**Введение**

Описание структуры**:** Вы работаете в спортивном зале и занимаетесь закреплением тренеров за группами занимающихся определенными видами спорта. В Вашем распоряжении имеются тренерах и группах. Каждый тренер получает определенный гонорара за проведенное занятие. При закреплении тренера за группой составляется договор, в котором прописывается дата закрепления тренера и длительность занятий с группой. По окончании занятий рассчитывается и выплачивается тренеру оговоренная сумма.

Консольное приложение “Спортзал” является проектом для коммерческих предприятий, которые призваны помочь людям заниматься спортом. Данный программный продукт помогает сотрудникам коммерческих предприятий рассчитывать зарплату для тренеров, ввести учет групп, составлять договора и хранить все данные в файлах.

Данный программный продукт предусматривает следующие задачи для реализации:

* Добавление записей
* Редактирование записей
* Сортировка записей
* Удаление записей
* Составление договора

Далее приведем краткое описание разделов пояснительной записки.

Первый раздел носит название “Анализ предметной области”. В нем вы сможете ознакомиться с постановкой задачи, которая включает в себя: исследование предметной области поставленной задачи, определение ее организационно-экономической сущности. Также в этом разделе вы сможете узнать о том, как данная задача решается в настоящее время. Все входные и выходные данные тоже будут описаны в первом разделе. В подразделе “Инструменты разработки” будет рассмотрена среда, в которой создается данный проект. Здесь также будут установлены минимальные и оптимальные требования к аппаратным характеристикам, обеспечивающим правильное функционирование поставленной задачей.

В разделе “Проектирование” будут рассмотрены основные аспекты разработки программного продукта. Здесь можно будет узнать об организации данных в контексте среды разработки. В данном разделе будет четко описан пользовательский интерфейс, составлены алгоритмы процесса обработки информации, описана разработка системы справочной информации.

“Построение программы” – это третий раздел пояснительной записки, в котором описываются все элементы и объекты, которые будут использованы при реализации данного приложения. В этом разделе будут четко описаны функции пользователя и их структура. Здесь можно будет найти таблицу, в которой будет представлена полная аннотация файлов, используемых в данном проекте.

Четвертый раздел – “Тестирование”. В нем будет описано полное и функциональное тестирование данной программы, т.е. будет оттестирован каждый пункт меню, каждая операция, которая выполняется приложением. Будут смоделированы все возможные действия пользователя при работе с программой, начиная от запуска до выхода.

В разделе “Применение” будет описано назначение, область применения, среда функционирования программного продукта. Также в нем будет описано использование справочной системы.

“Заключение” будет содержать краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств, описание степени автоматизации процессов на различных этапах разработки.

В “Список использованных источников ” будет приведен список используемых при разработке источников.

В приложениях к пояснительной записке будет приведен листинг программы с необходимыми комментариями.

**1. Анализ предметной области и формулировка требований к программе**

**1. 1. Исследование предметной области**

**Задачи которые должны быть автоматизированы:**

* Добавление записей
* Редактирование записей
* Сортировка записей
* Удаление записей
* Составление договора

**Обоснование необходимости компьютерной обработки информации:**

Процесс накопления, обработки и использования информации постоянно ускоряется. В связи с этим возникает необходимость использования вычислительной техники, позволяющей эффективно хранить, обрабатывать и распределять накопленные данные.

**Описание существующих аналогов:**

Существует множество программных продуктов, подходящих для выполнения поставленных задач по закреплению групп, однако большинство их них — продукты, предназначенные для различного рода задач широкого направления, иногда имеющие слишком большой и ненужный функционал. В новом ПП не будет лишнего функционала, он будет разработан под конкретный узкий круг людей.

**1.2. Инструменты разработки**

Для разработки данного проекта будет выбрана среда разработки Visual Studio, которая является наиболее актуальной средой для создания приложений данного типа. Разработка будет производится на таких языках программирования, как:

* C# - объектно-ориентированный язык программирования, язык разработки приложений для платформы Microsoft .NET Framework.

Иные инструменты, используемые при разработке и написании сопутствующей документации:

* WEB-ресурс DRAW.IO – будет использоваться для создания графической части и разработки UML-диаграмм;
* Microsoft Office Word – для написания документации к программному продукту;
* Smart Install Maker – нужен для создания инсталлятора.

Разработка проекта будет происходить на компьютере со следующими параметрами:

* Процессор Intel® Core i7 – 1065G7 1.30 GHz 1.50 GHz;
* Объём оперативной памяти 8.00 GB;
* Объём места на жестком диске 256 GB;
* Видеокарта Intel® Iris® Plus Graphics;
* ОС Windows 10 Pro.

Таблица 1 - Выбор модели жизненного цикла на основе характеристик требований

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категорий  требований | Каскадная | V-образная | RAD | Инкре- ментная | Быстрого  прототипи- рования | Эволюционная |
| 1. | Являются ли требования к проекту легко определимыми и реализуемыми? | Да | Да | Да | Нет | Нет | Нет |
| 2. | Могут ли требования быть  сформулированы в начале ЖЦ? | Да | Да | Да | Да | Нет | Нет |
| 3. | Часто ли будут изменяться требования на протяжении  ЖЦ? | Нет | Нет | Нет | Нет | Да | Да |
| 4. | Нужно ли демонстрировать требования с целью их определения? | Нет | Нет | Да | Нет | Да | Да |
| 5. | Требуется ли проверка концепции программного  средства или системы? | Нет | Нет | Да | Нет | Да | Да |
| 6. | Будут ли требования изменяться или уточняться с ростом сложности системы (программного средства) в ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 7. | Нужно ли реализовать основные требования на ранних этапах разработки? | Нет | Нет | Да | Да | Да | Да |

Итоги:

* Каскадная 2
* V-образная 2
* RAD 1
* Инкрементная 5
* Быстрого прототипирования 5
* Эволюционная 5

Таблица 2 - Выбор модели жизненного цикла на основе характеристики команды разработчиков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории команды разработчиков проекта | Каскадная | V-образная | RAD | Инкрементная | Быстрого  прототипирования | Эволюционная |
| 1. | Являются ли проблемы предметной области проекта новыми для большинства разработчиков? | Нет | Нет | Нет | Нет | Да | Да |
| 2. | Являются ли инструментальные средства, используемые в проекте, новыми для большинства разработчиков? | Да | Да | Нет | Нет | Нет | Да |
| 3. | Изменяются ли роли участников проекта на протяжении ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 4. | Является ли структура процесса разработки более значимой для разработчиков, чем гибкость? | Да | Да | Нет | Да | Нет | Нет |
| 5. | Важна ли легкость распределения человеческих ресурсов проекта? | Да | Да | Да | Да | Нет | Нет |
| 6. | Приемлет ли команда разработчиков оценки, проверки, стадии разработки? | Да | Да | Нет | Да | Да | Да |

Итоги:

* Каскадная 3
* V-образная 3
* RAD - 2
* Инкрементная 1
* Быстрого прототипирования 4
* Эволюционная 5

Таблица 3 - Выбор модели жизненного цикла на основе характеристик коллектива пользователей

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории коллектива пользователей | Каскадная | V-образная | RAD | Инкре- ментная | Быстрого  прототипирования | Эволюци онная |
| 1. | Будет ли присутствие пользователей ограничено в ЖЦ разработки? | Да | Да | Нет | Да | Нет | Да |
| 2. | Будут ли пользователи оценивать текущее состояние программного продукта (системы) в процессе разработки? | Нет | Нет | Нет | Да | Да | Да |
| 3. | Будут ли пользователи во-  влечены во все фазы ЖЦ разработки? | Нет | Нет | Да | Нет | Да | Нет |
| 4. | Будет ли заказчик отслежи вать ход выполнения проекта? | Нет | Нет | Нет | Нет | Да | Да |

Итоги:

* Каскадная 2
* V-образная 2
* RAD 0
* Инкрементная 3
* Быстрого прототипирования 2
* Эволюционная 4

