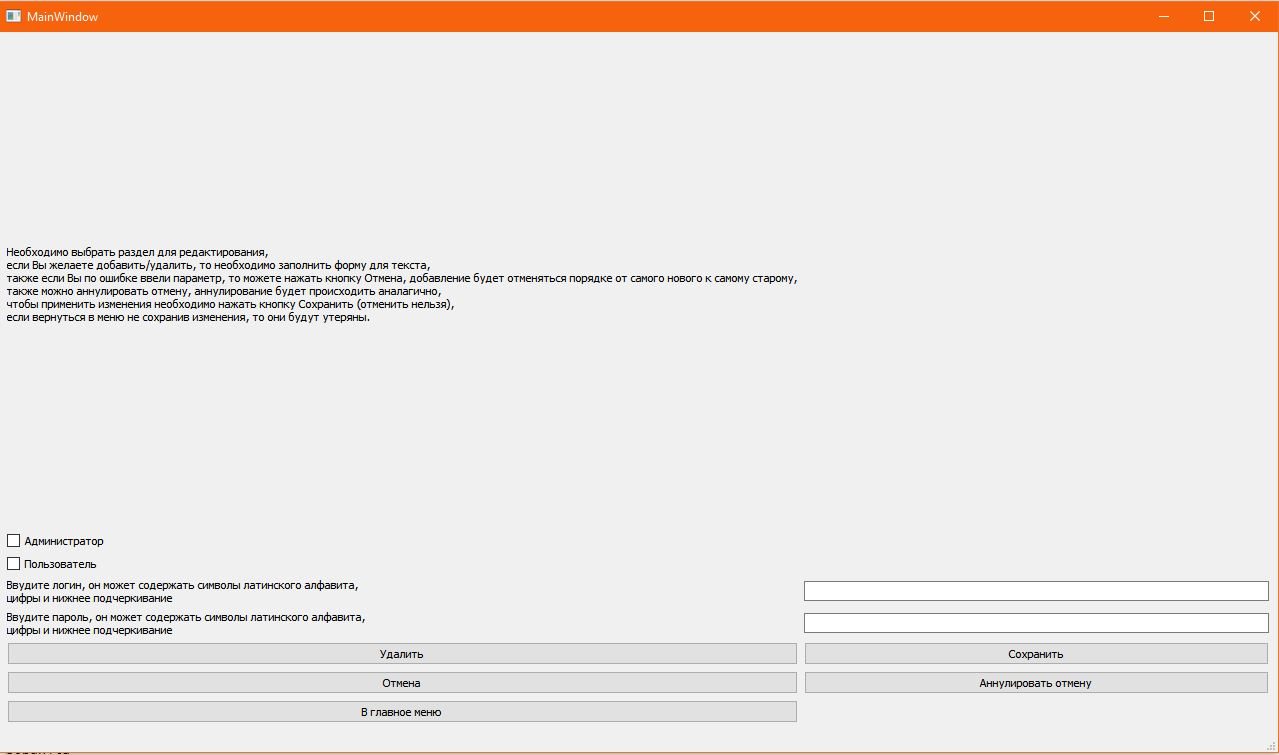
 Для того чтобы удалить пользователя, необходимо нажать кнопку «Удалить пользователя» после чего откроется окно для ввода информации о пользователе и выбора типа доступа, где необходимо следовать инструкциям (рисунок 23).

Рисунок 23 – Окно удаления пользователя

Для выхода из программы необходимо нажать кнопку «Выход».

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсового пректа было получено программное обеспечение для персонального компьютера.

Разработанная систма позволяет проверять достоверность данных о гражданине, добавлять граждан, названия областей, районов и городов, а также отчества, имена и фамилии, добавлять и удалять пользователей. Она имеент несколько режимов работы: «Пользователь», позволяющий проверять достоверность данных, и «Администратор», имеющий расширенный режим работы. Данная система кодирует все данные о гражданах и заносит их в бинарные файлы. Данная система поддерживает только русскойязычный интерфейс.

Для разработки использовалась кроссплатформенная среда Qt Creator.

Дальнейшая разработка системы должна включать расширение пользовательского интерфейса и разработку более совершенных алгоритмов поиска, кодровки и раскодировки.

**ЛИТЕРАТУРА**

1. Язык программирования С++ [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/C%2B%2B>. – Дата доступа 20.10.2017.

2. QT – кроссплатформенный фреймворк [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Qt>. Дата доступа 22.10.2017.

3. Луцик, Ю. А. Объектно-ориентированное программирование на языке С++: учеб. пособие по курсу «Объектно-ориентированное программирование» для студ. спец. «Вычислительные машины, системы и сети» всех форм обуч. / Ю. А. Луцик, А. М. Ковальчук, И. В. Лукьянова. – Минск: БГУИР, 2003. – 203 с.:ил.

4. Луцик, Ю. А. Объектно-ориентированное программирование на языке С++ : учеб. пособие / Ю. А. Луцик, В. Н. Комличенко. – Мн. : БГУИР, 2008. – 266 с. : ил.

5. Официальная документация для QT Creator [Электронный ресурс]. – Режим доступа: http://doc.qt.io/qt-5/. Дата доступа 19.11.2017.

**ПРИЛОЖЕНИЕ А**

*(обязательное)*

Диаграмма классов

**ПРИЛОЖЕНИЕ Б**

*(обязательное)*

Описание алгоритмов (блок- схемы)

**ПРИЛОЖЕНИЕ В**

*(обязательное)*

Описание алгоритмов (по шагам)

Описание алгоритма по шагам для функции аннулирования отмены (void cancellation\_cancellation()).

Шаг 1. Начало.

Шаг 2. Входные данные: ParametrList \*lise(контейнер для добавления информации), Node \*head\_of\_title, \*tail\_of\_title, \*head\_of\_title\_new, \*tail\_of\_title\_new, \*head\_of\_title\_buffer, \*tail\_of\_title\_buffer(адреса начала и конца списков исходных, новых и буферных параметров соответственно).

Шаг 3. Если значение tail\_of\_title\_buffer равно NULL, то перейти к шагу 4.

Шаг 4. Выполнить статический метод QMessageBox::warning с параметрами ,"Отказано","В данном разделе отсутствуют параметры для аннулирования отмены".

Шаг 5. Перейти на шаг 35.

Шаг 6. Если значение tail\_of\_title\_buffer равно head\_of\_title\_buffer, то перейти на шаг 7, иначе – на шаг 19.

Шаг 7. Если значение tail\_of\_title\_new равно NULL, то перейти к шагу 8, иначе – к шагу 13.

Шаг 8. Присвоить значение переменной head\_of\_title\_new, равное head\_of\_title\_buffer.

Шаг 9. Присвоить значение переменной tail\_of\_title\_new, равное tail\_of\_title\_buffer.

Шаг 10. Присвоить значение NULL переменным head\_of\_title\_buffer и tail\_of\_title\_buffer соответсвенно.

Шаг 11. Выполнить статический метод QMessageBox::information с параметрами "Одобрено","Аннулирование отмена успешно проведено".

Шаг 12. Перейти на шаг 35.

Шаг 13. Присвоить значение переменной tail\_of\_title\_new->next\_par, равное head\_of\_title\_buffer.

Шаг 14. Присвоить значение переменной head\_of\_title\_buffer->prev\_par, равное tail\_of\_title\_new.

Шаг 15. Присвоить значение переменной tail\_of\_title\_new, равное head\_of\_title\_buffer.

Шаг 16. Присвоить значение NULL переменным head\_of\_title\_buffer и tail\_of\_title\_buffer соответсвенно.

Шаг 17. Выполнить статический метод QMessageBox::information с параметрами "Одобрено","Аннулирование отмена успешно проведено".

Шаг 18. Перейти на шаг 35.

Шаг 19. Если значение tail\_of\_title\_new равно NULL, то перейти на шаг 20, иначе – на шаг 26.

Шаг 20. Присвоить значение переменной head\_of\_title\_new, равное tail\_of\_title\_buffer.

Шаг 21. Присвоить значение переменной tail\_of\_title\_new, равное tail\_of\_title\_buffer.

Шаг 22. Присвоить значение переменной tail\_of\_title\_buffer, равное tail\_of\_title\_buffer->prev\_par.

Шаг 23. Присвоить значение NULL tail\_of\_title\_buffer->next\_par и tail\_of\_title\_buffer->next\_par соответсвенно.

Шаг 24. Выполнить статический метод QMessageBox::information с параметрами ,"Одобрено","Аннулирование отмена успешно проведено".

Шаг 25. Перейти на шаг 35.

Шаг 26. Объявить переменную \*tmp типа Node и проинициализировать ее значением

tail\_of\_title\_buffer.

Шаг 27. Присвоить значение переменной tail\_of\_title\_buffer, равное tail\_of\_title\_buffer->prev\_par.

Шаг 28. Присвоить значение переменной tail\_of\_title\_buffer->next\_par, равное NULL.

Шаг 29. Присвоить значение переменной tmp->prev\_par, равное tail\_of\_title\_new.

Шаг 30. Присвоить значение переменной tail\_of\_title\_new->next\_par, равное tmp.

Шаг 31. Присвоить значение переменной tail\_of\_title\_new, равное tmp.

Шаг 32. Выполнить удаление tmp.

Шаг 34. Перейти к шагу 35.

Шаг 35. Конец.

Описание алгоритма по шагам для функции сохранения добавления параметров (void save\_add()).

Шаг 1. Начало.

Шаг 2. Входные данные: ParametrList \*lise(контейнер для добавления информации), Node \*head\_of\_title, \*tail\_of\_title, \*head\_of\_title\_new, \*tail\_of\_title\_new, \*head\_of\_title\_buffer, \*tail\_of\_title\_buffer(адреса начала и конца списков исходных, новых и буферных параметров соответственно).

Шаг 3. Если адрес head\_of\_title\_new == NULL, выполнить статический метод QMessageBox::warning с параметрами ,"Ошибка", "Список для сохранения дублирует имеющийся список".

Шаг 4. Объявление переменной tmp\_old типа Node \* и последующая инициализация переменной head\_of\_title.

Шаг 5. Объявление переменной tmp\_new типа Node \* и последующая инициализация переменной head\_of\_title\_new.

Шаг 6. Запуск цикла с параметром tmp\_new, пока tmp\_new != NULL.

Шаг 7. Инициализация переменной tmp\_new значением head\_of\_title\_new.

Шаг 8. Запуск цикла с параметром tmp\_new, пока tmp\_new!=NULL.

Шаг 9. Если значения tmp\_new->Parametr и tmp\_old->Parametr совпадают, то перейти к шагу 10, иначе к шагу 22.

Шаг 10. Если не существует tmp\_new->prev\_par и tmp\_new->next\_par, перейти к шагу 11, иначе к шагу 13.

Шаг 11. Установить значения переменных head\_of\_title\_new и tail\_of\_title\_new равным NULL.

Шаг 12. выполнить статический метод QMessageBox::warning с параметрами ,"Ошибка", "Список для сохранения дублирует имеющийся список", перейти к шагу 37.

Шаг 13. Если существуют адреса tmp\_new->prev\_par и tmp\_new->next\_par соответсвенно, то перейти к шагу 14, иначе – к шагу 16.

Шаг 14. Присвоить значение переменной tmp\_new->prev\_par->next\_par, равное tmp\_new->next\_par.

Шаг 15. Присвоить значение переменной tmp\_new->next\_par->prev\_par, равное tmp\_new->prev\_par.

Шаг 16. Если существует tmp\_new->prev\_par, перейти к шагу 17, иначе – к шагу 19.

Шаг 17. Присвоить значение переменной tmp\_new->prev\_par->next\_par, равное NULL.

Шаг 18. Присвоить значение переменной tail\_of\_title\_new, равное tmp\_new->prev\_par.

Шаг 19. Если существует tmp\_new->next\_par, перейти к шагу 20, иначе – к шагу 22.

Шаг 20. Присвоить значение переменной tmp\_new->next\_par->prev\_par, равное NULL.

Шаг 21. Присвоить значение переменной head\_of\_title\_new, равное tmp\_new->next\_par.

Шаг 22. Перейти на шаг 8.

Шаг 23. Изменить значение tmp\_new на tmp\_new->next\_par.

Шаг 24. Перейти на шаг 6.

Шаг 25. Изменить значение tmp\_old на tmp\_old->next\_par.

Шаг 26. Если существует tail\_of\_title, то перейти к шагу 27, иначе – к шагу 37.

Шаг 27. Объявить переменную code типа int и проинициализировать ее значением tail\_of\_title->Code\_Of\_Parametrs + 1.

Шаг 28. Присвоить значение переменной tmp\_new, равное head\_of\_title\_new

Шаг 29. Запуск цикла с параметром tmp\_new, пока tmp\_new!=NULL

Шаг 30. Изменить значение переменной tmp\_new->Code\_Of\_Parametrs на code++.

Шаг 31. Присвоить значение переменной tmp\_new, равное tmp\_new->next\_par.

Шаг 32. Перейти на шаг 29.

Шаг 33. Присвоить значение переменной tail\_of\_title->next\_par, равное head\_of\_title\_new.

Шаг 34. Присвоить значение переменной head\_of\_title\_new->prev\_par, равное tail\_of\_title.

Шаг 35. Присвоить значение переменной tail\_of\_title, равное tail\_of\_title\_new.

Шаг 36. Присвоить значение NULL переменным head\_of\_title\_new и tail\_of\_title\_new соответственно.

Шаг 37. Объявить переменную code типа int и проинициализировать ее значением 1.

Шаг 38. Присвоить значение переменной tmp\_new, равное head\_of\_title\_new

Шаг 39. Запуск цикла с параметром tmp\_new, пока tmp\_new!=NULL

Шаг 40. Изменить значение переменной tmp\_new->Code\_Of\_Parametrs на code++.

Шаг 41. Присвоить значение переменной tmp\_new, равное tmp\_new->next\_par.

Шаг 42. Перейти на шаг 39.

Шаг 33. Присвоить значение переменной head\_of\_title, равное head\_of\_title\_new.

Шаг 34. Присвоить значение переменной tail\_of\_title, равное tail\_of\_title\_new.

Шаг 35. Присвоить значение NULL переменным head\_of\_title\_new и tail\_of\_title\_new соответственно.

Шаг 36. Выполнить метод insert\_in\_file\_int().

Шаг 37. Конец.

