ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»

(СПбГУТ)

Факультет инфокоммуникационных Сетей и систем (иксс)

кафедра программной инженерии и вычислительной техники (пи и вт)

Курсовая работа по дисциплине «Объектно-ориентированное программирование»

Вариант №14

Выполнил:

Студент группы ИКПИ-05

Принял:

Доцент кафедры ПИиВТ

Молошников Ф.А.

Подпись\_\_\_\_\_\_\_\_\_\_\_\_

Коробов С.А.

Подпись\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_»\_\_\_\_\_\_\_\_2021

СОДЕРЖАНИЕ

[1. Постановка Задачи 4](#_Toc86353893)

[2. Выбор и обоснование типа разрабатываемого контейнера 5](#_Toc86353894)

[3. Разработка структуры программы 5](#_Toc86353895)

[4. Выбор и обоснование используемых классов 6](#_Toc86353896)

[5. Разработка собственных классов 6](#_Toc86353897)

[5.1. Vector 6](#_Toc86353898)

[5.1.1. Свойства: 6](#_Toc86353899)

[5.1.2. Методы: 6](#_Toc86353900)

[5.1.3. Типы 7](#_Toc86353901)

[5.2. Iterator 7](#_Toc86353902)

[5.2.1. Свойства 7](#_Toc86353903)

[5.2.2. Методы 7](#_Toc86353904)

[5.3. ConstIterator 8](#_Toc86353905)

[5.4. ProgramCharact 8](#_Toc86353906)

[5.4.1. Свойства 8](#_Toc86353907)

[5.4.2. Методы 8](#_Toc86353908)

[5.5. MainWindow 9](#_Toc86353909)

[5.5.1. Свойства 9](#_Toc86353910)

[5.5.2. Методы 9](#_Toc86353911)

[5.6. EditDialog 10](#_Toc86353912)

[5.6.1. Методы 10](#_Toc86353913)

[5.7. ShowTxtDialog 10](#_Toc86353914)

[5.7.1. Методы 10](#_Toc86353915)

[6. Разработка интерфейса пользователя 10](#_Toc86353916)

[6.1. Главное окно приложения (Размер можно изменять) 10](#_Toc86353917)

[6.2. Диалоговое окно «Редактировать» (ширину окна можно изменять) 11](#_Toc86353918)

[6.3. Диалоговое окно «Показать в текстовом виде» (размер можно изменять) 12](#_Toc86353919)

[7. Инструкция пользователя 12](#_Toc86353920)

[7.1. Работа с файлами 12](#_Toc86353921)

[7.1.1. Создать новую базу данных 12](#_Toc86353922)

[7.1.2. Открыть существующую базу данных 12](#_Toc86353923)

[7.1.3. Сохранить базу данных 13](#_Toc86353924)

[7.1.4. Объединить базу данных 13](#_Toc86353925)

[7.1.5. Выход 15](#_Toc86353926)

[7.2. Работа с записями 15](#_Toc86353927)

[7.2.1. Добавить новую запись 15](#_Toc86353928)

[7.2.2. Удалить запись 16](#_Toc86353929)

[7.2.3. Редактировать запись 16](#_Toc86353930)

[7.2.4. Поиск по базе данных 17](#_Toc86353931)

[7.3. Отображение 17](#_Toc86353932)

[7.3.1. Показать в текстовом виде (CSV) 17](#_Toc86353933)

[7.4. Помощь 18](#_Toc86353934)

[7.4.1. О программе 18](#_Toc86353935)

[7.5. Список горячих клавиш 18](#_Toc86353936)

[8. Заключение 19](#_Toc86353937)

[9. Список литературы 19](#_Toc86353938)

[10. Приложение 19](#_Toc86353939)

[10.1. Vector.h 19](#_Toc86353940)

[10.2. Vector.inl 22](#_Toc86353941)

[10.3. ProgramCharact.h 28](#_Toc86353942)

[10.4. ProgramCharact.cpp 28](#_Toc86353943)

[10.5. mainwindow.h 29](#_Toc86353944)

[10.6. mainwindow.cpp 31](#_Toc86353945)

[10.7. mainwindow.ui 48](#_Toc86353946)

[10.8. editdialog.h 53](#_Toc86353947)

[10.9. editdialog.cpp 54](#_Toc86353948)

[10.10. editdialog.ui 54](#_Toc86353949)

[10.11. showtxtdialog.h 57](#_Toc86353950)

[10.12. showtxtdialog.cpp 58](#_Toc86353951)

[10.13. showtxtdialog.ui 58](#_Toc86353952)

[10.14. main.cpp 59](#_Toc86353953)

1. Постановка Задачи

Создать базу данных, работающую на основе контейнера, который следует разработать в процессе выполнения курсовой работы. В качестве прототипа разрабатываемого контейнера можно использовать контейнеры vector или list из библиотеки STL. Для работы с контейнером следует предусмотреть итератор. Остальные требования к контейнеру определяются студентом самостоятельно.

Разрабатываемая база данных должна быть создана использованием C++ и библиотеки Qt5. Среда программировании Qt Creater.

В качестве элементов контейнеров должны использоваться объекты класса, характеризующего пакет программ (название, фирма - изготовитель, размер и т.д.).

База данных должна обеспечивать выполнение следующих операций

* создание базы данных,
* объединение баз данных,
* добавления и удаления записей,
* редактирования записей,
* просмотра базы данных,
* поиск данных (найти данные, удовлетворяющие заданному критерию),
* запись на диск,
* чтения с диска.

Интерфейсная часть программы должна содержать следующие компоненты:

* окно «ABOUT»,
* главное меню,
* всплывающее меню,
* строку состояния,
* панель инструментов,
* подсказки по командам меню и панели инструментов.

Пояснительная записка должна содержать:

* постановку задачи,
* обоснование типа разрабатываемого контейнера,
* разработку структуры программы.
* разработку отдельных модулей
* выбор и обоснование используемых классов,
* разработку собственных классов,
* разработку интерфейса пользователя,
* инструкцию пользователя,
* заключение,
* список литературы.

1. Выбор и обоснование типа разрабатываемого контейнера

В качестве прототипа контейнера был выбран vector из стандартной библиотеки шаблонов С++. Выбор связан с меньшей сложностью организации элементов в контейнере и возможностью произвольного доступа к ним.

Разрабатываемый контейнер имеет функции, необходимые для решения задачи:

* Создание пустого контейнера, копирование или перемещение существующего контейнера.
* Добавление нового элемента в конец контейнера путем копирования или непосредственного построения в ячейке памяти.
* Удаление произвольного набора элементов, следующих друг за другом.
* Очищение контейнера
* Произвольный доступ к элементам по индексу.
* begin() и end(), которые вместе с классами Iterator и ConstIterator позволяют использовать конструкцию «Range-based for loop» (начиная с C++11). В примере показан проход по контейнеру VectorDB и сохранение значений его элементов в файл.

|  |
| --- |
| for(auto &iterator1: VectorDB)  {  fout << iterator1.GetName() << ";" << iterator1.GetManufacturer() <<";"<< iterator1.GetSize()<< ";" << iterator1.GetYear()<< ";" << iterator1.GetOS()<< "\n"  } |

1. Разработка структуры программы

Программа состоит из графического интерфейса, который состоит из форм и файла ресурса с иконками, и кода.

Интерфейс:

* Основное окно программы (компоновка по вертикали):
  + Главное меню QMenuBar
  + Панель инструментов QToolBar
  + Таблицы QTableWidget
  + Строки поиска QLineEdit
  + Строка состояния QStatusBar
* Диалог редактирования элемента (компоновка по вертикали):
  + 5 надписей QLabel и 5 текстовых полей QLineEdit, скомпонованных по сетке
  + Диалоговые кнопки QDialogButtonBox
* Диалог отображения текущих данных в текстовом виде, соответствующем формату CSV.
  + текстовое поле QTextEdit

Окно «О программе» и предупреждающие окна выводятся с помощью объектов класса QMessageBox.

Диалоговое меню выводится с помощью объекта класса QMenu. Диалоговые окна открытия и сохранения файла выводятся с помощью статических функций getOpenFileName и getSaveFileName класса QFileDialog

Код:

* Основной класс формы QT — окно главной программы.
* Класс формы QT — диалог редактирования записи.
* Класс формы QT — диалог просмотра базы данных в текстовом виде.
* Шаблонный класс контейнера типа Vector.
* Класс, характеризующий пакет программ.

Таким образом, программа состоит из следующих модулей:

Vector.h+inl

ProgramCharact.h+cpp

mainwindow.h+cpp+ui

editdialog.h+cpp+ui

showtxtdialog.h+cpp+ui

main.cpp

IconsResource.qrc

1. Выбор и обоснование используемых классов

Vector – класс-контейнер выбранного типа.

MainWindow – класс, наследующий QMainWindow, для работы с основным окном программы.

EditDialog – класс, наследующий QDialog, для работы с диалогом редактирования записи.

ShowTxtDialog – класс, наследующий QDialog, для работы с диалогом отображения БД в текстовом виде.

ProgramCharact – класс, характеризующий пакет программ.

1. Разработка собственных классов
   1. Vector

Шаблонный класса контейнера.

* + 1. Свойства:

|  |  |
| --- | --- |
| T\* data; | Указатель на динамический массив, содержащий элементы. |
| size\_t size; | Текущее число элементов в массиве |
| size\_t capasity; | Текущий размер массива |

* + 1. Методы:

|  |  |
| --- | --- |
| Vector(); | Конструктор по умолчанию, резервирует место для 10 элементов |
| Vector(const Vector& r); | Конструктор копирования |
| Vector(Vector&& r) noexcept; | Конструктор перемещения |
| ~Vector(); | Деструктор |
| void pushBack(const T& el); | Добавление элемента в конец путем копирования |
| template <typename ...Args>  void emplaceBack(Args&&... args); | Добавление элемента в конец путем перемещения или построения в памяти |
| void reserve(size\_t new\_capasity); | Перемещение динамического массива в область памяти нового объема. |
| void erase(size\_t begin, size\_t end); | Удаление элементов с begin по end |
| void clear(); | Удаление элементов, при сохранении прежней вместимости. |
| size\_t getSize() const; | Текущий размер массива |
| Vector& operator=(const Vector& r); | Перегрузка оператора присваивания с использованием семантики копирования |
| Iterator begin(); | Возвращает итератор, представляющий начальный элемент контейнера |
| Iterator end(); | Возвращает итератор, представляющий элемент, который бы находился после последнего элемента контейнера |
| ConstIterator begin()const; | Перегрузка двух методов выше по константности. Позволяют работать с константными контейнерами |
| ConstIterator end()const; |
| ConstIterator cbegin()const; | Реализация полностью совпадает с двумя предыдущими методами. Позволяют при их использовании явно запретить изменение элементов контейнера. |
| ConstIterator cend()const; |
| T& operator[](size\_t i); | Для получения и изменения элементов контейнера. |
| const T& operator[](size\_t i) const; |

* + 1. Типы

|  |  |
| --- | --- |
| Iterator | Класс итератор. Относится к вектору отношением включения и является дружественным к нему. |
| ConstIterator | Константный итератор. Относится к вектору отношением включения и является дружественным к нему. Отличается от Iterator тем, что может использоваться с объектами вида «const Vector<T> obj1». Так же может использоваться для запрета редактирования элементов объектов вида «Vector<T> obj1». |

* 1. Iterator
     1. Свойства

|  |  |
| --- | --- |
| Vector\* v; | указатель на контейнер |
| size\_t i; | Текущий элемент, который представляет итератор |

* + 1. Методы

|  |  |
| --- | --- |
| Iterator(); | Конструктор |
| Iterator(Vector& r); | Конструктор с параметром |
| Iterator(const Iterator& r) = default; | Конструтор копирования, который создается компилятором автоматически (default) |
| Iterator& operator=(const Iterator& r) = default; | Оператор присваивания, который будет создан компилятором автоматически |
| Iterator& operator++(); //prefix | Префиксный и постфиксный инкременты |
| **Iterator operator++(int);//postfix** |
| Iterator operator+(size\_t i)const; | Перегрузка арифметических и логических операторов. |
| Iterator operator-(size\_t i)const; |
| bool operator==(const Iterator& r); |
| bool operator!=(const Iterator& r); |
| **T& operator\*()const;** | Перегрузка оператора разыменования. Позволяет получать доступ к элементу контейнера |

* 1. ConstIterator

Аналогичен классу Iterator, за исключением конструктора

**ConstIterator**(const Vector& r);

и перегрузки оператора разыменования

const T& operator\*()const;

* 1. ProgramCharact

Класс, описывающий программу.

