

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 10

Выполнил:
студент 2 курса
дневного отделения
группы ИКПИ-81
Коваленко Л. А.

Санкт-Петербург 2019

Оглавление

1. Постановка задачи_____	3
2. Выбор и обоснование типа разрабатываемого контейнера_____	4
3. Разработка структуры программы_____	5
4. Разработка собственных классов_____	6
5. Разработка интерфейса_____	9
6. Инструкция пользователю _____	11
7. Код программы _____	19
8. Заключение _____	44
9. Список литературы _____	44

1. 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.

2. 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  
}
```

3. Разработка структуры программы

Программа состоит из интерфейса и кода.

Интерфейс:

- **Основное окно программы.**
 - Главное меню QMenuBar.
 - Панель инструментов QToolBar.
 - Таблица QTableWidget.
 - Строка поиска по таблице QLineEdit.
 - Статусная строка QStatusBar.
- **Диалог добавления / редактирования элемента.**
 - 4 метки QLabel: тип, название, аргументы и комментарий к функции.
 - 4 текстовые поля QLineEdit.
- **Диалог просмотра базы данных в CSV-формате.**
 - Многострочное текстовое поле QTextEdit.
 - 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.

4. Разработка собственных классов

Function — класс функции языка программирования.

- **Поля класса:**

- Имя: QString name.
- Тип: QString type.
- Количество аргументов: int n_arguments.
- Указатель на массив с аргументами: QString *arguments.