**ПРИЛОЖЕНИЕ Г**

*(обязательное)*

Листинг программы для персонального компьютера

// addform.h

#pragma once

#include <QWidget>

#include <QLabel>

#include <QLineEdit>

#include <QPushButton>

#include <QCheckBox>

#include <QString>

#include <QRegExp>

#include <QGridLayout>

#include <QMessageBox>

class MainWindow;

namespace Ui {

class AddForm;

}

class AddForm : public QWidget

{

friend class MainWindow; // задаем класс MainWindow дружественным

Q\_OBJECT

int type = 0; // метка дальнейшего использования

QLabel \*instr; // поля для инструкций

QLabel \*info;

QLineEdit \*input; // поле для ввода парамера

QCheckBox \*dis\_check; // CheckBox для выбора редактируемого параметра

QCheckBox \*reg\_check;

QCheckBox \*town\_check;

QCheckBox \*fam\_check;

QCheckBox \*name\_check;

QCheckBox \*surname\_check;

QPushButton \*cansel; // кнопки итнерфейса

QPushButton \*recansel;

QPushButton \*save;

QPushButton \*add;

QPushButton \*back;

QGridLayout \*lay; // слой

public:

explicit **AddForm**(QWidget \*parent = 0);

~***AddForm***();

signals: // сигналы:

void **ADD**(QString arg,int type); // добавдение параметра

void **CANCEL**(int type); // отмена

void **RECANCEL**(int type); // аннулирование отмены

void **SAVE**(int type); // сохранение

private slots: // слоты:

void **ADD\_SLOT**(); // добавление параметра

void **CANCEL\_SLOT**(); // отмена

void **RECANCEL\_SLOT**(); // аннулирование отмены

void **SAVE\_SLOT**(); // сохранение

void **CHECKED\_D**(bool arg); // действия при выборе параметра

void **CHECKED\_R**(bool arg);

void **CHECKED\_T**(bool arg);

void **CHECKED\_F**(bool arg);

void **CHECKED\_N**(bool arg);

void **CHECKED\_S**(bool arg);

private:

Ui::AddForm \*ui;

};

// adduser.h

#pragma once

#include <QWidget>

#include <QLabel>

#include <QLineEdit>

#include <QPushButton>

#include <QCheckBox>

#include <QString>

#include <QRegExp>

#include <QGridLayout>

#include <QMessageBox>

namespace Ui {

class adduser;

}

class adduser : public QWidget

{

friend class MainWindow; // задание класса MainWindow дружественным

Q\_OBJECT

int mode = 0; // метки дальнейших действий

int type = 0;

QLabel \*instr; // поля инструкций

QLabel \*info\_log;

QLabel \*info\_pas;

QLineEdit \*log; // поля ввода параметров

QLineEdit \*pass;

QCheckBox \*user\_check; // выбор редактируемго раздела

QCheckBox \*admin\_check;

QPushButton \*cansel; // кнопки интерфейса

QPushButton \*recansel;

QPushButton \*save;

QPushButton \*add;

QPushButton \*back;

QGridLayout \*lay; // слой

public:

explicit **adduser**(QWidget \*parent = 0);

~***adduser***();

signals: // сигналы:

void **ADD\_OR\_DEL**(QString log,QString pass,int type); // проверка выбранного режима

void **CANCEL**(int type); // отмена

void **RECANCEL**(int type); // аннулирование отмены

void **SAVE**(int mode, int type); // сохранение

private slots: // слоты

void **ADD\_SLOT**(); // добавление

void **CANCEL\_SLOT**(); // отмена

void **RECANCEL\_SLOT**(); // аннулирование отмены

void **SAVE\_SLOT**(); // сохранение

void **CHECKED\_A**(bool arg); // действия при выборе раздела

void **CHECKED\_U**(bool arg);

private:

Ui::adduser \*ui;

};

// authorization.h

#pragma once

#include <QDialog>

#include "user.h"

#include <QRegExp>

#include <QRegExpValidator>

namespace Ui {

class Authorization;

}

class Authorization : public QDialog

{

Q\_OBJECT

ParametrList Users; // список пользователей

ParametrList Admins; // список администраторов

QString LOGIN; // логин

QString PASSWORD; // пароль

public:

explicit **Authorization**(QWidget \*parent=0);

~***Authorization***();

signals: // сигналы:

void **OUT**(); // выход

void **ERROR\_AUT**(); // ошибка авторизации

void **SIGNIN**(QString log, bool type); // авторизация

private slots: // слоты:

void ***closeEvent***(QCloseEvent \*event); // закрытие окна авторизации

void **on\_Login\_line\_textChanged**(const QString arg1); // дейстивия при изменении полей

void **on\_Pass\_line\_textChanged**(const QString arg1);

void **on\_User\_mode\_clicked**(bool checked);

void **on\_Adm\_mode\_clicked**(bool checked);

void **on\_OUT\_clicked**(); // нажата кнопка выход

void **on\_IN\_clicked**(); // нажата кнопка вход

private:

Ui::Authorization \*ui;

};

// citizen.h

#pragma once

#include "parametrlist.h"

class CitizenCode;

class Citizen // базовый класс Citizen, хряняший информацию о гражданине

{

friend class MainWindow; // задание дружественных классов

friend class CitizenCode;

friend class User;

protected:

QString family; // фамилия

QString name; // имя

QString surname; // отчество

int day\_of\_birth; // день рождения

int month\_of\_birth; // месяц рождения

int year\_of\_birth; // год рождения

QString District; // область регистрации

QString Region; // район регистрации

QString Town; // населенный пункт регистрации

QString number\_of\_passpor; // номер паспорта

QString ID; // идендификационный номер

public:

**Citizen**(); // конструктор без параметров

virtual void ***clear***(); // виртуальная функция ичистки кобъекта

virtual ~***Citizen***(); // виртуальный деструктор деструктор

};

// citizencode.h

#pragma once

#include "citizen.h"

class CitizenCode : public Citizen

{

friend class ParametrList; // задание дружественных классов

friend class CitizenList;

friend class User;

friend class MainWindow;

protected:

int i\_family=0; // код фамилия

int i\_name=0; // код имени

int i\_surname=0; // код отчества

int i\_District=0; // код области регистрации

int i\_Region=0; // код района регистрации

int i\_Town=0; // код населенного пункта регистрации

public:

**CitizenCode**();

virtual ~***CitizenCode***();

bool **verification**(CitizenList \*list); // поиск соответствий

};

// inputform.h

#pragma once

#include <QWidget>

#include <QPushButton>

#include <QGridLayout>

#include <QWidget>

#include <QLabel>

#include <QLineEdit>

#include <QRegExp>

#include <QRegExpValidator>

#include <QComboBox>

class MainWindow;

namespace Ui {

class InputForm;

}

class InputForm : public QWidget

{

Q\_OBJECT

friend class MainWindow;

int type; // метка дальнейших действий

QLabel \*fam; // поля инструкций

QLabel \*name;

QLabel \*surname;

QLabel \*day;

QLabel \*month;

QLabel \*year;

QLabel \*district;

QLabel \*region;

QLabel \*town;

QLabel \*number\_of\_passport;

QLabel \*ID;

QLineEdit \*fam\_line; // поля ввода

QLineEdit \*name\_line;

QLineEdit \*surname\_line;

QComboBox \*day\_line;

QComboBox \*month\_line;

QComboBox \*year\_line;

QLineEdit \*district\_line;

QLineEdit \*region\_line;

QLineEdit \*town\_line;

QLineEdit \*number\_of\_passport\_line;

QLineEdit \*ID\_line;

QPushButton \*cansel; // кнопки интерфейса

QPushButton \*recansel;

QPushButton \*save;

QPushButton \*search;

QPushButton \*back;

QGridLayout \*lay; // слой

public:

explicit **InputForm**(QWidget \*parent = 0);

~***InputForm***();

private:

Ui::InputForm \*ui;

};

// mainmenu.h

#pragma once

#include <QPushButton>

#include <QGridLayout>

#include <QWidget>

class MainWindow;

namespace Ui {

class MainMenu;

}

class MainMenu : public QWidget

{

Q\_OBJECT

friend class MainWindow; // задание дружественного класс

QGridLayout \*main\_menu\_lay = new QGridLayout; // слой

QPushButton \*aunthification = new QPushButton(); // кнопки интерфейса

QPushButton \*add\_new = new QPushButton();

QPushButton \*add\_new\_citizen = new QPushButton();

QPushButton \*add\_new\_user = new QPushButton();

QPushButton \*delete\_old\_user = new QPushButton();

QPushButton \*Quit = new QPushButton();

public:

explicit **MainMenu**(QWidget \*parent = 0);

~***MainMenu***();

private:

Ui::MainMenu \*ui;

};

// mainwindow.h

#pragma once

#include "authorization.h"

#include "mainmenu.h"

#include "inputform.h"

#include "addform.h"

#include "adduser.h"

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

bool type; // тип интерфейса

Authorization \*auth; // виждет авторизации

User \*us; // объект с данными

ParametrList \*admins; // список администраторов

ParametrList \*users; // список пользователей

MainMenu \*menu; // виджет главного меню

InputForm \*input; // форма ввода информации гражданина

AddForm \*add\_par; // форма добавления параметров

adduser \*add\_or\_del\_user; // форма добавления администраторов и пользователей

public:

explicit **MainWindow**(QWidget \*parent = 0);

~***MainWindow***();

public slots: // слоты

void **exit**(); // выход

void **OK\_SIGN**(QString log, bool type); // успешная авторизация

void **OPEN\_AUNTH**(); // открытие окна аунтефикации

void **OPEN\_ADD\_NEW\_PARAMETR**(); // открытие формы добавления параметра

void **OPEN\_ADD\_CITIZEN**(); // открытие формы добавления гражданина

void **OPEN\_ADD\_USER**(); // открытие формы добавления/ удаления

void **OPEN\_DEL\_USER**(); // пользователя и администратора

void **CHECK\_INPUT\_FOR\_AUNTH\_OR\_ADD**(); // проверка заполения формы ввода гражданина

void **START\_AUNTH**(); // начало аунтефикации

void **START\_ADD**(); // начало добавления гражданина

void **START\_ADD\_USER**(); // начало добавления пользователя во временный список

void **What\_type**(); // проверка типа работы с гражданином

void **BACK\_FORM\_INPUT**(); // возврат в главное меню

void **BACK\_FORM\_ADD\_PAR**();

void **BACK\_FORM\_ADD\_OR\_DEL\_USER**();

void **WHAT\_PARAMETR\_ADD**(QString data, int type); // проверка редактируемого раздела

void **WHAT\_PARAMETR\_CANCEL**(int type); // параметров и типа редактирования

void **WHAT\_PARAMETR\_RECANCEL**(int type);

void **WHAT\_PARAMETR\_SAVE**(int type);

void **WHAT\_TYPE\_USER**(QString log,QString pass, int type); // проверка редактируемого

void **WHAT\_TYPE\_USER\_C**(int type); // раздела пользователей

void **WHAT\_TYPE\_USER\_CC**(int type); // и типа редактирования

void **WHAT\_TYPE\_USER\_S**(int mode,int type);

signals: // сигналы:

void **GOTOAUNTH**(); // переход к авторизации

void **GOTOADD**(); // переход к добавлению

void **GO\_TO\_CHECK\_AUNTH\_OR\_ADD**(); // переход к проверке ввода

private:

Ui::MainWindow \*ui;

};

// parameterlist.h

#pragma once

#include <QWidget>

#include <QLabel>

#include <QLineEdit>

#include <QPushButton>

#include <QCheckBox>

#include <QString>

#include <QRegExp>

#include <QGridLayout>

#include <QMessageBox>

#include <QMainWindow>

class User;

class MainWindow;

template<class T>

class Iterator // шаблон итератора

{

friend class MainWindow; // задание дружественных классов

friend class User;

friend class ParametrList;

friend class CitizenList;

**Iterator**<**T**>(T \*head, T \*tail); // конструктор Iterator

~**Iterator**<**T**>(); // деструктор Iterator

T \*head\_of\_title=NULL; // адрес начала списка исходных параметров

T \*tail\_of\_title=NULL; // адрес конца списка исходных параметров

private:

bool operator == (Iterator<T> &second); // итератор сравнения контейнеров

bool operator != (Iterator<T> &second); // итератор проверки неравенства контейнеров

void operator + (Iterator<T> &second); // оператор сложения контейнеров

void operator - (Iterator<T> &second); // оператор вычетания контейнера

};

struct Node // структура для записи параметра

{

int Code\_Of\_Parametrs; // код параметра

QString Pass; // пароль

QString Parametr; // параметр ( в случае листа админстраторов и юзеров - логин)

Node \*next\_par=NULL; // адрес следующего элемента

Node \*prev\_par=NULL; // адрес предыдущего элемента

};

class ParametrList // контейнер параметров

{

friend class MainWindow;

friend class User;

friend class Iterator<Node>; // определение класса Iterator дружествнным классу ParametrList

Node \*head\_of\_title=NULL; // адрес начала списка исходных параметров

Node \*tail\_of\_title=NULL; // адрес конца списка исходных параметров

Node \*head\_of\_title\_new=NULL; // адрес начала списка новых параметров

Node \*tail\_of\_title\_new=NULL; // адрес конца списка новых параметров

Node \*head\_of\_title\_buffer=NULL; // адрес начала буферного списка параметров для аннулирования отмены

Node \*tail\_of\_title\_buffer=NULL; // адрес конца буферног списка параметров для аннулирования

Iterator<Node> Itr\_old = Iterator<Node>(head\_of\_title,tail\_of\_title); // итератор для исходного списка параметров

Iterator<Node> Itr\_new = Iterator<Node>(head\_of\_title\_new,tail\_of\_title\_new);// итератор для исходного списка параметров

public:

QString FileName; // имя файла с параметрами

**ParametrList**(); // конструктор ParametrList

~**ParametrList**(); // деструктор ParametrList

bool **eject\_without\_file\_int**(); // извлечение параметров из файлаc с кодом в int

bool **eject\_without\_file\_qstr**(); // извлечение параметров из файлаc с кодом в QString

void **reset\_itr\_new**(); // обновление указателей итератора

void **add\_parametr**(QString paramter); // добавить параметр

void **add\_or\_del\_user**(QString user, QString password); // добавить или удалить пользователя

bool **search**(QString,QString); // поиск по параметру

void **save\_add**(); // сохранить добавление

void **save\_add\_user**(); // сохранить добавление пользователей

void **save\_del\_user**(); // сохранить удаление пользователя

int **search\_code**(QString); // поиск кода

void **insert\_in\_file\_int**(); // запись измененных параметров в файл

void **insert\_in\_file\_qstr**(); // запись измененного списка пользователей в файл

void **cansel**(); // отмена внесения данных

void **cancellation\_cancellation**(); // аннулирование отмены

};

struct CitizenNode

{

int family; // код фамилии

int name; // код имеми

int surname; // код отчества

int day\_of\_birth; // день рождения

int month\_of\_birth; // месяц рождения

int year\_of\_birth; // год рождения

int District; // код области регистрации

int Region; // код района регистрации

int Town; // код населенного пункта регистрации

QString Number\_of\_Pas; // номер паспорта

QString ID; // идендификационный номер

CitizenNode \*next\_per=NULL; // адрес предыдущего гражданина

CitizenNode \*prev\_per=NULL; // адрес следующего гражданина

};

class CitizenList // контейнер со списком граждан

{

friend class MainWindow;

friend class User;

friend class Iterator<CitizenNode>; // определение класса Iterator дружествнным классу CitizenList

CitizenNode \*head\_of\_list\_new=NULL; // адрес начала списка новых граждан

CitizenNode \*tail\_of\_list\_new=NULL; // адрес конца списка новых граждан

CitizenNode \*head\_of\_list\_buffer=NULL; // адрес начала исходного списка

CitizenNode \*tail\_of\_list\_buffer=NULL; // адрес конца исходного списка

Iterator<CitizenNode> Itr\_old = Iterator<CitizenNode>(head\_of\_list,tail\_of\_list); // итератор для исходного списка граждан