Таблица 4 - Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № криерия | Критерии категории типов проекта и рисков | Каскадная | V-образная | RAD | Инкрементная | Быстрого  прототипирования | Эволюци онная |
| 1. | Разрабатывается ли в проекте продукт нового для организации направления? | Нет | Нет | Нет | Да | Да | Да |
| 2. | Будет ли проект являться расширением существующей системы? | Да | Да | Да | Да | Нет | Нет |
| Продолжение таблицы 4 | | | | | | | |
| 3. | Будет ли проект крупно или среднемасштабным? | Нет | Нет | Нет | Да | Да | Да |
| 4. | Ожидается ли длительная эксплуатация продукта? | Да | Да | Нет | Да | Нет | Да |
| 5. | Необходим ли высокий  уровень надежности про дукта проекта? | Нет | Да | Нет | Да | Нет | Да |
| 6. | Предполагается ли эволюция продукта проекта в течение ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 7. | Велика ли вероятность изменения системы (продукта) на этапе сопровожде ния? | Нет | Нет | Нет | Да | Да | Да |
| 8. | Является ли график сжатым? | Нет | Нет | Да | Да | Да | Да |
| 9. | Предполагается ли повторное использование компонентов? | Нет | Нет | Да | Да | Да | Да |
| 10. | Являются ли достаточными  ресурсы (время, деньги, инструменты, персонал)? | Нет | Нет | Нет | Нет | Да | Да |

Итоги:

* Каскадная 2
* V-образная 1
* RAD 3
* Инкрементная 7
* Быстрого прототипирования 9
* Эволюционная 9

Итоги:

* Каскадная 9
* V-образная 8
* RAD 6
* Инкрементная 16
* Быстрого прототипирования 20
* Эволюционная 23

Мы выбрали модель с наибольшим количеством отмеченных ответов. Это оказалась эволюционная модель.

**2. Проектирование**

**2.1. Диаграмма вариантов использования**

Диаграмма вариантов использования – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования.

Актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне (рисунок 1). Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик.

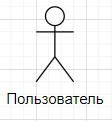


Рисунок 1 - Графическое обозначение актера

Вариант использования является стандартным языком UML и применяется для спецификации общих особенностей системы и любой другой сущности. Отдельные варианты использования обозначаются на диаграмме эллипсом, в котором содержится его краткое название (рисунок 2).



Рисунок 2 - Графическое обозначение отношения ассоциации

Для отображения взаимосвязи экземпляров отдельного варианта использования с более общим вариантом, используется отношение расширения, обозначаемое направленный пунктирной линией со стрелкой от исходного варианта. Данная линия помечается ключевым словом <<extended>> (рисунок 3).

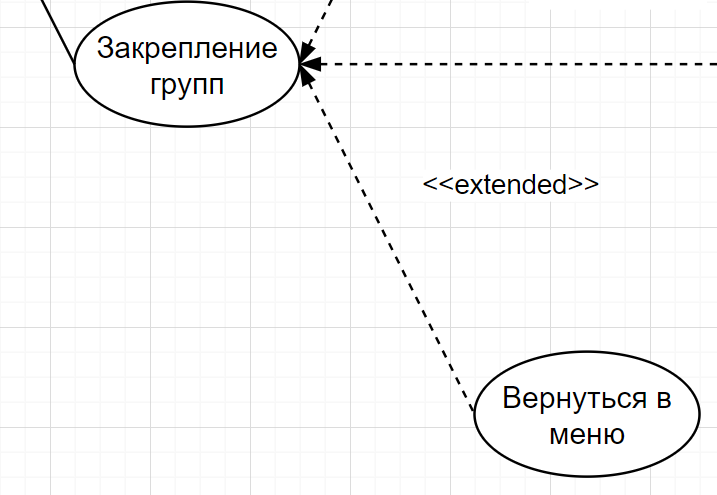


Рисунок 3 - Графическое изображение отношения расширения

Отношение включения между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. Данная линия помечается ключевым словом <<include>>.

Определяя для выбранного актера варианты использования и устанавливая отношения между вариантами использования, получим полную диаграмму вариантов использования, которая будет представлена в приложении Б.

**2.2. Диаграмма деятельности**

Диаграмма деятельности – это, по сути, блок-схема, представляющая поток от одного действия к другому. Деятельность может быть описана как работа системы.

Поток управления передается от одной операции к другой. Этот поток может быть последовательным, разветвленным или параллельным. Диаграммы действий касаются всех типов управления потоком с использованием различных элементов, таких как fork, join и т.д.

Полученная диаграмма деятельности представлена в приложении В.

**2.3. Описание тестов**

Далее представлена таблица в которой приведены тесты для выполнения:

Таблица 5 - Тесты

|  |  |  |
| --- | --- | --- |
| № Теста | Тест | Ожидаемый результат |
| 1 | 2 | 3 |
| 1 | Проверка вкладки меню “Тренера” | При нажатии на клавиатуре клавиши ‘1’, пользователь должен перейти в меню Тренера. |
| 2 | Проверка вкладки меню “Группы” | При нажатии на клавиатуре клавиши ‘2’, пользователь должен перейти в меню Группы. |
| 3 | Проверка вкладки меню “Закрепление групп” | При нажатии на клавиатуре клавиши ‘3’, пользователь должен перейти в меню Закрепления групп. |
| 4 | Проверка вкладки меню “Выход из программы” | При нажатии на клавиатуре клавиши ‘4’, программа должна завершить работу. |
| 5 | Проверка вкладки меню “Список тренеров” | При нажатии на клавиатуре клавиши ‘1’, пользователь должен перейти в меню для просмотра записей тренеров. |
| 6 | Проверка вкладки меню “Добавить тренера” | При нажатии на клавиатуре клавиши ‘2’, должно появиться поле для ввода. |
| 7 | Проверка вкладки меню “Удалить тренера” | При нажатии на клавиатуре клавиши ‘3’, должно появиться поле для ввода. |
| 8 | Проверка вкладки меню “Редактировать тренера” | При нажатии на клавиатуре клавиши ‘4’, пользователь должен перейти в меню редактирования тренера. |
| 9 | Проверка вкладки меню “В главное меню” | При нажатии на клавиатуре клавиши ‘5’, пользователь должен перейти в главное меню. |
| 10 | Проверка вкладки меню “Список групп” | При нажатии на клавиатуре клавиши ‘1’, пользователь должен перейти в меню для просмотра записей групп. |
| 11 | Проверка вкладки меню “Добавить группу” | При нажатии на клавиатуре клавиши ‘2’, должно появиться поле для ввода. |
| 12 | Проверка вкладки меню “Удалить группу” | При нажатии на клавиатуре клавиши ‘3’, должно появиться поле для ввода. |
|  |  |  |
| 1 | 2 | 3 |
| 13 | Проверка вкладки меню “Редактировать группу” | При нажатии на клавиатуре клавиши ‘4’, пользователь должен перейти в меню редактирования группы. |
| 14 | Проверка вкладки меню “В главное меню” | При нажатии на клавиатуре клавиши ‘5’, пользователь должен перейти в главное меню. |
| 15 | Проверка вкладки меню “Список договоров” | При нажатии на клавиатуре клавиши ‘1’, пользователь должен перейти в меню для просмотра списка договоров. |
| 16 | Проверка вкладки меню “Составить договор” | При нажатии на клавиатуре клавиши ‘2’, должно появиться поле для ввода. |
| 17 | Проверка вкладки меню “В главное меню” | При нажатии на клавиатуре клавиши ‘3’, пользователь должен перейти в главное меню. |
| 18 | Проверка сортировки тренеров по гонорару | При нажатии на клавиатуре клавиши ‘1’, должен появиться отсортированный список. |
| 19 | Проверка сортировки тренеров по алфавиту | При нажатии на клавиатуре клавиши ‘2’, должен появиться отсортированный список. |
| 20 | Проверка добавления тренера при вводе корректных данных | После заполнения всех полей должна добавиться новая запись. |
| 21 | Проверка добавления тренера при вводе некорректных данных | При заполнении полей должно появляться сообщение о вводе некорректных данных. |
| 22 | Проверка удаления тренера при вводе существующей записи | После ввода ФИО тренера, введённая запись должна удалиться из списка тренеров. |
| 23 | Проверка удаления тренера при вводе несуществующей записи | После ввода ФИО тренера, должно появляться сообщение о отсутствии введённой записи. |
| 24 | Проверка функции редактирования тренера при вводе корректных данных | После введения ФИО тренера, должно появиться меню для выбора атрибута редактирования, после выбора нужного атрибута и введя данные, запись о тренера должна поменяться. |
| 25 | Проверка функции редактирования тренера при вводе некорректных данных | При введении некорректных данных, должно появляться соответствующее сообщение. |
| 26 | Проверка сортировки групп по возрасту | При нажатии на клавиатуре клавиши ‘1’, должен появиться отсортированный список. |
| 27 | Проверка сортировки тренеров по алфавиту | При нажатии на клавиатуре клавиши ‘2’, должен появиться отсортированный список. |
| 28 | Проверка добавления группы при вводе корректных данных | После заполнения всех полей должна добавиться новая запись. |
| 29 | Проверка добавления группы при вводе некорректных данных | При заполнении полей должно появляться сообщение о вводе некорректных данных. |
| 30 | Проверка удаления группы при вводе существующей записи | После ввода номера группы, введённая запись должна удалиться из списка групп. |
| 31 | Проверка удаления тренера при вводе несуществ. записи | После ввода номера группы, должно появляться сообщение о отсутствии введённой записи. |
| 1 | 2 | 3 |
| 32 | Проверка функции редактирования группы при вводе корректных данных | После введения номера группы, должно появиться меню для выбора атрибута редактирования, после выбора нужного атрибута и введя данные, запись о группе должна поменяться. |
| 33 | Проверка функции редактирования тренера при вводе некорректных данных | При введении некорректных данных, должно появляться соответствующее сообщение. |
| 34 | Проверка сортировки договоров по длительности занятий | При нажатии на клавиатуре клавиши ‘1’, должен появиться отсортированный список. |
| 34 | Проверка сортировки договоров по алфавиту | При нажатии на клавиатуре клавиши ‘2’, должен появиться отсортированный список. |
| 35 | Проверка составления договора при вводе корректных данных | После заполнения всех полей должна добавиться новая запись. |
| 36 | Проверка составления договора при вводе некорректных данных | При заполнении полей должно появляться сообщение о вводе некорректных данных. |

