|  |  |
| --- | --- |
| **Изображение выглядит как ночное небо  Автоматически созданное описание** | **автономная некоммерческая образовательная организация**  **высшего образования Центросоюза Российской Федерации**  **«Сибирский университет потребительской кооперации»** |

Кафедра информатики

**ОТЧЕТ ОБ ОЗНАКОМИТЕЛЬНОЙ** **ПРАКТИКЕ**

Место прохождения практики

АНОО ВО Центросоюза РФ СибУПК

Обучающегося 2 курса

Соснин Юрий Сергеевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(Фамилия И.О.)*

группы ПИБ-11

Руководитель практики

доцент кафедры информатики,

канд. пед. наук

***Колдунова Ирина Дмитриевна***

Оценка после защиты \_\_\_\_\_\_\_\_\_\_\_

Дата защиты 30.12.2022

Новосибирск

2022

|  |  |
| --- | --- |
| **Изображение выглядит как ночное небо  Автоматически созданное описание** | **автономная некоммерческая образовательная организация**  **высшего образования Центросоюза Российской Федерации**  **«Сибирский университет потребительской кооперации»** |

Кафедра информатики

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Обучающемуся 2 курса очной формы обучения

направления подготовки 09.03.03 Прикладная информатика

направленность (профиль) Прикладная информатика в информационной сфере

ФИО обучающегося\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПИБО0121026

Соснин Юрий Сергеевич

Шифр

Вид практики: учебная

Тип практики: ознакомительная

Приказ о направлении на практику: № Сз-668-Э от 13 декабря 2022 г.

Срок прохождения практики с «17» декабря 2022г. по «30» декабря 2022г.

Место прохождения практики АНОО ВО Центросоюза РФ СибУПК

Срок сдачи студентом отчета на кафедру30.12.22

1. Перечень подлежащих разработке вопросов и общее направление работы:
2. Постановка задачи, краткое описание основных средств языка, которые предполагается использовать для решения поставленной задачи.
3. Разработка алгоритма решения поставленной задачи
4. Создание интерфейса приложения.
5. Тестирование приложения.

II. Объем отчета по практике \_\_\_\_\_\_\_\_\_\_ страниц (+приложение)

Задание выдано

Руководитель практики от университета Колдунова Ирина Дмитриевна, канд. пед. наук, доцент кафедры информатики СибУПК \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Задание получено

Обучающийся\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**Содержание**

[**Введение** 5](#_Toc90811128)

[**1.** **Теоретические основы разработки оконных приложений** 6](#_Toc90811129)

[**1.1.** **…** 6](#_Toc90811130)

[**1.2.** **…** 6](#_Toc90811131)

[**2.** **Разработка приложения** 7](#_Toc90811132)

[**2.1.** **Разработка сценария и алгоритма решения поставленной задачи** 7](#_Toc90811133)

[**2.2.** **Создание интерфейса приложения** 7](#_Toc90811134)

[**2.3.** **Тестирование приложения** 7](#_Toc90811135)

[**Заключение** 8](#_Toc90811136)

[**Библиографический список** 9](#_Toc90811137)

[**Приложения** 10](#_Toc90811138)

# **Введение**

Ознакомительная практика является неотъемлемой частью образовательного процесса. Данный вид практики направлен на закрепление, улучшение и углубление знаний, полученных в процессе обучения.

Актуальность выбранной темы для научно-исследовательской работы не подвергается критике. Данная тема, особенно в текущих реалиях, как никогда важна.

Целью данной научно-исследовательской работы является создание, тестирование и составление документации к разработанному приложению.

Язык программирования, используемый в данной работе: C#

Приложение - программа, ориентированная на решение конкретных задач, рассчитанная на взаимодействие с пользователем. В большинстве ОС прикладные программы не могут обращаться к ресурсам компьютера напрямую, взаимодействуя с оборудованием и другими программами через ОС.

Важно отметить, что приложение разрабатывается под ОС windows, должно отвечать современным стандартам и договорённостям, как со стороны пользовательского интерфейса, так и исходного кода.

Программа будет иметь возможность сохранения информации в формате CSV.

CSV — текстовый формат, предназначенный для представления табличных данных. Строка таблицы соответствует строке текста, которая содержит одно или несколько полей, разделенных запятыми. Формат CSV стандартизирован не полностью.

# **Теоретические основы разработки оконных приложений**

# **Структура проекта Windows Forms**

Рассмотрим базовую структуру простейшего стандартного проекта Windows Forms:

helloapp.csproj: стандартный файл проекта C#, который соответствует названию проекта (по умолчанию названию каталога) и описывает все его настройки.

helloapp.csproj.user: дополнительный файл проекта C#, который хранит специфичные для текущего пользователя настройки.

