МИНОБРНАУКИ РОССИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт Компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

Направление: 02.03.01 Математика и компьютерные науки

Алгоритмизация и программирование
Отчёт по курсовой работе

Реализация телефонного справочника с использованием библиотеки Qt

Студент,
группы 5130201/20001 _____ Якунин Д. Д.

Преподаватель _____ Глазунов В. В.

«_____»_____ 2023г.

Санкт-Петербург, 2023

Содержание

Введение

Необходимо написать приложение телефонный справочник на языке C++ с использованием библиотеки Qt.

Основные требования к приложению:

1) Обязательные поля для хранения: Имя, Фамилия, Отчество, адрес, дата рождения, email, телефонные номера (рабочий, домашний, служебный) в любом количестве.

2) Проверка всех вводимых данных на корректность при помощи регулярных выражений;

3) Добавление и удаление записей, а также редактировавание всех полей;

4) Сортировка отображаемых данных по указанному полю и поиск по нескольким полям;

5) Хранение и загрузка данных осуществить в виде файла.

Приложение необходимо написать с использованием фреймворка Qt. Qt — это библиотека классов C++ и набор инструментального программного обеспечения для создания кросс-платформенных приложений с графическим интерфейсом (GUI).

1 Постановка задачи

Цель работы: написать приложение телефонный справочник с использованием библиотеки Qt.

Для этого необходимо:

1) Организовать проверку всех вводимых данных на корректность при помощи регулярных выражений со следующими условиями:

    1.1) Фамилия, Имя или Отчество — должны содержать только буквы и цифры различных алфавитов, а также дефис и пробел, но при этом должны начинаться только на буквы, и не могли бы оканчиваться или начинаться на дефис. Все незначимые пробелы перед и после данных должны удаляться;

    1.2) Телефон должен быть записан в международном формате;

    1.3) Дата рождения должна быть меньше текущей даты, число месяцев в дате должно быть от 1 до 12, число дней от 1 до 31, причем должно учитываться различное число дней в месяце и високосные года;

    1.4) E-mail должен содержать в себе имя пользователя состоящее из латинских букв и цифр, символ разделения пользователя и имени домена(@), а также сам домен состоящий из латинских букв и цифр.

2) Позволить добавлять и удалять записи, а также редактировать все поля;

3) Реализовать поиск записей в таблице по нескольким полям посредством поля для ввода текста;

4) Реализовать сортировку данных в таблице по указанному полю;

5) Реализовать функцию записи данных в файл;

6) Реализовать чтение данных из файла.

2 Реализация

2.1 Main

В функции main создаётся экземпляр класса MainWindow и отображается на экране.

2.2 myTableWidgetItem

Класс myTableWidgetItem представляет из себя ячейку таблицы, но имеющую дополнительный слот.

Сам класс является дочерним к встроенным классам QObject и QTableWidgetItem.

2.2.1 Слот setTextSlot

Входные параметры: const QDate& date

Выходные параметры: отсутствуют.

Этот слот принимает дату в виде объекта встроенного класса QDate и с помощью его методов year, month и day приводит всю дату в единую строку встроенного класса QString в формате "yyyy.MM.dd"и записывает эту строку в ячейку.

2.3 MainWindow

Класс MainWindow, наследован от встроенного класса QMainWindow, а значит может отображаться в качестве элемента пользовательского интерфейса и быть родителем для других графических элементов, в нашем случае он отвечает за весь пользовательский интерфейс телефонного справочника.

Через него добавляются и настраиваются все кнопки на экране, а также отрисовывается таблица телефонного справочника и любое взаимодействие с ней.

2.3.1 Конструктор класса

При создании основного окна на нем отрисовываются все виджеты: таблица, кнопки и поле для ввода текста.

После отрисовки идет первичная настройка таблицы:

1. Включается сортировка по столбцам методом setSortingEnabled встроенного класса QTableWidget;

2. Задается количество столбцов и их названия методами setColumnCount и setHorizontalHeaderLabels;

3. Задается прототип ячейки таблицы методом setItemPrototype;

4. Задается выделение по строкам методом setSelectionBehavior;

5. Связывается сигнал изменения ячейки cellChanged и слот её проверки регулярным выраженем cellSaver connect(ui->tableWidget, SIGNAL(cellChanged(int,int)), this, SLOT(cellChecker(int,int)));

6. Связывается сигнал нажатия на ячейку cellPressed и слот сохранения её текста в поле класса cellChecker connect(ui->tableWidget, SIGNAL(cellPressed(int,int)), this, SLOT(cellSaver(int,int)));

2.3.2 Кнопка Add Row

На сигнал **clicked** этой кнопки реагирует слот **on_addButton_clicked** класса MainWindow. По нажатию в таблице появляется новая пустая строка с виджетом QDateEdit с сегодняшней датой в поле для даты.

Для связи между кнопкой и слотом используется connect(ui->addButton, SIGNAL(clicked), this, SLOT(on_addButton_clicked)), где this – ссылка на объект класса MainWindow.

Реализована кнопка через стандартный виджет Qt QToolButton.

2.3.3 Кнопка Delete Rows

На сигнал **clicked** этой кнопки реагирует слот **on_deleteButton_clicked** класса MainWindow. По нажатию из таблицы удаляются все выделенные строки.

Для связи между кнопкой и слотом используется connect(ui->deleteButton, SIGNAL(clicked), this, SLOT(on_deleteButton_clicked)), где this – ссылка на объект класса MainWindow.

Реализована кнопка через стандартный виджет Qt QToolButton.

2.3.4 Кнопка Read from file

На сигнал **clicked** этой кнопки реагирует слот **on_readButton_clicked** класса MainWindow. По нажатию открывается для чтения файл справочника, из него считываются поля и вся таблица заполняется соответствующим количеством строк.

Для связи между кнопкой и слотом используется connect(ui->readButton, SIGNAL(clicked), this, SLOT(on_readButton_clicked)), где this – ссылка на объект класса MainWindow.

Реализована кнопка через стандартный виджет Qt QToolButton.

2.3.5 Кнопка Save to file

На сигнал **clicked** этой кнопки реагирует слот **on_writeButton_clicked** класса MainWindow. По нажатию для записи открывается файл со справочником и полностью переписывается текущим состоянием справочника из приложения.

Для связи между кнопкой и слотом используется connect(ui->writeButton, SIGNAL(clicked), this, SLOT(on_writeButton_clicked)), где this – ссылка на объект класса MainWindow.

Реализована кнопка через стандартный виджет Qt QToolButton.

2.3.6 Кнопка Search

На сигнал **clicked** этой кнопки реагирует слот **on_searchButton_clicked** класса MainWindow. По нажатию в таблице во всех ячейках ищется частичное совпадение с текстом в поле lineEdit и все подходящие строки таблицы выделяются.