**3. Построение программы**

При разработке программы были определенны следующие классы:

Таблица 6 - перечень классов и их атрибутов.

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование класса | Описание класса | Наименование атрибута | Описание атрибута |
| 1 | 2 | 3 | 4 |
| Trainer | класс содержащий информацию о тренерах, также включающий в себя метод и конструктор. | FullName | атрибут предназначенный для определения информации о полном имени тренера, для атрибута используется тип string |
| Specialization | атрибут предназначенный для определения информации о специализации тренера, для атрибута используется тип string. |
| Qualification | атрибут предназначенный для определения информации о квалификации тренера, для атрибута используется тип string |
| Fee | атрибут предназначенный для определения информации о гонораре тренера, для атрибута используется тип int. |
| Salary | атрибут предназначенный для определения информации о зарплате тренера, для атрибута используется тип double. |
| ClassDuration | атрибут предназначенный для расчета зарплаты тренера по времени проведения занятий, для атрибута используется тип double. |
| Group | класс содержащий информацию о группах, также включающий в себя метод и конструктор. | GroupNumber | атрибут предназначенный для определения номера группы, для атрибута используется тип int. |
| TypeOfOccupation | атрибут предназначенный для определения информации о виде занятий, для атрибута используется тип string. |
| Age | атрибут предназначенный для определения информации о целевом возрасте, для атрибута используется тип int. |
| Status | атрибут предназначенный для определения информации о статусе группы, для атрибута используется тип string. |
| PinningGroups | класс содержащий информацию о договорах, включающий в себя конструктор. | TrainerFullName | атрибут предназначенный для определения информации о полном имени тренера, для атрибута используется тип string. |
| GroupNumber | атрибут предназначенный для определения номера группы, для атрибута используется тип int. |
| TreatyNumber | атрибут предназначенный для определения номера договора, для атрибута используется тип int |
| FixingDate | Атрибут предназначенный для определения даты подписания договора, для атрибута используется тип string. |
| ClassDuration | Атрибут предназначенный для определения длительности занятий, для атрибута используется тип int. |
| 1 | 2 | 3 | 4 |
| Program | Класс содержащий все основные методы для работы с классами и взаимодействиями между ними. | Code | Атрибут предназначенный для выбора вкладки меню, для атрибута используется тип char. |
| Exit | Атрибут предназначенный для проверки состояния меню, для атрибута используется тип bool. |
| Check | Атрибут предназначенный для проверки целочисленных переменный, для атрибута используется тип string. |

При разработке программы были определенны следующие методы:

Таблица 7 - Описание методов в программе

|  |  |  |
| --- | --- | --- |
| Имя метода | Принадлежность к классу | Описание |
| 1 | 2 | 3 |
| Add | Trainer | Запись в файл значений экземпляра класса |
| Add | Group | Запись в файл значений экземпляра класса |
| ShowPinningGroup | PinningGroups | Вывод на экран информации о значениях экземпляров класса |
| Main | Program | Главное меню |
| TrainMenu | Program | Меню для взаимодействия с классом Trainer |
| AddTrainer | Program | Ввод данных для нового экземпляра класса Trainer, и запись в файл |
| GetAllTrainers | Program | Функция для присвоения списку всех записей экземпляров класса Trainer из файла |
| ShowTrainer | Program | Функция выводящая на экран информацию о всех экземпляров класса Trainer с возможностью сортировки |
| RemoveTrainer | Program | Функция для удаления из файла записи соответствующей введённому имени |
| EditTrainer | Program | Функция позволяющая редактировать искомую запись. |
| GroupMenu | Program | Меню для взаимодействия с классом Group |
| AddGroup | Program | Ввод данных нового экземпляра класса Group, и запись в файл |
| GetAllGroups | Program | Функция присвоения списку всех записей экземпляра класса Group из файла |
| ShowGroup | Program | Функция выводящая на экран информацию о всех экземпляров класса Group с возможностью сортировки |
| RemoveGroup | Program | Функция для удаления из файла записи соответствующей введённому номеру группы |
| EditGroup | Program | Функция позволяющая редактировать искомую запись |
| PinningGroupMenu | Program | Меню для взаимодействия с классом PinningGroups |
| AddPinningGroup | Program | Ввод данных нового экземпляра класса PinningGroups, и запись в файл |
| GetAllPinningGroups | Program | Функция для присвоения списку всех записей экземпляров класса PinningGroups из файла |
| 1 | 2 | 3 |
| ShowPinningGroups | Program | Функция выводящая на экран информацию о всех экземпляров класса PinningGroups с возможностью сортировки |

Взаимодействие между классами представлена на диаграмме классов в приложении Г.

