МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

Факультет Инфокоммуникационных сетей и систем
Кафедра Защищенных систем связи
Дисциплина: «Методы и технологии программирования»

Отчет по лабораторной работе №5:

«Работа с SQLite, QT»

Выполнил студент гр. ИБС-01:

Гребенников Тимофей

Проверил:

Ерофеев Сергей Анатольевич

доцент кафедры ПИВТ, кандидат физико-математических наук

Санкт-Петербург

Оглавление

Постановка задачи

Требуется разработать программу с графическим интерфейсом, состоящую из двух основных блоков.

Первый блок должен включать в себя базу данных по странам с отражением таких свойств, как название страны, название столицы, численность населения и государственный строй, включающий информацию о форме правления, административно-территориальном устройстве, а также политическом режиме. База данных должна быть основана на использовании средств SQLite.

Второй блок должен содержать динамический массив случайных целых чисел в диапазоне, который задаёт пользователь. После приёма входных данных и создания массива с использованием указателей программа должна рассчитывать такие показатели, как наибольшее значение в массиве, наименьшее значение в массиве и среднее арифметическое всех значений.

Проект реализован на языке C++ в среде разработки QT Creator (Community) версии 7.0.0.

Техническое задание

- **Список входных данных:**

Название переменной	Назначение	Тип данных
data	все свойства добавляемого в базу данных объекта (название страны, название столицы, численность населения, форма правления, административное-территориальное устройство и политический режим)	QVariantList

- **Список выходных данных:**

Название переменной	Назначение	Тип данных
db	база данных всех добавленных объектов	QSqlDatabase
result	результат обработки массива (может содержать как наибольшее или наименьшее значение, так и среднее арифметическое всех значений, в зависимости от выбранной пользователем опции)	QString

- **Список промежуточных данных:**

Название переменной	Назначение	Тип данных
rowNumber	хранение значения выбранной пользователем строки в таблице базы данных для последующего удаления этой строки	int
warning_1, warning_2, warning_3	информация об ошибках пользователя, допущенных при вводе свойств добавляемого в базу данных объекта	QString
average	среднее арифметическое значений динамического массива	float
minimum, maximum	минимальное и максимальное значение динамического массива	int
styleSheetFile	файл, содержащий настройки стиля программы	QFile
styleSheet	хранение информации из styleSheetFile для применения стиля	QString

- **Контроль входных данных:**

Ввод данных о названии страны, названии столицы и численности населения при добавлении нового элемента в базу данных осуществляется посредством записи информации в специальные поля, именуемые как *LineEdit*. Для осуществления контроля ввода применяется проверка каждой подобной записи с помощью конструкции условного оператора *If-Else*. В случае нахождения программой ошибки в какой-либо из записей, информация об этой ошибке заносится в одну из переменных warning_1, warning_2 или warning_3 и выводится на экран в виде всплывающего окна *QMessageBox*.

Реализация контроля ввода названия страны, названия столицы и численности населения:

```
QString warning_1 = "", warning_2 = "", warning_3 = "";

    if (ui->lineEdit->text() == "") warning_1 = "*Поле <Название
страны> не должно быть пустым. ";
    if (ui->lineEdit_2->text() == "") warning_2 = "*Поле <Название
столицы> не должно быть пустым. ";
    if ((ui->lineEdit_3->text().toInt() == false) || (ui->lineEdit_3-
>text().toInt() <= 0))
        warning_3 = "*Поле <Численность населения> должно содержать
положительное целочисленное значение. ";


    if ((warning_1 == warning_2) && (warning_2 == warning_3))
    {
        ...
    }
    else QMessageBox::warning(this, "Ошибка ввода", "При вводе
              информации об объекте допущены следующие ошибки: "
                  + warning_1 + warning_2 + warning_3 +
       "Пожалуйста, исправьте неточности и повторите ввод.");
```

Остальные свойства добавляемого объекта, относящиеся к государственному строю страны, определяются выбранными пользователем переключателями, называемыми в среде разработки QT *RadioButtons*. Причём определённые переключатели заранее активированы на этапе

открытия окна добавления нового элемента. Это позволяет исключить любые ошибки при вводе.

Ввод диапазона значений динамического массива осуществляется с помощью ползунков (слайдеров). Таким образом, пользователь может выбрать только те границы диапазона, которые установлены на ползунках (для нижней границы предусмотрены значения от -1000 до 0, для верхней – от 1 до 1000).

- **Список функций:**

функция	описание

`void on_dbOpenBtn_clicked();`
`void on_arrOpenBtn_clicked();`
`void on_exitButton_clicked();`
`void on_backBtn_clicked();`
`void on_addButton_clicked();`
`void on_removeButton_clicked();`
`void on_clearButton_clicked();`
`void on_setButton_clicked();`
`void on_resultButton_clicked();`
`void on_resetButton_clicked();`
`void on_cancelButton_clicked();`
`void on_okButton_clicked();`
	Функции обработки нажатий кнопок навигации и кнопок расчёта/добавления значений.

`void on_sortBox_activated(int index);`	Функция выбора режима сортировки таблицы с базой данных.
`void on_databaseViewer_clicked(const QModelIndex &index);`	Функция получения индекса выбранной пользователем строки для последующего удаления этой строки.
`void databaseAdderSlot(QVariantList data);`	Функция-слот для добавления нового элемента в базу данных.
`void databaseAdder(QVariantList);`	Функция-сигнал для обработки функции добавления нового элемента в базу данных.
`void firstWindow();`	Функция-сигнал для инициализации окна главного меню при возврате пользователя с какого-либо блока программы.

`void on_minSlider_valueChanged(int value);` `void on_maxSlider_valueChanged(int value);`	Функции получения значений ползунков при регулировке пользователем границ диапазона массива.
`void databaseWindow();`	Функция-сигнал для открытия окна работы с базой данных.

Структурное описание

При запуске программы открывается окно главного меню, на котором расположены кнопки навигации по программе и кнопка выхода.

В случае нажатия пользователем на кнопку «Открыть базу данных» откроется окно работы с базой данных, содержащее таблицу добавленных элементов, кнопки «Добавить», «Удалить», «Очистить базу данных» и «Вернуться в меню», а также раскрывающийся список с вариантами сортировки таблицы.

При нажатии на кнопку «Удалить» произойдёт удаление из базы данных (и, соответственно из таблицы) выбранной строки. Если строка выбрана не была, то произойдёт удаление первой строки, расположенной в таблице в соответствии с используемым вариантом сортировки.

При нажатии на кнопку «Очистить базу данных» откроется всплывающее окно с предупреждением и кнопками «Yes» и «No». В случае выбора кнопки «Yes» база данных будет полностью очищена, а всплывающее окно закроется. В случае выбора кнопки «No» всплывающее окно просто закроется.

При нажатии кнопки «Добавить» откроется новое окно, в котором пользователь сможет ввести информацию о добавляемом элементе. В данном окне присутствуют кнопки «Ок» и «Отмена». При нажатии на «Ок» произойдёт проверка введённых пользователем данных и, если ошибок ввода не будет обнаружено, элемент добавится в базу, а окно добавления элемента закроется. В противном случае будет выведено всплывающее окно, содержащее информацию о допущенных ошибках. При нажатии на кнопку «Отмена» произойдёт закрытие окна добавления нового элемента.

При нажатии на кнопку «Вернуться в главное меню» произойдёт возврат пользователя к окну главного меню.

Если в главном меню пользователь нажмёт на кнопку «Открыть массив», то откроется окно работы с динамическим массивом. В данном окне присутствует область вывода массива, область вывода результата обработки массива, ползунки для настройки диапазона значений, элементы выбора параметра расчёта, кнопки «Задать», «Вычислить», «Сброс» и «Вернуться в меню».

