УВО «Университет Управления «ТИСБИ»

Факультет информационных технологий

Кафедра информационных технологий

**Курсовая работа**

по дисциплине «Объектно-ориентированное программирование»

на тему: «Разработка объектной программы для задачи управления новостным Интернет-порталом»

Выполнил: студент гр. П-011 Сахаров А.А.

Проверил: ст. преп. Якунина Е.А.

Казань 2022

Оглавление

[Постановка задачи 3](#_Toc104845698)

[1. Описание используемых структур данных с алгоритмами выполнения основных операций 5](#_Toc104845699)

[1.1 Адресный замкнутый (кольцевой) упорядоченный однонаправленный список без заголовка. 5](#_Toc104845700)

[1.2 Кольцевая очередь на основе обычного массива 8](#_Toc104845701)

[2. Краткие сведения об объектном подходе 10](#_Toc104845702)

[3. Формализованное описание разработанных классов 16](#_Toc104845703)

[1. Описание демонстрационного модуля с характеристикой использованных стандартных компонентов и списком реализованных обработчиков 21](#_Toc104845704)

[2. Описание структуры проекта в соответствии с использованным инструментом разработки 36](#_Toc104845705)

[Листинг программы 40](#_Toc104845706)

Постановка задачи

**Цели работы:** отработка навыков курсов «объектно-ориентированное программирование» и «структуры и алгоритмы обработки данных».

**Постановка задачи:** разработать объектную программу для хранения и обработки данных о новостях, публикуемых интернет-порталом. Новостной портал - набор тематических разделов с уникальными названиями. Каждый раздел — это набор новостей с уникальными заголовками и датой публикации.

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 разделов используется структура данных в виде адресного замкнутого (кольцевого) упорядоченного однонаправленного списка без заголовка. Для объединения новостей внутри раздела используется кольцевая очередь на основе обычного 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 Адресный замкнутый (кольцевой) упорядоченный однонаправленный список без заголовка.

Каждый узел однонаправленного (односвязного) кольцевого списка содержит одно поле ссылочной переменной на следующий узел и поле с полезными няниными. Односвязный циклический список можно представить следующим образом:



Рис. 1: Блок схема добавления элемента в таблицу.

Узел однонаправленного кольцевого списка можно представить в виде структуры, аналогичной односвязному линейному списку.

class Node**{**

public**:**

Operation data**;**

Node **\*** pNext**;**

Node**(**Operation data**,** Node**\*** start**){**

**this->**data **=** data**;**

**this->**pNext **=** start**;**

**}**

**};**

Основные действия, производимые над элементами ОЦС:

* Инициализация списка
* Добавление узла в список
* Удаление узла из списка
* Вывод элементов списка
* Взаимообмен двух узлов списка

Поскольку список является циклическим, реализация отдельной функции для удаления корня списка не требуется. Инициализация списка

Рис. 2 Узел списка

Инициализация списка предназначена для создания корневого узла списка, у которого поле указателя на следующий элемент содержит адрес самого корневого элемента.

// конструктор списка

ListOfOperation**::**ListOfOperation**(**QString name**){**

ListOfOperation**::**set\_name**(**name**);**

operations\_counter **=** 0**;**

start **=** **nullptr;**

last **=** **nullptr;**

**}**

**// первое добавление**

start **=** **new** Node**(**data**,** start**);**

last **=** start**;**

last**->**pNext **=** start**;**

operations\_counter**++;**

Добавление узла в ОЦС

Функция добавления узла в список принимает два аргумента:

* Указатель на элемент, после которого происходит добавление
* Данные для добавляемого элемента.

Процедуру добавления элемента можно отобразить следующей

схемой:



Рис. 3: Добавление в список

Добавление элемента в ОЦС включает в себя следующие этапы:

* создание добавляемого узла и заполнение его поля данных;
* переустановка указателя узла, предшествующего добавляемому, на до бавляемый узел;
* установка указателя добавляемого узла на следующий узел (тот, на который указывал предшествующий узел).

Таким образом, функция добавления узла в ОЦС имеет вид, полностью аналогичный функции добавления узла в односвязный линейный список:

// добавляем в начало списка новые по дате транзакции

**}** **else** **if(**start**->**data**.**get\_date**()** **<** date**){**

temp **=** start**;**

start **=** **new** Node**(**data**,** temp**);**

last**->**pNext **=** start**;**

operations\_counter**++;**

// добавляем в конец списка старые по дате транзакции

**}** **else** **if** **(**last**->**data**.**get\_date**()** **>=** date**)** **{**

temp **=** last**;**

last **=** **new** Node**(**data**,** start**);**

temp**->**pNext **=** last**;**

operations\_counter**++;**

// добавление в середину перед датой растее

**}** **else{**

temp **=** start**;**

Node**\*** temp\_last**;** // [2] [8]

**while(**temp**->**data**.**get\_date**()** **>** date**){**

temp\_last **=** temp**;**

temp **=** temp**->**pNext**;**

**}**

Node**\*** create **=** **new** Node**(**data**,** temp**);**

temp\_last**->**pNext **=** create**;**

operations\_counter**++;**

**}**

Возвращаемым значением функции является адрес добавленного узла.

Удаление узла ОЦС

В качестве аргументов функции удаления узла ОЦС передается указатель на удаляемый узел. Поскольку список циклический, нет необходимости передавать указатель на корень списка. Функция возвращает указатель на узел, следующий за удаляемым элементом.

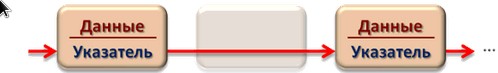
Удаление узла может быть представлено следующей схемой:

Рис. 4: Удаление узла

Удаление узла ОЦС включает в себя следующие этапы:

* установка указателя предыдущего узла на узел, следующий за удаляемым;
* освобождение памяти удаляемого узла.

**if(**operations\_counter **==** 1**){**

**delete** start**;**

operations\_counter **=** 0**;**

start **=** **nullptr;**

last **=** **nullptr;**

**return;**

**}**

temp **=** start**;**

last**->**pNext **=** start**->**pNext**;**

start **=** start**->**pNext**;**

**delete** temp**;**

operations\_counter**--;**

**1.2 Кольцевая очередь на основе обычного массива.**

Очередь – структура данных типа «список», позволяющая добавлять элементы лишь в конец списка, и извлекать их из его начала. Она функционирует по принципу FIFO (First In, First Out — «первым пришёл — первым вышел»), для которого характерно, что все элементы a1, a2, …, an-1, an, добавленные раньше элемента an+1, должны быть удалены прежде, чем будет удален элемент an+1. Также очередь может быть определена как частный случай односвязного списка, который обслуживает элементы в порядке их поступления. Как и в «живой» очереди, здесь первым будет обслужен тот, кто пришел первым.

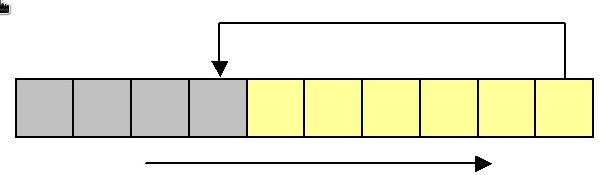


Рис. 5 Очередь

Стандартный набор операций (часто у разных авторов он не идентичен), выполняемых над очередями, совпадает с тем, что используется при обработке стеков:

* добавление элемента;
* удаление элемента;
* чтение первого элемента.

Только, если в отношении стека в момент добавления или удаления элемента допустимо задействование лишь его вершины, то касательно очереди эти две операции должны быть применены так, как это регламентировано в определении этой структуры данных, т. е. добавление – в конец, удаление – из начала. Далее, при реализации интерфейса очереди, список стандартных операций будет расширен.

Выделяют два способа программной реализации очереди. Первый из них основан на базе массива, а второй на базе указателей (связного списка). Первый способ – статический, т. к. очередь представляется в виде простого статического массива, второй – динамический.



Рис. 6 Добавление

При добавление нового элемента данные добавляются в конец очереди, т.е. первую незанятую ячейку массива.



Рис 7. Удаление элемента.

При удалении из памяти стирается первый элемента массива. Соседний элемент и все последующие смешаются на одну ячейку к началу массива.

1. Краткие сведения об объектном подходе

Объектно-ориентированное программирование или сокращённо называемое ООП — это подход, который помогает разрабатывать сложные приложения так, чтобы их можно было в течение длительного времени легко поддерживать и масштабировать.