**4. Тестирование**

Таблица 8 - Отчёт о результатах тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № Теста | Тест | Фактический результат | Результат тестирования |
| 1 | 2 | 3 | 4 |
| 1 | Проверка вкладки меню “Тренера” | После нажатия на клавиатуре клавиши ‘1’, пользователь переходит в меню Тренера. | Выполнен |
| 2 | Проверка вкладки меню “Группы” | После нажатия на клавиатуре клавиши ‘2’, пользователь переходит в меню Группы. | Выполнен |
| 3 | Проверка вкладки меню “Закрепление групп” | После нажатия на клавиатуре клавиши ‘3’, пользователь переходит в меню Закрепления групп. | Выполнен |
| 4 | Проверка вкладки меню “Выход из программы” | После нажатия на клавиатуре клавиши ‘4’, программа завершает работу. | Выполнен |
| 5 | Проверка вкладки меню “Список тренеров” | После нажатия на клавиатуре клавиши ‘1’, пользователь переходит в меню для просмотра записей тренеров. | Выполнен |
| 6 | Проверка вкладки меню “Добавить тренера” | После нажатия на клавиатуре клавиши ‘2’, появляется поле для ввода. | Выполнен |
| 7 | Проверка вкладки меню “Удалить тренера” | После нажатия на клавиатуре клавиши ‘3’, появляется поле для ввода. | Выполнен |
| 8 | Проверка вкладки меню “Редактировать тренера” | После нажатия на клавиатуре клавиши ‘4’, пользователь переходит в меню редактирования тренера. | Выполнен |
| 9 | Проверка вкладки меню “В главное меню” | После нажатия на клавиатуре клавиши ‘5’, пользователь переходит в главное меню. | Выполнен |
| 10 | Проверка вкладки меню “Список групп” | После нажатия на клавиатуре клавиши ‘1’, пользователь переходит в меню для просмотра записей групп. | Выполнен |
| 11 | Проверка вкладки меню “Добавить группу” | После нажатия на клавиатуре клавиши ‘2’, появляется поле для ввода. | Выполнен |
| 12 | Проверка вкладки меню “Удалить группу” | После нажатия на клавиатуре клавиши ‘3’, появляется поле для ввода. | Выполнен |
| 13 | Проверка вкладки меню “Редактировать группу” | После нажатия на клавиатуре клавиши ‘4’, пользователь переходит в меню редактирования групп. | Выполнен |
| 14 | Проверка вкладки меню “В главное меню” | После нажатия на клавиатуре клавиши ‘5’, пользователь переходит в главное меню. | Выполнен |
| 15 | Проверка вкладки меню “Список договоров” | После нажатия на клавиатуре клавиши ‘1’, пользователь переходит в меню для просмотра списка договоров. | Выполнен |
| 16 | Проверка вкладки меню “Составить договор” | После нажатия на клавиатуре клавиши ‘2’, появляется поле для ввода. | Выполнен |
| 17 | Проверка вкладки меню “В главное меню” | После нажатия на клавиатуре клавиши ‘3’, пользователь должен перейти в главное меню. | Выполнен |
| 1 | 2 | 3 | 4 |
| 18 | Проверка сортировки тренеров по гонорару | После нажатия на клавиатуре клавиши ‘1’, появляется список отсортированный по возрастанию гонораров. | Выполнен |
| 19 | Проверка сортировки тренеров по алфавиту | После нажатия на клавиатуре клавиши ‘2’, появляется список отсортированный по алфавиту. | Выполнен |
| 20 | Проверка добавления тренера при вводе корректных данных | После заполнения всех полей добавляется новая запись. | Выполнен |
| 21 | Проверка добавления тренера при вводе некорректных данных | После заполнении полей появляется сообщение о вводе некорректных данных. | Выполнен |
| 22 | Проверка удаления тренера при вводе существующей записи | После ввода ФИО тренера введённая запись удаляется из списка тренеров. | Выполнен |
| 23 | Проверка удаления тренера при вводе несуществующей записи | После ввода ФИО тренера появляется сообщение о отсутствии введённой записи. | Выполнен |
| 24 | Проверка функции редактирования тренера при вводе корректных данных | После введения ФИО тренера появляется меню для выбора атрибута редактирования, после выбора нужного атрибута и после ввода данных, запись о тренера меняется. | Выполнен |
| 25 | Проверка функции редактирования тренера при вводе некорректных данных | После введении некорректных данных появляется соответствующее сообщение. | Выполнен |
| 26 | Проверка сортировки групп по возрасту | После нажатия на клавиатуре клавиши ‘1’, появляется список отсортированный по возрастанию возраста. | Выполнен |
| 27 | Проверка сортировки тренеров по алфавиту | После нажатия на клавиатуре клавиши ‘2’, появляется список отсортированный по алфавиту. | Выполнен |
| 28 | Проверка добавления группы при вводе корректных данных | После заполнения всех полей добавляется новая запись. | Выполнен |
| 29 | Проверка добавления группы при вводе некорректных данных | После заполнении полей появляться сообщение о вводе некорректных данных. | Выполнен |
| 30 | Проверка удаления группы при вводе существующей записи | После ввода номера группы, введённая запись удаляется из списка групп. | Выполнен |
| 31 | Проверка удаления тренера при вводе несуществующей записи | После ввода номера группы появляется сообщение о отсутствии введённой записи. | Выполнен |
| 32 | Проверка функции редактирования группы при вводе корректных данных | После введения номера группы появляется меню для выбора атрибута редактирования, после выбора нужного атрибута и введя данные, запись о группе меняется. | Выполнен |
| 33 | Проверка функции редактирования тренера при вводе некорректных данных | После введения некорректных данных появляется соответствующее сообщение. | Выполнен |
| 34 | Проверка сортировки договоров по длительности занятий | После нажатия на клавиатуре клавиши ‘1’, появляется список отсортированный по возрастанию длительности занятий. | Выполнен |
| 34 | Проверка сортировки договоров по алфавиту | После нажатия на клавиатуре клавиши ‘2’, появляется список отсортированный по алфавиту. | Выполнен |
| 1 | 2 | 3 | 4 |
| 35 | Проверка составления договора при вводе корректных данных | После заполнения всех полей добавляется новая запись. | Выполнен |
| 36 | Проверка составления договора при вводе некорректных данных | После заполнения полей появляется сообщение о вводе некорректных данных. | Выполнен |

Вывод: все тесты были успешно выполнены, все обнаруженные ошибки исправлены. Весь изначально составленный функционал успешно реализован.

**5. Применение**

**5.1. Назначение и условия применения программы**

Разработанная программа является консольным приложением “Спортзал”.

Данный программный продукт предназначен для сотрудников коммерческого предприятия занимающихся закреплением тренеров за определёнными группами.

Быстродействие любой программы во многом зависит от характеристик выбранного персонального компьютера: рабочей частоты процессора, объёма оперативной памяти и т.д. Несмотря на все реализованные в ней задачи, программа легко запускается и функционирует на любых компьютерах.

Тестирование проводилось на разных классах ЭВМ, но даже на системе следующего типа работать было комфортно:

* процессор Pentium 300 Mhz;
* объём ОЗУ 32 мб;
* видеоподсистема: 1024\*768 точек с глубиной цвета 16 бит;
* операционная система Windows XP sp2

Далее рассмотрим входную и выходную информацию:

Входная информация - та, которую система воспринимает от окружающей среды. Такого рода информация называется входной информацией (по отношению к системе).

Входная информация в данном проекте представлена в таблице 9:

Таблица 9 – входная информация

|  |  |
| --- | --- |
| Название переменной | Информация |
| 1 | 2 |
| FullName | Имя тренера |
| Specialization | Специализация тренера |
| Qualification | Квалификация тренера |
| Fee | Гонорар тренера |
| TypeOfOccupation | Тип занятий группы |
| Age | Целевой возраст группы |
| Status | Статус группы |
| TrainerFullName | Имя тренера |
| FixingDate | Дата подписания договора |
| ClassDuration | Длительность занятий |

Выходная информация (по отношению к окружающей среде) - та, которую система выдает в окружающую среду.

Выходная информация представлена в таблице 10:

Таблица 10 – Выходная информация

|  |  |
| --- | --- |
| Название переменной | Информация |
| Salary | Зарплата тренера |
| GroupNumber | Номер группы |
| TreatyNumber | Номер договора |
| List<Trainer> trainers | Список тренеров |
| List<PinningGroups> pinningGroups | Список договоров |
| List<Group> groups | Список групп |

**5.2. Инсталляция**

Для того чтобы установить программу необходимо запустить файл Setup.exe. Появится окно установки консольного приложения “Спортзал” (Рисунок 4).