* + 1. Свойства

|  |  |
| --- | --- |
| QString Name="-"; | Название |
| QString Manufacturer="-"; | Фирма-изготовитель |
| long long Size=0; | Размер |
| int Year=0; | Год выпуска |
| QString OS="-"; | ОС |

* + 1. Методы

|  |  |
| --- | --- |
| ProgramCharact(); | Конструкторы |
| ProgramCharact(QString NName, QString NManufacture, long long NSize, int NYear, QString NOS); |
| QString GetManufacturer(); | Геттеры |
| long long GetSize(); |
| QString GetName(); |
| int GetYear(); |
| QString GetOS(); |
| void SetManufacturer(QString NManufacturer); | Сеттеры |
| void SetSize(long long NSize); |
| void SetName(QString NName); |
| void SetYear(int NYear); |
| void SetOS(QString NOS); |

* 1. MainWindow

Класс главного окна.

* + 1. Свойства

|  |  |
| --- | --- |
| Vector<ProgramCharact> VectorDB; | Контейнер с БД |
| Vector<size\_t> SearchRelationVector; | Контейнер, хранящий индексы найденных записей |
| QString FileName="Untitled"; | Путь к текущему открытому файлу |
| bool IsFilePathOpen=false; | Флаги |
| bool IsEditedFlag=false; |

* + 1. Методы

|  |  |
| --- | --- |
| MainWindow(QWidget \*parent) | Подготовка таблицы и связь сигнала со слотом для меню |
| void on\_Create\_triggered(); | Создать |
| void on\_Open\_triggered(); | Открыть |
| void on\_Save\_triggered(); | Сохранить |
| void on\_SaveAs\_triggered(); | Сохранить как |
| void on\_AddDB\_triggered(); | Объединить БД |
| void on\_Quit\_triggered(); | Выход |
| void on\_AddRecord\_triggered(); | Добавить |
| void on\_DeleteRecord\_triggered(); | Удалить |
| void on\_Edit\_triggered(); | Редактировать |
| void on\_ShowTxt\_triggered(); | Показать в текстовом виде |
| void on\_About\_triggered(); | О программе |
| void ShowContextMenu(const QPoint&); | Слот, создающий и отображающий контекстное меню |
| void on\_tableWidget\_cellChanged(int row, int column); | Ставит соответствующий флаг и добавляет \* в заголовок окна. |
| void on\_LineEditSearch\_textEdited(const QString &arg1); | Производит поиск по БД и отображает результат в таблице. |
| void on\_tableWidget\_cellClicked(int row, int column); | Показывает текущую ячейку в строке состояния |
| void TableToVectorSearch(); | Заполняет VectorDB данным таблицы в соответствии с SearchRelationVector |
| void VectorToTableSearch(); | Отображает данные, хранящиеся в VectorDB, в соответствии с SearchRelationVector |
| void UpdateRelation(); | Обновляет SearchRelationVector в соответствии с данными VectorDB и поискового запроса |
| void closeEvent(QCloseEvent \*event); | Переопределенный метод. Перехват сообщения закрытия программы |

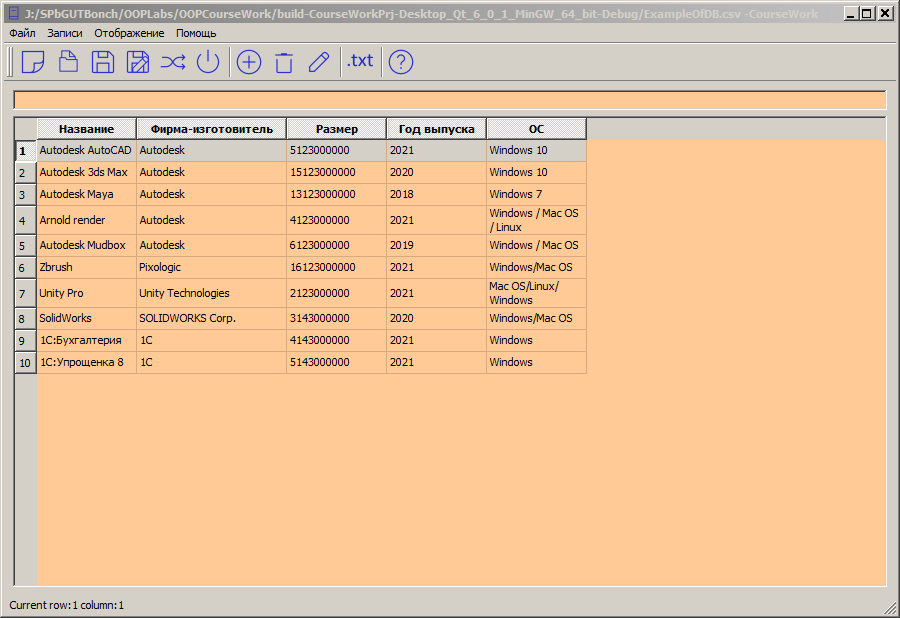
* 1. EditDialog
     1. Методы

|  |  |
| --- | --- |
| EditDialog(QString txt0,QString txt1, QString txt2, QString txt3, QString txt4, QWidget \*parent = nullptr); | Конструктор, устанавливающий значения текстовых полей элементов QLineEdit перед открытием диалога. |
| QString getName(); | Геттеры, вызываются после закрытия кнопкой «ок» диалога. |
| QString getManufacturer(); |
| QString getSize(); |
| QString getYear(); |
| QString getOS(); |

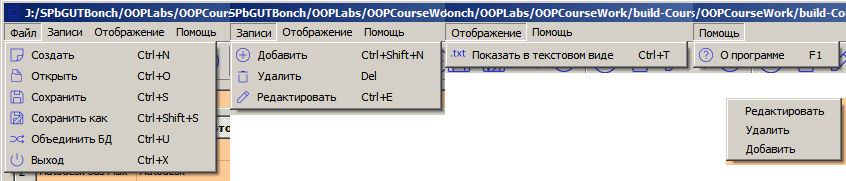
* 1. ShowTxtDialog
     1. Методы

|  |  |
| --- | --- |
| void TextEditSetText(QString StrCSV); | Установка текста в поле QTextEdit |

1. Разработка интерфейса пользователя
   1. Главное окно приложения (Размер можно изменять)



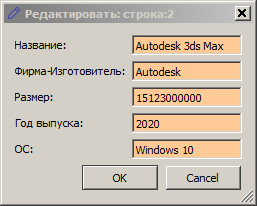
Главное окно приложения



Главное и контекстное меню

Главное меню состоит из разделов «Файл», «Записи», «Отображение», «Помощь». Пункты меню дублируются в панели инструментов класса QToolBar. Контекстное меню класса QMenu состоит из пунктов «Редактировать», «Удалить», «Добавить». Строка поиска реализована с помощью QLineEdit. Таблица реализована с помощью класса QTableWidget. Таблица позволяет выделять по одной строке и редактировать ячейки. Строка поиска реагирует на ввод поиска. Когда строка пустая, выводится вся таблица. Строка состояния класса QStatusBar отображает выбранное действие, положение текущей выделенной ячейки, число записей в БД и число найденных записей.

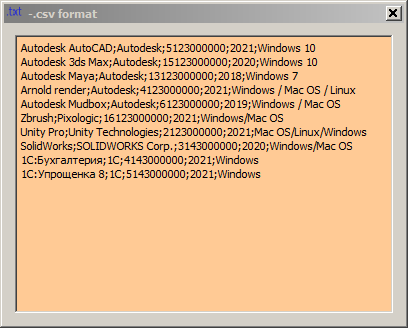
* 1. Диалоговое окно «Редактировать» (ширину окна можно изменять)



Диалоговое окно «Редактировать»

Диалоговое окно позволяет просмотреть и отредактировать выбранную запись

* 1. Диалоговое окно «Показать в текстовом виде» (размер можно изменять)

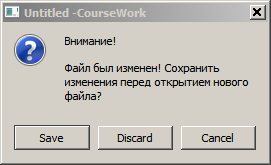


Диалоговое окно «Показать в текстовом виде»

Диалоговое окно позволяет просмотреть базу данных в том виде, в котором она будет сохранена в файл.

1. Инструкция пользователя
   1. Работа с файлами
      1. Создать новую базу данных

После открытия программы будет открыт новая база данных. Чтобы создать новую базу данных нужно выбрать пункт меню [Файл]->[Создать] либо воспользоваться значком J:\SPbGUTBonch\OOPLabs\OOPCourseWork\Отчет\Icons\sticker_121741.png на панели инструментов. Если текущая база данных не была сохранена, то программа откроет предупреждающий диалог.

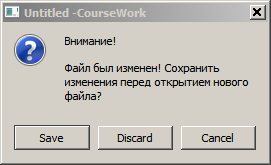


Предупреждающий диалог

После выбора нужного действия таблица станет пустой, заголовок изменится на «Untitled -CourseWork»

* + 1. Открыть существующую базу данных

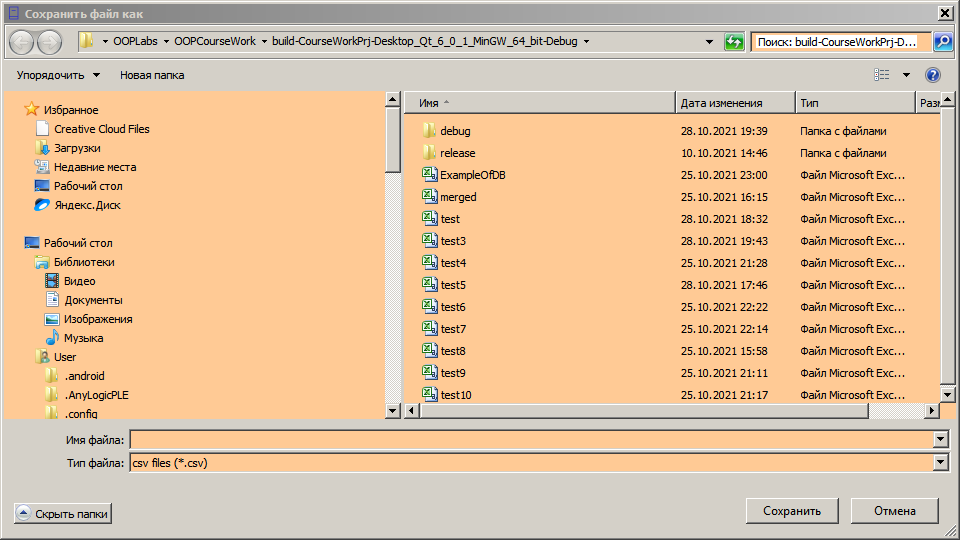
Чтобы открыть существующий файл, достаточно выбрать пункт меню [Файл]->[Открыть] либо либо воспользоваться значком J:\SPbGUTBonch\OOPLabs\OOPCourseWork\Отчет\Icons\openfile.png на панели инструментов. Если текущая база данных не была сохранена, то программа откроет предупреждающий диалог. После чего откроется диалоговое окно с выбором файла. После выбора файла в таблице отобразиться его содержимое.



Предупреждающий диалог

* + 1. Сохранить базу данных

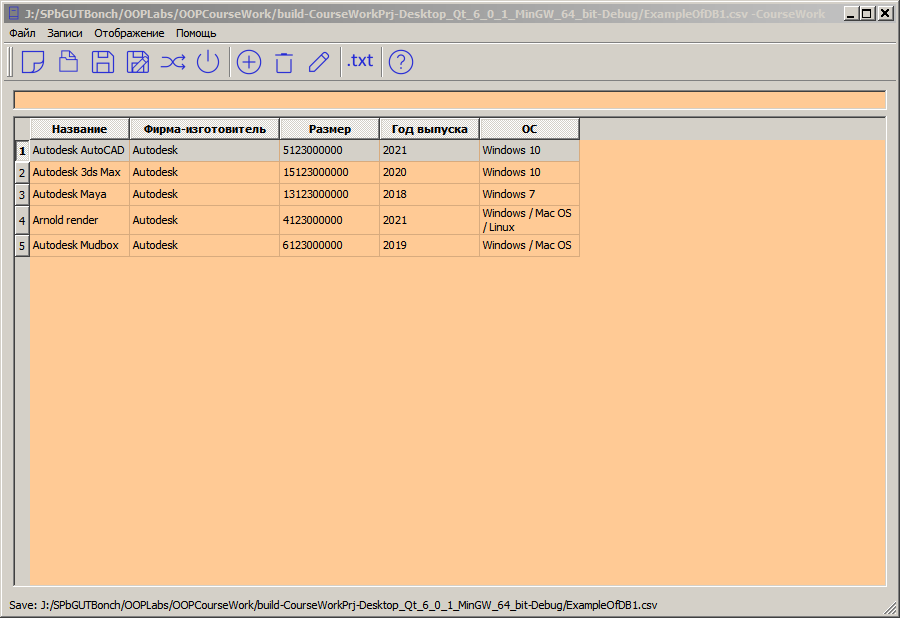
Чтобы сохранить базу данных нужно выбрать пункт меню [Файл]->[Сохранить] или [Файл]->[Сохранить как], либо воспользоваться значкамиJ:\SPbGUTBonch\OOPLabs\OOPCourseWork\Отчет\Icons\save_121760.png J:\SPbGUTBonch\OOPLabs\OOPCourseWork\Отчет\Icons\saveAs+.pngна панели инструментов. Если выбрано действие «Сохранить как» либо если база данных ещё не была сохранена после создания, то откроется диалоговое окно сохранения. После выбора имени файла база данных будет сохранена. При сохранении изменений зведочка в заголовке окна исчезает.



