

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное**

**учреждение Московской области**

**«Люберецкий техникум имени Героя Советского Союза,**

**лётчика-космонавта Ю.А. Гагарина»**

**КУРСОВАЯ РАБОТА**

**по МДК.01.01 «Разработка программных модулей »**

**по МДК.01.03 «Разработка мобильных приложений »**

специальность

09.02.07 «Информационные системы и программирование»

**Тема:**

Разработка генератора паролей с помощью технологии Windows Forms Net.Framework на языке C#.

Группа 185

Выполнил/а/ студент     \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_      Костин Макар Русланович

                                             (подпись)                         (ФИО полностью)

Руководитель                                                         Жирнова Юлия Витальевна

                                             (подпись)                           (ФИО полностью)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Люберцы-2021 год

**Содержание**

**Введение**

# Современный мир теперь сложно представить без информационных систем. Их можно встретить на каждом шагу и это неудивительно, ведь они значительно упростили нашу жизнь.

# Так как в наше время информационные системы очень востребованы, я предлагаю вам попробовать себя в роли программиста.

# Целью данной курсовой работы является разработка генератора паролей с помощью технологии Windows Forms Net.Framework на языке C#.

# Для достижения данной цели нам потребуется решить следующие задачи:

# Анализ предметной области;

# Изучение основных инструментов технологии WPF;

# Выявление особенности Windows Forms Net.Framework;

# Построение структуры приложения, которое включает:

# Блокнот;

# Счетчик;

# Генератор;

# Конвертер;

# Пароль;

# Тестирование работы приложения;

# Теоретическая часть

# Актуальность

# Начнем с того, что C# отлично подходит для начинающих программистов и именно с него стоит начинать изучение оконных приложений. Данная курсовая работа может быть полезна для тех, кто хотел бы попробовать себя в роли программиста.

# Для людей с опытом программирования на си-подобных языках изучение C# проблем не доставит, но и для новичков этот язык проще чем классический C++ или Java. А попробовать себя в нем стоит, ведь это довольно востребованный язык и специалисты со стажем от одного года могут зарабатывать уже около 100 тысяч рублей. Если верить hh.ru, разработчики с опытом более трех лет могут претендовать на 300 тыс. рублей в месяц и даже на более высокую заработную плату, и это не учитывая различных бонусов (не только денежных) от компании.

# C# так же может использоваться для разработки игр. На данный момент связка Unity(игровой движок) + С# популярна в небольших компаниях и у инди-разработчиков из-за отличной функциональности, дружелюбного сообщества и возможности использования для написания любых 2D и 3D игр.

# Теория

# Для того, чтобы создать эти мини-программы необходимо изучить основные инструменты технологии WPF.

# XAML — это язык разметки на основе XML, который служит для определения внешнего вида приложения в декларативной форме. Обычно он используется для создания окон, страниц и пользовательских элементов управления, а также их заполнения элементами управления, фигурами и графическими элементами.

# При разработке поведения приложения главной задачей является обеспечение реакции на действия пользователя, включая обработку событий (таких как выбор пункта меню или нажатие на кнопку), и вызов в ответ бизнес-логики и логики доступа к данным. В WPF такое поведение реализуется в коде, связанном с разметкой. Этот код называется кодом программной части. В приведенном ниже примере показана обновленная разметка из предыдущего примера и код программной части.

# Возможности взаимодействия с пользователем, обеспечиваемые моделью приложения, реализуются с помощью сконструированных элементов управления. В WPF элемент управления — это общий термин, который относится к категории классов WPF, размещаемых в окне или на странице, имеющих пользовательский интерфейс и реализующих некоторое поведение.

