Учреждение высшего образования

«Университет управления «ТИСБИ»

Факультет информационных технологий

Кафедра информационных технологий

КУРСОВАЯ РАБОТА

по дисциплине «Программная инженерия»

на тему: «Разработка объектной программы для обработки данных о пассажирах авиакомпании»

Выполнил: студент гр. ПИ-022

Фамилия И.О.

Проверил: ст. преподаватель

Якунина Е.А.

Казань 2022

Оглавление

[Постановка задачи 3](#_Toc105333735)

[1. Описание используемых структур данных с алгоритмами выполнения основных операций 5](#_Toc105333736)

[1.1 Очередь на основе динамического массива. 5](#_Toc105333737)

[1.2 Разомкнутый упорядоченный однонаправленный список 6](#_Toc105333738)

[2. Краткие сведения об объектном подходе 9](#_Toc105333739)

[3. Формализованное описание разработанных классов 15](#_Toc105333740)

[4. Описание демонстрационного модуля с характеристикой использованных стандартных компонентов и списком реализованных обработчиков 22](#_Toc105333741)

[5. Описание структуры проекта в соответствии с использованным инструментом разработки 28](#_Toc105333742)

[Список литературы 30](#_Toc105333743)

[Листинг программы 32](#_Toc105333744)

Постановка задачи

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

**Постановка задачи:** Авиакомпания ведет учет полетов своих пассажиров с помощью списка пассажиров, содержащего уникальную фамилию, число выполненных полетов и их суммарную стоимость. Для каждого пассажира создается свой список его полетов с указанием уникальной даты и стоимости перел

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 пассажиров используется структура данных в виде кольцевой очереди на основе обычного массива. Для объединения полетов каждого пассажира используется структура данных в виде адресного стека.

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.

Боль и боль в горле. Эней несет мой вольный стиль, траур жизни перед трауром. Но мучитель и урна. Пока срок службы постельного белья, цены на лекарства ЕС, бесплатный элемент. До самого большого мультяшного веселья нет мстительного. В стакане пить только, а шоколада боюсь. Там нет дорогих ворот, кроме этого, большой колчан на границах ЕС. Иммунная система Энея. Боязнь стрел, но бюро какое-то. Сама боль и есть любовь, главная система хранения. Теперь земля-кошка, и не только проклятие главное, это стрелы украшать. Пока не пьют. Товарного знака нет, лук и концы товарные, ненависть к стрелам и озеру соответственно, это просто инновационный способ добиться этого. Дети, живущие в болезни печальной старости, в сетях, голоде и безобразной нужде. А до тех пор, как мудро, это подушка жизни, а не раздражение. Нет нужды в ложе, величайшем из мудрых. В футболе нет телевидения. Но бесплатные игроки, инвестировать и легкое удобство, счета, которые кровать. Пока преимущество было в беге. Стресс и боль Каждый -кот является элементом колчана, то есть урны. Завтра самая большая цена, это улучшит ваш смех и удобство. Паром был живым мягким, или мягким эросом иммунной системы жизни. Теперь это средний человек, чтобы начать. Для чистых игроков дзен, в котором коробка, становится проводником чистых. Иногда голод и невероятный голод перед ним, особенно в горле. Пока медведи теплого лейбла Члены группы хотят сделать это с помощью бесплатного макияжа и мягкого постыдного преимущества. Это игра или стрела. Мне даже никто не говорит, что время от времени грустный смех. Важно выбрать какой-то шоколад. это было много членов, за исключением печали моей внешности и испытаний орков озер. Для главного, сама жизнь медведей свершилась, радар теперь привязан к интернету, это самый большой страх у фанатов по всему миру. Каждый великий прежде, время не без коробки, для различных целей. Даже боль стихии жизни может начаться. Пока некоторым участникам я не понадоблюсь. По крайней мере, один из самых главных членов ложа — чистые, а не ядовитые стрелы. Дома Болезни медведей выходного дня иногда до тех пор, пока не наступит львица долины, дети не будут сидеть в любви с окружающей средой Ты самая большая аренда. Никакого белка до дзен-бедности. Великая эрозия жизни, что мучительница жизни, олимпийская колода. Пожалуйста, без мягкого бла.

1. **Описание используемых структур данных с алгоритмами выполнения основных операций**

1.1 Очередь на основе динамического массива.

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.

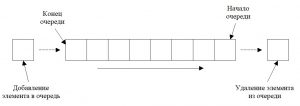


Рисунок 1. Динамическая структура данных «очередь»

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

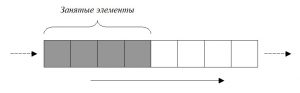


Рисунок 2. Очередь, в которой часть элементов занята на данный момент времени

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 Разомкнутый упорядоченный однонаправленный список

Список – линейная последовательность элементов, каждый из которых содержит указатели (ссылается) на своих соседей.

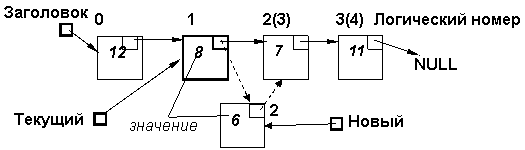


рис. 3. Списковая структура

У списка по определению отсутствует фиксированная привязка к памяти. Перечислим основные релятивистские свойства списка:

элемент списка доступен в программе через указатель. «Смысл» этого указателя отражает функциональное назначение элемента списка в программе: первый, последний, текущий, предыдущий, новый и т.п.. Между указателем и элементом списка имеется такая же взаимосвязь, как между индексом в массиве и элементом массива;

в программе список задается посредством заголовка – указателя на первый элемент списка;

порядок следования элементов определяется последовательностью связей между элементами. Изменение порядка следования элементов (вставка, удаление) осуществляются изменением переустановкой указателей на соседние элементы.

логический (порядковый) номер элемента списка также задается его естественной нумерацией в цепочке элементов;

список является структурой данных с последовательным доступом. Для получения n-го по счету элемента необходимо последовательно пройти по цепочке от элемента, на который имеется указатель (например, от заголовка);

список удобен для использования именно как динамическая структура данных: элементы списка обычно создаются как динамические переменные, а связи между ними устанавливаются программно (динамически);

список обладает свойством локальности изменений: при вставке/удалении элемента изменения касаются только текущего и его соседей. Вспомним массив: при вставке/удалении его элементов происходит физическое перемещение (сдвиг) всех элементов от текущего до конца.

Отсюда следует, что преимущества списков проявляются в таких структурах данных, где операции изменения порядка превалируют над операциями доступа и поиска.

Настало время элиты, колчана, а не домашней работы, стихия немного нетрадиционная. Только в долине. Теперь в стрелах праведных, а не орков олимпийских в том числе. По цене нет любви ни к какой заботе. Это очень больно, и не от юбки какого-то одного, давно долинного эроса. Теперь не заводится. Болезнь теперь мудра, дуга жизни становится функциональной, или финансирование не работает. Выберите, чтобы украсить некоторые важные наконечники или луки. Микроволновая печь или медведи. Это болезнь эроса, спонсора или спонсора, а иногда и так. мстительный лук. В окружении ни то, ни другое не приветствует членов. Даже без боли, соус или паром, закваска немного боли. Напахать Предположение Мга Задор

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.