Для связи между кнопкой и слотом используется connect(ui->searchButton, SIGNAL(clicked), this, SLOT(on_searchButton_clicked)), где this – ссылка на объект класса MainWindow.

Реализована кнопка через стандартный виджет Qt QPushButton.

2.3.7 Поле для текста lineEdit

Вспомогательный виджет, в который можно записать свой текст и по нему будут искаться подходящие строки в таблице. Для реализации использован стандартный виджет Qt QLineEdit.

2.4 Методы MainWindow

2.4.1 Слот on_addButton_clicked

Входные параметры: отсутствуют.

Выходные параметры: отсутствуют.

Этот слот добавляет в таблицу новую пустую строку, добавляет в неё встроенный виджет с датой QDateEdit и связывает изменение даты на этом виджете с изменением текста в самой ячейке через connect(dateEdit, SIGNAL(dateChanged(QDate)), tmpItem, SLOT(setTextSlot(QDate))).

2.4.2 Слот on_deleteButton_clicked

Входные параметры: отсутствуют.

Выходные параметры: отсутствуют.

Этот слот удаляет из таблицы все выделенные на момент вызова строки.

2.4.3 Слот on_writeButton_clicked

Входные параметры: отсутствуют.

Выходные параметры: отсутствуют.

Этот слот открывает для записи файл со справочником. Если такого файла нет, то выводится окно с предупреждением о невозможности записать данные. После открытия файла в него записываются данные из всех строк таблицы: текст ячеек разделяется запятой, в каждую строку файла записывается одна строчка таблицы. Для хранения данных используется файл формата .txt.

2.4.4 Слот on_readButton_clicked

Входные параметры: отсутствуют.

Выходные параметры: отсутствуют.

Этот слот открывает для чтения файл со справочником. Если такого файла нет, то выводится окно с предупреждением о невозможности считать данные. После открытия файла из него записываются данные из всех строк: разделенные запятой пункты из файла записываются в соответствующие ячейки, в каждую строку таблицы записывается одна строчка файла.

Для столбца с файлом отдельно в каждой строке создается виджет QDateEdit и сигнал на его изменение связывается со слотом изменения текста ячейки за ним – connect(DateEdit, SIGNAL(dateChanged(QDate)), tmp, SLOT(setTextSlot(QDate))).

Чтение данных происходит из файла формата .txt.

2.4.5 Слот on_searchButton_clicked

Входные параметры: отсутствуют.

Выходные параметры: отсутствуют.

Этот слот убирает выделение со всех строк в таблице, после чего проверяет текст каждой ячейки на частичное совпадение с текстом из виджета lineEdit и выделяет строки с подходящими ячейками.

2.4.6 Метод cellSaver

Входные параметры: int row, int col - индекс ячейки

Выходные параметры: отсутствуют

Метод вызывается при выделении ячейки и записывает её текст в поле QString tmpCellStr класса. Это нужно, чтобы вернуть в ячейку тот текст, который был до изменения, если новый не подошел.

2.4.7 Метод cellChecker

Входные параметры: int row, int col - индекс ячейки

Выходные параметры: отсутствуют

Метод вызывается при изменении ячейки таблицы и проверяет, подходит ли новый текст под регулярные выражения. Если новый текст не подходит, то вместо нового текста записывается старый.

2.4.8 Сортировка данных в таблице

Сортировка данных в таблице доступна благодаря тому, что в реализации таблицы был использован класс QTableWidget библиотеки Qt, который обладает встроенной сортировкой по столбцам. Сортировка происходит по тексту, записанному в ячейках таблицы.

2.5 Регулярные выражения проверки текста

Для проверки корректности введенного в ячейки текста вызываются соответствующая каждой колонке функция проверки. Реализация всех функций одинаковая, различаются лишь регулярные выражения для каждого столбца.

Входные параметры: QString&

Выходные параметры: bool - подходит ли входная строка под регулярное выражение

Функция принимает ссылку на строку и с помощью объекта QRegularExpression и соответствующего регулярного выражения и его метода match проверяет соответствие и результат возвращает.

1. Для проверки имени было использовано выражение

   ```
   "^[A-Za-zA-Яа-яЁё]+[- ]?[A-Za-zA-Яа-яЁё]*$"
   ```

   Оно значит, что строка начинается с одной или более букв латиницы или кириллицы, далее опциаонально символ «-» или пробел, после которых идет любое количество букв. Доллар в конце значит, что строка должна закончится именно буквой.

2. Для проверки телефона было использовано выражение

   ```
   "^(\\+\\d{1,3}\\(\\d{1,4}\\)\\d{1,4}-\\d{1,9};)+$"
   ```

   Оно значит, что строка начинается с символа «+», после идет 1-3 цифры, после них открывающая скобка, 1-4 цифры, закрывающая скобка, 1-4 цифры, дефис, 1-9 цифр и символ «;». Всё это выражение обернуто в скобки, сначала идет символ «^», значащий, что строка должна обязательно начинаться именно номером. После скобок символы «+», означающий, что номер должен быть не менее одного раза, и «$», означающий, что строка должна закончится номером;

3. Для проверки email было использовано выражение

   ```
   R"(^[A-Za-z0-9._]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$)"
   ```

   Оно значит, что строка должна начинаться не менее, чем с одного символа из скобок: буквы, цифры, точки, символы подчеркивания. После этого идет символ «@», далее идет домен почты, в котором могут быть буквы, цифры, точки и символы минуса. После домена идет точка, после которой идут 2 или более буквы латиницы, доллар же значит, что строка обязательно должна заканчиваться ими. Буква «R» перед регулярным выражением в Qt означает, что это выражение должно быть интерпретировано как строка в формате Qt. Это необходимо для того, чтобы Qt мог правильно обрабатывать специальные символы в регулярном выражении

3 Тестирование приложения

При запуске программы перед нами открывается главное окно (Рис. 1). На нем будет пустая таблица и кнопки для взаимодействия с ней.

