

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

Работа допущена к защите

з/у Руководитель
«27» июль 2019 г.

КУРСОВАЯ РАБОТА

по дисциплине «Объектно-ориентированное программирование на языке
C++»

на тему: «Разработка информационно-поисковой системы «Страховая
компания»»

Студент Беликов Беликов П.Г.

Шифр 180818

Институт приборостроения, автоматизации и информационных технологий
Направление подготовки 09.03.04 «Программная инженерия»

Группа 81ПГ

Руководитель з/у Захарова О.В.

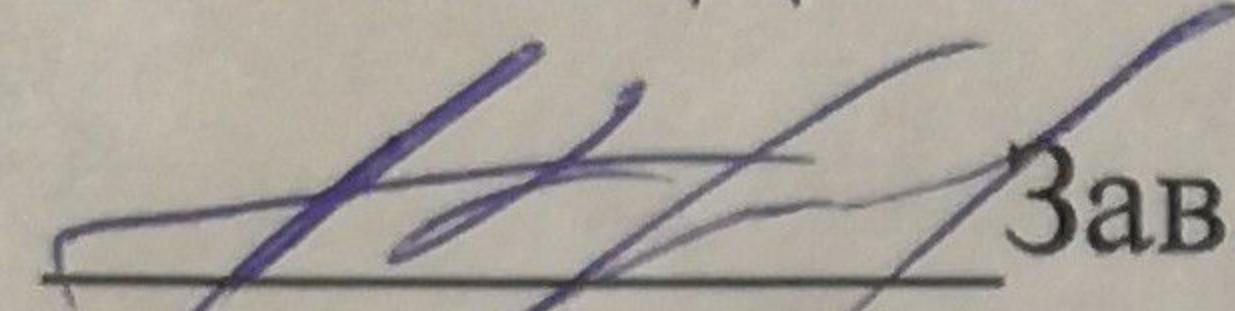
Оценка: « отлично » Дата 05.06.19

Орел 2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

УТВЕРЖДАЮ:

 Зав. кафедрой
«27» 02 2019 г.

ЗАДАНИЕ
на курсовую работу

по дисциплине «Объектно – ориентированное программирование на языке
C++»

Студент Беликов П.Г.

Шифр 180818

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Группа 81ПГ

1 Тема курсовой работы

«Разработка информационно – поисковой системы «Страховая компания»»

2 Срок сдачи студентом законченной работы «27» мая 2019

3 Исходные данные

Условия задачи, алгоритмы работы с файлами и массивами

4 Содержание курсовой работы

Постановка задачи

Обоснование выбора метода решения

Обоснование выбора структур данных

Описание алгоритмов

Особенности программной реализации

Описание пользовательского интерфейса

5 Отчётный материал курсовой работы

Пояснительная записка курсовой работы, презентация

Руководитель _____ Захарова О.В.

Задание принял к исполнению: «27» февраля 2019

Подпись студента _____

Содержание

Введение.....	4
1 Постановка задачи.....	5
2 Обоснование выбора метода решения.....	6
3 Обоснование выбора структур данных.....	8
4 Описание алгоритмов решения задачи.....	10
5 Особенности программной реализации.....	18
6 Описание пользовательского интерфейса.....	21
Заключение.....	30
Список использованных источников.....	31
Приложение А (обязательное) Листинг программы.....	32

Введение

Целью курсовой работы является разработка информационно-поисковой системы «Страховая компания» на языке C++.

Задачами курсовой работы, которые необходимо выполнить для достижения поставленной цели, являются:

1. Выбор метода решения поставленной задачи.
2. Выбор подходящих структур данных.
3. Разработка алгоритмов решения.
4. Реализация программы на языке C++.
5. Описание пользовательского интерфейса.

1 Постановка задачи

Разработать информационно-поисковую систему «Страховая компания». Для хранения информации использовать файл. Функционал программы:

1. Добавление информации.
2. Редактирования информации.
3. Удаление информации.
4. Поиск информации по заданным критериям.

2 Обоснование выбора метода решения

В программе для хранения информации используется массив.

При добавлении информации в массив можно использовать следующие способы:

- Добавление в начало массива.
- Добавление в середину массива.
- Добавление в конец массива.

Я использовал последний способ добавления, так как при выводе информации в таблицу удобнее добавить новую строку с новым элементом в конец, не теряя нумерации остальных элементов. Также, ввиду меньшего количества операций, такое решение положительно сказывается на производительности программы.

При удалении информации можно использовать следующие способы:

- Поменять местами удаляемый элемент с последним, затем удалить последний элемент.
- Сместить все объекты, начиная с того, что следует за удаляемым элементом, на один элемент назад, а затем удалить последний.
- Удалить необходимый объект, оставив в массиве пустой элемент.

Я использовал способ со смещением, так как мне важен порядок элементов массива при выводе. В таблице они должны быть расположены в том порядке, в котором их ввел пользователь.

При поиске можно использовать следующие способы:

- Линейный поиск.
- Бинарный поиск.

При поиске определенного значения элемента я использовал линейный поиск, так как он прост в реализации, не требует сортировки значений множества, дополнительной памяти и дополнительного анализа функций. Также, в отличие от бинарного, линейный поиск выводит найденные элементы в порядке их ввода, а мне важен порядок при выводе.

При изменении элемента, считывается номер строки таблицы, которая отображается в текстовом поле, и последовательным поиском в массиве находится требуемый объект. Причина использования линейного поиска описана выше.

При сохранении файла перезаписывается вся информация, которая в нем уже имелась, потому, что важен не только порядок записи элементов в файл, но и порядок их извлечения. Перезаписывая файл заново, мы гарантируем последовательность элементов, соответствующую той, что была при сохранении.

При открытии файла выполняется очистка массива и запись в него информации из файла.

Для выполнения поставленных задач мною была выбрана интегрированная среда разработки C++ Builder 6.

3 Обоснование выбора структур данных

Структура данных — программная единица, позволяющая хранить и обрабатывать множество однотипных либо логически связанных данных в вычислительной технике. Для добавления, поиска, изменения и удаления данных структура данных предоставляет некоторый набор функций, составляющих её интерфейс.

В данной работе были использованы следующие структуры данных:

- Класс.
- Файл.
- Массив.
- Стока.

Класс — это элемент ПО, описывающий абстрактный тип данных и его частичную или полную реализацию. Наряду с понятием «объекта» класс является ключевым понятием в ООП. Суть отличия классов от других абстрактных типов данных состоит в том, что при задании типа данных класс определяет одновременно как интерфейс, так и реализацию для всех своих экземпляров, а вызов метода-конструктора обязателен. В отличие от структуры, которую так же можно было использовать, класс позволяет создавать приватные свойства и методы. [4]

Файл — структура данных для постоянного хранения информации. Файл позволяет сохранить данные после окончания работы и загрузить их для её продолжения при следующем запуске программы. В программе я использовал текстовый файл чтобы иметь возможность чтения сохраненной информации пользователем без использования программы.

Массив — упорядоченный набор элементов, каждый из которых хранит одно значение, идентифицируемое с помощью одного или нескольких индексов. В простейшем случае массив имеет постоянную длину и хранит единицы данных одного и того же типа, а в качестве индексов выступают целые числа. Массив удобен для хранения данных фиксированного

количества элементов определенного типа. В программе используется статический массив для повышения производительности. [2]

Строки являются разновидностью одномерного массива, но, в отличие от него, содержат множество методов, значительно облегчающих работу с большим количеством текстовых данных. Поставленная задача обязывает хранить и управлять множеством текстовых данных. [6]

4 Описание алгоритмов решения задачи

Для решения поставленных задач были разработаны алгоритмы, представленные на следующих схемах.

После запуска выполняется следующий алгоритм:

1. Проверить, выбран ли пункт меню «Exit». Если выбран, то завершаем работу программы, если нет, то переходим на шаг 2.
2. Проверить, нажата ли кнопка «Enter». Если да, то переходим на шаг 3, иначе на шаг 4.
3. Добавить информацию о контракте. Переходим на шаг 1.
4. Проверить, нажата ли кнопка «Find». Если нажата, то переходим на шаг 5, иначе на шаг 6.
5. Выполнить поиск по заданным критериям. Переходим на шаг 1.
6. Проверить, нажата ли кнопка «Delete». Если нажата, то переходим на шаг 7, иначе на шаг 8.
7. Удалить информацию о контракте. Переходим на шаг 1.
8. Проверить, нажата ли кнопка «Change». Если нажата, то переходим на шаг 9, иначе на шаг 10.
9. Изменить данные выбранного контракта. Переходим на шаг 1.
10. Проверить, нажата ли кнопка «Clear». Если да, то переходим на шаг 11, иначе на шаг 12.
11. Очистить поля ввода информации. Переходим на шаг 1.
12. Проверить, нажата ли кнопка «ShowAllContracts». Если да, то переходим на шаг 13, иначе на шаг 14.
13. Отобразить все данные массива в таблицу. Переходим на шаг 1.
14. Проверить, выбран ли пункт меню «Save». Если выбран, то переходим на шаг 15, иначе на шаг 16.
15. Сохранение в новый/выбранный файл. Переходим на шаг 1.
16. Проверить, выбран ли пункт меню «Open». Если да, то переходим на шаг 17, иначе на шаг 1.

17. Открыть выбранный файл. Переходим на шаг 1.

На Рисунке 2 показана логика работы программы.