# Более подробную информацию см. в разделе Элементы управления.

# Функциональная классификация элементов управления WPF

# Ниже перечислены встроенные элементы управления WPF.

# Кнопки: и RepeatButton.

# Отображение данных: ListView и TreeView.

# Отображение и выбор даты: и DatePicker.

# Диалоговые окна: PrintDialog и SaveFileDialog.

# Цифровой рукописный ввод: и InkPresenter.

# Документы: FlowDocumentPageViewer, FlowDocumentReader, FlowDocumentScrollViewer и StickyNoteControl.

# Входные данные: RichTextBox и PasswordBox .

# Макет: BulletDecorator, Canvas, DockPanel, Expander, Grid, GridView, GridSplitter, GroupBox, Panel, ResizeGrip, Separator, ScrollBar, ScrollViewer, StackPanel, Thumb, Viewbox, VirtualizingStackPanel, Window и WrapPanel .

# Носитель: MediaElement и SoundPlayerAction.

# Меню: Menu и ToolBar.

# Навигация: Hyperlink, Page, NavigationWindow и TabControl.

# Выбор: ComboBox, ListBox, RadioButton и Slider.

# Сведения о пользователе: Label, Popup и ProgressBarStatusBarTextBlockToolTip.

# Платформа .NET Framework — это технология, которая поддерживает создание и выполнение веб-служб и приложений Windows. При разработке платформы .NET Framework учитывались следующие цели.

# Обеспечение согласованной объектно-ориентированной среды программирования для локального сохранения и выполнения объектного кода, для локального выполнения кода, распределенного в Интернете, либо для удаленного выполнения.

# Предоставление среды выполнения кода, в которой:

# сведена к минимуму вероятность конфликтов в процессе развертывания программного обеспечения и управления его версиями;

# гарантируется безопасное выполнение кода, включая код, созданный неизвестным или не полностью доверенным сторонним изготовителем;

# исключаются проблемы с производительностью сред выполнения скриптов или интерпретируемого кода;

# обеспечиваются единые принципы разработки для разных типов приложений, таких как приложения Windows и веб-приложения;

# обеспечивается взаимодействие на основе промышленных стандартов, которое гарантирует интеграцию кода платформы .NET Framework с любым другим кодом.

# Особенность IDE

# Java

# Данный язык/среда изначально задумывались как нечто мультиплатформенное. На Java написано большое количество приложений, крупные проекты вроде Eclipse используют именно этот фреймворк.

# Большой минус Java с точки зрения пользователя — необходимость устанавливать фреймворк. Это безусловно сложнее установки Flash player'а. И размер инсталлятора (я качал 80+ мб).

# Learning curve имеет критическое значение для любого проекта, который ставит на вирусное продвижение. Если 30% ваших пользователей не смогут поставить Java-машину (лень качать, неудобный сайт, отложили «на потом», ...) — у ваших конкурентов будет фора в 30%. Эта огромная доля пользователей, потерять которых мой проект себе позволить не может.

# Все приложения, которыми мне доводилось пользоваться, не использовали «родной» интерфейс Win32. Не знаю, чем руководствовались разработчики, но с точки зрения конечного пользователя это выглядит очень несимпатично.

# C#

# Язык C# и платформа .NET вышли из под крыла компании Microsoft, которая никогда не отличалась уважением к мультиплатформенности. Данный язык был тепло принят разработчиками и энтузиасты проекта Mono даже сделали аналог фреймворка под Linux/Unix/Mac Os X.

# Аналогично с Java, приложения на C# имеют большой минус — необходимость устанавливать фреймворк. Я сам отказался от установки несколько приложений, которые требовали этого фреймворка. Менее продвинутые пользователи будут ещё более требовательными.

# В остальном — сплошные плюсы, на мой взгляд.

# C++

# Старичок дотянул до наших дней и замечательно себя чувствует. Много приложений под платформы Linux и Windows до сих пор пишутся на этом языке.

# Программы, написанные на C++, являются примером для других по размеру дистрибутива и экономному использованию системных ресурсов (процессор, память). Тем не менее у разработчиков есть масса претензий к С++. По моему мнению, язык является «устаревшим» и его популярность в дальнейшем будет снижаться, что подтверждается индексом TIOBE.

# С точки зрения развития проекта, по сравнению с динамическими интерпретируемыми языками (вроде Ruby и Python), разработка на данном языке может иметь менее высокую скорость и более высокие издержки изменения проекта. Для стартапа, которому не столь важна производительность приложения, это может стать существенным минусом.