Form1.cs: содержит класс формы, которая по умолчанию запускается при старте приложения

Form1.Designer.cs: он содержит определение компонентов формы, добавленных на форму в графическом дизайнере (графический дизайнер Windows Forms на данный момент официально доступен только в Visual Studio)

Program.cs: определяет класс Program, который запускается при старте приложения и запускает форму Form1

Класс формы - Form1 представляет графическую форму - фактически то окно, которое мы увидим на экране при запуске проекта.

Этот класс определяется как частичный (с модификатором partial) и наследуется от встроенного класса Form, который содержит базовую функциональность форм.

В самом классе Form1 определён по умолчанию только конструктор, где вызывается метод InitializeComponent(), который выполняет инициализацию компонентов формы из файла дизайнера Form1.Designer.cs. По сути именно код этого файла передаётся выше через вызов InitializeComponent().

# **1.2. Элементы управления**

Элементы управления представляют собой визуальные классы, которые получают введённые пользователем данные и могут инициировать различные события. Все элементы управления наследуются от класса Control и поэтому имеют ряд общих свойств:

Anchor: Определяет, как элемент будет растягиваться

BackColor: Определяет фоновый цвет элемента

BackgroundImage: Определяет фоновое изображение элемента

ContextMenu: Контекстное меню, которое открывается при нажатии на элемент правой кнопкой мыши. Задаётся с помощью элемента ContextMenu

Cursor: Представляет, как будет отображаться курсор мыши при наведении на элемент

Dock: Задаёт расположение элемента на формеEnabled: Определяет, будет ли доступен элемент для использования. Если это свойство имеет значение False, то элемент блокируется.

Font: Устанавливает шрифт текста для элемента

ForeColor: Определяет цвет шрифта

Location: Определяет координаты верхнего левого угла элемента управления

Name: Имя элемента управления

Size: Определяет размер элемента

Width: ширина элемента

Height: высота элемента

TabIndex: Определяет порядок обхода элемента по нажатию на клавишу Tab

Tag: Позволяет сохранять значение, ассоциированное с этим элементом управления

# **1.2.1. Кнопка**

Чтобы управлять внешним отображением кнопки, можно использовать свойство FlatStyle. Оно может принимать следующие значения:

Flat - Кнопка имеет плоский вид

Popup - Кнопка приобретает объёмный вид при наведении на нее указателя, в иных случаях она имеет плоский вид

Standard - Кнопка имеет объёмный вид (используется по умолчанию)

System - Вид кнопки зависит от операционной системы

Кнопки по умолчанию

Форма, на которой размещаются все элементы управления, имеет свойства, позволяющие назначать кнопку по умолчанию и кнопку отмены.

Так, свойство формы AcceptButton позволяет назначать кнопку по умолчанию, которая будет срабатывать по нажатию на клавишу Enter.

Аналогично работает свойство формы CancelButton, которое назначает кнопку отмены. Назначив такую кнопку, мы можем вызвать ее нажатие, нажав на клавишу Esc.

# **1.2.2. ListBox**

Элемент ListBox представляет собой простой список. Ключевым свойством этого элемента является свойство Items, которое как раз и хранит набор всех элементов списка.

Элементы в список могут добавляться как во время разработки, так и программным способом. В Visual Studio в окне Properties (Свойства) для элемента ListBox мы можем найти свойство Items. После двойного щелчка на свойство нам отобразится окно для добавления элементов в список:Выделение элементов списка

При выделении элементов списка мы можем ими управлять как через индекс, так и через сам выделенный элемент. Получить выделенные элементы можно с помощью следующих свойств элемента ListBox:

SelectedIndex: возвращает или устанавливает номер выделенного элемента списка. Если выделенные элементы отсутствуют, тогда свойство имеет значение -1

SelectedIndices: возвращает или устанавливает коллекцию выделенных элементов в виде набора их индексов

SelectedItem: возвращает или устанавливает текст выделенного элемента

SelectedItems: возвращает или устанавливает выделенные элементы в виде коллекции

По умолчанию список поддерживает выделение одного элемента. Чтобы добавить возможность выделения нескольких элементов, надо установить у его свойства SelectionMode значение MultiSimple.

* + 1. **ComboBox**

Элемент ComboBox образует выпадающий список и совмещает функциональность компонентов ListBox и TextBox. Для хранения элементов списка в ComboBox также предназначено свойство Items.

Настройка оформления ComboBox:

С помощью ряда свойств можно настроить стиль оформления компонента. Так, свойство DropDownWidth задаёт ширину выпадающего списка. С помощью свойства DropDownHeight можно установить высоту выпадающего списка.

Ещё одно свойство MaxDropDownItems позволяет задать число видимых элементов списка - от 1 до 100. По умолчанию это число равно 8.