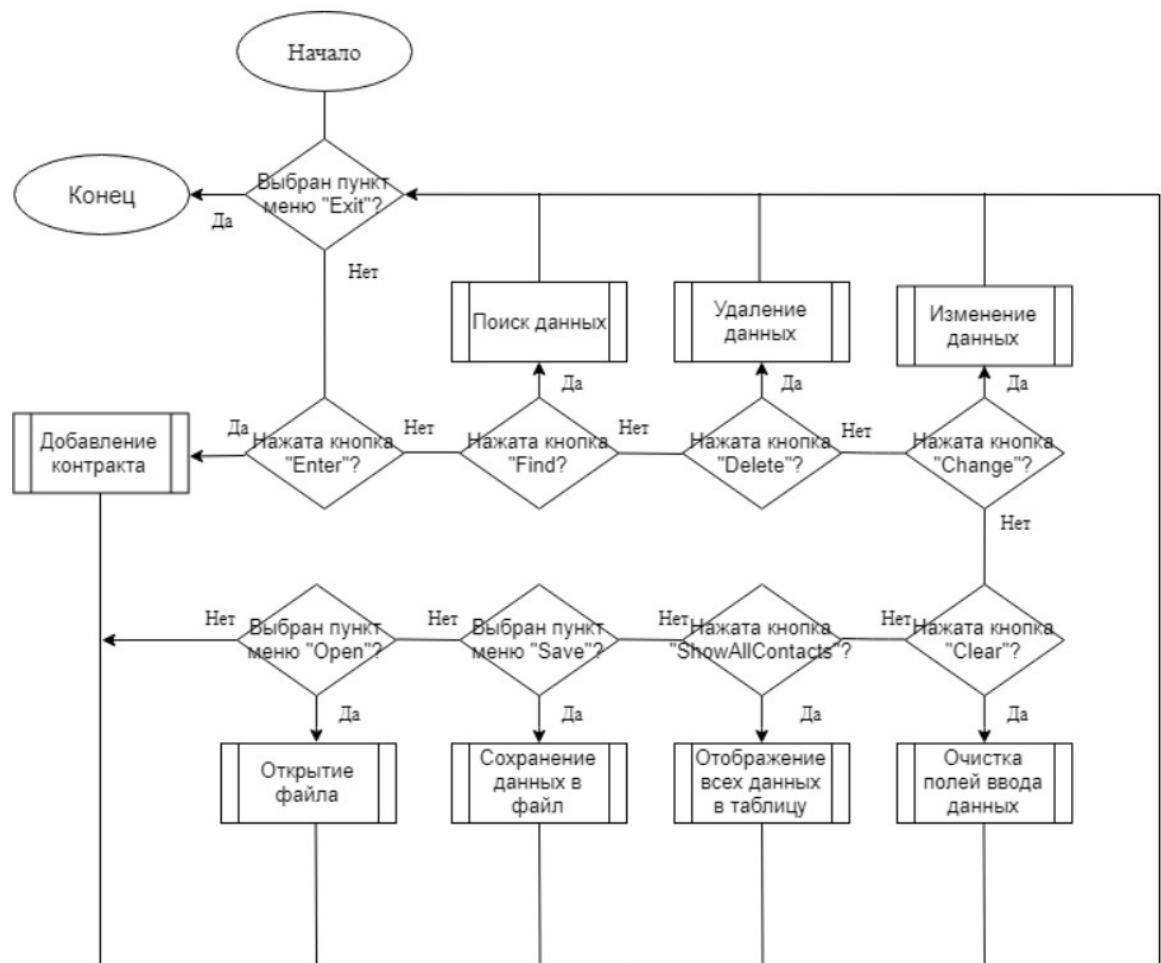


Рисунок 2 – Схема логики программы

Завершить программу можно в любой момент, информация из массива не будет сохранена, если не выполнено соответствующее действие.

При добавлении информации происходит следующий алгоритм:

1. Проверка на корректность ввода. Если данные введены корректно, то переходим на шаг 3, иначе на шаг 2.
2. Вывести сообщение об ошибке. Переходим на шаг 7.
3. Использовать конструктор с параметрами.
4. Поместить объект в ячейку массива, соответствующую значению счетчика.
5. Увеличить значение счетчика на один.

6. Отобразить информацию в таблице.
7. Завершить выполнение.

Описанное выше представлено на Рисунке 3.



Рисунок 3 – Схема работы кнопки «Enter»

При удалении элемента происходит следующее:

1. Проверить, выбран ли объект для удаления. Если выбран, то переходим на шаг 2, иначе на шаг 5.
2. Для каждого элемента массива, начиная со следующего за выбранным: сместить на одну позицию назад.
3. Уменьшить значение счетчика на один.
4. Отобразить изменения в таблице.
5. Завершить выполнение.

Работа кнопки «Удалить» представлена на Рисунке 4.

При поиске контракта происходит следующее:

1. Проверка на корректность ввода. Если есть пустое поле, то переходим на шаг 2, иначе на шаг 3.
2. Вывести сообщение об ошибке. Переходим на шаг 5.
3. Очистить таблицу.
4. Проверить, подходят ли введенные данные к данным элемента массива. Если да, то выводим подходящий элемент в таблицу и просматриваем следующий элемент массива, если нет, то просматриваем следующий элемент массива.
5. Завершить выполнение.



Рисунок 4 – Схема работы кнопки «Delete»

Работа кнопки «Find» показана на Рисунке 5.

При редактировании информации выполняется следующий алгоритм:

1. Проверить, выбран ли объект для изменения. Если выбран, то переходим на шаг 2, иначе на шаг 8.
2. Проверка на корректность ввода. Если есть пустые поля, то переходим на шаг 3, если нет пустых полей, то на шаг 4.
3. Вывести сообщение об ошибке. Переходим на шаг 8.
4. Удалить выбранный объект.
5. Использовать конструктор с параметрами.
6. Поместить созданный объект на место удаленного в массиве.
7. Отобразить изменения в таблице.
8. Завершить выполнение.

Работа кнопки «Change» показана на Рисунке 6.

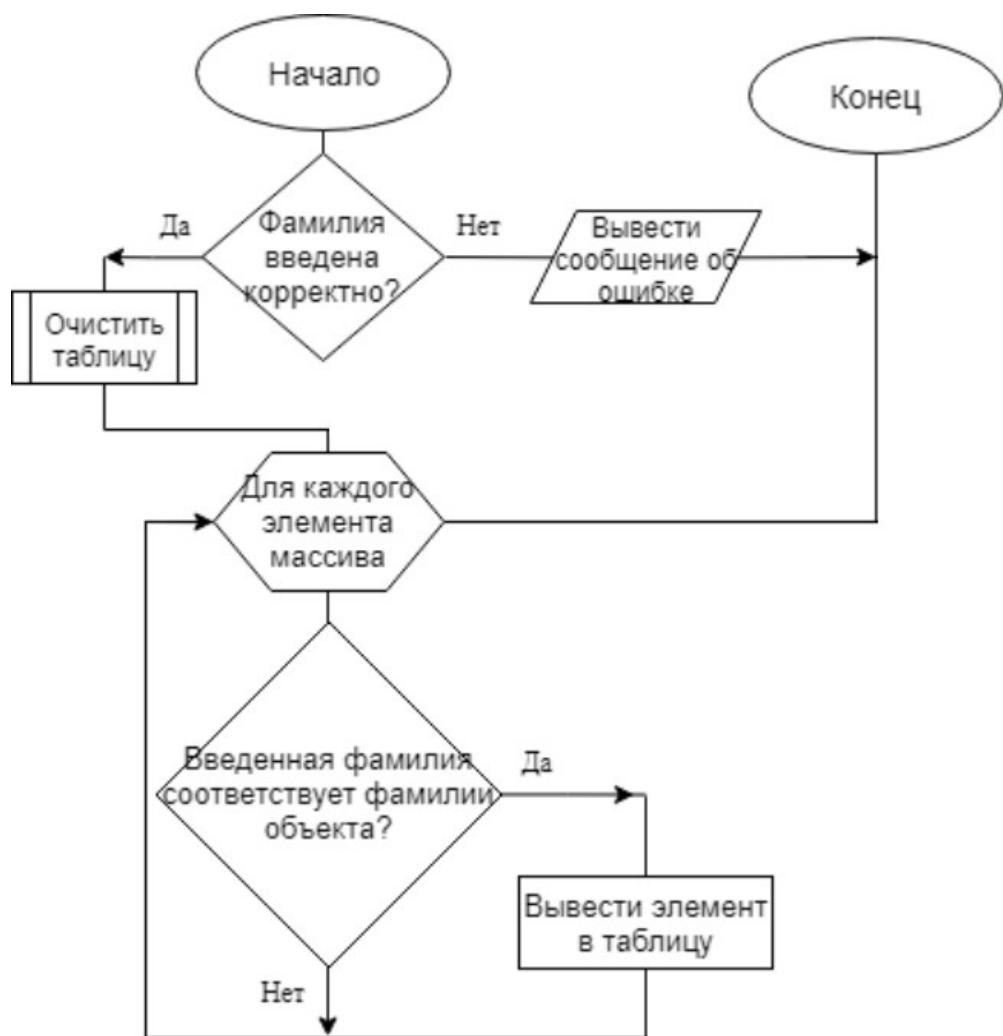


Рисунок 5 – Схема работы кнопки «Find» (пример: поиск по фамилии)

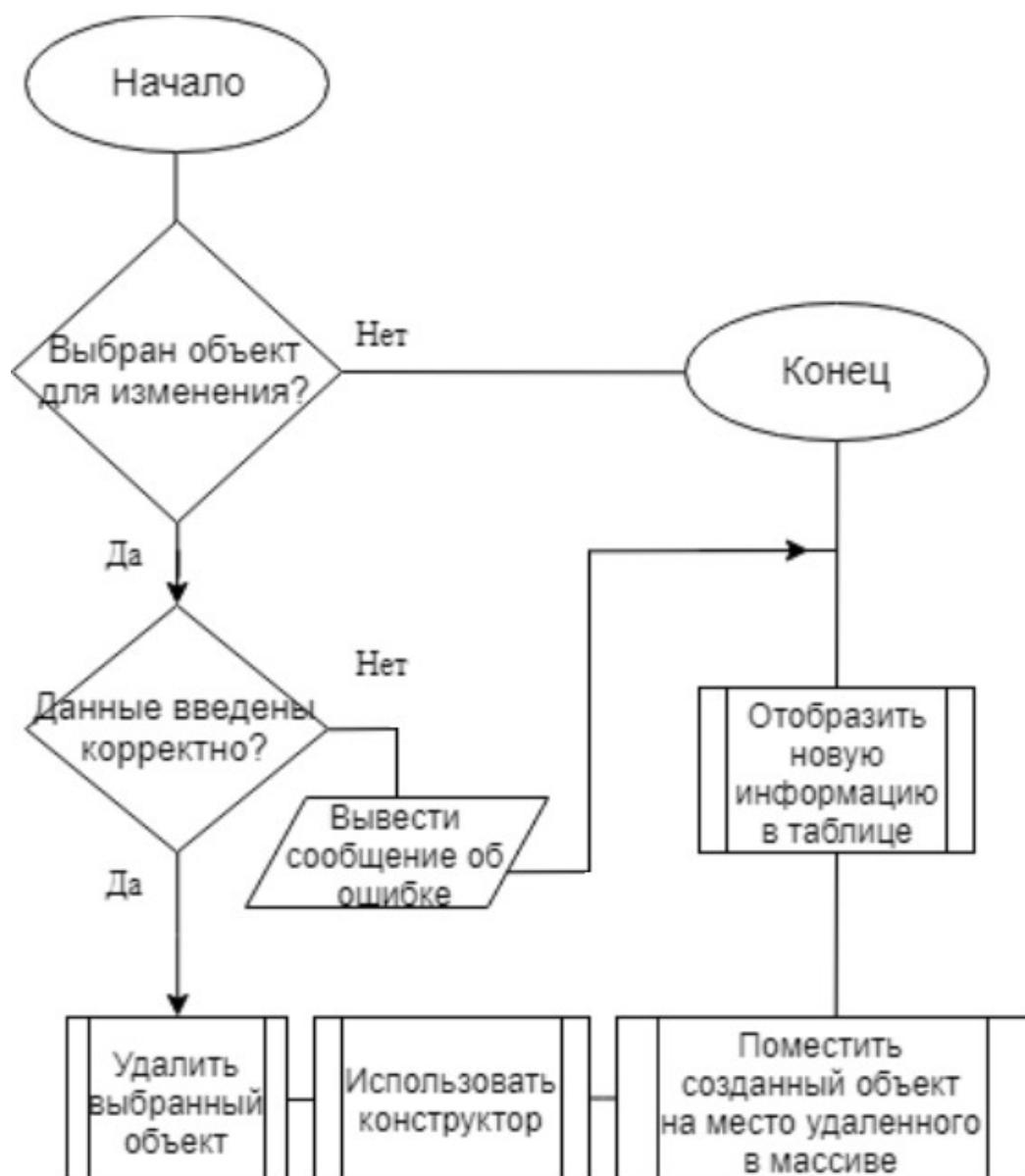


Рисунок 6 – Схема работы кнопки «Change»

