Учреждение высшего образования

«Университет управления «ТИСБИ»

Факультет информационных технологий

Кафедра информационных технологий

КУРСОВАЯ РАБОТА

по дисциплине «Программная инженерия»

на тему: «Разработка объектной программы для учета налогов физических лиц»

Выполнил: студент гр. ПИ-022

Фамилия И.О.

Проверил: ст. преподаватель

Якунина Е.А.

Казань 2022

Оглавление

[Постановка задачи 3](#_Toc105329067)

[1. Описание используемых структур данных с алгоритмами выполнения основных операций 5](#_Toc105329068)

[1.1 Динамическая очередь. 5](#_Toc105329069)

[1.2 Адресный однонаправленный список. 6](#_Toc105329070)

[2. Краткие сведения об объектном подходе 8](#_Toc105329071)

[3. Формализованное описание разработанных классов 14](#_Toc105329072)

[4. Описание демонстрационного модуля с характеристикой использованных стандартных компонентов и списком реализованных обработчиков 21](#_Toc105329073)

[5. Описание структуры проекта в соответствии с использованным инструментом разработки 28](#_Toc105329074)

[Список литературы 30](#_Toc105329075)

[Листинг программы 32](#_Toc105329076)

Постановка задачи

**Цели работы:** отработка навыков курсов «программная инженерия» и «структуры и алгоритмы обработки данных».

**Постановка задачи:** разработать объектную программу для хранения и обработки данных об уплате налогов физическими лицами. Программа должна поддерживать список налогоплательщиков с указанием уникального ИНН и фамилии. Для каждого налогоплательщика создается свой список уплаченных налогов с указанием вида налога (квартира, дом, земля, дача, автомобиль) и его сумм.

Разработка программы включает в себя:

* определение необходимых объектов и способов их взаимодействия;
* формальное описание объектов в виде классов;
* программную реализацию всех необходимых методов, включая корректировку вкладов и подсчет текущего суммарного объема всех вкладов;
* всестороннее тестирование методов с помощью консольного (при разработке) и оконного (в окончательном варианте) приложения.

Для объединения налогоплательщиков используется структура данных в виде адресного замкнутого упорядоченного однонаправленного списка с заголовком. Для объединения налогов у каждого налогоплательщика используется стек на основе динамического массива.

Разработка выполняется с учетом следующих требований:

1. имена классов, свойств и методов должны носить содержательный смысл и соответствовать информационной задаче
2. обязательное соблюдение принципа инкапсуляции – использование в классах только закрытых свойств и реализация необходимого набора методов доступа
3. наличие двух методов для сохранения всей объектной структуры во внешнем файле с обратной загрузкой, при этом стандартные механизмы сериализации разрешается использовать только как дополнение к самостоятельно реализованным методам
4. тестовое оконное приложение должно обладать удобным пользовательским интерфейсом с контролем вводимых данных и отображением текущего состояния объектной структуры с помощью списковых или табличных компонентов
5. стандартные контейнеры/коллекции (включая обобщенные классы) разрешается использовать только как дополнение к самостоятельно разработанным классам
6. в качестве языка разработки разрешается использовать Java, С, C++, Object/Free Pascal и соответствующие инструменты быстрой разработки приложений.
7. **Описание используемых структур данных с алгоритмами выполнения основных операций**

1.1 Динамическая очередь.

Очередь – это динамическая структура данных, которая состоит из набора элементов которые размещены последовательно друг за другом. При этом добавление элементов осуществляется с одной стороны, а удаление (вытягивание) – с другой стороны.

Очередь работает по принципу FIFO (First In — First Out), то есть «первым пришел – первым вышел». На рисунке 1 отображен принцип работы очереди.

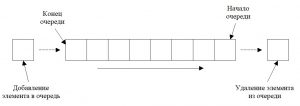


Рисунок 1. Динамическая структура данных «очередь»

Одной из особенностей очереди есть то, что в очереди часть последовательных элементов может быть занята на данный момент времени (рисунок 2).

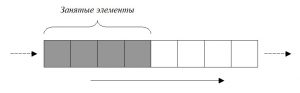


Рисунок 2. Очередь, в которой часть элементов занята на данный момент времени

В повседневной жизни использование очереди встречается довольно часто, например:

очередь в магазине;

очередь документов на печать на принтере;

последовательность операторов в алгоритме которые выполняются друг за другом;

другие случаи.

Виды очередей

Различают следующие виды очередей:

простая очередь;

кольцевая очередь. В такой очереди элемент, который выходит с начала очереди, будет помещен в ее конец;

очередь с приоритетами. В такой очереди элементы размещаются по их приоритетам (весовым коэффициентам). Первым из очереди выходит элемент с наивысшим приоритетом.

Любой вид очереди может иметь ограничение по размеру (количество элементов в очереди).

Способы реализации очереди

В программе очередь можно реализовывать в виде:

статического массива с ограничением на размер в очереди;

динамического массива;

односвязного списка;

двусвязного списка.

1.2 Адресный однонаправленный список.

Однонаправленный (односвязный) список – это структура данных, представляющая собой последовательность элементов, в каждом из которых хранится значение и указатель на следующий элемент списка ( рис. 3). В последнем элементе указатель на следующий элемент равен NULL.

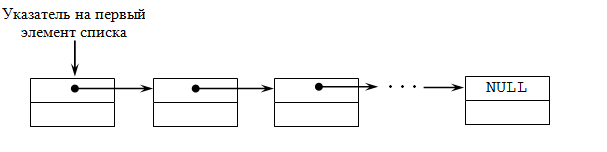


Рис. 3. Линейный однонаправленный список

Основными операциями, осуществляемыми с однонаправленными списками, являются:

* создание списка;
* печать (просмотр) списка;
* вставка элемента в список;
* удаление элемента из списка;
* поиск элемента в списке
* проверка пустоты списка;
* удаление списка.

Особое внимание следует обратить на то, что при выполнении любых операций с линейным однонаправленным списком необходимо обеспечивать позиционирование какого-либо указателя на первый элемент. В противном случае часть или весь список будет недоступен.

1. Краткие сведения об объектном подходе

Объектно-ориентированное программирование (ООП) – это концепция программирования, которая работает на принципах абстракции, инкапсуляции, наследования и полиморфизма. Это позволяет пользователям создавать объекты, которые они хотят, а затем создавать методы для обработки этих объектов. Основная концепция ООП заключается в создании объектов, их повторном использовании в программе и манипулировании этими объектами для получения результатов.

Абстракция

Абстра́кция — в объектно-ориентированном программировании это придание объекту характеристик, которые отличают его от всех объектов, четко определяя его концептуальные границы. Основная идея состоит в том, чтобы отделить способ использования составных объектов данных от деталей их реализации в виде более простых объектов, подобно тому, как функциональная абстракция разделяет способ использования функции и деталей её реализации в терминах более примитивных функций, таким образом, данные обрабатываются функцией высокого уровня с помощью вызова функций низкого уровня.

Такой подход является основой объектно-ориентированного программирования. Это позволяет работать с объектами, не вдаваясь в особенности их реализации. В каждом конкретном случае применяется тот или иной подход: инкапсуляция, полиморфизм или наследование. Например, при необходимости обратиться к скрытым данным объекта, следует воспользоваться инкапсуляцией, создав, так называемую, функцию доступа или свойство.

Абстракция данных — популярная и в общем неверно определяемая техника программирования. Фундаментальная идея состоит в разделении несущественных деталей реализации подпрограммы и характеристик существенных для корректного ее использования. Такое разделение может быть выражено через специальный «интерфейс», сосредотачивающий описание всех возможных применений программы [1].

С точки зрения теории множеств, процесс представляет собой организацию для группы подмножеств своего множества. См. также Закон обратного отношения между содержанием и объемом понятия.

Инкапсуляция

Инкапсуляция — свойство программирования, позволяющее пользователю не задумываться о сложности реализации используемого программного компонента (что у него внутри?), а взаимодействовать с ним посредством предоставляемого интерфейса (публичных методов и членов), а также объединить и защитить жизненно важные для компонента данные. При этом пользователю предоставляется только спецификация (интерфейс) объекта.

Пользователь может взаимодействовать с объектом только через этот интерфейс. Реализуется с помощью ключевого слова: public.

Пользователь не может использовать закрытые данные и методы. Реализуется с помощью ключевых слов: private, protected, internal.))

Инкапсуляция — один из четырёх важнейших механизмов объектно-ориентированного программирования (наряду с абстракцией, полиморфизмом и наследованием).

Сокрытие реализации целесообразно применять в следующих случаях:

предельная локализация изменений при необходимости таких изменений,

прогнозируемость изменений (какие изменения в коде надо сделать для заданного изменения функциональности) и прогнозируемость последствий изменений.

Наследование

Наследование — один из четырёх важнейших механизмов объектно-ориентированного программирования (наряду с инкапсуляцией, полиморфизмом и абстракцией), позволяющий описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом.

Другими словами, класс-наследник реализует спецификацию уже существующего класса (базовый класс). Это позволяет обращаться с объектами класса-наследника точно так же, как с объектами базового класса.

Простое наследование:

Класс, от которого произошло наследование, называется базовым или родительским (англ. base class). Классы, которые произошли от базового, называются потомками, наследниками или производными классами (англ. derived class).