# Python

# Взглянуть на Python в качестве платформы для desktop-приложений меня заставила программа MusicBrainz Picard. Несмотря на свою скриптовую сущность, Python легко собирается в один exe-файл, не требуя от пользователя установки дополнительных компонентов.

# В случае разработки небольшого приложения, интерпретируемые языки вроде Python будут большим плюсом. Легкость написания и высокая скорость изменения приложения пригодится любому стартапу.

# Огромным минусом различных «модных» технологий является их низкое распространение, а значит серьезные проблемы в поиске квалифицированных кадров. Ситуация с поиском программистов и так плачевная, а если ограничиться узким языком — можно вообще никого не найти. С другой стороны, храбрость перейти на новый язык имеют наиболее прогрессивные разработчики. Может получиться так, что выбрав «перспективный» язык, мы сразу отсечем миллионы середнячков, оставив себе выбор из нескольких перспективных разработчиков.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Java | C++ | C# | Python |
| Высокая скорость разработки | - | - | - | + |
| Мультиплатформенность | + | - | + | - |
| Большое кол-во библиотек | - | + | - | + |
| Большое кол-во кадров | + | + | + | - |
| Необходимость установки фреймворка | + | - | + | - |
| Развитость фреймворка | + | - | + | - |
| Отличная производительность. | - | + | + | - |