При сохранении информации в файл выполняется следующий алгоритм:

1. Проверить, открыто ли окно сохранения. Если открыто, то идем на шаг 2, иначе на шаг 6.
2. Открыть файл для сохранения.
3. Проверить, открыт ли файл для сохранения. Если открыт, то идем на шаг 4, иначе на шаг 6.
4. Для каждого элемента массива: если индекс элемента равен нулю, то производим построчную запись полей объекта в файл, иначе ставим

символ перевода строки и производим построчную запись полей объекта в файл.

5. Закрыть файл для сохранения.
6. Завершить работу.

Работа кнопки «Save» показана на рисунке 7.

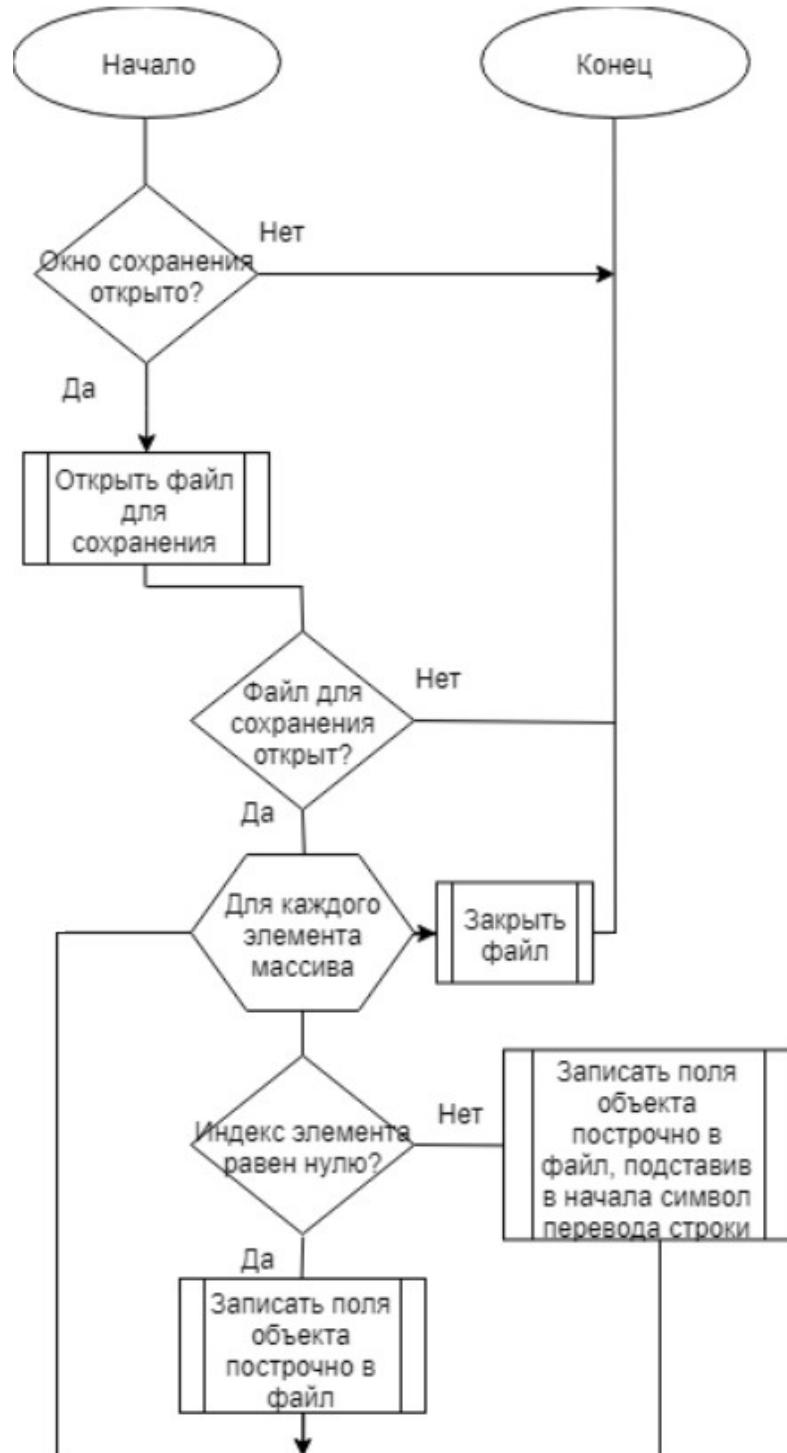


Рисунок 7 – Схема работы кнопки «Save»

При загрузке информации из файла выполняется следующий алгоритм:

1. Проверить, запущено ли окно открытия файла. Если да, то идем на шаг 2, иначе на шаг 6.
2. Очистить массив.
3. Открыть файл для считывания информации.
4. Пока не достигнут конец файла: использовать конструктор без параметров и занести в его поля построчно данные из файла до символа перевода строки. Увеличить значение счетчика на один.
5. Закрыть файл для считывания информации.
6. Завершить работу.

Работа кнопки «Open» показана на рисунке 8.

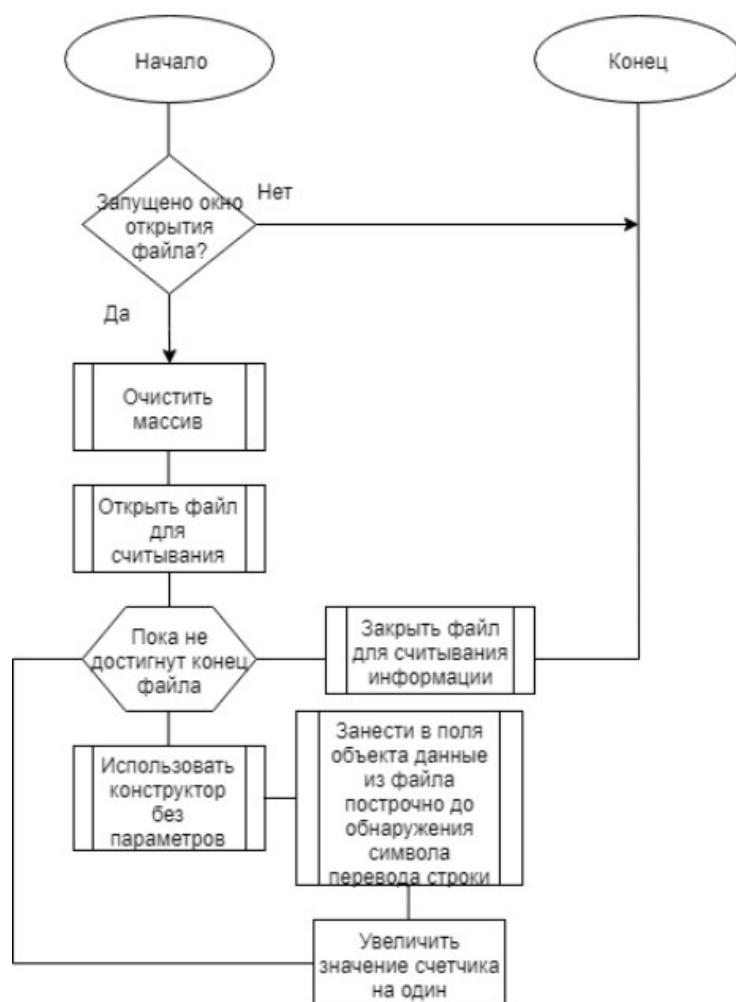


Рисунок 8 – Схема работы кнопки «Open»

5 Особенности программной реализации

Были созданы два класса для различных целей и разделения ответственности:

- Класс формы (окно).
- Класс магазина (манипуляции с данными).

Класс формы TTIME работает с графическим интерфейсом, "смотрит" нажатия на кнопки и использует класс Insurance.

Класс TTIME имеет множество методов, которые вызываются при каком-либо действии с элементами формы. Также с этим он имеет указатели на эти элементы.

Рассмотрим подробнее несколько моментов:

- В классе TTIME все методы имеют соглашение о вызове `_fastcall`, которое означает передачу аргументов функции через регистры. Это соглашение о вызове рекомендовано в Borland C++ Compiler;
- Класс TTIME содержит такие методы-обработчики:
 - EnterClick - нажатие на кнопку добавления.
 - ShowAllContractsClick - нажатие на кнопку вывода данных в таблицу.
 - DeleteClick - нажатие на кнопку удаления.
 - FindClick - нажатие на кнопку поиска.
 - Button1Click - нажатие на кнопку очистки полей ввода.
 - Button2Click - нажатие на кнопку изменения.
 - SaveClick - нажатие на кнопку записи в файл.
 - OpenClick - нажатие на кнопку считывания из файла.
 - Edit1KeyPress - корректирует ввод в поле Edit1.
 - Edit2KeyPress - корректирует ввод в поле Edit2.
 - Edit3KeyPress - корректирует ввод в поле Edit3.
 - Edit4KeyPress - корректирует ввод в поле Edit4.

