ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "САНКТ-ПЕТЕРБУРГСКИЙ

ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М. А.

БОНЧ-БРУЕВИЧА"

Факультет инфокоммуникационных сетей и систем

Кафедра программной инженерии и вычислительной техники

**КУРСОВАЯ РАБОТА**

**по дисциплине «Объектно-ориентированное программирование»**

**ВАРИАНТ 22**

Выполнил:

студент 2 курса дневного отделения группы ИКПИ-35

Рыбин И. П.

Санкт-Петербург 2024

**Оглавление**

[1. Постановка задачи 3](#_Toc183261285)

[2. Выбор и обоснование типа разрабатываемого контейнера 4](#_Toc183261286)

[3. Разработка структуры программы 4](#_Toc183261287)

[4. Разработка собственных классов 5](#_Toc183261288)

[5. Разработка интерфейса 9](#_Toc183261289)

[6. Инструкция пользователю 10](#_Toc183261290)

[7. Код программы 19](#_Toc183261291)

[8. Заключение 63](#_Toc183261292)

[9. Список литературы 64](#_Toc183261293)

# Постановка задачи

**Необходимо разработать приложение в QT Creator, которое позволит работать с базами данных на основе собственно-реализуемого контейнера — vector или list.** Для работы контейнера следует предусмотреть итератор, который позволит проходить по элементам следующим образом:

container<type> temp; for (auto k : temp) {

// k -- очередной элемент контейнера temp

}

**В качестве элементов базы данных по варианту 10 должны использоваться объекты функций языка (-ов) программирования.**

**База данных должна обеспечивать выполнение следующих операций:**

— Создание базы данных,

— Объединение баз данных,

— Добавления и удаления записей,

— Редактирования записей,

— Просмотра базы данных,

— Поиск данных, — Запись на диск,

— Чтение с диска.

**Интерфейсная часть программы должна содержать следующие компоненты:** — Окно «О программе»,

— Главное меню,

— Всплывающее меню,

— Строку состояния,

— Панель инструментов,

— Подсказки по командам меню и панели инструментов.

# Выбор и обоснование типа разрабатываемого контейнера

**В курсовой работе необходимо использовать контейнер для хранения объектов.**

**В качестве прототипа разрабатываемого контейнера выбран vector** (динамический массив), потому что он обладает меньшей сложностью при доступе к элементам в отличии от контейнера типа list.

**Разрабатываемый контейнер должен обладать следующими функциями**:

— Создание пустого контейнера;

— Создание контейнера размера n;

— Конструкторы и операторы копирования и перемещения;

— Получение элемента по индексу;

— Вставка элемента в произвольную позицию;

— Очистка контейнера;

— Удаление элемента по индексу;

— Наличие класса итератора и итераторов begin и end.

**Последняя возможность позволяет реализовать следующий проход по элементам** (в цикле foreach):Vector<type> temp; for (auto k : temp) {

// k -- очередной элемент вектора temp

}

# Разработка структуры программы

**Программа состоит из интерфейса и кода.**

**Интерфейс:**

* **Основное окно программы**.
  + Главное меню QMenuBar.
  + Панель инструментов QToolBar.

o Таблица QTableWidget.

o Строка поиска по таблице QLineEdit.

o Статусная строка QStatusBar.

* **Диалог добавления / редактирования элемента**.
  + 4 метки QLabel: название, адрес, дата открытия и художественное руководство к театру
  + . o 4 текстовые поля QLineEdit.
* **Диалог просмотра базы данных в CSV-формате**.
  + Многострочное текстовое поле QTextEdit.

o Кнопка закрытия окна QPushButton.

**Код:**

* Основной класс формы QT — окно главной программы.
* Класс формы QT — диалог добавления / редактирования записи.
* Класс формы QT — диалог просмотра базы данных в CSV-формате.
* Шаблонный класс контейнера типа vector.
* Класс функции языка программирования.
* Класс всех текстовых сообщений, которые будут видны в приложении.

# Разработка собственных классов

**Theater** — класс функции языка программирования.

* **Поля класса**:
  + Название: QString name.
  + Адрес: QString address.
  + Дата открытия: QString openingDate.
  + Художественное руководство: QString artisticDirection.
* **Методы класса**:
  + Изменение названия театра: setName.
  + Получение названия театра: getName.
  + Изменение адреса театра: setAddress.
  + Получение адреса театра: getAddress.
  + Изменение даты открытия театра: setOpeningDate.
  + Получение даты открытия театра: getOpeningDate.
  + Изменение художественного руководства театра: setArtisticDirection.
  + Получение художественного руководства театра: getArtisticDirection.

**Vector** — класс контейнера выбранного типа.

* **Поля класса**:
  + Текущий размер массива: int n\_.
  + Резервируемый размер массива: int m\_.
  + Указатель на массив с элементами: T \* value.
  + Стандартное число резервируемых элементов: int standart.
* **Методы класса**:
  + Оператор получения элемента по индексу.
  + Получение размера массива: size.
  + Вставка элемента в произвольную позицию: insert. o Добавление элемента в начало: push\_front. o Добавление элемента в конец: push\_back.
  + Очистка массива: clear. o Получение булевого значения "пустой ли массив". o Удаление элемента произвольного индекса: erase. o Удаление первого элемента: pop\_front.
  + Удаление последнего элемента: pop\_back.

**MainWindow** — класс главного окна.

* **Поля класса**:
  + Строка, содержащая текущее имя файла: QString file\_.
  + Контейнер с элементами: Vector<Function> main\_vector.
* **Методы класса**:
  + Выбор действия "Новый файл": on\_action\_New\_triggered.
  + Выбор действия "Открыть": on\_action\_Open\_triggered.
  + Выбор действия "Сохранить": on\_action\_Save\_triggered.
  + Выбор действия "Сохранить как": on\_action\_SaveAs\_triggered.
  + Выбор действия "Объединить": on\_action\_Merge\_triggered.
  + Выбор действия "Добавить запись": on\_action\_Add\_triggered.
  + Выбор действия "Редактировать запись": on\_action\_Edit\_triggered.
  + Выбор действия "Удалить запись": on\_action\_Delete\_triggered.
  + Выбор действия "Очистить": on\_action\_Clear\_triggered.
  + Выбор действия "Отобразить данные": on\_action\_ShowData\_triggered.
  + Выбор действия "О программе": on\_action\_About\_triggered.
  + Выбор действия "Выйти": on\_action\_Exit\_triggered.
  + Двойной клик по таблице: on\_tableWidget\_cellDoubleClicked.
  + Поиск в LineEdit: on\_lineEdit\_textChanged.
  + Вспомогательный метод для создания нового файла: createNewDatabase

**RedactorRecordWindow**— класс диалогового окна для добавления / редактирования записи.

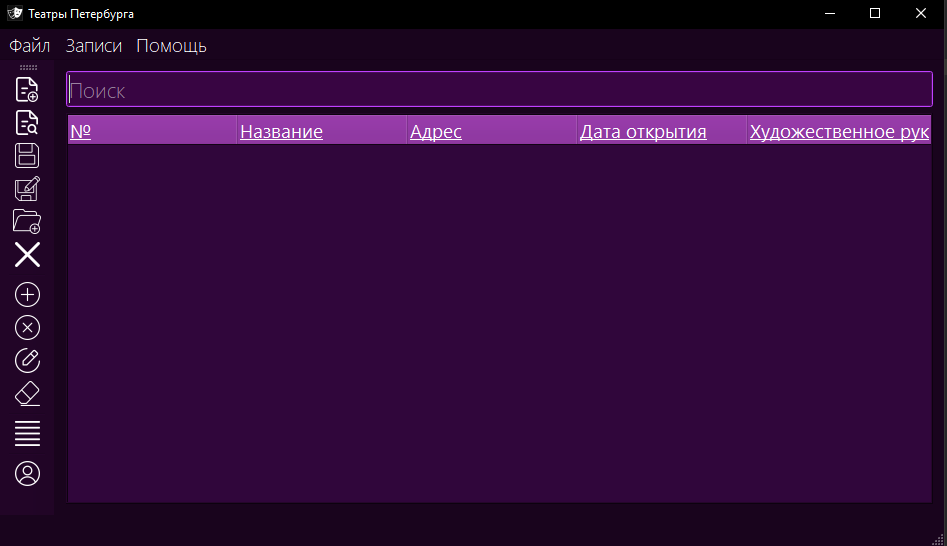
* **Методы класса**:
  + - Установка названия театра: setName.
    - Установка адреса театр: setGroup.
    - Установка даты открытия театра: setCourse.
    - Установка художественного руководства театра: setFaculty.
    - Получение данных в результате диалога: getData.

**ShowAllRecordsCode** — класс диалогового окна для отображения данных в CSV-формате.

* **Методы класса**:
  + - * Установка текста в многострочное поле: setText.
      * Получение текста из многострочного поля: getText.
      * Нажатие кнопки закрытия: on\_pushButton\_clicked.

# Разработка интерфейса

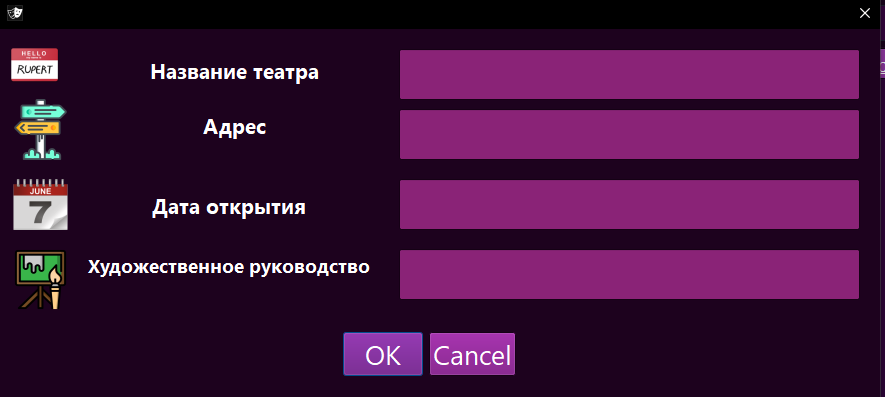
1. **Основное окно программы** (размер окна меняется).