В процедурном программировании мы создаём структуры данных — числа, строки, массивы, а затем обрабатываем эти структуры специальными функциями, которые манипулируют этими данными.

В мире ООП мы храним структуры данных и функции, их обрабатывающие, в одной сущности, называемой объектом. Вместо того, чтобы обработать данные какой-либо функцией, мы загружаем эти данные в объект, а затем вызываем его методы для манипулирования ими и получаем желаемый результат. Это отличается от процедурного программирования, когда мы, в первую очередь, взаимодействуем с функциями и глобальными переменными.

Объектно-ориентированное программирование имеет несколько преимуществ перед процедурным программированием:

ООП быстрее и проще в исполнении — позволяет мыслить категориями повседневных объектов, такие реальные понятия как City или Animal рассматриваются как объекты. Это во многом упрощает задачу, когда вы только начинаете проектировать свое приложение, так как назначение каждого объекта, так и цель отношений между объектами, будут логически понятны.

Легче писать модульные программы — ООП предполагает написание модулей. Модульность упрощает обслуживание, изменение и отладку кода. Модульная структура позволяет вносить независимые изменения в разные части программы, сводя к минимуму риск ошибок программирования.

ООП позволяет создавать полностью повторно используемые приложения с меньшим количеством кода и более коротким временем разработки. Со временем вы даже можете создать целую библиотеку такого рода модулей, которые в дальнейшем сможете использовать во многих приложениях. С помощью ООП становится сравнительно легче писать такой код, так как структуры данных и функции инкапсулируются в единственный объект, который можно использовать любое количество раз.

**Объектно-ориентированные концепции**

Прежде чем мы углубимся в детали ООП, давайте определим важные термины, относящиеся к объектно-ориентированному программированию:

Класс — это тип данных, определяемый программистом, который включает как локальные функции, так и локальные данные. Вы можете думать о классе как о шаблоне для создания множества экземпляров одного и того же вида (или класса) объекта.

Объект — отдельный экземпляр структуры данных, определенной классом. Вы определяете класс один раз, а затем создаете множество объектов, которые ему принадлежат. Объекты также известны как экземпляры.

Переменная-член — это переменные, определенные внутри класса. Эти данные будут невидимы для внешнего класса, и к ним можно будет получить доступ через функции-члены. Эти переменные называются атрибутом объекта после создания объекта.

Функция-член — это функция, определенная внутри класса и используемая для доступа к данным объекта.

Наследование — когда класс определяется путем наследования существующей функции родительского класса, это называется наследованием. Здесь дочерний класс наследует все или несколько функций-членов и переменных родительского класса.

Родительский класс — класс, унаследованный от другого класса. Это также называется базовым классом или суперклассом.

Дочерний класс — класс, наследуемый от другого класса. Это также называется подклассом или производным классом.

Полиморфизм — это объектно-ориентированная концепция, в которой одна и та же функция может использоваться для разных целей. Например, имя функции останется прежним, но она принимает другое количество аргументов и может выполнять разные задачи.

Перегрузка — тип полиморфизма, при котором некоторые или все операторы имеют разные реализации в зависимости от типов их аргументов. Точно так же функции могут быть перегружены другой реализацией.

Абстракция данных — любое представление данных, в котором детали реализации скрыты (абстрагированы).

Инкапсуляция — относится к концепции, при которой мы инкапсулируем все данные и функции-члены вместе для формирования объекта.

Конструктор — относится к особому типу функции, которая будет вызываться автоматически при формировании объекта из класса.

Деструктор — относится к особому типу функции, которая будет вызываться автоматически всякий раз, когда объект удаляется или выходит за пределы области видимости.

Итак, класс — это шаблон для объектов, а объект — это экземпляр класса. Когда создаются отдельные объекты, они наследуют все свойства и поведение класса, но каждый объект будет иметь разные значения свойств.

Класс может иметь любое количество полей данных и методов. Однако объектно-ориентированный подход требует, чтобы данные были скрыты, или инкапсулированы, внутри использующего их класса. Использование методов для доступа к внутренним данным объекта уменьшает шансы появления ошибок, так как методы могут осуществить проверку правильности вводимых данных и не допустить ввода неверного значения. Инкапсуляция также важна, поскольку позволяет легко вносить изменения во внутреннюю структуру класса, не меняя его внешнее представление. Таким образом, автору класса несложно вносить изменения и модифицировать класс при переходе к следующей версии.

Весь мир ООП держится на трех китах: инкапсуляции, наследовании и полиморфизме. Для начала о них надо иметь только самое общее представление.

Объединение данных и операций в одну сущность — объект — тесно связано с понятием инкапсуляции, которое означает сокрытие внутреннего устройства. Инкапсуляция делает объекты похожими на маленькие программные модули, в которых скрыты внутренние данные и у которых имеется интерфейс использования в виде подпрограмм. Переход от понятий «структура данных» и «алгоритм» к понятию «объект» значительно повысил ясность и надежность программ.

Второй кит ООП — наследование. Этот простой принцип означает, что если вы хотите создать новый класс объектов, который расширяет возможности уже существующего класса, то нет необходимости в переписывании заново всех полей, методов и свойств. Вы объявляете, что новый класс является потомком (или дочерним классом) имеющегося класса объектов, называемого предком (или родительским классом), и добавляете к нему новые поля, методы и свойства. Процесс порождения новых классов на основе других классов называется наследованием. Новые классы объектов имеют как унаследованные признаки, так и, возможно, новые. Очень важно, что в отношениях наследования любой класс может иметь только одного непосредственного предка и сколь угодно много потомков. Поэтому все связанные отношением наследования классы образуют иерархию. Примером иерархии классов является библиотека VCL; с ее помощью в среде Delphi обеспечивается разработка GUI-приложений.

Концепция инкапсуляции весьма проста: нужно просто думать о классе как о "чёрном ящике" с очень маленькой видимой частью. Видимая часть, которая называется интерфейсом класса, позволяет остальным частям программы осуществлять доступ к объектам этого класса и использовать их. Однако, когда вы работаете с объектами, большая часть их кода скрыта от вас. Вы, как правило, не знаете, как устроены внутренние данные объекта, и обычно у вас нет возможности прямого доступа к ним. При этом предполагается, что для доступа к данным вы будете использовать методы. Это объектно-ориентированный подход к классической программной концепции, называемой скрытием информации.

Существует 4 спецификатора доступа: private, protected, public и published:

1. Ключевое слово private определяет поля и методы класса, которые недоступны вне модуля (файла с исходным кодом), в котором определён класс. Секция позволяет скрыть те поля и методы, которые относятся к так называемым особенностям реализации.

2. Ключевое слово public определяет поля и методы класса, к которым может обращаться любая часть кода программы (включая модуль, в котором определён класс). Всё, что помещается в секцию, служит для манипуляций с объектами и составляет программный интерфейс класса.

3. Ключевое слово protected определяет частично доступные поля и методы. Доступ к ним имеют только методы данного класса и всех производных классов, остальным частям программы они не видны. Директива позволяет скрыть особенности реализации класса, но в отличие от private разрешает другим программистам порождать новые классы и обращаться к полям, методам и свойствам, которые составляют так называемый интерфейс разработчика. В эту секцию обычно помещаются виртуальные методы.

4. Ключевое слово published устанавливает правила видимости те же, что и директива public. Особенность состоит в том, что для элементов, помещённых в секцию published, компилятор генерирует информацию о типах этих элементов. Эта информация доступна во время выполнения программы, что позволяет превращать объекты в компоненты визуальной среды разработки. Секцию published разрешено использовать только тогда, когда для самого класса или его предка включена директива компилятора.

3. Формализованное описание разработанных классов

Класс NewsSet базовый для всех остальных реализованных. Содержит в себе информацию о дате публикации и названии заголовка статьи.

**using** System**;**

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

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

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

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

**namespace** \_3rdstep

**{**

**internal** class NewsSet

**{**

**private** string date**;** //дата публикации

**private** string title**;** //заголовок новости

**public** NewsSet**(**string ntitle**,** string ndate**)** //конструктор

**{**

date **=** ndate**;**

title **=** ntitle**;**

**}**

**public** string Date

**{**

**get** **=>** date**;** **set** **=>** date **=** **value;**

**}**//get- и set- для строки "дата публикации"

**public** string Title

**{**

**get** **=>** title**;** **set** **=>** title **=** **value;**

**}**//get- и set- для строки "заголовок новости"

**}**

**}**

Описанный выше класс используется в ThemeSection, который реализует кольцевую очередь на основе массива, для хранения всех публикаций в колонке статей. Из данного класса доступна работа с методами NewsSet.