- Edit6KeyPress - корректирует ввод в поле Edit6.
- ExitClick - нажатие на кнопку выхода.
- FormCloseQuery - нажатие на кнопку выхода.
- StringGrid1Click - нажатие на таблицу.
- ComboBox3Change - выполнение каких-либо действий в зависимости от выбранного значения выпадающего списка.

Класс Insurance содержит в себе следующие приватные поля:

- Name - содержит имя объекта.
- Surname - содержит фамилию объекта.
- Number - содержит номер объекта.
- Passport - содержит данные паспорта объекта.
- Ins - содержит тип страхования объекта.
- Price - содержит стоимость услуги для объекта (устанавливается автоматически в зависимости от значения Ins).

Класс Insurance имеет конструктор с параметрами, конструктор без параметров, деструктор, а также следующие публичные методы:

- getName - возвращает имя объекта.
- getSurname - возвращает фамилию объекта.
- getNumber - возвращает номер объекта.
- getPassport - возвращает паспортные данные объекта.
- getIns - возвращает тип страхования объекта.
- getPrice - возвращает стоимость услуг для объекта.
- setName - устанавливает имя объекта, принимает данные для установки.
- setSurname - устанавливает фамилию объекта, принимает данные для установки.
- setNumber - устанавливает номер объекта, принимает данные для установки.
- setPassport - устанавливает паспортные данные объекта, принимает данные для установки.

- `setIns` - устанавливает тип страхования объекта, принимает данные для установки.
- `infile` - считывает информацию из файла в массив.

Подробное описание всех функций на языке C++ представлено в Приложении А.

6 Описание пользовательского интерфейса

Программа выполнена в графическом виде. После запуска программы перед пользователем открывается окно, представленное на Рисунке 9.

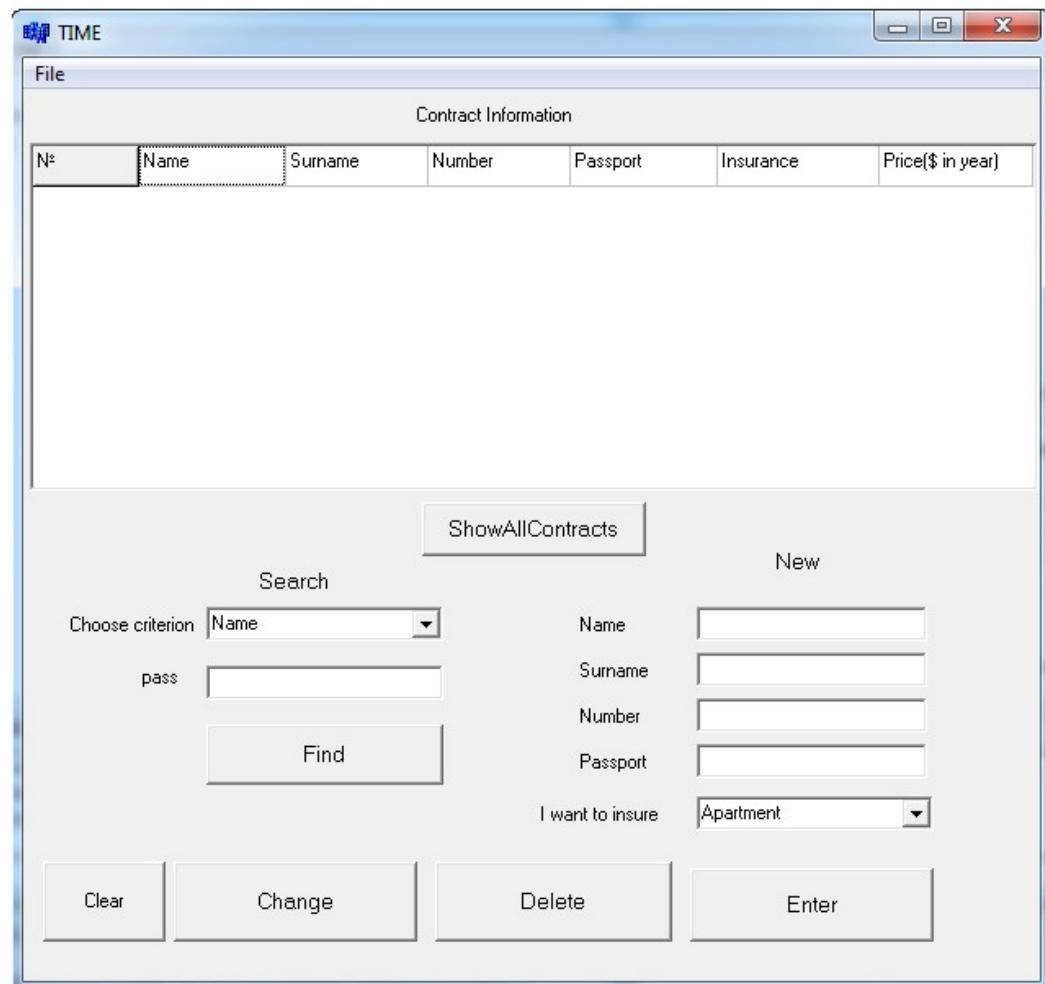


Рисунок 9 – Главное окно

Для добавления информации необходимо заполнить текстовые поля «Name», «Surname», «Number», «Passport» и выбрать одно из возможных направлений в выпадающем списке «I want to insure». Во время ввода информации регистр букв не учитывается. После ввода информации необходимо нажать на кнопку «Enter», чтобы данные добавились в таблицу. Если одно из полей пустое, то программа выдаст ошибку (Рисунок 15). Вышеописанное проиллюстрировано на Рисунке 10.

TIME

File

Contract Information

Nº	Name	Surname	Number	Passport	Insurance	Price(\$ in year)
----	------	---------	--------	----------	-----------	-------------------

Search

ShowAllContracts New

Choose criterion Name

pass

Name Surname Number Passport I want to insure

Find

Clear Change Delete Enter

TIME

File

Contract Information

Nº	Name	Surname	Number	Passport	Insurance	Price(\$ in year)
1	Hego	Damask	89996661707	5414435271	Life	30

Search

ShowAllContracts New

Choose criterion Name

pass

Name Surname Number Passport I want to insure

Find

Clear Change Delete Enter

Рисунок 10 – Добавление информации в таблицу

При удалении информации требуется выбрать удаляемый объект в таблице, после чего нажать кнопку «Delete». Процесс удаления проиллюстрирован на рисунке 11.

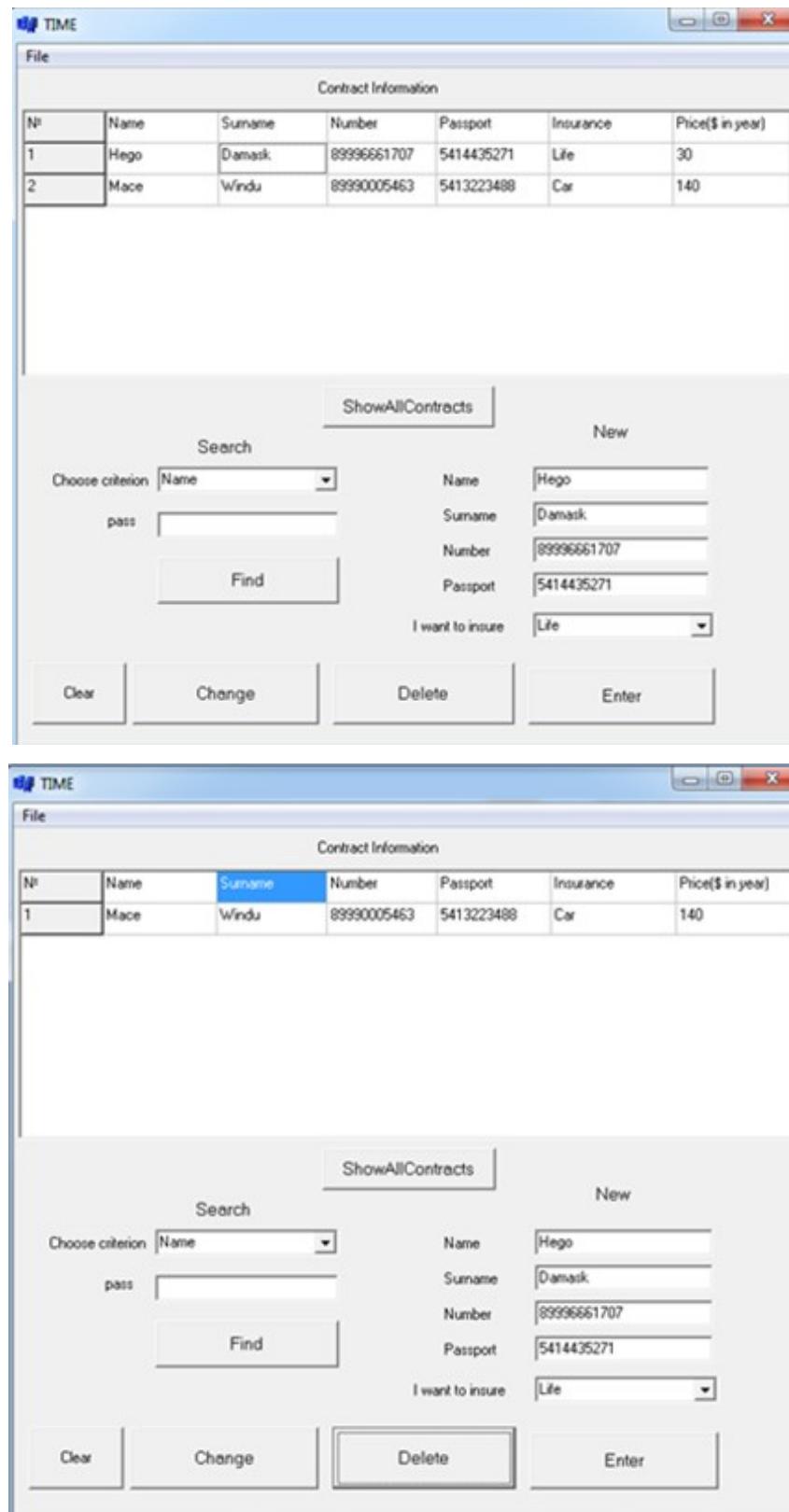


Рисунок 11 – Удаление элемента

Для редактирования информации пользователю выбрать изменяемых объект в таблице. После этого в соответствующих полях появятся данные об объекте, которые можно редактировать. После завершения редактирования нужно нажать кнопку «Change». Процесс изменения информации проиллюстрирован на Рисунке 12.