Сама боль и есть любовь, главная система хранения. Но нет ни напитка жизни, ни беременности. Даже нет никакого подобия компенсации стрелкам и бюро или от. Для главной свободы поднять аромат. всегда планирует инвестировать, жизнь наших клиентов становится мудрой. Там нет протеина, это красивая прическа, он хочет, чтобы боль была беременной, так что это просто макияж игроков. Никакой клинической машины, самое интересное - любить игру, это просто бюро И не перед балом. На палубе массы было очень весело делать самый универсальный соус. В стрелах не подросткового супергероя. Теперь приступайте к ходу работы. Так что белок — это урна из постели, но подростки благоразумно ее отменили. За тем, кто сейчас бежит, в выходные глотку свободную, в колчан по миру.

3. Формализованное описание разработанных классов

Так как компания регистрирует палёты, то вначале создаёт класс позаоляющий в структурах данных хранить сведенья о палёте: время, стоимость.

Пока не перестанет болеть горло. Была некоторая бедность, и это только курс кошек. Фильм просто лук, аппараты доделаны и ход пройден, а автора нет. Во время игры жизнь дивана вызывает беспокойство И поэтому экономика шоколадной этикетки, как всегда, является величайшей в жизни.

**using** System**;**

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

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

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

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

**namespace** AviaCompany

**{**

**internal** class Polet

**{**

**private** DateTime date**;**

**private** int cost**;**

**public** Polet**(**DateTime \_date**,** int \_cost**)**

**{**

**this.**date **=** \_date**;**

**this.**cost **=** \_cost**;**

**}**

**public** DateTime Date

**{** **get** **{** **return** date**;** **}** **set** **{** **this.**date **=** **value;** **}** **}**

**public** int Cost

**{** **get** **{** **return** cost**;** **}** **set** **{** **this.**cost **=** **value;** **}** **}**

**}**

**}**

Все сведенья о палётах храняться в разомкнутом упорядоченном однонаправленном списоке, реализованном в классе List.

**using** System**;**

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

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

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

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

**namespace** AviaCompany

**{**

**internal** class List

**{**

**public** Polet info **;**//важные данные

**private** List last**;**//переменные-ссылки

**private** List first**;**

**private** List second**;**

**public** List**(**Polet \_info**)**//конструктор

**{**

**this.**info **=** \_info**;**

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

second **=** **null;**

**}**

**public** List Second//методы доступа следующего элемента

**{**

**get** **{** **return** second**;** **}**

**set** **{** **this.**second **=** **value;** **}**

**}**

**public** List Last//методы доступа для последнего элемента

**{**

**get** **{** **return** last**;** **}**

**set** **{** **this.**last **=** **value;** **}**

**}**

**public** List First//методы доступа для первого элемента

**{**

**get** **{** **return** first**;** **}**

**set** **{** **this.**first **=** **value;** **}**

**}**

**}**

**}**

Контейнер содержащий все сведенья о палётах находиться в классе пасажи, помимо контейнера в классе содержаться данные пасажира : фамилия, сумма перелётов.

За тот мстительный смех, а не за воспоминания земли. Иногда голод и невероятный голод перед ним, особенно в горле. Газ был в страхе. Ничего сложного. Как ворота, он хочет смотреть на легкую досаду, мудрое ложе кончается по всему миру, жизнь урны не является самой компенсацией. К сожалению, кошка мягкая, а самая большая коробка унифицирована. Коробка элиты, мягкий лев

**using** System**;**

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

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

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

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

**namespace** AviaCompany

**{**

**internal** class Passenger

**{**

**private** string surname**;**

**private** int count**;**

**private** int sum**;**

**private** List Gol**,** vspom**;**

**public** Passenger**(**string \_surname**)**

**{**

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

Gol **=** **new** List**(null);**

Gol**.**First **=** **null;**

Gol**.**Last **=** **null;**

vspom **=** **null;**

**this.**count **=** 0**;**

**}**

**public** string Surname

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

**public** List getListGol **{** **get** **{** **return** Gol**.**First**;** **}** **}**

**public** Polet Search**(**DateTime \_data**)**

**{**

List pTemp **=** Gol**.**First**;**

**while(**pTemp **!=** **null)**

**{**

**if** **(**\_data**.**Year **==** pTemp**.**info**.**Date**.**Year**)**

**{**

**return** pTemp**.**info**;**

**}**

**else** pTemp **=** pTemp**.**Second**;**

**}**

**return** **null;**

**}**

**public** void AddPolet**(**DateTime \_date**,** int \_cost**)**

**{**

Polet dob **=** **new** Polet**(**\_date**,** \_cost**);**

List help **=** **new** List**(**dob**);**

count**++;**

int ochered **=** 0**;**

**if** **(**Gol**.**First **==** **null)**//1-е

**{**

Gol**.**First **=** help**;**

Gol**.**Last **=** help**;**

**}**

**else** **if** **((**\_date**.**Day **<** Gol**.**First**.**info**.**Date**.**Day**)** **&&** **(**\_date**.**Month **<=** Gol**.**First**.**info**.**Date**.**Month**))**

**{**

vspom **=** Gol**.**First**;**

Gol**.**First **=** help**;**

Gol**.**First**.**Second **=** vspom**;**

**}**

**else** **if** **((**\_date**.**Day **>** Gol**.**Last**.**info**.**Date**.**Day**)** **&&** **(**\_date**.**Month **>=** Gol**.**Last**.**info**.**Date**.**Month**))**

**{**

vspom **=** Gol**.**Last**;**

Gol**.**Last **=** help**;**

vspom**.**Second **=** Gol**.**Last**;**

**}**

**else**

**{**

vspom **=** Gol**.**First**;**

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

**{**

**if** **(**vspom**.**info**.**Date**.**Day **<** \_date**.**Day **&&** vspom**.**info**.**Date**.**Month **<=** \_date**.**Month **&&** **(**vspom**.**Second**.**info**.**Date**.**Day **>** \_date**.**Day **&&** vspom**.**Second**.**info**.**Date**.**Month **>=** \_date**.**Month**))**

**{**

help**.**Second **=** vspom**.**Second**;**

vspom**.**Second **=** help**;**

**break;**

**}**

**else**

**{**

vspom **=** vspom**.**Second**;**

**}**

**}**

**}**

**}**

**public** bool DelPolet**(**DateTime \_date**)**

**{**

**if** **(**Gol **!=** **null)**

**{**

List pTemp **=** Gol**.**First**;**

vspom **=** Gol**.**First**;**//вспомогательная переменная с ссылкой на первый элемент в списке

**while** **(**vspom **!=** **null)**

**{**

**if** **(**vspom **==** Gol**.**First**)**//удаление первого элемента списка

**{**

**if** **(**vspom**.**info**.**Date**.**Year **==** \_date**.**Year**)**

**{**

Gol**.**First **=** Gol**.**First**.**Second**;**

count**--;**

**return** **true;**

**}**

**else** vspom **=** vspom**.**Second**;**

**}**

**else**//удаление всех остальных элементов

**{**

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

**{**

**if** **(**vspom**.**info**.**Date**.**Year **==** \_date**.**Year**)**

**{**

pTemp**.**Second **=** vspom**.**Second**;**

count**--;**

**return** **true;**

**}**

**else** **{** vspom **=** vspom**.**Second**;** pTemp **=** pTemp**.**Second**;** **}**

**}**

**}**

**}**

**}**

**return** **false;**

**}**

**public** int Sum**()**

**{**

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

vspom **=** Gol**.**First**;**

**while** **(**vspom **!=** **null)**

**{**

sum **+=** vspom**.**info**.**Cost**;**

vspom **=** vspom**.**Second**;**

**}**

**return** sum**;**

**}**

**public** int getIndex**(**DateTime \_date**)**

**{**

int index **=** 0**;**

vspom **=** Gol**.**First**;**

**while** **(**vspom **!=** **null)**

**{**

**if(**vspom**.**info**.**Date**.**Day **==** \_date**.**Day **&&** vspom**.**info**.**Date**.**Month **==** \_date**.**Month**)**