Главное меню, панель инструментов, таблица, строка поиска и статусная строка

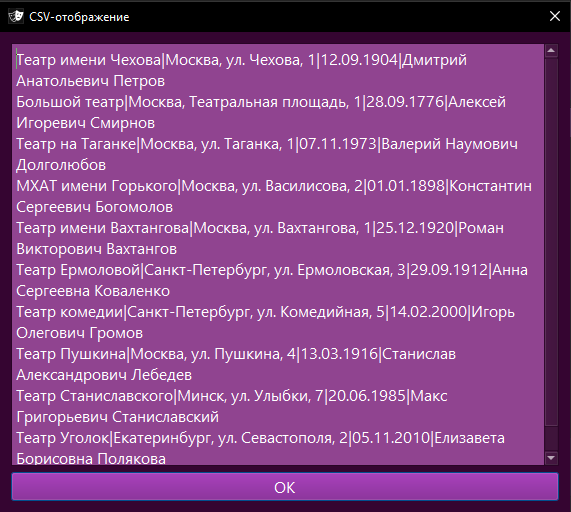
(пустая)

1. **Диалог добавления / редактирования записи** (размер окна не меняется).



После нажатия OK все пробелы в текстовых строках удаляются слева и справа и нормализуются в середине до одного. Таким образом, все элементы базы данных приводятся к одному виду.

1. **Диалог отображения данных в CSV-формате** (размер окна меняется).

****

Текст изменить нельзя. Только для чтения.

После нажатия OK окно закрывается.

# Инструкция пользователю

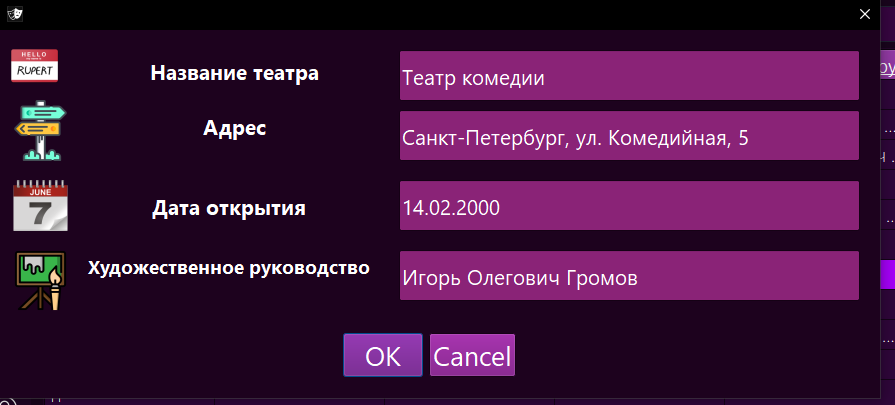


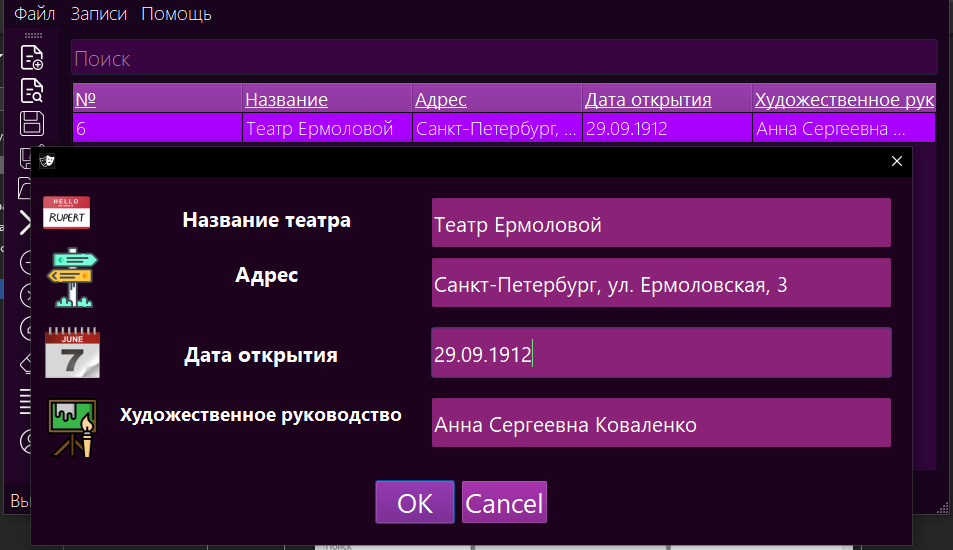
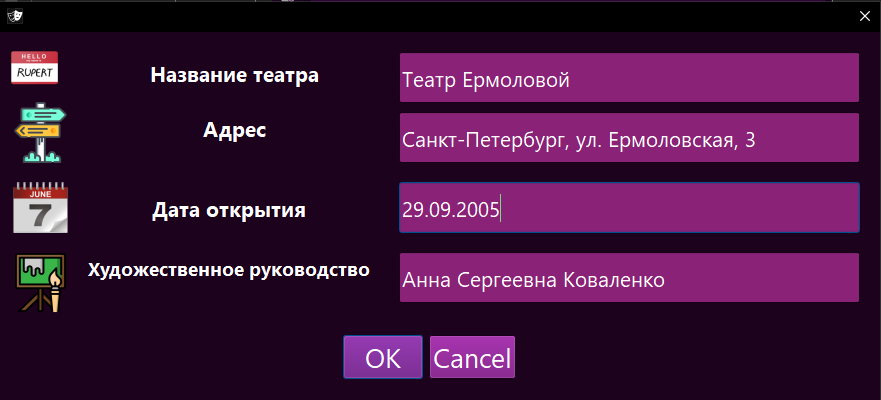
**1**

**.**

**Добавление записи**

**.**

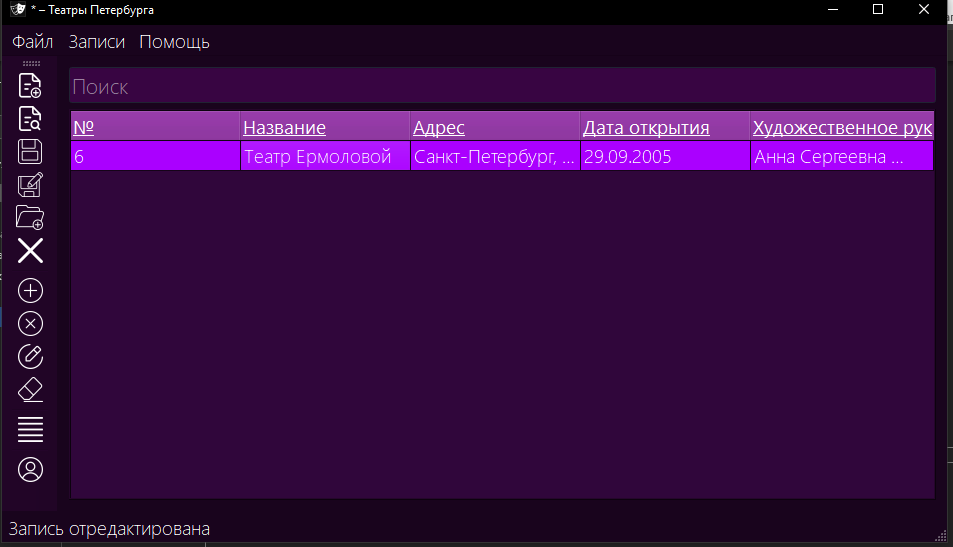
 Диалог добавления записи и отображение результата в главном окне в таблице и в статусной строке



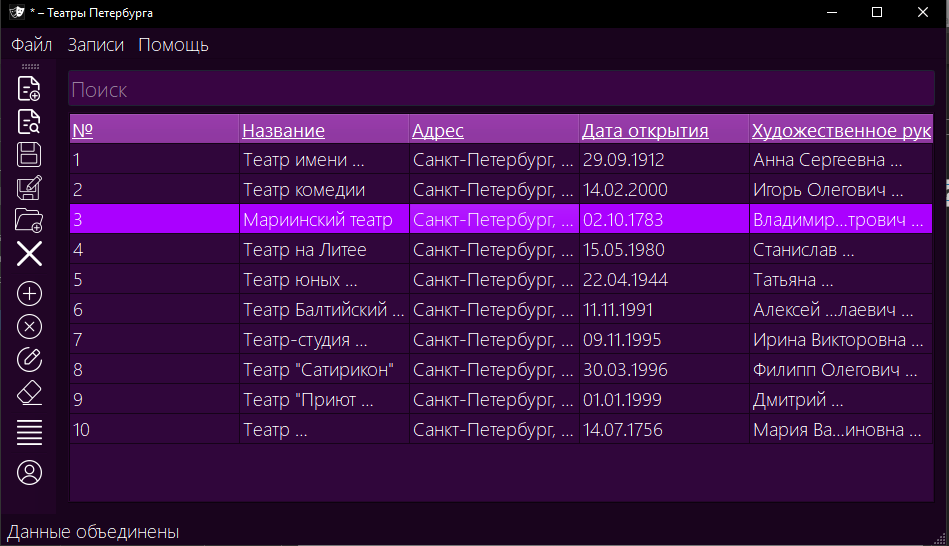
**2**

**. Редактирование записи**

**.**

Диалог редактирования записи и отображение результата в главном окне в таблице и в статусной строке.

Также можно дважды кликнуть по любой строке, чтобы её отредактировать.