При нажатии на кнопку «Задать» в область вывода массива запишется инициализированный по заданному диапазону целочисленный динамический массив.

При нажатии на кнопку «Вычислить» произойдёт расчёт одного из трёх параметров (наибольшее значение, наименьшее значение, среднее арифметическое значение). Результат расчёта будет выведен в область вывода результатов. Выбрать параметры расчёта пользователь может с помощью переключателей.

При нажатии на кнопку «Сброс» области вывода массива и вывода результатов обработки массива очистятся.

При нажатии на кнопку «Вернуться в меню» произойдёт закрытие текущего окна и открытие окна главного меню.

Поскольку программа выполняет работу с базой данных по технологии SQLite, все изменения, применённые к базе данных во время работы программы, автоматически сохраняются в файле базы данных *countries.db*.

Диаграмма базы данных

Страны	
PK	ID (INTEGER)
	Название страны (TEXT)
	Название столицы (TEXT)
	Численность населения (INTEGER)
	Форма правления (TEXT)
	Адм. устройство (TEXT)
	Политический режим (TEXT)
	Дата добавления (TEXT)

Листинг программы

1. Заголовочные файлы:

- mainwindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "databasemenu.h"
#include "arraymenu.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

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

private slots:
    void on_dbOpenBtn_clicked();

    void on_arrOpenBtn_clicked();

    void on_exitButton_clicked();

private:
    Ui::MainWindow *ui;
    DatabaseMenu *dbWindow;
    ArrayMenu *arrWindow;
};
#endif // MAINWINDOW_H
```

- databasemenu.h:

```
#ifndef DATABASEMENU_H
#define DATABASEMENU_H

#include <QWidget>
#include <QSqlDatabase>
#include <QSqlError>
#include <QSqlTableModel>
#include <QMessageBox>
#include "addingmenu.h"

namespace Ui {
```

```
class DatabaseMenu;
}

class DatabaseMenu : public QWidget
{
    Q_OBJECT

public:
    explicit DatabaseMenu(QWidget *parent = nullptr);
    ~DatabaseMenu();

private slots:
    void on_backBtn_clicked();

    void on_addButton_clicked();

    void on_removeButton_clicked();

    void on_databaseViewer_clicked(const QModelIndex &index);

    void on_sortBox_activated(int index);

    void on_clearButton_clicked();

public slots:
    void databaseAdderSlot(QVariantList data);

private:
    Ui::DatabaseMenu *ui;
    QSqlDatabase db;
    QSqlTableModel *model;
    AddingMenu *addWindow;
    int rowNumber;

signals:
    void firstWindow();
};


#endif // DATABASEMENU_H
```

- arraymenu.h:

```
#ifndef ARRAYMENU_H
#define ARRAYMENU_H

#include <QWidget>
#include <ctime>

namespace Ui {
class ArrayMenu;
}

class ArrayMenu : public QWidget
```

```
{
    Q_OBJECT

public:
    explicit ArrayMenu(QWidget *parent = nullptr);
    ~ArrayMenu();

private slots:
    void on_backBtn_clicked();

    void on_minSlider_valueChanged(int value);

    void on_maxSlider_valueChanged(int value);

    void on_setButton_clicked();

    void on_resultButton_clicked();

    void on_resetButton_clicked();

private:
    Ui::ArrayMenu *ui;
    float average;
    int minimum;
    int maximum;

signals:
    void firstWindow();
};

#endif // ARRAYMENU_H
```

- addingmenu.h:

```
#ifndef ADDINGMENU_H
#define ADDINGMENU_H

#include <QWidget>
#include <QMessageBox>
#include <QDateTime>

namespace Ui {
class AddingMenu;
}

class AddingMenu : public QWidget
{
    Q_OBJECT

public:
    explicit AddingMenu(QWidget *parent = nullptr);
    ~AddingMenu();

private:
```

```
    Ui::AddingMenu *ui;

signals:
    void databaseWindow();
    void databaseAdder(QVariantList);

private slots:
    void on_cancelButton_clicked();
    void on_okButton_clicked();
};

#endif // ADDINGMENU_H
```

2. Файлы исходного кода:

- main.cpp:

```
#include "mainwindow.h"

#include <QApplication>
#include <QFile>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    //добавление и применение файла со стилем проекта
    QFile styleSheetFile("./Combinear.qss");
    styleSheetFile.open(QFile::ReadOnly);
    QString styleSheet = QLatin1String(styleSheetFile.readAll());
    qApp->setStyleSheet(styleSheet);

    MainWindow w;
    w.show();

    return a.exec();
}
```

- mainwindow.cpp:

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

//конструктор виджета
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    this->setWindowTitle("Главное меню");

    //инициализация окна работы с базой данных и подключение сигнала к
слоту
    dbWindow = new DatabaseMenu();
    connect(dbWindow, &DatabaseMenu::firstWindow, this,
&MainWindow::show);

    //инициализация окна работы с массивом и подключение сигнала к
слоту
    arrWindow = new ArrayMenu();
    connect(arrWindow, &ArrayMenu::firstWindow, this,
&MainWindow::show);
}
```

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

//функция смены открытого окна программы при нажатии кнопки (открытие
окна работы с базой данных)
void MainWindow::on_dbOpenBtn_clicked()
{
    dbWindow->show();
    this->close();
}

//функция смены открытого окна программы при нажатии кнопки (открытие
окна работы с массивом)
void MainWindow::on_arrOpenBtn_clicked()
{
    arrWindow->show();
    this->close();
}

//функция кнопки завершения работы программы
void MainWindow::on_exitButton_clicked()
{
    qApp->exit();
}
```

- databasemenu.cpp:

```
#include "databasemenu.h"
#include "ui_databasemenu.h"

//конструктор виджета
DatabaseMenu::DatabaseMenu(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::DatabaseMenu)
{
    ui->setupUi(this);

    this->setWindowTitle("База данных");

    //инициализация базы данных с помощью переменной
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName("./../db/countries.db");

    //контроль ошибок при загрузке базы данных
    if (db.open())
    {
        ui->statusLabel->
            setText("База данных успешно загружена");

        //инициализация и настройка таблицы для вывода базы данных
        model = new QSqlTableModel(this, db);
        model->setTable("Страны");
        model->select();
```

```
        ui->databaseViewer->setModel(model);

        ui->databaseViewer->horizontalHeader()->
            setSectionResizeMode(QHeaderView::Stretch);

        //инициализация и обработка сигнала вызова окна добавления
элемента в базу данных
        addWindow = new AddingMenu();
        connect(addWindow, &AddingMenu::databaseWindow, this,
&DatabaseMenu::show);
        connect(addWindow, &AddingMenu::databaseAdder, this,
&DatabaseMenu::databaseAdderSlot);
    }
    else ui->statusLabel->
            setText("Ошибка загрузки базы данных: " +
db.lastError().databaseText());
}

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

//функция кнопки возврата в главное меню
void DatabaseMenu::on_backBtn_clicked()
{
    this->close();
    emit firstWindow();
}

//функция кнопки добавления элемента в базу данных
void DatabaseMenu::on_addButton_clicked()
{
    addWindow->show();
}

//слот, отрабатываемый при вызове сигнала открытия окна добавления
нового элемента в базу данных
void DatabaseMenu::databaseAdderSlot(QVariantList data)
{
    model->insertRow(model->rowCount());

    //добавление в базу данных нового элемента, информация о котором
получена в окне добавления
    for (int i = 0; i < 7; i++)
        model->setData(model->index(model->rowCount() - 1, i),
data[i]);

    model->submit();
}

//функция кнопки удаления элемента из базы данных
void DatabaseMenu::on_removeButton_clicked()
{
    model->removeRow(rowNumber);
    model->select();
}
```