Iterator<CitizenNode> Itr\_new = Iterator<CitizenNode>(head\_of\_list\_new,tail\_of\_list\_new);// итератор для исходного списка граждан

public:

QString FileName; // имя файла с гражданами

CitizenNode \*head\_of\_list=NULL; // адрес начала исходного списка

CitizenNode \*tail\_of\_list=NULL; // адрес конца исходного списка

**CitizenList**(QString name); // конструктор Citizen

~**CitizenList**(); // деструктор Citizen

bool **eject\_without\_file**(); // извлечение параметров из файла

void **insert\_in\_file**(); // запись измененных параметров в файл

void **add\_citizen**(int i\_f, int i\_n, int i\_s,// добавить гражданина

int DB,int MB, int YB,

int i\_d, int i\_r, int i\_t,

QString num\_of\_pass, QString ID);

void **save\_add**(); // сохранить добавление

void **del\_sitizen**(int i\_f, int i\_n, int i\_s,// удалить гражданина

int DB,int MB, int YB,

int i\_d, int i\_r, int i\_t,

QString num\_of\_pass, QString ID);

};

// user.h

#pragma once

#include "citizencode.h"

class User : virtual public CitizenCode

{

friend class MainWindow;

QString Login; // логин

ParametrList \*Familis; // листы параметиров

ParametrList \*Names;

ParametrList \*Surnames;

ParametrList \*Districts;

ParametrList \*Regions;

ParametrList \*Towns;

CitizenList \*year1900;

CitizenList \*year1910;

CitizenList \*year1920;

CitizenList \*year1930;

CitizenList \*year1940;

CitizenList \*year1950;

CitizenList \*year1960;

CitizenList \*year1970;

CitizenList \*year1980;

CitizenList \*year1990;

CitizenList \*year2000;

CitizenList \*year2010;

public:

**User**(QString Log);

virtual~***User***();

void ***clear***();

};

// addform.cpp

#include "addform.h"

#include "ui\_addform.h"

#include <QtDebug>

#include <QRegExpValidator>

AddForm::**AddForm**(QWidget \*parent) :

QWidget(parent),

ui(new Ui::AddForm)

{

ui->setupUi(this);

QRegExp text("[А-Я]{1,1}[а-я]{1,29}"); // задание ограничения на ввод

instr = new QLabel; // выделение памяти

info = new QLabel;

dis\_check = new QCheckBox;

reg\_check = new QCheckBox;

town\_check = new QCheckBox;

fam\_check = new QCheckBox;

name\_check = new QCheckBox;

surname\_check = new QCheckBox;

back = new QPushButton;

cansel = new QPushButton;

save = new QPushButton;

recansel = new QPushButton;

add = new QPushButton;

lay = new QGridLayout;

input = new QLineEdit;

// задание текста кнопок и полей

this->instr->setText("Необходимо выбрать раздел для редактирования,\n"

"если Вы желаете ввести параметр, то необходимо заполнить форму для текста,\n"

"также если Вы по ошибке ввели параметр, то можете нажать кнопку Отмена, добавление будет отменяться порядке от самого нового к самому старому,\n"

"также можно аннулировать отмену, аннулирование будет происходить аналагично,\n"

"чтобы применить изменения необходимо нажать кнопку Сохранить (отменить нельзя),\n"

"если вернуться в меню не сохранив изменения, то они будут утеряны.");

this->info->setText("Введите параметр");

this->dis\_check->setText("Область");

this->reg\_check->setText("Район");

this->town\_check->setText("Город");

this->fam\_check->setText("Фамилия");

this->name\_check->setText("Имя");

this->surname\_check->setText("Отчество");

this->back->setText("В главное меню");

this->cansel->setText("Отмена");

this->save->setText("Сохранить");

this->recansel->setText("Аннулировать отмену");

this->add->setText("Добавить параметр");

this->input->setValidator(new QRegExpValidator(text,this)); // установка ограничения на ввод

lay->addWidget(instr,0,0); // расстановка элементов на слой

lay->addWidget(dis\_check,1,0);

lay->addWidget(reg\_check,2,0);

lay->addWidget(town\_check,3,0);

lay->addWidget(fam\_check,4,0);

lay->addWidget(name\_check,5,0);

lay->addWidget(surname\_check,6,0);

lay->addWidget(info,7,0);

lay->addWidget(input,7,1);

lay->addWidget(add,8,0);

lay->addWidget(save,8,1);

lay->addWidget(cansel,9,0);

lay->addWidget(recansel,9,1);

lay->addWidget(back,10,0);

setLayout(lay); // установка слоя

// связывание сигналов и слотов

connect(dis\_check,SIGNAL(clicked(bool)),this,SLOT(CHECKED\_D(bool)));

connect(reg\_check,SIGNAL(clicked(bool)),this,SLOT(CHECKED\_R(bool)));

connect(town\_check,SIGNAL(clicked(bool)),this,SLOT(CHECKED\_T(bool)));

connect(fam\_check,SIGNAL(clicked(bool)),this,SLOT(CHECKED\_F(bool)));

connect(name\_check,SIGNAL(clicked(bool)),this,SLOT(CHECKED\_N(bool)));

connect(surname\_check,SIGNAL(clicked(bool)),this,SLOT(CHECKED\_S(bool)));

connect(add,SIGNAL(clicked()),this,SLOT(ADD\_SLOT()));

connect(cansel,SIGNAL(clicked()),this,SLOT(CANCEL\_SLOT()));

connect(recansel,SIGNAL(clicked()),this,SLOT(RECANCEL\_SLOT()));

connect(save,SIGNAL(clicked()),this,SLOT(SAVE\_SLOT()));

}

AddForm::~***AddForm***()

{

delete ui;

}

void AddForm::**ADD\_SLOT**()

{

if(!type)

{

QMessageBox::warning(this,"Ошибка выбора","Выберите редактируемый параметр");

return;

}

if(input->text().length()>1)

{

emit ADD(input->text(),type);

return;

}

else {

QMessageBox::warning(this,"Ошибка ввода","Введите корректные данные");

return;

}

}

void AddForm::**CANCEL\_SLOT**()

{

if(!type)

{

QMessageBox::warning(this,"Ошибка выбора","Выберите редактируемый параметр");

return;

}

emit CANCEL(type);

}

void AddForm::**RECANCEL\_SLOT**()

{

if(!type)

{

QMessageBox::warning(this,"Ошибка выбора","Выберите редактируемый параметр");

return;

}

emit RECANCEL(type);

}

void AddForm::**SAVE\_SLOT**()

{

if(!type)

{

QMessageBox::warning(this,"Ошибка выбора","Выберите редактируемый параметр");

return;

}

emit SAVE(type);

}

void AddForm::**CHECKED\_D**(bool arg)

{

if(arg) type=1;

else type=0;

fam\_check->setEnabled(!arg);

name\_check->setEnabled(!arg);

surname\_check->setEnabled(!arg);

reg\_check->setEnabled(!arg);

town\_check->setEnabled(!arg);

}

void AddForm::**CHECKED\_R**(bool arg)

{

if(arg) type=2;

else type=0;

fam\_check->setEnabled(!arg);

name\_check->setEnabled(!arg);

surname\_check->setEnabled(!arg);

dis\_check->setEnabled(!arg);

town\_check->setEnabled(!arg);

}

void AddForm::**CHECKED\_T**(bool arg)

{

if(arg) type=3;

else type=0;

fam\_check->setEnabled(!arg);

name\_check->setEnabled(!arg);

surname\_check->setEnabled(!arg);

reg\_check->setEnabled(!arg);

dis\_check->setEnabled(!arg);

}

void AddForm::**CHECKED\_F**(bool arg)

{

if(arg) type=4;

else type=0;

dis\_check->setEnabled(!arg);

name\_check->setEnabled(!arg);

surname\_check->setEnabled(!arg);

reg\_check->setEnabled(!arg);

town\_check->setEnabled(!arg);

}

void AddForm::**CHECKED\_N**(bool arg)

{

if(arg) type=5;

else type=0;

dis\_check->setEnabled(!arg);

fam\_check->setEnabled(!arg);

surname\_check->setEnabled(!arg);

reg\_check->setEnabled(!arg);

town\_check->setEnabled(!arg);

}

void AddForm::**CHECKED\_S**(bool arg)

{

if(arg) type=6;

else type=0;

dis\_check->setEnabled(!arg);

fam\_check->setEnabled(!arg);

name\_check->setEnabled(!arg);

reg\_check->setEnabled(!arg);

town\_check->setEnabled(!arg);

}

//adduser.cpp

#include "adduser.h"

#include "ui\_adduser.h"

adduser::**adduser**(QWidget \*parent) :

QWidget(parent),

ui(new Ui::adduser)

{

ui->setupUi(this);

QRegExp text("[A-Za-z0-9\_]{1,20}"); // задание ограничения на ввод

instr = new QLabel; // выделение пымяти

info\_log = new QLabel;

info\_pas = new QLabel;

log = new QLineEdit;

pass = new QLineEdit;

user\_check = new QCheckBox;

admin\_check = new QCheckBox;

cansel = new QPushButton;

recansel = new QPushButton;

save = new QPushButton;

add = new QPushButton;

back = new QPushButton;

lay = new QGridLayout;

this->log->setValidator(new QRegExpValidator(text,this)); // установка ограничения на ввод

// задание текста полей и кнопок

this->instr->setText("Необходимо выбрать раздел для редактирования,\n"

"если Вы желаете добавить/удалить, то необходимо заполнить форму для текста,\n"

"также если Вы по ошибке ввели параметр, то можете нажать кнопку Отмена, добавление будет отменяться порядке от самого нового к самому старому,\n"

"также можно аннулировать отмену, аннулирование будет происходить аналагично,\n"

"чтобы применить изменения необходимо нажать кнопку Сохранить (отменить нельзя),\n"

"если вернуться в меню не сохранив изменения, то они будут утеряны.");

this->back->setText("В главное меню");

this->cansel->setText("Отмена");

this->save->setText("Сохранить");

this->recansel->setText("Аннулировать отмену");

this->add->setText("Добавить параметр");

this->info\_log->setText("Ввудите логин, он может содержать символы латинского алфавита,\n"

"цифры и нижнее подчеркивание");

this->info\_pas->setText("Ввудите пароль, он может содержать символы латинского алфавита,\n"

"цифры и нижнее подчеркивание");

this->admin\_check->setText("Администратор");

this->user\_check->setText("Пользователь");

this->lay->addWidget(instr,0,0); // установка элементов на слой

this->lay->addWidget(admin\_check,1,0);

this->lay->addWidget(user\_check,2,0);

this->lay->addWidget(info\_log,3,0);

this->lay->addWidget(log,3,1);

this->lay->addWidget(info\_pas,4,0);

this->lay->addWidget(pass,4,1);

this->lay->addWidget(add,5,0);

this->lay->addWidget(save,5,1);

this->lay->addWidget(cansel,6,0);

this->lay->addWidget(recansel,6,1);

this->lay->addWidget(back,7,0);

setLayout(lay); // установка слоя

// связывание сигналов и слотов

connect(admin\_check,SIGNAL(clicked(bool)),this,SLOT(CHECKED\_A(bool)));

connect(user\_check,SIGNAL(clicked(bool)),this,SLOT(CHECKED\_U(bool)));

connect(add,SIGNAL(clicked()),this,SLOT(ADD\_SLOT()));

connect(cansel,SIGNAL(clicked()),this,SLOT(CANCEL\_SLOT()));

connect(recansel,SIGNAL(clicked()),this,SLOT(RECANCEL\_SLOT()));

connect(save,SIGNAL(clicked()),this,SLOT(SAVE\_SLOT()));

}

adduser::~***adduser***()

{

delete ui;

}

void adduser::**ADD\_SLOT**()

{

if(!type)

{

QMessageBox::warning(this,"Ошибка выбора","Выберите редактируемый параметр");

return;

}

if(log->text().length()>1 && pass->text().length()>1)

{

emit ADD\_OR\_DEL(log->text(),pass->text(),type);

return;

}

else {

QMessageBox::warning(this,"Ошибка ввода","Введите корректные данные");

return;

}

}

void adduser::**CANCEL\_SLOT**()

{

if(!type)

{

QMessageBox::warning(this,"Ошибка выбора","Выберите редактируемый параметр");

return;

}

emit CANCEL(type);

}

void adduser::**RECANCEL\_SLOT**()

{

if(!type)

{

QMessageBox::warning(this,"Ошибка выбора","Выберите редактируемый параметр");

return;

}

emit RECANCEL(type);

}

void adduser::**SAVE\_SLOT**()

{

if(!type)

{

QMessageBox::warning(this,"Ошибка выбора","Выберите редактируемый параметр");

return;

}

emit SAVE(mode, type);

}

void adduser::**CHECKED\_A**(bool arg)

{

if(arg) type=1;

else type=0;

user\_check->setEnabled(!arg);

}

void adduser::**CHECKED\_U**(bool arg)

{

if(arg) type=2;

else type=0;

admin\_check->setEnabled(!arg);

}

// authorization.cpp

#include "authorization.h"

#include "ui\_authorization.h"

#include <QDir>

#include <QDebug>

Authorization::**Authorization**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Authorization)

{

ui->setupUi(this);

this->setEnabled(true);

QRegExp log\_exp("[A-Za-z0-9]{4,20}"); // задание ограничений на ввод

QRegExp pass\_exp("[a-zA-Z0-9]{8,30}");

ui->Login\_line->setValidator(new QRegExpValidator(log\_exp,this)); // установка ограничений

ui->Pass\_line->setValidator(new QRegExpValidator(pass\_exp,this));

ui->Login\_line->setEnabled(false);

QString dir=QDir::homePath();

Users.FileName=(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/UserAcc.txt");

Admins.FileName=(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/AdminAcc.txt");

if(Users.eject\_without\_file\_qstr()) // открытие файлов для авторизации

if(Admins.eject\_without\_file\_qstr())

ui->Login\_line->setEnabled(true);

ui->Pass\_line->setEnabled(false);

ui->IN->setEnabled(false);

ui->User\_mode->setEnabled(false);

ui->User\_mode->setChecked(false);

ui->Adm\_mode->setEnabled(false);

ui->Adm\_mode->setChecked(false);

}

Authorization::~***Authorization***()

{

delete ui;

}

void Authorization::***closeEvent***(QCloseEvent \*event)

{

emit OUT();

this->close();

}

void Authorization::**on\_Login\_line\_textChanged**(const QString arg1)

{

if(ui->Login\_line->hasAcceptableInput())

{

LOGIN=arg1;

ui->Pass\_line->setEnabled(!arg1.isEmpty());

ui->Pass\_line->clear();

ui->IN->setEnabled(false);

ui->User\_mode->setEnabled(false);

ui->User\_mode->setChecked(false);

ui->Adm\_mode->setEnabled(false);

ui->Adm\_mode->setChecked(false);

}

}

void Authorization::**on\_Pass\_line\_textChanged**(const QString arg1)