Другое свойство DropDownStyle задаёт стиль ComboBox. Оно может принимать три возможных значения:

Dropdown: используется по умолчанию. Мы можем открыть выпадающий список вариантов при вводе значения в текстовое поле или нажав на кнопку со стрелкой в правой части элемента, и нам отобразится собственно выпадающий список, в котором можно выбрать возможный вариант

DropdownList: чтобы открыть выпадающий список, надо нажать на кнопку со стрелкой в правой стороне элемента

Simple: ComboBox представляет простое текстовое поле, в котором для перехода между элементами мы можем использовать клавиши клавиатуры вверх/вниз.

* + 1. **NumericUpDown**

Элемент NumericUpDown представляет пользователю выбор числа из определённого диапазона. Для определения диапазона чисел для выбора NumericUpDown имеет два свойства: Minimum (задаёт минимальное число) и Maximum (задаёт максимальное число).

Само значение элемента хранится в свойстве Value.

По умолчанию элемент отображает десятичные числа. Однако если мы установим его свойство Hexadecimal равным true, то элемент будет отображать все числа в шестнадцатеричной системе.

Если мы хотим отображать в поле дробные числа, то можно использовать свойство DecimalPlaces, которое указывает, сколько знаков после запятой должно отображаться. По умолчанию это свойство равно нулю.При этом надо учитывать, что если мы устанавливаем значение для свойства Value в окне свойств, то там в качестве разделителя целой и дробной части используется запятая. Если же мы устанавливаем данное свойство в коде, тогда в качестве разделителя используется точка.

По умолчанию при нажатии на стрелочки вверх-вниз на элементе значение будет увеличиваться, либо уменьшаться на единицу. Но с помощью свойства Increment можно задать другой шаг приращения, в том числе и дробный.

При работе с NumericUpDown следует учитывать, что его свойство Value (как и свойства Minimum и Maximum) хранит значение decimal. Поэтому в коде мы также должны с ним работать как с decimal, а не как с типом int или double.

* + 1. **OpenFileDialog и SaveFileDialog**

Окна открытия и сохранения файла представлены классами OpenFileDialog и SaveFileDialog. Они имеют во многом схожую функциональность, поэтому рассмотрим их вместе.

OpenFileDialog и SaveFileDialog имеют ряд общих свойств, среди которых можно выделить следующие:

DefaultExt: устанавливает расширение файла, которое добавляется по умолчанию, если пользователь ввёл имя файла без расширения

AddExtension: при значении true добавляет к имени файла расширение при его отсутствии. Расширение берётся из свойства DefaultExt или Filter

CheckFileExists: если имеет значение true, то проверяет существование файла с указанным именем

CheckPathExists: если имеет значение true, то проверяет существование пути к файлу с указанным именем

FileName: возвращает полное имя файла, выбранного в диалоговом окне

Filter: задаёт фильтр файлов, благодаря чему в диалоговом окне можно отфильтровать файлы по расширению. Фильтр задаётся в следующем формате Название\_файлов|\*.расширение. Например, Текстовые файлы(\*.txt)|\*.txt. Можно задать сразу несколько фильтров, для этого они разделяются вертикальной линией |. Например, Bitmap files (\*.bmp)|\*.bmp|Image files (\*.jpg)|\*.jpg

InitialDirectory: устанавливает каталог, который отображается при первом вызове окна

Title: заголовок диалогового окна

Отдельно у класса SaveFileDialog можно еще выделить пару свойств:

CreatePrompt: при значении true в случае, если указан не существующий файл, то будет отображаться сообщение о его создании

OverwritePrompt: при значении true в случае, если указан существующий файл, то будет отображаться сообщение о том, что файл будет перезаписан

Чтобы отобразить диалоговое окно, надо вызвать метод ShowDialog().

* 1. **Цикл foreach**

Цикл foreach предназначен для перебора набора или коллекции элементов.

После оператора foreach в скобках сначала идёт определение переменной. Затем ключевое слово in и далее коллекция, элементы которой надо перебрать.

При выполнении цикл последовательно перебирает элементы коллекции и помещает их в переменную, и таким образом в блоке цикла мы можем выполнить с ними некоторые действия.

Например, возьмём строку. Строка по сути - это коллекция символов. И .NET позволяет перебрать все элементы строки - её символы с помощью цикла foreach.

Стоит отметить, что определяемая в объявлении цикла должна по типу соответствовать типу элементов перебираемой коллекции. Так, элементы строки - значения типа char - символы. Поэтому переменная c имеет тип char. Однако в реальности не всегда бывает очевидно, какой тип представляют элементы коллекции. В этом случае мы можем определить переменную с помощью оператора var.

* 1. **Конструкция if..else и тернарная операция**

Условные конструкции - один из базовых компонентов многих языков программирования, которые направляют работу программы по одному из путей в зависимости от определённых условий. Одной из таких конструкций в языке программирования C# является конструкция if..else

Конструкция if/else проверяет истинность некоторого условия и в зависимости от результатов проверки выполняет определённый код.