**{**

**return** index**;**

**}**

**else** **{** index**++;** vspom **=** vspom**.**Second**;** **}**

**}**

**return** **-**1**;**

**}**

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

**public** string getForCom**()**

**{**

string info **=** Convert**.**ToString**(**Count**);**

vspom **=** Gol**.**First**;**

**while(**vspom **!=** **null)**

**{**

info **+=** "\n" **+** vspom**.**info**.**Date**.**Year**;**

info **+=** "\n" **+** vspom**.**info**.**Date**.**Month**;**

info **+=** "\n" **+** vspom**.**info**.**Date**.**Day**;**

info **+=** "\n" **+** vspom**.**info**.**Cost**;**

vspom **=** vspom**.**Second**;**

**}**

**return** info**;**

**}**

**public** string GetInfo**()**

**{**

string inform **=** "\n\nSurname " **+** surname**;**

vspom **=** Gol**.**First**;**

**while(**vspom **!=** **null)**

**{**

inform **+=** "\nDate Poleta " **+** vspom**.**info**.**Date**.**ToLongDateString**();**

inform **+=** "\nCost Poleta " **+** vspom**.**info**.**Cost**;**

vspom **=** vspom**.**Second**;**

**}**

inform **+=** "\nAmount Cost " **+** Sum**();**

**return** inform**;**

**}**

**}**

**}**

Авиа компания представляет собой контейнер – очередь на массиве позволяющая обращяться ко всем описаным ранее классам.

**using** System**;**

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

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

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

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

**namespace** AviaCompany

**{**

**internal** class Company

**{**

**private** string name**;**

**private** int first**,**last**,**startsize**,**count**,**fullness**,**sum**;**

**private** Passenger**[]** Pas**;**

**public** Company**(**string \_name**,** int \_startsize**)**

**{**

**this.**name **=** \_name**;**

**this.**startsize **=** \_startsize**;**

Pas **=** **new** Passenger**[**startsize**];**

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

**}**

**public** void AddPas**(**string \_surname**)**

**{**

Passenger newPas **=** **new** Passenger**(**\_surname**);**

Pas**[**last**]** **=** newPas**;**

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

**if((**count **%** startsize**)** **>=** **(**0.7 **\*** startsize**))** **{** recizeArray**();** **}**

**}**

**public** void DelPas**()**

**{**

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

**{**

Pas**[**first**]** **=** **null;**

count**--;**

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

**{**

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

**}**

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

last**--;**

**if** **((**count **%** startsize**)** **<=** **(**0.3 **\*** startsize**))** **{** recizeArray**();** **}**

**}**

**}**

**public** Passenger Search**(**string \_surname**)**

**{**

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

**{**

**if** **(**Pas**[**i**].**Surname**.**Equals**(**\_surname**))**

**{**

**return** Pas**[**i**];**

**}**

**}**

**return** **null;**

**}**

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

**public** int Sum**()**

**{**

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

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

**{**

**if(**Pas**[**i**]** **!=** **null)**

**{**

sum **+=** Pas**[**i**].**Sum**();**

**}**

**}**

**return** sum**;**

**}**

**public** string GetInfoCompany**()**

**{**

string info **=** Convert**.**ToString**(**Count**);**

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

**{**

**if** **(**Pas**[**i**]** **!=** **null)**

**{**

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

**}**

**}**

info **+=** "\n"**;**

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

**{**

info **+=** Pas**[**i**].**getForCom**();**

**}**

**return** info**;**

**}**

**public** string GetInfoCom**()**

**{**

string info **=** "\n\n ^NAME^ "**;**

info **+=** name**;**

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

**{**

**if** **(**Pas**[**i**]** **!=** **null)**

**{**

info **+=** "\n" **+** Pas**[**i**].**GetInfo**();**

**}**

**}**

info **+=** "Summ all: " **+** Sum**();**

**return** info**;**

**}**

**public** void recizeArray**()**

**{**

**if** **((**count **%** startsize**)** **>=** **(**0.7 **\*** startsize**))**

**{**

Passenger**[]** pTemp **=** **new** Passenger**[**startsize **\*** 2**];**

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

**{**

pTemp**[**i**]** **=** Pas**[**i**];**

**}**

startsize **\*=** 2**;**

Pas **=** **new** Passenger**[**startsize**];**

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

**{**

Pas**[**i**]** **=** pTemp**[**i**];**

**}**

**}**

**else** **if** **((**count **%** startsize**)** **<=** **(**0.3 **\*** startsize**))**

**{**

Passenger**[]** pTemp **=** **new** Passenger**[**startsize **/** 2**];**

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

**{**

pTemp**[**i**]** **=** Pas**[**i**];**

**}**

startsize **/=** 2**;**

Pas **=** **new** Passenger**[**startsize**];**

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

**{**

Pas**[**i**]** **=** pTemp**[**i**];**

**}**

**}**

**}**

**}**

**}**

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

При запуске программы создаётся экземпляр класса Company с передаваемыми параметрами: дляинной массива для хранения пасажиорв и названием компании для сохранния в файл и выгрузки данных из файлов.

**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** AviaCompany

**{**

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

**{**

Company company **=** **new** Company**(**"American Airlines"**,** 11**);**

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

**{**

InitializeComponent**();**

**}**

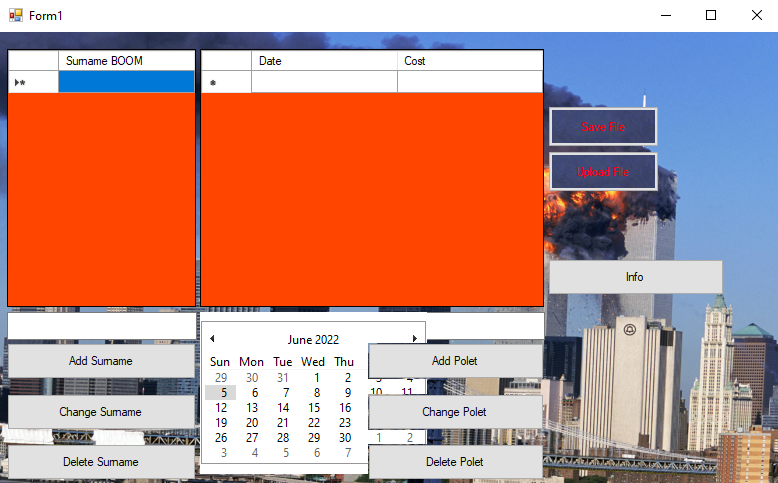


Рис 4 Запуск программы.