- Комментарий: QString comment.

- **Методы класса:**

- Оператор получения аргумента функции ЯП по индексу.
- Изменение имени функции: setName.
- Получение имени функции: getName.
- Изменение типа функции: setType.
- Получение типа функции: getType.
- Изменение комментария функции: setComment.
- Получение комментария функции: getComment.
- Изменение числа аргументов: setNArguments.
- Получение числа аргументов: getNArguments.

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.
- Добавление элемента в начало: push_front.
- Добавление элемента в конец: push_back.

- Очистка массива: `clear`.
- Получение булевого значения "пустой ли массив".
- Удаление элемента произвольного индекса: `erase`.
- Удаление первого элемента: `pop_front`.
- Удаление последнего элемента: `pop_back`.

MainWindow — класс главного окна.

- **Поля класса:**

- Строка, содержащая текущее имя файла: `QString file_`.
- Цвет типа функции в таблице: `QColor FunctionType`.
- Цвет комментария функции в таблице: `QColor FunctionComment`.
- 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`.

Add_Dialog — класс диалогового окна для добавления / редактирования записи.

- **Методы класса:**

- Установка типа функции ЯП: `setType`.
- Установка имени функции ЯП: `setName`.

- Установка аргументов функции ЯП: `setArgs`.
- Установка комментария функции ЯП: `setComment`.
- Получение данных в результате диалога: `getData`.

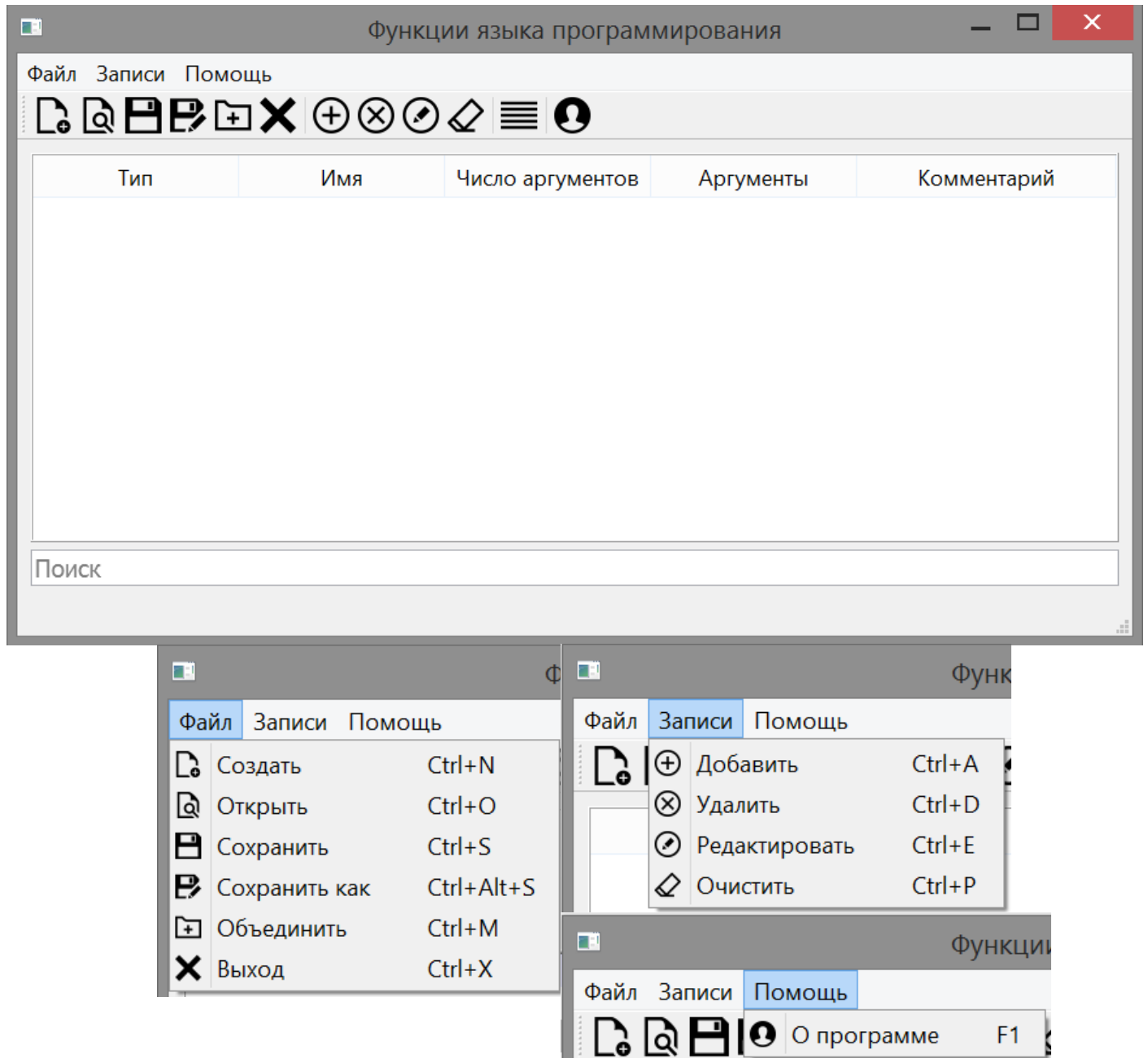