**using** System**;**

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

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

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

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

**namespace** \_3rdstep

**{**

**internal** class ThemeSection

**{**

**private** NewsSet**[]** queue**;** //очередь для хранения новостей

**private** int count**;** //количество новостей

**private** int first**,** last**;** //ссылки на первый и последний элементы

**private** int next**;** // ссылка на следующий раздел

**private** string namechapter**;** //переменная имя раздела

**public** ThemeSection**(**string \_name**)** //конструктор

**{**

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

queue **=** **new** NewsSet**[**10**];**

namechapter **=** \_name**;**

**}**

**public** string NameChapter

**{**

**get** **{** **return** namechapter**;** **}**

**set** **{** namechapter **=** **value;** **}**

**}**//get- и set- для строки "название раздела"

**public** int Count

**{**

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

**}**//get- для счетчика количества элементов

**public** NewsSet**[]** Queue

**{**

**get**

**{**

**return** queue**;**

**}**

**}**//get- для получения всего массива очереди для сохранения и загрузки файла

**public** void AddToList**(**string \_title**,** string \_date**)** //функция добавления

**{**

NewsSet First **=** **new** NewsSet**(**\_title**,** \_date**);**

**if** **(**count **<** 10**)**

**{**

**if** **(**last **>=** 10 **&&** first **>** 0**)** last **=** 0**;** //закольцовывание last

queue**[**last**]** **=** First**;**

count**++;**

last**++;**

**}**

**}**

**public** void DeleteFromQueue**()** //функция удаления

**{**

**if** **(**count **!=** 0**)**

**{**

**if** **(**first **>** 10**)** first **=** 0**;** //закольцовывание first

queue**[**first**]** **=** **null;**

count**--;**

first**++;**

**}**

**}**

**public** NewsSet Search**(**string \_title**)** //функция поиска в списке

**{**

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

**{**

**if** **(**queue**[**i**].**Title**.**Equals**(**\_title**))**

**{**

**return** queue**[**i**];**

**}**

**}**

**return** **null;**

**}**

**public** int GetSumm**()** //функция подсчета суммы всех новостей

**{**

**return** count**;**

**}**

**public** string GetData**()** //инфо

**{**

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

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

**{**

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

**{**

info **+=** "\n" **+** queue**[**i**].**Title**;**

info **+=** "\n" **+** queue**[**i**].**Date**;**

**}**

**}**

**return** info**;**

**}**

**}**

**}**

Класс NewsPortal находится на вершине в иерархии всех классов. В классе идёт работа с адресным замкнутым (кольцевым) упорядоченным однонаправленным списком без заголовка для управления всеми новостными колонками.

**using** System**;**

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

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

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

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

**namespace** \_3rdstep

**{**

**internal** class NewsPortal

**{**

**private** string namePortal**;**//имя новостного портала

**private** List pfirst**,** plast**,** ptemp**;**//переменные-ссылки на обьекты в структуре список(ссылка на первый, последний элементы)

**private** int count**;**//счетчик количества элементов в списке

**public** NewsPortal**(**string \_namePortal**)** //конструктор

**{**

**this.**namePortal **=** \_namePortal**;**

pfirst **=** **null;**

plast **=** **null;**

ptemp **=** **null;**

count **=** 0**;**

**}**

**public** int Count

**{**

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

**}**//метод get, для получения количества элементов

**public** string NamePortal

**{**

**get** **{** **return** namePortal**;** **}**

**}**//метод get, для имени новостного портала

**public** void addTheme**(**String \_nameTheme**)** //функция добавления раздела

**{**

ThemeSection newThemeSection **=** **new** ThemeSection**(**\_nameTheme**);**

List helper **=** **new** List**(**newThemeSection**);**

count**++;**

**if** **(**pfirst **==** **null)**

**{**

pfirst **=** helper**;**

plast **=** helper**;**

plast**.**Next **=** pfirst**;**

**}**

**else** **if** **(**\_nameTheme**[**0**]** **<** pfirst**.**data**.**NameChapter**[**0**])**

**{**

ptemp **=** pfirst**;**

pfirst **=** helper**;**

pfirst**.**Next **=** ptemp**;**

**}**

**else** **if** **(**\_nameTheme**[**0**]** **>=** plast**.**data**.**NameChapter**[**0**])**

**{**

ptemp **=** plast**;**

plast **=** helper**;**

ptemp**.**Next **=** plast**;**

plast**.**Next **=** pfirst**;**

**}**

**else**

**{**

ptemp **=** pfirst**;**

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

**{**

**if** **((**\_nameTheme**[**0**]** **>** ptemp**.**data**.**NameChapter**[**0**])** **&&** **(**\_nameTheme**[**0**]** **<** ptemp**.**Next**.**data**.**NameChapter**[**0**]))**

**{**

helper**.**Next **=** ptemp**.**Next**;**

ptemp**.**Next **=** helper**;**

**break;**

**}**

**else**

**{**

ptemp **=** ptemp**.**Next**;**

**}**

**}**

**}**

**}**

**public** ThemeSection FindTheme**(**String nameTheme**)** //функция поиска раздела по его названию

**{**

ptemp **=** pfirst**;**

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

**{**

**if** **(**ptemp**.**data**.**NameChapter**.**Equals**(**nameTheme**))**

**{**

**return** ptemp**.**data**;**

**}**

**else** ptemp **=** ptemp**.**Next**;**

**}**

**return** **null;**

**}**

**public** int FindThemeForInt**(**String nameTheme**)** //функция получения индекса от объекта

**{**

ptemp **=** pfirst**;**

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

**{**

**if** **(**ptemp**.**data**.**NameChapter**.**Equals**(**nameTheme**))**

**{**

**return** i**;**

**}**

**else** ptemp **=** ptemp**.**Next**;**

**}**

**return** 0**;**

**}**

**public** void delTheme**(**String nameTheme**)** //функция удаления раздела

**{**

ptemp **=** pfirst**;**

**if** **(**ptemp**.**data**.**NameChapter**.**Equals**(**nameTheme**))**

**{**

pfirst **=** pfirst**.**Next**;**

plast**.**Next **=** pfirst**;**

count**--;**

**}**

**else**

**{**

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

**{**

**if** **(**ptemp**.**Next**.**data**.**NameChapter**.**Equals**(**nameTheme**))**

**{**

ptemp**.**Next **=** ptemp**.**Next**.**Next**;**

count**--;**

**}**

**else** ptemp **=** ptemp**.**Next**;**

**}**

**}**

**}**

**public** void delThemeAll**()**

**{**

pfirst **=** **null;**

plast **=** **null;**

**}**//удаление всего списка для функции загрузки файла в программу

**public** string getQueue**()** //получение информации обо всех разделах, а также информации о новостях

**{**

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

ptemp **=** pfirst**;**

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

**{**

info **+=** "\n" **+** ptemp**.**data**.**NameChapter**;**

ptemp **=** ptemp**.**Next**;**

**}**

**return** info**;**

**}**

**public** string getQueueFor**()**

**{**

string info **=** ""**;**

ptemp **=** pfirst**;**

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

**{**

info **+=** "\n" **+** ptemp**.**data**.**GetData**();**

ptemp **=** ptemp**.**Next**;**

**}**

**return** info**;**

**}**//получение информации о всем новостном портале для функции сохранения

**public** int getSummQueue**()** //функция получения суммы всех новостей

**{**

int summ **=** 0**;**

ptemp **=** pfirst**;**

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

**{**

summ **+=** ptemp**.**data**.**GetSumm**();**

ptemp **=** ptemp**.**Next**;**

**}**

**return** summ**;**

**}**

**}**

**}**

Работа в классе NewsPortal происходит благодаря адресному списку, который реализован в классе List хранящий в себе экземпляры класса ThemeSection.