В некоторых языках используются абстрактные классы. Абстрактный класс — это класс, содержащий хотя бы один абстрактный метод, он описан в программе, имеет поля, методы и не может использоваться для непосредственного создания объекта. То есть от абстрактного класса можно только наследовать. Объекты создаются только на основе производных классов, наследованных от абстрактного. Например, абстрактным классом может быть базовый класс «сотрудник вуза», от которого наследуются классы «аспирант», «профессор» и т. д. Так как производные классы имеют общие поля и функции (например, поле «год рождения»), то эти члены класса могут быть описаны в базовом классе. В программе создаются объекты на основе классов «аспирант», «профессор», но нет смысла создавать объект на основе класса «сотрудник вуза».

Множественное наследование

При множественном наследовании у класса может быть более одного предка. В этом случае класс наследует методы всех предков. Достоинства такого подхода в большей гибкости. Множественное наследование реализовано в C++. Из других языков, предоставляющих эту возможность, можно отметить Python и Эйфель. Множественное наследование поддерживается в языке UML.

Множественное наследование — потенциальный источник ошибок, которые могут возникнуть из-за наличия одинаковых имен методов в предках. В языках, которые позиционируются как наследники C++ (Java, C# и др.), от множественного наследования было решено отказаться в пользу интерфейсов. Практически всегда можно обойтись без использования данного механизма. Однако, если такая необходимость все-таки возникла, то, для разрешения конфликтов использования наследованных методов с одинаковыми именами, возможно, например, применить операцию расширения видимости — «::» — для вызова конкретного метода конкретного родителя.

Попытка решения проблемы наличия одинаковых имен методов в предках была предпринята в языке Эйфель, в котором при описании нового класса необходимо явно указывать импортируемые члены каждого из наследуемых классов и их именование в дочернем классе.

Большинство современных объектно-ориентированных языков программирования (C#, Java, Delphi и др.) поддерживают возможность одновременно наследоваться от класса-предка и реализовать методы нескольких интерфейсов одним и тем же классом. Этот механизм позволяет во многом заменить множественное наследование — методы интерфейсов необходимо переопределять явно, что исключает ошибки при наследовании функциональности одинаковых методов различных классов-предков.

Полиморфизм

Полиморфи́зм — возможность объектов с одинаковой спецификацией иметь различную реализацию.

Язык программирования поддерживает полиморфизм, если классы с одинаковой спецификацией могут иметь различную реализацию — например, реализация класса может быть изменена в процессе наследования[1].

Кратко смысл полиморфизма можно выразить фразой: «Один интерфейс, множество реализаций».

Полиморфизм — один из четырёх важнейших механизмов объектно-ориентированного программирования (наряду с абстракцией, инкапсуляцией и наследованием).

Полиморфизм позволяет писать более абстрактные программы и повысить коэффициент повторного использования кода. Общие свойства объектов объединяются в систему, которую могут называть по-разному — интерфейс, класс. Общность имеет внешнее и внутреннее выражение:

внешняя общность проявляется как одинаковый набор методов с одинаковыми именами и сигнатурами (именем методов и типами аргументов и их количеством);

внутренняя общность — одинаковая функциональность методов. Её можно описать интуитивно или выразить в виде строгих законов, правил, которым должны подчиняться методы. Возможность приписывать разную функциональность одному методу (функции, операции) называется перегрузкой метода (перегрузкой функций, перегрузкой операций).

Формы полиморфизма

Используя Параметрический полиморфизм можно создавать универсальные базовые типы. В случае параметрического полиморфизма, функция реализуется для всех типов одинаково и таким образом функция реализована для произвольного типа. В параметрическом полиморфизме рассматриваются параметрические методы и типы.

Параметрические метод

Если полиморфизм включения влияет на наше восприятие объекта, то параметрический полиморфизм влияет на используемые методы, так как можно создавать методы родственных классов, откладывая объявление типов до времени выполнения. Для во избежание написания отдельного метода каждого типа применяется параметрический полиморфизм, при этом тип параметров будет являться таким же параметром, как и операнды...

Параметрические типы.

Вместо того, чтобы писать класс для каждого конкретного типа следует создать типы, которые будут реализованы во время выполнения программы то есть мы создаем параметрический тип.

3. Формализованное описание разработанных классов

Самый главный класс для налоговой – это сами налоги. В классе tax хранятся данные о размере налога, вид налога.

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Text**;**

**namespace** step3

**{**

class tax

**{**

**private** string typeTax**;** //тип налога

**private** int amount**;** //сумма налога

**public** tax**(**string atypeTax**,** int aMount**)**//конструктор

**{**

**this.**typeTax **=** atypeTax**;**

**this.**amount **=** aMount**;**

**}**

**public** string getTypeTax**(){** **return** typeTax**;** **}** //получить тип налога

**public** void setTypeTax**(**string \_type**){** **this.**typeTax **=** \_type**;** **}** //поменять тип налога

**public** int getAmount**()** **{** **return** amount**;** **}** //получить сумму налога

**public** void setAmount**(**int \_amount**)** **{** **this.**amount **=** \_amount**;** **}** //поменять сумму налога

**}**

**}**

В файле taxpayer написано 2 класса. Так как за год у налогоплательщика накапливается несколько видов налога, то для хранения данных используется адресный список реализованный в классе dinList.

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Text**;**

**namespace** step3

**{**

class taxpayer

**{**

**private** string surname**;** //фамилия налогоплательщика

**private** long INN**;**

**private** int count**,** vibor**;** //ИНН налогоплательщика

**private** dinList pHead**,**Temp**,**vspom**,**Temp2**;**//ссылка на первый элемент

**public** taxpayer**(**string \_surname**,** long \_INN**)** //конструктор

**{**

**this.**surname **=** \_surname**;**

**this.**INN **=** \_INN**;**

pHead **=** **new** dinList**(null);**

pHead**.**First **=** **null;**

pHead**.**Last **=** **null;**

**}**

**public** dinList PHead **{** **get** **{** **return** pHead**.**First**;** **}** **}**

**public** string Surname

**{**

**get** **{** **return** surname**;** **}**

**set** **{** surname **=** **value;** **}**

**}**

**public** long IINN

**{**

**get** **{** **return** INN**;** **}**

**set** **{** INN **=** **value;** **}**

**}**

**public** int Count

**{**

**get** **{** **return** count**;** **}**

**}**

**public** void addForUpload**(**string addTypeTax**,** int aMount**)**

**{**

tax **add** **=** **new** tax**(**addTypeTax**,** aMount**);**

dinList dobav **=** **new** dinList**(add);**

count**++;**

**if** **(**pHead**.**Last **==** **null)**

**{**

pHead**.**First **=** dobav**;**

pHead**.**Last **=** dobav**;**

**}**

**else**

**{**

Temp **=** pHead**.**Last**;**

pHead**.**Last **=** dobav**;**

Temp**.**Next **=** pHead**.**Last**;**

**}**

**}**

**public** void addtaxVtoroe**(**string \_choose**,**string addTypeTax**,** int aMount**,**int i**)** //добавление налога

**{**

tax **add** **=** **new** tax**(**addTypeTax**,** aMount**);**

dinList dobav **=** **new** dinList**(add);**

count**++;**

**if** **(**i **==** 1**)**

**{**

vspom **=** SearchListBefore**(**\_choose**);**

dobav**.**Next **=** vspom**.**Next**;**

vspom**.**Next **=** dobav**;**

**}**

**else** **if(**i**==** 2**)**

**{**

vspom **=** SearchListAfter**(**\_choose**);**

dobav**.**Next **=** vspom**.**Next**;**

vspom**.**Next **=** dobav**;**

**}**

**else** **if(**i **==** 3**)**

**{**

Temp **=** pHead**.**First**;**

pHead**.**First **=** dobav**;**

pHead**.**First**.**Next **=** Temp**;**

**}**

**else** **if(**i **==** 4**)**

**{**

**if** **(**pHead**.**Last **==** **null)**

**{**

pHead**.**First **=** dobav**;**

pHead**.**Last **=** dobav**;**

**}**

**}**

**}**

**public** dinList SearchListBefore**(**string aTypeTax**)**

**{**

Temp2 **=** pHead**.**First**;**

Temp **=** pHead**.**First**;**

**for(**int i **=** 0**;** i**<**count**;**i**++)**

**{**

**if** **(**Temp **!=** **null)**

**{**

**if** **(**Temp**.**typeTax**.**getTypeTax**().**Equals**(**aTypeTax**))**

**{**

**return** Temp2**;**

**}**

**else**

**{**

**if** **(**i **!=** 0**)** **{** Temp2 **=** Temp2**.**Next**;** **}**

Temp **=** Temp**.**Next**;**

**}**

**}**

**}**

**return** **null;**

**}**

**public** dinList SearchListAfter**(**string aTypeTax**)**

**{**

Temp **=** pHead**.**First**;**

**while** **(**Temp **!=** **null)**

**{**

**if** **(**Temp**.**typeTax**.**getTypeTax**().**Equals**(**aTypeTax**))**

**{**

**return** Temp**;**

**}**

**else** **{** Temp **=** Temp**.**Next**;** **}**

**}**

**return** **null;**

**}**

**public** tax Search**(**string aTypeTax**)**

**{**

Temp **=** pHead**.**First**;**

**while(**Temp **!=** **null)**

**{**

**if** **(**Temp**.**typeTax**.**getTypeTax**().**Equals**(**aTypeTax**))**

**{**

**return** Temp**.**typeTax**;**

**}**

**else** **{** Temp **=** Temp**.**Next**;** **}**

**}**

**return** **null;**

**}**

**public** bool delltax**(**String \_typeTax**,** int \_amount**)** //удаление налога

**{**

**if** **(**pHead **!=** **null)**

**{**

dinList vspom **=** pHead**.**First**;**

Temp **=** pHead**.**First**;**

**while(**Temp **!=** **null)**

**{**

**if** **(**Temp **==** pHead**.**First**)**

**{**

**if** **(**Temp**.**typeTax**.**getTypeTax**().**Equals**(**\_typeTax**)** **&&** **(**Temp**.**typeTax**.**getAmount**()** **==** \_amount**))**