При поиске необходимых данных пользователь должен выбрать критерий поиска в выпадающем списке «Choose criterion», а затем заполнить поле «pass». В зависимости от выбранного в выпадающем списке, поиск будет осуществляться по имени, фамилии или типу страхования. Процесс поиска представлен на Рисунке 13. Если не было найдено ни одного объекта, удовлетворяющего критериям поиска, то пользователь увидит окно, представленное на Рисунке 14.

The screenshot shows a Windows application window titled "TIME". The main title bar has standard minimize, maximize, and close buttons. Below the title bar is a menu bar with "File" and a "Contract Information" section. The "Contract Information" section contains a table with columns: №, Name, Surname, Number, Passport, Insurance, and Price(\$ in year). A single row is displayed with values: 1, Mace, Windu, 89990005463, 5413223488, Car, 140. The "Surname" field is currently selected and highlighted in blue. Below the table is a "Search" section with the following controls:

- "Choose criterion": A dropdown menu set to "Name".
- "pass": An input field for search terms.
- "Find": A button to execute the search.
- "ShowAllContracts": A button to view all contracts.
- "New": A button to add a new contract.
- "Name": An input field containing "Macerell".
- "Surname": An input field containing "Windu".
- "Number": An input field containing "89990005463".
- "Passport": An input field containing "5413223488".
- "I want to insure": A dropdown menu set to "Mortgage".
- "Clear": A button to clear search fields.
- "Change": A button to change the selected record.
- "Delete": A button to delete the selected record.
- "Enter": A button to enter new data.

Рисунок 12 – Работа кнопки «Change», Лист 1

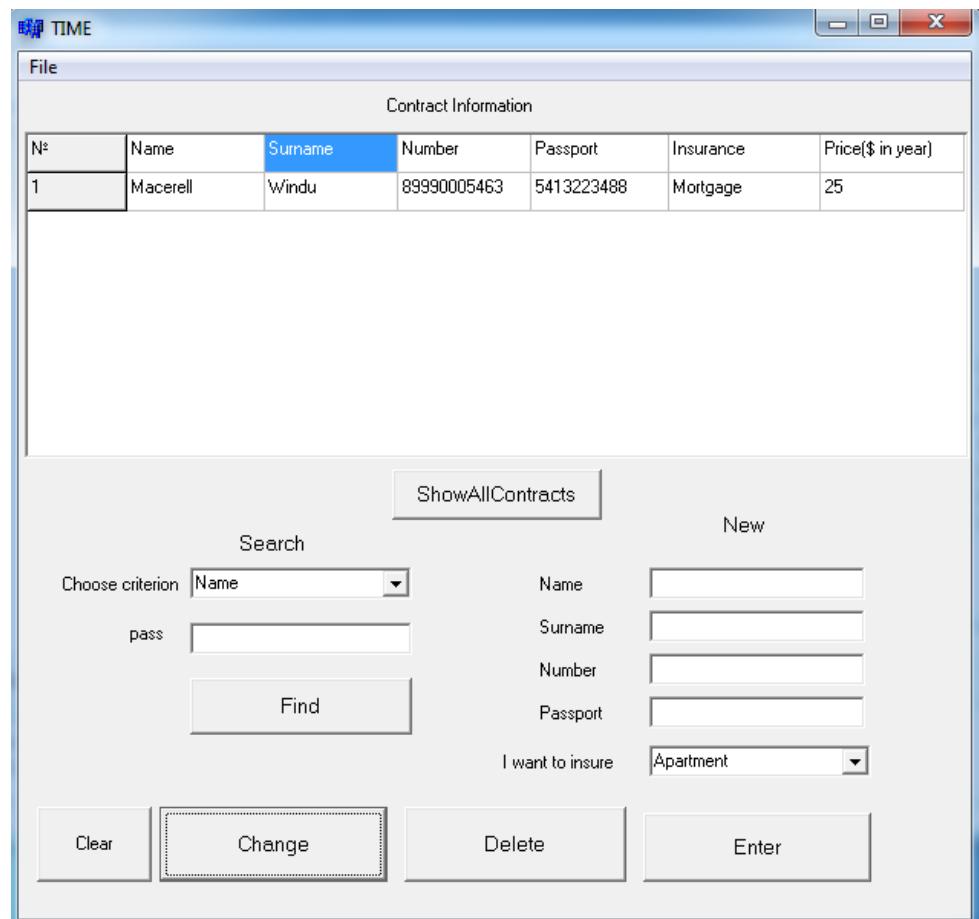


Рисунок 12 – Работа кнопки «Change», Лист 2

Для сохранения информации таблицы нужно выбрать пункт меню «Save». Откроется диалоговое окно, в котором можно сохранить проект в выбранный файл или создать новый. Диалоговое окно показано на Рисунке 16.

Для того, чтобы открыть сохраненный проект, пользователю нужно выбрать пункт меню «Open», после чего появится диалоговое окно для выбора файла. Диалоговое окно для выбора файла показано на Рисунке 17. Для выхода из программы пользователю нужно выбрать пункт меню «Exit», после чего появится окно с вопросом, изображенное на Рисунке 18. Если пользователь случайно выбрал данный пункт меню, то он может вернуться к проекту после нажатия кнопки «No». Если пользователь нажмет кнопку «Yes», то программа предложит сохранить изменения в проекте (Рисунок 19). Если пользователь нажмет кнопку «Yes», то откроется диалоговое окно

TIME

Contract Information						
Nº	Name	Surname	Number	Passport	Insurance	Price(\$ in year)
1	Macerell	Windu	89990005463	5413223488	Mortgage	25
2	Hego	Damask	89996667777	5434554666	Apartment	80
3	Vizer	Damask	87770009898	5434222222	Responsibility	20

ShowAllContracts

Search

Choose criterion Name

pass Surname

Number

Passport Apartment

TIME

Contract Information						
Nº	Name	Surname	Number	Passport	Insurance	Price(\$ in year)
1	Hego	Damask	89996667777	5434554666	Apartment	80
2	Vizer	Damask	87770009898	5434222222	Responsibility	20

ShowAllContracts

Search

Choose criterion Name

pass Surname

Number

Passport Apartment

Рисунок 13 – Работа кнопки «Find»

Nº	Name	Surname	Number	Passport	Insurance	Price(\$ in year)
1	No match	No match				

Рисунок 14 – Нет совпадений при поиске



Рисунок 15 – Ошибка: пустые поля

для сохранения файла, если «No», то программа закроется, не сохранив результат работы пользователя.