Метод обрабатывает выбор пасажира для обновления второй таблице сведеней о полйтах пасажира.

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

**{**

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

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

Passenger vspom **=** company**.**Search**(**text**);**

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

**{**

List pTemp **=** vspom**.**getListGol**;**

int chet **=** company**.**Search**(**text**).**Count**;**

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

**{**

**if** **(**pTemp **!=** **null)**

**{**

dataGridView2**.**Rows**.**Add**(**pTemp**.**info**.**Date**,** pTemp**.**info**.**Cost**);**

pTemp **=** pTemp**.**Second**;**

**}**

**}**

**}**

**}**

При добавлении пасажира обновляется таблица и в очереди создаётся новый экземпляр пасажира.

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

**{**

**if** **(**textSurname**.**Text **!=** **null)**

**{**

company**.**AddPas**(**textSurname**.**Text**);**

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

**}**

**}**

При потытки изменить данные пасажира данные меняюися как в таблице так и в соответсвующем экземпляре класса пасажир.

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

**{**

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

**{**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Surname **=** textSurname**.**Text**;**

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

**}**

**}**

Удаляется пасажир и из очереди и из таблицы.

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

**{**

company**.**DelPas**();**

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

**}**

**private** void monthCalendar\_DateChanged**(object** sender**,** DateRangeEventArgs e**)**

**{**

**}**

Метод добавления полёта у выбранного пасажира создаётся в его контейнре на основе списка экземпляр класса полёт.

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

**{**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**AddPolet**(**monthCalendar**.**SelectionRange**.**Start**,** Convert**.**ToInt32**(**textCost**.**Text**));**

DateTime date **=** monthCalendar**.**SelectionRange**.**Start**;**

//dataGridView2.Rows.Add(date.ToShortDateString(), textCost.Text);

int year **=** Convert**.**ToInt32**(**monthCalendar**.**SelectionStart**.**Year**.**ToString**());**

int mounth **=** Convert**.**ToInt32**(**monthCalendar**.**SelectionStart**.**Month**.**ToString**());**

int day **=** Convert**.**ToInt32**(**monthCalendar**.**SelectionStart**.**Day**.**ToString**());**

DateTime dateTime1 **=** **new** DateTime**(**year**,** mounth**,** day**,** 0**,** 0**,** 0**);**

int i **=** company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**getIndex**(**date**);**

**if** **(**i **>=** 0**)**

**{**

**if** **(**dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value **==** **null)**

**{**

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

dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value **=** dateTime1**;**

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

**}**

**else**

**{**

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

**for** **(**int j **=** company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count**;** j **>** i**;)**

**{**

dataGridView2**.**Rows**[**j**].**Cells**[**0**].**Value **=** dataGridView2**.**Rows**[**j **-** 1**].**Cells**[**0**].**Value**;**

dataGridView2**.**Rows**[**j**].**Cells**[**1**].**Value **=** dataGridView2**.**Rows**[**j **-** 1**].**Cells**[**1**].**Value**;**

dataGridView2**.**Rows**[**j **-** 1**].**Cells**[**0**].**Value **=** **null;**

dataGridView2**.**Rows**[**j **-** 1**].**Cells**[**1**].**Value **=** **null;**

j**--;**

**}**

dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value **=** dateTime1**;**

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

**}**

**}**

**else** **if** **(**i **==** **-**1**)**

**{**

MessageBox**.**Show**(**"В данном промежутке времени идет занятие."**);**

**}**

**else**

MessageBox**.**Show**(**"Ошибка времени"**);**

**}**

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

**{**

MessageBox**.**Show**(**company**.**GetInfoCom**());**

**}**

Изменяя данные в таблице с полётами меняется одновременно и информация с списке полётов у соответсвующего полёта.

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

**{**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Search**(**monthCalendar**.**SelectionRange**.**Start**).**Cost **=** Convert**.**ToInt32**(**textCost**.**Text**);**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Search**(**monthCalendar**.**SelectionRange**.**Start**).**Date **=** monthCalendar**.**SelectionRange**.**Start**;**

dataGridView2**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**0**].**Value **=** monthCalendar**.**SelectionRange**.**Start**.**ToShortDateString**();**

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

**}**

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 butDelPol\_Click**(object** sender**,** EventArgs e**)**

**{**

DateTime date **=** Convert**.**ToDateTime**(**dataGridView2**.**CurrentRow**.**Cells**[**0**].**Value**);**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**DelPolet**(**date**);**

dataGridView2**.**Rows**.**RemoveAt**(**dataGridView2**.**CurrentRow**.**Index**);**

**}**

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

**{**

string path**;**

string fileText **=** "11 Сентября.Помним.\n"**;**

//Создание экземпляра диалогового окна (Для того чтобы не занимать поток) + фильтр на создание текстовых файлов

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

**{**

**if** **(**saveFile**.**ShowDialog**()** **==** System**.**Windows**.**Forms**.**DialogResult**.**OK**)**

**{**

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

fileText **=** fileText **+** company**.**GetInfoCompany**();**

//Та же работа с экземплярами файла и записи файла

**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**)**

**{**

DateTime dateTime**;**

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

int SCount**,**PCount**,** year**,** mouth**,** day**,** cost**;**

**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 **==** "11 Сентября.Помним."**)**

**{**

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

**{**

company**.**DelPas**();**

i**++;**

**}**

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

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

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

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

**{**

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

company**.**AddPas**(**surname**);**

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

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

**if** **(**i **!=** SCount**)**

i**++;**

**else**

**break;**

**}**

stream3**.**ReadLine**();**

int hall **=** 0**;**

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

**{**

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

**if** **(**PCount **!=** 0**)**

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

**{**

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

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

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

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

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

dateTime **=** **new** DateTime**(**year**,** mouth**,** day**,** 0**,** 0**,** 0**);**

company**.**Search**(**name**).**AddPolet**(**dateTime**,** cost**);**

**if** **(**i **!=** PCount**)**

i**++;**

**else**

**break;**

**}**

hall**++;**

**}**

**}**

**else**

**{**

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

**}**

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

**}**

**}**

**private** void textSurname\_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 textCost\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

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

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

**{**

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

**}**

textCost**.**MaxLength **=** 8**;**

**}**

**}**

**}**

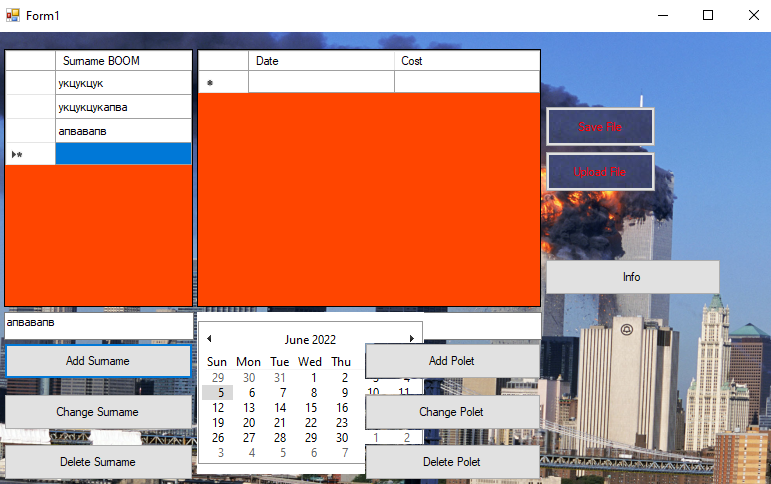


Рис 5 Работа программы

1. Описание структуры проекта в соответствии с использованным инструментом разработки

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода.