**using** System**;**

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

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

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

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

**namespace** \_3rdstep

**{**

**internal** class List //класс List для реализации адресного списка

**{**

**public** ThemeSection data**;** //переменная для записи полезных данных

**private** List next**;** //переменная-ссылка на следующий элемент в списке

**public** List**(**ThemeSection \_data**)** //конструктор, где входные параметры - объект класса ThemeSection

**{**

**this.**data **=** \_data**;**

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

**}**

**public** List Next //сеттер и геттер для переменной Next

**{**

**get** **=>** next**;** **set** **=>** next **=** **value;**

**}**

**}**

**}**

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

При запуске программы задаётся дизайн таблицы, создаётся экземпляр класса контейнера с передаваемым параметром названия компании **"Country News"**

**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** \_3rdstep

**{**

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

**{**

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

**{**

InitializeComponent**();**

dataGridView1**.**ColumnHeadersDefaultCellStyle**.**Font **=** **new** Font**(**"Syne"**,** 10**,** FontStyle**.**Bold**);**

// настройка шрифта у 1 таблицы

dataGridView2**.**ColumnHeadersDefaultCellStyle**.**Font **=** **new** Font**(**"Syne"**,** 10**,** FontStyle**.**Bold**);**

// настройка шрифта у 2 таблицы

**}**

NewsPortal Container **=** **new** NewsPortal**(**"Country News"**);**//создание контейнера новостного раздела

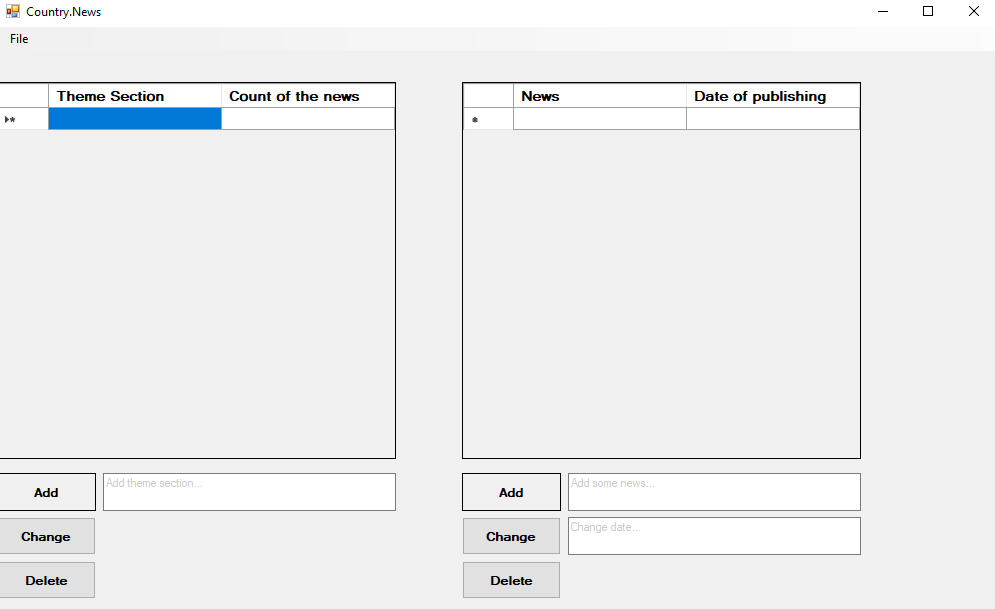


Рис 8. Запуск программы

Данный метод отвечает за вывод данных из новостной колонки. При выборе колонки вторая таблица очищается и отображаются соответствующие статьи.

**private** void dataGridView1\_CellContentClick**(object** sender**,** DataGridViewCellEventArgs e**)**//обновление второй таблицы при выборе элемента в первой

**{**

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null)**//проверка на "случайный" клик

**{**

dataGridView2**.**Rows**.**Clear**();**//очистка полностью второй таблицы

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**//запись в строку выбранного пользователем названия раздела

ThemeSection vspom **=** Container**.**FindTheme**(**text**);**//поиск выбранного пользователем раздела

**if** **(**vspom **!=** **null)**//проверка на пустое нахождение объекта

**{**

NewsSet**[]** copy1 **=** vspom**.**Queue**;**//копирование массива-очереди в вспомогательный массив

int chet **=** Container**.**FindTheme**(**text**).**Count**;**//получение количества новостей в найденном разделе

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

**{**

dataGridView2**.**Rows**.**Add**(**copy1**[**i**].**Title**,** copy1**[**i**].**Date**);**//с помощью цикла добавление

//во вторую таблицу обьектов(новостей) данного раздела

**}**

dataGridView1**.**Rows**[**dataGridView1**.**CurrentRow**.**Index**].**Cells**[**1**].**Value **=** Container**.**FindTheme**(**text**).**Count**;**

//обновление количества элементов у n-го раздела

**}**

**}**

**}**

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

**{**

**}**

Очищение данного текст бокса, если изначально там был текст

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

**{**

TextTheme**.**Clear**();**//очищение данного текст бокса, так как изначально там был текст

//для ориентации ввода клиента

**}**

Установка валидаторов на TextBox

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

**{**//обработчик текста у текстового поля, отвечающего за название раздела

//принимает максимум 20 символов и только буквы Руссого и Английского алфавитов обоих литеров

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

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

**{**

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

**}**

TextTheme**.**MaxLength **=** 20**;**

**}**

**private** void TextNews2\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**//обработчик длины текста в текстовом поле, отвечающего за дату публикации новости

TextNews2**.**MaxLength **=** 20**;**

**}**

**private** void TextNews1\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**//обработчик текста у текстового поля, отвечающего за название новости

//принимает максимум 40 символов и только буквы Руссого и Английского алфавитов обоих литеров

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

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

**{**

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

**}**

TextNews1**.**MaxLength **=** 40**;**

**}**

**private** void TextNews1\_Click**(object** sender**,** EventArgs e**)**

**{**

TextNews1**.**Clear**();**//очищение данного текст бокса, так как изначально там был текст

//для ориентации ввода клиента

**}**

**private** void TextNews2\_Click**(object** sender**,** EventArgs e**)**

**{**

TextNews2**.**Clear**();**//очищение данного текст бокса, так как изначально там был текст

//для ориентации ввода клиента

**}**

**private** void dataGridView2\_CellContentClick**(object** sender**,** DataGridViewCellEventArgs e**)**

**{**

**}**

Метод добавления новостного блока в новостной портал, данные берутся из TextBox и заносятся в таблицу, и создаётся экземпляр класса с новостной колонкой.

**private** void AddTheme\_Click**(object** sender**,** EventArgs e**)**//добавление новостного раздела

**{**

Container**.**addTheme**(**TextTheme**.**Text**);**//добавление раздела именно в структуру

int i **=** Container**.**FindThemeForInt**(**TextTheme**.**Text**);** //при вызове метода поиска возвращает i

//для индекса добавления во 1 табличку

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

**{**

dataGridView1**.**Rows**.**Add**();**//добавление в 1 таблицу пустой строки

**if** **(**dataGridView1**.**Rows**[**i**].**Cells**[**0**].**Value **==** **null)**//проверка на первое добавление в строку по индексу i

**{**

dataGridView1**.**Rows**[**i**].**Cells**[**0**].**Value **=** TextTheme**.**Text**;**//присваивание в первый столбец данной

//строки значение текстового поля(название раздела)

dataGridView1**.**Rows**[**i**].**Cells**[**1**].**Value **=** Container**.**FindTheme**(**TextTheme**.**Text**).**Count**;**//присваивание в первый столбец данной

//(количество элементов в этом разделе)

TextTheme**.**Clear**();**//очищение текстбокса для дальнейшего удобства

**}**

**else**

**{**

**for** **(**int j **=** Container**.**Count**;** j **>** i**;** j**--)**//с помощью цикла for смещение строк вниз, для

//добавления объекта в нужную строку

**{**

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

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

dataGridView1**.**Rows**[**j **-** 1**].**Cells**[**0**].**Value **=** **null;**

dataGridView1**.**Rows**[**j **-** 1**].**Cells**[**1**].**Value **=** **null;**

**}**

//добавление самой строки по индексу i

dataGridView1**.**Rows**[**i**].**Cells**[**0**].**Value **=** TextTheme**.**Text**;**

dataGridView1**.**Rows**[**i**].**Cells**[**1**].**Value **=** Container**.**FindTheme**(**TextTheme**.**Text**).**Count**;**

**}**

**}**

**else** MessageBox**.**Show**(**"Adding is not successful"**);**//если что то не введено или не выбран

//нужный элемент в 1 таблице, показ пользователю сообщения

**}**

Метод изменения новостной колонки, из TextBox данные попадаются в таблицу и соответствующий экземпляр.