Show_Data — класс диалогового окна для отображения данных в CSV-формате.

- **Методы класса:**

- Установка текста в многострочное поле: `setText`.
- Получение текста из многострочного поля: `getText`.
- Нажатие кнопки закрытия: `on_pushButton_clicked`.

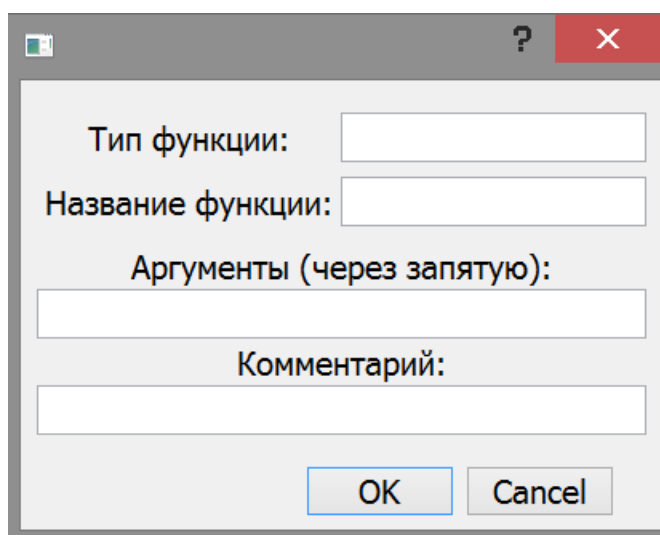
5. Разработка интерфейса

1. Основное окно программы (размер окна меняется).



Главное меню, панель инструментов, таблица, строка поиска и статусная строка (пустая)

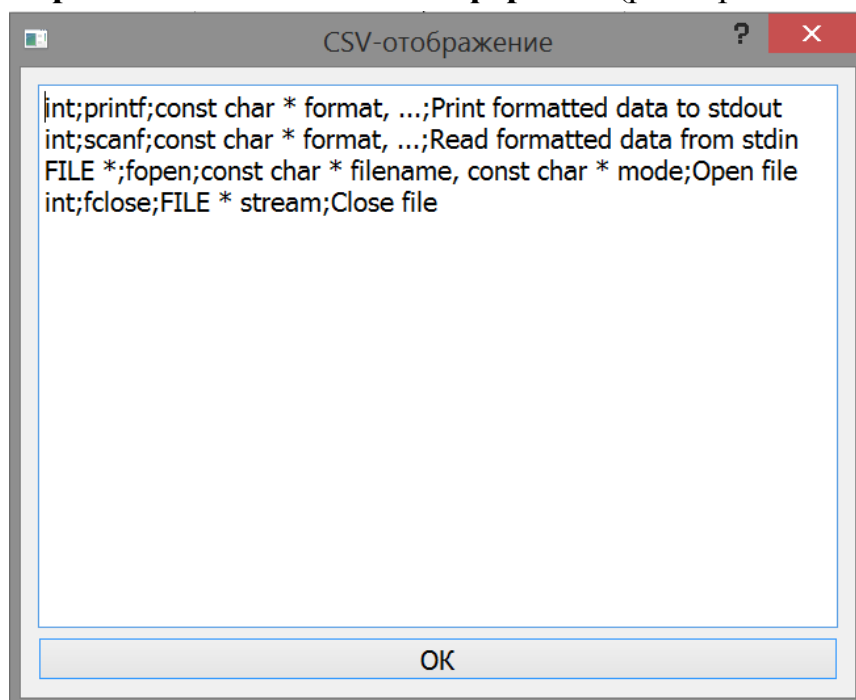
2. Диалог добавления / редактирования записи (размер окна не меняется).



The dialog box has a title bar with a question mark icon and a close button (X). It contains four input fields: 'Тип функции:' (Function type), 'Название функции:' (Function name), 'Аргументы (через запятую):' (Arguments (comma-separated)), and 'Комментарий:' (Comment). At the bottom are 'OK' and 'Cancel' buttons.

После нажатия ОК все пробелы в текстовых строках удаляются слева и справа и нормализуются в середине до одного, а точки с запятой преобразуются в запятые. Таким образом, все элементы базы данных приводятся к одному виду.

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



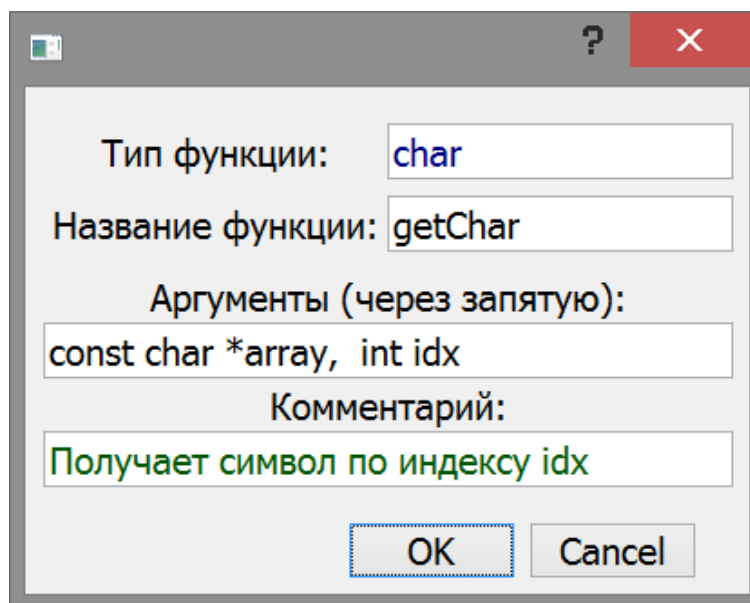
The dialog box is titled 'CSV-отображение' (CSV display). It contains a large text area with the following text: `int;printf;const char * format, ...;Print formatted data to stdout`
`int;scanf;const char * format, ...;Read formatted data from stdin`