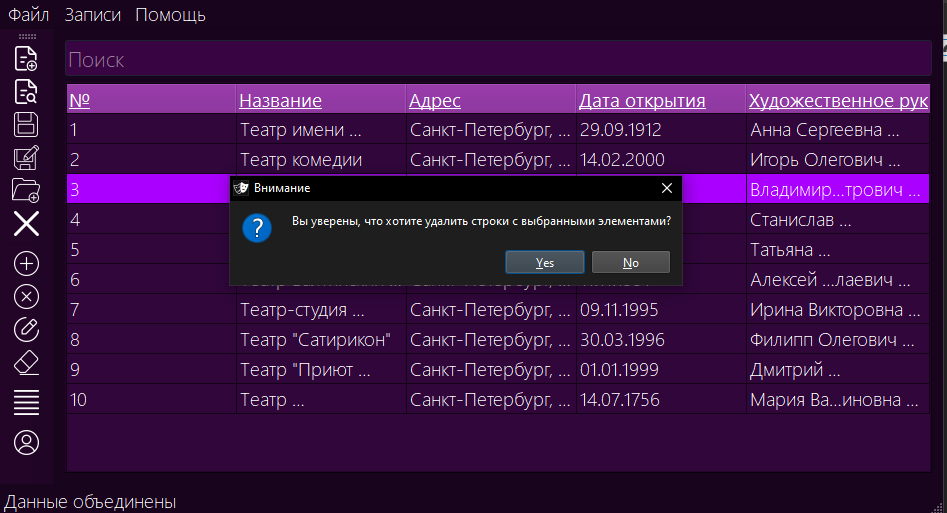
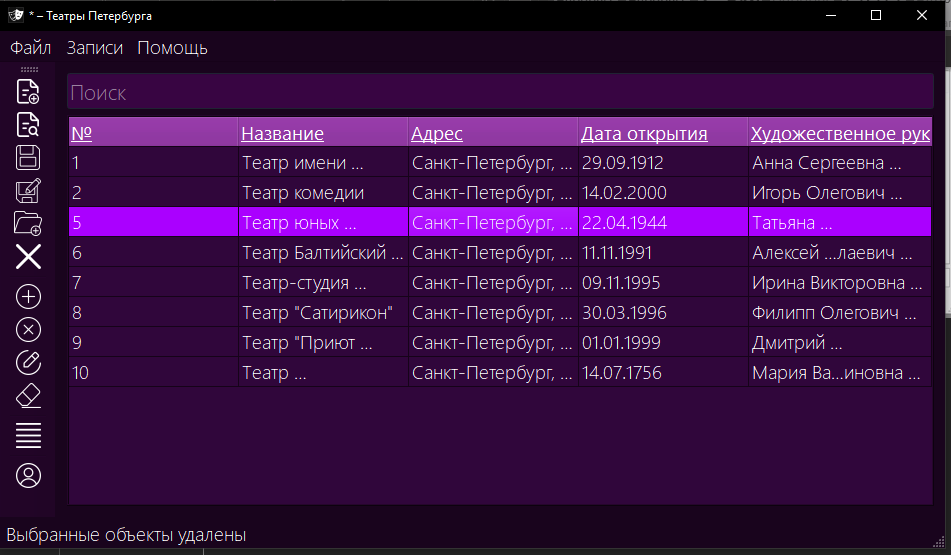
**3**

**.**

**Удаление записи**

**.**

Удаление записи (записей) с подтверждением.

Для того, чтобы выделить несколько, необходимо зажать Ctrl при выборе.

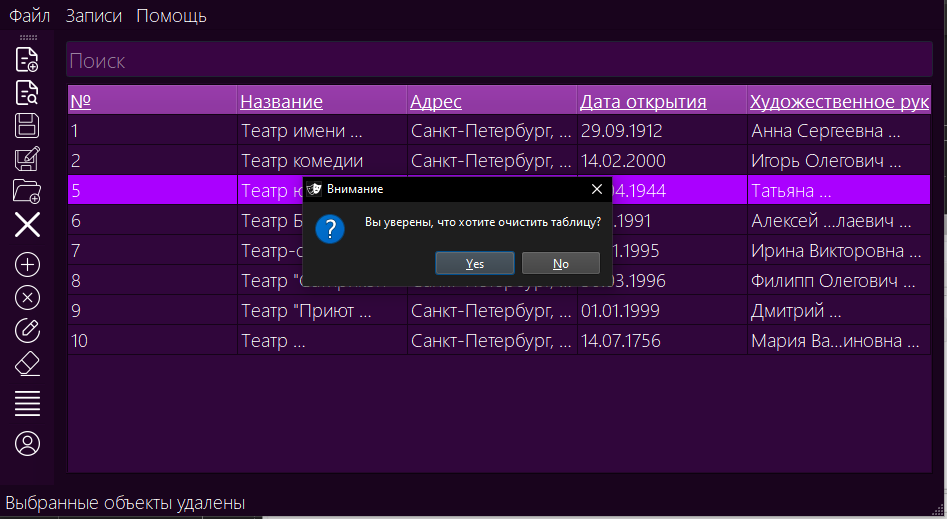
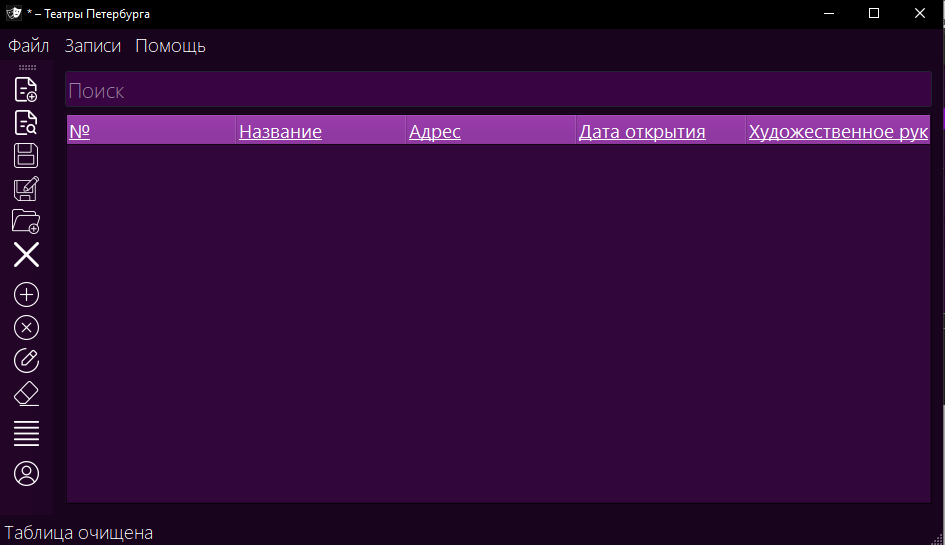


**3**

**.**

**Удаление записи**

**.**

Очистка записей с подтверждением.

Подтверждение не отображается, если таблица пуста.

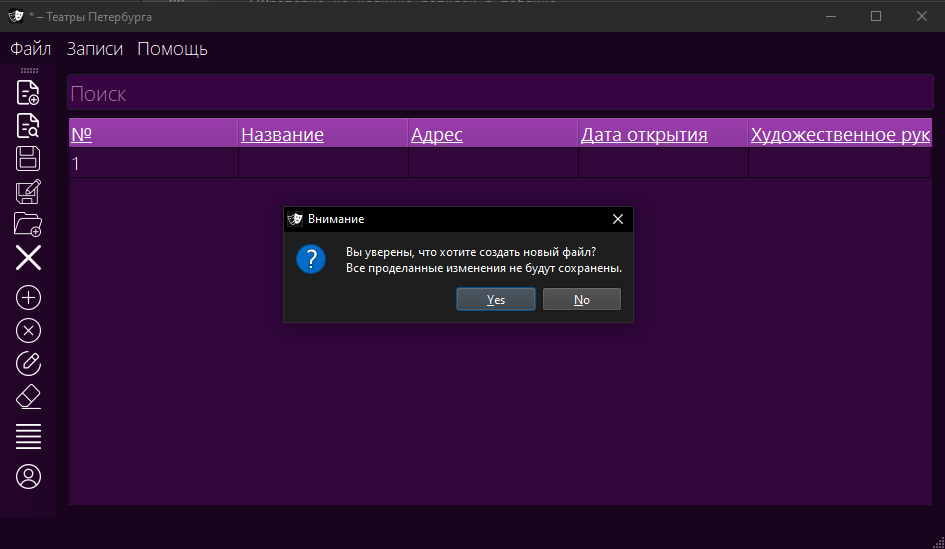
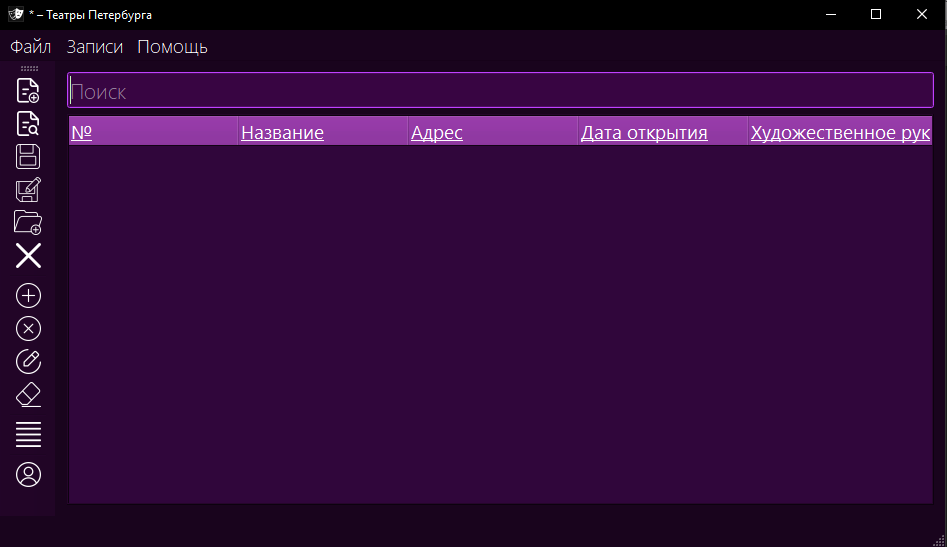