**{**

count**--;**

pHead**.**First **=** pHead**.**First**.**Next**;**

**return** **true;**

**}**

**else** Temp **=** Temp**.**Next**;**

**}**

**else**

**{**

**if** **(**Temp**.**typeTax**.**getTypeTax**().**Equals**(**\_typeTax**)** **&&** **(**Temp**.**typeTax**.**getAmount**()** **==** \_amount**))**

**{**

count**--;**

vspom**.**Next **=** Temp**.**Next**;**

**return** **true;**

**}**

**else** **{** Temp **=** Temp**.**Next**;** vspom **=** vspom**.**Next**;** **}**

**}**

**}**

**}**

**return** **false;**

**}**

**public** int sumTax**()** //сумма задолжности у одного плательщика

**{**

int sum **=** 0**;**

Temp **=** pHead**.**First**;**

**while** **(**Temp **!=** **null)**

**{**

sum **+=** Temp**.**typeTax**.**getAmount**();**

Temp **=** Temp**.**Next**;**

**}**

**return** sum**;**

**}**

**public** string getInfo**()** //получить информацию об одном налогоплательщике

**{**

string getinfo **=** "\nSurname taxpayer: " **+** surname**;**

Temp **=** pHead**.**First**;**

**while(**Temp **!=** **null)**

**{**

getinfo **+=** "\nType tax: " **+** Temp**.**typeTax**.**getTypeTax**();**

getinfo **+=** "\nAmount: " **+** Temp**.**typeTax**.**getAmount**();**

Temp **=** Temp**.**Next**;**

**}**

getinfo **+=** "\nsumm: " **+** sumTax**()** **+** "\n\n"**;**

**return** getinfo**;**

**}**

**public** string getInfoFor**()** //получить информацию об одном налогоплательщике

**{**

string getinfo **=** Convert**.**ToString**(**count**);**

Temp **=** pHead**.**First**;**

**while** **(**Temp **!=** **null)**

**{**

getinfo **+=** "\n" **+**Temp**.**typeTax**.**getTypeTax**();**

getinfo **+=** "\n" **+** Temp**.**typeTax**.**getAmount**();**

Temp **=** Temp**.**Next**;**

**}**

**return** getinfo**;**

**}**

**}**

class dinList //класс для записи элементов в список

**{**

**public** tax typeTax**;** //данные о налогоплательщике

**private** dinList next**,**first**,**last**;** //ссылка на следующий элемент в списке

**public** dinList**(**tax \_typeTax**)**//конструктор

**{**

typeTax **=** \_typeTax**;**

next **=** **null;**

first **=** **null;**

last **=** **null;**

**}**

**public** dinList Next **{** **get** **=>** next**;** **set** **=>** next **=** **value;** **}**

**public** dinList First **{** **get** **=>** first**;** **set** **=>** first **=** **value;** **}**

**public** dinList Last **{** **get** **=>** last**;** **set** **=>** last **=** **value;** **}**

**}**

**}**

Класс registry является контейнером для хранения всех налогоплательщиков с структуре данных очередь.

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Text**;**

**namespace** step3

**{**

class registry

**{**

**private** string nameRegistr**;** //название регистра

**private** taxpayer**[]** taxs**;** //массив объектов-налогоплательщиков

**private** int sizeArr**,**count**;** //стартовый размер массива

**private** int first**,** last**,**summa**;**

**public** registry**(**string \_nameRegistr**,**int \_sizeArr**)** //конструктор

**{**

nameRegistr **=** \_nameRegistr**;**

sizeArr **=** \_sizeArr**;**

taxs **=** **new** taxpayer**[**sizeArr**];**

first **=** 0**;** last **=** 0**;** count **=** 0**;**

**}**

**public** int Count **{** **get** **{** **return** count**;** **}** **}**

**public** void addtaxpayer**(**string \_surname**,** long \_INN**)** //добавление налогоплательщика

**{**

**if** **(**count **>** 0.75 **\*** sizeArr**)** **{** growthArr**();** **}**

taxpayer newTaxPay **=** **new** taxpayer**(**\_surname**,** \_INN**);**

taxs**[**last**]** **=** newTaxPay**;**

last**++;** count**++;**

**}**

**public** taxpayer searchtaxpayer**(**string \_surname**)** //поиск налогоплательщика

**{**

**for(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

**if(**taxs**[**i**]** **!=** **null** **&&** taxs**[**i**].**Surname**.**Equals**(**\_surname**))**

**{**

**return** taxs**[**i**];**

**}**

**}**

**return** **null;**

**}**

**public** void deltaxpayer**()** //удаление налогоплательщика

**{**

**if** **(**count **>** 0**)**

**{**

taxs**[**first**]** **=** **null;**

count**--;**

**for** **(**int i **=** **(**first **+** 1**);** i **<** last**;** i**++)**

**{**

taxs**[**i **-** 1**]** **=** taxs**[**i**];**

**}**

taxs**[**last**-**1**]** **=** **null;**

last**--;**

**if** **(**count **<** **(**0.25 **\*** sizeArr**))** **{** growthArr**();** **}**

**}**

**}**

**public** string Info**()** //информация по всему регистру

**{**

string info **=** "\nName registry - " **+** nameRegistr**;**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

info **+=**"\n" **+** taxs**[**i**].**getInfo**();**

**}**

info **+=** "\nSumm all taxpayer's: " **+** getSum**()** **+** "\n\n"**;**

**return** info**;**

**}**

**public** string InfoForSave**()**

**{**

string info **=** Convert**.**ToString**(**Count**)+** "\n"**;**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

info **+=**"\n" **+** taxs**[**i**].**Surname**;**

info **+=** "\n" **+** taxs**[**i**].**IINN**;**

**}**

**for** **(**int i **=** 0**;** i **<**count**;** i**++)**

**{**

info **+=** "\n" **+** taxs**[**i**].**getInfoFor**();**

**}**

**return** info**;**

**}**

**public** int getSum**()** //сумма всех задолжностей

**{**

summa **=** 0**;**

**for(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

summa **+=** taxs**[**i**].**sumTax**();**

**}**

**return** summa**;**

**}**

**private** void growthArr**()** //увеличить размер массива

**{**

**if** **(**count **>** 0.8 **\*** sizeArr**)**

**{**

taxpayer**[]** vspom **=** **new** taxpayer**[**sizeArr **\*** 2**];**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

vspom**[**i**]** **=** taxs**[**i**];**

**}**

sizeArr **\*=** 2**;**

taxs **=** **new** taxpayer**[**sizeArr**];**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

taxs**[**i**]** **=** vspom**[**i**];**

**}**

**}**

**else**

**{**

taxpayer**[]** vspom **=** **new** taxpayer**[**sizeArr **/** 2**];**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

vspom**[**i**]** **=** taxs**[**i**];**

**}**

sizeArr **/=** 2**;**

taxs **=** **new** taxpayer**[**sizeArr**];**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

taxs**[**i**]** **=** vspom**[**i**];**

**}**

**}**

**}**

**}**

**}**

4. Описание демонстрационного модуля с характеристикой использованных стандартных компонентов и списком реализованных обработчиков

При первом запуске программы создаётся экземпляр класса registry с параметром "Мой Налог" необходимый для файлов сохранения и добавления. После создания экземпляра создаётся форма программы.

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**IO**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Text**.**RegularExpressions**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

**namespace** step3

**{**

**public** **partial** class Form1 **:** Form

**{**

registry reg **=** **new** registry**(**"Мой Налог"**,** 5**);**

**public** Form1**()**

**{**

InitializeComponent**();**

**}**

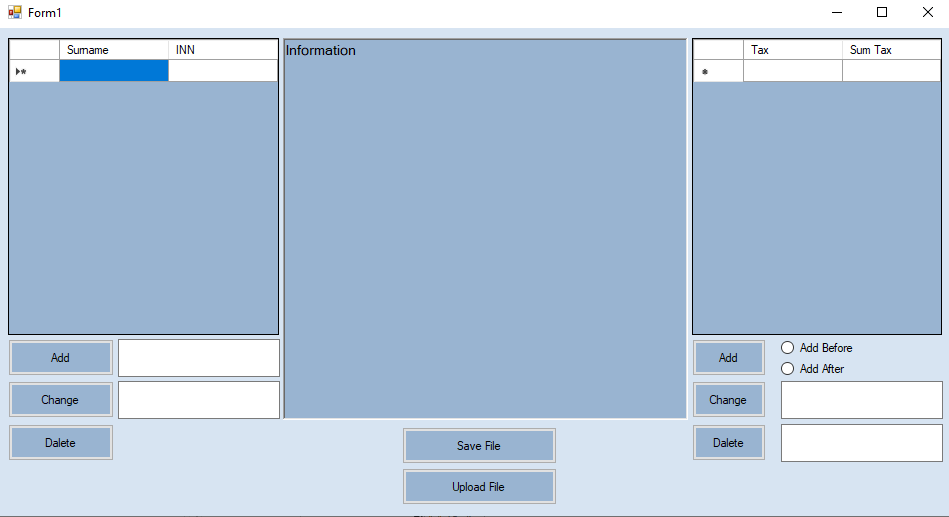


Рис.4 Работа программы

Настройка текстовых полей для правильного добавления

**private** void richTextBox1\_Click**(object** sender**,** EventArgs e**)**

**{**

richTextBox1**.**Text **=** ""**;**

string text **=** "Information:\n"**;**

richTextBox1**.**Text **=** text **+** "\n" **+** reg**.**Info**();**

**}**

**private** void textINN\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

ввод в texBox только цифр и кнопки Backspace

char ch **=** e**.**KeyChar**;**

**if** **(!**Char**.**IsDigit**(**ch**)** **&&** ch **!=** 8**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textSUM\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