`FILE *;fopen;const char * filename, const char * mode;Open file`
`int;fclose;FILE * stream;Close file`
At the bottom is an 'OK' button.

Текст изменить нельзя. Только для чтения.

После нажатия ОК окно закрывается.

6. Инструкция пользователю

1. Добавление записи ⊕.



Dialog box for adding a function record. It contains four input fields and two buttons.

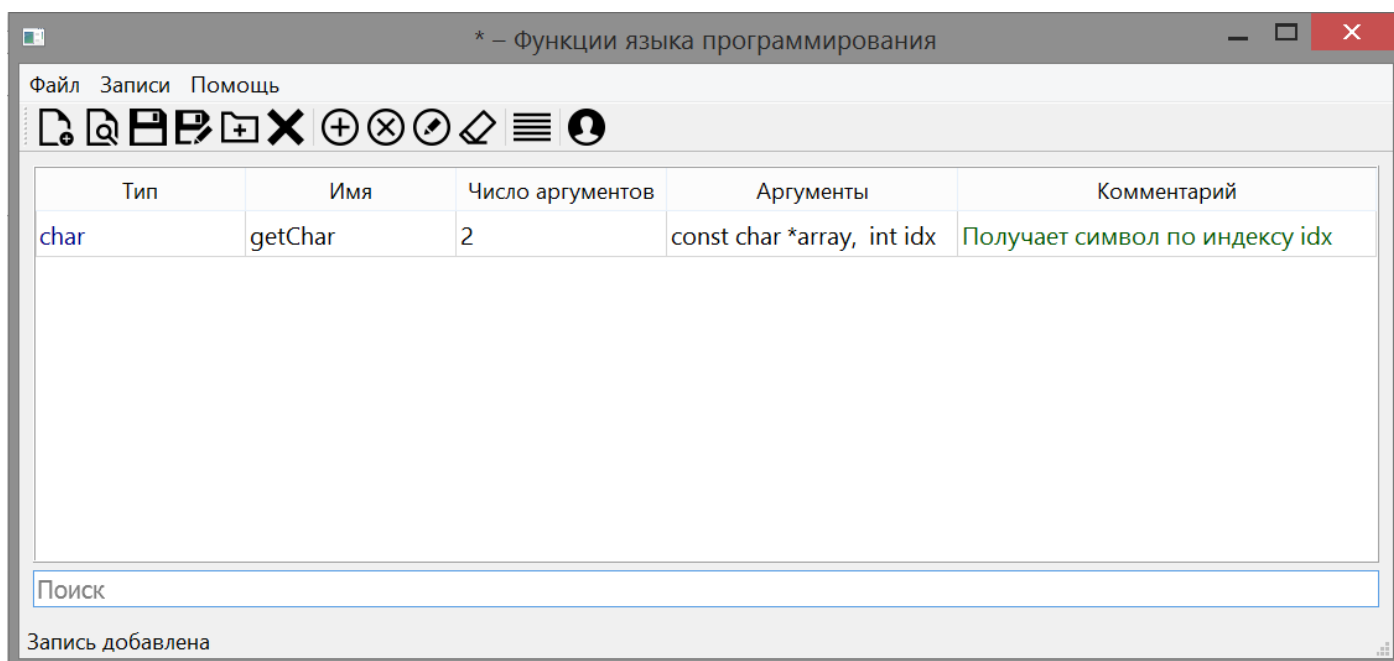
Тип функции:

Название функции:

Аргументы (через запятую):

Комментарий:

Buttons: OK, Cancel



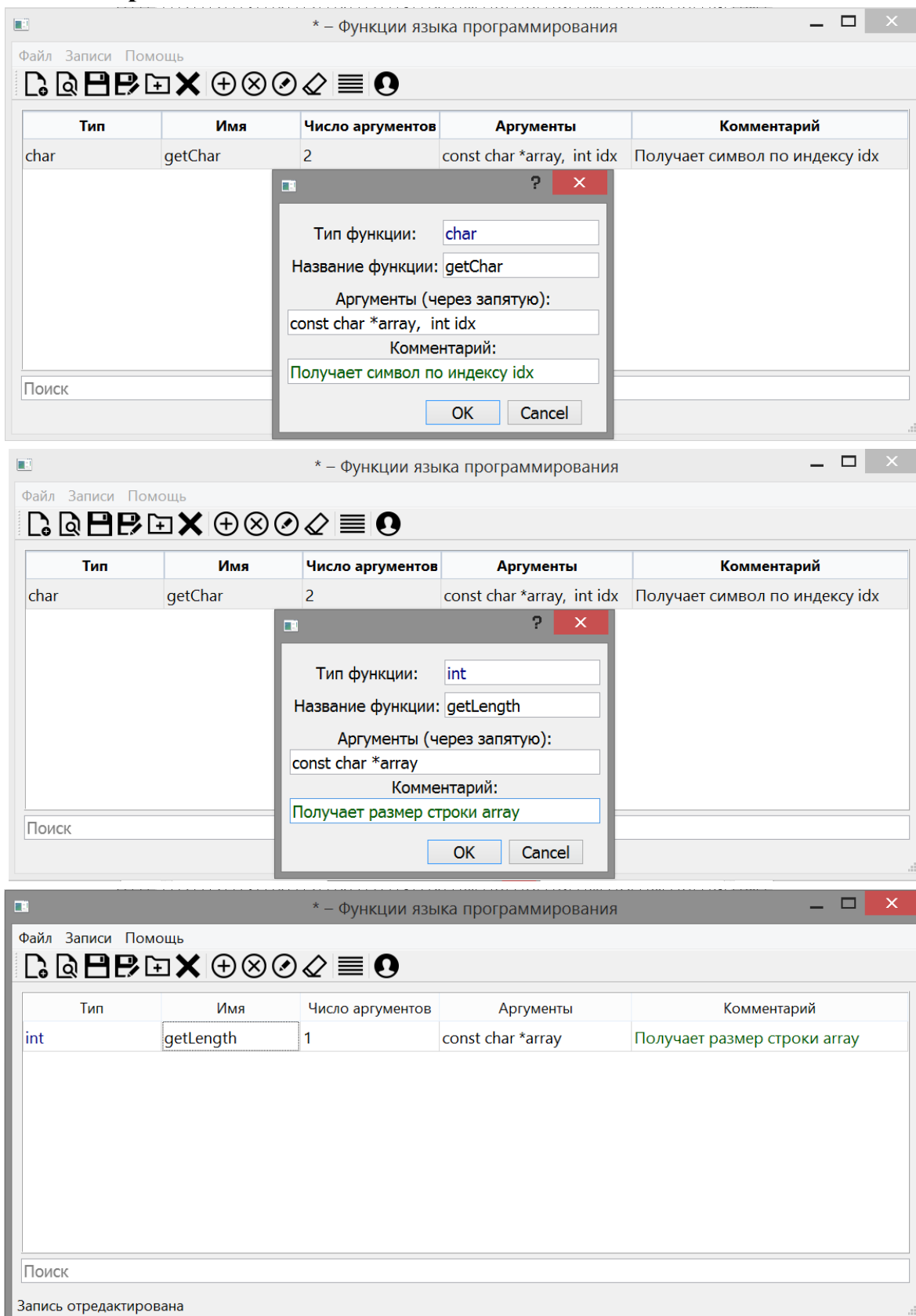
Main window titled "* – Функции языка программирования". It features a menu bar (Файл, Записи, Помощь), a toolbar with icons for file operations and editing, and a table listing functions.

Тип	Имя	Число аргументов	Аргументы	Комментарий
char	getChar	2	const char *array, int idx	Получает символ по индексу idx

Below the table is a search bar labeled "Поиск". At the bottom, a status bar displays "Запись добавлена".

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

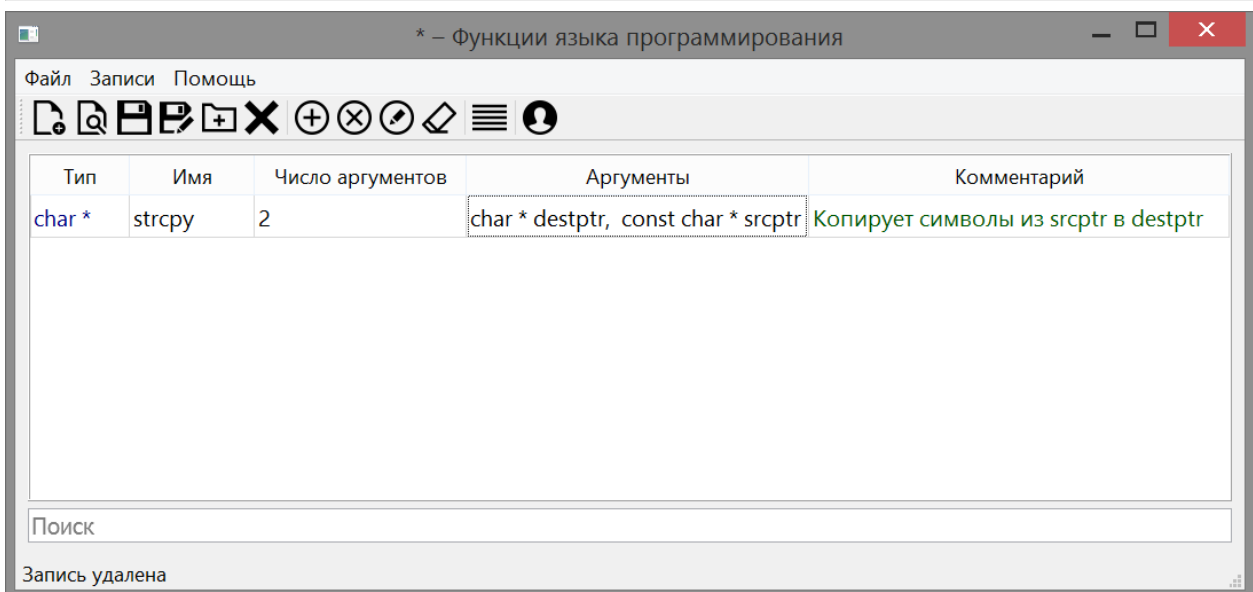
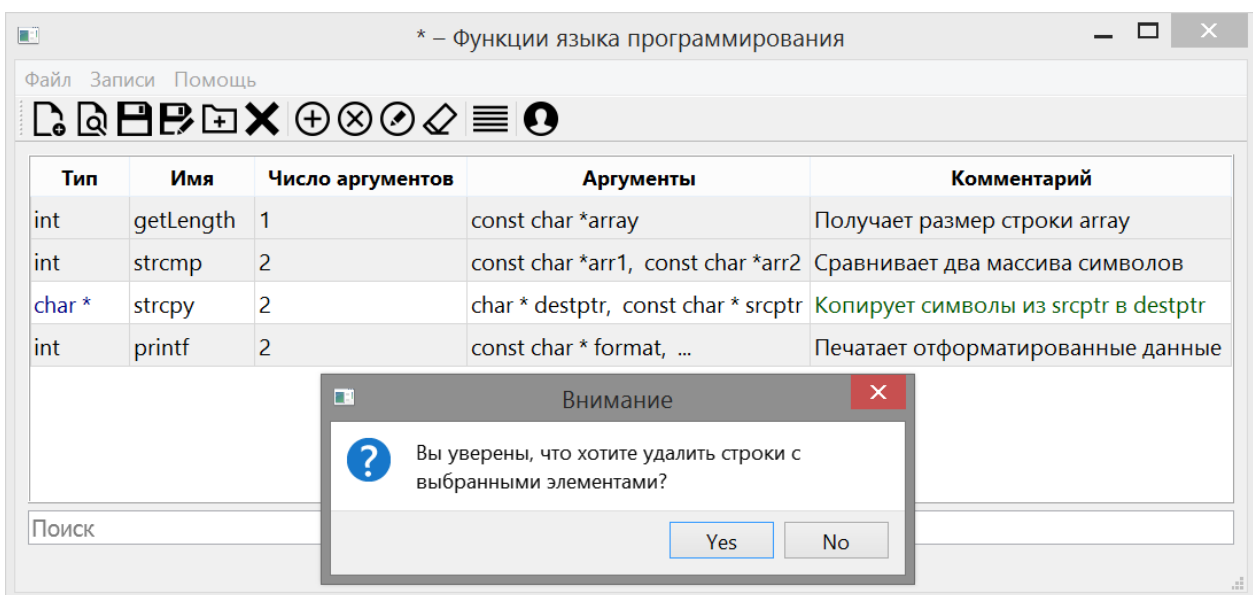
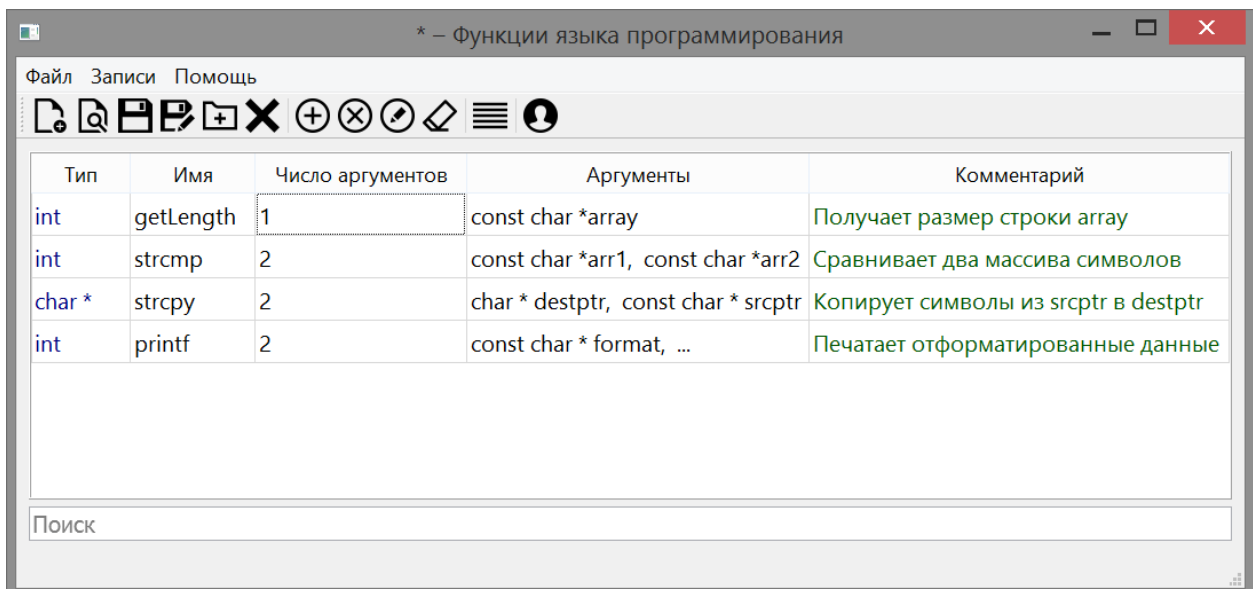
2. Редактирование записи .



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.

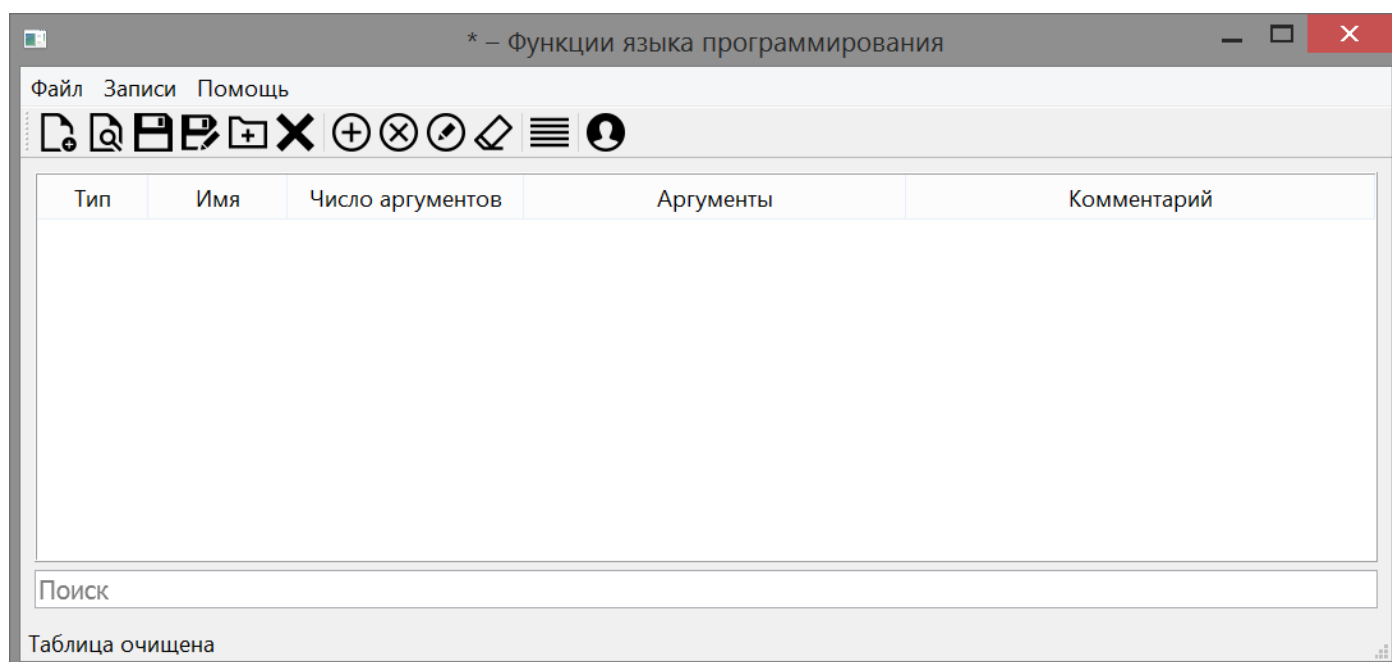
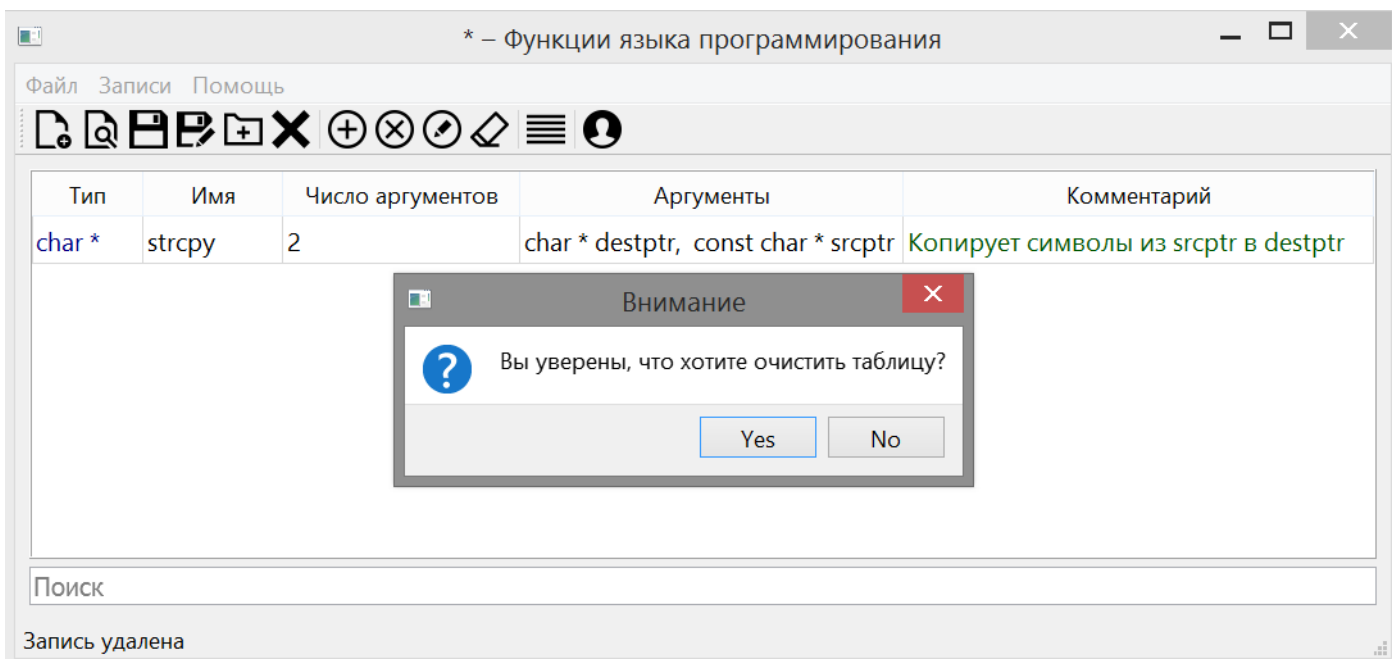
3. Удаление записи .



Удаление записи (записей) с подтверждением.

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

