Desktop-приложения

ВВЕДЕНИЕ

На сегодняшний день существует огромные множества технологий разработки Desktop-приложений, у каждой из которых есть свое предназначение, в зависимости от структуры создаваемого приложения, его функционала и т.п. Некоторые из технологий уже устарели и не позволяют создать приложения, которые бы удовлетворяли требованиям пользователя. А другие технологии, наоборот, становятся более популярными. Примерами технологий разработки Desktop-приложений являются Java, С++, С#, Python, Ruby on Rails.

Существуют огромное количество программных сред, которые используют данные технологии для разработки Desktop-приложений. Современные программные среды включают в себя не только редактор кода, но и компилятор, отладчик и многое другое.

1. Что такое Desktop -приложения?

Десктопные приложения – это полнофункциональные программы, которые работают вне зависимости от других приложений и требуют наличие оператора. Для их работы необходимы достаточные аппаратные ресурсы компьютера, само приложение и набор функций для работы с приложением.

Такие приложения размещаются на компьютере пользователя. Они не требуют для работы подключение к интернету, взаимодействуют с пользователем посредством стандартного интерфейса, имеют более высокое быстродействие, зависят от используемой операционной системы и требуют установку на каждый компьютер пользователя, желающего работать с данным приложением. Это текстовые редакторы, медиа-плееры, программы расчета, исчисления, изучения – в общем все программы, которые установлены у нас на компьютерах, являются desktop-приложениями. Так как мы имеем доступ к системным файлам программы, данный тип приложений более уязвим, и полностью зависит от действий пользователя.

1. Актуальные языки для написания desktop-приложение
   1. Java

Данный язык 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 часто становится pain in the ass. И размер инсталлятора (я качал 80+ мб), и постоянная путаница в названиях (JRE, J2SE, JDK, JVM, ...) не играют на руку разработчикам приложений под Java.

Learning curve имеет критическое значение для любого проекта, который ставит на вирусное продвижение. Если 30% ваших пользователей не смогут поставить Java-машину — 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.

Плюсы: мультиплатформенность, большое количество кадров, развитость фреймворка.

Минусы: необходимость установки фреймворка, кривость GUI, низкая производительность.

* 1. 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.

Плюсы: мультиплатформенность, большое количество кадров, хорошая производительность, развитость фреймворка.

Минусы: необходимость установки фреймворка.

* 1. 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. 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, 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.

Плюсы: отличная производительность, большое количество кадров, большое количество библиотек.

Минусы: невысокая скорость разработки.

* 1. 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 итак 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.

Плюсы: высокая скорость разработки и изменений, хорошая интеграция с библиотеками на С и С++.

Минусы: мало кадров, низкая производительность.

Выводы

К сожалению, любая из вышеперечисленных платформ имеет свои плюсы и минусы, однозначного решения найти не удалось. Я решил остановиться на C#.

1. Мой выбор С#

Я выбрал данный язык программирования так как он хорошо подходит для написания программ под Windows, и для всего остального так как он универсален.

* 1. Краткая история C# и его характеристики

С# – это объектно-ориентированный язык программирования. Он был создан в период с 1998 по 2002 год командой инженеров Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота.

Язык входит в семью С-подобных языков. Синтаксис приближен к Java и C++. Его особенности:

* статистическая типизация,
* поддерживается полиморфизм,
* поддерживается перегрузка операторов,
* доступна делегация, атрибуты, события, обобщенные типы и анонимные функции.

Разработка Microsoft много особенностей унаследовала у Delphi, Smalltalk и Java. При этом создатели нового языка исключили из своего детища многие практики и спецификации, считающиеся «проблемными».

* 1. Основные преимущества языка

С# популярен за счет своей «простоты». Простоты для современных программистов и больших команд разработчиков, чтобы те могли в сжатые сроки создавать функциональные и производительные приложения. Этому способствуют нетипичные конструкции языка и специфичный синтаксис, помогающий максимально органично реализовать намеченные функции.

Популярность языка – еще одно значимое преимущество. Большое количество поклонников C# способствуют его развитию. Также это благоприятно влияет на рост числа вакансий, связанных с разработкой на языке Microsoft. Программисты, хорошо знакомые с С#, востребованы в индустрии, несмотря на их большое и постоянно увеличивающееся количество.