```
//функция нажатия на таблицу с базой данных (выбор строки)
void DatabaseMenu::on_databaseViewer_clicked(const QModelIndex &index)
{
    rowNumber = index.row();
}

//функция нажатия на раскрывающийся список методов сортировки базы
данных
void DatabaseMenu::on_sortBox_activated(int index)
{
    //установка метода сортировки в зависимости от выбора пользователя
    switch (index)
    {
    case 0:
        model->sort(6, Qt::AscendingOrder);
        break;
    case 1:
        model->sort(6, Qt::DescendingOrder);
        break;
    case 2:
        model->sort(0, Qt::AscendingOrder);
        break;
    case 3:
        model->sort(0, Qt::DescendingOrder);
        break;
    case 4:
        model->sort(1, Qt::AscendingOrder);
        break;
    case 5:
        model->sort(1, Qt::DescendingOrder);
        break;
    case 6:
        model->sort(2, Qt::AscendingOrder);
        break;
    case 7:
        model->sort(2, Qt::DescendingOrder);
        break;
    case 8:
        model->sort(3, Qt::AscendingOrder);
        break;
    case 9:
        model->sort(3, Qt::DescendingOrder);
        break;
    case 10:
        model->sort(4, Qt::AscendingOrder);
        break;
    case 11:
        model->sort(4, Qt::DescendingOrder);
        break;
    case 12:
        model->sort(5, Qt::AscendingOrder);
        break;
    case 13:
        model->sort(5, Qt::DescendingOrder);
        break;
    }
}
```

```
//функция кнопки очистки базы данных (удаления всех полей)
void DatabaseMenu::on_clearButton_clicked()
{
    //открытие окна с предупреждением
    QMessageBox::StandardButton reply = QMessageBox::question(this,
"Подтвердите действие",
        "Внимание! Очистка базы данных приведёт к удалению всех
имеющихся полей. "
        "Отменить действие будет невозможно. Вы уверены, что желаете
продолжить?", QMessageBox::Yes | QMessageBox::No);

    //очистка базы данных
    if (reply == QMessageBox::Yes)
    {
        for (int i = 0; i < model->rowCount(); ++i)
                model->removeRow(i);
        model->select();
    }
}
```

- arraymenu.cpp

```
#include "arraymenu.h"
#include "ui_arraymenu.h"

//конструктор виджета
ArrayMenu::ArrayMenu(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::ArrayMenu)
{
    ui->setupUi(this);

    this->setWindowTitle("Динамический массив");

    //установка диапазона ползунков
    ui->minSlider->setRange(-1000, 0);
    ui->maxSlider->setRange(1, 1000);

    ui->maxRB->setChecked(true);
}

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

//функция нажатия на кнопку возврата в главное меню
void ArrayMenu::on_backBtn_clicked()
{
    this->close();
    emit firstWindow();
}

//функция перемещения ползунка с минимальным значением диапазона
```

```
void ArrayMenu::on_minSlider_valueChanged(int value)
{
    ui->minLcd->display(value);
}

//функция перемещения ползунка с максимальным значением диапазона
void ArrayMenu::on_maxSlider_valueChanged(int value)
{
    ui->maxLcd->display(value);
}

//функция кнопки "Задать"
void ArrayMenu::on_setButton_clicked()
{
    ui->startBrowser->clear();  //предварительная очистка окна вывода
массива

    //инициализация переменной для хранения массива и начальные
значения переменных для подсчета результата работы над массивом
    int *data = new int[10];
    this->average = 0;
    this->minimum = 1001;
    this->maximum = -1001;

    srand(time(NULL));

    //заполнение массива случайными числами, запись массива в окно для
вывода и подсчёт результата каждого действия над массивом
    for (int i = 0; i < 10; i++)
    {
        QString value;

        data[i] = rand() % (ui->maxLcd->intValue() - ui->minLcd-
>intValue() + 1) + ui->minLcd->intValue();
        value.setNum(data[i]);
        ui->startBrowser->append(value);

        average += data[i];
        if (data[i] < minimum) minimum = data[i];
        if (data[i] > maximum) maximum = data[i];
    }

    average = average / 10;
}

//функция вывода в окно результата обработки массива
void ArrayMenu::on_resultButton_clicked()
{
    ui->resultBrowser->clear();

    //вывод наименьшего значения массива
    if (ui->minRB->isChecked())
    {
        QString result;
        result.setNum(minimum);
        ui->resultBrowser->append(result);
    }
```

```
    //вывод наибольшего значения массива
    if (ui->maxRB->isChecked())
    {
        QString result;
        result.setNum(maximum);
        ui->resultBrowser->append(result);
    }

    //вывод среднего арифметического всех значений массива
    if (ui->averageRB->isChecked())
    {
        QString result;
        result.setNum(average);
        ui->resultBrowser->append(result);
    }
}