**private** void ChTheme\_Click**(object** sender**,** EventArgs e**)**//метод изменения выбранного объекта

**{**

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null** **&&** TextTheme**.**Text **!=** ""**)**//проверка

//на правильность вхождения данных

**{**

string change **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**//запись в строку выбранной строки

int index **=** dataGridView1**.**CurrentRow**.**Index**;**//получение индекса выбранной строки

Container**.**FindTheme**(**change**).**NameChapter **=** TextTheme**.**Text**;**//изменение данных в самой структуре

dataGridView1**.**Rows**[**index**].**Cells**[**0**].**Value **=** TextTheme**.**Text**;**//изменение данных в выбранной строке

TextTheme**.**Clear**();**//очищение текстбокса

**}**

**else** **{** MessageBox**.**Show**(**"The change failure"**);** **}**//вывод сообщени пользователю в случае недостающих данных

**}**

**private** void DelTheme\_Click**(object** sender**,** EventArgs e**)**//удаление новостного раздела

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**//запись выбранной строки в таблице

int index **=** dataGridView1**.**CurrentRow**.**Index**;**//запись индекса выбранной строки

Container**.**delTheme**(**text**);**//удаление элемента из самой структуры

dataGridView1**.**Rows**.**RemoveAt**(**index**);**//удаление выбранной строки из 1 таблицы

dataGridView2**.**Rows**.**Clear**();**//для правильности данных очищение всей 2 таблицы

//далее при нажатии на нужную строку выводится актуальная информация

**}**

**private** void Form1\_Load**(object** sender**,** EventArgs e**)**

**{**

**}**

Метод добавления статьи и в колонку.

**private** void AddNews\_Click**(object** sender**,** EventArgs e**)**//добавление новости в раздел

**{**

//обрабатываем вхождение нужных элементов

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null** **&&** TextNews1**.**Text **!=** "" **&&** TextNews2**.**Text **!=** ""**)**

**{**

//добавление новости в саму структуру

Container**.**FindTheme**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**AddToList**(**TextNews1**.**Text**,** TextNews2**.**Text**);**

dataGridView2**.**Rows**.**Add**(**TextNews1**.**Text**,** TextNews2**.**Text**);**//добавление новости в таблицу 2

//очищение текстбоксов

TextNews1**.**Clear**();**

TextNews2**.**Clear**();**

**}**

**else** **{** MessageBox**.**Show**(**"Adding news is not successful"**);** **}**//вывод сообщение пользователю

//в случае неправильности входных данных

**}**

У выбранной статьи во второй колонки изменяются данные, а также изменяется соответствующий элемент в адресном списке.

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

**{**

//обрабатываем вхождение нужных элементов

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null** **&&** dataGridView2**.**CurrentCell**.**Value **!=** **null** **&&** TextNews1**.**Text **!=** "" **&&** TextNews2**.**Text **!=** ""**)**

**{**

string change **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**//запись выбранной строки в таблице

string changeNews **=** dataGridView2**.**CurrentCell**.**Value**.**ToString**();**//запись выбранной строки в таблице

int index **=** dataGridView2**.**CurrentRow**.**Index**;**//запись индекса выбранной строки в таблице

Container**.**FindTheme**(**change**).**Search**(**changeNews**).**Title **=** TextNews1**.**Text**;**//изменение данных в самой структуре

Container**.**FindTheme**(**change**).**Search**(**TextNews1**.**Text**).**Date **=** TextNews2**.**Text**;**//изменение данных в самой структуре

dataGridView2**.**Rows**[**index**].**Cells**[**0**].**Value **=** TextNews1**.**Text**;**//изменение данных во 2 таблице

dataGridView2**.**Rows**[**index**].**Cells**[**1**].**Value **=** TextNews2**.**Text**;**//изменение данных во 2 таблице

//очищение текстбоксов

TextNews1**.**Clear**();**

TextNews2**.**Clear**();**

**}**

**else** **{** MessageBox**.**Show**(**"The change news failure"**);** **}**//вывод сообщение пользователю

//в случае неправильности входных данных

**}**

**private** void DelNews\_Click**(object** sender**,** EventArgs e**)**//удаление новости в разделе

**{**

string search **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**//запись выбранной строки в таблице

Container**.**FindTheme**(**search**).**DeleteFromQueue**();**//удаление новости из выбранного раздела в структуре

dataGridView2**.**Rows**.**RemoveAt**(**0**);**//удаление новости во 2 таблице

**}**

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

**private** void saveToolStripMenuItem\_Click**(object** sender**,** EventArgs e**)**//сохранение файла

**{**

string path**;**

string fileText **=** "Файл\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**()** **==** DialogResult**.**OK**)**

**{**

path **=** saveFile**.**FileName**;**//имя файла

fileText **+=** Container**.**getQueue**()** **+** "\n"**;**//получение названий разделов, и общее их количество

fileText **+=** Container**.**getQueueFor**();**//получение количества новостей и данных о них в каждом разделе

//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 uploadToolStripMenuItem\_Click**(object** sender**,** EventArgs e**)**//загрузка файла

**{**

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

int Count**,** LCount**;**

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

**{**

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

**{**

path **=** openFile**.**FileName**;**//получение имени выбранного файла

**}**

**}**

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

**{**

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

//Проверка на пригодность файла для работы с программой

**if** **(**doIt **==** "Файл"**)**

**{**

Container**.**delThemeAll**();**//удаление всех имеющихся данных в запущенной программе

dataGridView1**.**Rows**.**Clear**();**//удаление всех строк в 1 и 2 таблицах

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

Count **=** int**.**Parse**(**stream3**.**ReadLine**());**//количество разделов

**for** **(**int i **=** 1**;** i **<=** Count**;)**//цикл for по количеству разделов для их добавления в структуру и в 1 таблицу

**{**

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

Container**.**addTheme**(**name**);**

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

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

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

i**++;**

**else**

**break;**

**}**

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

**while** **(**stream3**.**Peek**()** **!=** **-**1**)**//цикл while для добавление в структуру новостей в каждый раздел

**{**

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

**if** **(**LCount **!=** 0**)**

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

**{**

string \_NewsName **=** stream3**.**ReadLine**();**

string \_DatePub **=** stream3**.**ReadLine**();**

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

Container**.**FindTheme**(**name**).**AddToList**(**\_NewsName**,** \_DatePub**);**

**if** **(**i **!=** LCount**)**

i**++;**

**else**

**break;**

**}**

hall**++;**

**}**

**}**

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

//в случае, когда первая строка не содержит фразу "Файл"

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

**}**

**}**

**}**

**}**

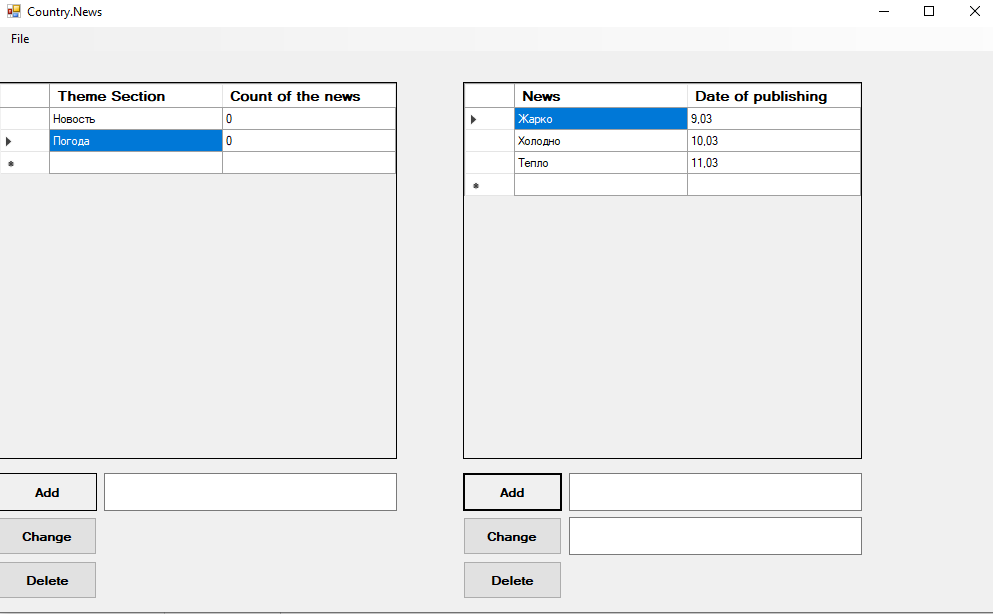


Рис.9 Работа программы.