{

if(ui->Pass\_line->hasAcceptableInput())

{

PASSWORD=arg1;

ui->IN->setEnabled(false);

ui->User\_mode->setEnabled(!arg1.isEmpty());

ui->User\_mode->setChecked(false);

ui->Adm\_mode->setEnabled(!arg1.isEmpty());

ui->Adm\_mode->setChecked(false);

}

}

void Authorization::**on\_User\_mode\_clicked**(bool checked)

{

ui->IN->setEnabled(checked);

ui->Adm\_mode->setEnabled(!checked);

ui->Adm\_mode->setChecked(false);

}

void Authorization::**on\_Adm\_mode\_clicked**(bool checked)

{

ui->IN->setEnabled(checked);

ui->User\_mode->setEnabled(!checked);

ui->User\_mode->setChecked(false);

}

void Authorization::**on\_OUT\_clicked**()

{

emit OUT();

this->close();

}

void Authorization::**on\_IN\_clicked**()

{

if(ui->Adm\_mode->isChecked())

{

if(this->Admins.search(PASSWORD,LOGIN))

{

emit SIGNIN(LOGIN,true);

ui->Login\_line->clear();

this->hide();

return;

} else{

QMessageBox::warning(NULL,"Ошибка аунтефиукации", "Введены несоответствующие данные");

}

} else if(this->Users.search(PASSWORD,LOGIN))

{

emit SIGNIN(LOGIN,false);

ui->Login\_line->clear();

this->hide();

return;

} else

{

QMessageBox::warning(NULL,"Ошибка аунтефиукации", "Введены несоответствующие данные");

}

}

//citizen.h

#include "citizen.h"

Citizen::**Citizen**()

{

}

void Citizen::***clear***()

{

}

Citizen::~***Citizen***()

{

delete this;

}

// citizencode.cpp

#include "citizencode.h"

CitizenCode::**CitizenCode**()

:Citizen ()

{

}

CitizenCode::~***CitizenCode***()

{

delete this;

}

bool CitizenCode::**verification**(CitizenList \*list) // верификация

{

CitizenNode \*tmp=list->head\_of\_list;

while(tmp){

if(this->i\_family==tmp->family &&

this->i\_name==tmp->name &&

this->i\_surname==tmp->surname &&

this->day\_of\_birth == tmp->day\_of\_birth &&

this->month\_of\_birth == tmp->month\_of\_birth &&

this->year\_of\_birth == tmp->year\_of\_birth &&

this->i\_District == tmp->District &&

this->i\_Region == tmp->Region &&

this->i\_Town == tmp->Town &&

this->number\_of\_passpor == tmp->Number\_of\_Pas &&

this->ID == tmp->ID)

{

return true;

}

tmp=tmp->next\_per;

}

return false;

}

// inputform.cpp

#include "inputform.h"

#include "ui\_inputform.h"

#include <QRegExpValidator>

InputForm::**InputForm**(QWidget \*parent) :

QWidget(parent),

ui(new Ui::InputForm)

{

ui->setupUi(this);

QRegExp text\_exp("[А-Я]{1,1}[а-я]{1,29}"); // задание ограничений

QRegExp number\_exp("[A-Z]{1,2}[0-9]{7,7}");

QRegExp ID\_exp("[0-9]{7,7}[A-Z]{1,1}[0-9]{3,3}[A-Z]{2,2}[0-9]{1,1}");

// выделение памяти и установка текста кнопок и полей

fam = new QLabel;

fam->setText("Фамилия");

name = new QLabel;

name->setText("Имя");

surname = new QLabel;

surname->setText("Отчество");

day = new QLabel;

day->setText("День рождения");

month = new QLabel;

month->setText("Месяц рождения");

year = new QLabel;

year->setText("Год ождения");

district = new QLabel;

district->setText("Область регистрации");

region = new QLabel;

region->setText("Район регистрации");

town = new QLabel;

town->setText("Город регистрации");

number\_of\_passport = new QLabel;

number\_of\_passport->setText("Номер паспорта");

ID = new QLabel;

ID->setText("Идентификационный номер");

fam\_line = new QLineEdit;

fam\_line->setValidator(new QRegExpValidator(text\_exp,this)); // установка ограничений

name\_line = new QLineEdit;

name\_line->setValidator(new QRegExpValidator(text\_exp,this));

surname\_line = new QLineEdit;

surname\_line->setValidator(new QRegExpValidator(text\_exp,this));

day\_line = new QComboBox; // задание текста для выбора даты

for (int i=0; i <=31; i++)

{

day\_line->addItem(QString::number(i));

}

month\_line = new QComboBox;

for (int i=0; i <=12; i++)

{

month\_line->addItem(QString::number(i));

}

year\_line = new QComboBox;

year\_line->addItem("0");

for (int i=2017; i >=1900; i--)

{

year\_line->addItem(QString::number(i));

}

district\_line = new QLineEdit;

district\_line->setValidator(new QRegExpValidator(text\_exp,this)); // установка ограничений

region\_line = new QLineEdit;

region\_line->setValidator(new QRegExpValidator(text\_exp,this));

town\_line = new QLineEdit;

town\_line->setValidator(new QRegExpValidator(text\_exp,this));

number\_of\_passport\_line = new QLineEdit;

number\_of\_passport\_line->setValidator(new QRegExpValidator(number\_exp,this));

ID\_line = new QLineEdit;

ID\_line->setValidator(new QRegExpValidator(ID\_exp,this));

search = new QPushButton;

search->setText("Проверить");

back=new QPushButton;

back->setText("Назад");

lay = new QGridLayout;

// установка объектов на слой

lay->addWidget(fam, 0,0);

lay->addWidget(fam\_line,0,1);

lay->addWidget(name,1,0);

lay->addWidget(name\_line,1,1);

lay->addWidget(surname,2,0);

lay->addWidget(surname\_line,2,1);

lay->addWidget(day,3,0);

lay->addWidget(day\_line,3,1);

lay->addWidget(month,4,0);

lay->addWidget(month\_line,4,1);

lay->addWidget(year,5,0);

lay->addWidget(year\_line,5,1);

lay->addWidget(district,6,0);

lay->addWidget(district\_line,6,1);

lay->addWidget(region,7,0);

lay->addWidget(region\_line,7,1);

lay->addWidget(town,8,0);

lay->addWidget(town\_line,8,1);

lay->addWidget(number\_of\_passport,9,0);

lay->addWidget(number\_of\_passport\_line,9,1);

lay->addWidget(ID,10,0);

lay->addWidget(ID\_line,10,1);

lay->addWidget(back,11,0);

lay->addWidget(search,11,1);

// установка слоя

setLayout(lay);

}

InputForm::~***InputForm***()

{

delete ui;

}

//main.cpp

#include "authorization.h"

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow \*w = new MainWindow;

return a.exec();

}

// mainmenu.cpp

#include "mainmenu.h"

#include "ui\_mainmenu.h"

MainMenu::**MainMenu**(QWidget \*parent) :

QWidget(parent),

ui(new Ui::MainMenu)

{

ui->setupUi(this);

aunthification->setText("Верификация"); // зажание текста кнопок и полей

add\_new->setText("Добавить параметры в базу");

add\_new\_citizen->setText("Добавить гражданина в базу");

add\_new\_user->setText("Добавить пользователя");

delete\_old\_user->setText("Удалить пользователя");

Quit->setText("Выход");

main\_menu\_lay->addWidget(aunthification,0,0); // расстановка элелементов на слой

main\_menu\_lay->addWidget(add\_new,1,0);

main\_menu\_lay->addWidget(add\_new\_citizen,2,0);

main\_menu\_lay->addWidget(add\_new\_user,3,0);

main\_menu\_lay->addWidget(delete\_old\_user,4,0);

main\_menu\_lay->addWidget(Quit,5,0);

setLayout(main\_menu\_lay); // задание слоя

}

MainMenu::~***MainMenu***()

{

delete ui;

}

// mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::**MainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

auth=new Authorization;

this->setEnabled(false);

auth->show();

menu = new MainMenu;

input = new InputForm;

add\_par = new AddForm;

add\_or\_del\_user = new adduser;

// связывание сигналов и слотов

connect(auth,SIGNAL(OUT()),this,SLOT(exit()));

connect(auth,SIGNAL(SIGNIN(QString, bool)),this,SLOT(OK\_SIGN(QString,bool)));

connect(menu->Quit,SIGNAL(clicked()),this,SLOT(exit()));

connect(menu->aunthification,SIGNAL(clicked()),this,SLOT(OPEN\_AUNTH()));

connect(menu->add\_new,SIGNAL(clicked()), this, SLOT(OPEN\_ADD\_NEW\_PARAMETR()));

connect(menu->add\_new\_citizen,SIGNAL(clicked()),this,SLOT(OPEN\_ADD\_CITIZEN()));

connect(menu->add\_new\_user, SIGNAL(clicked()),this, SLOT(OPEN\_ADD\_USER()));

connect(menu->delete\_old\_user, SIGNAL(clicked()),this, SLOT(OPEN\_DEL\_USER()));

connect(input->search,SIGNAL(clicked()), this,SLOT(What\_type()));

connect(input->back,SIGNAL(clicked()),this,SLOT(BACK\_FORM\_INPUT()));

connect(add\_par->back,SIGNAL(clicked()),this,SLOT(BACK\_FORM\_ADD\_PAR()));

connect(add\_par,SIGNAL(ADD(QString,int)), this, SLOT(WHAT\_PARAMETR\_ADD(QString, int)));

connect(add\_par,SIGNAL(CANCEL(int)), this, SLOT(WHAT\_PARAMETR\_CANCEL(int)));

connect(add\_par,SIGNAL(RECANCEL(int)), this, SLOT(WHAT\_PARAMETR\_RECANCEL(int)));

connect(add\_par,SIGNAL(SAVE(int)), this, SLOT(WHAT\_PARAMETR\_SAVE(int)));

connect(add\_or\_del\_user->back,SIGNAL(clicked()),this,SLOT(BACK\_FORM\_ADD\_OR\_DEL\_USER()));

connect(add\_or\_del\_user,SIGNAL(ADD\_OR\_DEL(QString, QString, int)),this,SLOT(WHAT\_TYPE\_USER(QString, QString, int)));

connect(add\_or\_del\_user,SIGNAL(CANCEL(int)), this, SLOT(WHAT\_TYPE\_USER\_C(int)));

connect(add\_or\_del\_user,SIGNAL(RECANCEL(int)), this, SLOT(WHAT\_TYPE\_USER\_CC(int)));

connect(add\_or\_del\_user,SIGNAL(SAVE(int,int)), this, SLOT(WHAT\_TYPE\_USER\_S(int, int)));

connect(this,SIGNAL(GO\_TO\_CHECK\_AUNTH\_OR\_ADD()),this,SLOT(CHECK\_INPUT\_FOR\_AUNTH\_OR\_ADD()));

connect(this,SIGNAL(GOTOAUNTH()),this,SLOT(START\_AUNTH()));

connect(this,SIGNAL(GOTOADD()),this,SLOT(START\_ADD()));

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**exit**()

{

this->close();

}

void MainWindow::**OK\_SIGN**(QString log, bool type)

{

this->type=type;

this->setEnabled(true);

this->show();

us=new User(log);

menu->delete\_old\_user->setEnabled(type);

menu->add\_new\_user->setEnabled(type);

menu->add\_new\_citizen->setEnabled(type);

menu->add\_new->setEnabled(type);

if(type)

{

users = new ParametrList;

admins = new ParametrList;

users->FileName="C:/Users/nikol/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/UserAcc.txt";

admins->FileName="C:/Users/nikol/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/AdminAcc.txt";

users->eject\_without\_file\_qstr();

admins->eject\_without\_file\_qstr();

}

setCentralWidget(menu);

}

void MainWindow::**OPEN\_AUNTH**()

{

input->type=1;

us->*clear*();

takeCentralWidget();

setCentralWidget(input);

input->search->setText("Проверить");

}

void MainWindow::**OPEN\_ADD\_CITIZEN**()

{

input->type=3;

us->*clear*();

takeCentralWidget();

setCentralWidget(input);

input->search->setText("Добававить");

}

void MainWindow::**OPEN\_ADD\_USER**()

{

add\_or\_del\_user->mode=1;

takeCentralWidget();

setCentralWidget(add\_or\_del\_user);

add\_or\_del\_user->add->setText("Добавить");

}

void MainWindow::**OPEN\_DEL\_USER**()

{

add\_or\_del\_user->mode=2;

takeCentralWidget();

setCentralWidget(add\_or\_del\_user);

add\_or\_del\_user->add->setText("Удалить");

}

void MainWindow::**OPEN\_ADD\_NEW\_PARAMETR**()

{

takeCentralWidget();

setCentralWidget(add\_par);

}

void MainWindow::**CHECK\_INPUT\_FOR\_AUNTH\_OR\_ADD**()

{

if(input->fam\_line->hasAcceptableInput())

{

us->family=input->fam\_line->text();

us->i\_family=us->Familis->search\_code(us->family);

if(!us->i\_family){

QMessageBox::warning(this,"Ошибка ввода", "Введенная фамилия отсутствует в базе");

return;

}

} else {

QMessageBox::warning(this,"Ошибка ввода", "Введенна некорректная фамилия");

return;

}

if(input->name\_line->hasAcceptableInput())

{

us->name=input->name\_line->text();

us->i\_name=us->Names->search\_code(us->name);

if(!us->i\_name){

QMessageBox::warning(this,"Ошибка ввода", "Введенное имя отсутствует в базе");

return;

}

} else {

QMessageBox::warning(this,"Ошибка ввода", "Введенно некорректное имя");

return;

}

if(input->surname\_line->hasAcceptableInput())

{

us->surname=input->surname\_line->text();

us->i\_surname=us->Surnames->search\_code(us->surname);

if(!us->i\_surname){

QMessageBox::warning(this,"Ошибка ввода", "Введенное отчество отсутствует в базе");

return;

}

} else {

QMessageBox::warning(this,"Ошибка ввода", "Введенно некорректное отчесвто");

return;

}

if(input->district\_line->hasAcceptableInput())

{

us->District=input->district\_line->text();

us->i\_District=us->Districts->search\_code(us->District);

if(!us->i\_District)

{

QMessageBox::warning(this,"Ошибка ввода", "Введенная область отсутствует в базе");

return;

}

}else {

QMessageBox::warning(this,"Ошибка ввода", "Введенна некорректная область");

return;

}

if(input->region\_line->hasAcceptableInput())

{

us->Region=input->region\_line->text();

us->i\_Region=us->Regions->search\_code(us->Region);

if(!us->i\_Region)

{

QMessageBox::warning(this,"Ошибка ввода", "Введенный район отсутствует в базе");

return;

}

}else {

QMessageBox::warning(this,"Ошибка ввода", "Введенно некорректное название района");

return;

}

if(input->town\_line->hasAcceptableInput())

{

us->Town=input->town\_line->text();

us->i\_Town=us->Towns->search\_code(us->Town);

if(!us->i\_Town)

{

QMessageBox::warning(this,"Ошибка ввода", "Введенный город отсутствует в базе");

return;