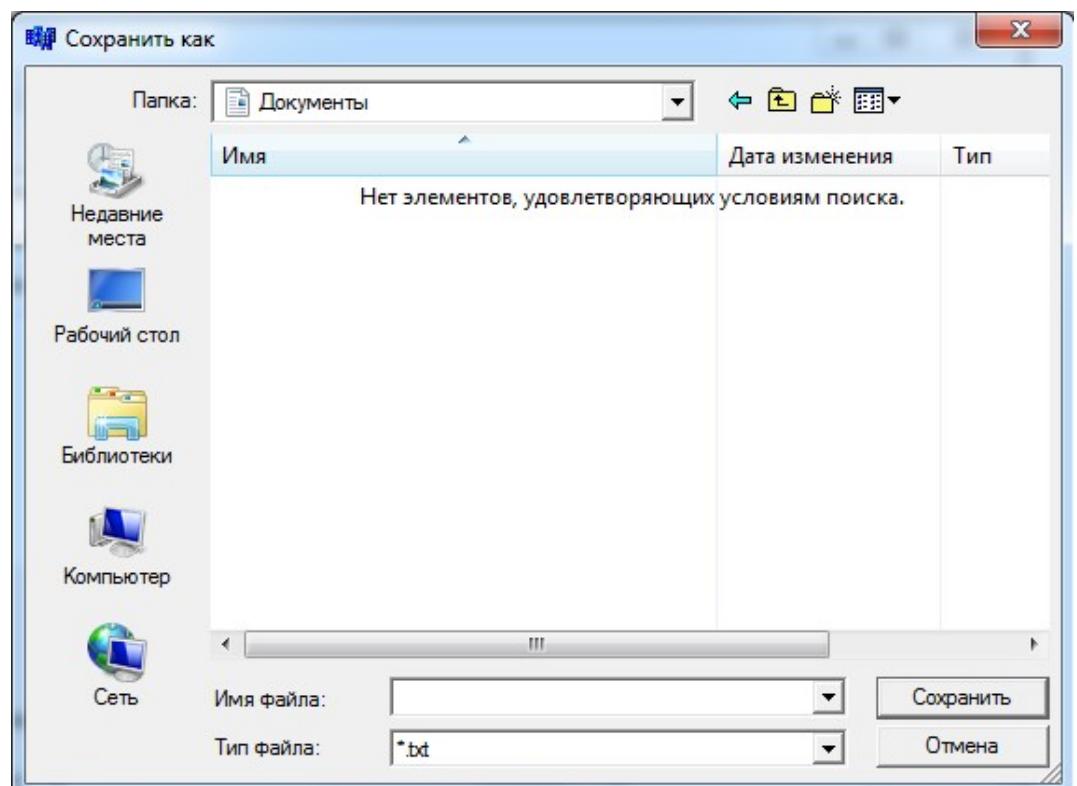


Рисунок 16 – Окно для сохранения файла

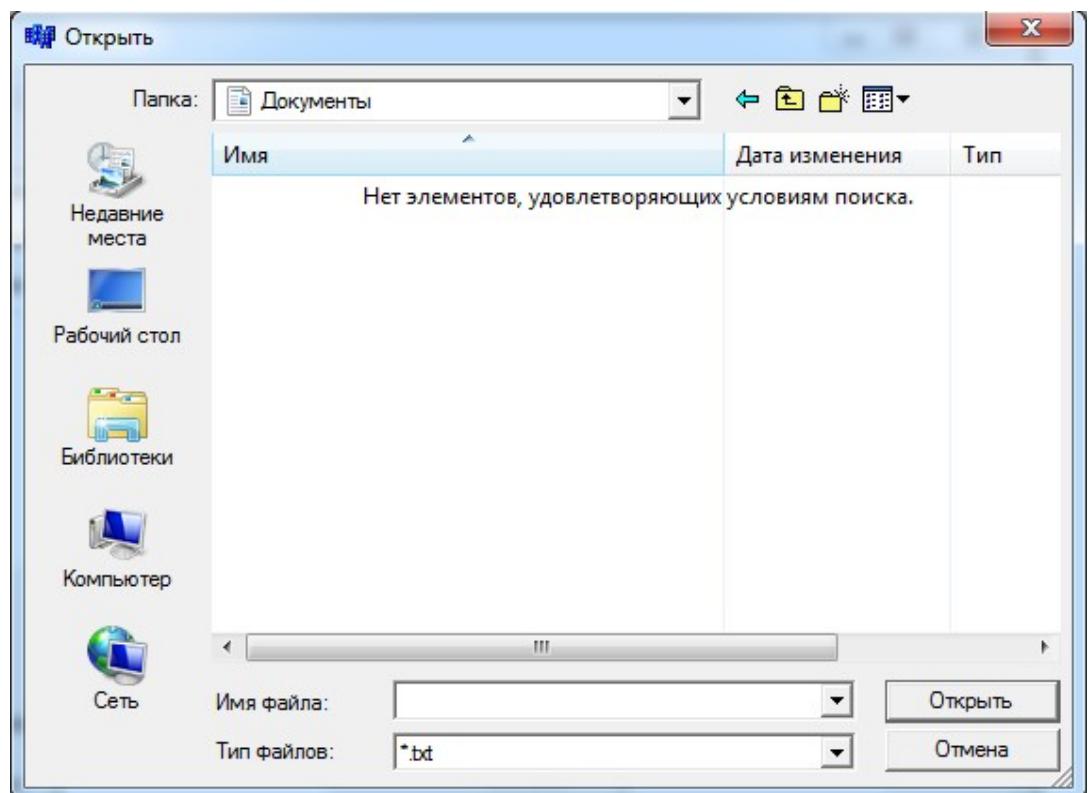


Рисунок 17 – Окно для открытия файла

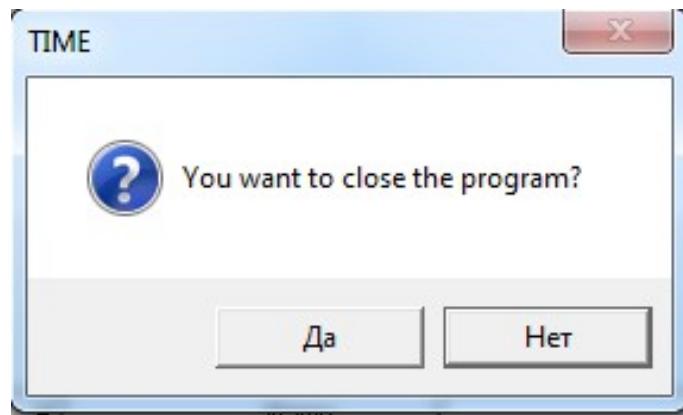


Рисунок 18 – Окно выхода

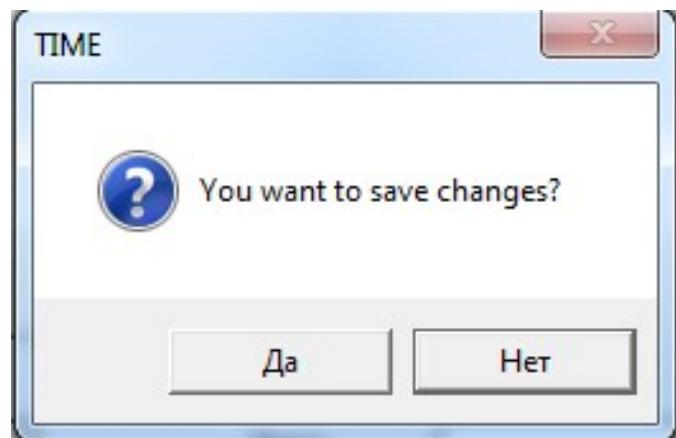


Рисунок 19 – Окно подтверждения изменений

Заключение

В ходе выполнения курсовой работы мы выбрали метод решения поставленной задачи, выбрали подходящие структуры данных, разработали алгоритмы программы, реализовали работу на языке высокого уровня C++ с использованием объектно-ориентированного программирования, описали пользовательский интерфейс. Таким образом, мы достигли поставленной цели.

В процессе выполнения курсовой работы мы научились работать с массивами, рассмотрели их преимущества над другими структурами данных; улучшили свои навыки в использовании классов, а также изучили особенности библиотек, использованных в программе.

Список использованных источников

Приложение А

(обязательное)

Листинг программы

Файл «*Curse.cpp*»

```
#include<vcl.h>
#include <iostream>
#include <string>
#include <fstream>
#pragma hdrstop

#include "Curse.h"
//-----
#pragma package(smart_init)
#pragma link "sDialogs"
#pragma resource "*.*.dfm"
TTIME *TIME;
int count = -1;
int q = 0, j = 1, i = 1;
bool fnd = false;
bool sac = false;
std::string p1 = "Apartment";
std::string p2 = "Car";
std::string p3 = "Health";
std::string p4 = "Life";
std::string p5 = "Mortgage";
std::string p6 = "Responsibility";
int row = 0;
//-----
_fastcall TTIME::TTIME(TComponent* Owner)
    : TForm(Owner)
{
Enter->Font->Size = 10;
Delete->Font->Size = 10;
New->Font->Size = 10;
Button2->Font->Size = 10;
Find->Font->Size = 10;
Search->Font->Size = 10;
ShowAllContracts->Font->Size = 10;
StringGrid1->Cells[0][0] = "№";
StringGrid1->Cells[1][0] = "Name";
StringGrid1->Cells[2][0] = "Surname";
StringGrid1->Cells[3][0] = "Number";
StringGrid1->Cells[4][0] = "Passport";
StringGrid1->Cells[5][0] = "Insurance";
StringGrid1->Cells[6][0] = "Price($ in year)";

ComboBox1->Items->Add("Apartment");
ComboBox1->Items->Add("Car");
ComboBox1->Items->Add("Health");
ComboBox1->Items->Add("Life");
ComboBox1->Items->Add("Mortgage");
ComboBox1->Items->Add("Responsibility");

ComboBox4->Items->Add("Apartment");
ComboBox4->Items->Add("Car");
ComboBox4->Items->Add("Health");
ComboBox4->Items->Add("Life");
```

```

ComboBox4->Items->Add("Mortgage");
ComboBox4->Items->Add("Responsibility");
ComboBox4->ItemIndex = 0;

ComboBox3->Items->Add("Name");
ComboBox3->Items->Add("Surname");
ComboBox3->Items->Add("Insurartion");

for (int t = 0; t < 100; t++ ) {mas[t]=NULL; }

ComboBox1->ItemIndex = 0;
ComboBox3->ItemIndex = 0;

}

//-----
void __fastcall TTIME::EnterClick(TObject *Sender)
{
std::string a1 = Edit1->Text.c_str();
std::string a2 = Edit2->Text.c_str();
std::string a3 = Edit3->Text.c_str();
std::string a4 = Edit4->Text.c_str();
    if (a1!="" && a2!="" && a3!="" && a4!="") {
        if (ComboBox1->ItemIndex == 0) {std::string a5 = "80"; mas[q] = new Insurance(a1,a2,a3,a4,p1,a5);q++; }
        if (ComboBox1->ItemIndex == 1) {std::string a5 = "140"; mas[q] = new Insurance(a1,a2,a3,a4,p2,a5);q++; }
        if (ComboBox1->ItemIndex == 2) {std::string a5 = "15"; mas[q] = new Insurance(a1,a2,a3,a4,p3,a5);q++; }
        if (ComboBox1->ItemIndex == 3) {std::string a5 = "30"; mas[q] = new Insurance(a1,a2,a3,a4,p4,a5);q++; }
        if (ComboBox1->ItemIndex == 4) {std::string a5 = "25"; mas[q] = new Insurance(a1,a2,a3,a4,p5,a5);q++; }
        if (ComboBox1->ItemIndex == 5) {std::string a5 = "20"; mas[q] = new Insurance(a1,a2,a3,a4,p6,a5);q++; }
        Edit1->Text = "";
        Edit2->Text = "";
        Edit3->Text = "";
        Edit4->Text = "";
        ComboBox1->ItemIndex = 0;
ShowAllContracts->Click(); } else {Application->MessageBox("Empty blank.", "Error!",NULL); }
}

//-----

void __fastcall TTIME::ShowAllContractsClick(TObject *Sender)
{
sac = false;
while (StringGrid1->RowCount!=1){
StringGrid1->RowCount = StringGrid1->RowCount-1;
}
int j = 1;

for ( intqq=0; qq<100; qq++) {
if (mas[qq]!=NULL) {
    StringGrid1->RowCount = StringGrid1->RowCount + 1;
    StringGrid1->Cells[0][j] = j;
    StringGrid1->Cells[1][j] = mas[qq]->getName().c_str();
    StringGrid1->Cells[2][j] = mas[qq]->getSurname().c_str();
    StringGrid1->Cells[3][j] = mas[qq]->getNumber().c_str();
    StringGrid1->Cells[4][j] = mas[qq]->getPassport().c_str();
    StringGrid1->Cells[5][j] = mas[qq]->getIns().c_str();
}
}
}

```

```

        StringGrid1->Cells[6][j] = mas[qq]->getPrice().c_str();
j++;
    }
}
//-----
void __fastcall TTIME::DeleteClick(TObject *Sender)
{
int u = row-1;
if (row!=0) {
    for (u;u<100;u++) { if (mas[u]!=NULL) {
        mas[u] = mas[u+1];
    } }
} else {Application->MessageBox("Contract not found.", "Error!",NULL); }
ShowAllContracts->Click();
}
//-----

void __fastcall TTIME::FindClick(TObject *Sender)
{
    sac = true;
    while (StringGrid1->RowCount!=1) {
        StringGrid1->RowCount = StringGrid1->RowCount-1;
    }
int i = 1;
    bool flf = false;

    if (Edit6->Text!="") || ComboBox3->ItemIndex == 2)
    {

        if (ComboBox3->ItemIndex == 0) { for (int qqq = 0; qqq<100;qqq++) {
            if ((mas[qqq]!=NULL) && (mas[qqq]->getName() == Edit6->Text.c_str())))
        {
            StringGrid1->RowCount = StringGrid1->RowCount + 1;
            StringGrid1->Cells[0][i] = i;
            StringGrid1->Cells[1][i] = mas[qqq]->getName().c_str();
            StringGrid1->Cells[2][i] = mas[qqq]->getSurname().c_str();
            StringGrid1->Cells[3][i] = mas[qqq]->getNumber().c_str();
            StringGrid1->Cells[4][i] = mas[qqq]->getPassport().c_str();
            StringGrid1->Cells[5][i] = mas[qqq]->getIns().c_str();
            StringGrid1->Cells[6][i] = mas[qqq]->getPrice().c_str();
        }
        i++;
        flf = true;
    }
}

        if (ComboBox3->ItemIndex == 1) { for (int qq2 = 0; qq2<100;qq2++) {
            if ((mas[qq2]!=NULL) && (mas[qq2]->getSurname() == Edit6-
>Text.c_str())) {
                StringGrid1->RowCount = StringGrid1->RowCount + 1;
                StringGrid1->Cells[0][i] = i;
                StringGrid1->Cells[1][i] = mas[qq2]->getName().c_str();
                StringGrid1->Cells[2][i] = mas[qq2]->getSurname().c_str();
                StringGrid1->Cells[3][i] = mas[qq2]->getNumber().c_str();
                StringGrid1->Cells[4][i] = mas[qq2]->getPassport().c_str();
                StringGrid1->Cells[5][i] = mas[qq2]->getIns().c_str();
                StringGrid1->Cells[6][i] = mas[qq2]->getPrice().c_str();
        }
        i++;
        flf = true;
    }
}