# Практическая часть

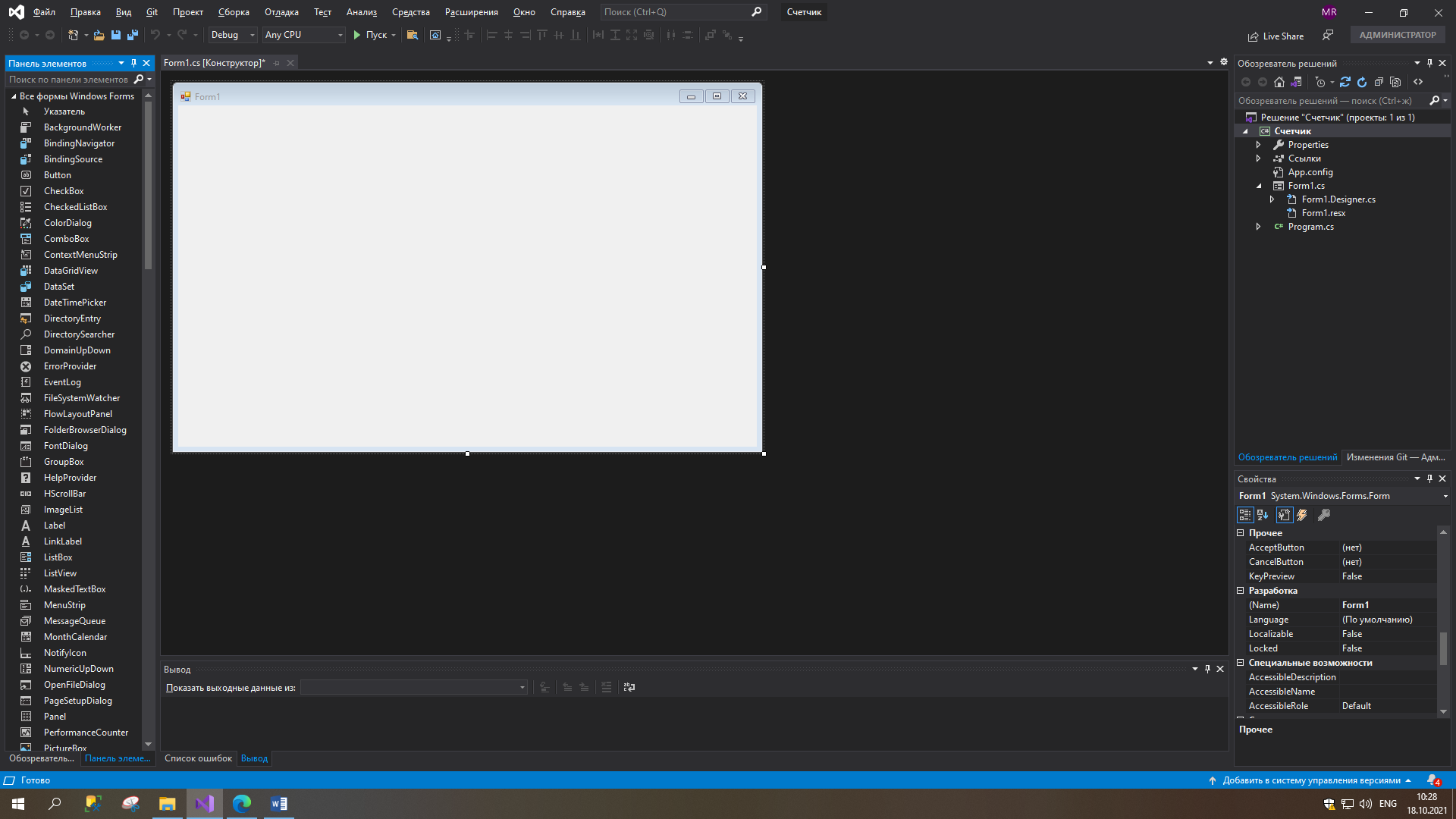
# Постановка задачи

# Структура приложения

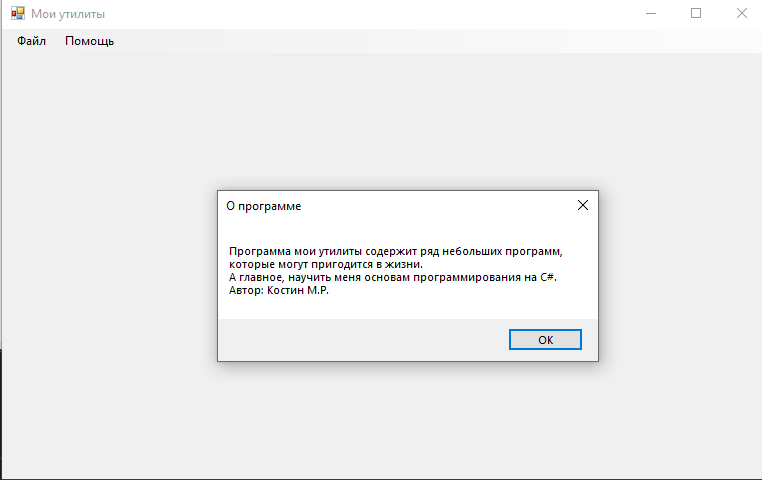
# C#. Мини-программы

**C#. Мини-программы.**

Для работы нам понадобится открыть Microsoft Visual Studio и создать проект Приложение Windows Form (.NET Framework). После создания в проекте нужно будет вывести на экран: панель управления, свойства проекта и обозреватель решений.



Дадим новое название нашей форме: «MainForm». А так же сделаем новый заголовок: «Мои утилиты». Далее нужно добавить MenuStrip из панели элементов. В панель добавим разделы: «Файл» и «Помощь», в «Файле» должна быть вкладка «Выход», а в «Помощь» должна быть «О программе». Дадим этим разделам новые имена для удобства: «tsmiExit» и «tsmiAbout». Присвоим им события, а точнее Click. Для «Выход» мы напишем команду «this.Close();», а для «О программе» напишем MessageBox.Show(«Программа мои утилиты содержит ряд небольших программ, которые могут пригодится в жизни. А главное, научить меня основам программирования на C#. Автор: Костин М.Р.», «О программе»);



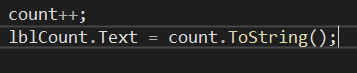
Теперь нам требуется добавить TabContorl из панели элементов. Расположим его по всей нашей форме данным свойством:



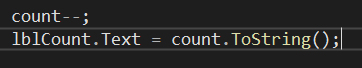
Изменим название tabPage1 на: «Счетчик». Добавим три Button и Label на первую страницу. Переименуем три кнопки в: «+», «Сброс» и «-». А так же дадим им Name: «btnPlus», «btnMinus» и «btnReset». Label же мы присвоим текст: «0», а Name: «lblCount». Далее чуть подправим местоположение кнопок, сделаем их полужирными и поставим другой размер.