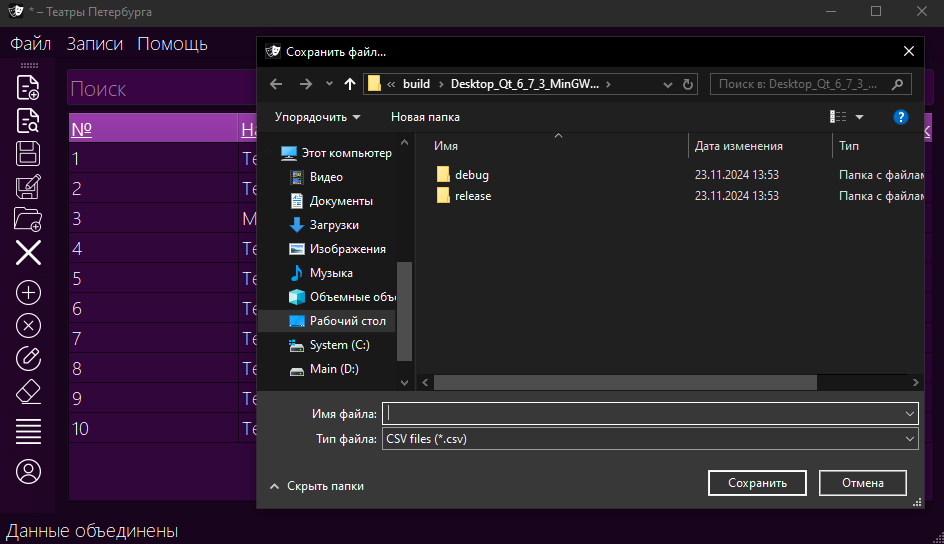
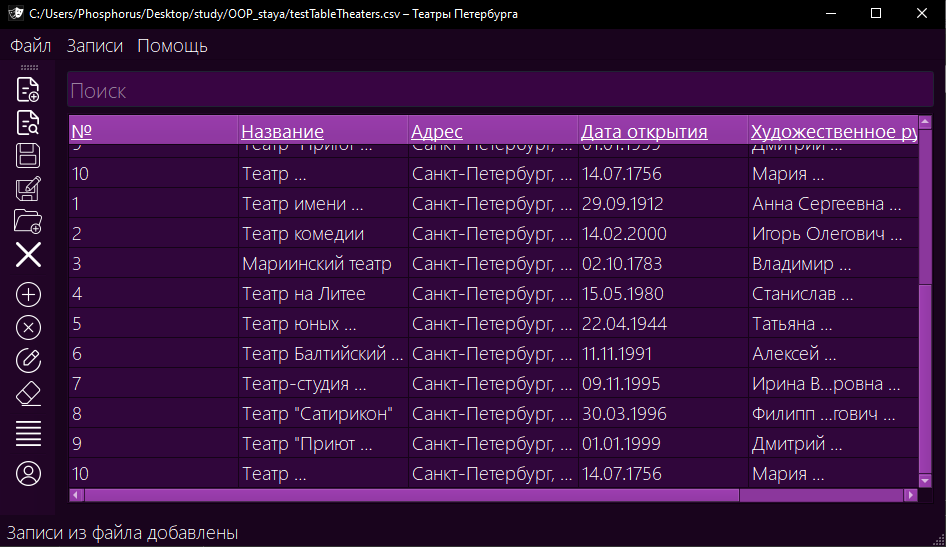
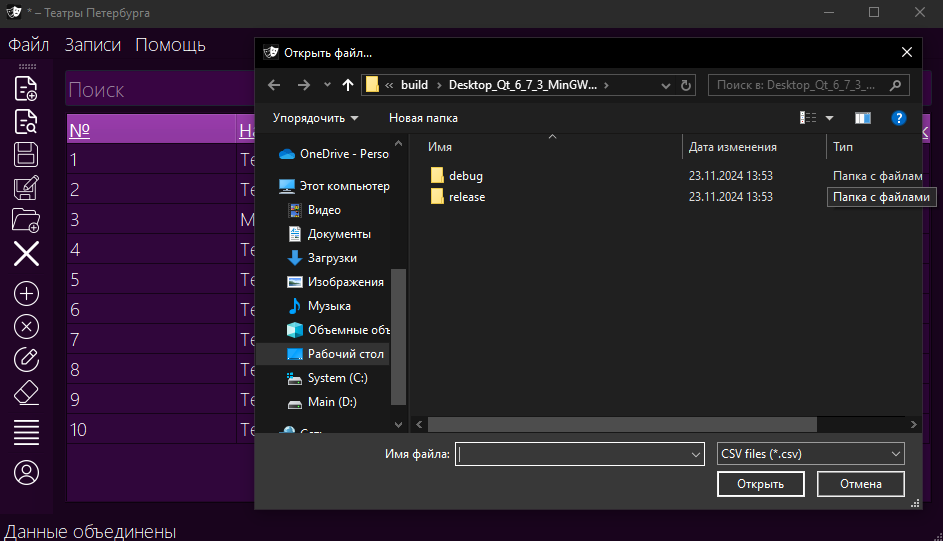
**5**

**. Создание новой базы данных**

**.**

Создание нового файла с подтверждением.

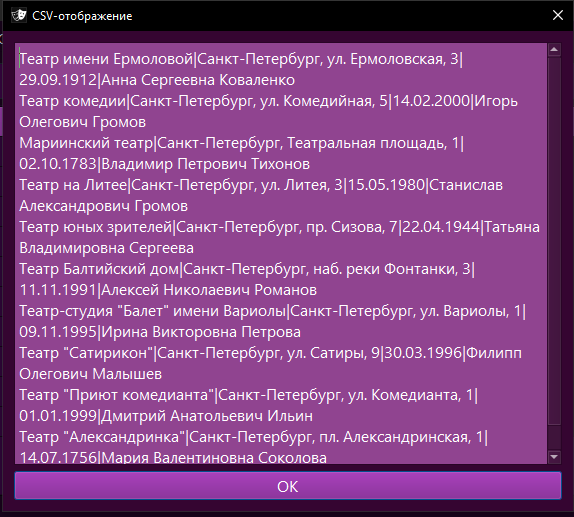
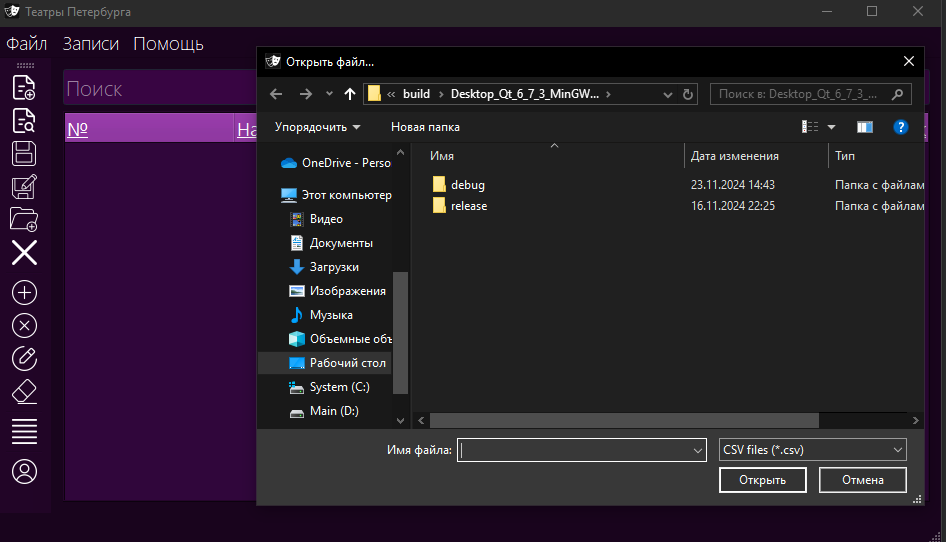
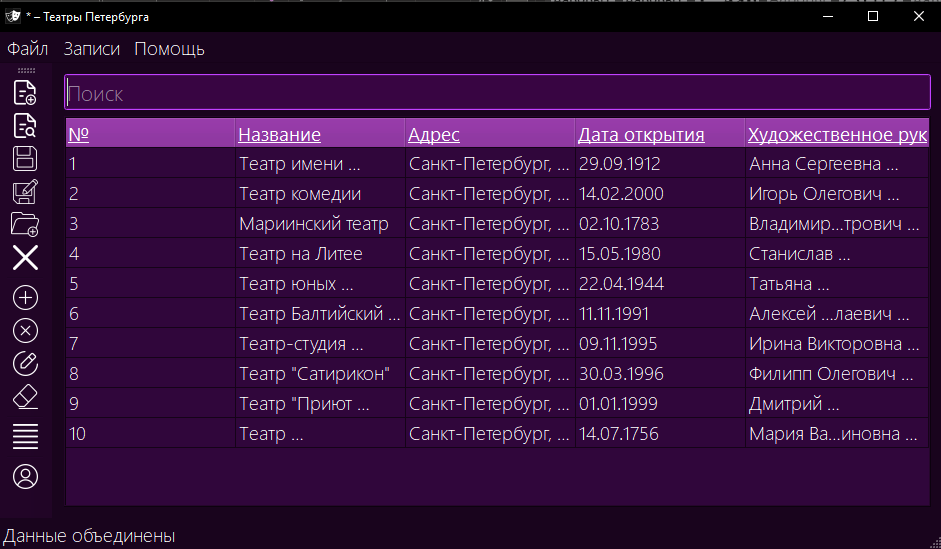
Подтверждение не отображается, если база данных пуста и никакой файл не открыт.