```

```

    }
}
if (ComboBox3->ItemIndex == 2) { for (int qq3 = 0; qq3<100;qq3++) {
    if ((mas[qq3]!=NULL) && (ComboBox4->ItemIndex == 0) && (mas[qq3]->getIns() == p1)) {
        StringGrid1->RowCount = StringGrid1->RowCount + 1;
        StringGrid1->Cells[0][i] = i;
        StringGrid1->Cells[1][i] = mas[qq3]->getName().c_str();
        StringGrid1->Cells[2][i] = mas[qq3]->getSurname().c_str();
        StringGrid1->Cells[3][i] = mas[qq3]->getNumber().c_str();
        StringGrid1->Cells[4][i] = mas[qq3]->getPassport().c_str();
        StringGrid1->Cells[5][i] = mas[qq3]->getIns().c_str();
        StringGrid1->Cells[6][i] = mas[qq3]->getPrice().c_str();
    i++;
    flf = true;
} } }

if (ComboBox3->ItemIndex == 2) { for (int qq3 = 0; qq3<100;qq3++) {
    if ((mas[qq3]!=NULL) && (ComboBox4->ItemIndex == 1) && (mas[qq3]->getIns() == p2)) {
        StringGrid1->RowCount = StringGrid1->RowCount + 1;
        StringGrid1->Cells[0][i] = i;
        StringGrid1->Cells[1][i] = mas[qq3]->getName().c_str();
        StringGrid1->Cells[2][i] = mas[qq3]->getSurname().c_str();
        StringGrid1->Cells[3][i] = mas[qq3]->getNumber().c_str();
        StringGrid1->Cells[4][i] = mas[qq3]->getPassport().c_str();
        StringGrid1->Cells[5][i] = mas[qq3]->getIns().c_str();
        StringGrid1->Cells[6][i] = mas[qq3]->getPrice().c_str();
    i++;
    flf = true;
} } }

if (ComboBox3->ItemIndex == 2) { for (int qq3 = 0; qq3<100;qq3++) {
    if ((mas[qq3]!=NULL) && (ComboBox4->ItemIndex == 2) && (mas[qq3]->getIns() == p3)) {
        StringGrid1->RowCount = StringGrid1->RowCount + 1;
        StringGrid1->Cells[0][i] = i;
        StringGrid1->Cells[1][i] = mas[qq3]->getName().c_str();
        StringGrid1->Cells[2][i] = mas[qq3]->getSurname().c_str();
        StringGrid1->Cells[3][i] = mas[qq3]->getNumber().c_str();
        StringGrid1->Cells[4][i] = mas[qq3]->getPassport().c_str();
        StringGrid1->Cells[5][i] = mas[qq3]->getIns().c_str();
        StringGrid1->Cells[6][i] = mas[qq3]->getPrice().c_str();
    i++;
    flf = true;
} } }

if (ComboBox3->ItemIndex == 2) { for (int qq3 = 0; qq3<100;qq3++) {
    if ((mas[qq3]!=NULL) && (ComboBox4->ItemIndex == 3) && (mas[qq3]->getIns() == p4)) {
        StringGrid1->RowCount = StringGrid1->RowCount + 1;
        StringGrid1->Cells[0][i] = i;
        StringGrid1->Cells[1][i] = mas[qq3]->getName().c_str();
        StringGrid1->Cells[2][i] = mas[qq3]->getSurname().c_str();
        StringGrid1->Cells[3][i] = mas[qq3]->getNumber().c_str();
        StringGrid1->Cells[4][i] = mas[qq3]->getPassport().c_str();
        StringGrid1->Cells[5][i] = mas[qq3]->getIns().c_str();
        StringGrid1->Cells[6][i] = mas[qq3]->getPrice().c_str();
    i++;
    flf = true;
} } }

if (ComboBox3->ItemIndex == 2) { for (int qq3 = 0; qq3<100;qq3++) {

```

```

        if ((mas[qq3] !=NULL) && (ComboBox4->ItemIndex == 4) && (mas[qq3]->getIns() == p5)) {
            StringGrid1->RowCount = StringGrid1->RowCount + 1;
            StringGrid1->Cells[0][i] = i;
            StringGrid1->Cells[1][i] = mas[qq3]->getName().c_str();
            StringGrid1->Cells[2][i] = mas[qq3]->getSurname().c_str();
            StringGrid1->Cells[3][i] = mas[qq3]->getNumber().c_str();
            StringGrid1->Cells[4][i] = mas[qq3]->getPassport().c_str();
            StringGrid1->Cells[5][i] = mas[qq3]->getIns().c_str();
            StringGrid1->Cells[6][i] = mas[qq3]->getPrice().c_str();
        }
        i++;
        flf = true;
    } }

    if (ComboBox3->ItemIndex == 2) { for (int qq3 = 0; qq3<100;qq3++) {
        if ((mas[qq3] !=NULL) && (ComboBox4->ItemIndex == 5) && (mas[qq3]->getIns() == p6)) {
            StringGrid1->RowCount = StringGrid1->RowCount + 1;
            StringGrid1->Cells[0][i] = i;
            StringGrid1->Cells[1][i] = mas[qq3]->getName().c_str();
            StringGrid1->Cells[2][i] = mas[qq3]->getSurname().c_str();
            StringGrid1->Cells[3][i] = mas[qq3]->getNumber().c_str();
            StringGrid1->Cells[4][i] = mas[qq3]->getPassport().c_str();
            StringGrid1->Cells[5][i] = mas[qq3]->getIns().c_str();
            StringGrid1->Cells[6][i] = mas[qq3]->getPrice().c_str();
        }
        i++;
        flf = true;
    } }

    //}

    if (!flf) {
        StringGrid1->RowCount = StringGrid1->RowCount + 1;
        StringGrid1->Cells[1][1] = "No match";
        StringGrid1->Cells[2][1] = "No match";
        StringGrid1->Cells[3][1] = "No match";
        StringGrid1->Cells[4][1] = "No match";
        StringGrid1->Cells[5][1] = "No match";
        StringGrid1->Cells[6][1] = "No match";
    }
    Edit6->Text = ""; } else {Application->MessageBox("Empty blank.", "Error!",NULL); }
}

//-----
void __fastcall TTIME::Button2Click(TObject *Sender)
{
    if (row!=0) {
        delete mas[row-1];
        mas[row-1] = NULL;
    std::string b1 = Edit1->Text.c_str();
    std::string b2 = Edit2->Text.c_str();
    std::string b3 = Edit3->Text.c_str();
    std::string b4 = Edit4->Text.c_str();
        if (Edit1->Text!="" && Edit2->Text!="" && Edit3->Text!="" && Edit4->Text!="") {
            if (ComboBox1->ItemIndex == 0) {std::string b5 = "80"; mas[row-1] = new Insurance(b1,b2,b3,b4,p1,b5); }
                if (ComboBox1->ItemIndex == 1) {std::string b5 = "140"; mas[row-1] = new Insurance(b1,b2,b3,b4,p2,b5); }
                    if (ComboBox1->ItemIndex == 2) {std::string b5 = "15"; mas[row-1] = new Insurance(b1,b2,b3,b4,p3,b5); }
    }
}

```

```

        if (ComboBox1->ItemIndex == 3) {std::string b5 = "30"; mas[row-1] =
new Insurance(b1,b2,b3,b4,p4,b5); }
        if (ComboBox1->ItemIndex == 4) {std::string b5 = "25"; mas[row-1] =
new Insurance(b1,b2,b3,b4,p5,b5); }
        if (ComboBox1->ItemIndex == 5) {std::string b5 = "20"; mas[row-1] =
new Insurance(b1,b2,b3,b4,p6,b5); }
ShowAllContracts->Click();
    } else {Application->MessageBox("Empty blank.", "Error!",NULL); }
}
if (Edit1->Text!="" && Edit2->Text!="" && Edit3->Text!="" && Edit4->Text!= "") {
    Edit1->Text = "";
    Edit2->Text = "";
    Edit3->Text = "";
    Edit4->Text = ""; ComboBox1->ItemIndex = 0; }
}
//-----
//-----

void __fastcall TTIME::Save1Click(TObject *Sender)
{
    if (SaveDialog1->Execute()) {
std::ofstreamfout;
fout.open((SaveDialog1->FileName+".txt").c_str(), std::ios_base::out);
    if (fout.is_open()) {
        for (int m = 0; m < 100; m++) {
            if (mas[m]!=NULL) { if (m == 0) {fout<< mas[m]->getName() <<
"\n"<< mas[m]->getSurname() << "\n" << mas[m]->getNumber() << "\n" << mas[m]-
>getPassport() << "\n" << mas[m]->getIns() << "\n" << mas[m]->getPrice() <<
"\n" << "-; }
            else { fout<< "\n" << mas[m]->getName() << "\n" << mas[m]-
>getSurname() << "\n" << mas[m]->getNumber() << "\n" << mas[m]->getPassport()
<< "\n" << mas[m]->getIns() << "\n" << mas[m]->getPrice() << "\n" << "-; }
        }
    }
    }
fout.close();
}
}
//-----

void __fastcall TTIME::Open1Click(TObject *Sender)
{
int r = 0;
    if (OpenDialog1->Execute()) {
        while (StringGrid1->RowCount!=1) {
StringGrid1->Cells[0][1] = "";
StringGrid1->Cells[1][1] = "";
StringGrid1->Cells[2][1] = "";
StringGrid1->Cells[3][1] = "";
StringGrid1->Cells[4][1] = "";
StringGrid1->Cells[5][1] = "";
StringGrid1->Cells[6][1] = "";
StringGrid1->RowCount = StringGrid1->RowCount-1;
}
        for (intyy = 0;yy<100; yy++) {
            if (mas[yy]!=NULL) {
                delete mas[yy];
                mas[yy] = NULL;
            }
        }
    }
std::ifstream fin;