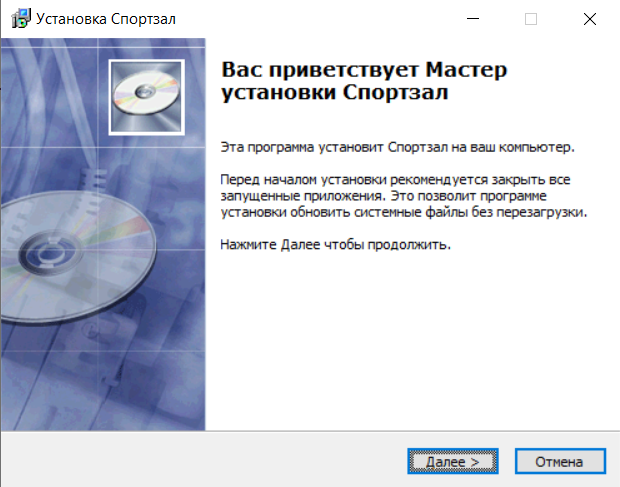


Рисунок 4 - Приветственное меню программы установки

После нажатия на кнопку Далее. Появится окно, где нужно выбрать путь к папке куда установится приложение (Рисунок 5).

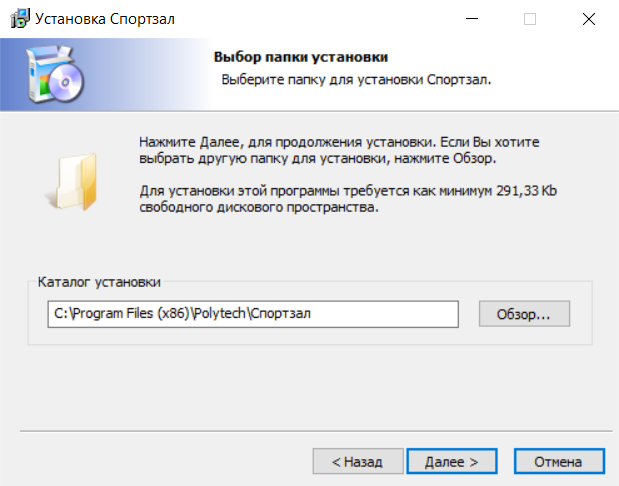


Рисунок 5 - Выбор каталога установки

После того как вы указали путь нажмите Далее. Появится окно, где нужно нажать на кнопку установить (Рисунок 6)

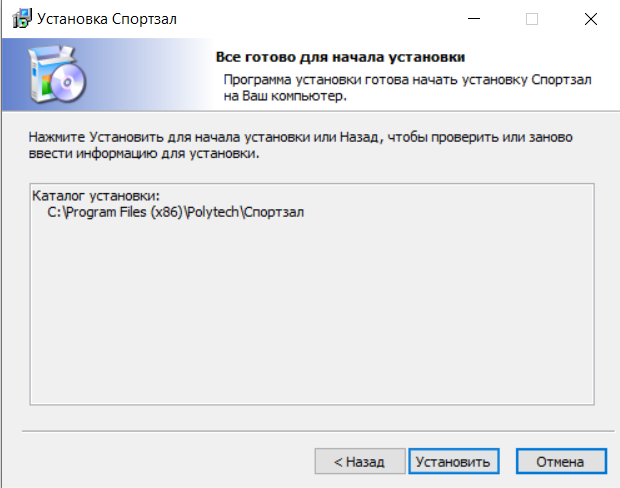


Рисунок 6 - Подтверждение установки

После завершения процесса установки появится окно с сообщением о том, что приложение успешно установлено и нужно нажать на кнопку готово (Рисунок #).

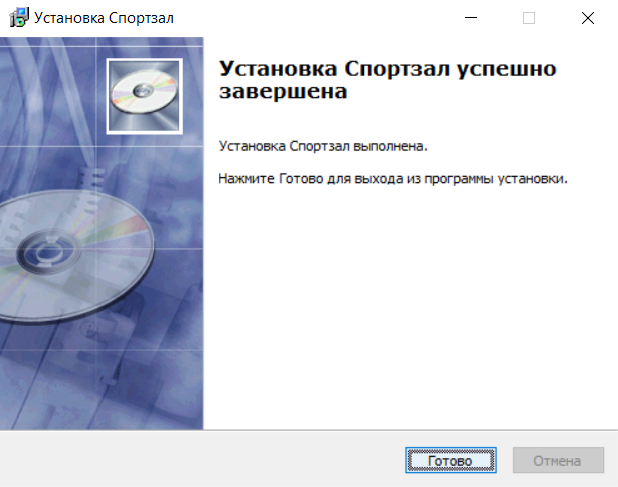
.

Рисунок 7 - Окно завершения установки

**5.3. Выполнение программы**

Для запуска программы необходимо запустить файл Спортзал.exe

После запуска приложения появится окно с главным меню (Рисунок 8)

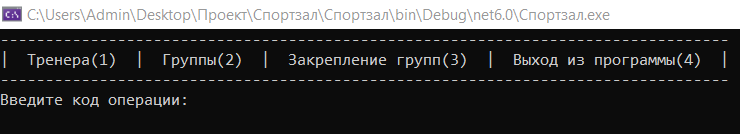


Рисунок 8 - Главное меню

Для навигации по меню пользователь должен вводить код соответствующий коду вкладки меню, так для перехода в вкладку Тренера пользователь должен нажать на клавиатуре клавишу ‘1’. После чего он перейдёт в меню Тренера (Рисунок 9).

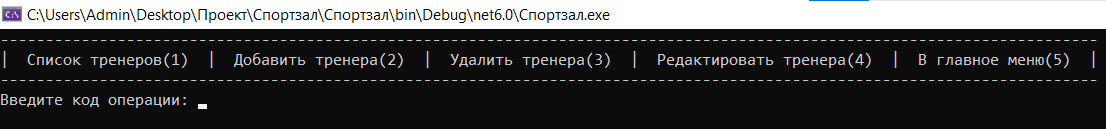


Рисунок 9 - Меню тренеров

При переходе в меню группы, пользователь сможет просмотреть список всех групп, добавить группу, удалить группу, редактировать тренера и вернуться в главное меню (Рисунок 10).

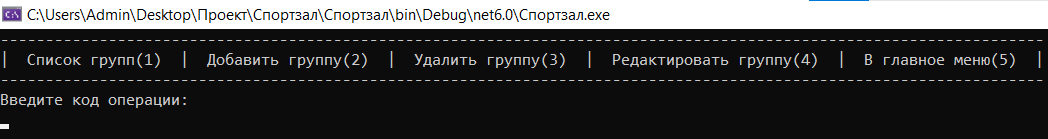


Рисунок 10 - Меню групп

При переходе в меню закрепления групп, пользователь может посмотреть список договоров и составить договор (Рисунок 11).

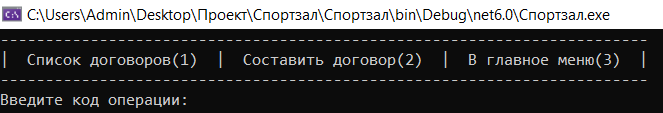


Рисунок 11 - Меню закрепления групп

Для добавления тренера пользователь должен выбрать соответствующую вкладку в меню тренеров, после чего появится поля для ввода информации куда пользователь должен ввести информации о тренере (Рисунок 12).

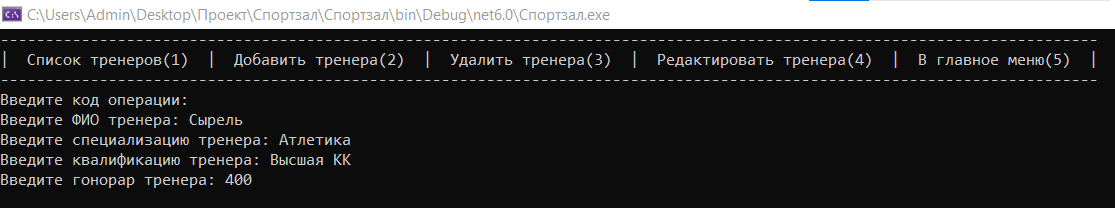


Рисунок 12 - Добавление тренера

После ввода пользователь может выбрать вкладку меню список тренеров для просмотра всех введённых тренеров и информацию о них, после выбора пользователь увидит меню списка тренеров где сможет отсортировать список по гонорару или по алфавиту (Рисунок 13).