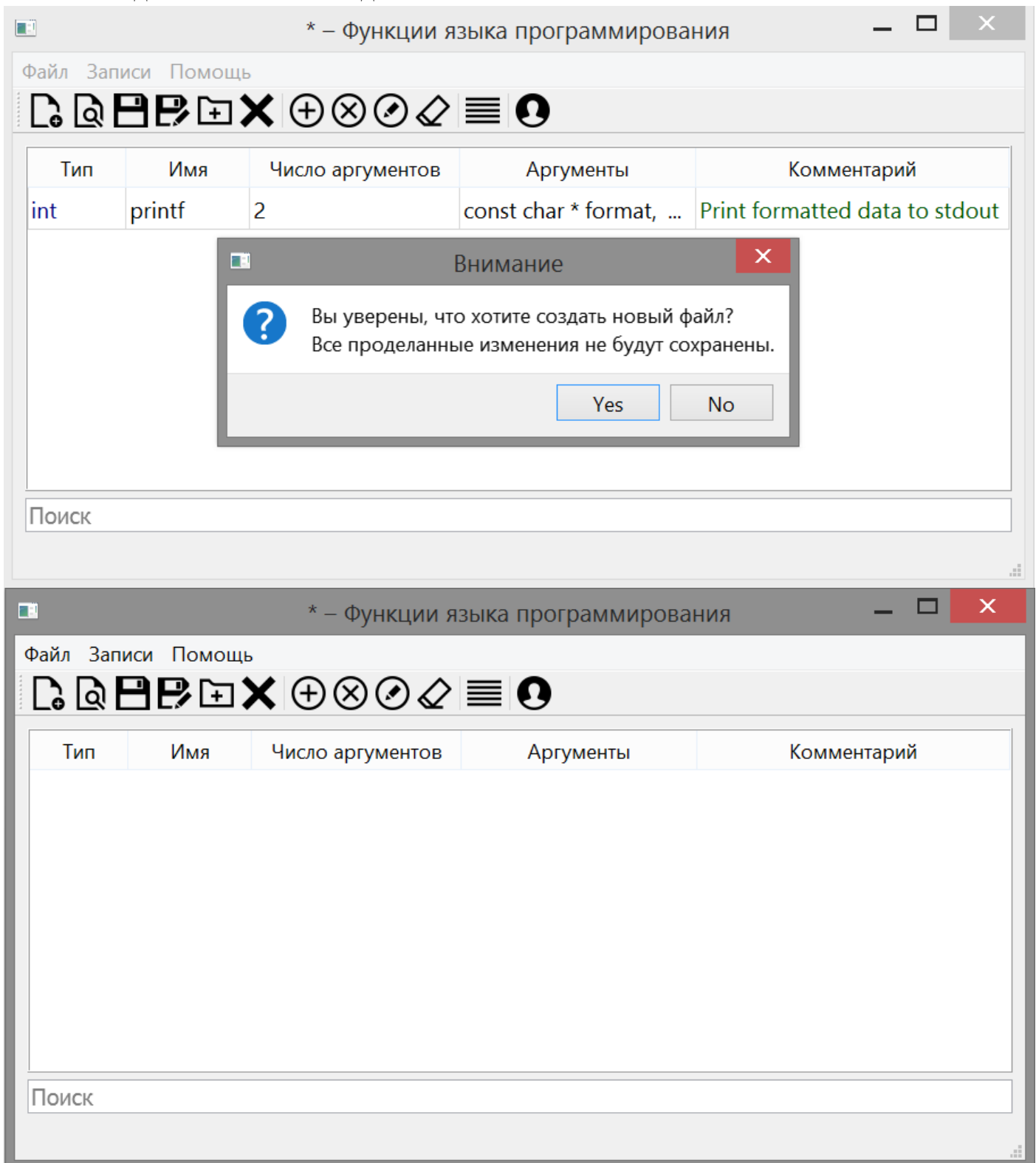
4. Очистить записи .



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

Сохранение файла:

Сохранить файл...

Упорядочить Создать папку

Имя	Дата изменения	Тип	Размер
Этот компьютер			
Видео			
debug	30.09.2019 9:59	Папка с файлами	

Открытие файла:

Открыть файл...

Упорядочить Создать папку

Имя	Дата изменения	Тип	Размер
Этот компьютер			
Видео			
Документы			
Загрузки			
my.csv	28.09.2019 10:...	Файл Microsoft...	5 КБ
my2.csv	30.09.2019 8:31	Файл Microsoft...	1 КБ
my3.csv	28.09.2019 10:...	Файл Microsoft...	0 КБ
tt.csv	30.09.2019 19:...	Файл Microsoft...	1 КБ

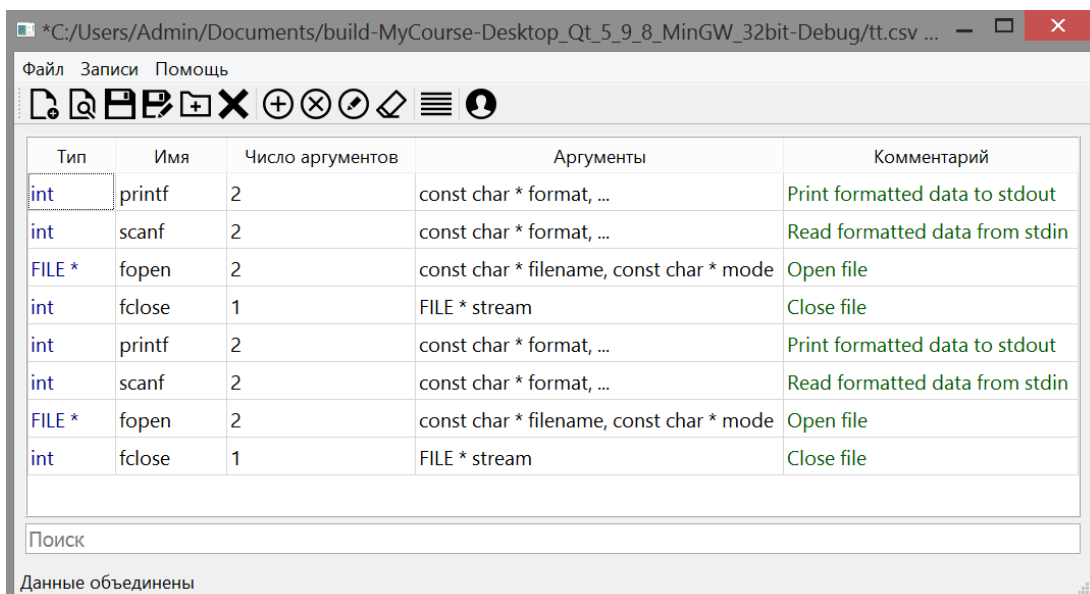
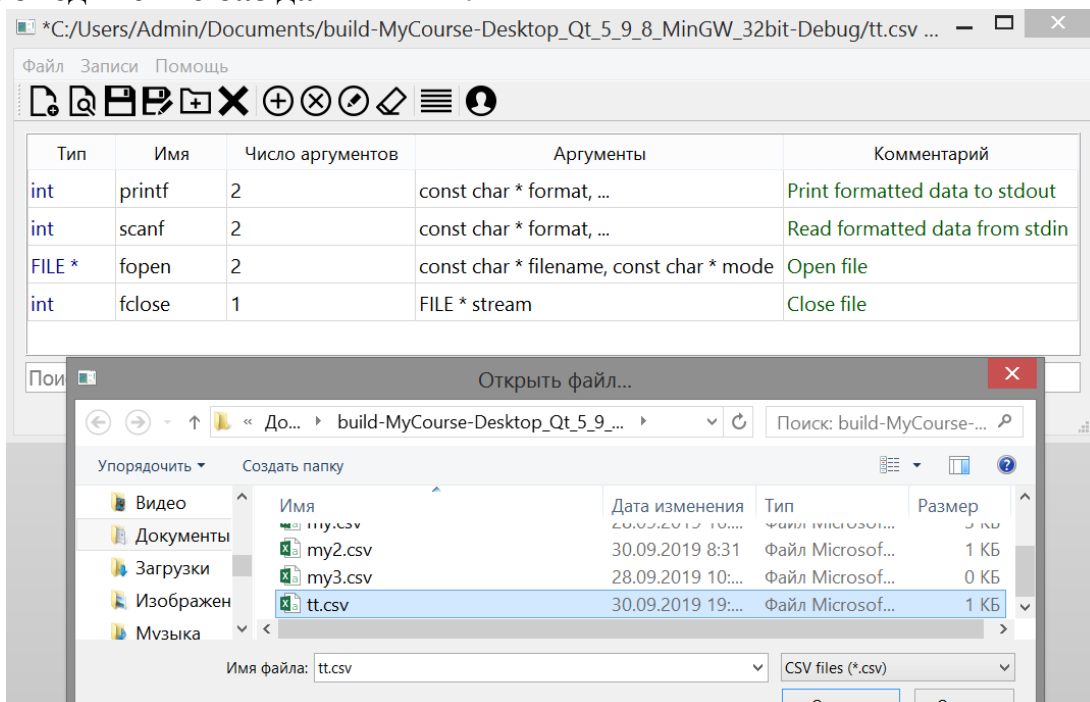
Загрузка данных из файла:

Записи из файла добавлены

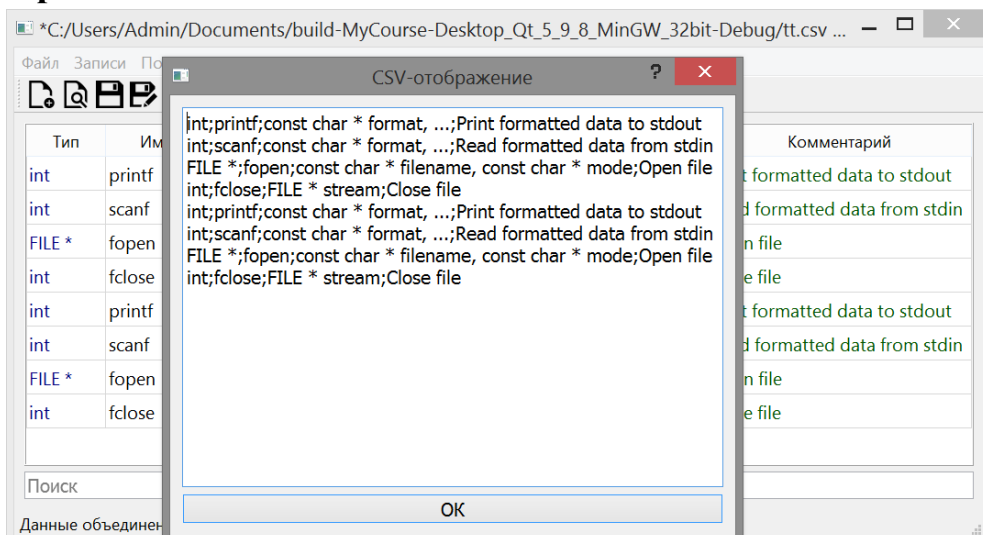
Функции языка программирования

Тип	Имя	Число аргументов	Аргументы	Комментарий
int	printf	2	const char * format, ...	Print formatted data to stdout
int	scanf	2	const char * format, ...	Read formatted data from stdin
FILE *	fopen	2	const char * filename, const char * mode	Open file
int	fclose	1	FILE * stream	Close file

7. Объединение баз данных



8. CSV-отображение данных



9. Отображение CSV-файла в Excel.

	A	B	C	D	E	F	G
1	int	printf	const char * format, ...	Print formatted data to stdout			
2	int	scanf	const char * format, ...	Read formatted data from stdin			
3	FILE *	fopen	const char * filename, const char * mode	Open file			