После ключевого слова if ставится условие. Условие должно представлять значение типа bool. Это может быть непосредственно значение типа bool или результат условного выражения или другого выражения, которое возвращает значение типа bool. И если это условие истинно (равно true), то срабатывает код, который помещён далее после условия внутри фигурных скобок.

Если блок if содержит одну инструкцию, то мы можем его сократить, убрав фигурные скобки

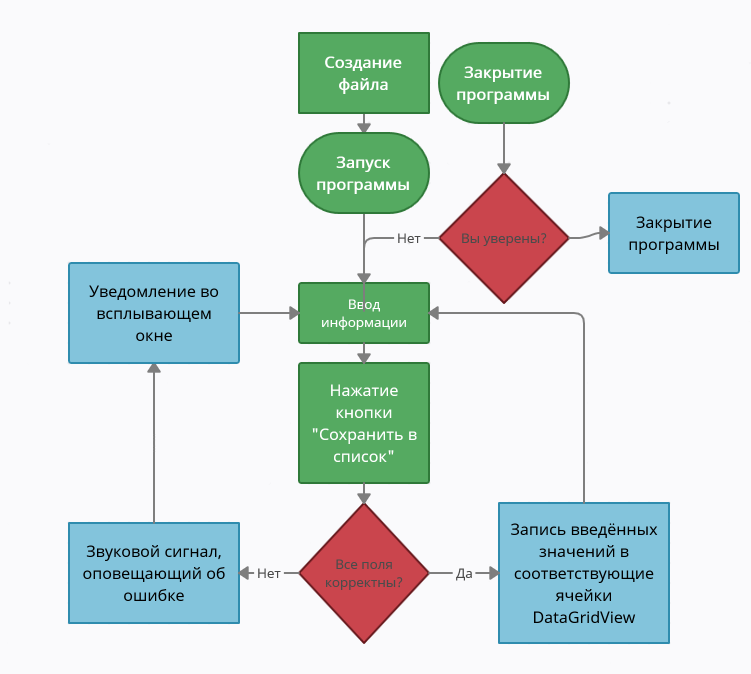
# **Разработка приложения**

# **Разработка сценария и алгоритма решения поставленной задачи**

Версия .net framework 4.8

Алгоритм работы приложения:

1. Создание файла “input.txt”, содержащий в себе виды услуг
   1. На каждой новой строке файла должен располагаться вид услуги
   2. Файл должен находиться в одной директории с исполняемым файлом
   3. Кодировка файла “UTF-8”
   4. Пример содержимого файла(построчно):
      1. Оценка недвижимости
      2. Услуги нотариуса
      3. Обслуживание ТС
   5. Если файл не создан, программа уведомит пользователя об его отсутствии и закроется
2. Запуск программы
   1. Программа подгружает содержимое файла из пункта 1 в выкидной список, на панели ввода информации о заказе
3. Ввод информации в соответствующие поля ввода приложения
4. Нажатие кнопки “Сохранить в список”
   1. Программа проверяет поля ввода информации регулярными выражениями
      1. Если какое-то значение не соответствует допустимому шаблону, то в строке состояния и во всплывающем окне отобразится текст, дающий понять какое поле введено неверно. Также посылается звуковое уведомление
      2. Если все значения полей ввода информации соответствуют ожидаемым шаблонам, то значение полей записываются в соответственные ячейки элемента DataGridView
5. Нажатие кнопки “Выгрузить таблицу в файл”
   1. Открывается окно ввода названия файла
   2. Ввод названия файла
      1. Если ввести некорректное название, выскочит MessageBox, сообщающий об ошибке
      2. Если выйти или нажать на кнопку отмены, выгрузки данных не произойдёт
      3. Если ввести корректное значение, форма закроется
   3. Программа выгрузит DataGridView в файл с расширением .csv
6. Выход из приложения
   1. Программа запросит подтверждение на своё закрытие
   2. Программа записывает в лог-файл строку, содержащую причину закрытия программы



# **Создание интерфейса приложения.**

Список компонентов, используемых в разработанном приложении(Не описаны дочерние компоненты, такие как ползунки для пролистывания(ScrollBar) и т.д.). Важно отметить, что интерфейс приложения должен соответствовать стандартам, быть удобным для использования. Человеческое восприятие цветов субъективно, поэтому интерфейс должен быть выполнен в нейтральных цветах(стандартная палитра windows).

Касаемо форм оконного приложения, их будет 3:

1. Основное окно, содержащее в себе поля для ввода пользовательских данных
2. Окно подтверждения экспорта данных в файл, формата CSV(также используется для запроса названия для сохраняемого файла)
3. Окно с логами

Диалог вывода информации для пользователя, возникающий при ошибках выполнен при помощи встроенного класса MessageBox).

Основное окно содержит строку состояния, что позволит ему проверить своё последнее действие. Если интересующее его действие уже успело скрыться, предусмотрена функция чтения текущего лога.