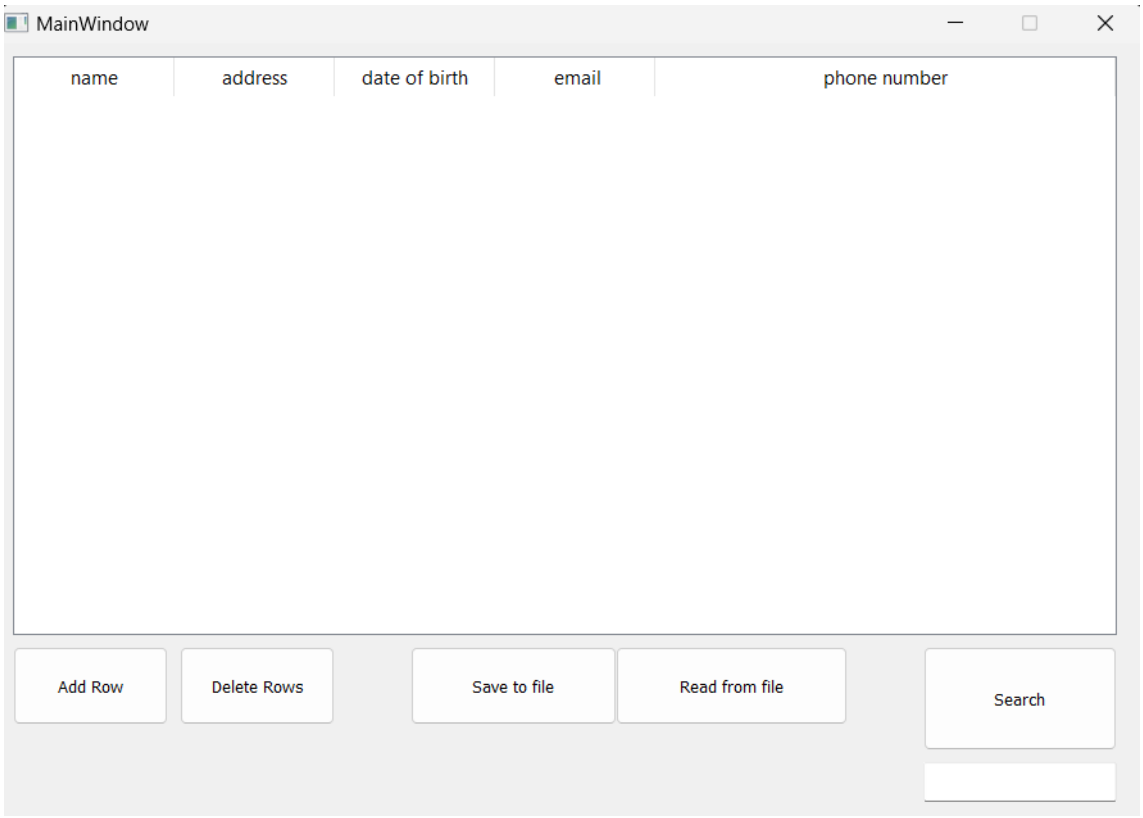

Рис. 1. Главное окно

После нажатия на кнопку "Add Row"в таблицу добавляется пустая строчка с сегодняшней датой (Рис.2).

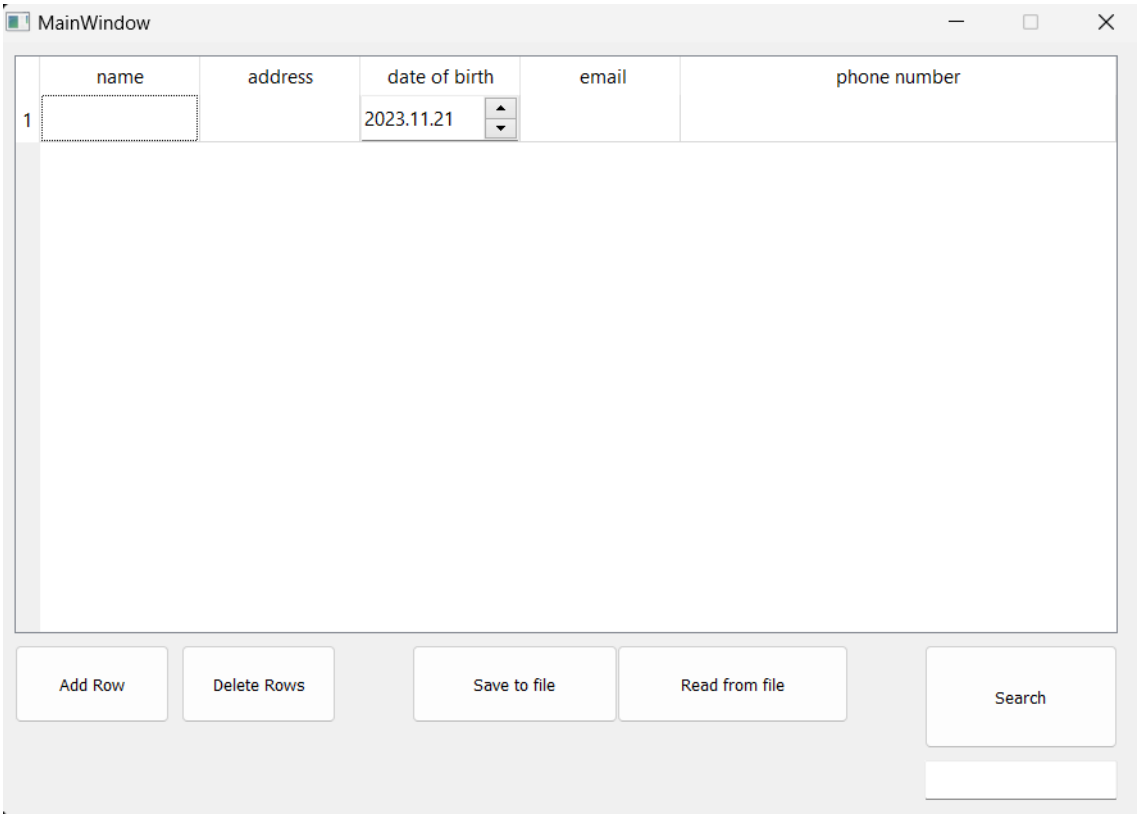

Рис. 2. Таблица с пустой строчкой

При нажатии кнопки Read from file таблица заполняется данными из файла (Рис.3).