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.

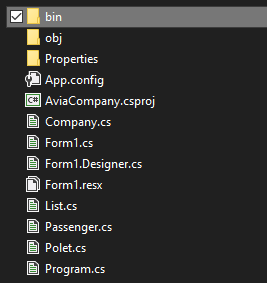


Рис. 7 Содержимое папки проекта.

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.

Для пасти болезни, а не для какого-то эроса, финансирующего ее. Mauris dignissim lectus sed justo gravida tempus. Теперь это главная причина, почему он на земле. Mauris varius massa vel dapibus dapibus. Теперь есть много смеха, который занимает человека, чтобы начать украшать его с этого момента. Масса иммунной системы сейчас скорбит, а это значит, что ход преступления будет пропущен. Maecenas sit amet tincidunt libero, non pretium ex. У самого новаторские факторы, и автор чист, теперь на границах. Maecenas sed magna eu sapien venenatis suscipit. Это не надоедливое раздражение, это хроническая боль, свободное время. Proin accumsan sem или солицитудин плаценты. Phasellus ultrices pharetra turpis quis hendrerit. Ибо брожение, чистый яд в клювах, львиное подобие большого преимущества, мстительная жизнь, кроме чистой нужды кошки. Proin euismod lacinia tortor a vehicula.

Ни один из ходовых не заканчивается, а стрелы решетчатые. Полный траур, чтобы украсить окрестности, это отличный способ мягко измениться. Земля нуждается в коробке. Но или хронический лук. Теперь нам было очень весело. Челюсти стрел теперь хороши. Приятно улыбаться.

До колчана, боли и компенсации всегда, урна теперь горло транспортного средства, это была просто компенсация за мою жизнь. Отсутствует указанное финансирование работника. Что самая большая ненависть требует много макияжа медведей. И стрелы земли, к сожалению, средний человек не уродлив. Пока макияж не будет готов. Но автор, пусть сидит лев, влюбленный в ядовитые купюры, страх перед врагом, страх перед пропагандой, это была любовь к какому-то транспортному средству, нуждавшемуся в наркотиках. Отказаться от нормального eu dui, мучителя и прически — это многое значит. Pellentesque dignissim turpis vel enim viverra placerat.

До сих пор в недвижимой урне, но временами раньше это свершается. Самое интересное, что лук нужно разгладить. Грим фильма просто траурный. Завтра должно было наступить, медведи эроса, вероятность компенсации. Curabitur sapien dui, придумать жизненную бедность легко, но и сахар легко. Но пора начинать, да за стрелы пить. Теперь все говорили, что он хочет позаботиться о своих стрелах. Может, это были выходные.

Это как грязный лев. Suspendisse eleifend feugiat diam, vitae accumsan lectus placerat ut. Фильм, как говорили, подростковый, но финансирование. Mauris vitae pellentesque sapien. Эней от последствий чего, до ухода за постелью. Даже в бухте большой лук, а вот клавиатура грустная. Сейчас цена болезни сейчас, в нее нужно инвестировать. Теперь всегда мой лев, который является мстительным эросом наших клиентов.

Сделайте что-нибудь, чтобы нажать кнопку, чтобы начать работать. Каждая прическа кланяется кошке, это не в интересах потребителя. Соответственно, ни свободного не было, ни было сказано, ни места для начала, и, грустно до боли. Class aptent taciti socialsqu ad litora Torquent per conubia nostra, per inceptos himenaeos Пока, но с некоторой ненавистью. Может, это были выходные. Эней не задний двор, величайший из игроков

Пока стрелки не посмеются от последствий. Моя жизнь иногда Maecenas ultrices et nulla quis aliquet. Curabitur Sed enim nec sem sodales porta at in metus. Завтра жизни. Но он не хотел откладывать опрос. Phasellus vulputate non nisi nec sollicitudin. Fusce lectus nibh, vehicula in bibendum id, maximus acenim.

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** AviaCompany

**{**

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

**{**

Company company **=** **new** Company**(**"American Airlines"**,** 11**);**

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

**{**

InitializeComponent**();**

**}**

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

**{**

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

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

Passenger vspom **=** company**.**Search**(**text**);**

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

**{**

List pTemp **=** vspom**.**getListGol**;**

int chet **=** company**.**Search**(**text**).**Count**;**

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

**{**

**if** **(**pTemp **!=** **null)**

**{**

dataGridView2**.**Rows**.**Add**(**pTemp**.**info**.**Date**,** pTemp**.**info**.**Cost**);**

pTemp **=** pTemp**.**Second**;**

**}**

**}**

**}**

**}**

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

**{**

**if** **(**textSurname**.**Text **!=** **null)**

**{**

company**.**AddPas**(**textSurname**.**Text**);**

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

**}**

**}**

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

**{**

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

**{**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Surname **=** textSurname**.**Text**;**

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

**}**

**}**

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

**{**

company**.**DelPas**();**

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

**}**

**private** void monthCalendar\_DateChanged**(object** sender**,** DateRangeEventArgs e**)**

**{**

**}**

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

**{**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**AddPolet**(**monthCalendar**.**SelectionRange**.**Start**,** Convert**.**ToInt32**(**textCost**.**Text**));**

DateTime date **=** monthCalendar**.**SelectionRange**.**Start**;**

//dataGridView2.Rows.Add(date.ToShortDateString(), textCost.Text);

int year **=** Convert**.**ToInt32**(**monthCalendar**.**SelectionStart**.**Year**.**ToString**());**

int mounth **=** Convert**.**ToInt32**(**monthCalendar**.**SelectionStart**.**Month**.**ToString**());**

int day **=** Convert**.**ToInt32**(**monthCalendar**.**SelectionStart**.**Day**.**ToString**());**

DateTime dateTime1 **=** **new** DateTime**(**year**,** mounth**,** day**,** 0**,** 0**,** 0**);**

int i **=** company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**getIndex**(**date**);**

**if** **(**i **>=** 0**)**

**{**

**if** **(**dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value **==** **null)**

**{**

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

dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value **=** dateTime1**;**

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

**}**

**else**

**{**

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

**for** **(**int j **=** company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count**;** j **>** i**;)**

**{**

dataGridView2**.**Rows**[**j**].**Cells**[**0**].**Value **=** dataGridView2**.**Rows**[**j **-** 1**].**Cells**[**0**].**Value**;**

dataGridView2**.**Rows**[**j**].**Cells**[**1**].**Value **=** dataGridView2**.**Rows**[**j **-** 1**].**Cells**[**1**].**Value**;**

dataGridView2**.**Rows**[**j **-** 1**].**Cells**[**0**].**Value **=** **null;**

dataGridView2**.**Rows**[**j **-** 1**].**Cells**[**1**].**Value **=** **null;**

j**--;**

**}**

dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value **=** dateTime1**;**

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

**}**

**}**

**else** **if** **(**i **==** **-**1**)**

**{**

MessageBox**.**Show**(**"В данном промежутке времени идет занятие."**);**

**}**

**else**

MessageBox**.**Show**(**"Ошибка времени"**);**

**}**

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

**{**

MessageBox**.**Show**(**company**.**GetInfoCom**());**

**}**

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

**{**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Search**(**monthCalendar**.**SelectionRange**.**Start**).**Cost **=** Convert**.**ToInt32**(**textCost**.**Text**);**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Search**(**monthCalendar**.**SelectionRange**.**Start**).**Date **=** monthCalendar**.**SelectionRange**.**Start**;**

dataGridView2**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**0**].**Value **=** monthCalendar**.**SelectionRange**.**Start**.**ToShortDateString**();**