//функция кнопки сброса (очистка всех окон для вывода, установка
ползунков в начальное положение)
void ArrayMenu::on_resetButton_clicked()
{
    ui->startBrowser->clear();
    ui->resultBrowser->clear();
    ui->maxLcd->display(1);
    ui->minLcd->display(0);
    ui->minSlider->setSliderPosition(0);
    ui->maxSlider->setSliderPosition(1);
}
```

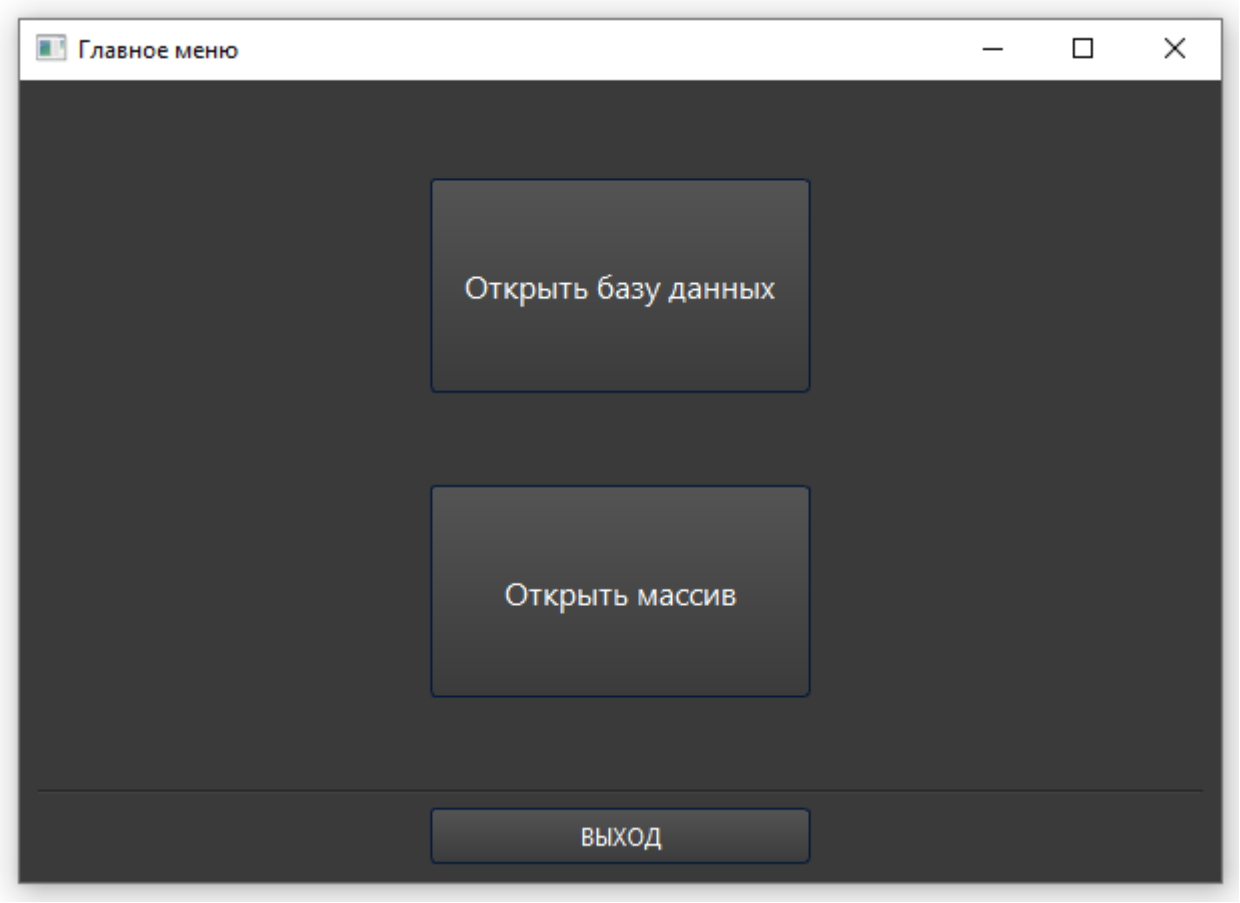
- addingmenu.cpp:

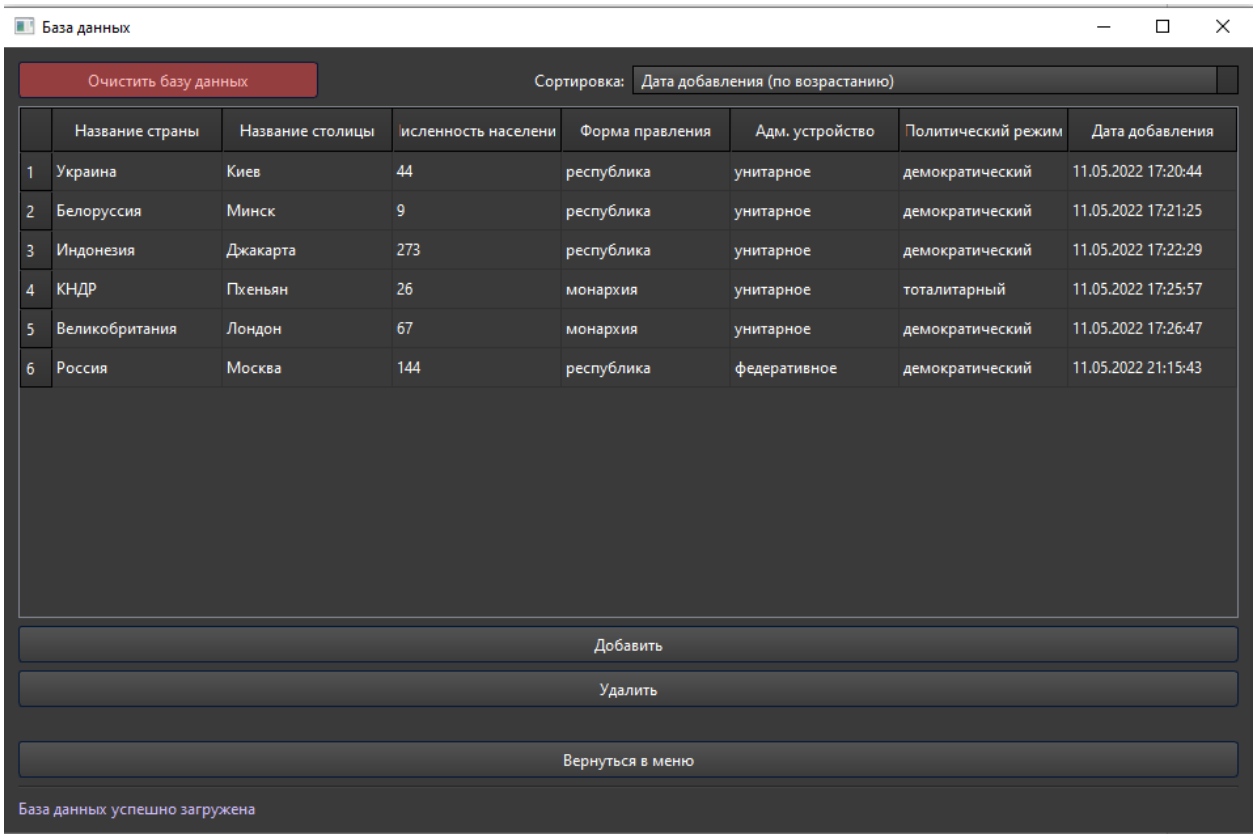
```
#include "addingmenu.h"
#include "ui_addingmenu.h"
#include <QMessageBox>

//конструктор виджета
AddingMenu::AddingMenu(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::AddingMenu)
{
    ui->setupUi(this);

    this->setWindowTitle("Добавление элемента...");  //установка
наименования окна

    ui->formRadio_1->setChecked(true);
    ui->regimeRadio_1->setChecked(true);
    ui->admRadio_1->setChecked(true);
}

AddingMenu::~AddingMenu()
{
    delete ui;
}
```

```
//функция нажатия на кнопку отмены
void AddingMenu::on_cancelButton_clicked()
{
    ui->lineEdit->clear();
    ui->lineEdit_2->clear();
    ui->lineEdit_3->clear();

    ui->formRadio_1->setChecked(true);
    ui->regimeRadio_1->setChecked(true);
    ui->admRadio_1->setChecked(true);


    this->close();
    emit databaseWindow();
}

//функция нажатия на кнопку "Ок"
void AddingMenu::on_okButton_clicked()
{
    //контроль входных данных
    QString warning_1 = "", warning_2 = "", warning_3 = "";

    if (ui->lineEdit->text() == "") warning_1 = "*Поле <Название
страны> не должно быть пустым. ";
    if (ui->lineEdit_2->text() == "") warning_2 = "*Поле <Название
столицы> не должно быть пустым. ";
    if ((ui->lineEdit_3->text().toInt() == false) || (ui->lineEdit_3-
>text().toInt() <= 0))
        warning_3 = "*Поле <Численность населения> должно содержать
положительное целочисленное значение. ";

    //заполнение переменной data для последующего добавления объекта в
БД при отсутствии ошибок ввода
    if ((warning_1 == warning_2) && (warning_2 == warning_3))
    {
        QVariantList data;

        data.append(ui->lineEdit->text());
        data.append(ui->lineEdit_2->text());
        data.append(ui->lineEdit_3->text());

        if (ui->formRadio_1->isChecked()) data.append("республика");
        if (ui->formRadio_2->isChecked()) data.append("монархия");

        if (ui->admRadio_1->isChecked()) data.append("унитарное");
        if (ui->admRadio_2->isChecked()) data.append("федеративное");
        if (ui->admRadio_3->isChecked())
data.append("конфедеративное");

        if (ui->regimeRadio_1->isChecked())
data.append("демократический");
        if (ui->regimeRadio_2->isChecked())
data.append("авторитарный");
        if (ui->regimeRadio_3->isChecked())
data.append("тоталитарный");