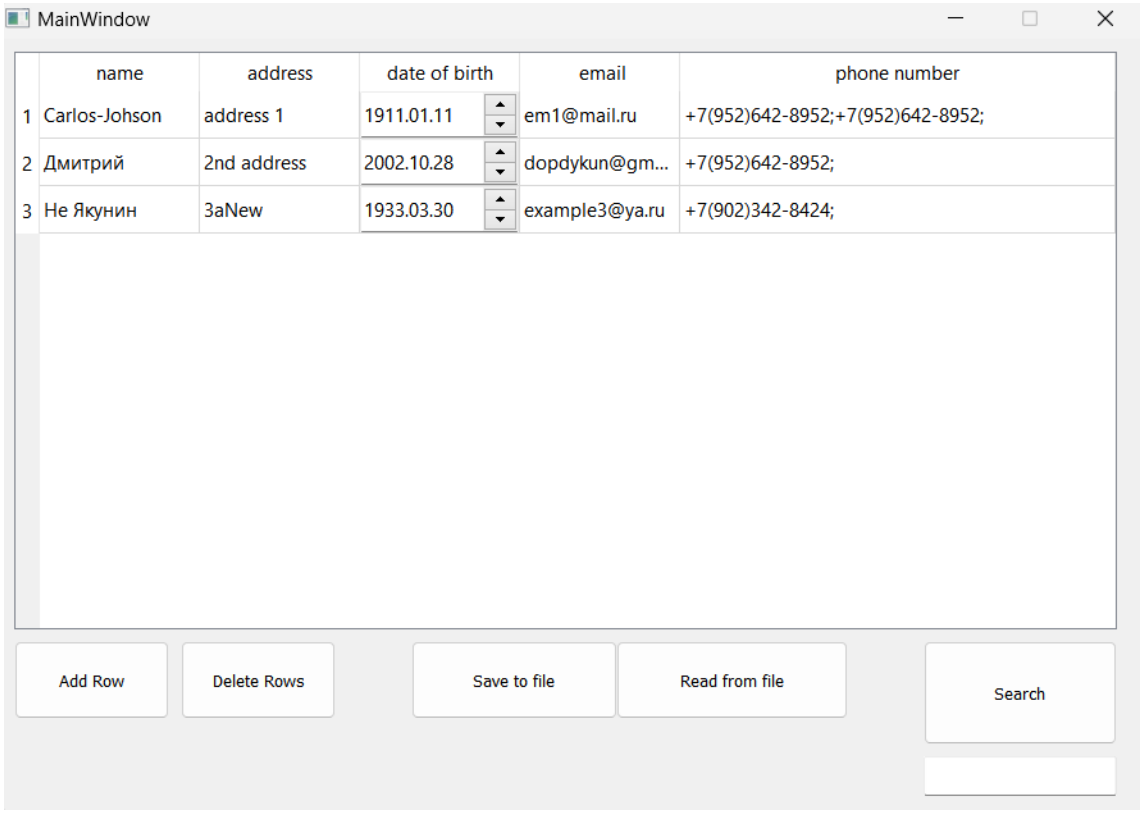

Рис. 3. Заполнение данными из файла

При двойном нажатии на ячейку появляется возможность изменить её текст (Рис.4).

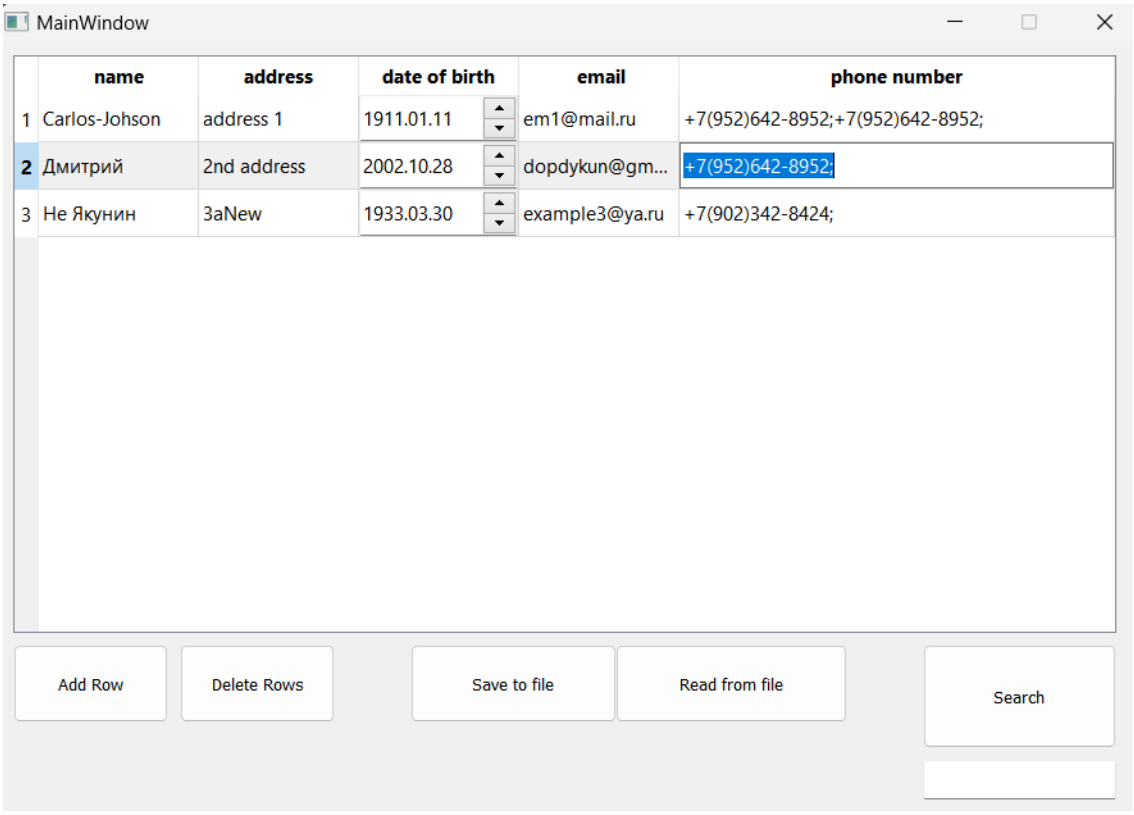

Рис. 4. Изменение ячейки

На рисунках 5 и 6 отображены различные сортировки по столбцу "date of birth такую же сортировку можно применять к любому из столбцов.