Согласно стандартам, была добавлена панель инструментов. Панель включает в себя:

* Действия
  + Выгрузить файл(аналог кнопки “Выгрузить таблицу в файл”)
  + Очистить таблицу(очищает всю таблицу)
* Открыть лог (открывает форму для просмотра истории строки состояния)
* Редактор таблицы
  + Включить редактирование(разрешает редактировать данные из dataViewGrid)
  + Отключить редактирование(запрещает редактировать данные из dataViewGrid)

Подробней про редактирование таблицы. Этими клавишами я вдохновился, благодаря операционной системе Linux, в частности команде sudo. Эта команда создана, дабы обезопасить пользователя от опасных действий в ОС. В ОС некоторые действия требуют определенных прав, команда sudo предоставляет эти права, однако запрашивает пароль пользователя, дав ему понять, что действие, которое он совершает, может нести риски для системы. Перед пользователем, которые впервые ввёл эту команду, выскакивает предупреждающее сообщение, о необходимости с осторожностью относиться к правам, которыми наделяет пользователя эта команда. После включение редактирования таблицы на экране отобразиться сообщение, содержащее информацию, позволяющую снять с разработчика ответственность за неверно введённые данные.

Данное сообщение будет появляться единожды, за каждый сеанс работы в приложении.

Также рекомендуется отключать режим редактирования таблицы на время ухода из-за клавиатуры, дабы избавиться от возможных проблем, связанных со случайным внесением изменений в данные.

Компоненты, используемые в работе:

* TextBox – Поле ввода текста, используется для ввода ФИО, названия услуги и стоимости
* MaskedTextBox – Поле ввода текста с маской, используется для ввода объёма услуги(времени выполнения)
* NumericUpDown – Поле ввода чисел со стрелками для увеличения/уменьшения значения, используется для ввода номера и скидки заказа
* Label – Надпись, используется для обозначения полей ввода
* ComboBox – Выпадающий список или поле со списком(зависит от значения свойства DropDownStyle, в приложении использовано DropDownList), используется для хранения видов услуг, наполняется из файла input.txt
* CheckBox – Флажок с надписью, используется для статуса услуги(выполнена/нет)
* RichTextBox – Форматируемое поле ввода текста, используется в качестве строки состояния, отображает последнее действие пользователя и информационные сообщения
* Panel – Контейнер для компонентов, используется для группировки
* FlowLayoutPanel – Контейнер для компонентов, используется для распределения полей ввода информации по всей ширине окна
* TableLayoutPanel – Контейнер для компонентов, используется для разграничения пространства внутри программы рядами
* DataGridView – Таблица, используется для хранения информации, введённой пользователем
* Button – Кнопка
* DateTimePicker – календарь, используется для выбора даты создания заказа
* Form – Окно приложения, используется для ввода имени файла для выгрузки из таблицы и подтверждения действия
* MessageBox – диалоговое окно, используется для уведомления пользователя о каком-то событии

# **Тестирование приложения**

Т.к. приложение работает с пользовательским вводом, то этот ввод нуждается в проверке.

Для проверки введённых данных был написан класс Parser. Класс Parser включает в себя 4 метода: Parser.Date, Parser.Time, Parser.Cost, Parser.File. Каждый из этих методов в качестве возвращаемого значения использует bool.

В методах используются регулярные выражения.

* Регулярное выражение для проверки валидности даты:
  + ^(0?[1-9]|[12][0-9]|3[01])\.(0?[1-9]|1[012])\.((19|20)\d\d)$
  + Может проверить дату, без учёта високосных годов и чередования 30-31 дней в месяце.
* Регулярное выражение для проверки валидности времени:
  + ^Дней:\d{1,2},Часов:\d{1,2},Минут:\d{1,2}\.$
* Регулярное выражение для проверки валидности стоимости:
  + ^\d+(?:[\,\.]\d+)?$
  + Пропускает как простое число, так и дробное.
* Регулярное выражение для проверки имени файла:
  + ^\w+(\.csv)?$
  + Пропускает как название файла, так и название файла.csv
* Регулярное выражение для проверки строки на наличие символов-разделителей:
* ^[^\;]+$
* Пропускает любые символы, кроме символов-разделителей

Чтобы убедиться в работоспособности написанных методов, был создан проект юнит тестирования. Проект включает в себя класс ParserTests, который включает в себя 6 методов для проверки методов из класса Parser.

Данные юнит тесты проверяют только на правильность работы регулярных выражение, без проверки на вызов исключения, т.к. их быть не может.