**6**

**. Сохранение и открытие базы данных**

**.**



**7**

**. Объединение баз данных**

**.**

**8**

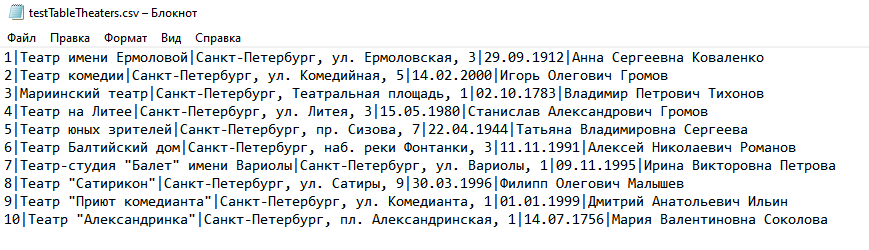
**. CSV**

**-**

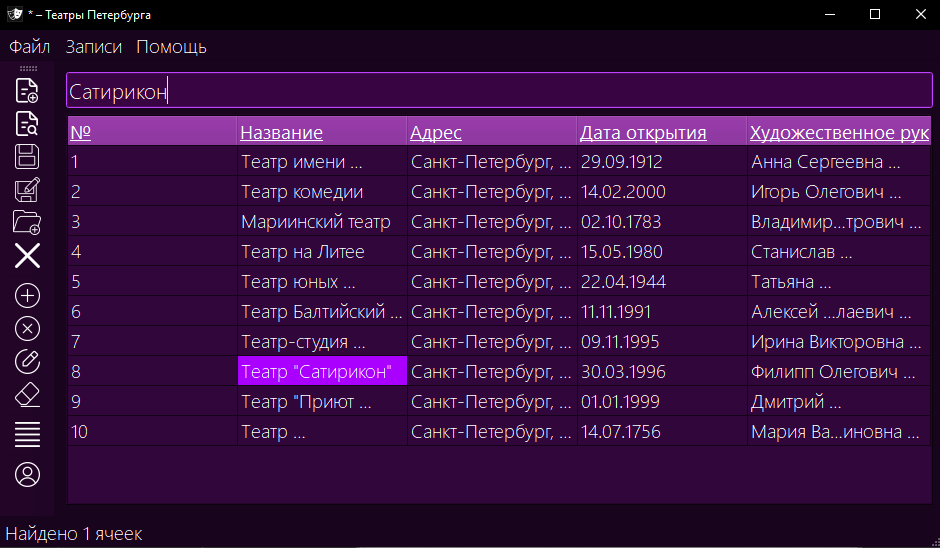
**отображение данных**

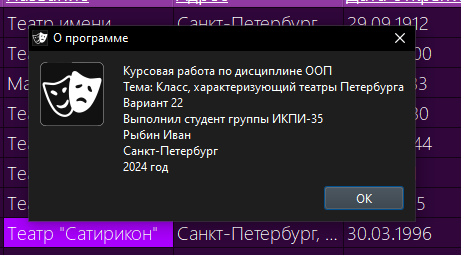
**.**

1. **Отображение CSV-файла.**



1. **Поиск.**





**11**

**. О программе**

**.**

# Код программы

|  |
| --- |
| **Файл "MyCourse.pro"** |
| QT += core gui  greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets  TARGET = MyCourse  TEMPLATE = app  DEFINES += QT\_DEPRECATED\_WARNINGS  CONFIG += c++14  SOURCES += \  MainWindow.cpp \  RedactorRecordWindow.cpp \  ShowAllRecordsCode.cpp \  Theater.cpp \  main.cpp  HEADERS += \  MainWindow.h \  RedactorRecordWindow.h \  ShowAllRecordsCode.h \  TextMessageDictionary.h \  Theater.h \  Vector.h  FORMS += \  MainWindow.ui \  RedactorRecordWindow.ui \  ShowAllRecordsCode.ui  qnx: target.path = /tmp/$${TARGET}/bin  else: unix:!android: target.path = /opt/$${TARGET}/bin  !isEmpty(target.path): INSTALLS += target  RESOURCES += \  resources.qrc |

|  |
| --- |
| **Файл "RedactorRecordWindow.h"**  #ifndef RedactorRecordWindow\_H  #define RedactorRecordWindow\_H  #include <QDialog>  #include "Theater.h"  namespace Ui {  class RedactorRecordWindow;  }  class RedactorRecordWindow : public QDialog  {  Q\_OBJECT  public:  explicit RedactorRecordWindow(QWidget \*parent = nullptr);  void setName(QString lineValue);  void setAddress(QString lineValue);  void setOpeningDate(QString lineValue);  void setArtisticDirection(QString lineValue);  Theater getData();  ~RedactorRecordWindow();  private:  Ui::RedactorRecordWindow \*ui;  };  #endif // RedactorRecordWindow\_H |

|  |
| --- |
| **Файл " RedactorRecordWindow.cpp"** |
| #include "RedactorRecordWindow.h"  #include "ui\_RedactorRecordWindow.h"  //Инициализация и открытие окна  RedactorRecordWindow::RedactorRecordWindow(QWidget \*parent) :  QDialog(parent),  ui(new Ui::RedactorRecordWindow)  {  ui->setupUi(this);  setFixedSize(size());  setWindowTitle(" ");  }  //Деструктор  RedactorRecordWindow::~RedactorRecordWindow()  {  delete ui;  }  //Все методы, которые устанавливают значения полей окна  void RedactorRecordWindow::setName(QString lineValue) {  ui->lineEdit->setText(lineValue);  }  void RedactorRecordWindow::setAddress(QString lineValue) {  ui->lineEdit\_2->setText(lineValue);  }  void RedactorRecordWindow::setOpeningDate(QString lineValue) {  ui->lineEdit\_3->setText(lineValue);  }  void RedactorRecordWindow::setArtisticDirection(QString lineValue) {  ui->lineEdit\_4->setText(lineValue);  }  //Метод, который обеспечивает считывание данных, которые пользователь ввёл в окне  Theater RedactorRecordWindow::getData() {  Theater resultRecord;  resultRecord.setName(ui->lineEdit->text().simplified());  resultRecord.setAddress(ui->lineEdit\_2->text().simplified());  resultRecord.setOpeningDate(ui->lineEdit\_3->text().simplified());  resultRecord.setArtisticDirection(ui->lineEdit\_4->text().simplified());  return resultRecord;  } |

|  |
| --- |
| **Файл " RedactorRecordWindow.ui"** |

|  |
| --- |
| **Файл "show\_data.h"** |
| #ifndef SHOW\_DATA\_H #define SHOW\_DATA\_H    #include <QDialog>    namespace Ui { class Show\_Data;  }  class Show\_Data : public QDialog  {  Q\_OBJECT  public: explicit Show\_Data(QWidget \*parent = nullptr);  void setText(const QString &str);  QString getText();  ~Show\_Data();  private slots:  void on\_pushButton\_clicked();  private:  Ui::Show\_Data \*ui;  }; |

#endif // SHOW\_DATA\_H

|  |
| --- |
| **Файл "show\_data.cpp"** |
| #include "show\_data.h"  #include "ui\_show\_data.h"    Show\_Data::Show\_Data(QWidget \*parent) : QDialog(parent), ui(**new** Ui::Show\_Data)  {  ui->setupUi(**this**);  }  void Show\_Data::setText(const QString &str) { ui->textEdit->setText(str);  }    QString Show\_Data::getText() { **return** ui->textEdit->toPlainText();  }    Show\_Data::~Show\_Data()  {  **delete** ui;  }  void Show\_Data::on\_pushButton\_clicked()  {  this->close(); |

}

|  |
| --- |
| **Файл "show\_data.ui"** |
| <?xml version="1.0" encoding="UTF-8"?><ui  version="4.0"><class>Show\_Data</class><widget class="QDialog" name="Show\_Data"><property  name="geometry"><rect><x>0</x><y>0</y><width>570</width><height>480</heigh t></rect></property><property  name="font"><font><pointsize>12</pointsize></font></property><property name="windowTitle"><string>CSV-отображение</string></property><layout class="QVBoxLayout" name="verticalLayout"><item><widget class="QTextEdit" name="textEdit"><property  name="font"><font><pointsize>12</pointsize></font></property><property name="textInteractionFlags"><set>Qt::TextSelectableByKeyboard|Qt::TextSele ctableByMouse</set></property></widget></item><item><widget class="QPushButton" name="pushButton"><property  name="text"><string>ОК</string></property></widget></item></layout></widge |

|  |
| --- |
| **Файл "Function.h"** |
| #ifndef FUNCTION\_H #define FUNCTION\_H  #include <QString>  #include <QtAlgorithms>    class Function { private:  // Имя  QString name = "";  // Тип  QString type = "";  // Количество аргументов int n\_arguments = 0;  // Аргументы  QString \*arguments = nullptr;  // Комментарий к функции  QString comment = "";  public:  // Конструктор по умолчанию  Function();  // Конструктор  explicit Function(QString name, QString type, QString comment, int n);  // Конструктор копирования  Function(const Function &da);    // Оператор копирования  Function &operator=(const Function &da);  // Конструктор перемещения  Function(Function &&da) noexcept;    // Оператор перемещения  Function &operator=(Function &&da) noexcept;  // Деструктор virtual ~Function();  // Оператор получения элемента по индексу I  QString &operator[](int i);    // Очистка массива void clear();  // Оператор приведения к типу bool explicit operator bool();  // Установка имени |

t><resources/><connections/></ui> void setName(QString arg);

// Получение имени const QString &getName();

// Установка типа void setType(QString arg);

// Получение имени const QString &getType();

// Установка комментария void setComment(QString arg);

// Получение комментария const QString &getComment();

// Получение количества аргументов const int &getNArguments();

// Установка числа аргументов void setNArguments(int n);

friend void swap(Function &first, Function &second) noexcept;

// Класс итератора по аргументам class iterator { public: QString \*i;

typedef std::random\_access\_iterator\_tag iterator\_category; typedef ptrdiff\_t difference\_type; typedef QString value\_type; typedef QString \*pointer; typedef QString &reference;

inline iterator() : i(nullptr) {}

inline explicit iterator(QString \*n) : i(n) {} inline iterator(const iterator &o): i(o.i){} inline QString &operator\*() const { **return** \*i; } inline QString \*operator->() const { **return** i; } inline QString &operator[](int j) const { **return** i[j]; } inline bool operator==(const iterator &o) const { **return** i == o.i;

}

inline bool operator!=(const iterator &o) const { **return** i != o.i;

}

inline bool operator<(const iterator& other) const { **return** i < other.i; }

inline bool operator<=(const iterator& other) const { **return** i <= other.i; }

inline bool operator>(const iterator& other) const { **return** i > other.i; }

inline bool operator>=(const iterator& other) const { **return** i >= other.i; }

inline iterator &operator++() { ++i; **return** \*this; } inline const iterator operator++(int) { QString \*n = i; ++i; **return** iterator(n); }

inline iterator &operator--() { i--; **return** \*this; }

|  |
| --- |
| inline const iterator operator--(int) { QString \*n = i; i--; **return** iterator(n); }  inline iterator &operator+=(int j) { i+=j; **return** \*this; } inline iterator &operator-=(int j) { i-=j; **return** \*this; } inline iterator operator+(int j) const { **return** iterator(i+j); } inline iterator operator-(int j) const { **return** iterator(i-j); } inline int operator-(iterator j) const { **return** int(i - j.i); } };  inline typename Function::iterator begin() { **return** iterator(&arguments[0]); }  inline typename Function::iterator end() { **return** iterator(&arguments[n\_arguments]); }  }; |

#endif // FUNCTION\_H

|  |
| --- |
| **Файл "Function.cpp"** |
| #include "Function.h"  #include <QtAlgorithms>  // Конструктор по умолчанию  Function::Function() {}    // Конструктор  Function::Function(QString name, QString type, QString comment, int n) : name(name), type(type), n\_arguments(n), comment(comment) { **if** (n\_arguments > 0) {  arguments = **new** QString[static\_cast<unsigned>(n\_arguments)]; }  }    // Конструктор копирования  Function::Function(const Function &da) : Function(da.name, da.type, da.comment, da.n\_arguments) { **if** (n\_arguments > 0) {  std::copy(da.arguments, da.arguments + da.n\_arguments, arguments); }  }    // Оператор копирования  Function & Function::operator=(const Function &da) {  Function temp(da); swap(\***this**, temp); **return** \*this;  }    // Конструктор перемещения  Function::Function(Function &&da) noexcept { name = std::move(da.name); type = std::move(da.type); n\_arguments = da.n\_arguments; arguments = da.arguments; |

comment = std::move(da.comment); da.n\_arguments = 0;

da.name = da.type = da.comment = ""; da.arguments = nullptr;