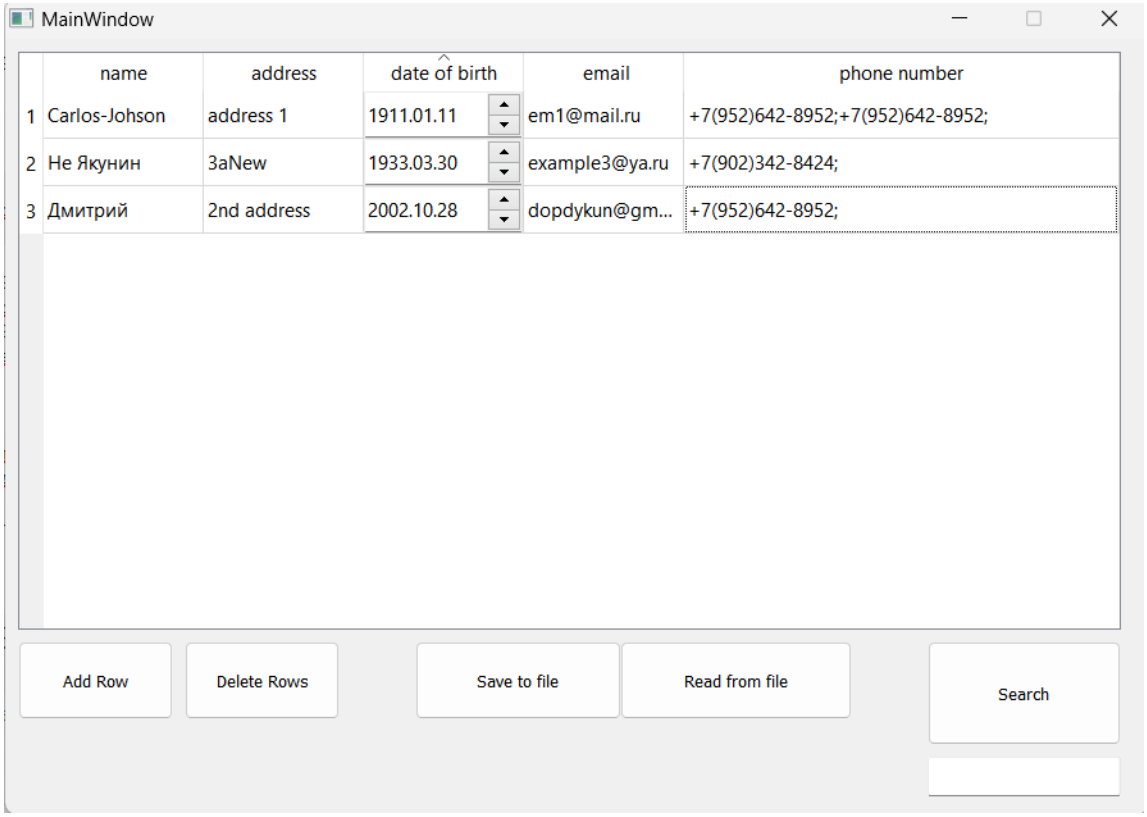

Рис. 5. Сортировка по возрастанию.

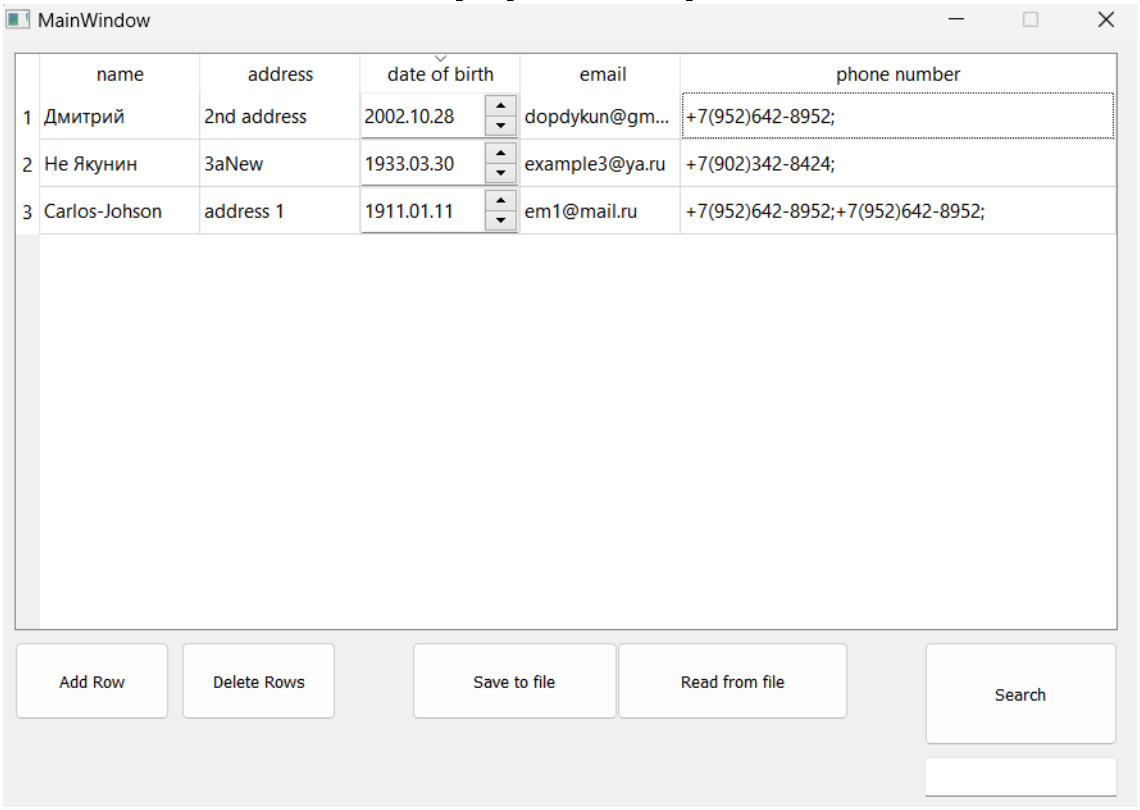

Рис. 6. Сортировка по убыванию.

При нажатии на кнопку Search выполняется поиск по тексту из поля под кнопкой. На рисунке 7 результаты поиска по слову "якунин".

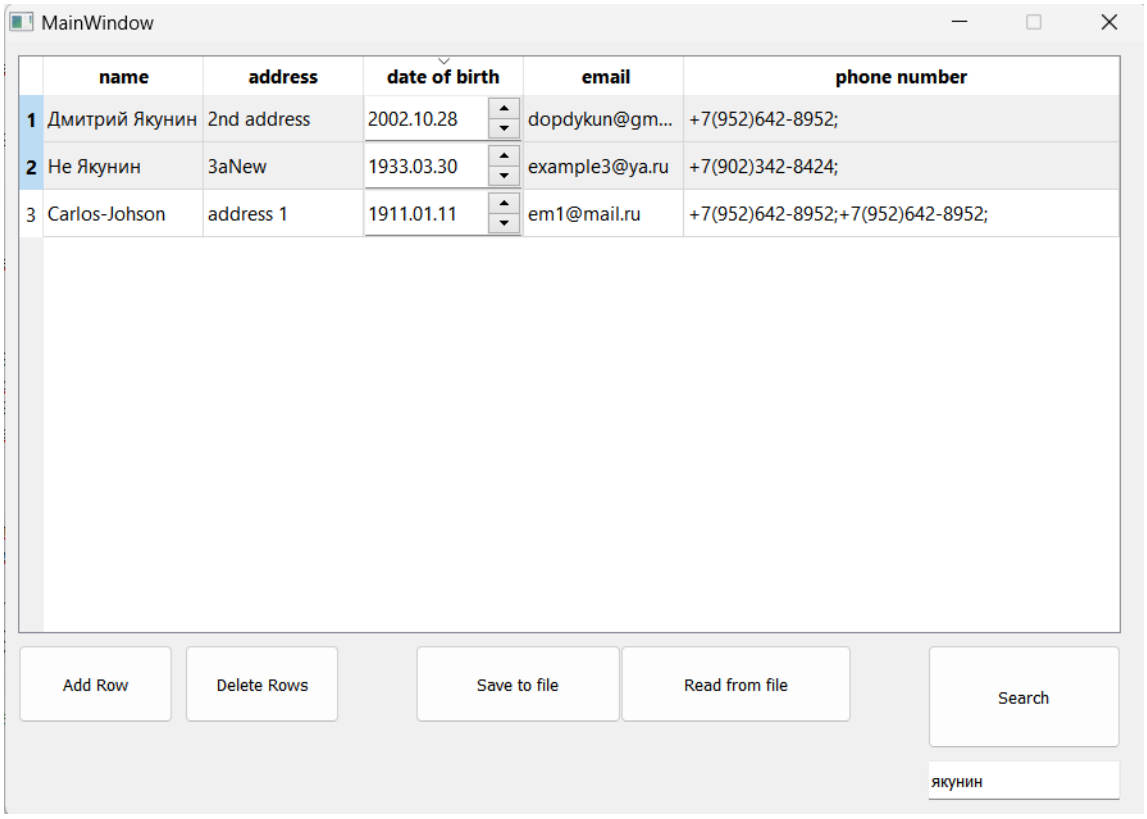

Рис. 7. Поиск по таблице

После поиска по колонкам можно сразу нажать на кнопку Delete Rows и найденные колонки будут удалены (Рис. 8).