Понятный синтаксис C# заметно упрощает не только разработку как таковую, но и другие важные аспекты совместной работы, например, чтение чужого кода. Это упрощает процесс рефакторинга и исправления ошибок при работе над приложениями в больших командах.

Также нельзя не упомянуть низкий порог вхождения. С# – популярная и достаточно простая в освоении технология. Уже через полгода можно поднатореть в разработке и начать делать полноценные программы.

* 1. Что разрабатывают с помощью С#

Язык C# практически универсален. Можно использовать его для создания любого ПО: продвинутых бизнес-приложений, видеоигр, функциональных веб-приложений, приложений для Windows, macOS, мобильных программ для iOS и Android.

Видеоигры

C# без преувеличения крайне популярен среди создателей видеоигр. Язык используется для разработки игр под Windows, macOS, Android и iOS. Все дело в Unity – платформе для работы с 3D-графикой. С# лучше остальных языков адаптирован под работу с этим движком. Поэтому программисты обычно не выбирают, а сразу используют связку Unity + C#.

ПО для защиты систем

Безопасность ваших программ и операционных систем обеспечивается благодаря мощным утилитам на базе C#. Колоссальное количество вирусов, на ежедневной основе атакующих компьютеры пользователей, блокируется инструментами, созданными с помощью языка Microsoft. Аналогичная ситуация наблюдается в крупном бизнесе – мировые корпорации защищаются от хакерских атак с помощью ПО, написанного на С#.

Приложения для Windows

Практически вся операционная система Microsoft существует благодаря С#. Привычные вам утилиты и приложения созданы с использованием этого языка и фреймворков, разработанных для него.

В эту категорию попадает мессенджер Skype, браузер Internet Explorer, среда для разработки Visual Studio 2012, Microsoft Office (все его составляющие, включая Word, PowerPoint, Excel, Outlook и так далее).

Сюда же можно отнести продукты компании Adobe (Photoshop, Lightroom), браузер Mozilla Firefox и Winamp.