Методы класса ParserTests:

1. DateTest
   1. Метод проверяет Parser.Date на корректных данных, ожидается что Parser.Date вернёт положительное значение
   2. Т.к. Я считаю, что заказы не могут иметь дату <1900 и >2099, поэтому даты, выходящие за этот промежуток, не вводятся в данном тесте
2. InvalidDateTest
   1. Метод проверяет Parser.Date на некорректных данных, ожидается что Parser.Date вернёт отрицательное значение
3. TimeTest
   1. Метод проверяет Parser.Time на корректных данных, ожидается что Parser.Time вернёт положительное значение
4. InvalidTimeTest
   1. Метод проверяет Parser.Time на некорректных данных, ожидается что Parser.Time вернёт отрицательное значение
5. CostTest
   1. Метод проверяет Parser.Cost на корректных данных, ожидается что Parser.Cost вернёт положительное значение
6. InvalidCostTest
   1. Метод проверяет Parser.Cost на корректных данных, ожидается что Parser.Cost вернёт положительное значение

Все тесты имеют общие черты.

1. Корректные данные в крайних значения, например, месяц в дате не может принимать значение >12, день не >31. Из этого следует, что разумно было бы проверить регулярное выражение на следующие даты: 31.12.2099, 01.01.1900
2. Нет проверки на пустые значения, т.к. ни одно из полей не может его вернуть
3. Проверяются как корректные, так и не корректные данные(кроме пустых, пункт 2)

Для проверки регулярного выражение на отлавливание символов-разделителей юнит тест написал не был, т.к. в полях, где используется данная проверка допускаются любые значения, без символов-разделителей.

Т.к. приложение разрабатывается исключительно под ос Windows 10 PRO Версия 22H2 (сборка ОС 19045.2364), тестирование на других платформах и версиях не проводились(Это исключает возможность работы на других версиях данной ОС).

Согласно документации для платформы .net FrameWork, приложение, разработанное для конкретной версии платформы, будет запускаться и в более поздних версиях этой платформы. На платформе .NET Framework в максимально возможной степени обеспечивается обратная совместимость: исходный код, написанный для одной версии платформы .NET Framework, должен компилироваться в более поздних версиях этой платформы, а двоичные файлы, работающие в одной версии платформы .NET Framework, должны точно так же работать в более поздних версиях этой платформы.

# **Замечания**

Как оказалось, метод Text, класса MaskedTextBox, при маске

“Дней:00 Часов:00 Минут:00”

И введённом значении Дней:\_\_ Часов:\_\_ Минут:\_\_

Вернёт не “Дней: Часов: Минут ”, а “Дней: Часов: Минут”.

Метод обрежет все пробелы, до и после последнего не пробельного символа, как это было бы при вызове SomeString.Trim(), где SomeString – какая-то строка, с пробелами в последних/первых символах. А так как для увеличения удобства работы с полем ввода объёма услуги, я заменяю все пробелы(пустые ячейки для цифр) на 0(считается если пользователь не ввёл никакого значения, то он имел ввиду 0, например Часов:\_\_, Минут:\_5. Тут явно видно, что пользователь имеет ввиду Часов:00, Минут:05), то наличие всех пробелов в строке для меня важно, вследствие этого я добавил последним символом маски точку. Новая маска имеет вид: Дней:00\,Часов:00\,Минут:00\.

Событие FormClosed, класса Form не всегда вызывается, например, по причине убийства процесса через диспетчер задач или выключении компьютера, поэтому в некоторых случаях, файл лога может не сохраниться, в следствии чего пользователь не сможет восстановить утерянные данные. Из-за этой проблемы, мне пришлось переписать логирование. Теперь лог-файл создаётся сразу при запуске программы и пополняется сообщениями по мере их поступления(Стоило так поступить в самом начале).

В процессе тестирования оконного приложения, было решено сохранять лог-файлы не в текущую директорию, а по пути logs/файл.log, относительно рабочей папки.

Чтобы не захламлять строку состояния и лог-файл, было выбрано событие Leave, которое есть у всех компонентов WinForms. Если бы мы использовали, к примеру, событие TextChanged, класса TextBox, то при каждом новом символе, введённом в это поле, у нас бы вызывалось TextChanged событие, со всеми вытекающими, как запись в лог-файл и строку состояния незаконченной строки.

При вызове метода Show, класса Form, выполнение кода в основном окне продолжается, поэтому для подтверждения действия пользователя он не подходит. Вместо него необходимо использовать ShowDialog. Тогда выполнение когда основного окна остановится, что даст возможность позже получить значение из открывшегося окна.

Из-за того, что формат CSV не полностью стандартизирован, то символ -разделитель может отличаться в разных условиях(версия ОС, ОС, дистрибутивы ОС). Это порождает проблему, когда не до конца известно, какие символы необходимо экранировать, для избегания съезда строки. Например, если символ-разделитель будет запятой, то при вводе дробного числа, с разделителем дробной части символом запятой, то программа будет работать некорректно. При экспорте данных из DataGridView в csv файл, строка заказа съедет(например, дробное число станет целым, а его дробная часть присвоится другой колонке). Потери информации из-за этого не предвидится, однако форматирование таблицы испортится.