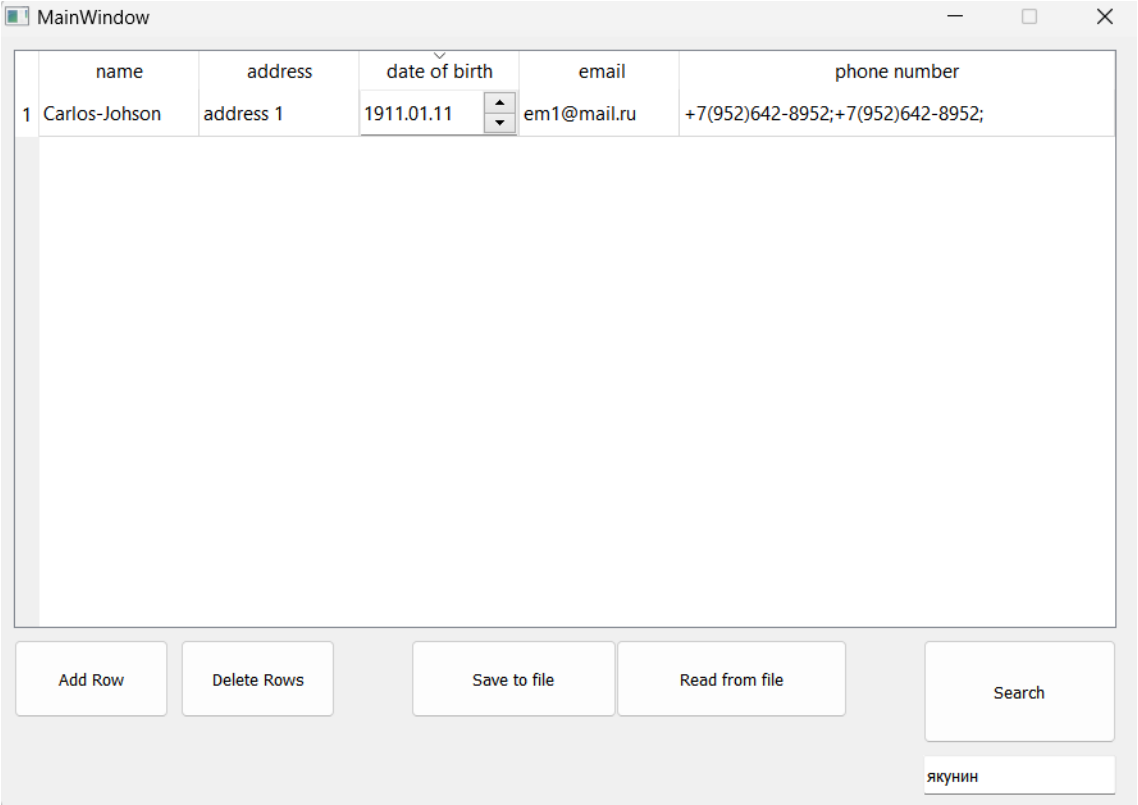

Рис. 8. Таблица с удаленными колонками

При вводе некорректного выражения оно будет заменено на то, которое было в ячейке до изменения (Рис. 9).