Мобильные приложения

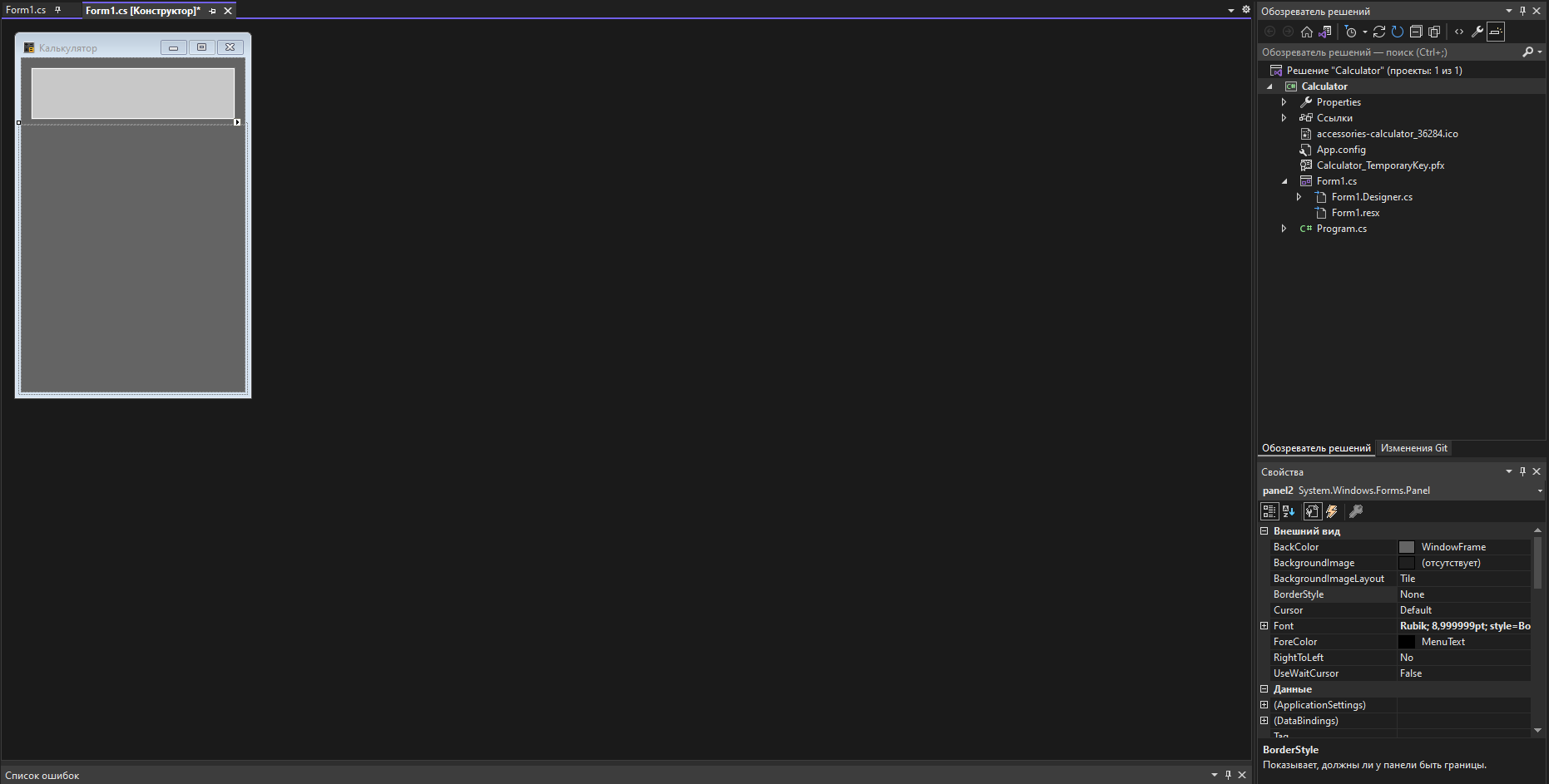
В некоторых кругах программистов С# считается чуть ли не лучшим языком для проектирования мобильных приложений. Все благодаря возможности создавать с помощью этого языка нативные программы для любых платформ (iOS, Android). Для создания приложений, которые идеально работают на Айфоне и на Андроид-смартфонах, используется IDE Xamarin.

Из известных программ, написанных на С#, стоит отметить Slack, Pinterest, Tableau, The World Bank и другие. «Плиточные» программы, появившиеся в Windows 8, практически все построены на базе C# и XAML.

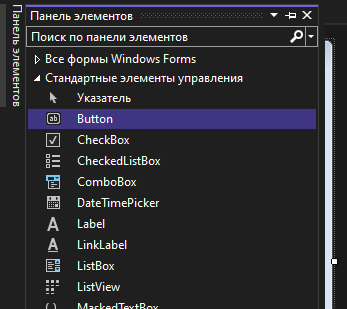
* 1. Создание программы на С#

Я решил сделать калькулятор

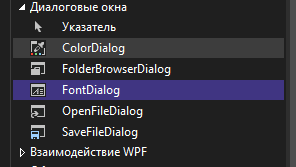
1. Скачал и установил Visual Studia 2022
2. Выбираем проект «Приложение Windows Forms»
3. Создаем форму калькулятора



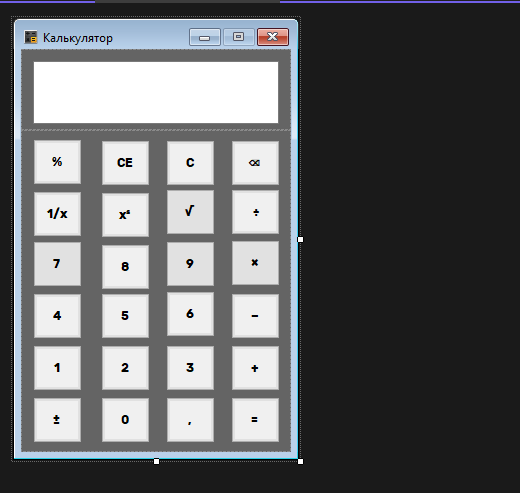
1. Добавляем кнопки с помощью «Панель элементов»