1. Описание структуры проекта в соответствии с использованным инструментом разработки

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

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

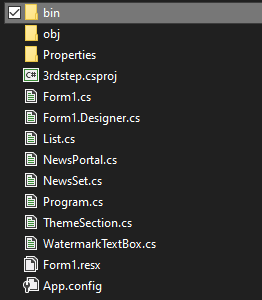


Рис. 10 Содержимое папки проекта.

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.

Программа была разработана в среде разработки Visual Studio 2022. Это универсальная среда разработке позволяющая разрабатывать программы на языке программирования C# , а так же С++ и Python. IDE позволяет созывать комплексные программы, а интуитивно понятный интерфейс позволяет легко разобраться в программе и концентрироваться только на разработке.

Помимо удобного интерфейса программа позволяет тестировать программу на ресурсоёмкость, показывая программисту потребление программой ресурсов оперативной памяти и процессора.

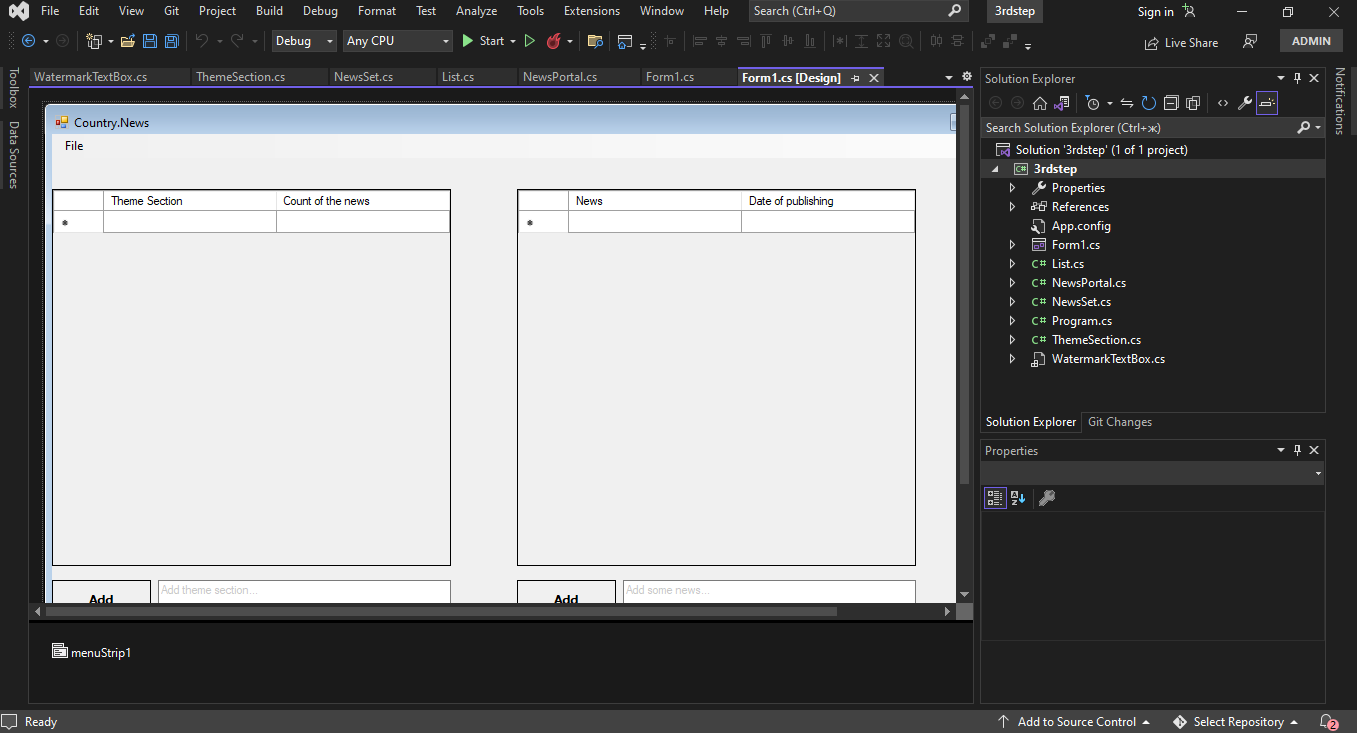


Рис. 11 Работа Visual Studio 2022

# **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**.**Linq**;**

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

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

**namespace** \_3rdstep

**{**

**internal** class NewsPortal

**{**

**private** string namePortal**;**//имя новостного портала

**private** List pfirst**,** plast**,** ptemp**;**//переменные-ссылки на обьекты в структуре список(ссылка на первый, последний элементы)

**private** int count**;**//счетчик количества элементов в списке

**public** NewsPortal**(**string \_namePortal**)** //конструктор

**{**

**this.**namePortal **=** \_namePortal**;**

pfirst **=** **null;**

plast **=** **null;**

ptemp **=** **null;**

count **=** 0**;**

**}**

**public** int Count

**{**

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

**}**//метод get, для получения количества элементов

**public** string NamePortal

**{**

**get** **{** **return** namePortal**;** **}**

**}**//метод get, для имени новостного портала

**public** void addTheme**(**String \_nameTheme**)** //функция добавления раздела

**{**

ThemeSection newThemeSection **=** **new** ThemeSection**(**\_nameTheme**);**

List helper **=** **new** List**(**newThemeSection**);**

count**++;**

**if** **(**pfirst **==** **null)**

**{**

pfirst **=** helper**;**

plast **=** helper**;**

plast**.**Next **=** pfirst**;**

**}**

**else** **if** **(**\_nameTheme**[**0**]** **<** pfirst**.**data**.**NameChapter**[**0**])**

**{**

ptemp **=** pfirst**;**

pfirst **=** helper**;**

pfirst**.**Next **=** ptemp**;**

**}**

**else** **if** **(**\_nameTheme**[**0**]** **>=** plast**.**data**.**NameChapter**[**0**])**

**{**

ptemp **=** plast**;**

plast **=** helper**;**

ptemp**.**Next **=** plast**;**

plast**.**Next **=** pfirst**;**

**}**

**else**

**{**

ptemp **=** pfirst**;**

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

**{**

**if** **((**\_nameTheme**[**0**]** **>** ptemp**.**data**.**NameChapter**[**0**])** **&&** **(**\_nameTheme**[**0**]** **<** ptemp**.**Next**.**data**.**NameChapter**[**0**]))**