}

}else {

QMessageBox::warning(this,"Ошибка ввода", "Введенно некорректное название города");

return;

}

if(input->year\_line->currentText().toInt()!=0)

us->year\_of\_birth=input->year\_line->currentText().toInt();

else{

QMessageBox::warning(this,"Ошибка ввода", "Выберите год рождения");

return;

}

if(input->month\_line->currentText().toInt()!=0)

us->month\_of\_birth=input->month\_line->currentText().toInt();

else{

QMessageBox::warning(this,"Ошибка ввода", "Выберите месяц рождения");

return;

}

if(input->day\_line->currentText().toInt()!=0)

us->day\_of\_birth=input->day\_line->currentText().toInt();

else{

QMessageBox::warning(this,"Ошибка ввода", "Выберите день рождения");

return;

}

if((us->day\_of\_birth>29 && us->month\_of\_birth ==2) ||

(us->day\_of\_birth==31 && (us->month\_of\_birth==4 ||

us->month\_of\_birth==6 ||

us->month\_of\_birth==9 ||

us->month\_of\_birth==11)))

{

QMessageBox::warning(this,"Ошибка ввода", "Выбраный день не соответствует выбраному месяцу");

return;

}

if(input->ID\_line->hasAcceptableInput())

{

us->ID=input->ID\_line->text();

}else {

QMessageBox::warning(this,"Ошибка ввода", "Введен некоректный идендификационный номер");

return;

}

if(input->number\_of\_passport\_line->hasAcceptableInput())

{

us->number\_of\_passpor=input->number\_of\_passport\_line->text();

}else {

QMessageBox::warning(this,"Ошибка ввода", "Введен некоректный номер паспорта");

return;

}

if(this->input->type==1)

emit GOTOAUNTH();

if(this->input->type==3)

emit GOTOADD();

}

void MainWindow::**START\_AUNTH**()

{

if(us->year\_of\_birth>=1900 && us->year\_of\_birth<1910){

if(us->verification(us->year1900))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1910 && us->year\_of\_birth<1920){

if(us->verification(us->year1910))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1920 && us->year\_of\_birth<1930){

if(us->verification(us->year1920))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1930 && us->year\_of\_birth<1940){

if(us->verification(us->year1930))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1940 && us->year\_of\_birth<1950){

if(us->verification(us->year1940))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1950 && us->year\_of\_birth<1960){

if(us->verification(us->year1950))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1960 && us->year\_of\_birth<1970){

if(us->verification(us->year1960))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1970 && us->year\_of\_birth<1980){

if(us->verification(us->year1970))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1980 && us->year\_of\_birth<1990){

if(us->verification(us->year1980))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1990 && us->year\_of\_birth<2000){

if(us->verification(us->year1990))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=2000 && us->year\_of\_birth<2010){

if(us->verification(us->year2000))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=2010 && us->year\_of\_birth<2020){

if(us->verification((us->year2010)))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

}

void MainWindow::**START\_ADD**()

{

if(us->year\_of\_birth>=1900 && us->year\_of\_birth<1910){

us->year1900->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1910 && us->year\_of\_birth<1920){

us->year1910->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1920 && us->year\_of\_birth<1930){

us->year1920->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1930 && us->year\_of\_birth<1940){

us->year1930->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1940 && us->year\_of\_birth<1950){

us->year1940->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1950 && us->year\_of\_birth<1960){

us->year1950->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1960 && us->year\_of\_birth<1970){

us->year1960->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1970 && us->year\_of\_birth<1980){

us->year1970->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1980 && us->year\_of\_birth<1990){

us->year1980->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1990 && us->year\_of\_birth<2000){

us->year1990->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=2000 && us->year\_of\_birth<2010){

us->year2000->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=2010 && us->year\_of\_birth<2020){

us->year2010->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

}

void MainWindow::**START\_ADD\_USER**()

{

}

void MainWindow::**What\_type**()

{

if(input->type==1 || input->type==3)

emit GO\_TO\_CHECK\_AUNTH\_OR\_ADD();

}

void MainWindow::**BACK\_FORM\_INPUT**()

{

input->fam\_line->clear();

input->name\_line->clear();

input->surname\_line->clear();

input->day\_line->setCurrentIndex(0);

input->month\_line->setCurrentIndex(0);

input->year\_line->setCurrentIndex(0);

input->district\_line->clear();

input->region\_line->clear();

input->town\_line->clear();

input->number\_of\_passport\_line->clear();

input->ID\_line->clear();

takeCentralWidget();

setCentralWidget(menu);

}

void MainWindow::**BACK\_FORM\_ADD\_PAR**()

{

add\_par->input->clear();

add\_par->type = 0;

add\_par->dis\_check->setChecked(false);

add\_par->reg\_check->setChecked(false);

add\_par->town\_check->setChecked(false);

add\_par->dis\_check->setEnabled(true);

add\_par->reg\_check->setEnabled(true);

add\_par->town\_check->setEnabled(true);

us->Districts->head\_of\_title\_new=NULL;

us->Districts->tail\_of\_title\_new=NULL;

us->Districts->head\_of\_title\_buffer=NULL;

us->Districts->tail\_of\_title\_buffer=NULL;

us->Regions->head\_of\_title\_new=NULL;

us->Regions->tail\_of\_title\_new=NULL;

us->Regions->head\_of\_title\_buffer=NULL;

us->Regions->tail\_of\_title\_buffer=NULL;

us->Towns->head\_of\_title\_new=NULL;

us->Towns->tail\_of\_title\_new=NULL;

us->Towns->head\_of\_title\_buffer=NULL;

us->Towns->tail\_of\_title\_buffer=NULL;

us->Familis->head\_of\_title\_new=NULL;

us->Familis->tail\_of\_title\_new=NULL;

us->Familis->head\_of\_title\_buffer=NULL;

us->Familis->tail\_of\_title\_buffer=NULL;

us->Names->head\_of\_title\_new=NULL;

us->Names->tail\_of\_title\_new=NULL;

us->Names->head\_of\_title\_buffer=NULL;

us->Names->tail\_of\_title\_buffer=NULL;

us->Surnames->head\_of\_title\_new=NULL;

us->Surnames->tail\_of\_title\_new=NULL;

us->Surnames->head\_of\_title\_buffer=NULL;

us->Surnames->tail\_of\_title\_buffer=NULL;

takeCentralWidget();

setCentralWidget(menu);

}

void MainWindow::**BACK\_FORM\_ADD\_OR\_DEL\_USER**()

{

add\_or\_del\_user->log->clear();

add\_or\_del\_user->pass->clear();

add\_or\_del\_user->mode=false;

add\_or\_del\_user->type=false;

add\_or\_del\_user->admin\_check->setEnabled(true);

add\_or\_del\_user->user\_check->setEnabled(true);

add\_or\_del\_user->admin\_check->setChecked(false);

add\_or\_del\_user->user\_check->setChecked(false);

admins->head\_of\_title\_new=NULL;

admins->tail\_of\_title\_new=NULL;

admins->head\_of\_title\_buffer=NULL;

admins->tail\_of\_title\_buffer=NULL;

takeCentralWidget();

setCentralWidget(menu);

}

void MainWindow::**WHAT\_PARAMETR\_ADD**(QString data, int type)

{

if(type==1){

us->Districts->add\_parametr(data);

}

if(type==2) {

us->Regions->add\_parametr(data);

}

if(type==3) {

us->Towns->add\_parametr(data);

}

if(type==4) {

us->Familis->add\_parametr(data);

}

if(type==5) {

us->Names->add\_parametr(data);

}

if(type==6) {

us->Surnames->add\_parametr(data);

}

add\_par->input->clear();

add\_par->type = 0;

add\_par->dis\_check->setChecked(false);

add\_par->reg\_check->setChecked(false);

add\_par->town\_check->setChecked(false);

add\_par->fam\_check->setChecked(false);

add\_par->name\_check->setChecked(false);

add\_par->surname\_check->setChecked(false);

add\_par->dis\_check->setEnabled(true);

add\_par->reg\_check->setEnabled(true);

add\_par->town\_check->setEnabled(true);

add\_par->fam\_check->setEnabled(true);

add\_par->name\_check->setEnabled(true);

add\_par->surname\_check->setEnabled(true);

}

void MainWindow::**WHAT\_PARAMETR\_CANCEL**(int type)

{

if(type==1)

us->Districts->cansel();

if(type==2)

us->Regions->cansel();

if(type==3)

us->Towns->cansel();

if(type==4)

us->Familis->cansel();

if(type==5)

us->Names->cansel();

if(type==6)

us->Surnames->cansel();

add\_par->input->clear();

add\_par->type = 0;

add\_par->dis\_check->setChecked(false);

add\_par->reg\_check->setChecked(false);

add\_par->town\_check->setChecked(false);

add\_par->fam\_check->setChecked(false);

add\_par->name\_check->setChecked(false);

add\_par->surname\_check->setChecked(false);

add\_par->dis\_check->setEnabled(true);

add\_par->reg\_check->setEnabled(true);

add\_par->town\_check->setEnabled(true);

add\_par->fam\_check->setEnabled(true);

add\_par->name\_check->setEnabled(true);

add\_par->surname\_check->setEnabled(true);

}

void MainWindow::**WHAT\_PARAMETR\_RECANCEL**(int type)

{

if(type==1)

us->Districts->cancellation\_cancellation();

if(type==2)

us->Regions->cancellation\_cancellation();

if(type==3)

us->Towns->cancellation\_cancellation();

if(type==4)

us->Familis->cancellation\_cancellation();

if(type==5)

us->Names->cancellation\_cancellation();

if(type==6)

us->Surnames->cancellation\_cancellation();

add\_par->input->clear();

add\_par->type = 0;

add\_par->dis\_check->setChecked(false);

add\_par->reg\_check->setChecked(false);

add\_par->town\_check->setChecked(false);

add\_par->fam\_check->setChecked(false);

add\_par->name\_check->setChecked(false);

add\_par->surname\_check->setChecked(false);

add\_par->dis\_check->setEnabled(true);

add\_par->reg\_check->setEnabled(true);

add\_par->town\_check->setEnabled(true);

add\_par->fam\_check->setEnabled(true);

add\_par->name\_check->setEnabled(true);

add\_par->surname\_check->setEnabled(true);

}

void MainWindow::**WHAT\_PARAMETR\_SAVE**(int type)

{

add\_par->input->clear();

add\_par->type = 0;

add\_par->dis\_check->setChecked(false);

add\_par->reg\_check->setChecked(false);

add\_par->town\_check->setChecked(false);

add\_par->fam\_check->setChecked(false);

add\_par->name\_check->setChecked(false);

add\_par->surname\_check->setChecked(false);

add\_par->dis\_check->setEnabled(true);

add\_par->reg\_check->setEnabled(true);

add\_par->town\_check->setEnabled(true);

add\_par->fam\_check->setEnabled(true);

add\_par->name\_check->setEnabled(true);

add\_par->surname\_check->setEnabled(true);

if(type==1)

{

us->Districts->save\_add();

}

if(type==2)

{

us->Regions->save\_add();

}

if(type==3)

{

us->Towns->save\_add();

}

if(type==4) {

us->Familis->save\_add();

}

if(type==5) {

us->Names->save\_add();

}

if(type==6) {

us->Surnames->save\_add();

}

}

void MainWindow::**WHAT\_TYPE\_USER**(QString log, QString pass, int type)

{

if(type==1)

{

admins->add\_or\_del\_user(log, pass);

}

if(type==2)

{

users->add\_or\_del\_user(log, pass);

}

add\_or\_del\_user->admin\_check->setEnabled(true);

add\_or\_del\_user->admin\_check->setChecked(false);

add\_or\_del\_user->user\_check->setEnabled(true);

add\_or\_del\_user->user\_check->setChecked(false);

add\_or\_del\_user->log->clear();

add\_or\_del\_user->pass->clear();

}

void MainWindow::**WHAT\_TYPE\_USER\_C**(int type)

{

if(type==1)

{

admins->cansel();

}

if(type==2)

{

users->cansel();

}

add\_or\_del\_user->type=0;

add\_or\_del\_user->log->clear();

add\_or\_del\_user->pass->clear();

add\_or\_del\_user->admin\_check->setEnabled(true);

add\_or\_del\_user->admin\_check->setChecked(false);

add\_or\_del\_user->user\_check->setEnabled(true);

add\_or\_del\_user->user\_check->setChecked(false);

}

void MainWindow::**WHAT\_TYPE\_USER\_CC**(int type)

{

if(type==1)

{

admins->cancellation\_cancellation();

}

if(type==2)

{

users->cancellation\_cancellation();

}

add\_or\_del\_user->type=0;

add\_or\_del\_user->log->clear();

add\_or\_del\_user->pass->clear();

add\_or\_del\_user->admin\_check->setEnabled(true);

add\_or\_del\_user->admin\_check->setChecked(false);

add\_or\_del\_user->user\_check->setEnabled(true);

add\_or\_del\_user->user\_check->setChecked(false);

}

void MainWindow::**WHAT\_TYPE\_USER\_S**(int mode, int type)

{

add\_or\_del\_user->type=0;

add\_or\_del\_user->log->clear();

add\_or\_del\_user->pass->clear();

add\_or\_del\_user->admin\_check->setEnabled(true);

add\_or\_del\_user->admin\_check->setChecked(false);

add\_or\_del\_user->user\_check->setEnabled(true);

add\_or\_del\_user->user\_check->setChecked(false);

if(mode==1)

{

if(type==1)

{

admins->save\_add\_user();

}

if(type==2)

{

users->save\_add\_user();

}

}

if(mode==2)

{

if(type==1)

{

admins->save\_del\_user();

}

if(type==2)

{

users->save\_del\_user();

}

}

}

// parametrlist.h

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::**MainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

auth=new Authorization;

this->setEnabled(false);

auth->show();

menu = new MainMenu;

input = new InputForm;

add\_par = new AddForm;

add\_or\_del\_user = new adduser;