4	int	fclose	FILE * stream	Close file			
5							

10. Поиск.

Тип	Имя	Число аргументов	Аргументы	Комментарий
int	printf	2	const char * format, ...	Print formatted data to stdout
int	scanf	2	const char * format, ...	Read formatted data from stdin
FILE *	fopen	2	const char * filename, const char * mode	Open file
int	fclose	1	FILE * stream	Close file
int	printf	2	const char * format, ...	Print formatted data to stdout
int	scanf	2	const char * format, ...	Read formatted data from stdin
FILE *	fopen	2	const char * filename, const char * mode	Open file
int	fclose	1	FILE * stream	Close file

file

Найдено 10 ячеек

11. О программе

Тип	Имя	Число аргументов	Аргументы	Комментарий
int	printf	2	const char * format, ...	Print formatted data to stdout
int	scanf	2	const char * format, ...	Read formatted data from stdin
FILE *	fopen	2	const char * filename, const char * mode	Open file
int	fclose	1	FILE * stream	Close file
int	printf	2	const char * format, ...	Print formatted data to stdout
int	scanf	2	const char * format, ...	Read formatted data from stdin
FILE *	fopen	2	const char * filename, const char * mode	Open file
int	fclose	1	FILE * stream	Close file

О программе

Курсовая работа по дисциплине ООП
Тема: Класс, характеризующий функцию (процедуру) любого языка программирования
Вариант 10
Выполнил студент группы ИКПИ-81
Коваленко Леонид
Санкт-Петербург
2019 год

OK

Поиск

7. Код программы

Файл "MyCourse.pro"