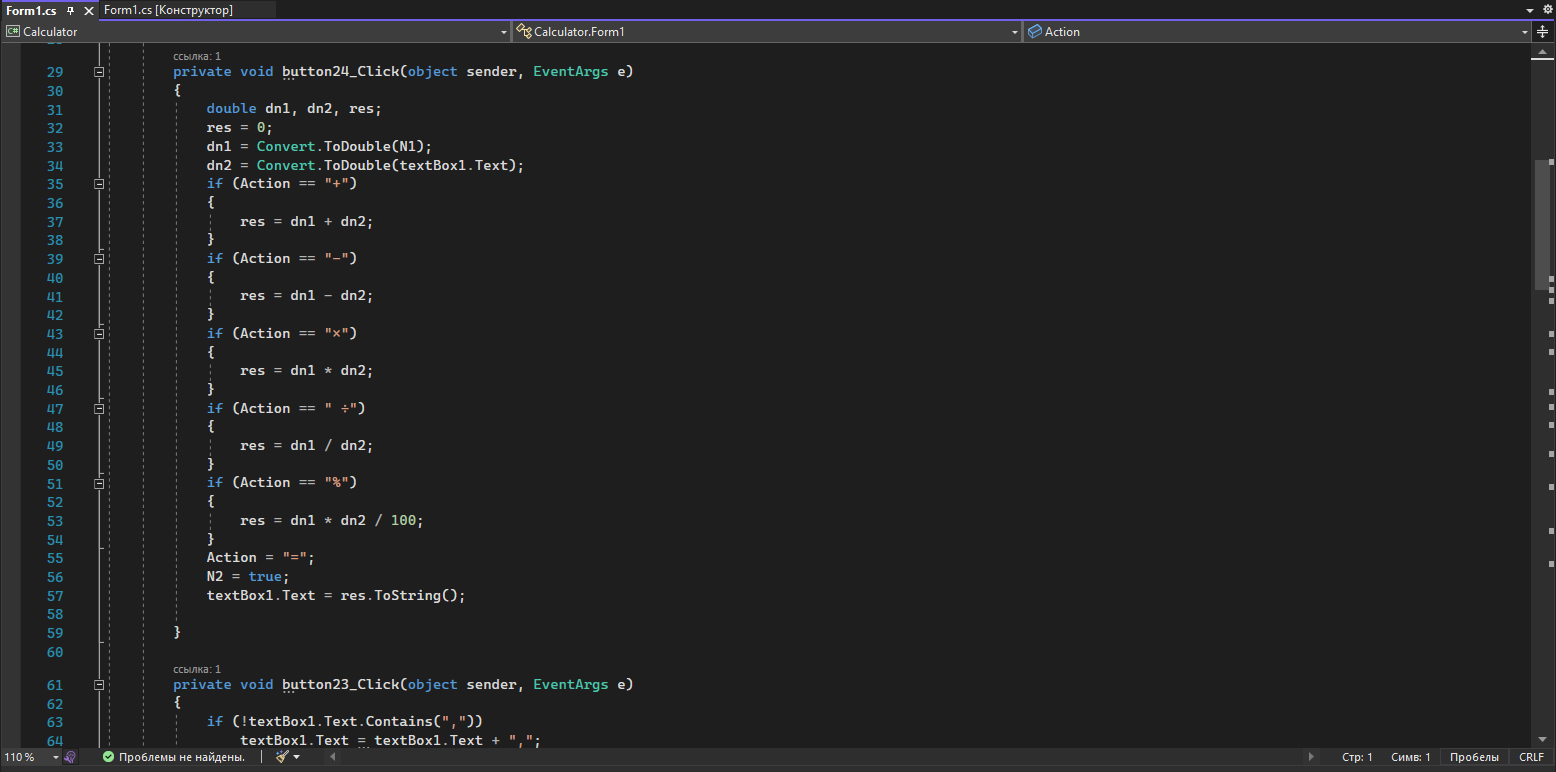
1. С помощью этой же панели добавляем диалоговое окно для ввода чисел.