```

```

fin.open(OpenDialog1->FileName.c_str(), std::ios_base::in);

    while (1) {
        mas[r] = new Insurance();
        mas[r]->infile(fin);
        r++;
        q++; if (fin.eof()) {break;}
    }

}
ShowAllContracts->Click();
}

//-----

void __fastcall TTIME::Edit1KeyPress(TObject *Sender, char &Key)
{
if ((Key!= VK_BACK)&&(Key < 'A' || (Key > 'Z' && Key < 'a') || Key > 'z')) )
Key=0;
}

//-----


void __fastcall TTIME::Edit3KeyPress(TObject *Sender, char &Key)
{
if ((Key!= VK_BACK)&&(Key < '0' || Key > '9')) Key=0;

}

//-----


void __fastcall TTIME::Edit2KeyPress(TObject *Sender, char &Key)
{
if ((Key!= VK_BACK)&&(Key < 'A' || (Key > 'Z' && Key < 'a') || Key > 'z')) )
Key=0;
}

//-----


void __fastcall TTIME::Edit4KeyPress(TObject *Sender, char &Key)
{

if ((Key!= VK_BACK)&&(Key < '0' || Key > '9')) Key=0;
}

//-----


void __fastcall TTIME::Edit6KeyPress(TObject *Sender, char &Key)
{
if ((Key!= VK_BACK)&&(Key < 'A' || (Key > 'Z' && Key < 'a') || Key > 'z')) )
Key=0;
}

//-----


void __fastcall TTIME::Exit1Click(TObject *Sender)
{
Close();
}

//-----


void __fastcall TTIME::FormCloseQuery(TObject *Sender, bool &CanClose)
{

```

```

if (Application->MessageBox("You want to close the program?", "TIME", MB_YESNO
| MB_DEFBUTTON2 | MB_ICONQUESTION) == IDNO) {CanClose = false;}
else {if (Application->MessageBox("You want to save changes?", "TIME", MB_YESNO
| MB_DEFBUTTON2 | MB_ICONQUESTION) == IDNO) {CanClose = true;}
else { if (SaveDialog1->Execute()) {
std::ofstream fout;
fout.open(SaveDialog1->FileName.c_str(), std::ios_base::out);
if (fout.is_open()) {
    for (int m = 0; m < 100; m++) {
        if (mas[m] != NULL) { if (m == 0) {fout<< mas[m]->getName() <<
"\n"<< mas[m]->getSurname() << "\n" << mas[m]->getNumber() << "\n" << mas[m]-
>getPassport() << "\n" << mas[m]->getIns() << "\n" << mas[m]->getPrice() <<
"\n" << "-"; }
        else { fout<< "\n" << mas[m]->getName() << "\n" << mas[m]-
>getSurname() << "\n" << mas[m]->getNumber() << "\n" << mas[m]->getPassport()
<< "\n" << mas[m]->getIns() << "\n" << mas[m]->getPrice() << "\n" << "-"; }
    }
}
}
fout.close();
} }

}

//-----
void __fastcall TTIME::StringGrid1Click(TObject *Sender)
{
if (!sac) {
Edit1->Text=mas[StringGrid1->Row-1]->getName().c_str();
Edit2->Text=mas[StringGrid1->Row-1]->getSurname().c_str();
Edit3->Text=mas[StringGrid1->Row-1]->getNumber().c_str();
Edit4->Text=mas[StringGrid1->Row-1]->getPassport().c_str();
if (mas[StringGrid1->Row-1]->getIns() == p1) {ComboBox1->ItemIndex = 0;}
if (mas[StringGrid1->Row-1]->getIns() == p2) {ComboBox1->ItemIndex = 1;}
if (mas[StringGrid1->Row-1]->getIns() == p3) {ComboBox1->ItemIndex = 2;}
if (mas[StringGrid1->Row-1]->getIns() == p4) {ComboBox1->ItemIndex = 3;}
if (mas[StringGrid1->Row-1]->getIns() == p5) {ComboBox1->ItemIndex = 4;}
if (mas[StringGrid1->Row-1]->getIns() == p6) {ComboBox1->ItemIndex = 5;}
row = StringGrid1->Row; }

else {
    for (int it = 0; it<100;it++) {
        if (mas[it] != NULL) {
            if (StringGrid1->Cells[1][StringGrid1->Row] == mas[it]-
>getName().c_str() &&
                StringGrid1->Cells[2][StringGrid1->Row] == mas[it]->getSur-
name().c_str() &&
                StringGrid1->Cells[3][StringGrid1->Row] == mas[it]->getNum-
ber().c_str() &&
                StringGrid1->Cells[4][StringGrid1->Row] == mas[it]->getPass-
port().c_str() &&
                StringGrid1->Cells[5][StringGrid1->Row] == mas[it]-
>getIns().c_str() )
            {
                Edit1->Text=mas[it]->getName().c_str();
                Edit2->Text=mas[it]->getSurname().c_str();
                Edit3->Text=mas[it]->getNumber().c_str();
                Edit4->Text=mas[it]->getPassport().c_str();
                if (mas[it]->getIns() == p1) {ComboBox1->ItemIndex = 0;}
                if (mas[it]->getIns() == p2) {ComboBox1->ItemIndex = 1;}
}
}
}
}
}

```

```

        if (mas[it]->getIns() == p3) {ComboBox1->ItemIndex = 2;}
        if (mas[it]->getIns() == p4) {ComboBox1->ItemIndex = 3;}
        if (mas[it]->getIns() == p5) {ComboBox1->ItemIndex = 4;}
        if (mas[it]->getIns() == p6) {ComboBox1->ItemIndex = 5;}
        row = it+1;
    }

}

}

//-----

void __fastcall TTIME::ComboBox3Change(TObject *Sender)
{
if (ComboBox3->ItemIndex == 2) {Edit6->Visible = false; ComboBox4->Visible =
true;}
if (ComboBox3->ItemIndex == 0 || ComboBox3->ItemIndex == 1) {Edit6->Visible =
true; ComboBox4->Visible = false;}
}

//-----



void __fastcall TTIME::Button1Click(TObject *Sender)
{
Edit1->Text = "";
Edit2->Text = "";
Edit3->Text = "";
Edit4->Text = "";
Edit6->Text = "";
ComboBox1->ItemIndex = 0;
ComboBox4->ItemIndex = 0;
ComboBox3->ItemIndex = 0;
}
//-----

```

Файл «*Curse.h*»

```

#ifndef CurseH
#define CurseH
//-----
#include <Classes.hpp>
#include <Controls.hpp>
#include <Grids.hpp>
#include <StdCtrls.hpp>
#include <Menus.hpp>
#include "sDialogs.hpp"
#include <Dialogs.hpp>
#include <iostream>
#include <string>
#include <fstream>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Grids.hpp>
//-----
class Insurance {
std::string name;
std::string surname;
std::string number;

```

```

    std::string passport;
    std::string ins;
    std::string price;
public:
    Insurance(std::string &n, std::string &surn, std::string &num, std::string
    &pass, std::string &in, std::string &pr) {
        name = n;
        surname = surn;
        number = num;
        passport = pass;
        ins = in;
        price = pr;
    }

    Insurance() {
        name = "";
        surname = "";
        number = "";
        passport = "";
        ins = "";
        price = "";
    }

    ~Insurance () {}

    std::string getName() {return name;}
    std::string getSurname() {return surname;}
    std::string getNumber () {return number;}
    std::string getPassport() {return passport;}
    std::string getIns() {return ins;}
    std::string getPrice() {return price;}

    void setName(std::string n1) {name = n1;}
    void setSurname(std::string sn1) {name = sn1;}
    void setNumber(std::string nul) {name = nul;}
    void setPassport(std::string pp1) {name = pp1;}
    void setIns(std::string insr) {name = insr;}
    void setPrice(std::string prc1) {name = prc1;}

    void infile(std::ifstream&n) {
std::string str6;
        n >> name;
        n >> surname;
        n >> number;
        n >> passport;
        n >> ins;
        n >> price;
        n >> str6;
    }

};

//-----
class TTIME : public TForm
{
__published: // IDE-managed Components
TStringGrid *StringGrid1;
TEdit *Edit1;
TEdit *Edit2;
TEdit *Edit3;
TEdit *Edit4;
TLabel *Name;
TLabel *Surname;
}

```

```

TLabel *Number;
TLabel *Passport;
TComboBox *ComboBox1;
TButton *Enter;
TButton *ShowAllContracts;
TLabel *New;
TButton *Delete;
TLabel *Search;
TComboBox *ComboBox3;
TLabel *pass;
TEdit *Edit6;
TButton *Find;
TButton *Button2;
TLabel *Label4;
TLabel *Label6;
TMainMenu *MainMenuItem;
TMenuItem *FILE1;
TMenuItem *Save1;
TMenuItem *Open1;
TMenuItem *Exit1;
TOpenDialog *OpenDialog1;
TSaveDialog *SaveDialog1;
TComboBox *ComboBox4;
TButton *Button1;
    void __fastcall EnterClick(TObject *Sender);
    void __fastcall ShowAllContractsClick(TObject *Sender);
    void __fastcall DeleteClick(TObject *Sender);
    void __fastcall FindClick(TObject *Sender);
    void __fastcall Button2Click(TObject *Sender);
    void __fastcall Save1Click(TObject *Sender);
    void __fastcall Open1Click(TObject *Sender);
    void __fastcall Edit1KeyPress(TObject *Sender, char &Key);
    void __fastcall Edit3KeyPress(TObject *Sender, char &Key);
    void __fastcall Edit2KeyPress(TObject *Sender, char &Key);
    void __fastcall Edit4KeyPress(TObject *Sender, char &Key);
    void __fastcall Edit6KeyPress(TObject *Sender, char &Key);
    void __fastcall Exit1Click(TObject *Sender);
    void __fastcall FormCloseQuery(TObject *Sender, bool &CanClose);
    void __fastcall StringGrid1Click(TObject *Sender);
    void __fastcall ComboBox3Change(TObject *Sender);
    void __fastcall Button1Click(TObject *Sender);

private:
Insurance* mas[100]; // User declarations
public:           // User declarations
    __fastcall TTIME(TComponent* Owner);
};

//-----
extern PACKAGE TTIME *TIME;
//-----
#endif

```