ввод в texBox только цифр и кнопки Backspace

char ch **=** e**.**KeyChar**;**

**if** **(!**Char**.**IsDigit**(**ch**)** **&&** ch **!=** 8**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textSur\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

string Symbol **=** e**.**KeyChar**.**ToString**();**

**if** **(!**Regex**.**Match**(**Symbol**,** @"[а-яА-Я]|[a-zA-Z]"**).**Success **&&** e**.**KeyChar **!=** **(**char**)**Keys**.**Back**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textTAX\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

string Symbol **=** e**.**KeyChar**.**ToString**();**

**if** **(!**Regex**.**Match**(**Symbol**,** @"[а-яА-Я]|[a-zA-Z]"**).**Success **&&** e**.**KeyChar **!=** **(**char**)**Keys**.**Back**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textSUR\_TextChanged**(object** sender**,** EventArgs e**)**

**{**

**}**

**private** void textINN\_TextChanged**(object** sender**,** EventArgs e**)**

**{**

textINN**.**MaxLength **=** 16**;**

**}**

**private** void textSUM\_TextChanged**(object** sender**,** EventArgs e**)**

**{**

textSUM**.**MaxLength **=** 7**;**

**}**

настройка кнопок для налогоплательщика, оновляется таблица и заполняются данные в экземпляре класса registry

**private** void butAddSur\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**textSUR**.**Text **!=** "" **&&** textINN**.**Text **!=** ""**)**

**{**

reg**.**addtaxpayer**(**textSUR**.**Text**,** Convert**.**ToInt64**(**textINN**.**Text**));**

dataGridView1**.**Rows**.**Add**(**textSUR**.**Text**,** textINN**.**Text**);**

textSUR**.**Clear**();**textINN**.**Clear**();**

**}**

**else** **{** MessageBox**.**Show**(**"Добавление не выполнено!\nЗаполните все нужные данные."**);** **}**

**}**

При нажатии на кнопку изменения данных меняются данные как в таблице так и в экземпляре класса registry.

**private** void butChSur\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**textSUR**.**Text **!=** "" **&&** textINN**.**Text **!=** "" **&&** dataGridView1**.**CurrentCell**.**Value **!=** **null)**

**{**

reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Surname **=** textSUR**.**Text**;**

reg**.**searchtaxpayer**(**textSUR**.**Text**).**IINN **=** Convert**.**ToInt64**(**textINN**.**Text**);**

dataGridView1**.**Rows**[**dataGridView1**.**CurrentRow**.**Index**].**Cells**[**0**].**Value **=** textSUR**.**Text**;**

dataGridView1**.**Rows**[**dataGridView1**.**CurrentRow**.**Index**].**Cells**[**1**].**Value **=** textINN**.**Text**;**

textINN**.**Clear**();**textSUR**.**Clear**();**

**}**

**else** **{** MessageBox**.**Show**(**"Изменение не выполнено!\nЗаполните все нужные данные,\nили выберите изменяемый элемент."**);** **}**

**}**

При удаление очищается таблица и данные в динамической очереди.

**private** void butDelSur\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**reg**.**Count **!=** 0**)**

**{**

reg**.**deltaxpayer**();**

dataGridView1**.**Rows**.**RemoveAt**(**0**);**

dataGridView2**.**Rows**.**Clear**();**

**}**

**else** **{** MessageBox**.**Show**(**"Удаление не выполнено!\nНечего удалять."**);** **}**

**}**

//настройка кнопок для налогов

**private** void butAddTax\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**textSUM**.**Text **!=** "" **&&** textTAX**.**Text **!=** "" **&&** dataGridView1**.**CurrentCell**.**Value **!=** **null)**

**{**

dataGridView2**.**Rows**.**Add**();**

int index **=** dataGridView2**.**CurrentRow**.**Index**;**

**if(**reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count **==** 0 **&&** radioButAddAfter**.**Checked **==** **false** **&&** radioButAddBefore**.**Checked **==** **false)**

**{**

reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**addtaxVtoroe**(**""**,** textTAX**.**Text**,** Convert**.**ToInt32**(**textSUM**.**Text**),**4**);**

dataGridView2**.**Rows**[**0**].**Cells**[**0**].**Value **=** textTAX**.**Text**;**

dataGridView2**.**Rows**[**0**].**Cells**[**1**].**Value **=** textSUM**.**Text**;**

**}**

**else** **if** **(**radioButAddBefore**.**Checked **==** **true)**

**{**

**if(**index **==** 0**)**

**{**

reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**addtaxVtoroe**(**dataGridView2**.**CurrentCell**.**Value**.**ToString**(),** textTAX**.**Text**,** Convert**.**ToInt32**(**textSUM**.**Text**),** 3**);**

**for** **(**int i **=** **(**reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count **-** 1**);** i **>=** index**;** i**--)**

**{**

dataGridView2**.**Rows**[**i **+** 1**].**Cells**[**0**].**Value **=** dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value**;**

dataGridView2**.**Rows**[**i **+** 1**].**Cells**[**1**].**Value **=** dataGridView2**.**Rows**[**i**].**Cells**[**1**].**Value**;**

**}**

dataGridView2**.**Rows**[**index**].**Cells**[**0**].**Value **=** textTAX**.**Text**;**

dataGridView2**.**Rows**[**index**].**Cells**[**1**].**Value **=** textSUM**.**Text**;**

**}**

**else** **{**

reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**addtaxVtoroe**(**dataGridView2**.**CurrentCell**.**Value**.**ToString**(),** textTAX**.**Text**,** Convert**.**ToInt32**(**textSUM**.**Text**),** 1**);**

**for** **(**int i **=** **(**reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count **-** 1**);** i **>=** index**;** i**--)**

**{**

dataGridView2**.**Rows**[**i **+** 1**].**Cells**[**0**].**Value **=** dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value**;**

dataGridView2**.**Rows**[**i **+** 1**].**Cells**[**1**].**Value **=** dataGridView2**.**Rows**[**i**].**Cells**[**1**].**Value**;**

**}**

dataGridView2**.**Rows**[**index**].**Cells**[**0**].**Value **=** textTAX**.**Text**;**

dataGridView2**.**Rows**[**index**].**Cells**[**1**].**Value **=** textSUM**.**Text**;**

**}**

**}**

**else** **if(**radioButAddAfter**.**Checked **==** **true)**

**{**

reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**addtaxVtoroe**(**dataGridView2**.**CurrentCell**.**Value**.**ToString**(),** textTAX**.**Text**,** Convert**.**ToInt32**(**textSUM**.**Text**),** 2**);**

**for** **(**int i **=** **(**reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count**-**1**);** i **>** index**;** i**--)**

**{**

dataGridView2**.**Rows**[**i **+** 1**].**Cells**[**0**].**Value **=** dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value**;**

dataGridView2**.**Rows**[**i **+** 1**].**Cells**[**1**].**Value **=** dataGridView2**.**Rows**[**i**].**Cells**[**1**].**Value**;**

**}**

dataGridView2**.**Rows**[**index**+**1**].**Cells**[**0**].**Value **=** textTAX**.**Text**;**

dataGridView2**.**Rows**[**index**+**1**].**Cells**[**1**].**Value **=** textSUM**.**Text**;**

**}**

**else** **{** MessageBox**.**Show**(**"Выберите куда добавить налог."**);** **}**

**}**

**}**

**private** void butChTax\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null** **&&** dataGridView2**.**CurrentCell**.**Value **!=** **null** **&&** textTAX**.**Text **!=** "" **&&** textSUM**.**Text **!=** ""**)**

**{**

reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Search**(**dataGridView2**.**CurrentCell**.**Value**.**ToString**()).**setTypeTax**(**textTAX**.**Text**);**

reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Search**(**textTAX**.**Text**).**setAmount**(**Convert**.**ToInt32**(**textSUM**.**Text**));**

dataGridView2**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**0**].**Value **=** textTAX**.**Text**;**

dataGridView2**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**1**].**Value **=** textSUM**.**Text**;**

**}**

**else** MessageBox**.**Show**(**"Изменить не получилось!\nВведите нужные данные.\nИли выберите изменяемый объект."**);**

**}**

При удалении налога удаляются все данные как из таблице с налогами, так и из адресного списка.