// связывание сигналов и слотов

connect(auth,SIGNAL(OUT()),this,SLOT(exit()));

connect(auth,SIGNAL(SIGNIN(QString, bool)),this,SLOT(OK\_SIGN(QString,bool)));

connect(menu->Quit,SIGNAL(clicked()),this,SLOT(exit()));

connect(menu->aunthification,SIGNAL(clicked()),this,SLOT(OPEN\_AUNTH()));

connect(menu->add\_new,SIGNAL(clicked()), this, SLOT(OPEN\_ADD\_NEW\_PARAMETR()));

connect(menu->add\_new\_citizen,SIGNAL(clicked()),this,SLOT(OPEN\_ADD\_CITIZEN()));

connect(menu->add\_new\_user, SIGNAL(clicked()),this, SLOT(OPEN\_ADD\_USER()));

connect(menu->delete\_old\_user, SIGNAL(clicked()),this, SLOT(OPEN\_DEL\_USER()));

connect(input->search,SIGNAL(clicked()), this,SLOT(What\_type()));

connect(input->back,SIGNAL(clicked()),this,SLOT(BACK\_FORM\_INPUT()));

connect(add\_par->back,SIGNAL(clicked()),this,SLOT(BACK\_FORM\_ADD\_PAR()));

connect(add\_par,SIGNAL(ADD(QString,int)), this, SLOT(WHAT\_PARAMETR\_ADD(QString, int)));

connect(add\_par,SIGNAL(CANCEL(int)), this, SLOT(WHAT\_PARAMETR\_CANCEL(int)));

connect(add\_par,SIGNAL(RECANCEL(int)), this, SLOT(WHAT\_PARAMETR\_RECANCEL(int)));

connect(add\_par,SIGNAL(SAVE(int)), this, SLOT(WHAT\_PARAMETR\_SAVE(int)));

connect(add\_or\_del\_user->back,SIGNAL(clicked()),this,SLOT(BACK\_FORM\_ADD\_OR\_DEL\_USER()));

connect(add\_or\_del\_user,SIGNAL(ADD\_OR\_DEL(QString, QString, int)),this,SLOT(WHAT\_TYPE\_USER(QString, QString, int)));

connect(add\_or\_del\_user,SIGNAL(CANCEL(int)), this, SLOT(WHAT\_TYPE\_USER\_C(int)));

connect(add\_or\_del\_user,SIGNAL(RECANCEL(int)), this, SLOT(WHAT\_TYPE\_USER\_CC(int)));

connect(add\_or\_del\_user,SIGNAL(SAVE(int,int)), this, SLOT(WHAT\_TYPE\_USER\_S(int, int)));

connect(this,SIGNAL(GO\_TO\_CHECK\_AUNTH\_OR\_ADD()),this,SLOT(CHECK\_INPUT\_FOR\_AUNTH\_OR\_ADD()));

connect(this,SIGNAL(GOTOAUNTH()),this,SLOT(START\_AUNTH()));

connect(this,SIGNAL(GOTOADD()),this,SLOT(START\_ADD()));

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**exit**()

{

this->close();

}

void MainWindow::**OK\_SIGN**(QString log, bool type)

{

this->type=type;

this->setEnabled(true);

this->show();

us=new User(log);

menu->delete\_old\_user->setEnabled(type);

menu->add\_new\_user->setEnabled(type);

menu->add\_new\_citizen->setEnabled(type);

menu->add\_new->setEnabled(type);

if(type)

{

users = new ParametrList;

admins = new ParametrList;

users->FileName="C:/Users/nikol/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/UserAcc.txt";

admins->FileName="C:/Users/nikol/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/AdminAcc.txt";

users->eject\_without\_file\_qstr();

admins->eject\_without\_file\_qstr();

}

setCentralWidget(menu);

}

void MainWindow::**OPEN\_AUNTH**()

{

input->type=1;

us->*clear*();

takeCentralWidget();

setCentralWidget(input);

input->search->setText("Проверить");

}

void MainWindow::**OPEN\_ADD\_CITIZEN**()

{

input->type=3;

us->*clear*();

takeCentralWidget();

setCentralWidget(input);

input->search->setText("Добававить");

}

void MainWindow::**OPEN\_ADD\_USER**()

{

add\_or\_del\_user->mode=1;

takeCentralWidget();

setCentralWidget(add\_or\_del\_user);

add\_or\_del\_user->add->setText("Добавить");

}

void MainWindow::**OPEN\_DEL\_USER**()

{

add\_or\_del\_user->mode=2;

takeCentralWidget();

setCentralWidget(add\_or\_del\_user);

add\_or\_del\_user->add->setText("Удалить");

}

void MainWindow::**OPEN\_ADD\_NEW\_PARAMETR**()

{

takeCentralWidget();

setCentralWidget(add\_par);

}

void MainWindow::**CHECK\_INPUT\_FOR\_AUNTH\_OR\_ADD**()

{

if(input->fam\_line->hasAcceptableInput())

{

us->family=input->fam\_line->text();

us->i\_family=us->Familis->search\_code(us->family);

if(!us->i\_family){

QMessageBox::warning(this,"Ошибка ввода", "Введенная фамилия отсутствует в базе");

return;

}

} else {

QMessageBox::warning(this,"Ошибка ввода", "Введенна некорректная фамилия");

return;

}

if(input->name\_line->hasAcceptableInput())

{

us->name=input->name\_line->text();

us->i\_name=us->Names->search\_code(us->name);

if(!us->i\_name){

QMessageBox::warning(this,"Ошибка ввода", "Введенное имя отсутствует в базе");

return;

}

} else {

QMessageBox::warning(this,"Ошибка ввода", "Введенно некорректное имя");

return;

}

if(input->surname\_line->hasAcceptableInput())

{

us->surname=input->surname\_line->text();

us->i\_surname=us->Surnames->search\_code(us->surname);

if(!us->i\_surname){

QMessageBox::warning(this,"Ошибка ввода", "Введенное отчество отсутствует в базе");

return;

}

} else {

QMessageBox::warning(this,"Ошибка ввода", "Введенно некорректное отчесвто");

return;

}

if(input->district\_line->hasAcceptableInput())

{

us->District=input->district\_line->text();

us->i\_District=us->Districts->search\_code(us->District);

if(!us->i\_District)

{

QMessageBox::warning(this,"Ошибка ввода", "Введенная область отсутствует в базе");

return;

}

}else {

QMessageBox::warning(this,"Ошибка ввода", "Введенна некорректная область");

return;

}

if(input->region\_line->hasAcceptableInput())

{

us->Region=input->region\_line->text();

us->i\_Region=us->Regions->search\_code(us->Region);

if(!us->i\_Region)

{

QMessageBox::warning(this,"Ошибка ввода", "Введенный район отсутствует в базе");

return;

}

}else {

QMessageBox::warning(this,"Ошибка ввода", "Введенно некорректное название района");

return;

}

if(input->town\_line->hasAcceptableInput())

{

us->Town=input->town\_line->text();

us->i\_Town=us->Towns->search\_code(us->Town);

if(!us->i\_Town)

{

QMessageBox::warning(this,"Ошибка ввода", "Введенный город отсутствует в базе");

return;

}

}else {

QMessageBox::warning(this,"Ошибка ввода", "Введенно некорректное название города");

return;

}

if(input->year\_line->currentText().toInt()!=0)

us->year\_of\_birth=input->year\_line->currentText().toInt();

else{

QMessageBox::warning(this,"Ошибка ввода", "Выберите год рождения");

return;

}

if(input->month\_line->currentText().toInt()!=0)

us->month\_of\_birth=input->month\_line->currentText().toInt();

else{

QMessageBox::warning(this,"Ошибка ввода", "Выберите месяц рождения");

return;

}

if(input->day\_line->currentText().toInt()!=0)

us->day\_of\_birth=input->day\_line->currentText().toInt();

else{

QMessageBox::warning(this,"Ошибка ввода", "Выберите день рождения");

return;

}

if((us->day\_of\_birth>29 && us->month\_of\_birth ==2) ||

(us->day\_of\_birth==31 && (us->month\_of\_birth==4 ||

us->month\_of\_birth==6 ||

us->month\_of\_birth==9 ||

us->month\_of\_birth==11)))

{

QMessageBox::warning(this,"Ошибка ввода", "Выбраный день не соответствует выбраному месяцу");

return;

}

if(input->ID\_line->hasAcceptableInput())

{

us->ID=input->ID\_line->text();

}else {

QMessageBox::warning(this,"Ошибка ввода", "Введен некоректный идендификационный номер");

return;

}

if(input->number\_of\_passport\_line->hasAcceptableInput())

{

us->number\_of\_passpor=input->number\_of\_passport\_line->text();

}else {

QMessageBox::warning(this,"Ошибка ввода", "Введен некоректный номер паспорта");

return;

}

if(this->input->type==1)

emit GOTOAUNTH();

if(this->input->type==3)

emit GOTOADD();

}

void MainWindow::**START\_AUNTH**()

{

if(us->year\_of\_birth>=1900 && us->year\_of\_birth<1910){

if(us->verification(us->year1900))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1910 && us->year\_of\_birth<1920){

if(us->verification(us->year1910))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1920 && us->year\_of\_birth<1930){

if(us->verification(us->year1920))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1930 && us->year\_of\_birth<1940){

if(us->verification(us->year1930))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1940 && us->year\_of\_birth<1950){

if(us->verification(us->year1940))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1950 && us->year\_of\_birth<1960){

if(us->verification(us->year1950))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1960 && us->year\_of\_birth<1970){

if(us->verification(us->year1960))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1970 && us->year\_of\_birth<1980){

if(us->verification(us->year1970))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1980 && us->year\_of\_birth<1990){

if(us->verification(us->year1980))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=1990 && us->year\_of\_birth<2000){

if(us->verification(us->year1990))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=2000 && us->year\_of\_birth<2010){

if(us->verification(us->year2000))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

if(us->year\_of\_birth>=2010 && us->year\_of\_birth<2020){

if(us->verification((us->year2010)))

QMessageBox::information(this,"Результат верификации", "Личность подтверждена");

else QMessageBox::warning(this,"Результат верификации", "Личность сфальсифицирована");

}

}

void MainWindow::**START\_ADD**()

{

if(us->year\_of\_birth>=1900 && us->year\_of\_birth<1910){

us->year1900->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1910 && us->year\_of\_birth<1920){

us->year1910->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1920 && us->year\_of\_birth<1930){

us->year1920->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1930 && us->year\_of\_birth<1940){

us->year1930->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1940 && us->year\_of\_birth<1950){

us->year1940->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1950 && us->year\_of\_birth<1960){

us->year1950->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1960 && us->year\_of\_birth<1970){

us->year1960->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1970 && us->year\_of\_birth<1980){

us->year1970->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1980 && us->year\_of\_birth<1990){

us->year1980->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=1990 && us->year\_of\_birth<2000){

us->year1990->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=2000 && us->year\_of\_birth<2010){

us->year2000->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

if(us->year\_of\_birth>=2010 && us->year\_of\_birth<2020){

us->year2010->add\_citizen(us->i\_family,us->i\_name,us->i\_surname,us->day\_of\_birth,

us->month\_of\_birth,us->year\_of\_birth,us->i\_District,

us->i\_Region,us->i\_Town,us->number\_of\_passpor,us->ID);

}

}

void MainWindow::**START\_ADD\_USER**()

{

}

void MainWindow::**What\_type**()

{

if(input->type==1 || input->type==3)

emit GO\_TO\_CHECK\_AUNTH\_OR\_ADD();

}

void MainWindow::**BACK\_FORM\_INPUT**()

{

input->fam\_line->clear();

input->name\_line->clear();

input->surname\_line->clear();

input->day\_line->setCurrentIndex(0);

input->month\_line->setCurrentIndex(0);

input->year\_line->setCurrentIndex(0);

input->district\_line->clear();

input->region\_line->clear();

input->town\_line->clear();

input->number\_of\_passport\_line->clear();

input->ID\_line->clear();

takeCentralWidget();

setCentralWidget(menu);

}

void MainWindow::**BACK\_FORM\_ADD\_PAR**()

{

add\_par->input->clear();

add\_par->type = 0;

add\_par->dis\_check->setChecked(false);

add\_par->reg\_check->setChecked(false);

add\_par->town\_check->setChecked(false);

add\_par->dis\_check->setEnabled(true);

add\_par->reg\_check->setEnabled(true);

add\_par->town\_check->setEnabled(true);

us->Districts->head\_of\_title\_new=NULL;

us->Districts->tail\_of\_title\_new=NULL;

us->Districts->head\_of\_title\_buffer=NULL;

us->Districts->tail\_of\_title\_buffer=NULL;

us->Regions->head\_of\_title\_new=NULL;

us->Regions->tail\_of\_title\_new=NULL;

us->Regions->head\_of\_title\_buffer=NULL;

us->Regions->tail\_of\_title\_buffer=NULL;

us->Towns->head\_of\_title\_new=NULL;

us->Towns->tail\_of\_title\_new=NULL;

us->Towns->head\_of\_title\_buffer=NULL;

us->Towns->tail\_of\_title\_buffer=NULL;

us->Familis->head\_of\_title\_new=NULL;

us->Familis->tail\_of\_title\_new=NULL;

us->Familis->head\_of\_title\_buffer=NULL;

us->Familis->tail\_of\_title\_buffer=NULL;

us->Names->head\_of\_title\_new=NULL;

us->Names->tail\_of\_title\_new=NULL;

us->Names->head\_of\_title\_buffer=NULL;

us->Names->tail\_of\_title\_buffer=NULL;

us->Surnames->head\_of\_title\_new=NULL;

us->Surnames->tail\_of\_title\_new=NULL;

us->Surnames->head\_of\_title\_buffer=NULL;

us->Surnames->tail\_of\_title\_buffer=NULL;

takeCentralWidget();

setCentralWidget(menu);

}

void MainWindow::**BACK\_FORM\_ADD\_OR\_DEL\_USER**()

{

add\_or\_del\_user->log->clear();

add\_or\_del\_user->pass->clear();

add\_or\_del\_user->mode=false;

add\_or\_del\_user->type=false;

add\_or\_del\_user->admin\_check->setEnabled(true);

add\_or\_del\_user->user\_check->setEnabled(true);

add\_or\_del\_user->admin\_check->setChecked(false);

add\_or\_del\_user->user\_check->setChecked(false);

admins->head\_of\_title\_new=NULL;

admins->tail\_of\_title\_new=NULL;

admins->head\_of\_title\_buffer=NULL;

admins->tail\_of\_title\_buffer=NULL;

takeCentralWidget();

setCentralWidget(menu);

}

void MainWindow::**WHAT\_PARAMETR\_ADD**(QString data, int type)

{

if(type==1){

us->Districts->add\_parametr(data);

}

if(type==2) {

us->Regions->add\_parametr(data);

}

if(type==3) {

us->Towns->add\_parametr(data);

}

if(type==4) {

us->Familis->add\_parametr(data);

}

if(type==5) {

us->Names->add\_parametr(data);

}

if(type==6) {

us->Surnames->add\_parametr(data);

}

add\_par->input->clear();

add\_par->type = 0;

add\_par->dis\_check->setChecked(false);

add\_par->reg\_check->setChecked(false);

add\_par->town\_check->setChecked(false);

add\_par->fam\_check->setChecked(false);

add\_par->name\_check->setChecked(false);

add\_par->surname\_check->setChecked(false);

add\_par->dis\_check->setEnabled(true);

add\_par->reg\_check->setEnabled(true);

add\_par->town\_check->setEnabled(true);

add\_par->fam\_check->setEnabled(true);

add\_par->name\_check->setEnabled(true);

add\_par->surname\_check->setEnabled(true);

}

void MainWindow::**WHAT\_PARAMETR\_CANCEL**(int type)

{

if(type==1)

us->Districts->cansel();

if(type==2)

us->Regions->cansel();

if(type==3)

us->Towns->cansel();

if(type==4)

us->Familis->cansel();

if(type==5)

us->Names->cansel();

if(type==6)

us->Surnames->cansel();

add\_par->input->clear();

add\_par->type = 0;

add\_par->dis\_check->setChecked(false);

add\_par->reg\_check->setChecked(false);

add\_par->town\_check->setChecked(false);

add\_par->fam\_check->setChecked(false);

add\_par->name\_check->setChecked(false);

add\_par->surname\_check->setChecked(false);

add\_par->dis\_check->setEnabled(true);

add\_par->reg\_check->setEnabled(true);

add\_par->town\_check->setEnabled(true);

add\_par->fam\_check->setEnabled(true);

add\_par->name\_check->setEnabled(true);

add\_par->surname\_check->setEnabled(true);