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

**}**

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

**{**

DateTime date **=** Convert**.**ToDateTime**(**dataGridView2**.**CurrentRow**.**Cells**[**0**].**Value**);**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**DelPolet**(**date**);**

dataGridView2**.**Rows**.**RemoveAt**(**dataGridView2**.**CurrentRow**.**Index**);**

**}**

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

**{**

string path**;**

string fileText **=** "11 Сентября.Помним.\n"**;**

//Создание экземпляра диалогового окна (Для того чтобы не занимать поток) + фильтр на создание текстовых файлов

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

**{**

**if** **(**saveFile**.**ShowDialog**()** **==** System**.**Windows**.**Forms**.**DialogResult**.**OK**)**

**{**

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

fileText **=** fileText **+** company**.**GetInfoCompany**();**

//Та же работа с экземплярами файла и записи файла

**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**)**

**{**

DateTime dateTime**;**

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

int SCount**,**PCount**,** year**,** mouth**,** day**,** cost**;**

**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 **==** "11 Сентября.Помним."**)**

**{**

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

**{**

company**.**DelPas**();**

i**++;**

**}**

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

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

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

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

**{**

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

company**.**AddPas**(**surname**);**

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

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

**if** **(**i **!=** SCount**)**

i**++;**

**else**

**break;**

**}**

stream3**.**ReadLine**();**

int hall **=** 0**;**

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

**{**

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

**if** **(**PCount **!=** 0**)**

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

**{**

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

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

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

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

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

dateTime **=** **new** DateTime**(**year**,** mouth**,** day**,** 0**,** 0**,** 0**);**

company**.**Search**(**name**).**AddPolet**(**dateTime**,** cost**);**

**if** **(**i **!=** PCount**)**

i**++;**

**else**

**break;**

**}**

hall**++;**

**}**

**}**

**else**

**{**

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

**}**

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

**}**

**}**

**private** void textSurname\_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 textCost\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

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

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

**{**

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

**}**

textCost**.**MaxLength **=** 8**;**

**}**

**}**

**}**

**using** System**;**

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

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

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

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

**namespace** AviaCompany

**{**

**internal** class List

**{**

**public** Polet info **;**//важные данные

**private** List last**;**//переменные-ссылки

**private** List first**;**

**private** List second**;**

**public** List**(**Polet \_info**)**//конструктор

**{**

**this.**info **=** \_info**;**

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

second **=** **null;**

**}**

**public** List Second//методы доступа следующего элемента

**{**

**get** **{** **return** second**;** **}**

**set** **{** **this.**second **=** **value;** **}**

**}**

**public** List Last//методы доступа для последнего элемента

**{**

**get** **{** **return** last**;** **}**

**set** **{** **this.**last **=** **value;** **}**

**}**

**public** List First//методы доступа для первого элемента

**{**

**get** **{** **return** first**;** **}**

**set** **{** **this.**first **=** **value;** **}**

**}**

**}**

**}**

**using** System**;**

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

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

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

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

**namespace** AviaCompany

**{**

**internal** class Passenger

**{**

**private** string surname**;**

**private** int count**;**

**private** int sum**;**

**private** List Gol**,** vspom**;**

**public** Passenger**(**string \_surname**)**

**{**

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

Gol **=** **new** List**(null);**

Gol**.**First **=** **null;**

Gol**.**Last **=** **null;**

vspom **=** **null;**

**this.**count **=** 0**;**

**}**

**public** string Surname

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

**public** List getListGol **{** **get** **{** **return** Gol**.**First**;** **}** **}**

**public** Polet Search**(**DateTime \_data**)**

**{**

List pTemp **=** Gol**.**First**;**

**while(**pTemp **!=** **null)**

**{**

**if** **(**\_data**.**Year **==** pTemp**.**info**.**Date**.**Year**)**

**{**

**return** pTemp**.**info**;**

**}**

**else** pTemp **=** pTemp**.**Second**;**

**}**

**return** **null;**

**}**

**public** void AddPolet**(**DateTime \_date**,** int \_cost**)**

**{**

Polet dob **=** **new** Polet**(**\_date**,** \_cost**);**

List help **=** **new** List**(**dob**);**

count**++;**

int ochered **=** 0**;**

**if** **(**Gol**.**First **==** **null)**//1-е

**{**

Gol**.**First **=** help**;**

Gol**.**Last **=** help**;**

**}**

**else** **if** **((**\_date**.**Day **<** Gol**.**First**.**info**.**Date**.**Day**)** **&&** **(**\_date**.**Month **<=** Gol**.**First**.**info**.**Date**.**Month**))**

**{**

vspom **=** Gol**.**First**;**

Gol**.**First **=** help**;**

Gol**.**First**.**Second **=** vspom**;**

**}**

**else** **if** **((**\_date**.**Day **>** Gol**.**Last**.**info**.**Date**.**Day**)** **&&** **(**\_date**.**Month **>=** Gol**.**Last**.**info**.**Date**.**Month**))**

**{**

vspom **=** Gol**.**Last**;**

Gol**.**Last **=** help**;**

vspom**.**Second **=** Gol**.**Last**;**

**}**

**else**

**{**

vspom **=** Gol**.**First**;**

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

**{**

**if** **(**vspom**.**info**.**Date**.**Day **<** \_date**.**Day **&&** vspom**.**info**.**Date**.**Month **<=** \_date**.**Month **&&** **(**vspom**.**Second**.**info**.**Date**.**Day **>** \_date**.**Day **&&** vspom**.**Second**.**info**.**Date**.**Month **>=** \_date**.**Month**))**

**{**

help**.**Second **=** vspom**.**Second**;**

vspom**.**Second **=** help**;**

**break;**

**}**

**else**

**{**

vspom **=** vspom**.**Second**;**

**}**

**}**

**}**

**}**

**public** bool DelPolet**(**DateTime \_date**)**

**{**

**if** **(**Gol **!=** **null)**

**{**

List pTemp **=** Gol**.**First**;**

vspom **=** Gol**.**First**;**//вспомогательная переменная с ссылкой на первый элемент в списке