**private** void butDelTax\_Click**(object** sender**,** EventArgs e**)**

**{**

**if(**dataGridView1**.**CurrentCell**.**Value **!=** **null** **&&** dataGridView2**.**CurrentCell**.**Value **!=** **null)**

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**

int index **=** dataGridView2**.**CurrentRow**.**Index**;**

reg**.**searchtaxpayer**(**text**).**delltax**(**dataGridView2**.**Rows**[**index**].**Cells**[**0**].**Value**.**ToString**(),** Convert**.**ToInt32**(**dataGridView2**.**Rows**[**index**].**Cells**[**1**].**Value**));**

dataGridView2**.**Rows**.**RemoveAt**(**index**);**

dataGridView2**.**AllowUserToAddRows **=** **false;**

**}**

**else** **{** MessageBox**.**Show**(**"Удалить не получилось.\nВыберите удаляемый объект."**);** **}**

**}**

**private** void dataGridView1\_CellContentClick**(object** sender**,** DataGridViewCellEventArgs e**)**

**{**

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null)**

**{**

dataGridView2**.**Rows**.**Clear**();**

radioButAddAfter**.**Checked **=** **false;**

radioButAddBefore**.**Checked **=** **false;**

taxpayer vspom **=** reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**());**

**if** **(**vspom **!=** **null)**

**{**

dinList pTemp **=** vspom**.**PHead**;**

int chet **=** reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count**;**

**for** **(**int i **=** 0**;** i **<** chet**;** i**++)**

**{**

dataGridView2**.**Rows**.**Add**(**pTemp**.**typeTax**.**getTypeTax**(),** pTemp**.**typeTax**.**getAmount**());**

pTemp **=** pTemp**.**Next**;**

**}**

**}**

**}**

**else** MessageBox**.**Show**(**"Выбрана пустая строчка"**);**

**}**

Метод выгрузки данных в файл формата txt. Для дальнейшего чтения в начале файла пишем "Файл", чтобы можно было идентифицировать файл программы от других txt.

**private** void butSave\_Click**(object** sender**,** EventArgs e**)**

**{**

string path**;**

string fileText **=** "Tax\n"**;**

**using** **(**SaveFileDialog saveFile **=** **new** SaveFileDialog**()** **{** Filter **=** "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*" **})**

**{**

**if** **(**saveFile**.**ShowDialog**()** **==** DialogResult**.**OK**)**

**{**

path **=** saveFile**.**FileName**;**

fileText **+=** reg**.**InfoForSave**();**

**using** **(**FileStream file **=** **new** FileStream**(**path**,** FileMode**.**OpenOrCreate**))**

**using** **(**StreamWriter stream **=** **new** StreamWriter**(**file**))**

**{**

stream**.**Write**(**fileText**);**

stream**.**Close**();**

file**.**Close**();**

**}**

**}**

**}**

**}**

Метод загрузки из файла. Открывает диалоговое окно для отображения txt файлов. Если файл удалось открыть происходит проверка на корректные данные, т.е. первая сточка в файле должна быть "Файл"после этого происходит дальнейшее чтение файла, заполнение структуры и таблицы данными из файла.

**private** void butUpload\_Click**(object** sender**,** EventArgs e**)**

**{**

string path **=** ""**,** doIt **=** ""**;**

int Count**,** CusCount**;**

**using** **(**OpenFileDialog openFile **=** **new** OpenFileDialog**()** **{** Filter **=** "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*" **})**

**{**

**if** **(**openFile**.**ShowDialog**()** **==** DialogResult**.**OK**)**

**{**

path **=** openFile**.**FileName**;**

**}**

**}**

**using** **(**StreamReader stream3 **=** **new** StreamReader**(**path**))**

**{**

doIt **=** stream3**.**ReadLine**();**

//Проверка на пригодность файла для работы с программой

**if** **(**doIt **==** "Tax"**)**

**{**

**for** **(**int i **=** 0**;** i **<=** reg**.**Count**;** i**++)**

**{**

reg**.**deltaxpayer**();**

dataGridView1**.**Rows**.**Clear**();**

dataGridView2**.**Rows**.**Clear**();**

**}**

Count **=** int**.**Parse**(**stream3**.**ReadLine**());**

**for** **(**int i **=** 1**;** i **<=** Count**;)**

**{**

string Surname **=** stream3**.**ReadLine**();**

long INN **=** long**.**Parse**(**stream3**.**ReadLine**());**

reg**.**addtaxpayer**(**Surname**,** INN**);**

dataGridView1**.**Rows**.**Add**();**

dataGridView1**.**Rows**[**i **-** 1**].**Cells**[**0**].**Value **=** Surname**;**

dataGridView1**.**Rows**[**i **-** 1**].**Cells**[**1**].**Value **=** INN**;**

**if** **(**i **!=** Count**)**

i**++;**

**else**

**break;**

**}**

stream3**.**ReadLine**();** int hall **=** 0**;**

**while** **(**stream3**.**Peek**()** **!=** **-**1**)**

**{**

CusCount **=** int**.**Parse**(**stream3**.**ReadLine**());**

**if** **(**CusCount **!=** 0**)**

**for** **(**int i **=** 1**;** i **<=** CusCount**;)**

**{**

string atypeTax **=** stream3**.**ReadLine**();**

int aMount **=** int**.**Parse**(**stream3**.**ReadLine**());**

string name **=** Convert**.**ToString**(**dataGridView1**.**Rows**[**hall**].**Cells**[**0**].**Value**);**

reg**.**searchtaxpayer**(**name**).**addForUpload**(**atypeTax**,** aMount**);**

**if** **(**i **!=** CusCount**)**

i**++;**

**else**

**break;**

**}**

hall**++;**

**}**

**}**

**else** **{** MessageBox**.**Show**(**"Данный файл не подходит для работы с программой"**);** **}**

stream3**.**Close**();**

**}**

**}**

**}**

**}**

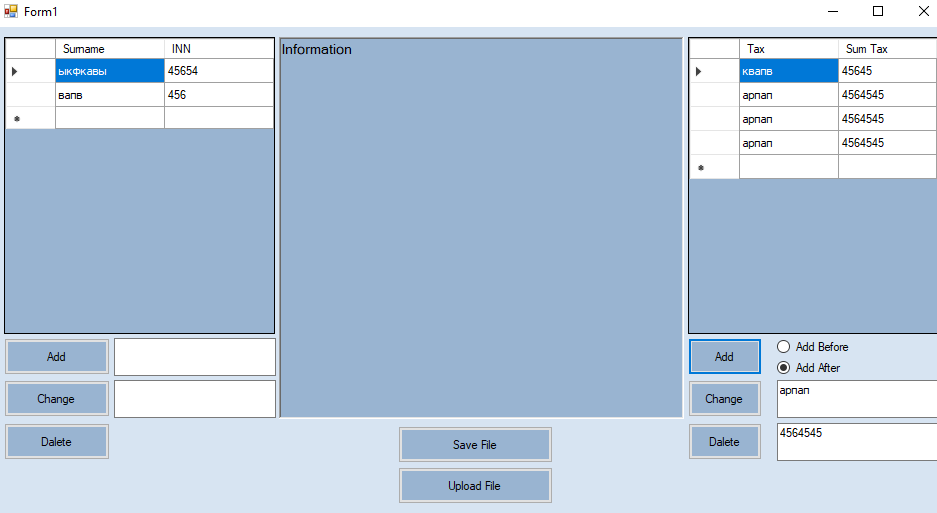


Рис 5 Работа программы

1. Описание структуры проекта в соответствии с использованным инструментом разработки

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, UWP а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, .NET Core, .NET, MAUI, Xbox, Windows Phone .NET Compact Framework и Silverlight. После покупки компании Xamarin корпорацией Microsoft появилась возможность разработки IOS и Android программ.

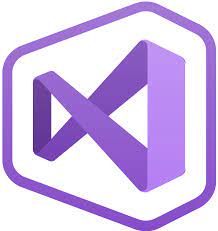


Рис. 6 Visual studio

В главной папке проекта находятся разработанные классы \*.cs , так же в App.config находятся настройки для среды разработки Visual Studio находятся настройки конфигурации для запуска приложения. Файл \*.resx содержит в себе дизайн оконного приложения, составленный на языке разметки XML.

В папке Propertis находятся все подключенные в разработанных классах стандартные библиотеки для установления зависимостей в проекте.

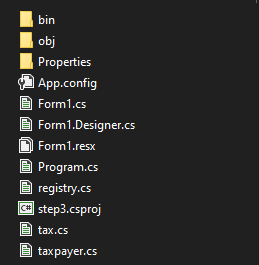


Рис. 7 Содержимое папки проекта.

Папка obj содержит в себе данные последнего Debug запуска проекта с соответствующими файлами конфигурации и логами.

В парке bin находиться главный исполняемый файл программы с последней сборки проекта, а также папка Relies с исполняемым файлом готовым к установке программы со всеми зависимостями на любой Windows операционной системе.