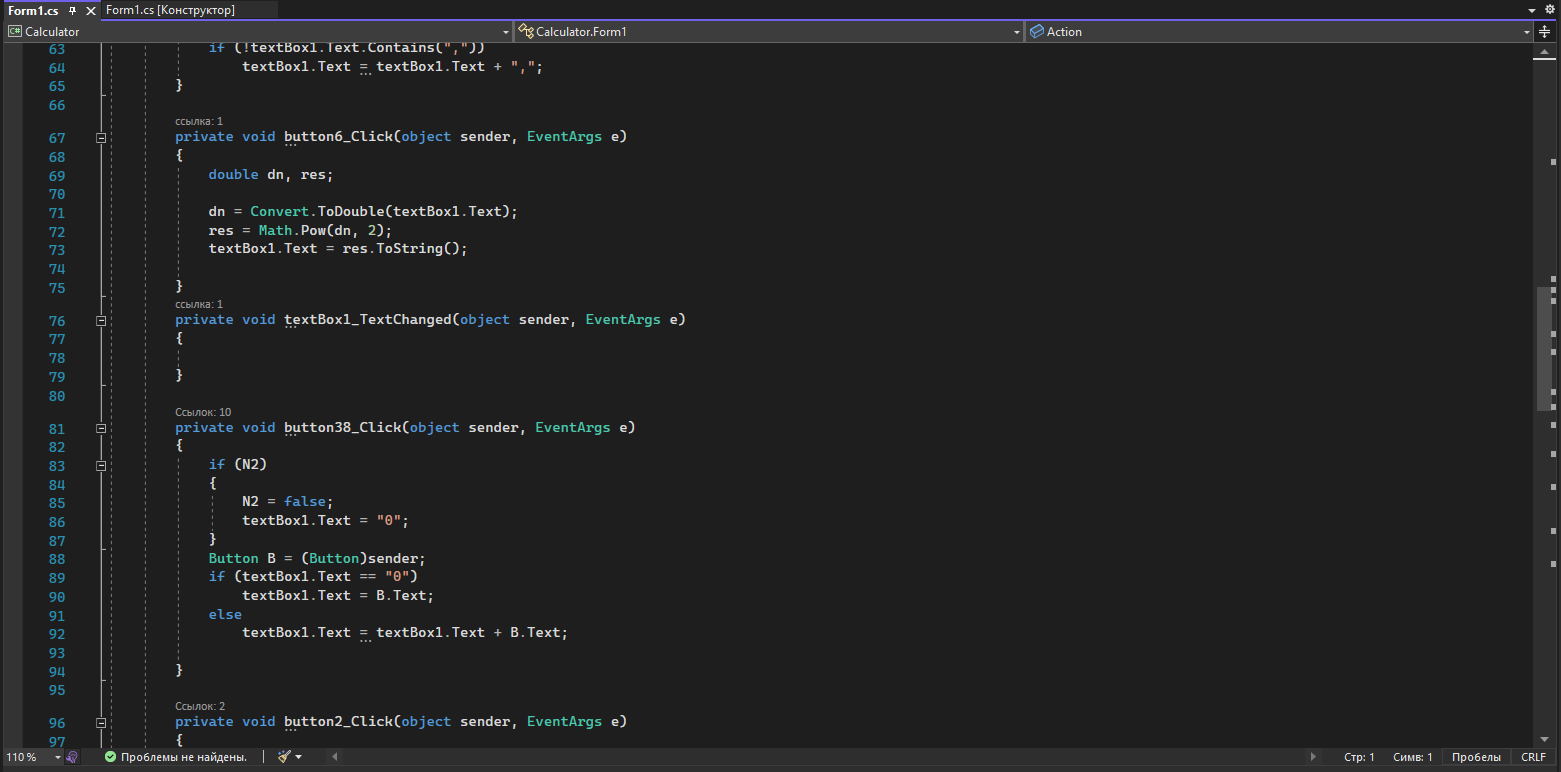


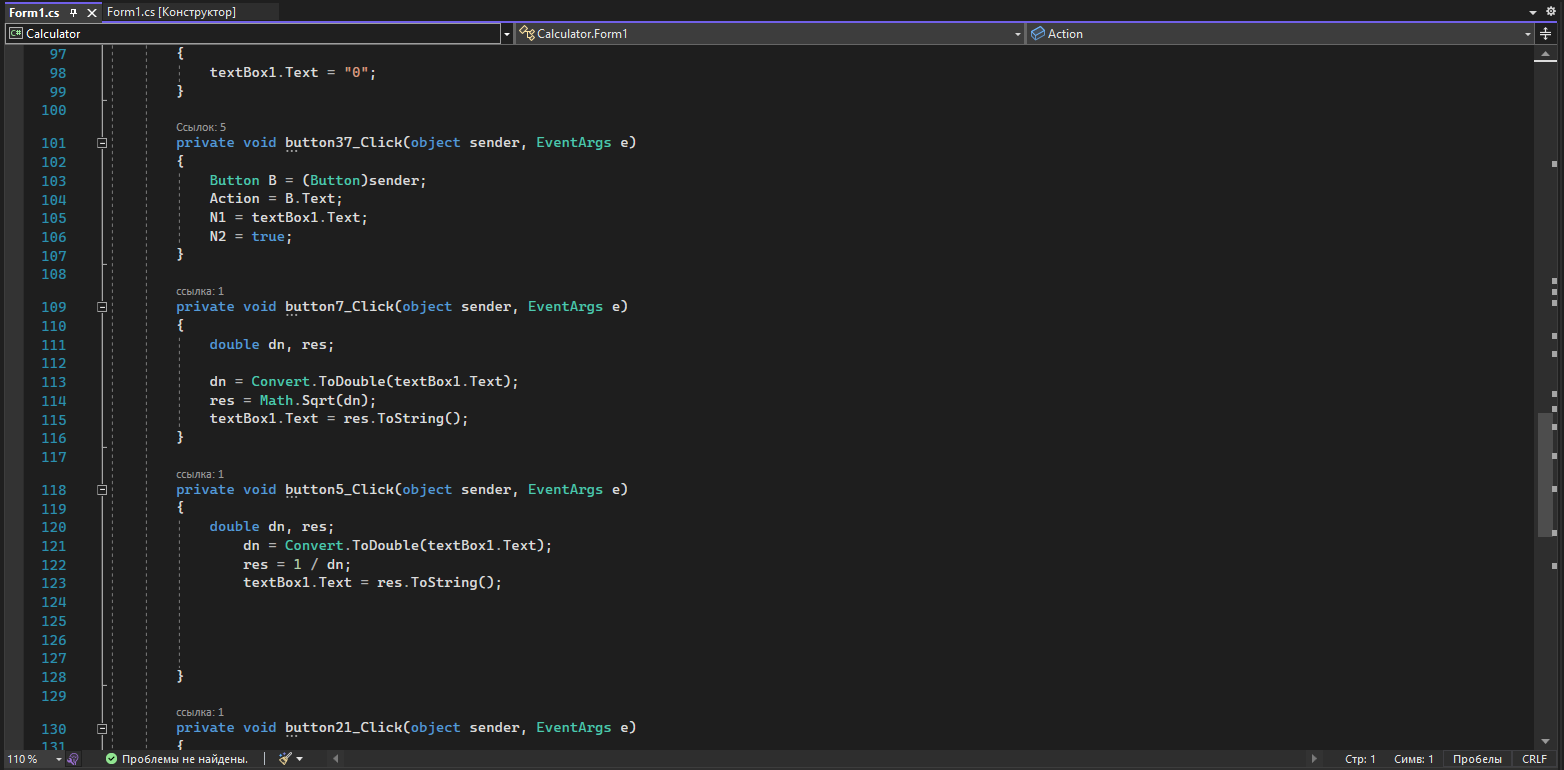
1. В результате получилось окно калькулятора.