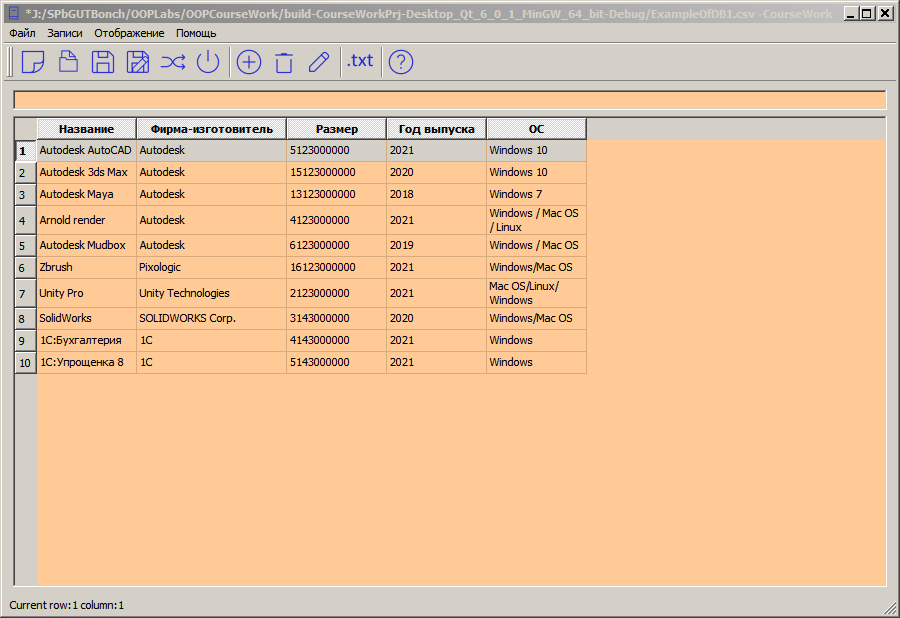
Диалоговое окно сохранения

* + 1. Объединить базу данных

Чтобы объединить базу данных с базой данных из файла, достаточно выбрать пункт меню [Файл]->[Объединить БД] либо воспользоваться значком J:\SPbGUTBonch\OOPLabs\OOPCourseWork\Отчет\Icons\shuffle_121752.pngна панели инструментов. После выбора файла в диалоговом окне открытия файла в конец таблицы будут добавлены содержащиеся в нем записи.



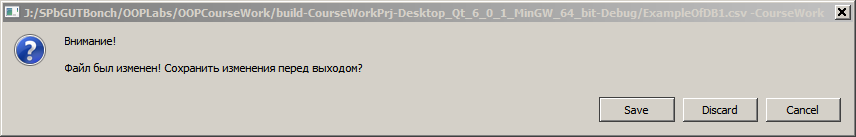
До объединения



После объединения

* + 1. Выход

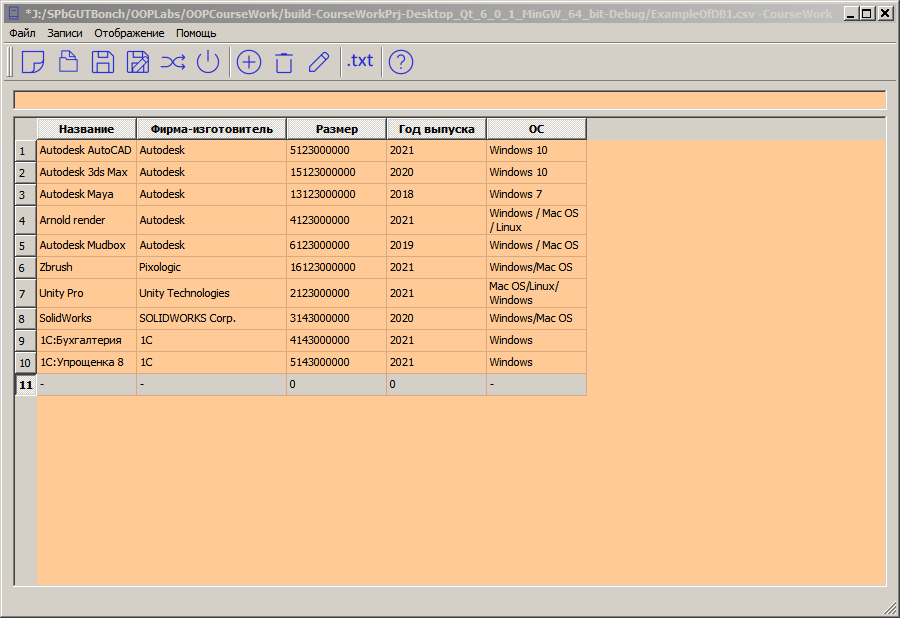
Для выхода нужно выбрать или пункт меню [Файл]->[Выход], или нажать на значокJ:\SPbGUTBonch\OOPLabs\OOPCourseWork\Отчет\Icons\power_121775.png на панели инструментов, или нажать на крестик [X] в заголовке окна. Если файл не был сохранен, то программа откроет предупреждающий диалог.



Предупреждающий диалог

* 1. Работа с записями
     1. Добавить новую запись

Чтобы добавить новую запись можно выбрать пункт меню [Записи]->[Добавить], либо воспользоваться значком J:\SPbGUTBonch\OOPLabs\OOPCourseWork\Отчет\Icons\add_121935.png на панели инструментов, либо воспользоваться контекстным меню. После этого в конце таблицы появится новая пустая запись.



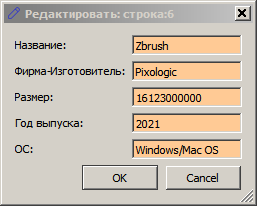
Новая запись в строке 11 таблицы.

* + 1. Удалить запись

Для удаления записи нужно выбрать пункт меню [Записи]->[Удалить], либо воспользоваться значком J:\SPbGUTBonch\OOPLabs\OOPCourseWork\Отчет\Icons\bin_121907.png на панели инструментов, либо воспользоваться контекстным меню. После этого запись будет удалена из базы данных.

* + 1. Редактировать запись

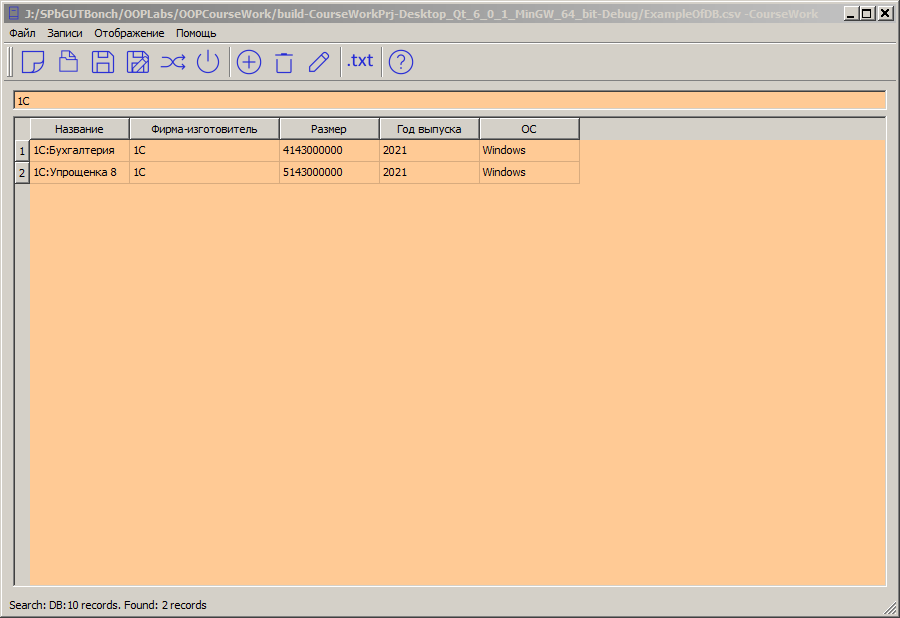
Для редактирования записи можно выбрать пункт меню [Записи]->[Редактировать], либо воспользоваться значком J:\SPbGUTBonch\OOPLabs\OOPCourseWork\Отчет\Icons\edit_121852.png на панели инструментов, либо воспользоваться контекстным меню. После этого запись будет открыто диалоговое окно, где можно будет изменить поля записи. Также редактировать запись можно двойным щелчком по ячейкам таблицы.



Диалоговое окно редактирования

* + 1. Поиск по базе данных

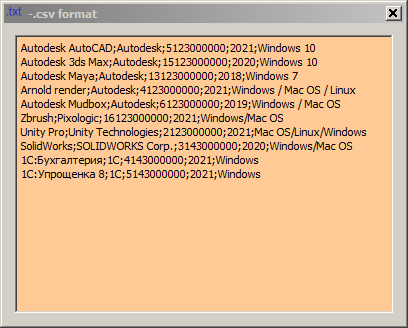
Для поиска по базе данных необходимо ввести запрос в текстовое поле под панелью инструментов, после чего программа отобразит результаты в таблице без дополнительных действий пользователя. Для очистки запроса нужно просто очистить текстовое поле.



Результаты поиска отобразились в таблице

* 1. Отображение
     1. Показать в текстовом виде (CSV)

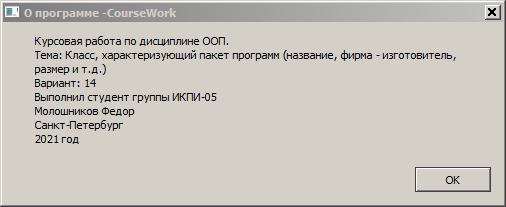
Для просмотра содержимого базы данных в текстовом виде нажмите [Отображение]->[Показать в текстовом виде], либо нажать на значок J:\SPbGUTBonch\OOPLabs\OOPCourseWork\Отчет\Icons\txt.png на панели инструментов, после чего можно будет видеть диалоговое окно с базой данных в формате CSV.



Диалоговое окно с базой данных в формате CSV

* 1. Помощь
     1. О программе

Для просмотра информации о программе выберете пункт [Помощь]->[О программе], либо нажмите на значок J:\SPbGUTBonch\OOPLabs\OOPCourseWork\Отчет\Icons\help_121828.png на панели инструментов. Будет открыто диалоговое окно(рис.15)



Диалоговое окно

* 1. Список горячих клавиш

|  |  |
| --- | --- |
| Создать | Ctrl+N |
| Открыть | Ctrl+O |
| Сохранить | Ctrl+S |
| Сохранить как | Ctrl+Shift+S |
| Объединить БД | Ctrl+U |
| Выход | Ctrl+X |
| Добавить | Ctrl+Shift+N |
| Удалить | Del |
| Редактировать | Ctrl+E |
| Показать в текстовом виде | Ctrl+T |
| О программе | F1 |

1. Заключение
2. Получены навыки работы с библиотекой Qt в среде Qt Creator.
3. Получен опыт создания приложения с оконным интерфейсом.
4. Получен опыт создания шаблонного класса контейнера типа vector
5. Программа обеспечивает выполнение операций согласно условиям задания.
6. Список литературы
7. Коробов С. А. Лекции по дисциплине Объектно–ориентированное программирование.
8. <https://doc.qt.io/> - Справочные материалы Qt
9. Приложение
   1. Vector.h

#ifndef VECTOR\_H\_INCLUDED

#define VECTOR\_H\_INCLUDED

// Vector.h==================================

#include <utility> //для вариативного шаблона

#include <iostream>

template <typename T>

class Vector

{

private:

T\* data;

size\_t size;

size\_t capasity;

public:

/\* Итератор\*/

class Iterator

{

private:

Vector\* v;

size\_t i;

public:

/\* Конструкторы по умолчанию, с параметром, копирования\*/

Iterator();

Iterator(Vector& r);

Iterator(const Iterator& r) = default; //конструктор копирования будет создан компилятором автоматически.