# **Список литературы**

1. Алгоритмы и структуры данных: учебное пособие / О.Б. Фофанов; Томский политехнический университет. – Томск: Изд-во Томского политех нического университета, 2019.
2. Васильев А. Н. Java. Объектно-ориентированное программирование. Учебное пособие. Стандарт третьего поколения. – "Издательский дом Питер 2021.
3. Кувшинов Д. Р., Осипов С. И. Основы программирования: язык C++: учебное пособие. – 2021.
4. Ляпин А. А., Быкова Ю. А. РАЗРАБОТКА БАЛЛИСТИЧЕСКОЙ ФУНКЦИИ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C/C++ //ПОЛИ- ХОТОМИЧЕСКИЕ КЛАССИФИКАЦИИ СПЕЦИАЛЬНЫХ ТЕРМИ- НОВ В НАУЧНОЙ И УЧЕБНОЙ ЛИТЕРАТУРЕ. – 2021. – С. 308-311.
5. Объектно-ориентированное программирование. Учебно-методическое пособие/ Козин.А.Н., Якунина Е.А. – Казань: УВО «Университет управления «ТИСБИ», 2020.
6. Структуры и алгоритмы обработки данных. Часть 2. Усложнённые структуры данных. Теория и методика обучения. Учебно-методическое пособие / А.Н. Козин, Л.Б. Таренко. - Казань: УВО "Университет управления "ТИСБИ 2018
7. Павловская Т. А., Щупак Ю. А. C/C++. Структурное и объектно- ориентированное программирование. Практикум. – "Издательский дом Питер 2021.
8. Структуры и алгоритмы обработки данных. Часть 3. Поиск и сортировка данных. Теория и методика обучения. Учебно-методическое пособие/ А.Н. Козин, Л.Б. Таренко. - Казань: УВО "Университет управления "ТИСБИ 2020
9. Puuronen M. Implementing Horizontal Layout for the Qt Design Studio’s Component Library. – 2021.
10. Sherriff N. Learn Qt 5: Build modern, responsive cross-platform desktop applications with Qt, C++, and QML. – Packt Publishing Ltd, 2018.

Листинг программы

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**IO**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Text**.**RegularExpressions**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

**namespace** step3

**{**

**public** **partial** class Form1 **:** Form

**{**

registry reg **=** **new** registry**(**"Мой Налог"**,** 5**);**

**public** Form1**()**

**{**

InitializeComponent**();**

**}**

//Настройка текстовых полей для правильного добавления

**private** void richTextBox1\_Click**(object** sender**,** EventArgs e**)**

**{**

richTextBox1**.**Text **=** ""**;**

string text **=** "Information:\n"**;**

richTextBox1**.**Text **=** text **+** "\n" **+** reg**.**Info**();**

**}**

**private** void textINN\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

// ввод в texBox только цифр и кнопки Backspace

char ch **=** e**.**KeyChar**;**

**if** **(!**Char**.**IsDigit**(**ch**)** **&&** ch **!=** 8**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textSUM\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

// ввод в texBox только цифр и кнопки Backspace

char ch **=** e**.**KeyChar**;**

**if** **(!**Char**.**IsDigit**(**ch**)** **&&** ch **!=** 8**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textSur\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

string Symbol **=** e**.**KeyChar**.**ToString**();**

**if** **(!**Regex**.**Match**(**Symbol**,** @"[а-яА-Я]|[a-zA-Z]"**).**Success **&&** e**.**KeyChar **!=** **(**char**)**Keys**.**Back**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textTAX\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

string Symbol **=** e**.**KeyChar**.**ToString**();**

**if** **(!**Regex**.**Match**(**Symbol**,** @"[а-яА-Я]|[a-zA-Z]"**).**Success **&&** e**.**KeyChar **!=** **(**char**)**Keys**.**Back**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textSUR\_TextChanged**(object** sender**,** EventArgs e**)**

**{**

**}**

**private** void textINN\_TextChanged**(object** sender**,** EventArgs e**)**

**{**

textINN**.**MaxLength **=** 16**;**

**}**

**private** void textSUM\_TextChanged**(object** sender**,** EventArgs e**)**

**{**

textSUM**.**MaxLength **=** 7**;**

**}**

//настройка кнопок для налогоплательщика

**private** void butAddSur\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**textSUR**.**Text **!=** "" **&&** textINN**.**Text **!=** ""**)**

**{**

reg**.**addtaxpayer**(**textSUR**.**Text**,** Convert**.**ToInt64**(**textINN**.**Text**));**

dataGridView1**.**Rows**.**Add**(**textSUR**.**Text**,** textINN**.**Text**);**

textSUR**.**Clear**();**textINN**.**Clear**();**

**}**

**else** **{** MessageBox**.**Show**(**"Добавление не выполнено!\nЗаполните все нужные данные."**);** **}**

**}**

**private** void butChSur\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**textSUR**.**Text **!=** "" **&&** textINN**.**Text **!=** "" **&&** dataGridView1**.**CurrentCell**.**Value **!=** **null)**

**{**

reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Surname **=** textSUR**.**Text**;**

reg**.**searchtaxpayer**(**textSUR**.**Text**).**IINN **=** Convert**.**ToInt64**(**textINN**.**Text**);**

dataGridView1**.**Rows**[**dataGridView1**.**CurrentRow**.**Index**].**Cells**[**0**].**Value **=** textSUR**.**Text**;**

dataGridView1**.**Rows**[**dataGridView1**.**CurrentRow**.**Index**].**Cells**[**1**].**Value **=** textINN**.**Text**;**

textINN**.**Clear**();**textSUR**.**Clear**();**

**}**

**else** **{** MessageBox**.**Show**(**"Изменение не выполнено!\nЗаполните все нужные данные,\nили выберите изменяемый элемент."**);** **}**

**}**

**private** void butDelSur\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**reg**.**Count **!=** 0**)**

**{**

reg**.**deltaxpayer**();**

dataGridView1**.**Rows**.**RemoveAt**(**0**);**

dataGridView2**.**Rows**.**Clear**();**

**}**

**else** **{** MessageBox**.**Show**(**"Удаление не выполнено!\nНечего удалять."**);** **}**

**}**

//настройка кнопок для налогов

**private** void butAddTax\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**textSUM**.**Text **!=** "" **&&** textTAX**.**Text **!=** "" **&&** dataGridView1**.**CurrentCell**.**Value **!=** **null)**

**{**

dataGridView2**.**Rows**.**Add**();**

int index **=** dataGridView2**.**CurrentRow**.**Index**;**

**if(**reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count **==** 0 **&&** radioButAddAfter**.**Checked **==** **false** **&&** radioButAddBefore**.**Checked **==** **false)**

**{**

reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**addtaxVtoroe**(**""**,** textTAX**.**Text**,** Convert**.**ToInt32**(**textSUM**.**Text**),**4**);**

dataGridView2**.**Rows**[**0**].**Cells**[**0**].**Value **=** textTAX**.**Text**;**

dataGridView2**.**Rows**[**0**].**Cells**[**1**].**Value **=** textSUM**.**Text**;**

**}**

**else** **if** **(**radioButAddBefore**.**Checked **==** **true)**

**{**

**if(**index **==** 0**)**

**{**

reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**addtaxVtoroe**(**dataGridView2**.**CurrentCell**.**Value**.**ToString**(),** textTAX**.**Text**,** Convert**.**ToInt32**(**textSUM**.**Text**),** 3**);**

**for** **(**int i **=** **(**reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count **-** 1**);** i **>=** index**;** i**--)**

**{**

dataGridView2**.**Rows**[**i **+** 1**].**Cells**[**0**].**Value **=** dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value**;**

dataGridView2**.**Rows**[**i **+** 1**].**Cells**[**1**].**Value **=** dataGridView2**.**Rows**[**i**].**Cells**[**1**].**Value**;**

**}**

dataGridView2**.**Rows**[**index**].**Cells**[**0**].**Value **=** textTAX**.**Text**;**

dataGridView2**.**Rows**[**index**].**Cells**[**1**].**Value **=** textSUM**.**Text**;**

**}**

**else** **{**

reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**addtaxVtoroe**(**dataGridView2**.**CurrentCell**.**Value**.**ToString**(),** textTAX**.**Text**,** Convert**.**ToInt32**(**textSUM**.**Text**),** 1**);**

**for** **(**int i **=** **(**reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count **-** 1**);** i **>=** index**;** i**--)**

**{**

dataGridView2**.**Rows**[**i **+** 1**].**Cells**[**0**].**Value **=** dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value**;**

dataGridView2**.**Rows**[**i **+** 1**].**Cells**[**1**].**Value **=** dataGridView2**.**Rows**[**i**].**Cells**[**1**].**Value**;**

**}**

dataGridView2**.**Rows**[**index**].**Cells**[**0**].**Value **=** textTAX**.**Text**;**

dataGridView2**.**Rows**[**index**].**Cells**[**1**].**Value **=** textSUM**.**Text**;**

**}**

**}**

**else** **if(**radioButAddAfter**.**Checked **==** **true)**

**{**

reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**addtaxVtoroe**(**dataGridView2**.**CurrentCell**.**Value**.**ToString**(),** textTAX**.**Text**,** Convert**.**ToInt32**(**textSUM**.**Text**),** 2**);**

**for** **(**int i **=** **(**reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count**-**1**);** i **>** index**;** i**--)**

**{**

dataGridView2**.**Rows**[**i **+** 1**].**Cells**[**0**].**Value **=** dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value**;**

dataGridView2**.**Rows**[**i **+** 1**].**Cells**[**1**].**Value **=** dataGridView2**.**Rows**[**i**].**Cells**[**1**].**Value**;**

**}**

dataGridView2**.**Rows**[**index**+**1**].**Cells**[**0**].**Value **=** textTAX**.**Text**;**

dataGridView2**.**Rows**[**index**+**1**].**Cells**[**1**].**Value **=** textSUM**.**Text**;**