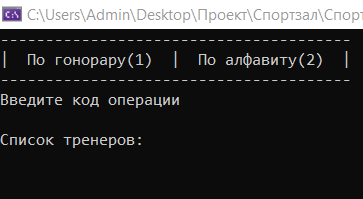


Рисунок 13 - Меню списка тренеров

После выбора нужной операции появится список тренеров, далее для перехода в главное меню нужно нажать на любую клавишу (Рисунок 14).

  
Рисунок 14 - Список тренеров

Для удаления нужного тренера нужно выбрать соответствующую вкладку меню тренеров, после чего появится поле для ввода ФИО тренера, которого пользователь желает удалить из списка (Рисунок 15).

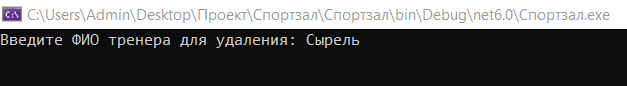
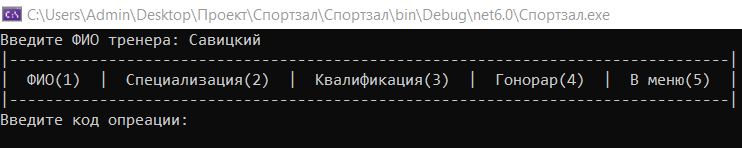
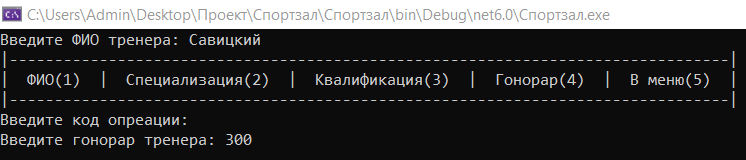


Рисунок 15 - Удаление тренера

У пользователя есть возможность редактировать нужную запись, для чего нужно выбрать соответствующую вкладку меню тренеров, после чего появится поле для ввода ФИО тренера. При вводе существующего тренера появится меню выбора редактирования атрибутов тренера (Рисунок 16).

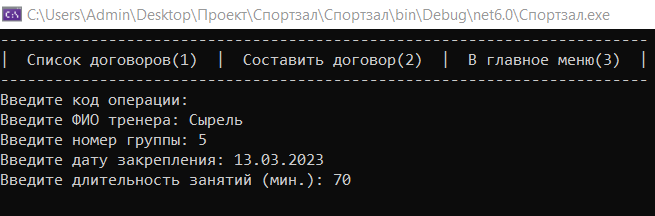
  
Рисунок 16 - Меню редактирования тренера

После чего пользователь может выбрать требующейся ему атрибут и выбрав соответствующую операцию появится поле для ввода, для примера было выбрана вкладка Гонорар (Рисунок 17).

  
Рисунок 17 - Редактирование гонорара

Просмотр списка, добавление, удаление и редактирование в меню групп происходит по тому же принципу, что и в меню тренеров (Рисунки 12 - 17).

В меню закрепления групп пользователь может составить договор, для чего нужно выбрать соответствующую вкладку в меню. При составлении договора нужно вводить уже существующих тренеров, которые были ранее добавлены в список, также существующий номер группы. Нужно помнить, что специализация тренера должна соответствовать виду занятий группы. При корректном вводе всех данных, составится договор по которому будет рассчитываться зарплата тренера (Рисунок 18).

  
Рисунок 18 - Составление договора

Просмотр списка договоров осуществляется по тому же принципу, что и у тренеров или групп, но с другим параметром сортировки.

Для завершения работы с программой нужно закрыть вкладку консоли или нажать на кнопку «4» в главном меню приложения.

**Заключение**

Целью данного проекта была разработка консольного приложения “Спортзал”.

В поставленной задаче был реализован простой и понятный пользовательский интерфейс, в ходе тестирования все исключительные ситуации были обработаны. Проект работает без сбоев и ошибок.

В процессе разработки программного продукта я улучшил свои навыки использования C#, закрепил умение создания собственных функций, научился обрабатывать все исключительные ситуации.

Продукт имеет аккуратный и законченный вид. Можно сказать, что продукт готов к эксплуатации, все изначально поставленные задачи реализованы и протестированы.

Исходя из всего вышеперечисленного, можно сделать выводы, что программа реализована успешно.

**Список использованных источников:**

1. <https://learn.microsoft.com/ru-ru/dotnet/csharp/> - документация по C#
2. <https://www.geeksforgeeks.org/> - руководство по C#
3. <https://metanit.com/sharp/> - руководство по C#
4. <https://www.techiedelight.com/> - коллекция задач по C#

**Приложение А**

**Листинг программы**

class Trainer

{

public string FullName;

public string Specialization;

public string Qualification;

public int Fee;

public double Salary;

public double ClassDuration;

public Trainer(string FullName, string Specialization, string Qualification, int Fee)

{

this.FullName = FullName;

this.Specialization = Specialization;

this.Qualification = Qualification;

this.Fee = Fee;

}

public void Add()

{

StreamWriter sw = new StreamWriter("trainer.txt", true);

sw.WriteLine(FullName);

sw.WriteLine(Specialization);

sw.WriteLine(Qualification);

sw.WriteLine(Fee);

sw.WriteLine("");

sw.Close();

}

}

class Group

{

public int GroupNumber;

public string TypeOfOccupation;

public int Age;

public string Status;

public Group(int GroupNumber, string TypeOfOccupation, int Age, string Status)

{

this.GroupNumber = GroupNumber;

this.TypeOfOccupation = TypeOfOccupation;

this.Age = Age;

this.Status = Status;

}

public void Add()

{

StreamWriter sw = new StreamWriter("group.txt", true);

sw.WriteLine(GroupNumber);

sw.WriteLine(TypeOfOccupation);

sw.WriteLine(Age);

sw.WriteLine(Status);

sw.WriteLine("");

sw.Close();

}

}

class PinningGroups

{

public string TrainerFullName;

public int GroupNumber;

public int TreatyNumber;

public string FixingDate;

public double ClassDuration;

public PinningGroups(string TrainerFullName, int GroupNumber, int TreatyNumber, string FixingDate, int ClassDuration)

{

this.TrainerFullName = TrainerFullName;

this.GroupNumber = GroupNumber;

this.TreatyNumber = TreatyNumber;

this.FixingDate = FixingDate;

this.ClassDuration = ClassDuration;

}

public void ShowPinningGroup()

{

var ts = TimeSpan.FromMinutes(ClassDuration);

Console.WriteLine("---------------------------------------------\n" +

$"Имя тренера: {TrainerFullName}\n" +

$"№ группы: {GroupNumber}\n" +

$"№ договора: {TreatyNumber}\n" +

$"Дата закрепления: {FixingDate}\n" +

"Длительность занятий: {0} ч. {1} мин.", ts.Hours, ts.Minutes);

Console.WriteLine("---------------------------------------------");

}

static void Main()

{

Console.Clear();

Console.WindowHeight = Console.LargestWindowHeight;

Console.WindowWidth = Console.LargestWindowWidth;

Console.WriteLine("---------------------------------------------------------------------------------\n" +

"| Тренера(1) | Группы(2) | Закрепление групп(3) | Выход из программы(4) |\n" +

"---------------------------------------------------------------------------------");

Console.Write("Введите код операции: ");

char Code = '0';

while (Code != '4')

{

Code = Console.ReadKey(true).KeyChar;

if (Code == '1')

{

TrainMenu();

break;

}

if (Code == '2')

{

GroupMenu();

break;

}

if (Code == '3')

{

PinningGroupMenu();

break;

}

}

}

public static void TrainMenu()

{

Console.Clear();

Console.WriteLine("--------------------------------------------------------------------------------------------------------------------------\n" +

"| Список тренеров(1) | Добавить тренера(2) | Удалить тренера(3) | Редактировать тренера(4) | В главное меню(5) |\n" +

"--------------------------------------------------------------------------------------------------------------------------");

Console.Write("Введите код операции: ");

bool Exit = true; char Code;

while (Exit)

{

Code = Console.ReadKey(true).KeyChar;

if (Code == '1')

{

ShowTrainer();

TrainMenu();