}

void MainWindow::**WHAT\_PARAMETR\_RECANCEL**(int type)

{

if(type==1)

us->Districts->cancellation\_cancellation();

if(type==2)

us->Regions->cancellation\_cancellation();

if(type==3)

us->Towns->cancellation\_cancellation();

if(type==4)

us->Familis->cancellation\_cancellation();

if(type==5)

us->Names->cancellation\_cancellation();

if(type==6)

us->Surnames->cancellation\_cancellation();

add\_par->input->clear();

add\_par->type = 0;

add\_par->dis\_check->setChecked(false);

add\_par->reg\_check->setChecked(false);

add\_par->town\_check->setChecked(false);

add\_par->fam\_check->setChecked(false);

add\_par->name\_check->setChecked(false);

add\_par->surname\_check->setChecked(false);

add\_par->dis\_check->setEnabled(true);

add\_par->reg\_check->setEnabled(true);

add\_par->town\_check->setEnabled(true);

add\_par->fam\_check->setEnabled(true);

add\_par->name\_check->setEnabled(true);

add\_par->surname\_check->setEnabled(true);

}

void MainWindow::**WHAT\_PARAMETR\_SAVE**(int type)

{

add\_par->input->clear();

add\_par->type = 0;

add\_par->dis\_check->setChecked(false);

add\_par->reg\_check->setChecked(false);

add\_par->town\_check->setChecked(false);

add\_par->fam\_check->setChecked(false);

add\_par->name\_check->setChecked(false);

add\_par->surname\_check->setChecked(false);

add\_par->dis\_check->setEnabled(true);

add\_par->reg\_check->setEnabled(true);

add\_par->town\_check->setEnabled(true);

add\_par->fam\_check->setEnabled(true);

add\_par->name\_check->setEnabled(true);

add\_par->surname\_check->setEnabled(true);

if(type==1)

{

us->Districts->save\_add();

}

if(type==2)

{

us->Regions->save\_add();

}

if(type==3)

{

us->Towns->save\_add();

}

if(type==4) {

us->Familis->save\_add();

}

if(type==5) {

us->Names->save\_add();

}

if(type==6) {

us->Surnames->save\_add();

}

}

void MainWindow::**WHAT\_TYPE\_USER**(QString log, QString pass, int type)

{

if(type==1)

{

admins->add\_or\_del\_user(log, pass);

}

if(type==2)

{

users->add\_or\_del\_user(log, pass);

}

add\_or\_del\_user->admin\_check->setEnabled(true);

add\_or\_del\_user->admin\_check->setChecked(false);

add\_or\_del\_user->user\_check->setEnabled(true);

add\_or\_del\_user->user\_check->setChecked(false);

add\_or\_del\_user->log->clear();

add\_or\_del\_user->pass->clear();

}

void MainWindow::**WHAT\_TYPE\_USER\_C**(int type)

{

if(type==1)

{

admins->cansel();

}

if(type==2)

{

users->cansel();

}

add\_or\_del\_user->type=0;

add\_or\_del\_user->log->clear();

add\_or\_del\_user->pass->clear();

add\_or\_del\_user->admin\_check->setEnabled(true);

add\_or\_del\_user->admin\_check->setChecked(false);

add\_or\_del\_user->user\_check->setEnabled(true);

add\_or\_del\_user->user\_check->setChecked(false);

}

void MainWindow::**WHAT\_TYPE\_USER\_CC**(int type)

{

if(type==1)

{

admins->cancellation\_cancellation();

}

if(type==2)

{

users->cancellation\_cancellation();

}

add\_or\_del\_user->type=0;

add\_or\_del\_user->log->clear();

add\_or\_del\_user->pass->clear();

add\_or\_del\_user->admin\_check->setEnabled(true);

add\_or\_del\_user->admin\_check->setChecked(false);

add\_or\_del\_user->user\_check->setEnabled(true);

add\_or\_del\_user->user\_check->setChecked(false);

}

void MainWindow::**WHAT\_TYPE\_USER\_S**(int mode, int type)

{

add\_or\_del\_user->type=0;

add\_or\_del\_user->log->clear();

add\_or\_del\_user->pass->clear();

add\_or\_del\_user->admin\_check->setEnabled(true);

add\_or\_del\_user->admin\_check->setChecked(false);

add\_or\_del\_user->user\_check->setEnabled(true);

add\_or\_del\_user->user\_check->setChecked(false);

if(mode==1)

{

if(type==1)

{

admins->save\_add\_user();

}

if(type==2)

{

users->save\_add\_user();

}

}

if(mode==2)

{

if(type==1)

{

admins->save\_del\_user();

}

if(type==2)

{

users->save\_del\_user();

}

}

}

// user.cpp

#include "user.h"

#include <QDir>

User::**User**(QString Log)

: CitizenCode ()

{

this->Login=Log;

Familis = new ParametrList(); // выделение памяти

Names = new ParametrList();

Surnames = new ParametrList();

Districts = new ParametrList();

Regions = new ParametrList();

Towns = new ParametrList();

// задание путуй к файлам

QString dir=QDir::homePath();

Districts->FileName=(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/Districts.txt");

Regions->FileName=(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/Regions.txt");

Towns->FileName=(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/Towns.txt");

Familis->FileName=(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/Familis.txt");

Names->FileName=(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/Names.txt");

Surnames->FileName=(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/Surnames.txt");

year1900 = new CitizenList(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/year1900.bin");

year1910 = new CitizenList(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/year1910.bin");

year1920 = new CitizenList(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/year1920.bin");

year1930 = new CitizenList(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/year1930.bin");

year1940 = new CitizenList(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/year1940.bin");

year1950 = new CitizenList(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/year1950.bin");

year1960 = new CitizenList(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/year1960.bin");

year1970 = new CitizenList(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/year1970.bin");

year1980 = new CitizenList(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/year1980.bin");

year1990 = new CitizenList(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/year1990.bin");

year2000 = new CitizenList(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/year2000.bin");

year2010 = new CitizenList(dir + "/Documents/Republican\_system\_of\_registration\_of\_citizens/Base/year2010.bin");

// чтение из файла

Districts->eject\_without\_file\_int();

Regions->eject\_without\_file\_int();

Towns->eject\_without\_file\_int();

Familis->eject\_without\_file\_int();

Names->eject\_without\_file\_int();

Surnames->eject\_without\_file\_int();

year1900->eject\_without\_file();

year1910->eject\_without\_file();

year1920->eject\_without\_file();

year1930->eject\_without\_file();

year1940->eject\_without\_file();

year1950->eject\_without\_file();

year1960->eject\_without\_file();

year1970->eject\_without\_file();

year1980->eject\_without\_file();

year1990->eject\_without\_file();

year2000->eject\_without\_file();

year2010->eject\_without\_file();

}

User::~***User***()

{

delete this;

}

void User::***clear***() // очистка полей гражданина

{

i\_family=0;

i\_name=0;

i\_surname=0;

i\_District=0;

i\_Region=0;

i\_Town=0;

family.clear();

name.clear();

surname.clear();

day\_of\_birth=0;

month\_of\_birth=0;

year\_of\_birth=0;

District.clear();

Region.clear();

Town.clear();

number\_of\_passpor.clear();

ID.clear();

}

**ПРИЛОЖЕНИЕ Д**

*(обязательное)*

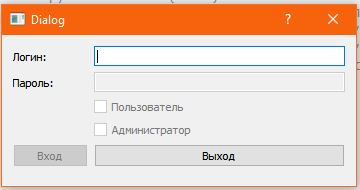
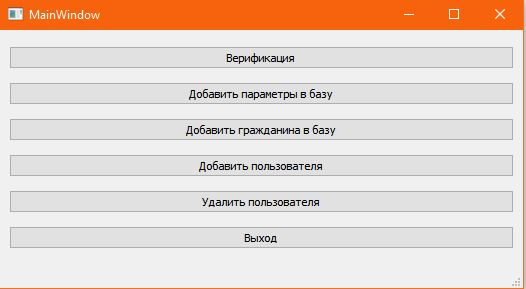
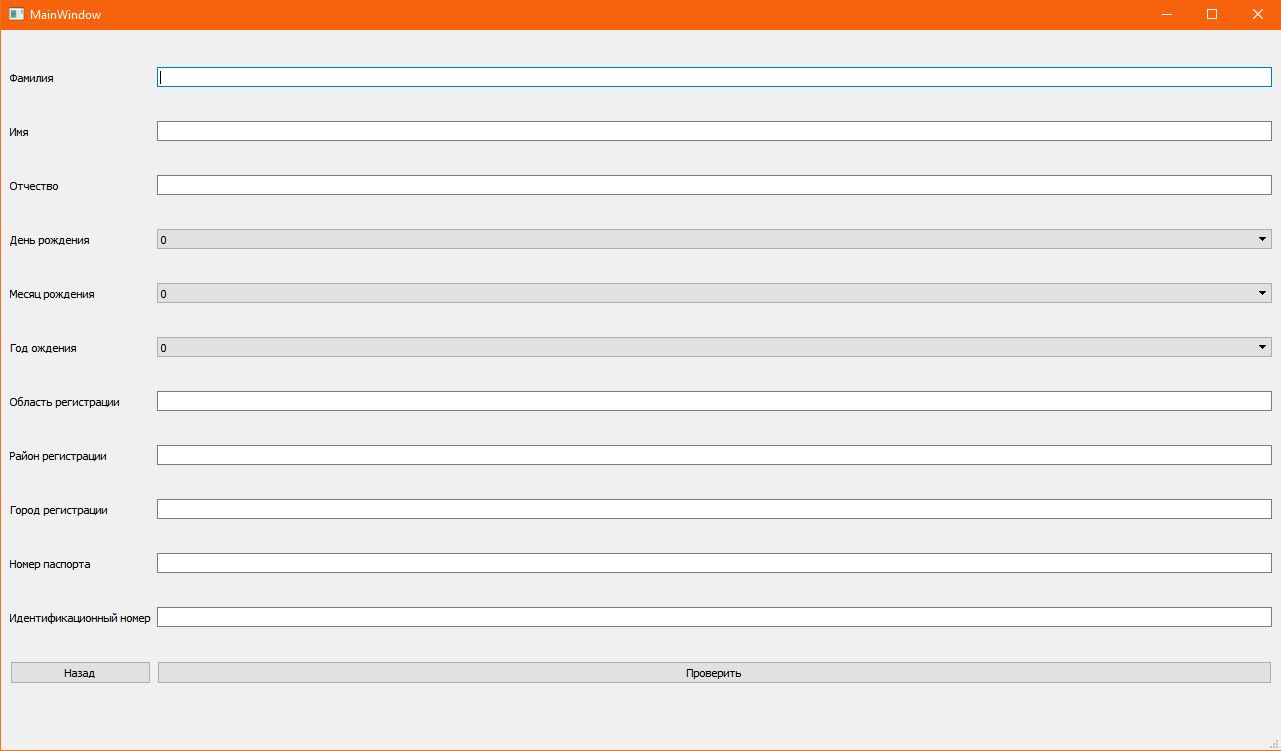
Скриншоты работы программы