**}**

**else** **{** MessageBox**.**Show**(**"Выберите куда добавить налог."**);** **}**

**}**

**}**

**private** void butChTax\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null** **&&** dataGridView2**.**CurrentCell**.**Value **!=** **null** **&&** textTAX**.**Text **!=** "" **&&** textSUM**.**Text **!=** ""**)**

**{**

reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Search**(**dataGridView2**.**CurrentCell**.**Value**.**ToString**()).**setTypeTax**(**textTAX**.**Text**);**

reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Search**(**textTAX**.**Text**).**setAmount**(**Convert**.**ToInt32**(**textSUM**.**Text**));**

dataGridView2**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**0**].**Value **=** textTAX**.**Text**;**

dataGridView2**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**1**].**Value **=** textSUM**.**Text**;**

**}**

**else** MessageBox**.**Show**(**"Изменить не получилось!\nВведите нужные данные.\nИли выберите изменяемый объект."**);**

**}**

**private** void butDelTax\_Click**(object** sender**,** EventArgs e**)**

**{**

**if(**dataGridView1**.**CurrentCell**.**Value **!=** **null** **&&** dataGridView2**.**CurrentCell**.**Value **!=** **null)**

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**

int index **=** dataGridView2**.**CurrentRow**.**Index**;**

reg**.**searchtaxpayer**(**text**).**delltax**(**dataGridView2**.**Rows**[**index**].**Cells**[**0**].**Value**.**ToString**(),** Convert**.**ToInt32**(**dataGridView2**.**Rows**[**index**].**Cells**[**1**].**Value**));**

dataGridView2**.**Rows**.**RemoveAt**(**index**);**

dataGridView2**.**AllowUserToAddRows **=** **false;**

**}**

**else** **{** MessageBox**.**Show**(**"Удалить не получилось.\nВыберите удаляемый объект."**);** **}**

**}**

**private** void dataGridView1\_CellContentClick**(object** sender**,** DataGridViewCellEventArgs e**)**

**{**

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null)**

**{**

dataGridView2**.**Rows**.**Clear**();**

radioButAddAfter**.**Checked **=** **false;**

radioButAddBefore**.**Checked **=** **false;**

taxpayer vspom **=** reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**());**

**if** **(**vspom **!=** **null)**

**{**

dinList pTemp **=** vspom**.**PHead**;**

int chet **=** reg**.**searchtaxpayer**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count**;**

**for** **(**int i **=** 0**;** i **<** chet**;** i**++)**

**{**

dataGridView2**.**Rows**.**Add**(**pTemp**.**typeTax**.**getTypeTax**(),** pTemp**.**typeTax**.**getAmount**());**

pTemp **=** pTemp**.**Next**;**

**}**

**}**

**}**

**else** MessageBox**.**Show**(**"Выбрана пустая строчка"**);**

**}**

**private** void butSave\_Click**(object** sender**,** EventArgs e**)**

**{**

string path**;**

string fileText **=** "Tax\n"**;**

**using** **(**SaveFileDialog saveFile **=** **new** SaveFileDialog**()** **{** Filter **=** "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*" **})**

**{**

**if** **(**saveFile**.**ShowDialog**()** **==** DialogResult**.**OK**)**

**{**

path **=** saveFile**.**FileName**;**

fileText **+=** reg**.**InfoForSave**();**

**using** **(**FileStream file **=** **new** FileStream**(**path**,** FileMode**.**OpenOrCreate**))**

**using** **(**StreamWriter stream **=** **new** StreamWriter**(**file**))**

**{**

stream**.**Write**(**fileText**);**

stream**.**Close**();**

file**.**Close**();**

**}**

**}**

**}**

**}**

**private** void butUpload\_Click**(object** sender**,** EventArgs e**)**

**{**

string path **=** ""**,** doIt **=** ""**;**

int Count**,** CusCount**;**

**using** **(**OpenFileDialog openFile **=** **new** OpenFileDialog**()** **{** Filter **=** "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*" **})**

**{**

**if** **(**openFile**.**ShowDialog**()** **==** DialogResult**.**OK**)**

**{**

path **=** openFile**.**FileName**;**

**}**

**}**

**using** **(**StreamReader stream3 **=** **new** StreamReader**(**path**))**

**{**

doIt **=** stream3**.**ReadLine**();**

//Проверка на пригодность файла для работы с программой

**if** **(**doIt **==** "Tax"**)**

**{**

**for** **(**int i **=** 0**;** i **<=** reg**.**Count**;** i**++)**

**{**

reg**.**deltaxpayer**();**

dataGridView1**.**Rows**.**Clear**();**

dataGridView2**.**Rows**.**Clear**();**

**}**

Count **=** int**.**Parse**(**stream3**.**ReadLine**());**

**for** **(**int i **=** 1**;** i **<=** Count**;)**

**{**

string Surname **=** stream3**.**ReadLine**();**

long INN **=** long**.**Parse**(**stream3**.**ReadLine**());**

reg**.**addtaxpayer**(**Surname**,** INN**);**

dataGridView1**.**Rows**.**Add**();**

dataGridView1**.**Rows**[**i **-** 1**].**Cells**[**0**].**Value **=** Surname**;**

dataGridView1**.**Rows**[**i **-** 1**].**Cells**[**1**].**Value **=** INN**;**

**if** **(**i **!=** Count**)**

i**++;**

**else**

**break;**

**}**

stream3**.**ReadLine**();** int hall **=** 0**;**

**while** **(**stream3**.**Peek**()** **!=** **-**1**)**

**{**

CusCount **=** int**.**Parse**(**stream3**.**ReadLine**());**

**if** **(**CusCount **!=** 0**)**

**for** **(**int i **=** 1**;** i **<=** CusCount**;)**

**{**

string atypeTax **=** stream3**.**ReadLine**();**

int aMount **=** int**.**Parse**(**stream3**.**ReadLine**());**

string name **=** Convert**.**ToString**(**dataGridView1**.**Rows**[**hall**].**Cells**[**0**].**Value**);**

reg**.**searchtaxpayer**(**name**).**addForUpload**(**atypeTax**,** aMount**);**

**if** **(**i **!=** CusCount**)**

i**++;**

**else**

**break;**

**}**

hall**++;**

**}**

**}**

**else** **{** MessageBox**.**Show**(**"Данный файл не подходит для работы с программой"**);** **}**

stream3**.**Close**();**

**}**

**}**

**}**

**}**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Text**;**

**namespace** step3

**{**

class taxpayer

**{**

**private** string surname**;** //фамилия налогоплательщика

**private** long INN**;**

**private** int count**,** vibor**;** //ИНН налогоплательщика

**private** dinList pHead**,**Temp**,**vspom**,**Temp2**;**//ссылка на первый элемент

**public** taxpayer**(**string \_surname**,** long \_INN**)** //конструктор

**{**

**this.**surname **=** \_surname**;**

**this.**INN **=** \_INN**;**

pHead **=** **new** dinList**(null);**

pHead**.**First **=** **null;**

pHead**.**Last **=** **null;**

**}**

**public** dinList PHead **{** **get** **{** **return** pHead**.**First**;** **}** **}**

**public** string Surname

**{**

**get** **{** **return** surname**;** **}**

**set** **{** surname **=** **value;** **}**

**}**

**public** long IINN

**{**

**get** **{** **return** INN**;** **}**

**set** **{** INN **=** **value;** **}**

**}**

**public** int Count

**{**

**get** **{** **return** count**;** **}**

**}**

**public** void addForUpload**(**string addTypeTax**,** int aMount**)**

**{**

tax **add** **=** **new** tax**(**addTypeTax**,** aMount**);**

dinList dobav **=** **new** dinList**(add);**

count**++;**

**if** **(**pHead**.**Last **==** **null)**

**{**

pHead**.**First **=** dobav**;**

pHead**.**Last **=** dobav**;**

**}**

**else**

**{**

Temp **=** pHead**.**Last**;**

pHead**.**Last **=** dobav**;**

Temp**.**Next **=** pHead**.**Last**;**

**}**

**}**

**public** void addtaxVtoroe**(**string \_choose**,**string addTypeTax**,** int aMount**,**int i**)** //добавление налога