        data.append(QDateTime::currentDateTime().toString("dd.MM.yyyy
HH:mm:ss"));
```

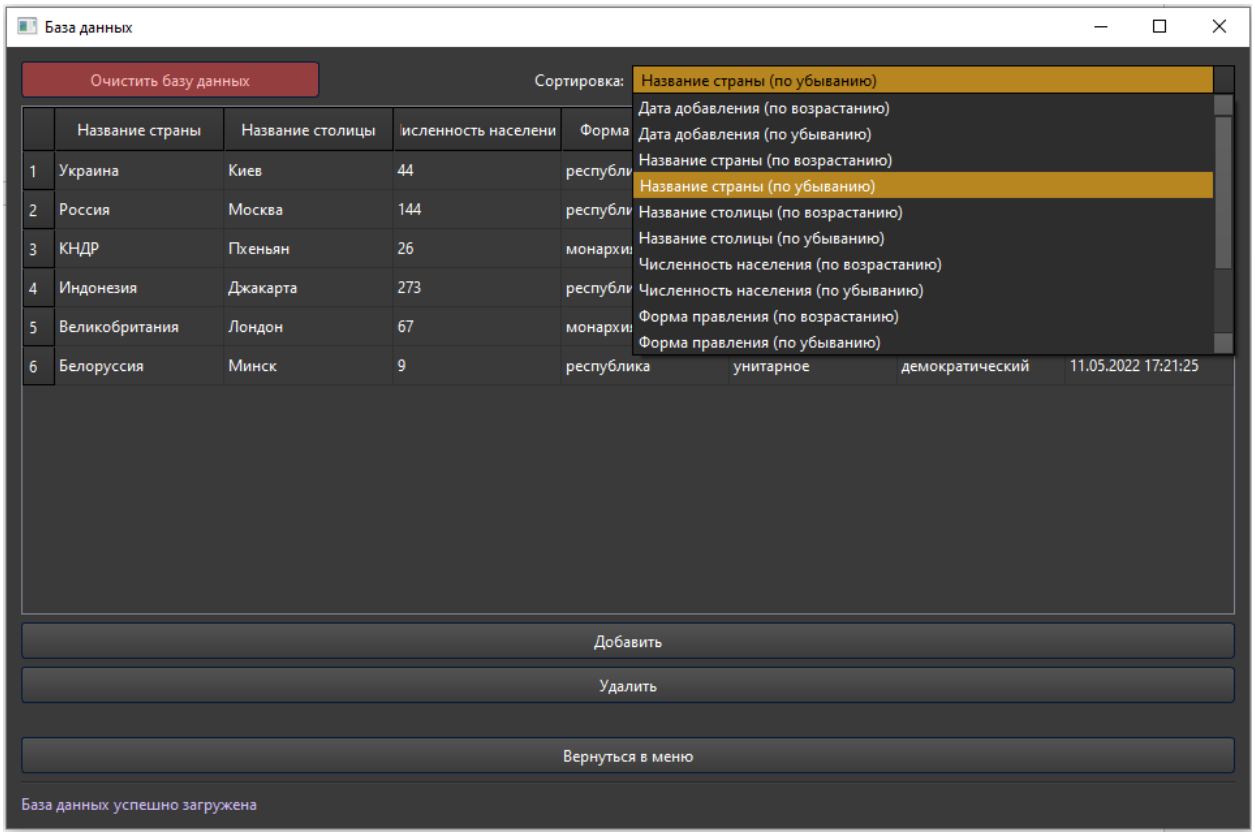
```
        emit databaseAdder(data);

        //очистка окна после ввода данных
        ui->lineEdit->clear();
        ui->lineEdit_2->clear();
        ui->lineEdit_3->clear();

        ui->formRadio_1->setChecked(true);
        ui->regimeRadio_1->setChecked(true);
        ui->admRadio_1->setChecked(true);