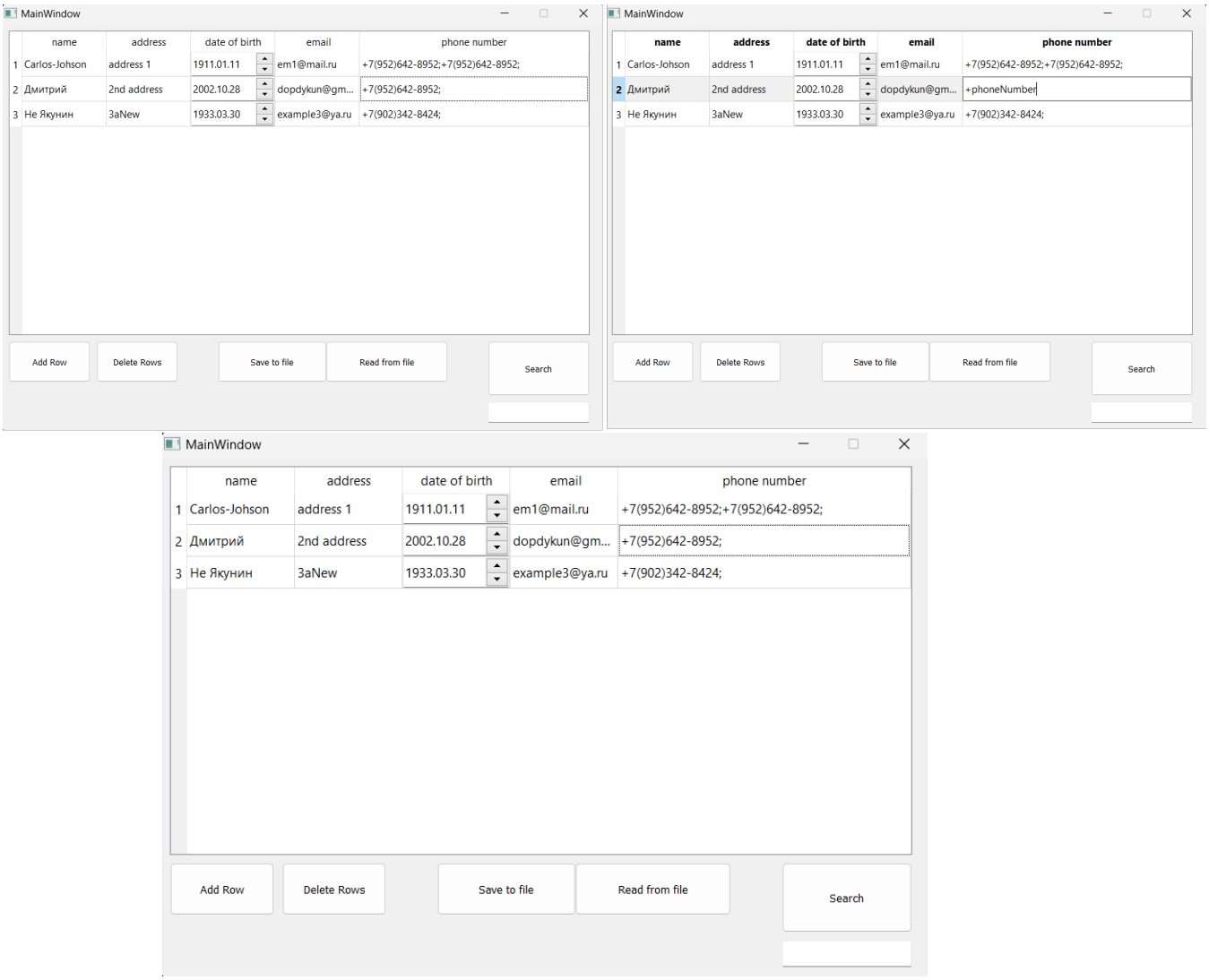

Рис. 9. Таблица до, во время и после ввода некорректного текста

Заключение

В процессе выполнения работы были выполнены следующие задачи:

1) Организована проверка всех вводимых данных на корректность. Для этого реализованы функции, основанные на регулярных выражениях, вызывающиеся каждый раз при изменении данных ячейки;

2) Реализована возможность добавления новых данных в таблицу, полученных от пользователя при помощи метода on_addbutton_clicked класса MainWindow;

3) Реализована возможность удаления записей в таблице при помощи метода on_deletebutton_clicked класса MainWindow;

4) Уже существующие записи в таблице возможно изменять благодаря тому, что ячейки таблицы myTableWidgetItem – класс-наследник от QTableWidgetItem, в котором возможность изменения ячейки уже реалиована;

5) Реализован поиск записей в таблице по нескольким полям посредством поля для ввода текста QLineEdit при помощи метода on_searchButton_clicked класса MainWindow;

6) Сортировать данные в таблице возможно благодаря уже реализованному в классе QTableWidget функционалу. Для использования встроенной сортировка использовался метод setSortingEnabled;

7) Реализована функция записи данных в файл формата .txt при помощи метода on_writeButton_clicked класса MainWindow;

8) Реализована функция чтения данных из файла формата .txt при помощи метода on_readButton_clicked класса MainWindow.

Была достигнута основная поставленная задача - написан телефонный справочник, в приложении имеются все необходимые поля и функции.

Для написания приложения был использован язык C++, стандарт ISO C++ 11. Для работы была использована среда Qt Creator 5.0.2 с компилятором MinGW 7.3.0 64-bit.

Дополнительная информация для написания кода бралась из программы Qt Assistant.

Был получен опыт разработки полноценного приложения, который может быть использован в будущем в учебе и работе.

Приложение А. Исходный код функции main

A.1 main.cpp

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Приложение Б. Исходный код MainWindow

Б.1 mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QFile>
#include <QMessageBox>
#include <QTextStream>
#include <QDebug>
#include <QCalendarWidget>
#include <QDateEdit>
#include <QRegularExpression>
#include "mytablewidgetitem.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

QString dateToString(QDate date);


class MainWindow : public QMainWindow
{
  Q_OBJECT

  public:
  MainWindow(QWidget *parent = nullptr);
  ~MainWindow();

  private slots:

  void on_deleteButton_clicked();
  void on_addButton_clicked();
  void on_writeButton_clicked();
  void on_readButton_clicked();
  void on_searchButton_clicked();

  void cellChecker(int row, int col);
  void cellSaver(int row, int col);