**{**

tax **add** **=** **new** tax**(**addTypeTax**,** aMount**);**

dinList dobav **=** **new** dinList**(add);**

count**++;**

**if** **(**i **==** 1**)**

**{**

vspom **=** SearchListBefore**(**\_choose**);**

dobav**.**Next **=** vspom**.**Next**;**

vspom**.**Next **=** dobav**;**

**}**

**else** **if(**i**==** 2**)**

**{**

vspom **=** SearchListAfter**(**\_choose**);**

dobav**.**Next **=** vspom**.**Next**;**

vspom**.**Next **=** dobav**;**

**}**

**else** **if(**i **==** 3**)**

**{**

Temp **=** pHead**.**First**;**

pHead**.**First **=** dobav**;**

pHead**.**First**.**Next **=** Temp**;**

**}**

**else** **if(**i **==** 4**)**

**{**

**if** **(**pHead**.**Last **==** **null)**

**{**

pHead**.**First **=** dobav**;**

pHead**.**Last **=** dobav**;**

**}**

**}**

**}**

**public** dinList SearchListBefore**(**string aTypeTax**)**

**{**

Temp2 **=** pHead**.**First**;**

Temp **=** pHead**.**First**;**

**for(**int i **=** 0**;** i**<**count**;**i**++)**

**{**

**if** **(**Temp **!=** **null)**

**{**

**if** **(**Temp**.**typeTax**.**getTypeTax**().**Equals**(**aTypeTax**))**

**{**

**return** Temp2**;**

**}**

**else**

**{**

**if** **(**i **!=** 0**)** **{** Temp2 **=** Temp2**.**Next**;** **}**

Temp **=** Temp**.**Next**;**

**}**

**}**

**}**

**return** **null;**

**}**

**public** dinList SearchListAfter**(**string aTypeTax**)**

**{**

Temp **=** pHead**.**First**;**

**while** **(**Temp **!=** **null)**

**{**

**if** **(**Temp**.**typeTax**.**getTypeTax**().**Equals**(**aTypeTax**))**

**{**

**return** Temp**;**

**}**

**else** **{** Temp **=** Temp**.**Next**;** **}**

**}**

**return** **null;**

**}**

**public** tax Search**(**string aTypeTax**)**

**{**

Temp **=** pHead**.**First**;**

**while(**Temp **!=** **null)**

**{**

**if** **(**Temp**.**typeTax**.**getTypeTax**().**Equals**(**aTypeTax**))**

**{**

**return** Temp**.**typeTax**;**

**}**

**else** **{** Temp **=** Temp**.**Next**;** **}**

**}**

**return** **null;**

**}**

**public** bool delltax**(**String \_typeTax**,** int \_amount**)** //удаление налога

**{**

**if** **(**pHead **!=** **null)**

**{**

dinList vspom **=** pHead**.**First**;**

Temp **=** pHead**.**First**;**

**while(**Temp **!=** **null)**

**{**

**if** **(**Temp **==** pHead**.**First**)**

**{**

**if** **(**Temp**.**typeTax**.**getTypeTax**().**Equals**(**\_typeTax**)** **&&** **(**Temp**.**typeTax**.**getAmount**()** **==** \_amount**))**

**{**

count**--;**

pHead**.**First **=** pHead**.**First**.**Next**;**

**return** **true;**

**}**

**else** Temp **=** Temp**.**Next**;**

**}**

**else**

**{**

**if** **(**Temp**.**typeTax**.**getTypeTax**().**Equals**(**\_typeTax**)** **&&** **(**Temp**.**typeTax**.**getAmount**()** **==** \_amount**))**

**{**

count**--;**

vspom**.**Next **=** Temp**.**Next**;**

**return** **true;**

**}**

**else** **{** Temp **=** Temp**.**Next**;** vspom **=** vspom**.**Next**;** **}**

**}**

**}**

**}**

**return** **false;**

**}**

**public** int sumTax**()** //сумма задолжности у одного плательщика

**{**

int sum **=** 0**;**

Temp **=** pHead**.**First**;**

**while** **(**Temp **!=** **null)**

**{**

sum **+=** Temp**.**typeTax**.**getAmount**();**

Temp **=** Temp**.**Next**;**

**}**

**return** sum**;**

**}**

**public** string getInfo**()** //получить информацию об одном налогоплательщике

**{**

string getinfo **=** "\nSurname taxpayer: " **+** surname**;**

Temp **=** pHead**.**First**;**

**while(**Temp **!=** **null)**

**{**

getinfo **+=** "\nType tax: " **+** Temp**.**typeTax**.**getTypeTax**();**

getinfo **+=** "\nAmount: " **+** Temp**.**typeTax**.**getAmount**();**

Temp **=** Temp**.**Next**;**

**}**

getinfo **+=** "\nsumm: " **+** sumTax**()** **+** "\n\n"**;**

**return** getinfo**;**

**}**

**public** string getInfoFor**()** //получить информацию об одном налогоплательщике

**{**

string getinfo **=** Convert**.**ToString**(**count**);**

Temp **=** pHead**.**First**;**

**while** **(**Temp **!=** **null)**

**{**

getinfo **+=** "\n" **+**Temp**.**typeTax**.**getTypeTax**();**

getinfo **+=** "\n" **+** Temp**.**typeTax**.**getAmount**();**

Temp **=** Temp**.**Next**;**

**}**

**return** getinfo**;**

**}**

**}**

class dinList //класс для записи элементов в список

**{**

**public** tax typeTax**;** //данные о налогоплательщике

**private** dinList next**,**first**,**last**;** //ссылка на следующий элемент в списке

**public** dinList**(**tax \_typeTax**)**//конструктор

**{**

typeTax **=** \_typeTax**;**

next **=** **null;**

first **=** **null;**

last **=** **null;**

**}**

**public** dinList Next **{** **get** **=>** next**;** **set** **=>** next **=** **value;** **}**

**public** dinList First **{** **get** **=>** first**;** **set** **=>** first **=** **value;** **}**

**public** dinList Last **{** **get** **=>** last**;** **set** **=>** last **=** **value;** **}**

**}**

**}**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Text**;**

**namespace** step3

**{**

class tax

**{**

**private** string typeTax**;** //тип налога

**private** int amount**;** //сумма налога

**public** tax**(**string atypeTax**,** int aMount**)**//конструктор

**{**

**this.**typeTax **=** atypeTax**;**

**this.**amount **=** aMount**;**

**}**

**public** string getTypeTax**(){** **return** typeTax**;** **}** //получить тип налога

**public** void setTypeTax**(**string \_type**){** **this.**typeTax **=** \_type**;** **}** //поменять тип налога

**public** int getAmount**()** **{** **return** amount**;** **}** //получить сумму налога

**public** void setAmount**(**int \_amount**)** **{** **this.**amount **=** \_amount**;** **}** //поменять сумму налога

**}**

**}**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Text**;**

**namespace** step3

**{**

class registry

**{**

**private** string nameRegistr**;** //название регистра

**private** taxpayer**[]** taxs**;** //массив объектов-налогоплательщиков

**private** int sizeArr**,**count**;** //стартовый размер массива

**private** int first**,** last**,**summa**;**

**public** registry**(**string \_nameRegistr**,**int \_sizeArr**)** //конструктор

**{**

nameRegistr **=** \_nameRegistr**;**

sizeArr **=** \_sizeArr**;**

taxs **=** **new** taxpayer**[**sizeArr**];**

first **=** 0**;** last **=** 0**;** count **=** 0**;**

**}**

**public** int Count **{** **get** **{** **return** count**;** **}** **}**

**public** void addtaxpayer**(**string \_surname**,** long \_INN**)** //добавление налогоплательщика

**{**

**if** **(**count **>** 0.75 **\*** sizeArr**)** **{** growthArr**();** **}**

taxpayer newTaxPay **=** **new** taxpayer**(**\_surname**,** \_INN**);**

taxs**[**last**]** **=** newTaxPay**;**

last**++;** count**++;**

**}**

**public** taxpayer searchtaxpayer**(**string \_surname**)** //поиск налогоплательщика

**{**

**for(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

**if(**taxs**[**i**]** **!=** **null** **&&** taxs**[**i**].**Surname**.**Equals**(**\_surname**))**

**{**

**return** taxs**[**i**];**

**}**

**}**

**return** **null;**

**}**

**public** void deltaxpayer**()** //удаление налогоплательщика

**{**

**if** **(**count **>** 0**)**

**{**

taxs**[**first**]** **=** **null;**

count**--;**

**for** **(**int i **=** **(**first **+** 1**);** i **<** last**;** i**++)**

**{**

taxs**[**i **-** 1**]** **=** taxs**[**i**];**

**}**

taxs**[**last**-**1**]** **=** **null;**

last**--;**

**if** **(**count **<** **(**0.25 **\*** sizeArr**))** **{** growthArr**();** **}**

**}**

**}**

**public** string Info**()** //информация по всему регистру

**{**

string info **=** "\nName registry - " **+** nameRegistr**;**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

info **+=**"\n" **+** taxs**[**i**].**getInfo**();**

**}**

info **+=** "\nSumm all taxpayer's: " **+** getSum**()** **+** "\n\n"**;**

**return** info**;**

**}**

**public** string InfoForSave**()**

**{**

string info **=** Convert**.**ToString**(**Count**)+** "\n"**;**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

info **+=**"\n" **+** taxs**[**i**].**Surname**;**

info **+=** "\n" **+** taxs**[**i**].**IINN**;**

**}**

**for** **(**int i **=** 0**;** i **<**count**;** i**++)**

**{**

info **+=** "\n" **+** taxs**[**i**].**getInfoFor**();**

**}**

**return** info**;**

**}**

**public** int getSum**()** //сумма всех задолжностей

**{**

summa **=** 0**;**

**for(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

summa **+=** taxs**[**i**].**sumTax**();**

**}**

**return** summa**;**

**}**

**private** void growthArr**()** //увеличить размер массива

**{**

**if** **(**count **>** 0.8 **\*** sizeArr**)**

**{**

taxpayer**[]** vspom **=** **new** taxpayer**[**sizeArr **\*** 2**];**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

vspom**[**i**]** **=** taxs**[**i**];**

**}**

sizeArr **\*=** 2**;**

taxs **=** **new** taxpayer**[**sizeArr**];**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

taxs**[**i**]** **=** vspom**[**i**];**

**}**

**}**

**else**

**{**

taxpayer**[]** vspom **=** **new** taxpayer**[**sizeArr **/** 2**];**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

vspom**[**i**]** **=** taxs**[**i**];**

**}**

sizeArr **/=** 2**;**

taxs **=** **new** taxpayer**[**sizeArr**];**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

taxs**[**i**]** **=** vspom**[**i**];**

**}**

**}**

**}**

**}**

**}**