/\* Перегрузка операторов присваивания, инкрементов, сложения\_вычитания, сравнения \*/

Iterator& operator=(const Iterator& r) = default; //оператор присваивания будет перегружен компилятором автоматически.

Iterator& operator++(); //prefix

Iterator operator++(int);//postfix

Iterator operator+(size\_t i)const; //константные методы size\_t беззнаковый тип предназначенный для хранения размера массива, индексации массивов и для адресной арифметики. Возвращаемое значение ф-ии sizeof() имеет такой тип.

Iterator operator-(size\_t i)const;

bool operator==(const Iterator& r);

bool operator!=(const Iterator& r);

T& operator\*()const;

friend Vector; //чтобы класс Vector имел доступ к закрытым членам класса Iterator. Обращу внимание, что Vector только включает в себя класс Iterator, но не наследует его, поэтому protected здесь не сработает.

};

/\* Константный итератор\*/

class ConstIterator //для временных объектов и константного контейнера.

{

private:

const Vector\* v;

size\_t i;

public:

/\* Тут все то же самое, что и для обычного итератора, кроме конструктора с параметром и типа возвращаемого значения при разыменовании (и const Vector\* v; т.е. указателем на константный вектор). Нужен для работы с константным контейнером\*/

ConstIterator();

ConstIterator(const Vector& r);

ConstIterator(const ConstIterator& r) = default;

ConstIterator& operator=(const ConstIterator& r) = default;

ConstIterator& operator++(); //prefix

ConstIterator operator++(int);//postfix

ConstIterator operator+(size\_t i)const;

ConstIterator operator-(size\_t i)const;

bool operator==(const ConstIterator& r);

bool operator!=(const ConstIterator& r);

const T& operator\*()const;

friend Vector;

};

/\* Конструкторы по-умолчанию, копирования, перемещения. Деструктор\*/

Vector();

Vector(const Vector& r);

Vector(Vector&& r) noexcept; //noexcept спецификация исключения. Т.е. сообщаем компилятору, что функция не будет выбрасывать исключения. Нам это малополезно. Это конструктор перемещения. Принимает ссылку на rvalue. Эта ссылка позволяет получать доступ к неконстантным временным объектам. ссылка lvalue на временный объект может быть только константной и не позволяет изменять временный объект. термин "константная ссылка" следует понимать как "ссылка на константу". Этот термин - этакая идиома, поскольку ссылку в любом случае нельзя изменять, т.е. она не может быть не константной.

~Vector();

void pushBack(const T& el); //помещаем в конец вектора элемент el путем вызова конструктора копирования (а именно, с помощью Placement new)

template <typename ...Args> //variadic temlate (начиная с C++11) вариативный шаблон или шаблон с переменным числом аргументов. где ...Args - пакет типов, args - пакет параметров, типы которых задаются пакетом типов Args. По сути, этот метод принимает набор аргументов, из которых можно построить объект el, этими аргументами могут являтся отсутствие аргументов, тогда сработает конструктор по умолчанию, аргументы конструктора с параметрами или готовый объект. Во всех случаях метод будет использовать семантику перемещения(с этой целью здесь и берется ссылка на rvalue для каждого аргумента из пакета параметров). Т.е. после его работы исходные параметры могут стать недействительными.

void emplaceBack(Args&&... args);

void reserve(size\_t new\_capasity); //Тут выделяем новый кусок памяти под data, если new\_capasity больше capasity. При этом мы используем семантику перемещения. Тут важно, чтобы конструктор перемещения класса T ставил nullptr там, где это необходимо. Иначе этот метод во время вызова деструктора может освободить динамическую память, которая уже передана в пользование другому объекту.

void erase(size\_t begin, size\_t end); //стираем элементы с begin включительно по end (не включая end).

void clear(); //вызывает деструкторы всех элементов, оставляя прежнюю вместимость.

size\_t getSize() const;

Vector& operator=(const Vector& r);

Iterator begin();

Iterator end();

ConstIterator begin()const; //константные методы. Перегрузка 2х методов выше по константности. Для const Vector<someclass> vector1 будет вызываться этот метод, а для Vector<someclass> vector2 будет вызываться метод выше. "выбор делается в соответствии с частью "Best viable function" ( 13.3.3 ) стандарта C++, в которой, короче говоря, говорится, что лучшая функция-это та, которая требует наименьшего количества преобразований для своих аргументов". На практике, можно было бы обойтись без 2х векторов сверху. Однако считается, что каждый метод должен быть в 4х экземплярах foo();foo() const;foo() volatile;foo() const volatile;

ConstIterator end()const;

ConstIterator cbegin()const; //А эти методы полностью совпадают с 2мя методами выше.

ConstIterator cend()const;

T& operator[](size\_t i);

const T& operator[](size\_t i) const; //для работы с const Vector<someclass> vector1

};

#include "Vector.inl"

#endif // VECTOR\_H\_INCLUDED

* 1. Vector.inl

#ifndef VECTOR\_INL\_INCLUDED

#define VECTOR\_INL\_INCLUDED

// Vector.inl==================================

//---Iterator---//

template<typename T>

inline Vector<T>::Iterator::Iterator(): v(nullptr), i(0) {}

template<typename T>

inline Vector<T>::Iterator::Iterator(Vector& r) : v(&r), i(0) {}

template<typename T>

inline typename Vector<T>::Iterator& Vector<T>::Iterator::operator++() //prefix

{

++i;

return \*this;

}

template<typename T>

typename Vector<T>::Iterator inline Vector<T>::Iterator::operator++(int)//postfix

{

auto t = \*this;

++i;

return t;

}

template<typename T>

typename Vector<T>::Iterator Vector<T>::Iterator::operator+(size\_t i)const

{

auto t = \*this;

t.i += i;

}

template<typename T>

typename Vector<T>::Iterator Vector<T>::Iterator::operator-(size\_t i)const

{

auto t = \*this;

t.i -= i;

}

template<typename T>

bool Vector<T>::Iterator::operator==(const Iterator& r)

{

return v == r.v && i == r.i;

}

template<typename T>

bool Vector<T>::Iterator::operator!=(const Iterator& r)

{

return v != r.v || i != r.i;

}

template<typename T>

inline T& Vector<T>::Iterator::operator\*() const

{

return v->operator[](i);

}

//---iterator---//

//---ConstIterator---//

template<typename T>

inline Vector<T>::ConstIterator::ConstIterator() :v(nullptr), i(0) {}

template<typename T>

Vector<T>::ConstIterator::ConstIterator(const Vector& r) : v(&r), i(0) {}

template<typename T>

inline typename Vector<T>::ConstIterator& Vector<T>::ConstIterator::operator++() //prefix

{

++i;

return \*this;

}

template<typename T>

typename Vector<T>::ConstIterator inline Vector<T>::ConstIterator::operator++(int)//postfix

{

auto t = \*this;

++i;

return t;

}

template<typename T>

inline typename Vector<T>::ConstIterator Vector<T>::ConstIterator::operator+(size\_t i)const

{

auto t = \*this;

t.i += i;

}

template<typename T>

inline typename Vector<T>::ConstIterator Vector<T>::ConstIterator::operator-(size\_t i)const

{

auto t = \*this;

t.i -= i;

}

template<typename T>

bool Vector<T>::ConstIterator::operator==(const ConstIterator& r)

{

return v == r.v && i == r.i;

}

template<typename T>

bool Vector<T>::ConstIterator::operator!=(const ConstIterator& r)

{

return v != r.v || i != r.i;

}

template<typename T>

inline const T& Vector<T>::ConstIterator::operator\*() const

{

return v->operator[](i);

}

//---iterator---//

//---vector---//

template<typename T>

inline Vector<T>::Vector():

data(static\_cast<T\*>(operator new[](10 \* sizeof(T)))), size(0), capasity(10) //static\_cast в отличие от () умеет приводить тиделает только преобразование арифметических типов (для арифметических типов) и преобразование указателей вверх-вниз по иерархии наследования (для указателей). Т.е. static\_cast помогает избегать ошибок, которые бы пропустил ()

{}

template<typename T>

inline Vector<T>::Vector(const Vector& r):

data(static\_cast<T\*>(operator new[](r.size \* sizeof(T)))), size(r.size), capasity(r.size)

{

for (size\_t i = 0; i < size; i++)

new (data + i) T(r.data[i]); // Так называемый "placement new". Позволяет отделить выделение динамической памяти для объектов от её инициализации, т.е. непосредственного конструирования объектов в выделенной памяти. Инициализация в си++ вообще не простой вопрос. Из других способов в данной ситуации приходят на ум только использование что-то вроде \*p=T(argument) , т.е. конструктора копирования, вызываемого дважды(если не учитывать оптимизацию). При Placement new конструктор копирования, по видимому, вызывается только 1 раз. Впрочем, конструктор копирования при Placement new вызывается в данном случае, но вообще может вызываться просто какой-нибудь конструктор с параметрами или без параметров.

}

template<typename T>

inline Vector<T>::Vector(Vector&& r) noexcept: //Конструктор перемещения

data(r.data), size(r.size), capasity(r.capasity) // Тут все логично: ресурсы передаем в пользование другому объекту, у старого объекта обнуляем указатель.

{

r.data = nullptr;

}

template<typename T>

inline Vector<T>::~Vector()

{

if (data == nullptr)

return;

for (size\_t i = 0; i < size; i++)

data[i].~T(); // Необходимо вручную вызывать деструкторы, поскольку при выделении памяти был использован placement new, поэтому компилятор ничего не знает о том, где именно в памяти расположены объекты и не вызывает для них деструкторов.

//Because the memory allocation process did not happen through operator new, delete cannot deallocate the memory.

operator delete[](data); //И потом уже вызвать сам оператор delete

}

template<typename T>

void inline Vector<T>::pushBack(const T& el)

{

if (size == capasity)

reserve(size \* 2);

new (data + size) T(el);

size++;

}

template<typename T>

template<typename ...Args>

void inline Vector<T>::emplaceBack(Args&&... args)

{

if (size == capasity)

reserve(size \* 2);

new (data + size) T(std::move(args ...));

size++;

}

template<typename T>

void inline Vector<T>::reserve(size\_t new\_capasity)

{

if (this->capasity >= new\_capasity)

return;

T\* temp\_data = static\_cast<T\*>(operator new[](new\_capasity \* sizeof(T)));

for (size\_t i = 0; i < size; i++)

new (temp\_data + i) T(std::move(data[i]));

for (size\_t i = 0; i < size; i++)

data[i].~T(); // странный момент. Вызывать деструктор после работы конструктора перемещения нужно, если конструктор перемещения передал в пользование новому объекту не всю динамическую память, которой владел объект, а часть скопировал в новый участок памяти.

operator delete[](data);

data = temp\_data;

capasity = new\_capasity;

}

template<typename T>

void inline Vector<T>::erase(size\_t begin, size\_t end)

{

for (size\_t i = begin; i != end; i++)

{

data[i].~T();

}

size -= end - begin;

for (size\_t i = begin; i < size; i++)

{

new (data + i) T(std::move(data[i + end - begin]));

data[i + end - begin].~T();

}

}

template<typename T>

void inline Vector<T>::clear()

{

for (size\_t i = 0; i < size; i++)

data[i].~T();

size = 0;

}

template<typename T>

inline size\_t Vector<T>::getSize() const

{

return size;

}

template<typename T>

Vector<T>& Vector<T>::operator=(const Vector& r)

{

for (size\_t i = 0; i < size; i++)

data[i].~T();

if (capasity < r.size)

{

operator delete[](data);

data = static\_cast<T\*>(operator new[](r.size \* sizeof(T)));

capasity = r.size;

}

size = r.size;

for (size\_t i = 0; i < size; i++)

new (data + i) T(r.data[i]);

return \*this;