```
#-----  
#  
# Project created by QtCreator 2019-09-04T17:36:20  
#  
#-----  
QT += core gui  
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets  
TARGET = MyCourse  
TEMPLATE = app  
  
# The following define makes your compiler emit warnings if you use  
# any feature of Qt which has been marked as deprecated (the exact warnings  
# depend on your compiler). Please consult the documentation of the  
# deprecated API in order to know how to port your code away from it.  
DEFINES += QT_DEPRECATED_WARNINGS  
  
# You can also make your code fail to compile if you use deprecated APIs.  
# In order to do so, uncomment the following line.  
# You can also select to disable deprecated APIs only up to a certain  
version of Qt.  
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs  
deprecated before Qt 6.0.0  
  
CONFIG += c++14  
  
SOURCES += \  
    Function.cpp \  
    add_dialog.cpp \  
    main.cpp \  
    mainwindow.cpp \  
    show_data.cpp  
  
HEADERS += \  
    Function.h \  
    Vector.h \  
    add_dialog.h \  
    mainwindow.h \  
    show_data.h  
  
FORMS += \  
    add_dialog.ui \  
    mainwindow.ui \  
    show_data.ui  
  
# Default rules for deployment.  
qnx: target.path = /tmp/${TARGET}/bin  
else: unix:!android: target.path = /opt/${TARGET}/bin  
!isEmpty(target.path): INSTALLS += target  
  
RESOURCES += \  
    resources.qrc
```

Файл "add_dialog.h"

```
#ifndef ADD_DIALOG_H
#define ADD_DIALOG_H

#include <QDialog>
#include "Function.h"

namespace Ui {
    class Add_Dialog;
}

class Add_Dialog : public QDialog
{
    Q_OBJECT

public:
    explicit Add_Dialog(QWidget *parent = nullptr);
    ~Add_Dialog();
    Function getData();
    void setType(QString arg);
    void setName(QString arg);
    void setArgs(QString arg);
    void setComment(QString arg);

private:
    Ui::Add_Dialog *ui;
};

#endif // ADD_DIALOG_H
```

Файл "add_dialog.cpp"

```
#include "add_dialog.h"
#include "ui_add_dialog.h"

Add_Dialog::Add_Dialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Add_Dialog)
{
    ui->setupUi(this);
    setFixedSize(size());
    setWindowTitle(" ");
}

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

void Add_Dialog::setType(QString arg) {
    ui->lineEdit->setText(arg);
}

void Add_Dialog::setName(QString arg) {
```

```

    ui->lineEdit_2->setText(arg);
}

void Add_Dialog::setArgs(QString arg) {
    ui->lineEdit_3->setText(arg);
}

void Add_Dialog::setComment(QString arg) {
    ui->lineEdit_4->setText(arg);
}

Function Add_Dialog::getData() {
    Function result;
    QString edit3 = ui->lineEdit_3->text().simplified().replace(";", ",");
    if (edit3.size() > 0) {
        QStringList temp = edit3.replace(";", ",").split(",");
        int len = temp.size();
        result.setNArguments(len);
        std::copy(temp.begin(), temp.end(), result.begin());
    }
    result.setType(ui->lineEdit->text().simplified().replace(";", ","));
    result.setName(ui->lineEdit_2->text().simplified().replace(";", ","));
    result.setComment(ui->lineEdit_4->text().simplified().replace(";",
", "));
    return result;
}

```

Файл "add_dialog.ui"

```

<?xml version="1.0" encoding="UTF-8"?><ui
version="4.0"><class>Add_Dialog</class><widget class="QDialog"
name="Add_Dialog"><property
name="geometry"><rect><x>0</x><y>0</y><width>400</width><height>280</height></rect></property><property
name="font"><font><pointsize>12</pointsize></font></property><property
name="windowTitle"><string/></property><widget class="QDialogButtonBox"
name="buttonBox"><property
name="geometry"><rect><x>10</x><y>240</y><width>361</width><height>32</height></rect></property><property
name="orientation"><enum>Qt::Horizontal</enum></property><property
name="standardButtons"><set>QDialogButtonBox::Cancel|QDialogButtonBox::Ok</set></property></widget><widget class="QLineEdit"
name="lineEdit_2"><property
name="geometry"><rect><x>200</x><y>60</y><width>191</width><height>31</height></rect></property></widget><widget class="QLabel"
name="label"><property
name="geometry"><rect><x>10</x><y>60</y><width>191</width><height>31</height></rect></property><property name="text"><string>Название
функции:</string></property><property
name="alignment"><set>Qt::AlignCenter</set></property></widget><widget
class="QLineEdit" name="lineEdit"><property
name="geometry"><rect><x>200</x><y>20</y><width>191</width><height>31</height></rect></property><property name="palette"><palette><active><colorrole
role="WindowText"><brush brushstyle="SolidPattern"><color
alpha="255"><red>0</red><green>0</green><blue>0</blue></color></brush></co

```