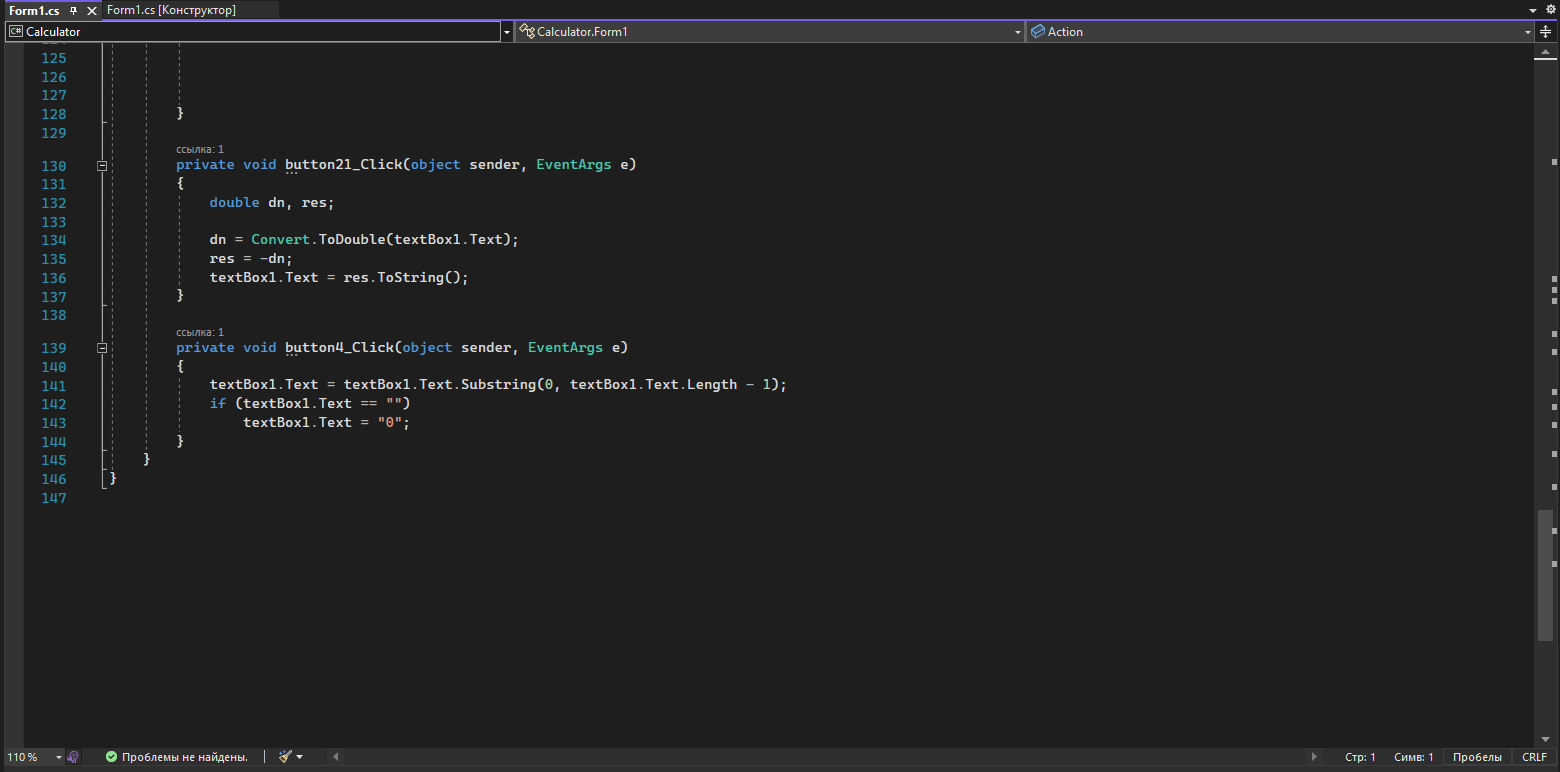


1. Далее программируем кнопки (числа, знаки и символы).

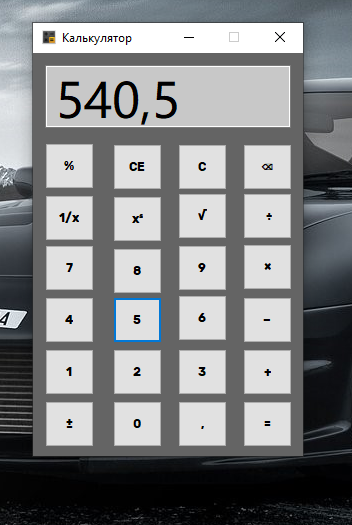








1. В итоге у меня получился следующий калькулятор:



1. Ссылка на мой код: - https://github.com/Semeensr/Practice.git