**{**

helper**.**Next **=** ptemp**.**Next**;**

ptemp**.**Next **=** helper**;**

**break;**

**}**

**else**

**{**

ptemp **=** ptemp**.**Next**;**

**}**

**}**

**}**

**}**

**public** ThemeSection FindTheme**(**String nameTheme**)** //функция поиска раздела по его названию

**{**

ptemp **=** pfirst**;**

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

**{**

**if** **(**ptemp**.**data**.**NameChapter**.**Equals**(**nameTheme**))**

**{**

**return** ptemp**.**data**;**

**}**

**else** ptemp **=** ptemp**.**Next**;**

**}**

**return** **null;**

**}**

**public** int FindThemeForInt**(**String nameTheme**)** //функция получения индекса от объекта

**{**

ptemp **=** pfirst**;**

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

**{**

**if** **(**ptemp**.**data**.**NameChapter**.**Equals**(**nameTheme**))**

**{**

**return** i**;**

**}**

**else** ptemp **=** ptemp**.**Next**;**

**}**

**return** 0**;**

**}**

**public** void delTheme**(**String nameTheme**)** //функция удаления раздела

**{**

ptemp **=** pfirst**;**

**if** **(**ptemp**.**data**.**NameChapter**.**Equals**(**nameTheme**))**

**{**

pfirst **=** pfirst**.**Next**;**

plast**.**Next **=** pfirst**;**

count**--;**

**}**

**else**

**{**

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

**{**

**if** **(**ptemp**.**Next**.**data**.**NameChapter**.**Equals**(**nameTheme**))**

**{**

ptemp**.**Next **=** ptemp**.**Next**.**Next**;**

count**--;**

**}**

**else** ptemp **=** ptemp**.**Next**;**

**}**

**}**

**}**

**public** void delThemeAll**()**

**{**

pfirst **=** **null;**

plast **=** **null;**

**}**//удаление всего списка для функции загрузки файла в программу

**public** string getQueue**()** //получение информации обо всех разделах, а также информации о новостях

**{**

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

ptemp **=** pfirst**;**

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

**{**

info **+=** "\n" **+** ptemp**.**data**.**NameChapter**;**

ptemp **=** ptemp**.**Next**;**

**}**

**return** info**;**

**}**

**public** string getQueueFor**()**

**{**

string info **=** ""**;**

ptemp **=** pfirst**;**

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

**{**

info **+=** "\n" **+** ptemp**.**data**.**GetData**();**

ptemp **=** ptemp**.**Next**;**

**}**

**return** info**;**

**}**//получение информации о всем новостном портале для функции сохранения

**public** int getSummQueue**()** //функция получения суммы всех новостей

**{**

int summ **=** 0**;**

ptemp **=** pfirst**;**

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

**{**

summ **+=** ptemp**.**data**.**GetSumm**();**

ptemp **=** ptemp**.**Next**;**

**}**

**return** summ**;**

**}**

**}**

**}**

**using** System**;**

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

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

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

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

**namespace** \_3rdstep

**{**

**internal** class List //класс List для реализации адресного списка

**{**

**public** ThemeSection data**;** //переменная для записи полезных данных

**private** List next**;** //переменная-ссылка на следующий элемент в списке

**public** List**(**ThemeSection \_data**)** //конструктор, где входные параметры - объект класса ThemeSection

**{**

**this.**data **=** \_data**;**

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

**}**

**public** List Next //сеттер и геттер для переменной Next

**{**

**get** **=>** next**;** **set** **=>** next **=** **value;**

**}**

**}**

**}**

**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** \_3rdstep

**{**

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

**{**

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

**{**

InitializeComponent**();**

dataGridView1**.**ColumnHeadersDefaultCellStyle**.**Font **=** **new** Font**(**"Syne"**,** 10**,** FontStyle**.**Bold**);**

// настройка шрифта у 1 таблицы

dataGridView2**.**ColumnHeadersDefaultCellStyle**.**Font **=** **new** Font**(**"Syne"**,** 10**,** FontStyle**.**Bold**);**

// настройка шрифта у 2 таблицы

**}**

NewsPortal Container **=** **new** NewsPortal**(**"Country News"**);**//создание контейнера новостного раздела

**private** void dataGridView1\_CellContentClick**(object** sender**,** DataGridViewCellEventArgs e**)**//обновление второй таблицы при выборе элемента в первой

**{**

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null)**//проверка на "случайный" клик

**{**

dataGridView2**.**Rows**.**Clear**();**//очистка полностью второй таблицы

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**//запись в строку выбранного пользователем названия раздела

ThemeSection vspom **=** Container**.**FindTheme**(**text**);**//поиск выбранного пользователем раздела

**if** **(**vspom **!=** **null)**//проверка на пустое нахождение объекта

**{**

NewsSet**[]** copy1 **=** vspom**.**Queue**;**//копирование массива-очереди в вспомогательный массив

int chet **=** Container**.**FindTheme**(**text**).**Count**;**//получение количества новостей в найденном разделе

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

**{**

dataGridView2**.**Rows**.**Add**(**copy1**[**i**].**Title**,** copy1**[**i**].**Date**);**//с помощью цикла добавление

//во вторую таблицу обьектов(новостей) данного раздела

**}**

dataGridView1**.**Rows**[**dataGridView1**.**CurrentRow**.**Index**].**Cells**[**1**].**Value **=** Container**.**FindTheme**(**text**).**Count**;**

//обновление количества элементов у n-го раздела

**}**

**}**

**}**

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

**{**

**}**

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

**{**

TextTheme**.**Clear**();**//очищение данного текст бокса, так как изначально там был текст

//для ориентации ввода клиента

**}**

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

**{**//обработчик текста у текстового поля, отвечающего за название раздела

//принимает максимум 20 символов и только буквы Руссого и Английского алфавитов обоих литеров

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

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

**{**

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

**}**

TextTheme**.**MaxLength **=** 20**;**

**}**

**private** void TextNews2\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**//обработчик длины текста в текстовом поле, отвечающего за дату публикации новости

TextNews2**.**MaxLength **=** 20**;**

**}**

**private** void TextNews1\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**//обработчик текста у текстового поля, отвечающего за название новости

//принимает максимум 40 символов и только буквы Руссого и Английского алфавитов обоих литеров

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

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

**{**

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

**}**

TextNews1**.**MaxLength **=** 40**;**

**}**

**private** void TextNews1\_Click**(object** sender**,** EventArgs e**)**

**{**

TextNews1**.**Clear**();**//очищение данного текст бокса, так как изначально там был текст

//для ориентации ввода клиента

**}**

**private** void TextNews2\_Click**(object** sender**,** EventArgs e**)**

**{**

TextNews2**.**Clear**();**//очищение данного текст бокса, так как изначально там был текст

//для ориентации ввода клиента

**}**

**private** void dataGridView2\_CellContentClick**(object** sender**,** DataGridViewCellEventArgs e**)**

**{**

**}**

**private** void AddTheme\_Click**(object** sender**,** EventArgs e**)**//добавление новостного раздела

**{**

Container**.**addTheme**(**TextTheme**.**Text**);**//добавление раздела именно в структуру

int i **=** Container**.**FindThemeForInt**(**TextTheme**.**Text**);** //при вызове метода поиска возвращает i

//для индекса добавления во 1 табличку

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

**{**

dataGridView1**.**Rows**.**Add**();**//добавление в 1 таблицу пустой строки

**if** **(**dataGridView1**.**Rows**[**i**].**Cells**[**0**].**Value **==** **null)**//проверка на первое добавление в строку по индексу i

**{**

dataGridView1**.**Rows**[**i**].**Cells**[**0**].**Value **=** TextTheme**.**Text**;**//присваивание в первый столбец данной

//строки значение текстового поля(название раздела)

dataGridView1**.**Rows**[**i**].**Cells**[**1**].**Value **=** Container**.**FindTheme**(**TextTheme**.**Text**).**Count**;**//присваивание в первый столбец данной

//(количество элементов в этом разделе)

TextTheme**.**Clear**();**//очищение текстбокса для дальнейшего удобства

**}**

**else**

**{**

**for** **(**int j **=** Container**.**Count**;** j **>** i**;** j**--)**//с помощью цикла for смещение строк вниз, для

//добавления объекта в нужную строку

**{**

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

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

dataGridView1**.**Rows**[**j **-** 1**].**Cells**[**0**].**Value **=** **null;**

dataGridView1**.**Rows**[**j **-** 1**].**Cells**[**1**].**Value **=** **null;**

**}**

//добавление самой строки по индексу i

dataGridView1**.**Rows**[**i**].**Cells**[**0**].**Value **=** TextTheme**.**Text**;**

dataGridView1**.**Rows**[**i**].**Cells**[**1**].**Value **=** Container**.**FindTheme**(**TextTheme**.**Text**).**Count**;**

**}**

**}**

**else** MessageBox**.**Show**(**"Adding is not successful"**);**//если что то не введено или не выбран

//нужный элемент в 1 таблице, показ пользователю сообщения

**}**

**private** void ChTheme\_Click**(object** sender**,** EventArgs e**)**//метод изменения выбранного объекта

**{**

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null** **&&** TextTheme**.**Text **!=** ""**)**//проверка

//на правильность вхождения данных

**{**

string change **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**//запись в строку выбранной строки

int index **=** dataGridView1**.**CurrentRow**.**Index**;**//получение индекса выбранной строки

Container**.**FindTheme**(**change**).**NameChapter **=** TextTheme**.**Text**;**//изменение данных в самой структуре

dataGridView1**.**Rows**[**index**].**Cells**[**0**].**Value **=** TextTheme**.**Text**;**//изменение данных в выбранной строке

TextTheme**.**Clear**();**//очищение текстбокса

**}**

**else** **{** MessageBox**.**Show**(**"The change failure"**);** **}**//вывод сообщени пользователю в случае недостающих данных

**}**

**private** void DelTheme\_Click**(object** sender**,** EventArgs e**)**//удаление новостного раздела

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**//запись выбранной строки в таблице

int index **=** dataGridView1**.**CurrentRow**.**Index**;**//запись индекса выбранной строки