В public partial class MainForm : Form нам нужно написать: «int count = 0;». Далее кликнем по btnPlus два раза и напишем код в появившийся private void:



Аналогичную работу проделываем с btnMinus:



И осталось заполнить только btnReset:

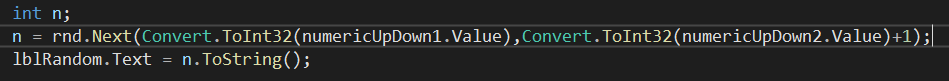


Дадим новое название второй странице: «Генератор». Добавим Button, три Label и два NumericUpDown. Сменим текст Button на: «Задать», а Name: «btnRandom». Label2: «От», Label3: «До» и Label1: «lblRandom».

Зададим Value для numericUpDown1: «1», а для numericUpDown2: «6».

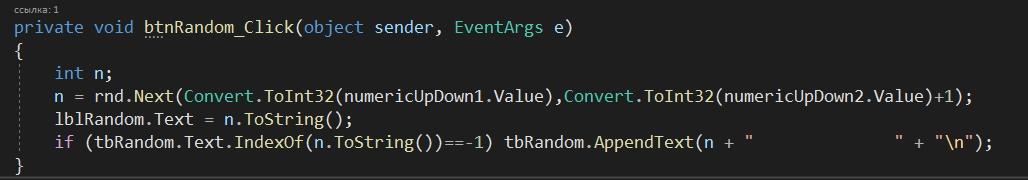
В public MainForm() напишем код: «rnd = new Random();», а в public partial class MainForm : Form нужно будет написать: «Random rnd;».

Создаем btnRandom\_Click и заполняем его:

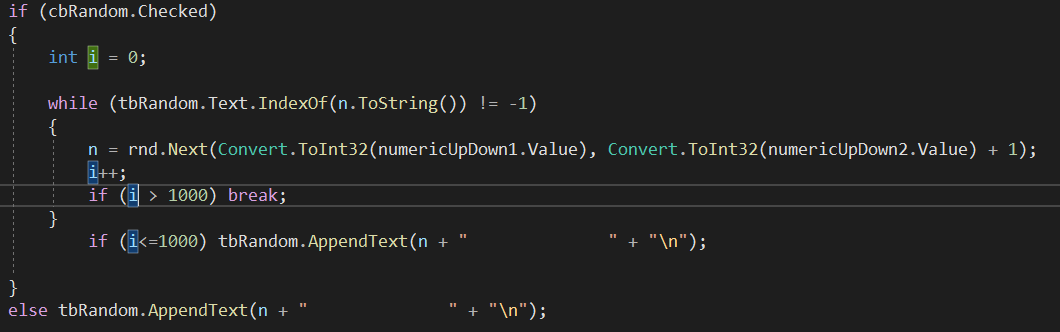


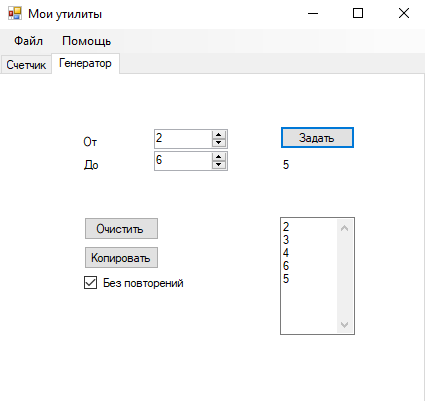
Добавим TextBox и дадим ему Name: «tbRandom» и включим MultiLine. Напишем код для btnRandom\_Click: «tbRandom.AppendText(n + «\n»);». Создадим кнопку и дадим ей имя: «btnRandomClear», а текст: «Очистить». Создаем private void btnRandomClear\_Click и пишем код: «tbRandom.Clear();». Добавим еще одну кнопку с именем: «btnRandomCopy», и текстом: «Копировать». И пишем для него код: «Clipboard.SetText(tbRandom.Text);».