}

template<typename T>

typename Vector<T>::Iterator inline Vector<T>::begin(){ std::cout<<"BeginIterator"<<std::endl; return Iterator(\*this); } //typename для того, чтобы указать компилятору, что Vector<T>::Iterator это тип, а не статический член.

template<typename T>

typename Vector<T>::Iterator inline Vector<T>::end(){Iterator i; i.i = size; i.v = this; return i;};

template<typename T>

typename Vector<T>::ConstIterator inline Vector<T>::begin()const {std::cout<<"BeginConstIterator"<<std::endl; return ConstIterator(\*this); }; //для работы с константным вектором

template<typename T>

typename Vector<T>::ConstIterator inline Vector<T>::end()const { ConstIterator i; i.i = size; i.v = this; return i; };

template<typename T>

typename Vector<T>::ConstIterator inline Vector<T>::cbegin()const {std::cout<<"CBeginConstIterator"<<std::endl; return ConstIterator(\*this); };

template<typename T>

typename Vector<T>::ConstIterator inline Vector<T>::cend()const { ConstIterator i; i.i = size; i.v = this; return i; };

template<typename T>

T& Vector<T>::operator[](size\_t i)

{

return data[i];

}

template<typename T>

const T& Vector<T>::operator[](size\_t i) const

{

return data[i];

}

//---vector---//

#endif // VECTOR\_INL\_INCLUDED

* 1. ProgramCharact.h

#ifndef PROGRAMCHARACT\_H

#define PROGRAMCHARACT\_H

#include <qstring.h>

class ProgramCharact

{

private:

QString Name="-";

QString Manufacturer="-";

long long Size=0;

int Year=0;

QString OS="-";

public:

/\* Конструкторы и деструктор\*/

ProgramCharact();

ProgramCharact(QString NName, QString NManufacture, long long NSize, int NYear, QString NOS);

/\* Getters Setters\*/

QString GetManufacturer();

long long GetSize();

QString GetName();

int GetYear();

QString GetOS();

void SetManufacturer(QString NManufacturer);

void SetSize(long long NSize);

void SetName(QString NName);

void SetYear(int NYear);

void SetOS(QString NOS);

};

#endif // PROGRAMCHARACT\_H

* 1. ProgramCharact.cpp

#include "ProgramCharact.h"

/\* Конструкторы и деструктор\*/

ProgramCharact::ProgramCharact(){}

ProgramCharact::ProgramCharact(QString NName, QString NManufacturer, long long NSize, int NYear, QString NOS)

:Name(NName), Manufacturer(NManufacturer), Size(NSize), Year(NYear), OS(NOS)

{

}

/\* Getters Setters\*/

QString ProgramCharact::GetManufacturer(){return Manufacturer;}

long long ProgramCharact::GetSize(){return Size;}

QString ProgramCharact::GetName(){return Name;}

int ProgramCharact::GetYear(){return Year;}

QString ProgramCharact::GetOS(){return OS;}

void ProgramCharact::SetManufacturer(QString NManufacturer){Manufacturer=NManufacturer;}

void ProgramCharact::SetSize(long long NSize){Size=NSize;}

void ProgramCharact::SetName(QString NName){Name=NName;}

void ProgramCharact::SetYear(int NYear){Year=NYear;}

void ProgramCharact::SetOS(QString NOS){OS=NOS;}

* 1. mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QFileDialog>

#include <QFileInfo> //для проверки файлов на то, что они существуют

#include <QMessageBox>

#include <QMouseEvent> //для крестика

//#include <iostream>

#include "Vector.h"

#include "ProgramCharact.h"

#include "showtxtdialog.h"

#include "editdialog.h"

#define FIELDCOUNT 5 //Задаем число полей

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_Create\_triggered();

void on\_Open\_triggered();

void on\_Save\_triggered();

void on\_SaveAs\_triggered();

void on\_AddDB\_triggered();

void on\_Quit\_triggered();

void on\_AddRecord\_triggered();

void on\_DeleteRecord\_triggered();

void on\_Edit\_triggered();

void on\_ShowTxt\_triggered();

void on\_About\_triggered();

//для меню

void ShowContextMenu(const QPoint&);

/\* Для определения текущей выбранной строки\*/

// void TlSlot(QModelIndex, QModelIndex);

// void on\_tableWidget\_cellEntered(int row, int column);

void on\_tableWidget\_cellChanged(int row, int column);

void on\_LineEditSearch\_textEdited(const QString &arg1);

void on\_tableWidget\_cellClicked(int row, int column);

private:

Ui::MainWindow \*ui;

Vector<ProgramCharact> VectorDB;

Vector<size\_t> SearchRelationVector; //вектор показывающий отношение между VectorDB и тем, что нашлось в поиске

QString FileName="Untitled"; //путь с именем к текущему файлу таблицы

bool IsFilePathOpen=false; //флаг того, что открыт какойто файл. диалог getSaveFileName при нажатии cansel возвращает пустую строку, поэтому мы не знаем что произошло нажад ли пользователь отмена или мы разорвали связь с именем файла

bool IsEditedFlag=false; //флаг того, что документ редактировали.

// void TableSelectionConnect();

// Методы для работы с таблицей и вектором=================================

// void TableToVector(); //рабочие методы без учета поиска

// void VectorToTable(); //рабочие методы без учета поиска

void TableToVectorSearch(); //то же самое, только с учетом поиска

void VectorToTableSearch(); //то же самое, только с учетом поиска

void UpdateRelation(); //вспомогательная функция, обновляет SearchRelationVector

// void OnDocumentEdit(); //функция, вызываемая после того, как документ был отредактирован

void closeEvent(QCloseEvent \*event); /\* Перехват сообщения закрытия программы \*/

};

#endif // MAINWINDOW\_H

* 1. mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

/\* Подготавливаем таблицу\*/

// ui->tableWidget->setRowCount(10);

this->setWindowTitle(QString(FileName).append(" -CourseWork"));

ui->tableWidget->setColumnCount(5);

ui->tableWidget->setHorizontalHeaderLabels(QStringList() << "Название" << "Фирма-изготовитель" << "Размер" << "Год выпуска" << "ОС");

ui->tableWidget->setColumnWidth(1, 150); //подгоняем ширину столбца "Фирма-изготовитель"

// TableSelectionConnect();

ui->tableWidget->setContextMenuPolicy(Qt::CustomContextMenu);

connect(ui->tableWidget, SIGNAL(customContextMenuRequested(const QPoint&)), this, SLOT(ShowContextMenu(const QPoint&)));

}

MainWindow::~MainWindow()

{

delete ui;

}

/\* Слоты для сигналов от кнопок меню\*/

void MainWindow::on\_Create\_triggered()

{

ui->statusbar->showMessage("Create");

if(IsEditedFlag)

{

QMessageBox msgBox1;

msgBox1.setIcon(QMessageBox::Question);

msgBox1.setWindowTitle(QString(FileName).append(" -CourseWork"));

msgBox1.setText("Внимание!");

msgBox1.setInformativeText("Файл был изменен! Сохранить изменения перед открытием нового файла?");

msgBox1.setStandardButtons(QMessageBox::Save | QMessageBox::Discard | QMessageBox::Cancel);

msgBox1.setDefaultButton(QMessageBox::Save);

// QMessageBox::StandardButton reply;

// reply=QMessageBox::question(this, "Открыть", "Файл был изменен. Все изменения будут утеряны. Продолжить?", QMessageBox::StandardButtons(Yes|No|Cancel));

int reply=msgBox1.exec();

switch(reply)

{

case QMessageBox::Save:

on\_Save\_triggered();

if(!IsEditedFlag) //проверяем, что мы действительно сохранили, а не нажали отмену в диалоге сохранения

{

FileName="Untitled";

this->setWindowTitle(QString(FileName).append(" -CourseWork"));

IsFilePathOpen=false; //устанавливаем флаг

VectorDB.clear();

UpdateRelation();

//VectorToTable();

VectorToTableSearch();

IsEditedFlag=false;

this->setWindowTitle(QString("").append(FileName).append(" -CourseWork"));

}

break;

case QMessageBox::Discard:

FileName="Untitled";

this->setWindowTitle(QString(FileName).append(" -CourseWork"));

IsFilePathOpen=false; //устанавливаем флаг

VectorDB.clear();

UpdateRelation();

//VectorToTable();

VectorToTableSearch();

IsEditedFlag=false;

this->setWindowTitle(QString("").append(FileName).append(" -CourseWork"));

break;

case QMessageBox::Cancel:

break;

}

/\* if(reply==QMessageBox::Yes)

{

FileName="Untitled";

this->setWindowTitle(QString(FileName).append(" -CourseWork"));

IsFilePathOpen=false; //устанавливаем флаг

VectorDB.clear();

UpdateRelation();

//VectorToTable();

VectorToTableSearch();

IsEditedFlag=false;

this->setWindowTitle(QString("").append(FileName).append(" -CourseWork"));

}\*/

}

else

{

FileName="Untitled";

this->setWindowTitle(QString(FileName).append(" -CourseWork"));

IsFilePathOpen=false; //устанавливаем флаг

VectorDB.clear();

UpdateRelation();

//VectorToTable();

VectorToTableSearch();

IsEditedFlag=false;

this->setWindowTitle(QString("").append(FileName).append(" -CourseWork"));

}

}

void MainWindow::on\_Open\_triggered()

{

ui->statusbar->showMessage("Open");

if(IsEditedFlag)

{

QMessageBox msgBox1;

msgBox1.setIcon(QMessageBox::Question);

msgBox1.setWindowTitle(QString(FileName).append(" -CourseWork"));

msgBox1.setText("Внимание!");

msgBox1.setInformativeText("Файл был изменен! Сохранить изменения перед открытием нового файла?");

msgBox1.setStandardButtons(QMessageBox::Save | QMessageBox::Discard | QMessageBox::Cancel);

msgBox1.setDefaultButton(QMessageBox::Save);

// QMessageBox::StandardButton reply;

// reply=QMessageBox::question(this, "Открыть", "Файл был изменен. Все изменения будут утеряны. Продолжить?");

// if(reply==QMessageBox::Yes)

int reply=msgBox1.exec();

switch(reply)

{

case QMessageBox::Save:

on\_Save\_triggered();

if(!IsEditedFlag) //проверяем, что мы действительно сохранили, а не нажали отмену в диалоге сохранения

{

QString TempFileName=QFileDialog::getOpenFileName(this, "Открыть файл...", ".", "csv files (\*.csv);;All files (\*.\*)"); //в случае неудачи не обновляем Filename

QFileInfo CheckFile(TempFileName);

if (CheckFile.exists() && CheckFile.isFile()) //проверяем, что файл существует и не является директорией

{

QFile File(TempFileName);

if (File.open(QIODevice::ReadOnly)) //открываем файл для чтения

{

/\* очищаем вектор обновляем таблицу \*/

VectorDB.clear();

UpdateRelation();

VectorToTableSearch();

/\* \*/

QTextStream fin(&File); //создаем поток

while (!fin.atEnd())

{

QString line = fin.readLine();

QStringList fields = line.split(";"); //line.split(";") возвращает список, состоящий из строк, разделенных ;

VectorDB.pushBack(ProgramCharact(fields[0], fields[1], fields[2].toLongLong(), fields[3].toInt(), fields[4]));

// ui->statusbar->showMessage("123");

}

FileName=TempFileName;

this->setWindowTitle(QString(FileName).append(" -CourseWork"));

File.close();

IsFilePathOpen=true; //устанавливаем флаг

UpdateRelation();

VectorToTableSearch();

IsEditedFlag=false;

this->setWindowTitle(QString("").append(FileName).append(" -CourseWork"));

}

}

else

{

QMessageBox msgBox;

msgBox.setText("Файл не существует или выбрана директория!");

}

}

break;

case QMessageBox::Discard:

QString TempFileName=QFileDialog::getOpenFileName(this, "Открыть файл...", ".", "csv files (\*.csv);;All files (\*.\*)"); //в случае неудачи не обновляем Filename

QFileInfo CheckFile(TempFileName);

if (CheckFile.exists() && CheckFile.isFile()) //проверяем, что файл существует и не является директорией

{

QFile File(TempFileName);

if (File.open(QIODevice::ReadOnly)) //открываем файл для чтения

{

/\* очищаем вектор обновляем таблицу \*/

VectorDB.clear();

UpdateRelation();

VectorToTableSearch();

/\* \*/

QTextStream fin(&File); //создаем поток

while (!fin.atEnd())

{

QString line = fin.readLine();

QStringList fields = line.split(";"); //line.split(";") возвращает список, состоящий из строк, разделенных ;