Exit = false;

}

if (Code == '2')

{

AddTrainer();

TrainMenu();

Exit = false;

}

if (Code == '3')

{

RemoveTrainer();

TrainMenu();

Exit = false;

}

if (Code == '4')

{

EditTrainer();

TrainMenu();

Exit = false;

}

if (Code == '5')

{

Main();

Exit = false;

}

}

}

public static void AddTrainer()

{

C0:

Console.Write("\nВведите ФИО тренера: ");

string FullName = Console.ReadLine();

if (FullName.Any(c => char.IsNumber(c)) || FullName.Any(c => char.IsPunctuation(c)))

{

Console.WriteLine("В данной строке не может быть цифр или спец. символов\n");

goto C0;

}

C1:

Console.Write("Введите специализацию тренера: ");

string Specialization = Console.ReadLine();

if (Specialization.Any(c => char.IsNumber(c)) || Specialization.Any(c => char.IsPunctuation(c)))

{

Console.WriteLine("В данной строке не может быть цифр или спец. символов\n");

goto C1;

}

C2:

Console.Write("Введите квалификацию тренера: ");

string Qualification = Console.ReadLine();

if (Qualification.Any(c => char.IsNumber(c)) || Qualification.Any(c => char.IsPunctuation(c)))

{

Console.WriteLine("В данной строке не может быть цифр или спец. символов\n");

goto C2;

}

Check:

Console.Write("Введите гонорар тренера: ");

string Check = Console.ReadLine();

if (Check.Any(c => char.IsLetter(c)) || Check.Any(c => char.IsPunctuation(c)) || Check.Any(c => char.IsSeparator(c)))

{

Console.WriteLine("Некорректно введённое число\n");

goto Check;

}

int Fee = int.Parse(Check);

if (Fee < 0)

{

Console.WriteLine("Введённое число не может быть отрицательным\n");

goto Check;

}

Trainer trainer = new Trainer(FullName, Specialization, Qualification, Fee);

if (File.Exists("trainer.txt"))

{

StreamWriter sw = new StreamWriter("trainer.txt", true);

sw.WriteLine(FullName);

sw.WriteLine(Specialization);

sw.WriteLine(Qualification);

sw.WriteLine(Fee);

sw.WriteLine("");

sw.Close();

}

else

{

using (FileStream fs = File.Create("trainer.txt")) ;

StreamWriter sw = new StreamWriter("trainer.txt", true);

sw.WriteLine(FullName);

sw.WriteLine(Specialization);

sw.WriteLine(Qualification);

sw.WriteLine(Fee);

sw.WriteLine("");

sw.Close();

}

}

public static List<Trainer> GetAllTrainers()

{

StreamReader reader = new StreamReader("trainer.txt");

List<Trainer> trainers = new List<Trainer>();

List<string> info = new List<string>();

while (!reader.EndOfStream)

{

string infoLine = reader.ReadLine();

if (infoLine == "")

{

trainers.Add(new Trainer(info[0], info[1], info[2], int.Parse(info[3])));

info.Clear();

}

else

{

info.Add(infoLine);

}

}

reader.Close();

return trainers;

}

public static void ShowTrainer()

{

Console.Clear();

if (File.Exists("trainer.txt"))

{

List<Trainer> trainers = GetAllTrainers();

if (trainers.Count == 0)

{

Console.WriteLine("Нет тренеров");

Console.WriteLine("\nНажмите любую клавишу, чтобы продолжить...");

Console.ReadKey();

}

else

{

Console.WriteLine("---------------------------------------\n" +

"| По гонорару(1) | По алфавиту(2) |\n" +

"---------------------------------------");

Console.WriteLine("Введите код операции\n");

Console.WriteLine("Список тренеров:");

char Code = Console.ReadKey(true).KeyChar;

if (Code == '1')

{

trainers.Sort(delegate (Trainer x, Trainer y) {

return x.Fee.CompareTo(y.Fee);

});

}

if (Code == '2')

{

trainers.Sort(delegate (Trainer x, Trainer y) {

return x.FullName.CompareTo(y.FullName);

});

}

if (File.Exists("PinningGroup.txt"))

{

List<PinningGroups> pinningGroups = GetAllPinningGroups();

foreach (PinningGroups pinningGroup in pinningGroups)

{

foreach (Trainer trainer in trainers)

{

if (pinningGroup.TrainerFullName == trainer.FullName)

{

trainer.ClassDuration = trainer.ClassDuration + pinningGroup.ClassDuration;

trainer.Salary = (trainer.ClassDuration / 60) \* trainer.Fee;

}

}

}

}

foreach (Trainer trainer in trainers)

{

Console.WriteLine("---------------------------------------------\n" +

$"Имя: {trainer.FullName}\n" +

$"Специализация: {trainer.Specialization}\n" +

$"Квалификация: {trainer.Qualification}\n" +

$"Гонорар: {trainer.Fee} руб.\n" +

$"Зарплата: {trainer.Salary:F2} руб.\n" +

"---------------------------------------------");

}

Console.WriteLine("Нажмите любую клавишу, чтобы продолжить...");

Console.ReadKey();

}

}

else

{

Console.WriteLine("Нет тренеров для просмотра\n\nНажмите любую клавишу, чтобы продолжить...");

Console.ReadKey();

}

}

public static void RemoveTrainer()

{

Console.Clear();

List<Trainer> trainers = GetAllTrainers();

C0:

Console.Write("Введите ФИО тренера для удаления: ");

string Name = Console.ReadLine();

if (Name.Any(c => char.IsNumber(c)) || Name.Any(c => char.IsPunctuation(c)))

{

Console.WriteLine("В данной строке не может быть цифр или спец. символов\n");

goto C0;

}

Trainer searchtrainers = trainers.Find(trainer => trainer.FullName == Name);

if (searchtrainers != null)

{

trainers.Remove(searchtrainers);

StreamWriter sw = new StreamWriter("trainer.txt", false);

sw.Close();

foreach (Trainer trainer in trainers)

{

trainer.Add();

}

}

else

{

Console.WriteLine("Нет тренеров для удаления");

Console.WriteLine("\nНажмите любую клавишу, чтобы продолжить...");

Console.ReadKey();

}

}

public static void EditTrainer()

{

Console.Clear();

List<Trainer> trainers = GetAllTrainers();

C0:

Console.Write("Введите ФИО тренера: ");

string Name = Console.ReadLine();

if (Name.Any(c => char.IsNumber(c)) || Name.Any(c => char.IsPunctuation(c)))

{

Console.WriteLine("В данной строке не может быть цифр или спец. символов\n");

goto C0;

}

Trainer search = trainers.Find(trainer => trainer.FullName == Name);

if (search != null)

{

int index = trainers.IndexOf(search);

Console.WriteLine("|——————————————————————————————————————————————————————————————————--------------|\n" +

"| ФИО(1) | Специализация(2) | Квалификация(3) | Гонорар(4) | В меню(5) |\n" +

"|——————————————————————————————————————————————————————————————————--------------|");

Console.Write("Введите код опреации: \n");

bool Exit = true; char Code;

while (Exit)

{

Code = Console.ReadKey(true).KeyChar;

if (Code == '1')

{

C1:

Console.Write("Введите ФИО тренера: ");

search.FullName = Console.ReadLine();

if (search.FullName.Any(c => char.IsNumber(c)) || search.FullName.Any(c => char.IsPunctuation(c)))

{

Console.WriteLine("В данной строке не может быть цифр или спец. символов\n");

goto C1;

}

Exit = false;

}

if (Code == '2')

{

C2:

Console.Write("Введите специализацию тренера: ");

search.Specialization = Console.ReadLine();

if (search.Specialization.Any(c => char.IsNumber(c)) || search.Specialization.Any(c => char.IsPunctuation(c)))

{

Console.WriteLine("В данной строке не может быть цифр или спец. символов\n");

goto C2;

}

Exit = false;

}

if (Code == '3')

{

C3:

Console.Write("Введите квалификацию тренера: ");

search.Qualification = Console.ReadLine();

if (search.Qualification.Any(c => char.IsNumber(c)) || search.Qualification.Any(c => char.IsPunctuation(c)))