Далее немного обновим наш код для того, чтобы не было повторений среди чисел:



Создадим CheckBox и дадим ей имя: «cbRandom», а текст: «Без повторений». И еще раз обновим наш код для btnRandom\_Click:





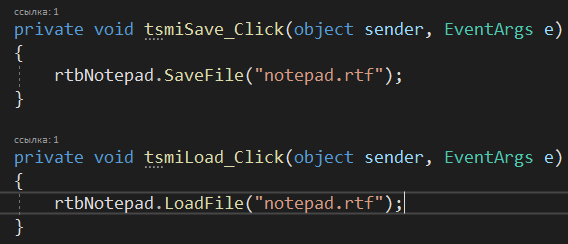
Далее нам нужно добавить еще один tabPage, дадим ему Text: «Блокнот». На эту страничку нам нужно будет добавить RichTextBox и растянуть его на всю страницу. Затем добавить еще один menuStrip и переименуем его в: «Блокнот». Добавим ему два действия: «Вставить дату» и «Вставить время». А так же дадим им Name: «tsmiInsertDate» и «tsmiInsertTime».

Напишем код для tsmiInsertDate\_Click: «rtbNotepad.AppendText(DateTime.Now.ToShortDateString() + "\n");».

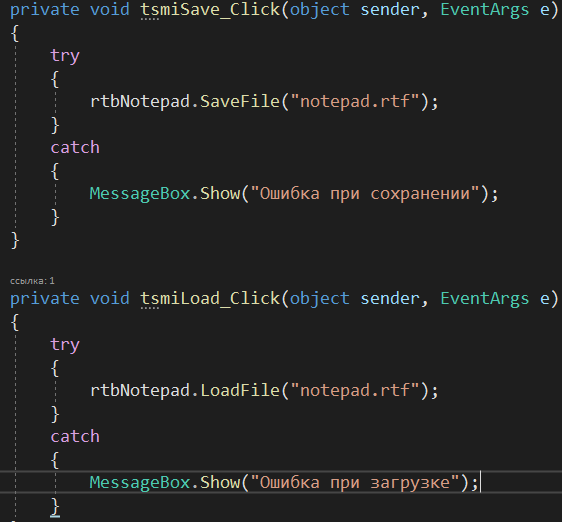
И напишем код для tsmiInsertTime\_Click: «rtbNotepad.AppendText(DateTime.Now.ToShortTimeString() + "\n");».

Теперь добавим еще пару действий к блокноту: «Сохранить» и «Загрузить». Переименуем их: «tsmiSave» и «tsmiLoad».

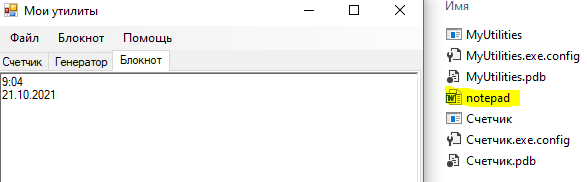
Напишем код:



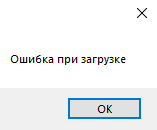
Подправим код для того, чтобы он мог вывести ошибку:



Проверяем наш код:

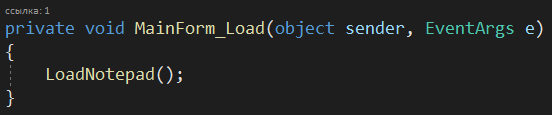


Проверяем на ошибку:

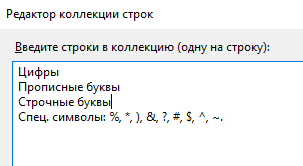


Создадим void LoadNotepad() и перенесем в него код из tsmiLoad\_Click, а в tsmiLoad\_Click напишем: «LoadNotepad();».