Рисунок Д.1 –­ Окно входа

Рисунок Д.2 –­ Главное меню

Рисунок Д.3 –­ Окно верификации

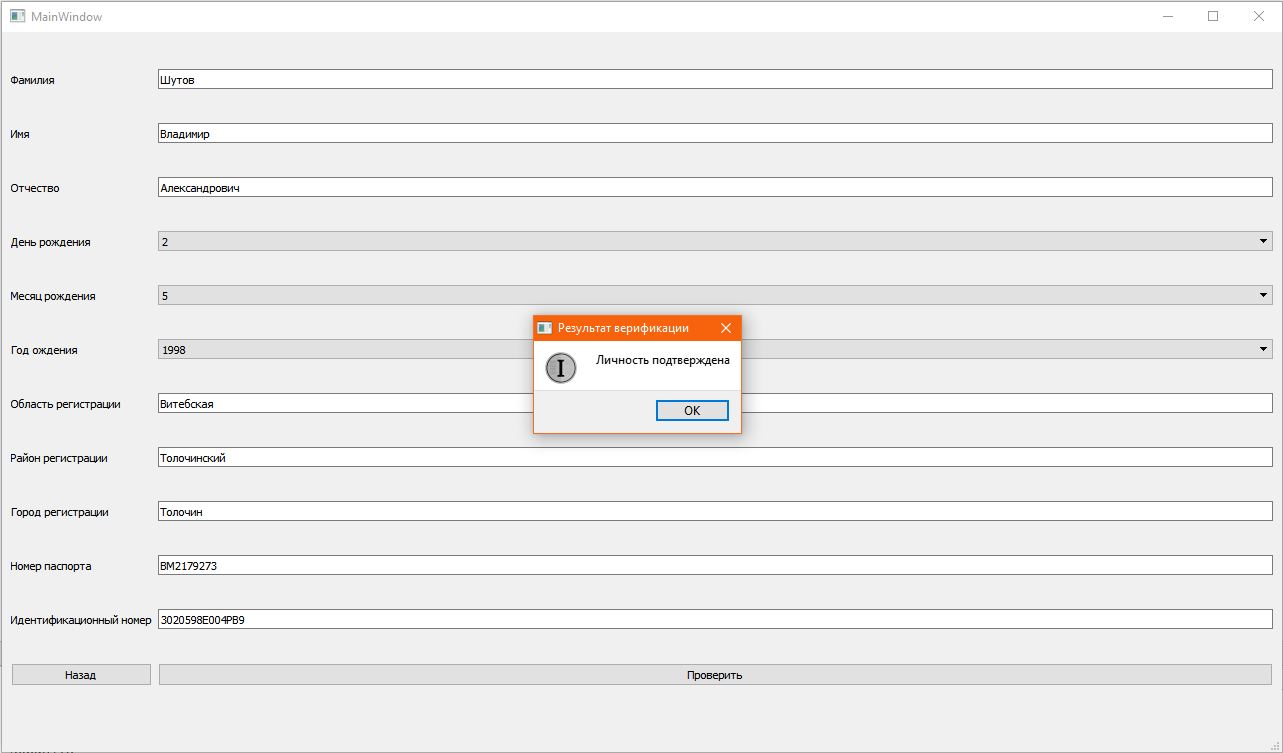


Рисунок Д.4 –­ Положительный результат верификации

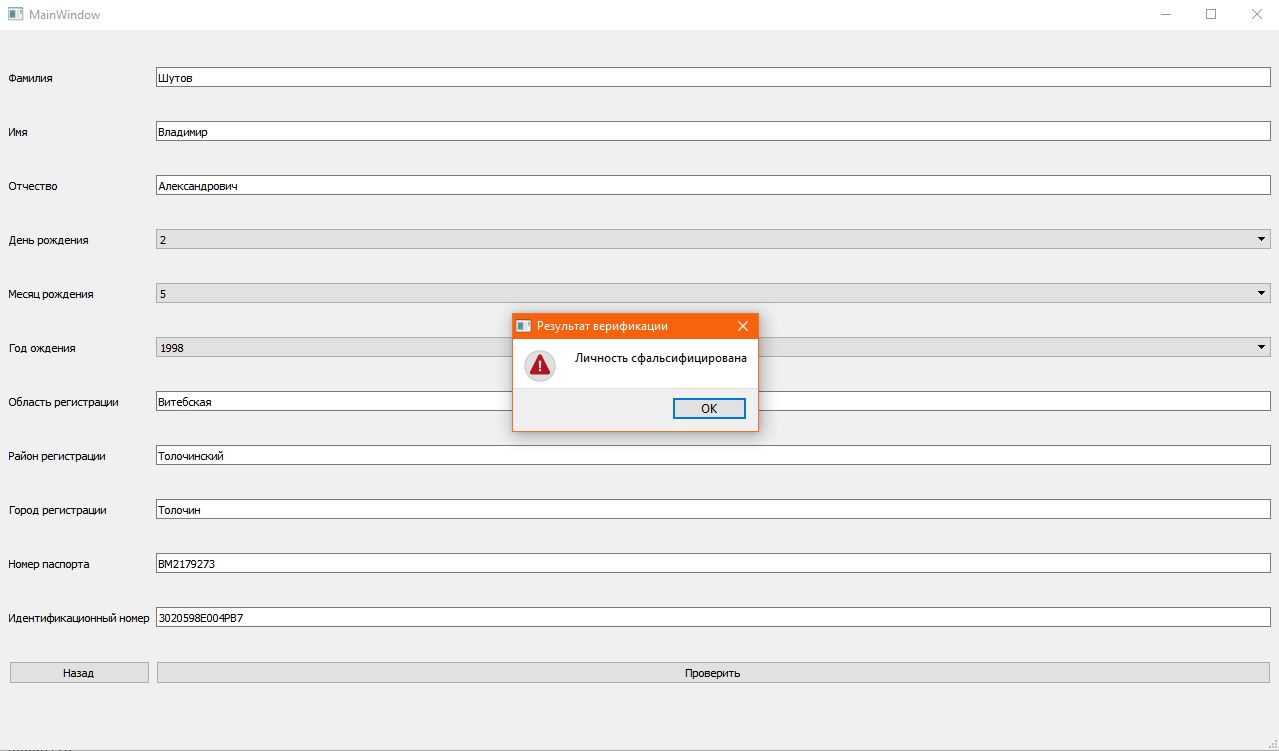
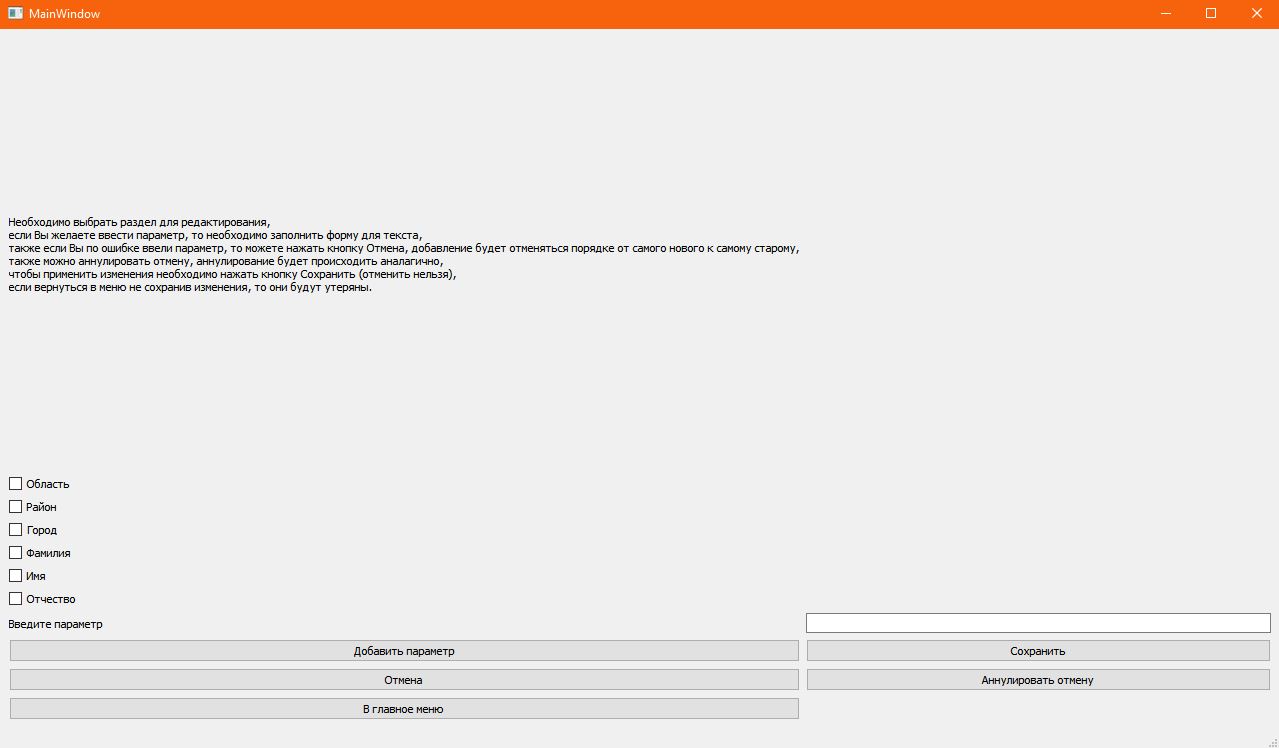
Рисунок Д.5 –­ Отрицательный результат верификации

Рисунок Д.6 –­ Окно добавления параметра

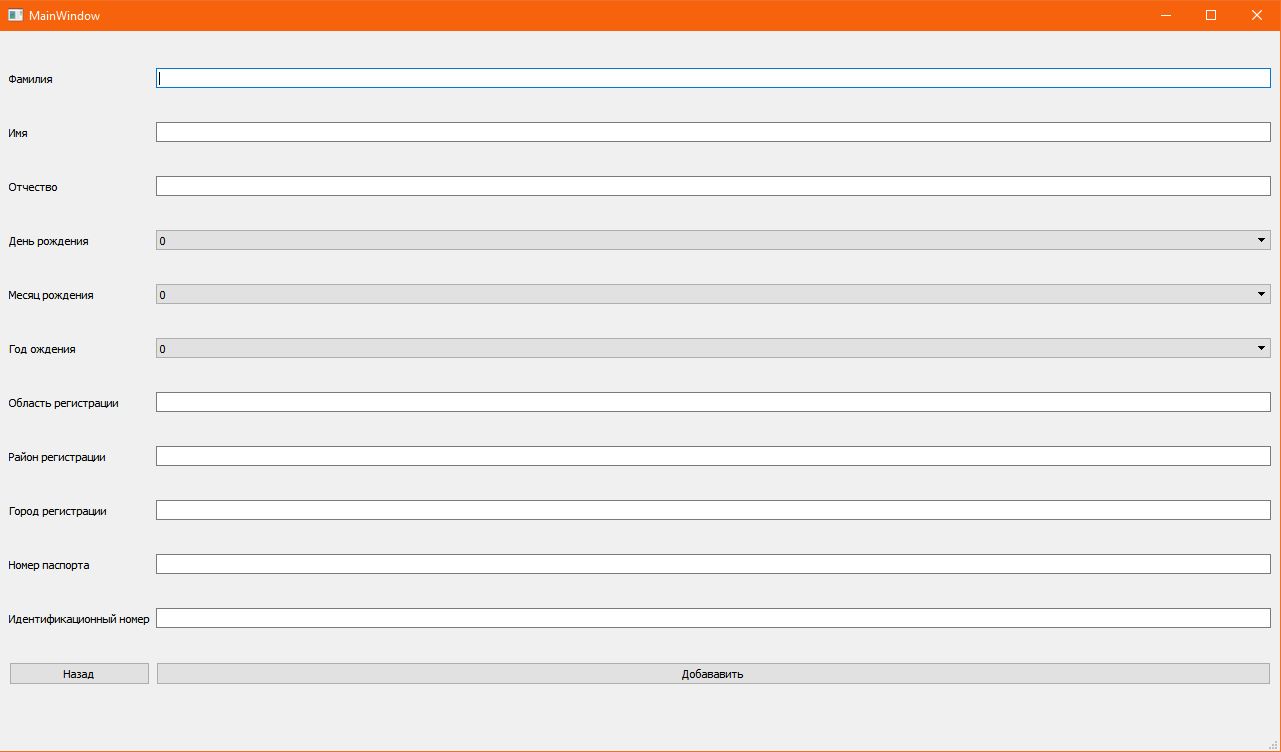


Рисунок Д.7 –­ Окон добавления гражданина

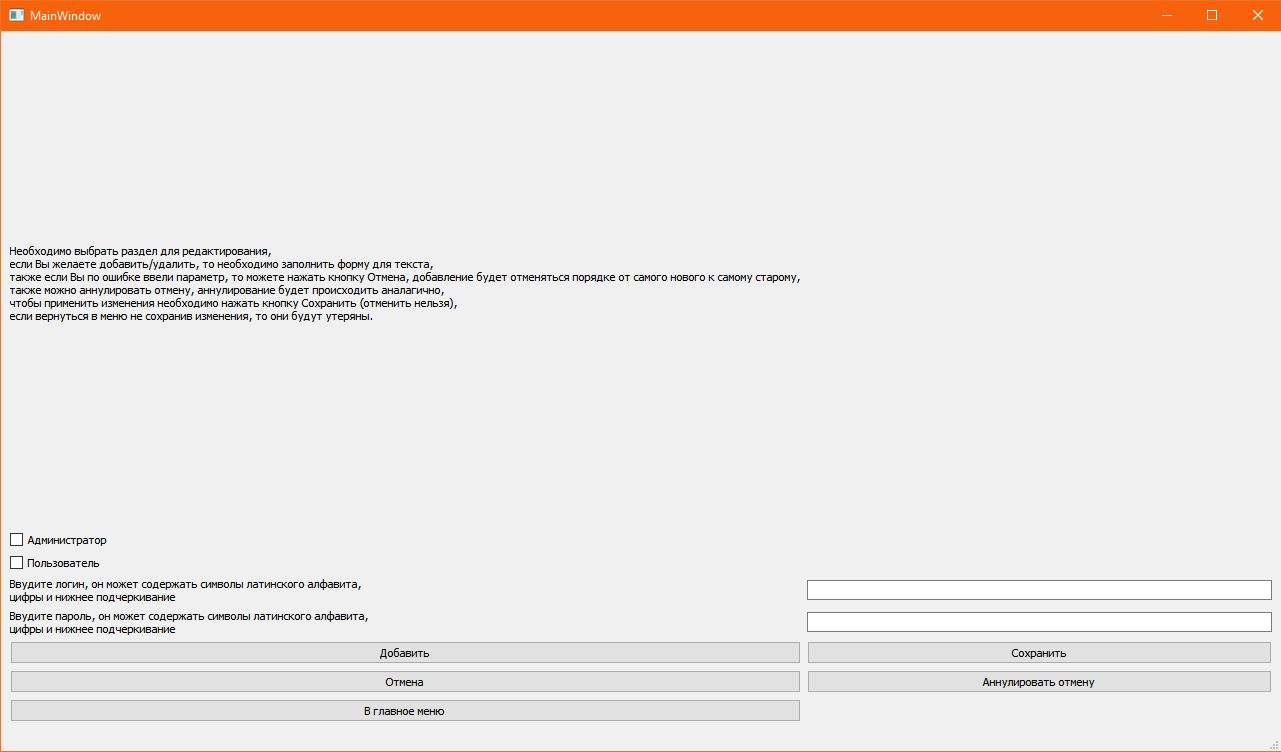


Рисунок Д.8 –­ Окон добавления пользователя

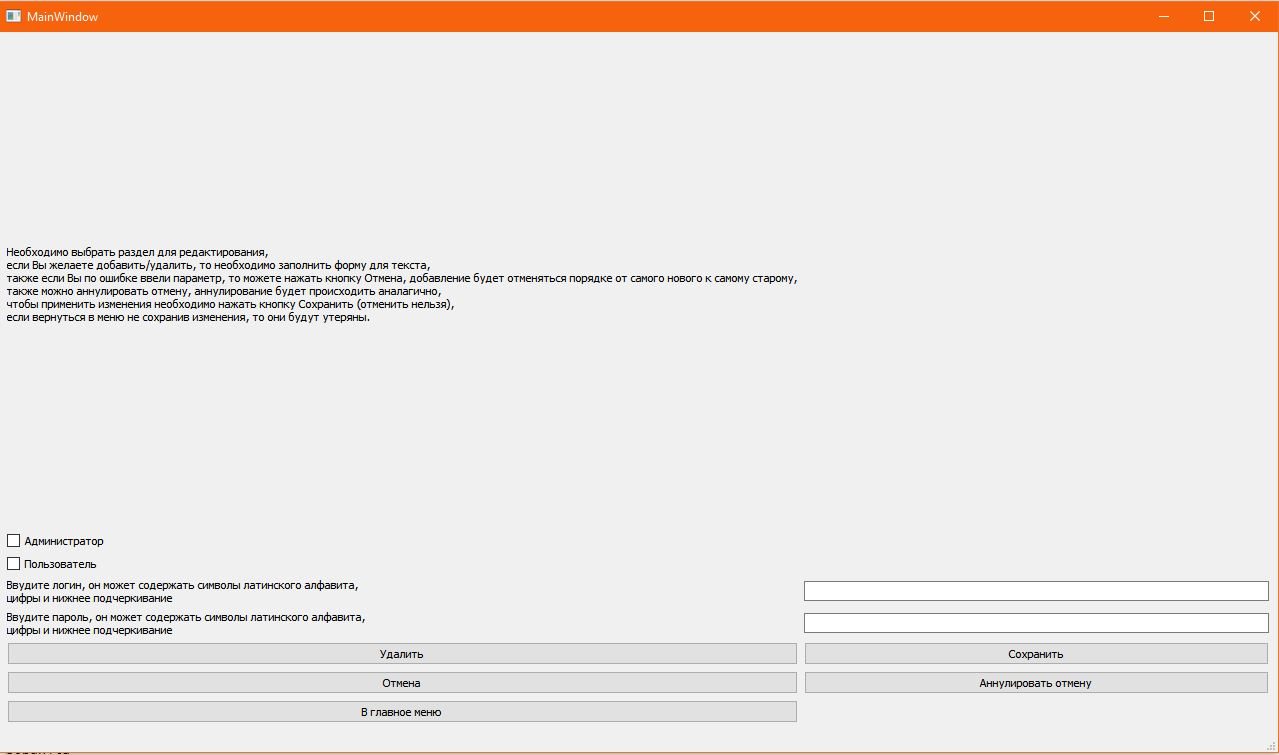


Рисунок Д.9 –­ Окон удаления пользователя

**ПРИЛОЖЕНИЕ Е**

*(обязательное)*

Ведомость документов