{

Console.WriteLine("В данной строке не может быть цифр или спец. символов\n");

goto C3;

}

Exit = false;

}

if (Code == '4')

{

Check:

Console.Write("Введите гонорар тренера: ");

string Check = Console.ReadLine();

if (Check.Any(c => char.IsLetter(c)) || Check.Any(c => char.IsPunctuation(c)) || Check.Any(c => char.IsSeparator(c)))

{

Console.WriteLine("Некорректно введённое число\n");

goto Check;

}

search.Fee = int.Parse(Check);

if (search.Fee < 0)

{

Console.WriteLine("Введённое число не может быть отрицательным\n");

goto Check;

}

Exit = false;

}

if (Code == '5')

{

Exit = false;

}

}

trainers[index] = search;

StreamWriter sw = new StreamWriter("trainer.txt", false);

sw.Close();

foreach (Trainer trainer in trainers)

{

trainer.Add();

}

}

else

{

Console.WriteLine("Данного тренера нет в списке, попробуйте ещё раз");

Console.ReadKey();

TrainMenu();

}

}

public static void AddPinningGroups()

{

List<Group> groups = GetAllGroups();

List<Trainer> trainers = GetAllTrainers();

Check:

bool flag = true;

bool Checked = false;

string Specialization;

string TypeOfOccupation;

C0:

Console.Write("Введите ФИО тренера: ");

string TrainerFullName = Console.ReadLine();

if (TrainerFullName.Any(c => char.IsNumber(c)) || TrainerFullName.Any(c => char.IsPunctuation(c)))

{

Console.WriteLine("В данной строке не может быть цифр или спец. символов\n");

goto C0;

}

check:

Console.Write("Введите номер группы: ");

string Check = Console.ReadLine();

if (Check.Any(c => char.IsLetter(c)) || Check.Any(c => char.IsPunctuation(c)) || Check.Any(c => char.IsSeparator(c)))

{

Console.WriteLine("Некорректно введённое число\n");

goto check;

}

int GroupNumber = int.Parse(Check);

if (GroupNumber < 0)

{

Console.WriteLine("Введённое число не может быть отрицательным\n");

goto check;

}

while (flag)

{

foreach (Group group in groups)

{

foreach (Trainer trainer in trainers)

{

if (TrainerFullName == trainer.FullName && GroupNumber == group.GroupNumber)

{

flag = false;

Checked = true;

TypeOfOccupation = group.TypeOfOccupation;

Specialization = trainer.Specialization;

if (Specialization != TypeOfOccupation)

{

Console.WriteLine("Специализация тренера не подходит к группе \n");

goto Check;

}

}

}

}

flag = false;

if (Checked == false)

{

Console.WriteLine("Нет такой группы или тренера \n");

goto Check;

}

}

CheckDate:

Console.Write("Введите дату закрепления: ");

string FixingDate = Console.ReadLine();

DateTime dDate;

if (DateTime.TryParse(FixingDate, out dDate))

{

String.Format("{0:d/MM/yyyy}", dDate);

}

else

{

Console.WriteLine("Некорректная дата\n");

goto CheckDate;

}

Check1:

Console.Write("Введите длительность занятий (мин.): ");

string Check1 = Console.ReadLine();

if (Check1.Any(c => char.IsLetter(c)) || Check1.Any(c => char.IsPunctuation(c)) || Check1.Any(c => char.IsSeparator(c)))

{

Console.WriteLine("Некорректно введённое число\n");

goto Check1;

}

int ClassDuration = int.Parse(Check1);

if (ClassDuration < 0)

{

Console.WriteLine("Введённое число не может быть отрицательным\n");

goto Check1;

}

List<PinningGroups> pinningGroups = GetAllPinningGroups();

int TreatyNumber = 1;

if (pinningGroups.Count != 0)

{

int i = 1;

foreach (PinningGroups group in pinningGroups)

{

if (pinningGroups.Count == i)

TreatyNumber = i + 1;

if (group.TreatyNumber == i)

i++;

else

TreatyNumber = i;

}

}

if (File.Exists("PinningGroup.txt"))

{

StreamWriter sw = new StreamWriter("PinningGroup.txt", true);

sw.WriteLine(TrainerFullName);

sw.WriteLine(GroupNumber);

sw.WriteLine(TreatyNumber);

sw.WriteLine(FixingDate);

sw.WriteLine(ClassDuration);

sw.WriteLine("");

sw.Close();

}

else

{

using (FileStream fs = File.Create("PinningGroup.txt")) ;

StreamWriter sw = new StreamWriter("PinningGroup.txt", true);

sw.WriteLine(TrainerFullName);

sw.WriteLine(GroupNumber);

sw.WriteLine(TreatyNumber);

sw.WriteLine(FixingDate);

sw.WriteLine(ClassDuration);

sw.WriteLine("");

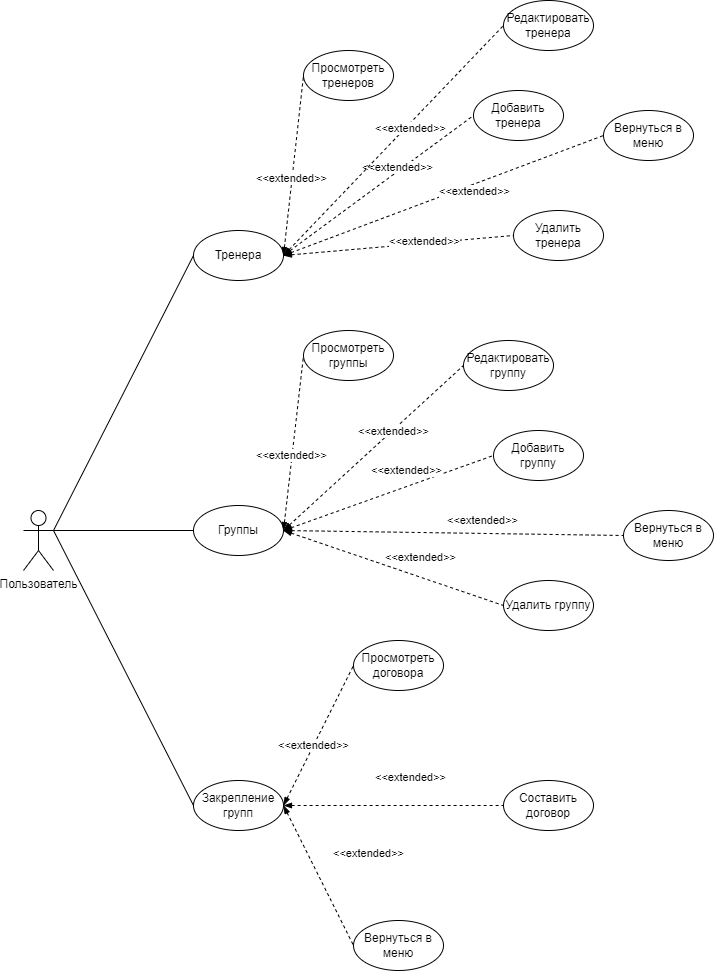
sw.Close();

}

}

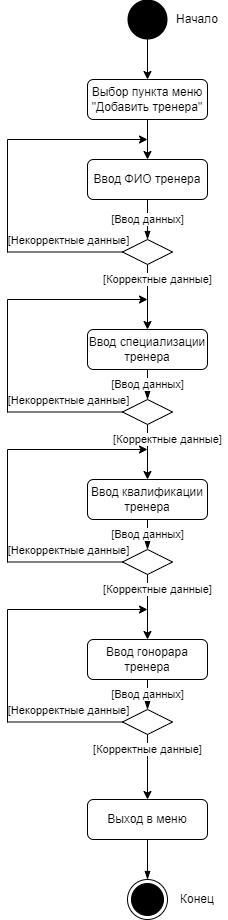
**Приложение Б**

**Диаграмма вариантов использования**



**Приложение В**

**Диаграмма деятельности**

****

**Приложение Г**

**Диаграмма классов**