        this->close();
        emit databaseWindow();
    }

    //вывод сообщения об ошибках ввода
    else QMessageBox::warning(this, "Ошибка ввода", "При вводе
информации об объекте допущены следующие ошибки: "
                                + warning_1 + warning_2 + warning_3 +
"Пожалуйста, исправьте неточности и повторите ввод.");
}
```

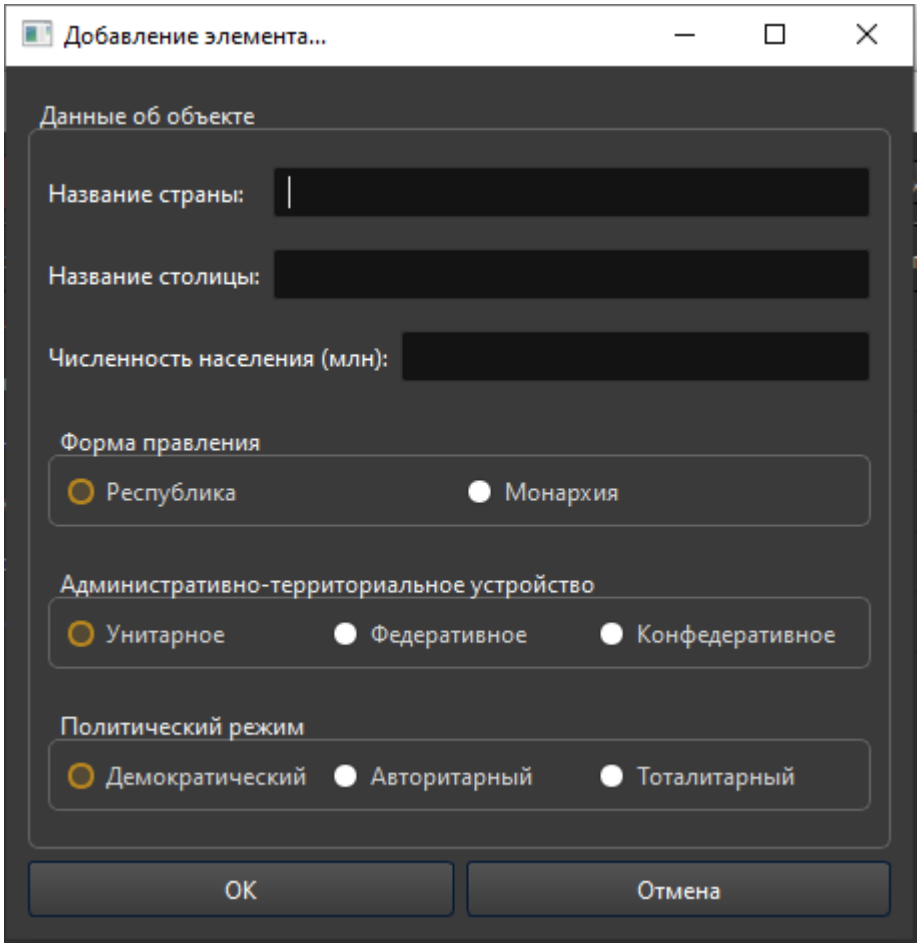
Тестирование

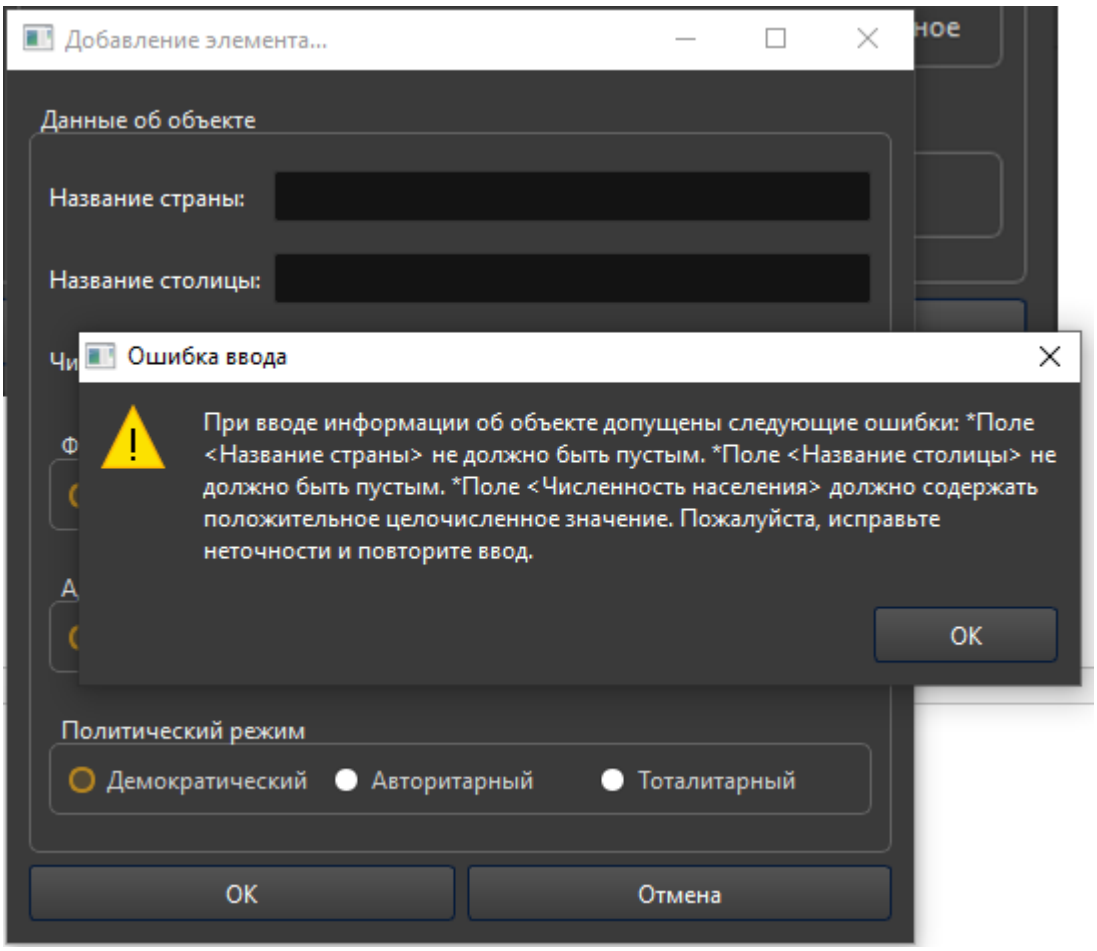


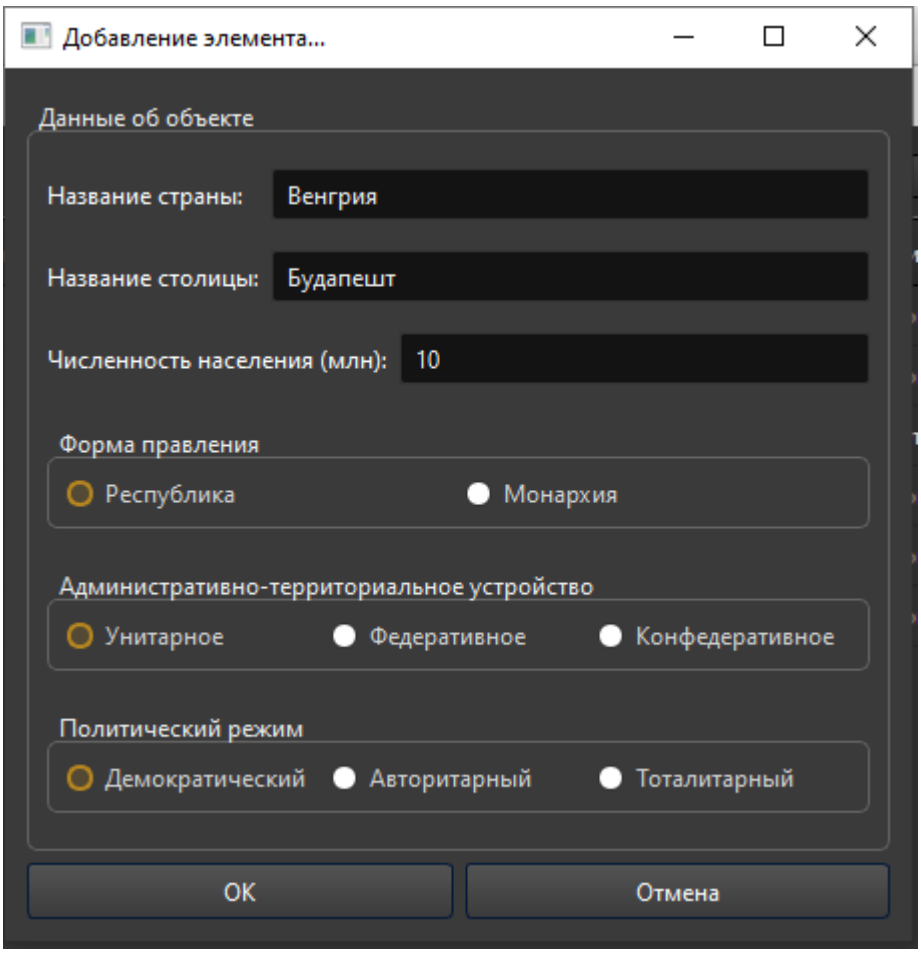


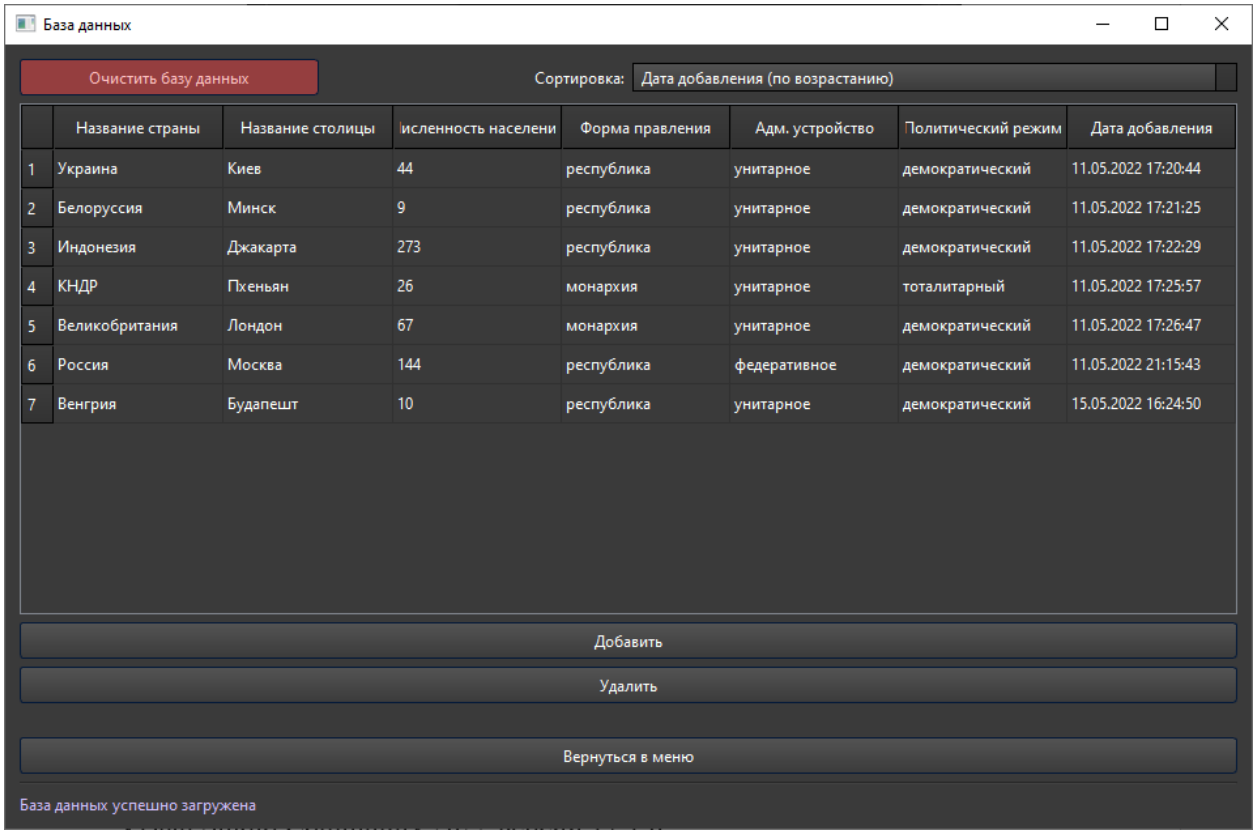
	Название страны	Название столицы	Численность населени	Форма правления	Адм. устройство	Политический режим	Дата добавления
1	Украина	Киев	44	республика	унитарное	демократический	11.05.2022 17:20:44
2	Белоруссия	Минск	9	республика	унитарное	демократический	11.05.2022 17:21:25
3	Индонезия	Джакарта	273	республика	унитарное	демократический	11.05.2022 17:22:29
4	КНДР	Пхеньян	26	монархия	унитарное	тоталитарный	11.05.2022 17:25:57
5	Великобритания	Лондон	67	монархия	унитарное	демократический	11.05.2022 17:26:47
6	Россия	Москва	144	республика	федеративное	демократический	11.05.2022 21:15:43


База данных
Очистить базу данных
Сортировка:
Название страны (по убыванию)
Дата добавления (по возрастанию)
Дата добавления (по убыванию)
Название страны (по возрастанию)
Название страны (по убыванию)
Название столицы (по возрастанию)
Название столицы (по убыванию)
Численность населения (по возрастанию)
Численность населения (по убыванию)
Форма правления (по возрастанию)
Форма правления (по убыванию)
Название страны
Название столицы
исленность населени
Форма
1 Украина Киев 44 республи
2 Россия Москва 144 республи
3 КНДР Пхеньян 26 монархи
4 Индонезия Джакарта 273 республи
5 Великобритания Лондон 67 монархи