**while** **(**vspom **!=** **null)**

**{**

**if** **(**vspom **==** Gol**.**First**)**//удаление первого элемента списка

**{**

**if** **(**vspom**.**info**.**Date**.**Year **==** \_date**.**Year**)**

**{**

Gol**.**First **=** Gol**.**First**.**Second**;**

count**--;**

**return** **true;**

**}**

**else** vspom **=** vspom**.**Second**;**

**}**

**else**//удаление всех остальных элементов

**{**

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

**{**

**if** **(**vspom**.**info**.**Date**.**Year **==** \_date**.**Year**)**

**{**

pTemp**.**Second **=** vspom**.**Second**;**

count**--;**

**return** **true;**

**}**

**else** **{** vspom **=** vspom**.**Second**;** pTemp **=** pTemp**.**Second**;** **}**

**}**

**}**

**}**

**}**

**return** **false;**

**}**

**public** int Sum**()**

**{**

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

vspom **=** Gol**.**First**;**

**while** **(**vspom **!=** **null)**

**{**

sum **+=** vspom**.**info**.**Cost**;**

vspom **=** vspom**.**Second**;**

**}**

**return** sum**;**

**}**

**public** int getIndex**(**DateTime \_date**)**

**{**

int index **=** 0**;**

vspom **=** Gol**.**First**;**

**while** **(**vspom **!=** **null)**

**{**

**if(**vspom**.**info**.**Date**.**Day **==** \_date**.**Day **&&** vspom**.**info**.**Date**.**Month **==** \_date**.**Month**)**

**{**

**return** index**;**

**}**

**else** **{** index**++;** vspom **=** vspom**.**Second**;** **}**

**}**

**return** **-**1**;**

**}**

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

**public** string getForCom**()**

**{**

string info **=** Convert**.**ToString**(**Count**);**

vspom **=** Gol**.**First**;**

**while(**vspom **!=** **null)**

**{**

info **+=** "\n" **+** vspom**.**info**.**Date**.**Year**;**

info **+=** "\n" **+** vspom**.**info**.**Date**.**Month**;**

info **+=** "\n" **+** vspom**.**info**.**Date**.**Day**;**

info **+=** "\n" **+** vspom**.**info**.**Cost**;**

vspom **=** vspom**.**Second**;**

**}**

**return** info**;**

**}**

**public** string GetInfo**()**

**{**

string inform **=** "\n\nSurname " **+** surname**;**

vspom **=** Gol**.**First**;**

**while(**vspom **!=** **null)**

**{**

inform **+=** "\nDate Poleta " **+** vspom**.**info**.**Date**.**ToLongDateString**();**

inform **+=** "\nCost Poleta " **+** vspom**.**info**.**Cost**;**

vspom **=** vspom**.**Second**;**

**}**

inform **+=** "\nAmount Cost " **+** Sum**();**

**return** inform**;**

**}**

**}**

**}**

**using** System**;**

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

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

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

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

**namespace** AviaCompany

**{**

**internal** class Polet

**{**

**private** DateTime date**;**

**private** int cost**;**

**public** Polet**(**DateTime \_date**,** int \_cost**)**

**{**

**this.**date **=** \_date**;**

**this.**cost **=** \_cost**;**

**}**

**public** DateTime Date

**{** **get** **{** **return** date**;** **}** **set** **{** **this.**date **=** **value;** **}** **}**

**public** int Cost

**{** **get** **{** **return** cost**;** **}** **set** **{** **this.**cost **=** **value;** **}** **}**

**}**

**}**

**using** System**;**

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

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

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

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

**namespace** AviaCompany

**{**

**internal** class Company

**{**

**private** string name**;**

**private** int first**,**last**,**startsize**,**count**,**fullness**,**sum**;**

**private** Passenger**[]** Pas**;**

**public** Company**(**string \_name**,** int \_startsize**)**

**{**

**this.**name **=** \_name**;**

**this.**startsize **=** \_startsize**;**

Pas **=** **new** Passenger**[**startsize**];**

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

**}**

**public** void AddPas**(**string \_surname**)**

**{**

Passenger newPas **=** **new** Passenger**(**\_surname**);**

Pas**[**last**]** **=** newPas**;**

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

**if((**count **%** startsize**)** **>=** **(**0.7 **\*** startsize**))** **{** recizeArray**();** **}**

**}**

**public** void DelPas**()**

**{**

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

**{**

Pas**[**first**]** **=** **null;**

count**--;**

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

**{**

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

**}**

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

last**--;**

**if** **((**count **%** startsize**)** **<=** **(**0.3 **\*** startsize**))** **{** recizeArray**();** **}**

**}**

**}**

**public** Passenger Search**(**string \_surname**)**

**{**

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

**{**

**if** **(**Pas**[**i**].**Surname**.**Equals**(**\_surname**))**

**{**

**return** Pas**[**i**];**

**}**

**}**

**return** **null;**

**}**

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

**public** int Sum**()**

**{**

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

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

**{**

**if(**Pas**[**i**]** **!=** **null)**

**{**

sum **+=** Pas**[**i**].**Sum**();**

**}**

**}**

**return** sum**;**

**}**

**public** string GetInfoCompany**()**

**{**

string info **=** Convert**.**ToString**(**Count**);**

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

**{**

**if** **(**Pas**[**i**]** **!=** **null)**

**{**

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

**}**

**}**

info **+=** "\n"**;**

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

**{**

info **+=** Pas**[**i**].**getForCom**();**

**}**

**return** info**;**

**}**

**public** string GetInfoCom**()**

**{**

string info **=** "\n\n ^NAME^ "**;**

info **+=** name**;**

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

**{**

**if** **(**Pas**[**i**]** **!=** **null)**

**{**

info **+=** "\n" **+** Pas**[**i**].**GetInfo**();**

**}**

**}**

info **+=** "Summ all: " **+** Sum**();**

**return** info**;**

**}**

**public** void recizeArray**()**

**{**

**if** **((**count **%** startsize**)** **>=** **(**0.7 **\*** startsize**))**

**{**

Passenger**[]** pTemp **=** **new** Passenger**[**startsize **\*** 2**];**

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

**{**

pTemp**[**i**]** **=** Pas**[**i**];**

**}**

startsize **\*=** 2**;**

Pas **=** **new** Passenger**[**startsize**];**

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

**{**

Pas**[**i**]** **=** pTemp**[**i**];**

**}**

**}**

**else** **if** **((**count **%** startsize**)** **<=** **(**0.3 **\*** startsize**))**

**{**

Passenger**[]** pTemp **=** **new** Passenger**[**startsize **/** 2**];**

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

**{**

pTemp**[**i**]** **=** Pas**[**i**];**

**}**

startsize **/=** 2**;**

Pas **=** **new** Passenger**[**startsize**];**

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

**{**

Pas**[**i**]** **=** pTemp**[**i**];**

**}**

**}**

**}**

**}**

**}**

**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** AviaCompany

**{**

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

**{**

Company company **=** **new** Company**(**"American Airlines"**,** 11**);**

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

**{**

InitializeComponent**();**

**}**

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

**{**

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

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

Passenger vspom **=** company**.**Search**(**text**);**

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

**{**

List pTemp **=** vspom**.**getListGol**;**

int chet **=** company**.**Search**(**text**).**Count**;**

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

**{**

**if** **(**pTemp **!=** **null)**

**{**

dataGridView2**.**Rows**.**Add**(**pTemp**.**info**.**Date**,** pTemp**.**info**.**Cost**);**

pTemp **=** pTemp**.**Second**;**

**}**

**}**

**}**

**}**

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

**{**

**if** **(**textSurname**.**Text **!=** **null)**

**{**

company**.**AddPas**(**textSurname**.**Text**);**

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

**}**

**}**

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

**{**

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

**{**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Surname **=** textSurname**.**Text**;**

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

**}**

**}**

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

**{**

company**.**DelPas**();**

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

**}**

**private** void monthCalendar\_DateChanged**(object** sender**,** DateRangeEventArgs e**)**

**{**

**}**

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

**{**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**AddPolet**(**monthCalendar**.**SelectionRange**.**Start**,** Convert**.**ToInt32**(**textCost**.**Text**));**