}

// Оператор перемещения

Function & Function::operator=(Function &&da) noexcept { **if** (**this** != &da) { swap(\***this**, da); da.n\_arguments = 0;

da.name = da.type = da.comment = ""; **delete**[] da.arguments; da.arguments = nullptr;

} **return** \*this;

}

// Деструктор

Function::~Function() { **delete**[] arguments;

}

// Оператор получения элемента по индексу I QString & Function::operator[](int i) { **return** arguments[i];

}

// Очистка массива void Function::clear() { n\_arguments = 0; name = type = comment = ""; **delete**[] arguments; arguments = nullptr;

}

// Оператор приведения к типу bool Function::operator bool() {

**return** !name.isEmpty() || !type.isEmpty() || !comment.isEmpty() || n\_arguments != 0 || arguments != nullptr;

}

// Установка имени

void Function::setName(QString arg) { name = arg;

}

// Получение имени

const QString & Function::getName() { **return** name;

}

// Установка типа

void Function::setType(QString arg) { type = arg;

|  |
| --- |
| // Получение имени  const QString & Function::getType() { **return** type;  }    // Установка комментария  void Function::setComment(QString arg) { comment = arg;  }    // Получение комментария  const QString & Function::getComment() { **return** comment;  }    // Получение количества аргументов const int & Function::getNArguments() { **return** n\_arguments;  }    // Установка числа аргументов void Function::setNArguments(int n) {  **if** (n > 0) {  QString \*temp = **new** QString[static\_cast<unsigned>(n)];  std::copy(arguments, arguments + std::min(n, n\_arguments), temp); std::swap(arguments, temp);  n\_arguments = n; **delete**[] temp; temp = nullptr; return;  }  // if n == 0 **delete**[] arguments; arguments = nullptr; n\_arguments = 0;  }  void swap(Function &first, Function &second) noexcept { std::swap(first.name, second.name); std::swap(first.type, second.type);  std::swap(first.n\_arguments, second.n\_arguments); std::swap(first.arguments, second.arguments); std::swap(first.comment, second.comment); |

} }

|  |
| --- |
| **Файл "Vector.h"** |
| #ifndef VECTOR\_H #define VECTOR\_H    #include <cstddef>  #include <stdexcept>  #include <algorithm> |

template <class T> class Vector { private:

// Текущий размер массива int n\_ = 0;

// Резервируемый размер массива (m\_ >= n\_) int m\_ = 10;

// Указатель на массив из m\_ элементов

T \*values\_ = nullptr; protected:

// Стандартное количество резервируемых элементов в массиве int standart = 10;

public:

// Конструктор

explicit Vector(int n = 0) : n\_(n) { **if** (n >= m\_) { m\_ = n \* 2;

}

values\_ = **new** T[m\_] {}; }

// Конструктор копирования

Vector(const Vector &da) : Vector(da.n\_) { n\_ = da.n\_;

std::copy(da.values\_, da.values\_ + n\_, values\_); }

// Оператор копирования

Vector &operator=(const Vector &da){ Vector temp(da); swap(\***this**, temp); **return** \*this; }

// Конструктор перемещения Vector(Vector &&da) noexcept {

n\_ = da.n\_, m\_ = da.m\_, values\_ = da.values\_, standart = da.standart;

da.n\_ = 0, da.m\_ = 0, da.values\_ = nullptr; }

// Оператор перемещения

Vector &operator=(Vector &&da) noexcept { **if** (**this** != &da) { swap(\***this**, da);

**delete**[] da.values\_, da.values\_ = nullptr; da.n\_ = 0, da.m\_ = 0;

} **return** \*this; }

// Деструктор virtual ~Vector() { **delete**[] values\_;

}

// Оператор получения элемента по индексу I

T &operator[](int i) { **return** values\_[i];

}

// [static] Получение размера массива static int size(const Vector &da) { **return** da.n\_;

}

// Добавление K элементов из массива DA в конец void add(const Vector &da, const int &k) { int mn = std::min(Vector::size(da), k); **for** (int i = 0; i < mn; ++i) { push\_back(da.values\_[i]);

}

}

// Вставка элемента D в позицию K

void insert(const T &d, const int &k) noexcept(false) { **if** (k == 0) { push\_front(d); return;

} **else** **if** (k == n\_) { push\_back(d); return;

} **else** **if** (k > 0 && k < n\_) { push\_back(d); T t = values\_[k], p; **for** (int i = k; i < n\_; ++i) {

p = values\_[i], values\_[i] = t, t = p;

}

values\_[k] = d; return;

}

throw std::out\_of\_range("Vector. Method insert. Out of range"); }

// Добавление элемента D в конец void push\_back(const T &d) { **if** (n\_ < m\_) { values\_[n\_] = d, ++n\_;

return;

}

T \*b = values\_; values\_ = **new** T[m\_ \* 2] {}; std::copy(b, b + n\_, values\_); **delete**[] b; m\_ \*= 2;

}

// Добавление элемента D в начало void push\_front(const T &d) { push\_back(d); T t = values\_[0], p; **for** (int i = 1; i < n\_; ++i) {

p = values\_[i], values\_[i] = t, t = p;

}

values\_[0] = d;

}

// Очистка массива void clear() { **delete**[] values\_; values\_ = **new** T[standart] {}; n\_ = 0, m\_ = standart;

}

// Получение размера массива

int size() const noexcept { **return** n\_; }

// Пустой ли массив

bool empty() const noexcept { **return** n\_ == 0; }

// Удаление элемента с индексом I T erase(int i) noexcept(false) { **if** (n\_ > 0 && i < n\_) { T d = values\_[i];

std::copy(values\_ + i + 1, values\_ + n\_, values\_ + i);

--n\_; **return** d;

}

throw std::out\_of\_range("Vector. Method erase. Out of range"); }

// Удаление первого (нулевого) элемента T pop\_front() { **return** erase(0);

}

// Удаление последнего элемента

T pop\_back() { T d = T(); **if** (n\_ > 0) {

d = values\_[n\_ - 1], --n\_;

} **return** d;

}

// Оператор приведения к типу bool explicit operator bool() { **return** n\_ != 0;

}

friend void swap(Vector &first, Vector &second) noexcept {

std::swap(first.standart, second.standart); std::swap(first.values\_, second.values\_); std::swap(first.m\_, second.m\_); std::swap(first.n\_, second.n\_);

}

// Класс итератора по элементам class iterator { public:

T \*i;

|  |
| --- |
| typedef std::random\_access\_iterator\_tag iterator\_category; typedef ptrdiff\_t difference\_type; typedef T value\_type; typedef T \*pointer; typedef T &reference;    inline iterator() : i(0) {}  inline explicit iterator(T \*n) : i(n) {} inline iterator(const iterator &o): i(o.i){} inline T &operator\*() const { **return** \*i; } inline T \*operator->() const { **return** i; } inline T &operator[](int j) const { **return** i[j]; }  inline bool operator==(const iterator &o) const { **return** i == o.i;  }  inline bool operator!=(const iterator &o) const { **return** i != o.i;  }  inline bool operator<(const iterator& other) const { **return** i < other.i; }  inline bool operator<=(const iterator& other) const { **return** i <= other.i; }  inline bool operator>(const iterator& other) const { **return** i > other.i; }  inline bool operator>=(const iterator& other) const { **return** i >= other.i; }  inline iterator &operator++() { ++i; **return** \*this; } inline const iterator operator++(int) { T \*n = i; ++i; **return** iterator(n); }  inline iterator &operator--() { i--; **return** \*this; } inline const iterator operator--(int) { T \*n = i; i--; **return** iterator(n); }  inline iterator &operator+=(int j) { i+=j; **return** \*this; } inline iterator &operator-=(int j) { i-=j; **return** \*this; } inline iterator operator+(int j) const { **return** iterator(i+j); } inline iterator operator-(int j) const { **return** iterator(i-j); } inline int operator-(iterator j) const { **return** int(i - j.i); }  };  inline typename Vector<T>::iterator begin() { **return** iterator(&values\_[0]); }  inline typename Vector<T>::iterator end() { **return** iterator(&values\_[n\_]); }    }; |

#endif // VECTOR\_H

|  |
| --- |
| **Файл "mainwindow.h"** |
| #ifndef MAINWINDOW\_H #define MAINWINDOW\_H    #include <QMainWindow>  #include <QMessageBox> #include "Vector.h" |

#include "Function.h"

|  |
| --- |
| namespace Ui { class MainWindow;  }  class MainWindow : public QMainWindow  {  Q\_OBJECT public: explicit MainWindow(QWidget \*parent = nullptr);  ~MainWindow();  private slots: void on\_action\_New\_triggered();    void on\_action\_About\_triggered();    void on\_action\_Exit\_triggered();    void on\_action\_Add\_triggered();    void on\_lineEdit\_textChanged(const QString &arg1);    void on\_action\_Edit\_triggered();    void on\_action\_Delete\_triggered();    void on\_action\_Clear\_triggered();    void on\_action\_Open\_triggered(bool isClean = true);    void on\_action\_Save\_triggered();    void on\_action\_SaveAs\_triggered();    void on\_action\_Merge\_triggered();    void on\_tableWidget\_cellDoubleClicked(int, int);    void on\_action\_ShowData\_triggered();  private:  Ui::MainWindow \*ui;  QString file\_ = "";  QColor FunctionType = QColor(0, 0, 127);  QColor FunctionComment = QColor(0, 85, 0);  Vector<Function> main\_vector; }; |