Container**.**delTheme**(**text**);**//удаление элемента из самой структуры

dataGridView1**.**Rows**.**RemoveAt**(**index**);**//удаление выбранной строки из 1 таблицы

dataGridView2**.**Rows**.**Clear**();**//для правильности данных очищение всей 2 таблицы

//далее при нажатии на нужную строку выводится актуальная информация

**}**

**private** void Form1\_Load**(object** sender**,** EventArgs e**)**

**{**

**}**

**private** void AddNews\_Click**(object** sender**,** EventArgs e**)**//добавление новости в раздел

**{**

//обрабатываем вхождение нужных элементов

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null** **&&** TextNews1**.**Text **!=** "" **&&** TextNews2**.**Text **!=** ""**)**

**{**

//добавление новости в саму структуру

Container**.**FindTheme**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**AddToList**(**TextNews1**.**Text**,** TextNews2**.**Text**);**

dataGridView2**.**Rows**.**Add**(**TextNews1**.**Text**,** TextNews2**.**Text**);**//добавление новости в таблицу 2

//очищение текстбоксов

TextNews1**.**Clear**();**

TextNews2**.**Clear**();**

**}**

**else** **{** MessageBox**.**Show**(**"Adding news is not successful"**);** **}**//вывод сообщение пользователю

//в случае неправильности входных данных

**}**

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

**{**

//обрабатываем вхождение нужных элементов

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null** **&&** dataGridView2**.**CurrentCell**.**Value **!=** **null** **&&** TextNews1**.**Text **!=** "" **&&** TextNews2**.**Text **!=** ""**)**

**{**

string change **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**//запись выбранной строки в таблице

string changeNews **=** dataGridView2**.**CurrentCell**.**Value**.**ToString**();**//запись выбранной строки в таблице

int index **=** dataGridView2**.**CurrentRow**.**Index**;**//запись индекса выбранной строки в таблице

Container**.**FindTheme**(**change**).**Search**(**changeNews**).**Title **=** TextNews1**.**Text**;**//изменение данных в самой структуре

Container**.**FindTheme**(**change**).**Search**(**TextNews1**.**Text**).**Date **=** TextNews2**.**Text**;**//изменение данных в самой структуре

dataGridView2**.**Rows**[**index**].**Cells**[**0**].**Value **=** TextNews1**.**Text**;**//изменение данных во 2 таблице

dataGridView2**.**Rows**[**index**].**Cells**[**1**].**Value **=** TextNews2**.**Text**;**//изменение данных во 2 таблице

//очищение текстбоксов

TextNews1**.**Clear**();**

TextNews2**.**Clear**();**

**}**

**else** **{** MessageBox**.**Show**(**"The change news failure"**);** **}**//вывод сообщение пользователю

//в случае неправильности входных данных

**}**

**private** void DelNews\_Click**(object** sender**,** EventArgs e**)**//удаление новости в разделе

**{**

string search **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**//запись выбранной строки в таблице

Container**.**FindTheme**(**search**).**DeleteFromQueue**();**//удаление новости из выбранного раздела в структуре

dataGridView2**.**Rows**.**RemoveAt**(**0**);**//удаление новости во 2 таблице

**}**

**private** void saveToolStripMenuItem\_Click**(object** sender**,** EventArgs e**)**//сохранение файла

**{**

string path**;**

string fileText **=** "Файл\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**()** **==** DialogResult**.**OK**)**

**{**

path **=** saveFile**.**FileName**;**//имя файла

fileText **+=** Container**.**getQueue**()** **+** "\n"**;**//получение названий разделов, и общее их количество

fileText **+=** Container**.**getQueueFor**();**//получение количества новостей и данных о них в каждом разделе

//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 uploadToolStripMenuItem\_Click**(object** sender**,** EventArgs e**)**//загрузка файла

**{**

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

int Count**,** LCount**;**

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

**{**

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

**{**

path **=** openFile**.**FileName**;**//получение имени выбранного файла

**}**

**}**

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

**{**

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

//Проверка на пригодность файла для работы с программой

**if** **(**doIt **==** "Файл"**)**

**{**

Container**.**delThemeAll**();**//удаление всех имеющихся данных в запущенной программе

dataGridView1**.**Rows**.**Clear**();**//удаление всех строк в 1 и 2 таблицах

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

Count **=** int**.**Parse**(**stream3**.**ReadLine**());**//количество разделов

**for** **(**int i **=** 1**;** i **<=** Count**;)**//цикл for по количеству разделов для их добавления в структуру и в 1 таблицу

**{**

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

Container**.**addTheme**(**name**);**

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

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

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

i**++;**

**else**

**break;**

**}**

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

**while** **(**stream3**.**Peek**()** **!=** **-**1**)**//цикл while для добавление в структуру новостей в каждый раздел

**{**

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

**if** **(**LCount **!=** 0**)**

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

**{**

string \_NewsName **=** stream3**.**ReadLine**();**

string \_DatePub **=** stream3**.**ReadLine**();**

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

Container**.**FindTheme**(**name**).**AddToList**(**\_NewsName**,** \_DatePub**);**

**if** **(**i **!=** LCount**)**

i**++;**

**else**

**break;**

**}**

hall**++;**

**}**

**}**

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

//в случае, когда первая строка не содержит фразу "Файл"

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

**}**

**}**

**}**

**}**

**using** System**;**

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

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

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

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

**namespace** \_3rdstep

**{**

**internal** class ThemeSection

**{**

**private** NewsSet**[]** queue**;** //очередь для хранения новостей

**private** int count**;** //количество новостей

**private** int first**,** last**;** //ссылки на первый и последний элементы

**private** int next**;** // ссылка на следующий раздел

**private** string namechapter**;** //переменная имя раздела

**public** ThemeSection**(**string \_name**)** //конструктор

**{**

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

queue **=** **new** NewsSet**[**10**];**

namechapter **=** \_name**;**

**}**

**public** string NameChapter

**{**

**get** **{** **return** namechapter**;** **}**

**set** **{** namechapter **=** **value;** **}**

**}**//get- и set- для строки "название раздела"

**public** int Count

**{**

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

**}**//get- для счетчика количества элементов

**public** NewsSet**[]** Queue

**{**

**get**

**{**

**return** queue**;**

**}**

**}**//get- для получения всего массива очереди для сохранения и загрузки файла

**public** void AddToList**(**string \_title**,** string \_date**)** //функция добавления

**{**

NewsSet First **=** **new** NewsSet**(**\_title**,** \_date**);**

**if** **(**count **<** 10**)**

**{**

**if** **(**last **>=** 10 **&&** first **>** 0**)** last **=** 0**;** //закольцовывание last

queue**[**last**]** **=** First**;**

count**++;**

last**++;**

**}**

**}**

**public** void DeleteFromQueue**()** //функция удаления

**{**

**if** **(**count **!=** 0**)**

**{**

**if** **(**first **>** 10**)** first **=** 0**;** //закольцовывание first

queue**[**first**]** **=** **null;**

count**--;**

first**++;**

**}**

**}**

**public** NewsSet Search**(**string \_title**)** //функция поиска в списке

**{**

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

**{**

**if** **(**queue**[**i**].**Title**.**Equals**(**\_title**))**

**{**

**return** queue**[**i**];**

**}**

**}**

**return** **null;**

**}**

**public** int GetSumm**()** //функция подсчета суммы всех новостей

**{**

**return** count**;**

**}**

**public** string GetData**()** //инфо

**{**

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

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

**{**

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

**{**

info **+=** "\n" **+** queue**[**i**].**Title**;**

info **+=** "\n" **+** queue**[**i**].**Date**;**

**}**

**}**

**return** info**;**

**}**

**}**

**}**

**using** System**;**

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

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

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

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

**namespace** \_3rdstep

**{**

**internal** class NewsSet

**{**

**private** string date**;** //дата публикации

**private** string title**;** //заголовок новости

**public** NewsSet**(**string ntitle**,** string ndate**)** //конструктор

**{**

date **=** ndate**;**

title **=** ntitle**;**

**}**

**public** string Date

**{**

**get** **=>** date**;** **set** **=>** date **=** **value;**

**}**//get- и set- для строки "дата публикации"

**public** string Title

**{**

**get** **=>** title**;** **set** **=>** title **=** **value;**

**}**//get- и set- для строки "заголовок новости"

**}**

**}**