VectorDB.pushBack(ProgramCharact(fields[0], fields[1], fields[2].toLongLong(), fields[3].toInt(), fields[4]));

// ui->statusbar->showMessage("123");

}

FileName=TempFileName;

this->setWindowTitle(QString(FileName).append(" -CourseWork"));

File.close();

IsFilePathOpen=true; //устанавливаем флаг

UpdateRelation();

VectorToTableSearch();

IsEditedFlag=false;

this->setWindowTitle(QString("").append(FileName).append(" -CourseWork"));

}

}

else

{

QMessageBox msgBox;

msgBox.setText("Файл не существует или выбрана директория!");

}

break;

}

}

else

{

QString TempFileName=QFileDialog::getOpenFileName(this, "Открыть файл...", ".", "csv files (\*.csv);;All files (\*.\*)"); //в случае неудачи не обновляем Filename

QFileInfo CheckFile(TempFileName);

if (CheckFile.exists() && CheckFile.isFile()) //проверяем, что файл существует и не является директорией

{

QFile File(TempFileName);

if (File.open(QIODevice::ReadOnly)) //открываем файл для чтения

{

/\* очищаем вектор обновляем таблицу \*/

VectorDB.clear();

UpdateRelation();

VectorToTableSearch();

/\* \*/

QTextStream fin(&File); //создаем поток

while (!fin.atEnd())

{

QString line = fin.readLine();

QStringList fields = line.split(";"); //line.split(";") возвращает список, состоящий из строк, разделенных ;

VectorDB.pushBack(ProgramCharact(fields[0], fields[1], fields[2].toLongLong(), fields[3].toInt(), fields[4]));

// ui->statusbar->showMessage("123");

}

FileName=TempFileName;

this->setWindowTitle(QString(FileName).append(" -CourseWork"));

File.close();

IsFilePathOpen=true; //устанавливаем флаг

UpdateRelation();

VectorToTableSearch();

IsEditedFlag=false;

this->setWindowTitle(QString("").append(FileName).append(" -CourseWork"));

}

}

else

{

QMessageBox msgBox;

msgBox.setText("Файл не существует или выбрана директория!");

}

}

}

void MainWindow::on\_Save\_triggered()

{

ui->statusbar->showMessage(QString("Save: ").append(FileName));

TableToVectorSearch();

UpdateRelation();

int TempCurrentRow=ui->tableWidget->currentRow();

VectorToTableSearch();

ui->tableWidget->setCurrentIndex(ui->tableWidget->model()->index(TempCurrentRow, 0));

if(IsFilePathOpen) //Если мы уже имеем связь с файлом

{

QFile File(FileName);

if (File.open(QIODevice::WriteOnly|QIODevice::Text)) //открываем файл для записи. QIODevice::Text для локального окончания строк. Т.е. вместо "\n" в потоке в windows будет "\r\n"

{

QTextStream fout(&File);

for(auto &iterator1: VectorDB) //Пример использования итератора. auto определяет тип iterator1 по значению, которым iterator1 инициализируется. использование ссылки позволяет избежать копирования в переменную iterator1. Поскольку Vector<T> имеет функции begin() end(), то их вызовы и определят границы диапазона для цикла for range-based.

{

fout << iterator1.GetName() << ";" << iterator1.GetManufacturer() <<";"<< iterator1.GetSize()<< ";" << iterator1.GetYear()<< ";" << iterator1.GetOS()<< "\n";

}

IsEditedFlag=false;

this->setWindowTitle(QString("").append(FileName).append(" -CourseWork"));

}

}

else

{

on\_SaveAs\_triggered(); //если никакой файл не открыт, то сохраняем как.

}

}

void MainWindow::on\_SaveAs\_triggered()

{

ui->statusbar->showMessage("Save As");

QString TempFileName=QFileDialog::getSaveFileName(this, "Сохранить файл как", ".", "csv files (\*.csv);; All files (\*.\*)");

TableToVectorSearch();

UpdateRelation();

int TempCurrentRow=ui->tableWidget->currentRow();

VectorToTableSearch();

ui->tableWidget->setCurrentIndex(ui->tableWidget->model()->index(TempCurrentRow, 0));

if(TempFileName!="") //если не нажали на cansel

{

// QFileInfo CheckFile(FileName);

// if (CheckFile.exists() && CheckFile.isFile())

// {

QFile File(TempFileName);

if (File.open(QIODevice::WriteOnly|QIODevice::Text)) //открываем файл для записи. QIODevice::Text для локального окончания строк. Т.е. вместо "\n" в потоке в windows будет "\r\n"

{

QTextStream fout(&File);

size\_t i;

for(i=0;i<VectorDB.getSize(); i++)

{

fout << VectorDB[i].GetName() << ";" << VectorDB[i].GetManufacturer() <<";"<< VectorDB[i].GetSize()<< ";" <<VectorDB[i].GetYear()<< ";" <<VectorDB[i].GetOS()<< "\n";

}

FileName=TempFileName;

this->setWindowTitle(QString(FileName).append(" -CourseWork"));

IsFilePathOpen=true; //устанавливаем флаг

IsEditedFlag=false;

this->setWindowTitle(QString("").append(FileName).append(" -CourseWork"));

}

// }

}

}

void MainWindow::on\_AddDB\_triggered()

{

ui->statusbar->showMessage("AddDB");

TableToVectorSearch();

QString TempFileName=QFileDialog::getOpenFileName(this, "Открыть для добавления...", ".", "csv files (\*.csv);;All files (\*.\*)"); //Оставляем Filename прежним

QFileInfo CheckFile(TempFileName);

if (CheckFile.exists() && CheckFile.isFile()) //проверяем, что файл существует и не является директорией

{

QFile File(TempFileName);

if (File.open(QIODevice::ReadOnly)) //открываем файл для чтения

{

QTextStream fin(&File); //создаем поток

while (!fin.atEnd())

{

QString line = fin.readLine();

QStringList fields = line.split(";"); //line.split(";") возвращает список, состоящий из строк, разделенных ;

VectorDB.pushBack(ProgramCharact(fields[0], fields[1], fields[2].toLongLong(), fields[3].toInt(), fields[4]));

// ui->statusbar->showMessage("123");

}

// FileName=TempFileName;

// this->setWindowTitle(QString(FileName).append(" -CourseWork"));

File.close();

// IsFilePathOpen=true; //устанавливаем флаг

}

}

else

{

QMessageBox msgBox;

msgBox.setText("Файл не существует или выбрана директория!");

}

UpdateRelation();

VectorToTableSearch();

}

void MainWindow::on\_Quit\_triggered()

{

ui->statusbar->showMessage("Quit");

// QCloseEvent event;

// closeEvent(&event);

close(); //строчка closeEvent(&event); не работает. вместо этого вызываем метод close();

// QApplication::quit(); тоже можно использовать. вызовет также closeEvent(&event);

/\* if(IsEditedFlag)

{

QMessageBox msgBox1;

msgBox1.setIcon(QMessageBox::Question);

msgBox1.setWindowTitle(QString(FileName).append(" -CourseWork"));

msgBox1.setText("Внимание!");

msgBox1.setInformativeText("Файл был изменен! Сохранить изменения перед выходом?");

msgBox1.setStandardButtons(QMessageBox::Save | QMessageBox::Discard | QMessageBox::Cancel);

msgBox1.setDefaultButton(QMessageBox::Save);

//QMessageBox::StandardButton reply;

int reply=msgBox1.exec();

switch(reply)

{

case QMessageBox::Save:

on\_Save\_triggered();

if(!IsEditedFlag) //проверяем, что мы действительно сохранили, а не нажали отмену в диалоге сохранения

{

QApplication::quit();

// QMainWindow::closeEvent();

}

break;

case QMessageBox::Discard:

QApplication::quit();

break;

}

// reply=QMessageBox::question(this, "Открыть", "Файл был изменен. Все изменения будут утеряны. Продолжить?");

// if(reply==QMessageBox::Yes) QApplication::quit();

}

else

{

QApplication::quit();

}\*/

}

void MainWindow::closeEvent(QCloseEvent \*event) /\* Перехват сообщения закрытия программы \*/

{

/\* QMessageBox::StandardButton reply;

if(IsEditedFlag)

{

reply=QMessageBox::question(this, "Открыть", "Файл был изменен. Все изменения будут утеряны. Продолжить?");

if(reply==QMessageBox::No) event->ignore();

}

// on\_Quit\_triggered(); \*/

if(IsEditedFlag)

{

QMessageBox msgBox1;

msgBox1.setIcon(QMessageBox::Question);

msgBox1.setWindowTitle(QString(FileName).append(" -CourseWork"));

msgBox1.setText("Внимание!");

msgBox1.setInformativeText("Файл был изменен! Сохранить изменения перед выходом?");

msgBox1.setStandardButtons(QMessageBox::Save | QMessageBox::Discard | QMessageBox::Cancel);

msgBox1.setDefaultButton(QMessageBox::Save);

//QMessageBox::StandardButton reply;

int reply=msgBox1.exec();

switch(reply)

{

case QMessageBox::Save:

on\_Save\_triggered();

if(!IsEditedFlag) //проверяем, что мы действительно сохранили, а не нажали отмену в диалоге сохранения

{

// QApplication::quit();

event->accept();

}

else

{

event->ignore();

}

break;

case QMessageBox::Discard:

event->accept();

QMainWindow::closeEvent(event);

// QApplication::quit();

// event->accept();

// QApplication::quit();

break;

case QMessageBox::Cancel:

event->ignore();

break;

}

// reply=QMessageBox::question(this, "Открыть", "Файл был изменен. Все изменения будут утеряны. Продолжить?");

// if(reply==QMessageBox::Yes) QApplication::quit();

}

else

{

QApplication::quit();

}

}

void MainWindow::on\_AddRecord\_triggered()

{

ui->statusbar->showMessage("AddRecord");

//TableToVector();

TableToVectorSearch();

VectorDB.pushBack(ProgramCharact());

UpdateRelation();

//VectorToTable();

VectorToTableSearch();

ui->tableWidget->scrollToBottom();

if(SearchRelationVector[SearchRelationVector.getSize()-1]==VectorDB.getSize()-1) ui->tableWidget->setCurrentIndex(ui->tableWidget->model()->index(SearchRelationVector.getSize()-1, 0)); //т.е. если добавленный элемент, который последний в векторе, прошел фильтр, то его выделяем.

// ui->statusbar->showMessage(QString::number(VectorDB[0].GetSize()));

// ui->statusbar->showMessage(VectorDB[0].GetName());

}

void MainWindow::on\_DeleteRecord\_triggered()

{

int TempCurrentRow=ui->tableWidget->currentRow();

if(ui->tableWidget->currentRow()!=-1) //после вызова VectorToTable пропадает выделение. Если ничего не выделено, то устанавливается -1

{

ui->statusbar->showMessage("DeleteRecord");

//TableToVector();

TableToVectorSearch();

VectorDB.erase(SearchRelationVector[ui->tableWidget->currentRow()], SearchRelationVector[ui->tableWidget->currentRow()]+1);

//VectorToTable();

UpdateRelation();

VectorToTableSearch();

// ui->statusbar->showMessage(QString("currentRow=").append(QString::number(ui->tableWidget->currentRow())));

//Выделяем следующую строку в таблице

if(TempCurrentRow==SearchRelationVector.getSize()) ui->tableWidget->setCurrentIndex(ui->tableWidget->model()->index(SearchRelationVector.getSize()-1, 0));

else ui->tableWidget->setCurrentIndex(ui->tableWidget->model()->index(TempCurrentRow, 0));

}

}

void MainWindow::on\_Edit\_triggered()

{

ui->statusbar->showMessage("Edit");

if(ui->tableWidget->currentRow()!=-1) //если есть выделение

{

EditDialog EditDialog1(ui->tableWidget->item(ui->tableWidget->currentRow(), 0)->text(), ui->tableWidget->item(ui->tableWidget->currentRow(), 1)->text(),ui->tableWidget->item(ui->tableWidget->currentRow(), 2)->text(),ui->tableWidget->item(ui->tableWidget->currentRow(), 3)->text(),ui->tableWidget->item(ui->tableWidget->currentRow(), 4)->text());

EditDialog1.adjustSize(); //вначале подгоняем размер, а потом фиксируем его

EditDialog1.setFixedHeight(EditDialog1.geometry().height());

EditDialog1.setWindowTitle(QString("Редактировать: строка:").append(QString::number(ui->tableWidget->currentRow()+1)));

if(EditDialog1.exec()==QDialog::Accepted)