А так же создадим новый обработчик:



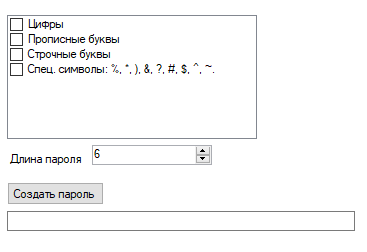
Создаем новую страничку: «Пароли», и добавляем на нее checkedListBox. Заполним Item:



Включим CheckOnClick. Добавим NumericUpDown, Label, Button и TextBox.

Теперь поменяем Name: «nudPassLength», «clbPassword», «btnCreatePassword» и «tbPassword».

Для NumericUpDown выставим максимальное(20) и минимальное(1) значения.



Создаем обработчик событий для Button.

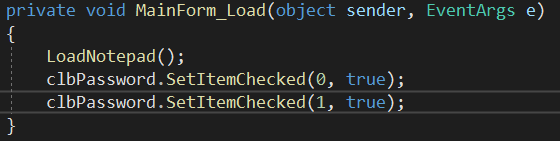
Напишем для него код:



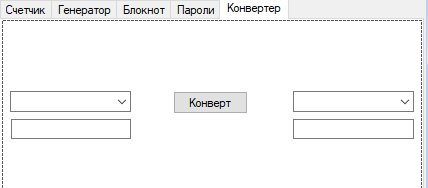
И так же в public partial class MainForm : Form запишем: «char[] special\_chars = new char[] { '%', '\*', ')', '&', '?', '#', '$', '^', '~' };»

И добавим еще один код для того, чтобы пароль сохранялся в буфер обмена: «Clipboard.SetText(password);».

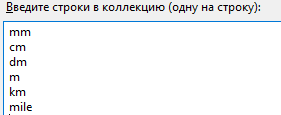
Напишем еще один код:



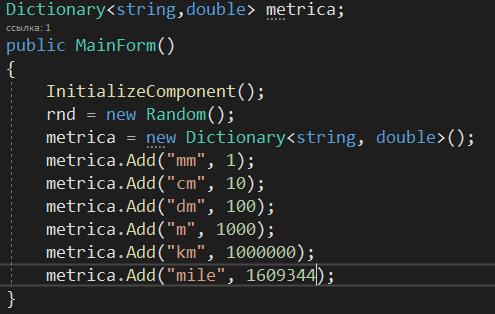
Добавим новый tabPage с именем: «Конвертер». На новый tabPage перенесем два ComboBox, Button и два TextBox.



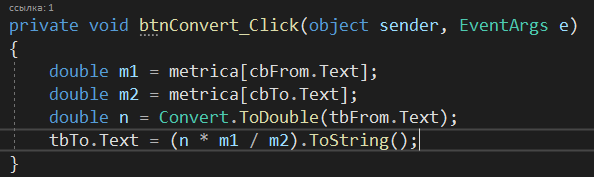
Заполним Items для cbFrom и cbTo:



Обновим код:



Напишем код для нашей кнопки:

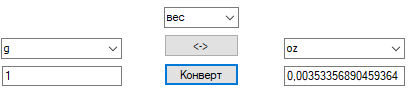


Добавим новую кнопку btnSwap и cbMetric. Заполним Items: «длина, вес».

Напишем код:



И напишем такой же код, но для case «вес». С единственным отличием, теперь нам будут нужны: «g, kg, t, lb, oz».



На этом все задания выполнены, работа закончена.

**Выводы**

# Целью данной курсовой работы являлось разработка генератора паролей с помощью технологии Windows Forms Net.Framework на языке C#.

# Для достижения данной цели были решены следующие задачи:

# Анализ предметной области.

# Изучение основных инструментов технологии WPF.

# Выявление особенности Windows Forms Net.Framework

# Построение структуры приложения, которое включает:

# Блокнот;

# Счетчик;

# Генератор;

# Конвертер;

# Пароли;

# Тестирование работы приложения.