```

lorrole><colorrole role="Text"><brush brushstyle="SolidPattern"><color
alpha="255"><red>0</red><green>0</green><blue>127</blue></color></brush></
colorrole><colorrole role="PlaceholderText"><brush
brushstyle="SolidPattern"><color
alpha="128"><red>0</red><green>0</green><blue>127</blue></color></brush></
colorrole></active><inactive><colorrole role="WindowText"><brush
brushstyle="SolidPattern"><color
alpha="255"><red>0</red><green>0</green><blue>0</blue></color></brush></co
lorrole><colorrole role="Text"><brush brushstyle="SolidPattern"><color
alpha="255"><red>0</red><green>0</green><blue>127</blue></color></brush></
colorrole><colorrole role="PlaceholderText"><brush
brushstyle="SolidPattern"><color
alpha="128"><red>0</red><green>0</green><blue>127</blue></color></brush></
colorrole></inactive><disabled><colorrole role="WindowText"><brush
brushstyle="SolidPattern"><color
alpha="255"><red>120</red><green>120</green><blue>120</blue></color></brus
h></colorrole><colorrole role="Text"><brush
brushstyle="SolidPattern"><color
alpha="255"><red>120</red><green>120</green><blue>120</blue></color></brus
h></colorrole><colorrole role="PlaceholderText"><brush
brushstyle="SolidPattern"><color
alpha="128"><red>0</red><green>0</green><blue>0</blue></color></brush></co
lorrole></disabled></palette></property></widget><widget class="QLabel"
name="label_2"><property
name="geometry"><rect><x>10</x><y>20</y><width>191</width><height>31</heig
ht></rect></property><property name="text"><string>Тип
функции:</string></property><property
name="alignment"><set>Qt::AlignCenter</set></property></widget><widget
class="QLineEdit" name="lineEdit_3"><property
name="geometry"><rect><x>10</x><y>130</y><width>381</width><height>31</hei
ght></rect></property></widget><widget class="QLabel"
name="label_3"><property
name="geometry"><rect><x>10</x><y>100</y><width>381</width><height>31</hei
ght></rect></property><property name="text"><string>Аргументы (через
запятую):</string></property><property
name="alignment"><set>Qt::AlignCenter</set></property></widget><widget
class="QLineEdit" name="lineEdit_4"><property
name="geometry"><rect><x>10</x><y>190</y><width>381</width><height>31</hei
ght></rect></property><property name="palette"><palette><active><colorrole
role="Text"><brush brushstyle="SolidPattern"><color
alpha="255"><red>0</red><green>85</green><blue>0</blue></color></brush></c
olorrole><colorrole role="PlaceholderText"><brush
brushstyle="SolidPattern"><color
alpha="128"><red>0</red><green>85</green><blue>0</blue></color></brush></c
olorrole></active><inactive><colorrole role="Text"><brush
brushstyle="SolidPattern"><color
alpha="255"><red>0</red><green>85</green><blue>0</blue></color></brush></c
olorrole><colorrole role="PlaceholderText"><brush
brushstyle="SolidPattern"><color
alpha="128"><red>0</red><green>85</green><blue>0</blue></color></brush></c
olorrole></inactive><disabled><colorrole role="Text"><brush
brushstyle="SolidPattern"><color
alpha="255"><red>120</red><green>120</green><blue>120</blue></color></brus
h></colorrole><colorrole role="PlaceholderText"><brush
brushstyle="SolidPattern"><color
alpha="128"><red>0</red><green>0</green><blue>0</blue></color></brush></co

```

```

lorrole></disabled></palette></property></widget><widget class="QLabel"
name="label_4"><property
name="geometry"><rect><x>10</x><y>160</y><width>381</width><height>31</hei
ght></rect></property><property
name="text"><string>Комментарий:</string></property><property
name="alignment"><set>Qt::AlignCenter</set></property></widget></widget><t
abstops><tabstop>lineEdit</tabstop><tabstop>lineEdit_2</tabstop><tabstop>l
ineEdit_3</tabstop><tabstop>lineEdit_4</tabstop></tabstops><resources/><co
nnections><connection><sender>buttonBox</sender><signal>accepted()</signal
><receiver>Add_Dialog</receiver><slot>accept()</slot><hints><hint
type="sourcelabel"><x>248</x><y>254</y></hint><hint
type="destinationlabel"><x>157</x><y>274</y></hint></hints></connection><c
onnection><sender>buttonBox</sender><signal>rejected()</signal><receiver>A
dd_Dialog</receiver><slot>reject()</slot><hints><hint
type="sourcelabel"><x>316</x><y>260</y></hint><hint
type="destinationlabel"><x>286</x><y>274</y></hint></hints></connection></
connections></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</height></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::TextSelectableByMouse</set></property></widget></item><item><widget
class="QPushButton" name="pushButton"><property
name="text"><string>OK</string></property></widget></item></layout></widget></ui>
```


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
    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 "QFileInfo"

```

```

#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?\nВсе  

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?\nВсе 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()
{
    QMessageBox::about(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);
    int len = find_items.size();

```

```

    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;
        msgBox.setWindowTitle("Внимание");
    }
}

```

```

        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());
                    t.setName(fields[1].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 args = ui->tableWidget->item(i, 3)->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</height></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><property
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></property>
name="selectionBehavior"><enum>QAbstractItemView::SelectRows</enum></property>
name="textElideMode"><enum>Qt::ElideMiddle</enum></property><attribute
name="horizontalHeaderCascadingSectionResizes"><bool>true</bool></attribute>
name="horizontalHeaderMinimumSectionSize"><number>60</number></attribute>
name="horizontalHeaderDefaultSectionSize"><number>170</number></attribute>
<attribute name="horizontalHeaderShowSortIndicator"
stdset="0"><bool>false</bool></attribute><attribute
name="horizontalHeaderStretchLastSection"><bool>true</bool></attribute>
name="verticalHeaderVisible"><bool>false</bool></attribute><column><property
name="text"><string>Тип</string></property><property
name="textAlignment"><set>AlignCenter</set></property></column><column><property
name="text"><string>Имя</string></property><property
name="textAlignment"><set>AlignCenter</set></property></column><column><property
name="text"><string>Число аргументов</string></property><property
name="textAlignment"><set>AlignCenter</set></property></column><column><property
name="text"><string>Аргументы</string></property><property
name="textAlignment"><set>AlignCenter</set></property></column><column><property
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
name="title"><string>Записи</string></property><addaction
```

```

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>:/i
mages/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>:/i
mages/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
name="text"><string>Редактировать</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();
}

```

8. 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, интуитивно-понятное отображение всех возможностей программы.

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

9. 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.