{

ui->tableWidget->item(ui->tableWidget->currentRow(), 0)->setText(EditDialog1.getName());

ui->tableWidget->item(ui->tableWidget->currentRow(), 1)->setText(EditDialog1.getManufacturer());

ui->tableWidget->item(ui->tableWidget->currentRow(), 2)->setText(EditDialog1.getSize());

ui->tableWidget->item(ui->tableWidget->currentRow(), 3)->setText(EditDialog1.getYear());

ui->tableWidget->item(ui->tableWidget->currentRow(), 4)->setText(EditDialog1.getOS());

QModelIndex TempIndex=ui->tableWidget->currentIndex();

int TempRow=SearchRelationVector[TempIndex.row()];

TableToVectorSearch();

UpdateRelation();

VectorToTableSearch();

int i;

for(i=0; i<SearchRelationVector.getSize();i++)

{

if(SearchRelationVector[i]==TempRow)

{

TempIndex=ui->tableWidget->model()->index(i, TempIndex.column());

ui->tableWidget->setCurrentIndex(TempIndex);

break;

}

}

// ui->tableWidget->setCurrentIndex(TempIndex);

}

}

}

void MainWindow::on\_ShowTxt\_triggered() //окошко, где содержимое представленно в текстовом виде, соответствующем формату csv

{

ui->statusbar->showMessage("ShowTxt");

disconnect(ui->tableWidget, SIGNAL(cellChanged(int, int)), this, SLOT(on\_tableWidget\_cellChanged(int, int)));

QModelIndex TempIndex=ui->tableWidget->currentIndex();

TableToVectorSearch();

UpdateRelation();

VectorToTableSearch();

ui->tableWidget->setCurrentIndex(TempIndex);

ShowTxtDialog ShowTxtDialog1;

size\_t i;

QString StrCSV;

for(i=0; i<VectorDB.getSize(); i++)

{

StrCSV+=VectorDB[i].GetName()+";"+VectorDB[i].GetManufacturer()+";"+QString::number(VectorDB[i].GetSize())+";"+QString::number(VectorDB[i].GetYear())+";"+VectorDB[i].GetOS() + "\n";

}

ShowTxtDialog1.TextEditSetText(StrCSV);

ShowTxtDialog1.setWindowTitle(" -.csv format");

ShowTxtDialog1.exec();

connect(ui->tableWidget, SIGNAL(cellChanged(int, int)), this, SLOT(on\_tableWidget\_cellChanged(int, int)));

}

void MainWindow::on\_About\_triggered()

{

ui->statusbar->showMessage("About");

QMessageBox msgAbout;

// msgAbout.setIconPixmap(QPixmap(":/IconPrefix1/Icons/help\_121828.ico"));

msgAbout.setWindowTitle("О программе -CourseWork");

msgAbout.setText("Курсовая работа по дисциплине ООП.\n"

"Тема: Класс, характеризующий пакет программ (название, фирма - изготовитель, размер и т.д.)\n"

"Вариант: 14\n"

"Выполнил студент группы ИКПИ-05\n"

"Молошников Федор\n"

"Санкт-Петербург\n"

"2021 год");

msgAbout.exec();

}

/\* Слоты для сигналов от кнопок для поиска\*/

/\* Слоты для работы с выделением ячеек\*/

/\* void MainWindow::TlSlot(QModelIndex r, QModelIndex)

{

QString tempStr;

tempStr.append("row=");

tempStr.append(QString::number(r.row()));

tempStr.append(" column=");

// tempStr.append(QString::number(column));

ui->statusbar->showMessage(tempStr);

}

void MainWindow::TableSelectionConnect()

{

connect(ui->tableWidget->selectionModel(), SIGNAL(currentChanged(QModelIndex, QModelIndex)), this, SLOT(TlSlot(QModelIndex, QModelIndex)));

ui->statusbar->showMessage("Trying to connect");

} \*/

/\* void MainWindow::on\_tableWidget\_cellEntered(int row, int column)

{

TableSelectionConnect();

QString tempStr;

tempStr.append("row=");

tempStr.append(QString::number(row));

tempStr.append(" column=");

tempStr.append(QString::number(column));

ui->statusbar->showMessage(tempStr);

} \*/

/\* void MainWindow::TableToVector() //версия простая, без поиска

{

// VectorDB.pushBack(ProgramCharact()); //для отладки

size\_t CurrentSize=VectorDB.getSize();

ui->statusbar->showMessage(QString::number(CurrentSize).append("=CurrentSize"));

VectorDB.clear();

size\_t i;

for(i=0; i<CurrentSize; i++) //заполняем каждый объект вектора. с помощью конструктора с параметрами

{

VectorDB.pushBack(ProgramCharact(ui->tableWidget->item(i, 0)->text(), ui->tableWidget->item(i, 1)->text(), ui->tableWidget->item(i, 2)->text().toLongLong(), ui->tableWidget->item(i, 3)->text().toInt(), ui->tableWidget->item(i, 4)->text()));

}

} \*/

/\* void MainWindow::VectorToTable() //версия простая, без поиска

{

ui->tableWidget->clearContents(); //удаляет item ы из динамической памяти,а не просто скрывает их

ui->tableWidget->setRowCount(VectorDB.getSize());

ui->statusbar->showMessage(QString::number(VectorDB.getSize()).append("=VectorDB.getSize()"));

size\_t i;

for(i=0; i<VectorDB.getSize(); i++)

{

ui->tableWidget->setItem(i, 0, new QTableWidgetItem(VectorDB[i].GetName()));

ui->tableWidget->setItem(i, 1, new QTableWidgetItem(VectorDB[i].GetManufacturer()));

ui->tableWidget->setItem(i, 2, new QTableWidgetItem(QString::number(VectorDB[i].GetSize())));

ui->tableWidget->setItem(i, 3, new QTableWidgetItem(QString::number(VectorDB[i].GetYear())));

ui->tableWidget->setItem(i, 4, new QTableWidgetItem(VectorDB[i].GetOS()));

}

} \*/

void MainWindow::TableToVectorSearch()

{

// ui->statusbar->showMessage(QString::number(SearchRelationVector.getSize()).append("=CurrentSize"));

size\_t i;

for(i=0; i<SearchRelationVector.getSize(); i++) //заполняем каждый объект вектора. с помощью конструктора с параметрами

{

VectorDB[SearchRelationVector[i]].SetName(ui->tableWidget->item(i, 0)->text());

VectorDB[SearchRelationVector[i]].SetManufacturer(ui->tableWidget->item(i, 1)->text());

VectorDB[SearchRelationVector[i]].SetSize(ui->tableWidget->item(i, 2)->text().toLongLong());

VectorDB[SearchRelationVector[i]].SetYear(ui->tableWidget->item(i, 3)->text().toInt());

VectorDB[SearchRelationVector[i]].SetOS(ui->tableWidget->item(i, 4)->text());

}

}

void MainWindow::VectorToTableSearch() //версия для поиска

{

ui->tableWidget->clearContents(); //удаляет item ы из динамической памяти,а не просто скрывает их

ui->tableWidget->setRowCount(SearchRelationVector.getSize());

// ui->statusbar->showMessage(QString::number(SearchRelationVector.getSize()).append("=SearchRelationVector.getSize()"));

size\_t i;

for(i=0; i<SearchRelationVector.getSize(); i++)

{

ui->tableWidget->setItem(i, 0, new QTableWidgetItem(VectorDB[SearchRelationVector[i]].GetName()));

ui->tableWidget->setItem(i, 1, new QTableWidgetItem(VectorDB[SearchRelationVector[i]].GetManufacturer()));

ui->tableWidget->setItem(i, 2, new QTableWidgetItem(QString::number(VectorDB[SearchRelationVector[i]].GetSize())));

ui->tableWidget->setItem(i, 3, new QTableWidgetItem(QString::number(VectorDB[SearchRelationVector[i]].GetYear())));

ui->tableWidget->setItem(i, 4, new QTableWidgetItem(VectorDB[SearchRelationVector[i]].GetOS()));

}

ui->tableWidget->resizeRowsToContents();

}

void MainWindow::UpdateRelation()

{

SearchRelationVector.clear();

size\_t i;

for(i=0;i<VectorDB.getSize();i++) //заполняем вектор отношения

{

if(VectorDB[i].GetName().indexOf(ui->LineEditSearch->text())!=-1 || VectorDB[i].GetManufacturer().indexOf(ui->LineEditSearch->text())!=-1 || QString::number(VectorDB[i].GetSize()).indexOf(ui->LineEditSearch->text())!=-1 || QString::number(VectorDB[i].GetYear()).indexOf(ui->LineEditSearch->text())!=-1 || VectorDB[i].GetOS().indexOf(ui->LineEditSearch->text())!=-1)

{

// ui->statusbar->showMessage((ui->LineEditSearch->text()).append(" find!"));

SearchRelationVector.pushBack(i);

}

}

}

void MainWindow::on\_tableWidget\_cellChanged(int row, int column)

{

//Тут будем добавлять зведочку в заголовке

//this->setWindowTitle("\*Untitled"); //что-то подобное

IsEditedFlag=true;

this->setWindowTitle(QString("\*").append(FileName).append(" -CourseWork"));

}

void MainWindow::on\_LineEditSearch\_textEdited(const QString &arg1)

{

disconnect(ui->tableWidget, SIGNAL(cellChanged(int, int)), this, SLOT(on\_tableWidget\_cellChanged(int, int))); //отключаем и подключаем сигнал, чтобы при поиске не было такого, как будто таблица отредактирована

TableToVectorSearch(); //Тут можно по флагу изменения таблицы, когда он у нас будет.

UpdateRelation();

VectorToTableSearch();

connect(ui->tableWidget, SIGNAL(cellChanged(int, int)), this, SLOT(on\_tableWidget\_cellChanged(int, int)));

ui->statusbar->showMessage(QString("Search: DB:").append(QString::number(VectorDB.getSize())).append(" records. Found: ").append(QString::number(SearchRelationVector.getSize())).append(" records"));

}

void MainWindow::ShowContextMenu(const QPoint& pos)

{

// for most widgets

//QPoint globalPos = ui->tableWidget->mapToGlobal(pos);

// for QAbstractScrollArea and derived classes you would use:

QPoint globalPos = ui->tableWidget->viewport()->mapToGlobal(pos);

QMenu myMenu;

myMenu.addAction("Редактировать", this, SLOT(on\_Edit\_triggered()));

myMenu.addAction("Удалить", this, SLOT(on\_DeleteRecord\_triggered()));

myMenu.addAction("Добавить", this, SLOT(on\_AddRecord\_triggered()));

myMenu.exec(globalPos);

}

void MainWindow::on\_tableWidget\_cellClicked(int row, int column)

{

ui->statusbar->showMessage(QString("Current row:").append(QString::number(row+1)).append(" column:").append(QString::number(column+1)));

}

* 1. 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>892</width>

<height>590</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<property name="windowIcon">

<iconset resource="IconsResource.qrc">

<normaloff>:/IconPrefix1/Icons/notebook\_121787.ico</normaloff>:/IconPrefix1/Icons/notebook\_121787.ico</iconset>

</property>

<property name="iconSize">

<size>

<width>28</width>

<height>28</height>

</size>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QVBoxLayout" name="verticalLayout">

<item>

<layout class="QHBoxLayout" name="horizontalLayout">

<item>

<widget class="QLineEdit" name="LineEditSearch"/>

</item>

</layout>

</item>

<item>

<widget class="QTableWidget" name="tableWidget">

<property name="sizeAdjustPolicy">

<enum>QAbstractScrollArea::AdjustToContents</enum>

</property>

<property name="selectionMode">

<enum>QAbstractItemView::SingleSelection</enum>

</property>

<property name="selectionBehavior">

<enum>QAbstractItemView::SelectRows</enum>

</property>

</widget>

</item>

</layout>

</widget>

<widget class="QMenuBar" name="menubar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>892</width>