Для решения этой проблемы, я условился, что символом-разделителем всегда будет являться точка с запятой, т.к. его поддерживает табличный процессор Excel, Версии 2002 (сборка 12527.22197, нажми и работай), установленный на моём компьютере. Полями, которые позволяют ввести данный символ-разделитель являются: ФИО/Название, Наименование услуги, Стоимость услуги. Исходя из этого, я разработал дополнительные регулярные выражения.

Что касается логирования изменения значений ячеек dataGridView1 вручную, событие CellValueChanged вызывается при любом изменении значения, даже если оно совершалось из кода программы. Из-за этого мне пришлось использовать событие CellEndEdit. В свою очередь, это позволило мне сохранять в строку состояния старое и новое значения ячейки .

# **Заключение**

Язык C# и платформа .net служат хорошими инструментами для разработки оконных приложений, а благодаря интегрированной среде разработки Visual Studio, написание кода значительно облегчается. Присутствие конструктора форм ускоряет создание оконных программ в разы. Можно изменять параметры компонентов и следить за мгновенными изменениями. VS разработана с учётом всех потребностей разработчиков программного обеспечения. Благодаря расширениям, открывается возможность ускорить некоторые моменты в процессе разработки.

Сам язык имеет низкий порог вхождения, что позволяет обучиться работе с ним людям, далёким от программирования.

Благодаря ознакомительной практике мне удалось подчеркнуть для себя некоторые особенности, описанные в главе “Замечания”, которые, несомненно, помогут в решении будущих задач. Теперь, применяя эти особенности на практике, я смогу сэкономить время во время разработки следующих программ и приложений.

Также эта работа подтолкнула меня к изучению: особенностей платформы .net, текстового формата представления табличных данных(CSV), особенностям разных операционных систем и версиям ОС windows, в частности.

Разработанная программа может служить базой для последующих работ, связанных с администрированием баз данных. Также эта программа может послужить и для офисных работников, использующих Excel.

Проведённая практика систематизировала знания, полученные в процессе обучения в трёх семестрах. Практическое применение всегда помогает закрепить пройденный материал, а когда делаешь такой объёмный проект, появляется желание узнать ещё больше, что я прочувствовал на себе, дополнив приложение дополнительным функционалом, таким как сохранение данных в новом для меня формате.

# **Библиографический список**

Microsoft Learn: сайт. – URL: https://learn.microsoft.com/ru-ru/ (дата обращения: 18.12.2022). – Текст: электронный.

Форум программистов и сисадминов: сайт. ­ URL: <https://www.cyberforum.ru/> (дата обращения: 18.12.2022). – Текст: электронный.

StackOverFlow: сайт. URL: https://stackoverflow.co/ (дата обращения: 18.12.2022). – Текст: электронный.

Regex101: сайт. URL: https://regex101.com/ (дата обращения: 18.12.2022). – Текст: электронный.

GitHub: сайт. URL: https://github.com/ (дата обращения: 18.12.2022). – Текст: электронный.

Википедия: сайт. URL: https://ru.wikipedia.org (дата обращения: 18.12.2022). – Текст: электронный.

Metanit: сайт. URL: https://metanit.com/ (дата обращения: 18.12.2022). – Текст: электронный.

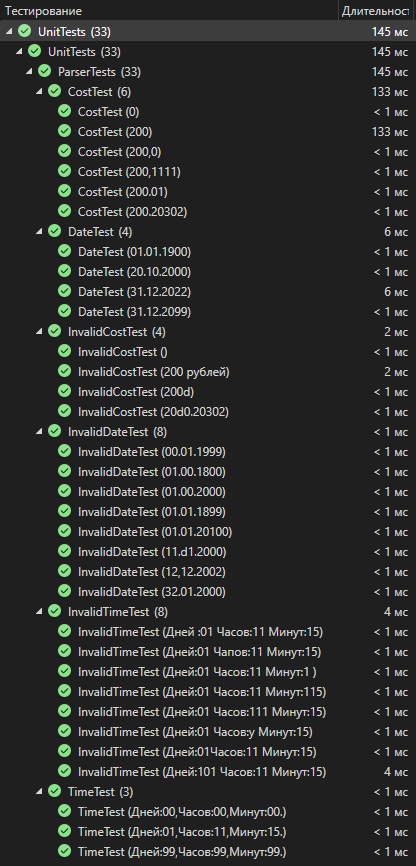
Habr: сайт. URL: https://habr.com (дата обращения: 18.12.2022). – Текст: электронный.

Форум программистов: сайт. URL: https://programmersforum.ru/(дата обращения: 18.12.2022). – Текст: электронный.