#endif // MAINWINDOW\_H

|  |
| --- |
| **Файл "mainwindow.cpp"** |
| #include "mainwindow.h"  #include "ui\_mainwindow.h"  #include "add\_dialog.h" #include "show\_data.h" |

|  |
| --- |
| #include "QFileDialog" #include "QTextStream"  #include "Vector.h"    MainWindow::MainWindow(QWidget \*parent) : QMainWindow(parent), ui(**new** Ui::MainWindow)  {  ui->setupUi(**this**);  }    MainWindow::~MainWindow()  {  **delete** ui;  }  void MainWindow::on\_action\_New\_triggered()  {  **if** (ui->tableWidget->rowCount() > 0) { QMessageBox msgBox;  msgBox.setWindowTitle("Внимание");  msgBox.setText("Вы уверены, что хотите создать новый файл?**\n**Все  проделанные изменения не будут сохранены."); msgBox.setIcon(QMessageBox::Question);  msgBox.setStandardButtons(QMessageBox::Yes | QMessageBox::No); msgBox.setDefaultButton(QMessageBox::Yes); **if** (msgBox.exec() == QMessageBox::No) { return;  } }  ui->tableWidget->setRowCount(0); main\_vector.clear(); file\_ = ""; setWindowTitle("\*" + file\_ + " – Функции языка программирования");  }  void MainWindow::on\_action\_Exit\_triggered()  {  **if** (ui->tableWidget->rowCount() > 0 || file\_ != "") { QMessageBox msgBox;  msgBox.setWindowTitle("Внимание");  msgBox.setText("Вы уверены, что хотите выйти?**\n**Все проделанные изменения не будут сохранены.");  msgBox.setIcon(QMessageBox::Question);  msgBox.setStandardButtons(QMessageBox::Yes | QMessageBox::No); msgBox.setDefaultButton(QMessageBox::Yes); **if** (msgBox.exec() == QMessageBox::No) { return;  } } exit(0);  }  void MainWindow::on\_action\_About\_triggered()  { **this** |
| "О программе",  "Курсовая работа по дисциплине ООП**\n**"  "Тема: Класс, характеризующий функцию (процедуру)  любого языка программирования**\n**" "Вариант 10**\n**"  "Выполнил студент группы ИКПИ-81**\n**"  "Коваленко Леонид**\n**"  "Санкт-Петербург**\n**"  "2019 год");  }  void MainWindow::on\_action\_Add\_triggered()  {  Add\_Dialog myDialog;  **if** (myDialog.exec() == QDialog::Accepted)  {  Function t = myDialog.getData(); auto len = t.getNArguments();  ui->tableWidget->setRowCount(ui->tableWidget->rowCount() + 1); ui->tableWidget->setItem(ui->tableWidget->rowCount() - 1, 0, **new** QTableWidgetItem(t.getType()));  ui->tableWidget->item(ui->tableWidget->rowCount() - 1, 0)-  >setTextColor(FunctionType);  ui->tableWidget->setItem(ui->tableWidget->rowCount() - 1, 1, **new** QTableWidgetItem(t.getName()));  ui->tableWidget->setItem(ui->tableWidget->rowCount() - 1, 2, **new**  QTableWidgetItem(QString::number(len)));  QString arg = ""; **if** (len > 0) { arg = t[0];  **for** (int i = 1; i < len; ++i) { arg += ", " + t[i];  } }  ui->tableWidget->setItem(ui->tableWidget->rowCount() - 1, 3, **new** QTableWidgetItem(arg));  ui->tableWidget->setItem(ui->tableWidget->rowCount() - 1, 4, **new** QTableWidgetItem(t.getComment()));  ui->tableWidget->item(ui->tableWidget->rowCount() - 1, 4)-  >setTextColor(FunctionComment); main\_vector.push\_back(t);  ui->statusBar->showMessage("Запись добавлена");  setWindowTitle("\*" + file\_ + " – Функции языка программирования"); }  }  void MainWindow::on\_lineEdit\_textChanged(const QString &arg1)  {  ui->tableWidget->setCurrentCell(-1, -1); **if** (arg1 == "") { return;  }  ui->tableWidget-  >setSelectionMode(QAbstractItemView::SelectionMode::MultiSelection); auto find\_items = ui->tableWidget->findItems(arg1, Qt::MatchContains); |
| **for** (int i = 0; i < len; ++i) { auto item = find\_items.at(i);  ui->tableWidget->setItemSelected(item, true);  }  ui->tableWidget-  >setSelectionMode(QAbstractItemView::SelectionMode::SingleSelection); ui->statusBar->showMessage("Найдено " + QString::number(len) + "  ячеек");  }  void MainWindow::on\_action\_Edit\_triggered()  {  auto list = ui->tableWidget->selectedItems();  **if** (list.size() > 0) { Add\_Dialog myDialog;  myDialog.setType(list.at(0)->text()); myDialog.setName(list.at(1)->text()); myDialog.setArgs(list.at(3)->text()); myDialog.setComment(list.at(4)->text()); **if** (myDialog.exec() == QDialog::Accepted)  {  Function t = myDialog.getData(); list.at(0)->setText(t.getType()); list.at(1)->setText(t.getName());  list.at(2)->setText(QString::number(t.getNArguments())); auto len = t.getNArguments();  QString arg = ""; **if** (len > 0) { arg = t[0];  **for** (int i = 1; i < len; ++i) { arg += ", " + t[i];  } }  list.at(3)->setText(arg);  list.at(4)->setText(t.getComment());  main\_vector[list.at(0)->row()].setType(t.getType()); main\_vector[list.at(1)->row()].setName(t.getName());  main\_vector[list.at(2)>row()].setNArguments(t.getNArguments()); int k = list.at(3)->row();  std::copy(t.begin(), t.end(), main\_vector[k].begin()); main\_vector[list.at(4)->row()].setComment(t.getComment()); ui->statusBar->showMessage("Запись отредактирована"); setWindowTitle("\*" + file\_ + " – Функции языка  программирования"); }  }  }  void MainWindow::on\_action\_Delete\_triggered()  {  auto list = ui->tableWidget->selectionModel()->selectedRows(); **if** (list.size() > 0) {  QMessageBox msgBox; "Внимание" |

#include "QFileInfo" QMessageBox::about( , int len = find\_items.size(); msgBox.setWindowTitle( );

|  |
| --- |
| msgBox.setText("Вы уверены, что хотите удалить строки с выбранными элементами?");  msgBox.setIcon(QMessageBox::Question);  msgBox.setStandardButtons(QMessageBox::Yes | QMessageBox::No); msgBox.setDefaultButton(QMessageBox::Yes); **if** (msgBox.exec() == QMessageBox::Yes) {  **for** (auto k = list.rbegin(); k != list.rend(); ++k) { auto idx = (\*k).row(); main\_vector.erase(idx);  ui->tableWidget->removeRow(idx);  }  ui->statusBar->showMessage("Запись удалена"); setWindowTitle("\*" + file\_ + " – Функции языка  программирования"); }  } **else** ui->statusBar->showMessage("Таблица пуста");  }  void MainWindow::on\_action\_Clear\_triggered()  {  **if** (ui->tableWidget->rowCount() > 0) { QMessageBox msgBox;  msgBox.setWindowTitle("Внимание"); msgBox.setText("Вы уверены, что хотите очистить таблицу?"); msgBox.setIcon(QMessageBox::Question);  msgBox.setStandardButtons(QMessageBox::Yes | QMessageBox::No); msgBox.setDefaultButton(QMessageBox::Yes); **if** (msgBox.exec() == QMessageBox::Yes) { ui->tableWidget->setRowCount(0); main\_vector.clear();  ui->statusBar->showMessage("Таблица очищена"); setWindowTitle("\*" + file\_ + " – Функции языка  программирования"); }  } **else** ui->statusBar->showMessage("Таблица пуста");  }  void MainWindow::on\_action\_Open\_triggered(bool isClean)  {  QString file\_name = QFileDialog::getOpenFileName(**this**, "Открыть файл...", ".", "CSV files (\*.csv);;All files (\*.\*);;"); QFileInfo check\_file(file\_name);  **if** (check\_file.exists() && check\_file.isFile()) { **if** (isClean) {  MainWindow::on\_action\_New\_triggered();  }  QFile file(file\_name);  **if** (file.open(QIODevice::ReadOnly)) { QTextStream in(&file); **while** (!in.atEnd()) {  Function t;  QString line = in.readLine();  QStringList fields = line.split(";"); **if** (fields.size() == 4) {  t.setType(fields[0].simplified()); |