<height>18</height>

</rect>

</property>

<widget class="QMenu" name="menu">

<property name="title">

<string>Файл</string>

</property>

<addaction name="Create"/>

<addaction name="Open"/>

<addaction name="Save"/>

<addaction name="SaveAs"/>

<addaction name="AddDB"/>

<addaction name="Quit"/>

</widget>

<widget class="QMenu" name="menu\_2">

<property name="title">

<string>Записи</string>

</property>

<addaction name="AddRecord"/>

<addaction name="DeleteRecord"/>

<addaction name="Edit"/>

</widget>

<widget class="QMenu" name="menu\_3">

<property name="title">

<string>Отображение</string>

</property>

<addaction name="ShowTxt"/>

</widget>

<widget class="QMenu" name="menu\_4">

<property name="title">

<string>Помощь</string>

</property>

<addaction name="About"/>

</widget>

<addaction name="menu"/>

<addaction name="menu\_2"/>

<addaction name="menu\_3"/>

<addaction name="menu\_4"/>

</widget>

<widget class="QStatusBar" name="statusbar"/>

<widget class="QToolBar" name="toolBar">

<property name="windowTitle">

<string>toolBar</string>

</property>

<attribute name="toolBarArea">

<enum>TopToolBarArea</enum>

</attribute>

<attribute name="toolBarBreak">

<bool>false</bool>

</attribute>

<addaction name="Create"/>

<addaction name="Open"/>

<addaction name="Save"/>

<addaction name="SaveAs"/>

<addaction name="AddDB"/>

<addaction name="Quit"/>

<addaction name="separator"/>

<addaction name="AddRecord"/>

<addaction name="DeleteRecord"/>

<addaction name="Edit"/>

<addaction name="separator"/>

<addaction name="ShowTxt"/>

<addaction name="separator"/>

<addaction name="About"/>

</widget>

<action name="Create">

<property name="icon">

<iconset resource="IconsResource.qrc">

<normaloff>:/IconPrefix1/Icons/sticker\_121741.ico</normaloff>:/IconPrefix1/Icons/sticker\_121741.ico</iconset>

</property>

<property name="text">

<string>Создать</string>

</property>

<property name="shortcut">

<string>Ctrl+N</string>

</property>

</action>

<action name="Open">

<property name="icon">

<iconset resource="IconsResource.qrc">

<normaloff>:/IconPrefix1/Icons/openfile.ico</normaloff>:/IconPrefix1/Icons/openfile.ico</iconset>

</property>

<property name="text">

<string>Открыть</string>

</property>

<property name="shortcut">

<string>Ctrl+O</string>

</property>

</action>

<action name="Save">

<property name="icon">

<iconset resource="IconsResource.qrc">

<normaloff>:/IconPrefix1/Icons/save\_121760.ico</normaloff>:/IconPrefix1/Icons/save\_121760.ico</iconset>

</property>

<property name="text">

<string>Сохранить</string>

</property>

<property name="toolTip">

<string>Сохранить</string>

</property>

<property name="shortcut">

<string>Ctrl+S</string>

</property>

</action>

<action name="SaveAs">

<property name="icon">

<iconset resource="IconsResource.qrc">

<normaloff>:/IconPrefix1/Icons/saveAs+.ico</normaloff>:/IconPrefix1/Icons/saveAs+.ico</iconset>

</property>

<property name="text">

<string>Сохранить как</string>

</property>

<property name="shortcut">

<string>Ctrl+Shift+S</string>

</property>

</action>

<action name="AddDB">

<property name="icon">

<iconset resource="IconsResource.qrc">

<normaloff>:/IconPrefix1/Icons/shuffle\_121752.ico</normaloff>:/IconPrefix1/Icons/shuffle\_121752.ico</iconset>

</property>

<property name="text">

<string>Объединить БД</string>

</property>

<property name="shortcut">

<string>Ctrl+U</string>

</property>

</action>

<action name="Quit">

<property name="icon">

<iconset resource="IconsResource.qrc">

<normaloff>:/IconPrefix1/Icons/power\_121775.ico</normaloff>:/IconPrefix1/Icons/power\_121775.ico</iconset>

</property>

<property name="text">

<string>Выход</string>

</property>

<property name="shortcut">

<string>Ctrl+X</string>

</property>

</action>

<action name="AddRecord">

<property name="icon">

<iconset resource="IconsResource.qrc">

<normaloff>:/IconPrefix1/Icons/add\_121935.ico</normaloff>:/IconPrefix1/Icons/add\_121935.ico</iconset>

</property>

<property name="text">

<string>Добавить</string>

</property>

<property name="toolTip">

<string>Добавить</string>

</property>

<property name="shortcut">

<string>Ctrl+Shift+N</string>

</property>

</action>

<action name="DeleteRecord">

<property name="icon">

<iconset resource="IconsResource.qrc">

<normaloff>:/IconPrefix1/Icons/bin\_121907.ico</normaloff>:/IconPrefix1/Icons/bin\_121907.ico</iconset>

</property>

<property name="text">

<string>Удалить</string>

</property>

<property name="shortcut">

<string>Del</string>

</property>

</action>

<action name="Edit">

<property name="icon">

<iconset resource="IconsResource.qrc">

<normaloff>:/IconPrefix1/Icons/edit\_121852.ico</normaloff>:/IconPrefix1/Icons/edit\_121852.ico</iconset>

</property>

<property name="text">

<string>Редактировать</string>

</property>

<property name="shortcut">

<string>Ctrl+E</string>

</property>

</action>

<action name="ShowTxt">

<property name="icon">

<iconset resource="IconsResource.qrc">

<normaloff>:/IconPrefix1/Icons/txt.ico</normaloff>:/IconPrefix1/Icons/txt.ico</iconset>

</property>

<property name="text">

<string>Показать в текстовом виде</string>

</property>

<property name="shortcut">

<string>Ctrl+T</string>

</property>

</action>

<action name="About">

<property name="icon">

<iconset resource="IconsResource.qrc">

<normaloff>:/IconPrefix1/Icons/help\_121828.ico</normaloff>:/IconPrefix1/Icons/help\_121828.ico</iconset>

</property>

<property name="text">

<string>О программе</string>

</property>

<property name="shortcut">

<string>F1</string>

</property>

</action>

</widget>

<resources>

<include location="IconsResource.qrc"/>

</resources>

<connections/>

</ui>

* 1. editdialog.h

#ifndef EDITDIALOG\_H

#define EDITDIALOG\_H

#include <QDialog>

#include <QString>

namespace Ui {

class EditDialog;

}

class EditDialog : public QDialog

{

Q\_OBJECT

public:

explicit EditDialog(QWidget \*parent = nullptr);

EditDialog(QString txt0,QString txt1, QString txt2, QString txt3, QString txt4, QWidget \*parent = nullptr);

~EditDialog();

QString getName();

QString getManufacturer();

QString getSize();

QString getYear();

QString getOS();

private:

Ui::EditDialog \*ui;

};

#endif // EDITDIALOG\_H

* 1. editdialog.cpp

#include "editdialog.h"

#include "ui\_editdialog.h"

EditDialog::EditDialog(QWidget \*parent) :

QDialog(parent),

ui(new Ui::EditDialog)

{

ui->setupUi(this);

}

EditDialog::EditDialog(QString txt0,QString txt1, QString txt2, QString txt3, QString txt4, QWidget \*parent):QDialog(parent), ui(new Ui::EditDialog)

{

ui->setupUi(this);

ui->lineEdit->setText(txt0);

ui->lineEdit\_2->setText(txt1);

ui->lineEdit\_3->setText(txt2);

ui->lineEdit\_4->setText(txt3);

ui->lineEdit\_5->setText(txt4);

}

EditDialog::~EditDialog()

{

delete ui;

}

QString EditDialog::getName(){return ui->lineEdit->text();}

QString EditDialog::getManufacturer(){return ui->lineEdit\_2->text();}

QString EditDialog::getSize(){return ui->lineEdit\_3->text();}

QString EditDialog::getYear(){return ui->lineEdit\_4->text();}

QString EditDialog::getOS(){return ui->lineEdit\_5->text();}

* 1. editdialog.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>EditDialog</class>

<widget class="QDialog" name="EditDialog">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>434</width>

<height>175</height>

</rect>

</property>

<property name="sizePolicy">

<sizepolicy hsizetype="Preferred" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="contextMenuPolicy">

<enum>Qt::DefaultContextMenu</enum>

</property>

<property name="windowTitle">

<string>Dialog</string>

</property>

<property name="windowIcon">

<iconset resource="IconsResource.qrc">

<normaloff>:/IconPrefix1/Icons/edit\_121852.ico</normaloff>:/IconPrefix1/Icons/edit\_121852.ico</iconset>

</property>

<property name="sizeGripEnabled">

<bool>true</bool>

</property>

<layout class="QVBoxLayout" name="verticalLayout">

<property name="sizeConstraint">

<enum>QLayout::SetDefaultConstraint</enum>

</property>

<item>

<layout class="QGridLayout" name="gridLayout">

<item row="6" column="1">

<widget class="QLabel" name="label\_5">

<property name="text">

<string>ОС:</string>

</property>

</widget>

</item>

<item row="2" column="1">

<widget class="QLabel" name="label">

<property name="text">

<string>Название:</string>

</property>

</widget>

</item>

<item row="4" column="2">

<widget class="QLineEdit" name="lineEdit\_3"/>

</item>

<item row="3" column="1">

<widget class="QLabel" name="label\_2">

<property name="text">

<string>Фирма-Изготовитель:</string>

</property>

</widget>

</item>

<item row="4" column="1">

<widget class="QLabel" name="label\_3">

<property name="text">

<string>Размер:</string>

</property>

</widget>

</item>

<item row="3" column="2">

<widget class="QLineEdit" name="lineEdit\_2"/>

</item>

<item row="5" column="2">

<widget class="QLineEdit" name="lineEdit\_4"/>

</item>

<item row="2" column="2">

<widget class="QLineEdit" name="lineEdit"/>

</item>

<item row="5" column="1">

<widget class="QLabel" name="label\_4">

<property name="text">

<string>Год выпуска:</string>

</property>

</widget>

</item>

<item row="6" column="2">

<widget class="QLineEdit" name="lineEdit\_5"/>

</item>

</layout>

</item>

<item>

<widget class="QDialogButtonBox" name="buttonBox">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="standardButtons">

<set>QDialogButtonBox::Cancel|QDialogButtonBox::Ok</set>

</property>

<property name="centerButtons">

<bool>false</bool>

</property>

</widget>

</item>

</layout>

</widget>

<tabstops>

<tabstop>lineEdit</tabstop>

<tabstop>lineEdit\_2</tabstop>

<tabstop>lineEdit\_3</tabstop>

<tabstop>lineEdit\_4</tabstop>

<tabstop>lineEdit\_5</tabstop>

</tabstops>

<resources>

<include location="IconsResource.qrc"/>

</resources>

<connections>

<connection>

<sender>buttonBox</sender>

<signal>accepted()</signal>

<receiver>EditDialog</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>

<connection>

<sender>buttonBox</sender>

<signal>rejected()</signal>

<receiver>EditDialog</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>

* 1. showtxtdialog.h

#ifndef SHOWTXTDIALOG\_H

#define SHOWTXTDIALOG\_H

#include <QDialog>

namespace Ui {

class ShowTxtDialog;

}

class ShowTxtDialog : public QDialog

{

Q\_OBJECT

public:

explicit ShowTxtDialog(QWidget \*parent = nullptr);

~ShowTxtDialog();

void TextEditSetText(QString StrCSV);

private:

Ui::ShowTxtDialog \*ui;

};

#endif // SHOWTXTDIALOG\_H

* 1. showtxtdialog.cpp

#include "showtxtdialog.h"

#include "ui\_showtxtdialog.h"

ShowTxtDialog::ShowTxtDialog(QWidget \*parent) :

QDialog(parent),

ui(new Ui::ShowTxtDialog)

{

ui->setupUi(this);

}

ShowTxtDialog::~ShowTxtDialog()

{

delete ui;

}

void ShowTxtDialog::TextEditSetText(QString StrCSV)

{

ui->textEdit->setText(StrCSV);

}

* 1. showtxtdialog.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>ShowTxtDialog</class>

<widget class="QDialog" name="ShowTxtDialog">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>400</width>

<height>300</height>

</rect>

</property>

<property name="windowTitle">

<string>Dialog</string>

</property>

<property name="windowIcon">

<iconset resource="IconsResource.qrc">

<normaloff>:/IconPrefix1/Icons/txt.ico</normaloff>:/IconPrefix1/Icons/txt.ico</iconset>

</property>

<layout class="QVBoxLayout" name="verticalLayout\_2">

<item>

<widget class="QTextEdit" name="textEdit">

<property name="readOnly">

<bool>true</bool>

</property>

</widget>

</item>

</layout>

</widget>

<resources>

<include location="IconsResource.qrc"/>

</resources>

<connections/>

</ui>

* 1. main.cpp

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}