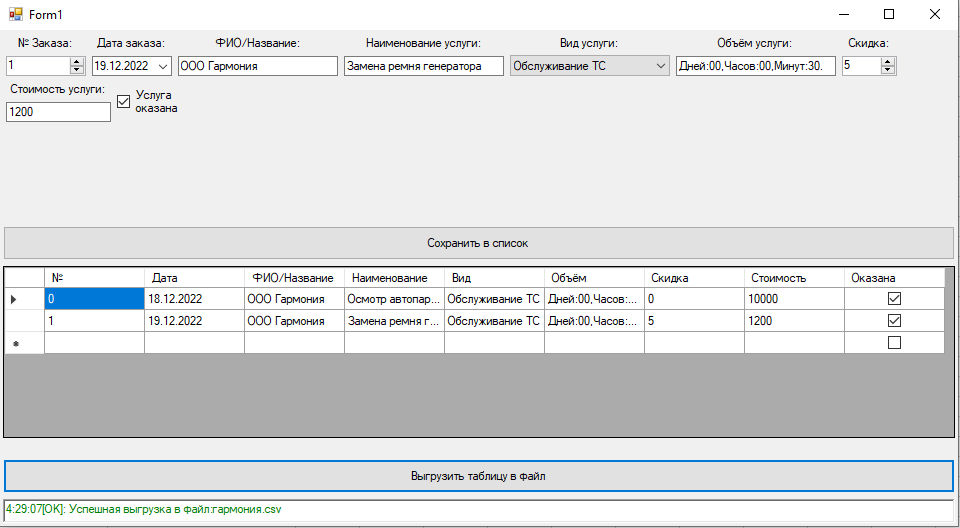
PB: сайт. URL: <https://www.programbeginner.ru/> (дата обращения: 18.12.2022). – Текст: электронный.

# **Приложения**

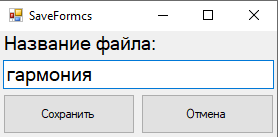
Результаты тестирования:



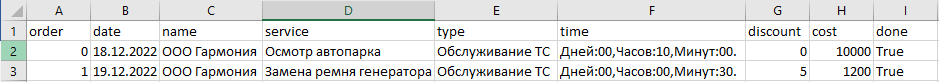
Пример введённых данных:



Окно ввода названия файла:



Пример содержания экспортированных данных в файл.csv:



Пример содержания лог-файла:

4:26:51[OK]: Программа запущена

4:26:56[INFO]: №Заказа:0

4:27:12[INFO]: Название/ФИО:ООО Гармония

4:27:29[INFO]: Услуга:Осмотр автопарка

4:27:33[INFO]: Вид услуги:Обслуживание ТС

4:27:38[INFO]: Объём услуги:Дней:00,Часов:10,Минут:00.

4:27:49[INFO]: Стоимость услуги:10000

4:27:49[INFO]: Услуга выполнена

4:27:52[OK]: Запись сохранена

4:27:58[INFO]: Дата:19.12.2022

4:27:59[INFO]: Название/ФИО:ООО Гармония

4:28:29[INFO]: Услуга:Замена ремня генератора

4:28:37[INFO]: Объём услуги:Дней:00,Часов:00,Минут:30.

4:28:38[INFO]: Скидка:5

4:28:45[INFO]: Стоимость услуги:1200

4:28:49[INFO]: №Заказа:1

4:28:49[OK]: Запись сохранена

4:29:07[OK]: Успешная выгрузка в файл:гармония.csv

4:34:33[SHUTDOWN]: Завершение работы, причина:UserClosing

Класс для экспорта таблицы в формат csv:

using System;

using System.IO;

using System.Text;

using System.Windows.Forms;

namespace Practical

{

static public class CSV

{

/// <summary>

/// Сохраняет данные из компонента DataGridView в формате CSV

/// Если файл с данным именем существует, он будет перезаписан

/// </summary>

/// <param name="filename">Название файла</param>

/// <returns></returns>

static public bool Save(string filename, DataGridView dataGridView)

{

StreamWriter streamWriter = new StreamWriter(filename, false, Encoding.UTF8);

try

{

for (int i = 0; i < dataGridView.Columns.Count; i++)

{

streamWriter.Write(dataGridView.Columns[i].Name + (i == dataGridView.Columns.Count - 1 ? "" : ";"));

}

streamWriter.Write(Environment.NewLine);

for (int i = 0; i < dataGridView.Rows.Count; i++)

{

for (int j = 0; j < dataGridView.Rows[i].Cells.Count; j++)

{

streamWriter.Write(dataGridView.Rows[i].Cells[j].Value + (j == dataGridView.Rows[i].Cells.Count - 1 ? "" : ";"));

}

streamWriter.Write(Environment.NewLine);

}

streamWriter.Close();

streamWriter.Dispose();

return true;

}

catch(Exception ex)

{

streamWriter.Close();

streamWriter.Dispose();

return false;

}

}

}

}

Пример экспортированной таблицы, при введённом символе-разделителе в поле ввода ФИО:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| order | date | name | service | type | time | discount | cost | done |  |
| 0 | 18.12.2022 | иван | иванов | Физ-ра | Репетиторство | Дней:02,Часов:00,Минут:00. | 1 | 23 | True |