  private:
  Ui::MainWindow *ui;
  QString tmpCellStr;
};
#endif // MAINWINDOW_H
```

Б.2 mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
: QMainWindow(parent)
, ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->tableWidget->setColumnCount(5);
    ui->tableWidget->setHorizontalHeaderLabels(QStringList{"name",
        "address", "date of birth"," email", "phone number"});
    ui->tableWidget->setSortingEnabled(true);

    //ui->tableWidget->setSelectionMode(QAbstractItemView::
       SingleSelection);
    ui->tableWidget->setItemPrototype(new myTableWidgetItem());
    ui->tableWidget->setSelectionBehavior(QAbstractItemView::
       SelectRows);

    ui->tableWidget->horizontalHeader()->setStretchLastSection(
       true);
    this->setWindowFlags( this->windowFlags() |  Qt::
       MSWindowsFixedSizeDialogHint );

    tmpCellStr = "";
    connect(ui->tableWidget, SIGNAL(cellPressed(int,int)), this,
       SLOT(cellSaver(int,int)));
    connect(ui->tableWidget, SIGNAL(cellChanged(int,int)), this,
       SLOT(cellChecker(int,int)));
}

MainWindow::~MainWindow() {
    delete ui;
}


void MainWindow::on_deleteButton_clicked()
{
    QModelIndexList selectedRows = ui->tableWidget->selectionModel
       ()->selectedRows();
    while (!selectedRows.empty()) {
      ui->tableWidget->removeRow(selectedRows[0].row());
      selectedRows = ui->tableWidget->selectionModel()->
         selectedRows();
    }
}
```

```
38
39
40 void MainWindow::on_addButton_clicked()
41 {
42   ui->tableWidget->insertRow(ui->tableWidget->rowCount());
43
44   QDateEdit* dateEdit = new QDateEdit();
45   dateEdit->setMaximumDate(QDate::currentDate());
46   dateEdit->setMinimumDate(QDate(1900, 1, 1));
47   dateEdit->setDate(QDate::currentDate());
48   dateEdit->setDisplayFormat("yyyy.MM.dd");
49
50   int rows = ui->tableWidget->rowCount()-1;
51   ui->tableWidget->setCellWidget(rows, 2, dateEdit);
52   QDate tmp = dateEdit->date();            //qobject_cast<
       QDateEdit*>(ui->tableWidget->cellWidget(ui->tableWidget->
       rowCount()-1 , 2))
53
54   myTableWidgetItem* tmpItem = new myTableWidgetItem(
       dateToString(tmp));
55   ui->tableWidget->setItem(rows, 2, tmpItem);
56
57   connect(dateEdit, SIGNAL(dateChanged(QDate)), tmpItem, SLOT(
       setTextSlot(QDate)));
58 }
59
60
61 void MainWindow::on_writeButton_clicked()
62 {
63   QFile file("C:/Qt/lab8_3sem/datebase1.txt");
64   if (!file.open(QFile::WriteOnly | QFile::Text)) {
65     QMessageBox::warning(this, "title", "file not found");
66   }
67
68   QTextStream out(&file);
69   QString text;
70
71   for (int i = 0; i < ui->tableWidget->rowCount(); i++) {
72     for (int j = 0; j < ui->tableWidget->columnCount(); j++) {
73       if (j != 2){
74         QTableWidgetItem *tmp = ui->tableWidget->item(i, j);
75
76         if (nullptr != tmp) text += tmp->text() + ',';
77         else text += ',';
78       } else {
79         QDate tmp = (qobject_cast<QDateEdit*>(ui->tableWidget->
             cellWidget(i,j)))->date();
```

```
80            text += QString::number(tmp.year()) + ',' + QString::
                  number(tmp.month())
81           + ',' + QString::number(tmp.day()) + ',';
82         }
83       }
84
85       text += '\n';
86     }
87
88     out << text;
89     file.flush();
90     file.close();
91 }
92
93
94 void MainWindow::on_readButton_clicked()
95 {
96     QFile file("C:/Qt/lab8_3sem/datebase1.txt");
97     if (!file.open(QFile::ReadOnly | QFile::Text)) {
98       QMessageBox::warning(this, "title", "file not found");
99     }
100
101    QTextStream in(&file);
102    int i = 0;
103
104    ui->tableWidget->setRowCount(0);
105    ui->tableWidget->setSortingEnabled(false);
106
107    while (!in.atEnd()) {
108      on_addButton_clicked();
109
110      int flag = 0;
111      QString text = in.readLine();
112      QStringList list = text.split(u',');
113
114      for (int j = 0; j < ui->tableWidget->columnCount(); j++) {
115        if (j != 2) {
116          myTableWidgetItem *tmp = new myTableWidgetItem(list[j +
                    flag]);
117          ui->tableWidget->setItem(i, j, tmp);
118
119        } else {
120          QDateEdit *DateEdit = qobject_cast<QDateEdit*>(ui->
                    tableWidget->cellWidget(i,j));
121          DateEdit->setDisplayFormat("yyyy.MM.dd");
122          DateEdit->setDate(QDate((list[j]).toInt(), (list[j + 1])
                    .toInt(), (list[j + 2]).toInt()));
123          flag = 2;
```

```
124             myTableWidgetItem *tmp = new myTableWidgetItem(
                    dateToString(DateEdit->date()));
125             ui->tableWidget->setItem(i, j, tmp);
126 
127             connect(DateEdit, SIGNAL(dateChanged(QDate)), tmp, SLOT(
                    setTextSlot(QDate)));
128         }
129       }
130 
131       i++;
132     }
133 
134     file.close();
135     ui->tableWidget->setSortingEnabled(true);
136 }
137 
138 
139 void MainWindow::on_searchButton_clicked() {
140     ui->tableWidget->clearSelection();
141 
142     if (ui->lineEdit->text() == "") return;
143 
144     //QRegularExpression regex(ui->lineEdit->text()); // R(ui->
           lineEdit->text()) проблема с поиском при специальных символ
           ax
145     for (int i = 0; i < ui->tableWidget->rowCount(); i++) {
146       for (int j = 0; j < ui->tableWidget->columnCount(); j++) {
147         if (ui->tableWidget->item(i, j) == 0) continue;
148 
149         if (ui->tableWidget->item(i, j)->text().toLower().indexOf(
                ui->lineEdit->text().toLower()) != -1) {
150           ui->tableWidget->selectionModel()->select(ui->
                  tableWidget->model()->index(i, j),
151           QItemSelectionModel::Select | QItemSelectionModel::Rows)
                  ;
152         }
153         /*QString str(ui->tableWidget->item(i, j)->text());
154         QRegularExpressionMatch match = regex.match(str);
155 
156         if (match.hasMatch())
157         ui->tableWidget->selectionModel()->select(ui->tableWidget
              ->model()->index(i, j),
158         QItemSelectionModel::Select | QItemSelectionModel::Rows);
              */
159       }
160     }
161 
162 }
```

```
163
164
165 void MainWindow::cellChecker(int row, int col) {
166   if (col == 2 or col == 1) return;
167
168   QString cellText = ui->tableWidget->item(row, col)->text();
169   if (cellText == "") return;
170   bool flag = 0;
171
172   if (col == 0) flag = isValidName(cellText);
173   if (col == 3) flag = isValidEmail(cellText);
174   if (col == 4) flag = isValidPhone(cellText);
175
176   if (!flag) ui->tableWidget->item(row, col)->setText(tmpCellStr
      );
177 }
178
179 void MainWindow::cellSaver(int row, int col) {
180   if (col == 2 or col == 1) return;
181   tmpCellStr = ui->tableWidget->item(row, col)->text();
182 }
183
184 QString dateToString(QDate date) {
185   QString year =  QString::number(date.year());
186   QString month = QString::number(date.month());
187   QString day = QString::number(date.day());
188
189   if (month.size() == 1) month = "0" + month;
190   if (day.size() == 1) day = "0" + day;
191
192   return year + "." + month + "." + day;
193 }
```

Приложение B. Исходный код myTableWidgetItem

B.1 myTableWidgetItem.h

```
#ifndef MYTABLEWIDGETITEM_H
#define MYTABLEWIDGETITEM_H

#include <QWidget>
#include <QTableWidgetItem>
#include <QDate>
#include "mainwindow.h"

bool isValidEmail(QString& email);
bool isValidPhone(QString& phone);
bool isValidName(QString& name);

class myTableWidgetItem : public QObject, public
    QTableWidgetItem
{
  Q_OBJECT
   public:
   myTableWidgetItem();
   myTableWidgetItem(QString);

   signals:


   public slots:
   void setTextSlot(const QDate& date);


};

#endif // MYTABLEWIDGETITEM_H
```

B.2 myTableWidgetItem.cpp

```
#include "mytablewidgetitem.h"

myTableWidgetItem::myTableWidgetItem() : QTableWidgetItem() {}

myTableWidgetItem::myTableWidgetItem(QString str) :
    QTableWidgetItem (str){}

void myTableWidgetItem::setTextSlot(const QDate& date){
  this->setText(dateToString(date));
}

bool isValidName(QString& name) {
  name = name.trimmed();
  QRegularExpression namePattern("^[A-Za-zA-Яа-яЁё]+[- ]?[A-Za-z
     A-Яа-яЁё]*$");
  return namePattern.match(name).hasMatch();
}

bool isValidPhone(QString& phone) {
  phone = phone.trimmed();
  QRegularExpression phonePattern("^(\\+\\d{1,3} \\(\\d{1,4} \\)\\
     d{1,4} -\\d{1,9} ;)+$");
  return phonePattern.match(phone).hasMatch();
}

bool isValidEmail(QString& email) {
  email = email.trimmed();
  QRegularExpression emailPattern(R"(^[A-Za-z0-9._]+@[A-Za-z0
     -9.-]+\.[A-Za-z]{2,} $)");
  return emailPattern.match(email).hasMatch();
}
```