6 Белоруссия Минск 9 республика унитарное демократический 11.05.2022 17:21:25
Добавить
Удалить
Вернуться в меню
База данных успешно загружена

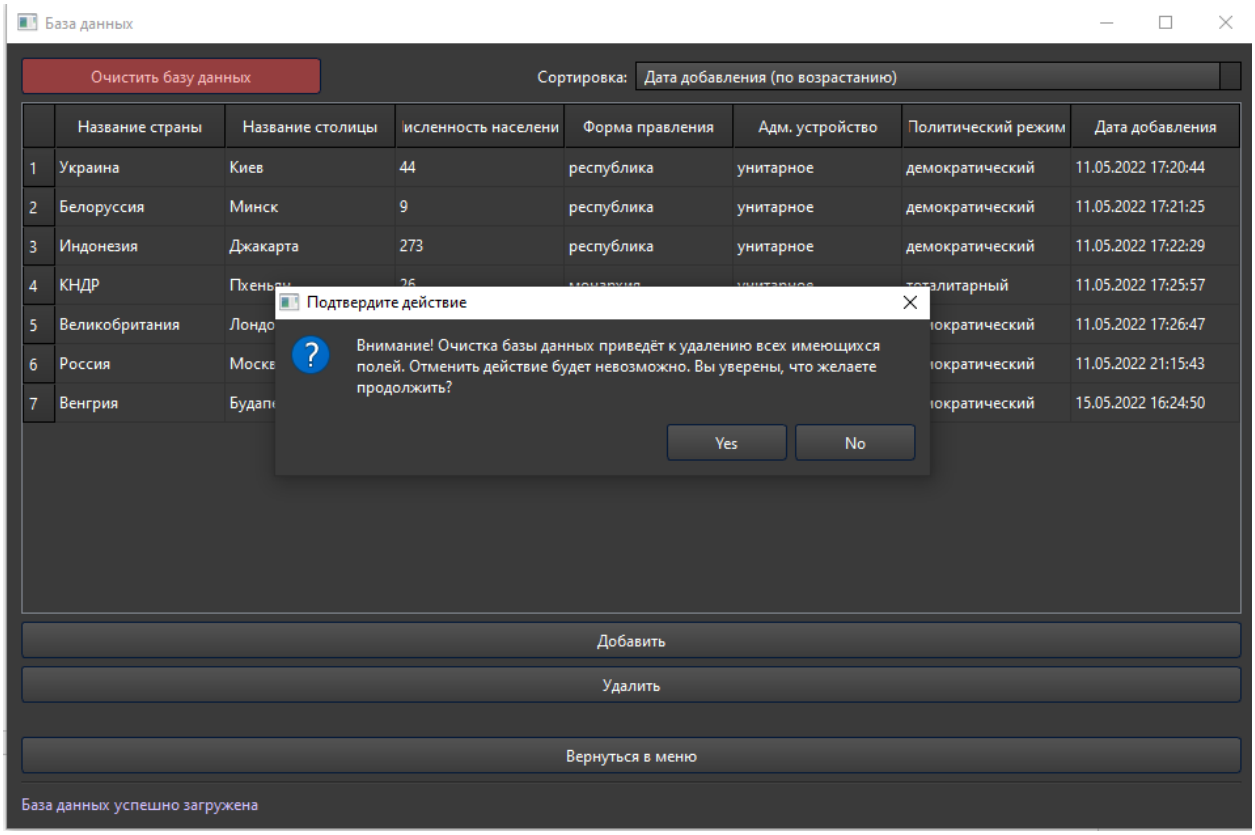

Добавление элемента...
Данные об объекте
Название страны:
Название столицы:
Численность населения (млн):
Форма правления
Республика
Монархия
Административно-территориальное устройство
Унитарное
Федеративное
Конфедеративное
Политический режим
Демократический
Авторитарный
Тоталитарный
OK
Отмена

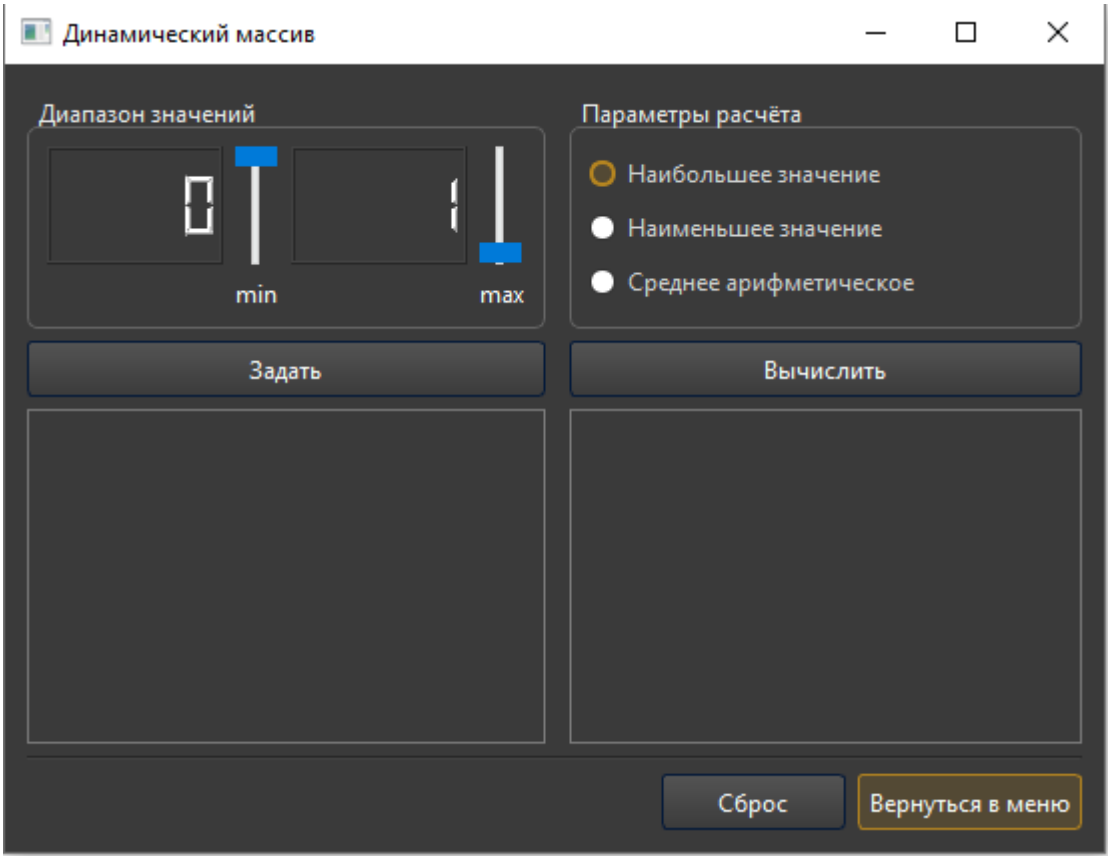

Добавление элемента...
Данные об объекте
Название страны:
Название столицы:
Ошибка ввода
При вводе информации об объекте допущены следующие ошибки: *Поле <Название страны> не должно быть пустым. *Поле <Название столицы> не должно быть пустым. *Поле <Численность населения> должно содержать положительное целочисленное значение. Пожалуйста, исправьте неточности и повторите ввод.
OK
Политический режим
Демократический
Авторитарный
Тоталитарный
OK
Отмена

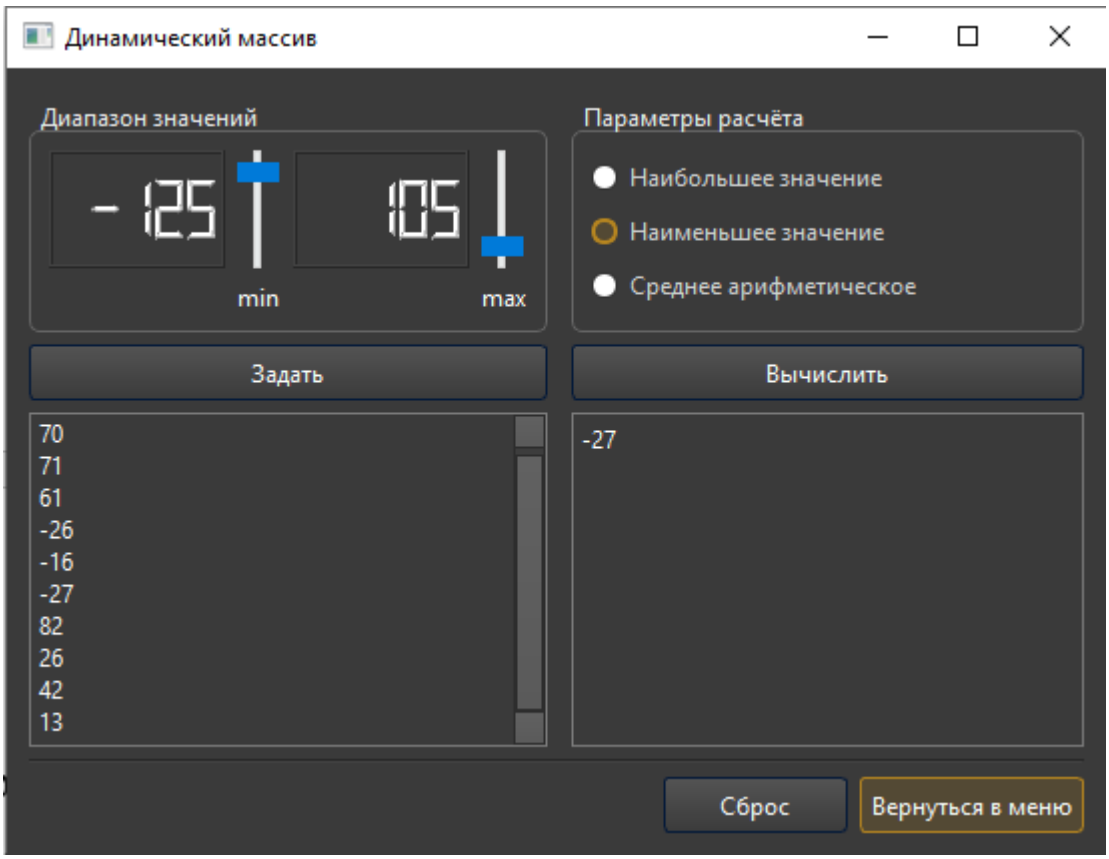




	Название страны	Название столицы	Численность населени	Форма правления	Адм. устройство	Политический режим	Дата добавления
1	Украина	Киев	44	республика	унитарное	демократический	11.05.2022 17:20:44
2	Белоруссия	Минск	9	республика	унитарное	демократический	11.05.2022 17:21:25
3	Индонезия	Джакарта	273	республика	унитарное	демократический	11.05.2022 17:22:29
4	КНДР	Пхеньян	26	монархия	унитарное	тоталитарный	11.05.2022 17:25:57
5	Великобритания	Лондон	67	монархия	унитарное	демократический	11.05.2022 17:26:47
6	Россия	Москва	144	республика	федеративное	демократический	11.05.2022 21:15:43
7	Венгрия	Будапешт	10	республика	унитарное	демократический	15.05.2022 16:24:50


База данных
Очистить базу данных
Сортировка:
Дата добавления (по возрастанию)
Название страны
Название столицы
исленность населени
Форма правления
Адм. устройство
Политический режим
Дата добавления
1 Украина Киев 44 республика унитарное демократический 11.05.2022 17:20:44
2 Белоруссия Минск 9 республика унитарное демократический 11.05.2022 17:21:25
3 Индонезия Джакарта 273 республика унитарное демократический 11.05.2022 17:22:29
4 КНДР Пхеньян тоталитарный 11.05.2022 17:25:57
5 Великобритания Лондо кратический 11.05.2022 17:26:47
6 Россия Москв кратический 11.05.2022 21:15:43
7 Венгрия Будап кратический 15.05.2022 16:24:50
Подтвердите действие
Внимание! Очистка базы данных приведёт к удалению всех имеющихся полей. Отменить действие будет невозможно. Вы уверены, что желаете продолжить?
Yes
No
Добавить
Удалить
Вернуться в меню
База данных успешно загружена


Динамический массив
Диапазон значений
0
1
min
max
Параметры расчёта
Наибольшее значение
Наименьшее значение
Среднее арифметическое
Задать
Вычислить
Сброс
Вернуться в меню


Динамический массив
Диапазон значений
-125
105
min
max
Параметры расчёта
Наибольшее значение
Наименьшее значение
Среднее арифметическое
Задать
Вычислить
70
71
61
-26
-16
-27
82
26
42
13
-27
Сброс
Вернуться в меню

Вывод

При получении задания по лабораторной работе №5 была поставлена задача написать на языке C++ программу с графическим интерфейсом, содержащую базу данных SQLite и динамический массив. Поставленная задача была успешно выполнена в среде разработки QT Creator (Community) версии 7.0.0.

Выполнение лабораторной работы помогло ознакомиться с процессом разработки графических приложений, а также с принципами работы с базами данных.

Весь функционал протестирован при различных сценариях, программа работает корректно.