|  |
| --- |
| QString s\_args = "";  QStringList args = fields[2].split(","); auto len = args.size();  t.setNArguments(len); **if** (len > 0) {  t[0] = args[0].simplified(); s\_args = t[0];  **for** (int i = 1; i < len; ++i) { t[i] = args[i].simplified(); s\_args += ", " + t[i];  }  }  t.setComment(fields[3].simplified()); ui->tableWidget->setRowCount(ui->tableWidget-  >rowCount() + 1);  ui->tableWidget->setItem(ui->tableWidget->rowCount() -  1, 0, **new** QTableWidgetItem(t.getType()));  ui->tableWidget->item(ui->tableWidget->rowCount() - 1,  0)->setTextColor(FunctionType);  ui->tableWidget->setItem(ui->tableWidget->rowCount() -  1, 1, **new** QTableWidgetItem(t.getName()));  ui->tableWidget->setItem(ui->tableWidget->rowCount() -  1, 2, **new** QTableWidgetItem(QString::number(t.getNArguments())));  ui->tableWidget->setItem(ui->tableWidget->rowCount() -  1, 3, **new** QTableWidgetItem(s\_args));  ui->tableWidget->setItem(ui->tableWidget->rowCount() -  1, 4, **new** QTableWidgetItem(t.getComment()));  ui->tableWidget->item(ui->tableWidget->rowCount() - 1,  4)->setTextColor(FunctionComment);  main\_vector.push\_back(t);  } }  file.close();  ui->statusBar->showMessage("Записи из файла добавлены"); file\_ = file\_name;  setWindowTitle(file\_ + " – Функции языка программирования"); } **else** ui->statusBar->showMessage("Файл '" + file\_name + "' не удалось открыть на чтение (" + file.errorString() + ")");  } **else** ui->statusBar->showMessage("Файл '" + file\_name + "' не  существует");  }  void MainWindow::on\_action\_Save\_triggered()  {  **if** (file\_ == "") {  MainWindow::on\_action\_SaveAs\_triggered(); return;  }  QFile file(file\_);  **if** (file.open(QIODevice::WriteOnly)) { QTextStream out(&file);  int len = ui->tableWidget->rowCount(); **for** (int i = 0; i < len; ++i) {  QString type = ui->tableWidget->item(i, 0)->text();  QString name = ui->tableWidget->item(i, 1)->text(); |
| QString comment = ui->tableWidget->item(i, 4)->text();  out << type << ";" << name << ";" << args << ";" << comment <<  "**\n**";  }  file.close();  ui->statusBar->showMessage("Файл записан");  setWindowTitle(file\_ + " – Функции языка программирования"); }  }  void MainWindow::on\_action\_SaveAs\_triggered()  {  QString file\_name = QFileDialog::getSaveFileName(**this**, "Сохранить файл...", ".", "CSV files (\*.csv);;All files (\*.\*);;"); **if** (file\_name != "") { file\_ = file\_name;  MainWindow::on\_action\_Save\_triggered();  }  }  void MainWindow::on\_action\_Merge\_triggered()  {  QString old\_file = file\_;  MainWindow::on\_action\_Open\_triggered(false); file\_ = old\_file;  ui->statusBar->showMessage("Данные объединены");  setWindowTitle("\*" + file\_ + " – Функции языка программирования");  }  void MainWindow::on\_tableWidget\_cellDoubleClicked(int, int) {  MainWindow::on\_action\_Edit\_triggered();  }  void MainWindow::on\_action\_ShowData\_triggered()  {  Show\_Data myDialog; QString s;  **for** (auto &k : main\_vector) { QString arg = "";  int len = k.getNArguments(); **if** (len > 0) { arg = k[0];  **for** (int i = 1; i < len; ++i) { arg += ", " + k[i];  } }  s += k.getType() + ";" + k.getName() + ";" + arg + ";" + k.getComment() + "**\n**";  }  myDialog.setText(s); myDialog.exec(); |
| **Файл "mainwindow.ui"** |
| <?xml version="1.0" encoding="UTF-8"?><ui  version="4.0"><class>MainWindow</class><widget class="QMainWindow" name="MainWindow"><property  name="geometry"><rect><x>0</x><y>0</y><width>920</width><height>480</heigh t></rect></property><property  name="font"><font><pointsize>12</pointsize></font></property><property name="windowTitle"><string>Функции языка  программирования</string></property><widget class="QWidget" name="centralWidget"><layout class="QVBoxLayout" name="verticalLayout"><item><widget class="QTableWidget" name="tableWidget"><property  name="font"><font><pointsize>12</pointsize></font></property><property name="contextMenuPolicy"><enum>Qt::DefaultContextMenu</enum></property><pr operty name="frameShape"><enum>QFrame::Box</enum></property><property name="frameShadow"><enum>QFrame::Raised</enum></property><property name="editTriggers"><set>QAbstractItemView::NoEditTriggers</set></property  ><property  name="selectionMode"><enum>QAbstractItemView::ExtendedSelection</enum></pr operty><property  name="selectionBehavior"><enum>QAbstractItemView::SelectRows</enum></prope rty><property  name="textElideMode"><enum>Qt::ElideMiddle</enum></property><attribute name="horizontalHeaderCascadingSectionResizes"><bool>true</bool></attribut e><attribute  name="horizontalHeaderMinimumSectionSize"><number>60</number></attribute>< attribute  name="horizontalHeaderDefaultSectionSize"><number>170</number></attribute>  <attribute name="horizontalHeaderShowSortIndicator" stdset="0"><bool>false</bool></attribute><attribute  name="horizontalHeaderStretchLastSection"><bool>true</bool></attribute><at tribute  name="verticalHeaderVisible"><bool>false</bool></attribute><column><proper ty name="text"><string>Тип</string></property><property  name="textAlignment"><set>AlignCenter</set></property></column><column><pr operty name="text"><string>Имя</string></property><property  name="textAlignment"><set>AlignCenter</set></property></column><column><pr operty name="text"><string>Число аргументов</string></property><property name="textAlignment"><set>AlignCenter</set></property></column><column><pr operty name="text"><string>Аргументы</string></property><property name="textAlignment"><set>AlignCenter</set></property></column><column><pr operty name="text"><string>Комментарий</string></property><property name="textAlignment"><set>AlignCenter</set></property></column></widget></ item><item><widget class="QLineEdit" name="lineEdit"><property name="text"><string/></property><property  name="placeholderText"><string>Поиск</string></property></widget></item></ layout></widget><widget class="QMenuBar" name="menuBar"><property  name="geometry"><rect><x>0</x><y>0</y><width>920</width><height>31</height  ></rect></property><widget class="QMenu" name="menu"><property name="title"><string>Файл</string></property><addaction name="action\_New"/><addaction name="action\_Open"/><addaction name="action\_Save"/><addaction name="action\_SaveAs"/><addaction name="action\_Merge"/><addaction name="action\_Exit"/></widget><widget class="QMenu" name="menu\_2"><property  Записи |
| name="action\_Add"/><addaction name="action\_Delete"/><addaction name="action\_Edit"/><addaction name="action\_Clear"/></widget><widget class="QMenu" name="menu\_3"><property  name="title"><string>Помощь</string></property><addaction name="action\_About"/></widget><addaction name="menu"/><addaction name="menu\_2"/><addaction name="menu\_3"/></widget><widget class="QToolBar" name="mainToolBar"><attribute  name="toolBarArea"><enum>TopToolBarArea</enum></attribute><attribute name="toolBarBreak"><bool>false</bool></attribute><addaction name="action\_New"/><addaction name="action\_Open"/><addaction name="action\_Save"/><addaction name="action\_SaveAs"/><addaction name="action\_Merge"/><addaction name="action\_Exit"/><addaction name="separator"/><addaction name="action\_Add"/><addaction name="action\_Delete"/><addaction name="action\_Edit"/><addaction name="action\_Clear"/><addaction name="separator"/><addaction name="action\_ShowData"/><addaction name="separator"/><addaction name="action\_About"/></widget><widget class="QStatusBar"  name="statusBar"/><action name="action\_New"><property name="icon"><iconset resource="resources.qrc"><normaloff>:/images/icons/new.png</normaloff>:/im ages/icons/new.png</iconset></property><property name="text"><string>Создать</string></property><property name="shortcut"><string>Ctrl+N</string></property></action><action name="action\_Open"><property name="icon"><iconset  resource="resources.qrc"><normaloff>:/images/icons/open.png</normaloff>:/i mages/icons/open.png</iconset></property><property name="text"><string>Открыть</string></property><property name="shortcut"><string>Ctrl+O</string></property></action><action name="action\_Save"><property name="icon"><iconset  resource="resources.qrc"><normaloff>:/images/icons/save.png</normaloff>:/i mages/icons/save.png</iconset></property><property name="text"><string>Сохранить</string></property><property name="shortcut"><string>Ctrl+S</string></property></action><action name="action\_SaveAs"><property name="icon"><iconset  resource="resources.qrc"><normaloff>:/images/icons/save\_as.png</normaloff>  :/images/icons/save\_as.png</iconset></property><property name="text"><string>Сохранить как</string></property><property name="shortcut"><string>Ctrl+Alt+S</string></property></action><action name="action\_Add"><property name="icon"><iconset  resource="resources.qrc"><normaloff>:/images/icons/add.png</normaloff>:/im ages/icons/add.png</iconset></property><property name="text"><string>Добавить</string></property><property name="shortcut"><string>Ctrl+A</string></property></action><action name="action\_Delete"><property name="icon"><iconset resource="resources.qrc"><normaloff>:/images/icons/delete.png</normaloff>:  /images/icons/delete.png</iconset></property><property name="text"><string>Удалить</string></property><property name="shortcut"><string>Ctrl+D</string></property></action><action name="action\_Clear"><property name="icon"><iconset  resource="resources.qrc"><normaloff>:/images/icons/clear.png</normaloff>:/ images/icons/clear.png</iconset></property><property name="text"><string>Очистить</string></property><property name="shortcut"><string>Ctrl+P</string></property></action><action name="action\_Edit"><property name="icon"><iconset  resource="resources.qrc"><normaloff>:/images/icons/edit.png</normaloff>:/i mages/icons/edit.png</iconset></property><property  Редактировать |

t.setName(fields[1].simplified()); QString args = ui->tableWidget->item(i, 3)->text(); } name="title"><string> </string></property><addaction name="text"><string> </string></property><property name="shortcut"><string>Ctrl+E</string></property></action><action name="action\_About"><property name="icon"><iconset

resource="resources.qrc"><normaloff>:/images/icons/about.png</normaloff>:/ images/icons/about.png</iconset></property><property name="text"><string>О программе</string></property><property

name="shortcut"><string>F1</string></property></action><action name="action\_Merge"><property name="icon"><iconset

resource="resources.qrc"><normaloff>:/images/icons/merge.png</normaloff>:/ images/icons/merge.png</iconset></property><property name="text"><string>Объединить</string></property><property name="shortcut"><string>Ctrl+M</string></property></action><action name="action\_Exit"><property name="icon"><iconset

resource="resources.qrc"><normaloff>:/images/icons/exit.png</normaloff>:/i mages/icons/exit.png</iconset></property><property name="text"><string>Выход</string></property><property name="shortcut"><string>Ctrl+X</string></property></action><action name="action\_ShowData"><property name="icon"><iconset

resource="resources.qrc"><normaloff>:/images/icons/show\_data.png</normalof f>:/images/icons/show\_data.png</iconset></property><property name="text"><string>Показать</string></property><property name="toolTip"><string>Показать внутренние

данные</string></property></action></widget><layoutdefault spacing="6" margin="11"/><resources><include

location="resources.qrc"/></resources><connections/></ui>

|  |
| --- |
| **Файл "main.cpp"** |
| #include "mainwindow.h" #include <QApplication>    int main(int argc, char \*argv[])  {  QApplication a(argc, argv); MainWindow w;  w.show(); **return** a.exec();  } |

# Заключение

Программа выполняет все поставленные задачи: работа с базами данных

(открытие, сохранение, добавление записи, удаление записей, редактирование записи и т. д.), работа с вектором функций языка (языков) программирования, интуитивнопонятное отображение всех возможностей программы.

В ходе выполнения курсовой работы была освоена работа с основными средствами языка C++, библиотеки QT и редактором QT Creator.

# Список литературы

1. С. В. Козин. Лекции по дисциплине "Объектно-ориентированное программирование".
2. Справочные материалы QT Creator.
3. М. Шлее — QT 5.3. Профессиональное программирование на C++. 2015.