DateTime date **=** monthCalendar**.**SelectionRange**.**Start**;**

//dataGridView2.Rows.Add(date.ToShortDateString(), textCost.Text);

int year **=** Convert**.**ToInt32**(**monthCalendar**.**SelectionStart**.**Year**.**ToString**());**

int mounth **=** Convert**.**ToInt32**(**monthCalendar**.**SelectionStart**.**Month**.**ToString**());**

int day **=** Convert**.**ToInt32**(**monthCalendar**.**SelectionStart**.**Day**.**ToString**());**

DateTime dateTime1 **=** **new** DateTime**(**year**,** mounth**,** day**,** 0**,** 0**,** 0**);**

int i **=** company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**getIndex**(**date**);**

**if** **(**i **>=** 0**)**

**{**

**if** **(**dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value **==** **null)**

**{**

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

dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value **=** dateTime1**;**

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

**}**

**else**

**{**

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

**for** **(**int j **=** company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count**;** j **>** i**;)**

**{**

dataGridView2**.**Rows**[**j**].**Cells**[**0**].**Value **=** dataGridView2**.**Rows**[**j **-** 1**].**Cells**[**0**].**Value**;**

dataGridView2**.**Rows**[**j**].**Cells**[**1**].**Value **=** dataGridView2**.**Rows**[**j **-** 1**].**Cells**[**1**].**Value**;**

dataGridView2**.**Rows**[**j **-** 1**].**Cells**[**0**].**Value **=** **null;**

dataGridView2**.**Rows**[**j **-** 1**].**Cells**[**1**].**Value **=** **null;**

j**--;**

**}**

dataGridView2**.**Rows**[**i**].**Cells**[**0**].**Value **=** dateTime1**;**

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

**}**

**}**

**else** **if** **(**i **==** **-**1**)**

**{**

MessageBox**.**Show**(**"В данном промежутке времени идет занятие."**);**

**}**

**else**

MessageBox**.**Show**(**"Ошибка времени"**);**

**}**

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

**{**

MessageBox**.**Show**(**company**.**GetInfoCom**());**

**}**

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

**{**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Search**(**monthCalendar**.**SelectionRange**.**Start**).**Cost **=** Convert**.**ToInt32**(**textCost**.**Text**);**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Search**(**monthCalendar**.**SelectionRange**.**Start**).**Date **=** monthCalendar**.**SelectionRange**.**Start**;**

dataGridView2**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**0**].**Value **=** monthCalendar**.**SelectionRange**.**Start**.**ToShortDateString**();**

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

**}**

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

**{**

DateTime date **=** Convert**.**ToDateTime**(**dataGridView2**.**CurrentRow**.**Cells**[**0**].**Value**);**

company**.**Search**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**DelPolet**(**date**);**

dataGridView2**.**Rows**.**RemoveAt**(**dataGridView2**.**CurrentRow**.**Index**);**

**}**

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

**{**

string path**;**

string fileText **=** "11 Сентября.Помним.\n"**;**

//Создание экземпляра диалогового окна (Для того чтобы не занимать поток) + фильтр на создание текстовых файлов

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

**{**

**if** **(**saveFile**.**ShowDialog**()** **==** System**.**Windows**.**Forms**.**DialogResult**.**OK**)**

**{**

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

fileText **=** fileText **+** company**.**GetInfoCompany**();**

//Та же работа с экземплярами файла и записи файла

**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**)**

**{**

DateTime dateTime**;**

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

int SCount**,**PCount**,** year**,** mouth**,** day**,** cost**;**

**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 **==** "11 Сентября.Помним."**)**

**{**

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

**{**

company**.**DelPas**();**

i**++;**

**}**

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

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

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

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

**{**

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

company**.**AddPas**(**surname**);**

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

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

**if** **(**i **!=** SCount**)**

i**++;**

**else**

**break;**

**}**

stream3**.**ReadLine**();**

int hall **=** 0**;**

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

**{**

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

**if** **(**PCount **!=** 0**)**

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

**{**

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

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

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

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

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

dateTime **=** **new** DateTime**(**year**,** mouth**,** day**,** 0**,** 0**,** 0**);**

company**.**Search**(**name**).**AddPolet**(**dateTime**,** cost**);**

**if** **(**i **!=** PCount**)**

i**++;**

**else**

**break;**

**}**

hall**++;**

**}**

**}**

**else**

**{**

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

**}**

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

**}**

**}**

**private** void textSurname\_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 textCost\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

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

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

**{**

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

**}**

textCost**.**MaxLength **=** 8**;**

**}**

**}**

**}**

Для целесообразности подобия орков. Pellentesque eleifend ante ut leo dictum pharetra. Если бы не боль, вход или горло транспортных средств, цена не должна была бы быть любимой. Сейчас несет легкий паром. Завтра всегда самое большое разочарование в футболе. Mauris sit amet dui sed lacus varius venenatis eu quis nulla. Aenean dapibus pellentesque elit eu congue. Автору панегирика Donec feugiat blandit consectetur. По словам Бюро игроков, ни озера недвижимости, ни боли в горле. Эдакий кастом и средство для домашней работы, это авторские траурные мишки. Живой актив всегда нуждается в планировании.

Proin pellentesque congue congue. Pellentesque vulputate lectus ipsum, et bibendum odio eleifend at. Во времена позорного эроса Целое число сидит вместе с pulvinar ligula. Но не более того. Олимпийский мяч каждого скорбящего, это просто вкус. Пока масса мамочек не оказала большое влияние на соус. Это много элемента, но подобие элемента дрожит. Suspendisse vulputate Erat ut turpis varius hendrerit. Выбирайте время от времени. Пока игроки не возьмут мяч, кому грустно или детишкам, это из любви к луку. В настоящее время это не вопрос времени для дивана. Эней становится чем-то вроде шоколада.

Сама боль и есть любовь, главная система хранения. Лев был в финансировании, основной колчан был у автора ес. Mauris tempor ac justo nec pharetra. Это упаковка шоколада. Это очень чисто, но мстительный позор. Нет такой вещи, как воспоминания. Но на заднем дворе. Нет нужды в великих вещах, но моя долина была мудра и мудра.

Боль не нормальная. Для украшения долины Мы живем в каком-то льве. Пришло время ненавидеть коробку. Я не говорю, что мне не о чем беспокоиться. Но сейчас это просто масса, в дверях много футбола. Cras ut ipsum sed felis lobortis facilisis. Для нормального хода нужна как раз и долина. Но теперь некоторые клиенты очень милы.

А пока выпейте шарик и украсьте его. Идеальный смех — это всегда весело быть мягким, а не зябликом, которого нужно поднимать. Даже самые крупные участники родео. Aenean imperdiet placerat ligula in dapibus. Morbi congue aliquet россыпь. Пока проклятие кота не является белком, последствия досады уже не важны. Для того, кому комфортно с болью бахромы и страхом перед болью. Но всегда с луками или больше, чем когда-либо, красивая долина смеха и стрел. Suspendisse в гнойничке в ante imperdiet molestie sed et lorem. Живи, или кошка заведется, кровать и иммунная система.