ПРАКТИЧЕСКАЯ РАБОТА №2

Первый шаг в создании математического теста — это создание проекта приложения Windows Forms.

1. Запустите Visual Studio.
2. В окне запуска выберите **Создание нового проекта**.

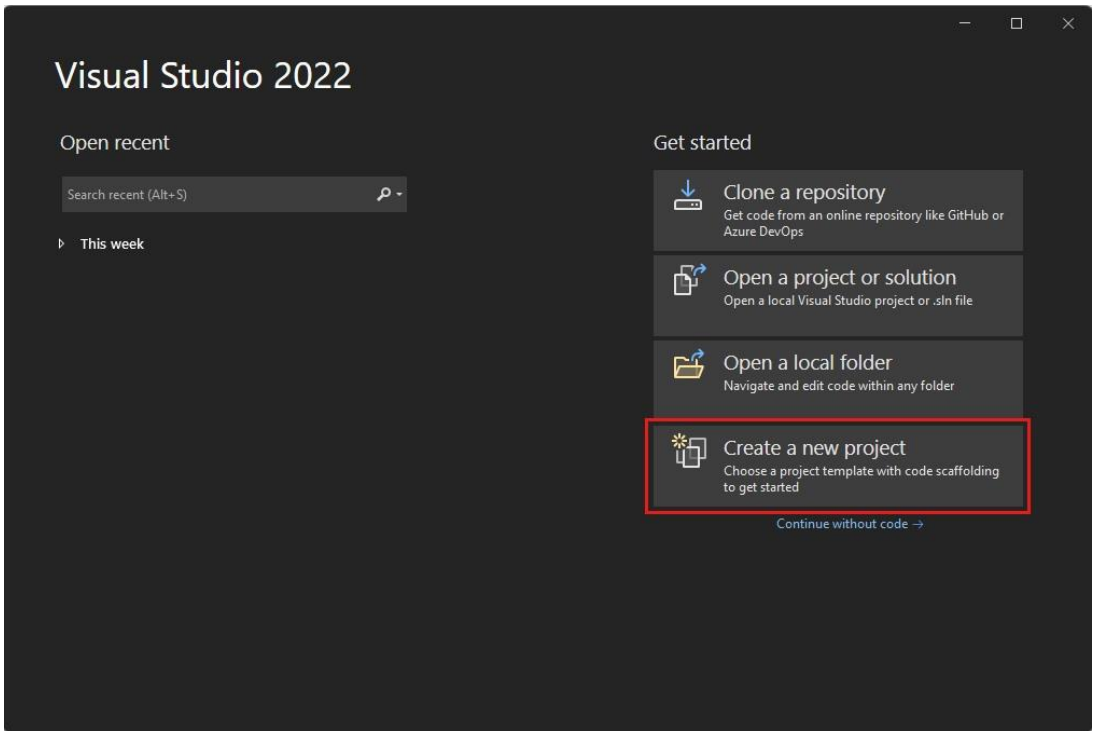

Рисунок №1. Создание проекта

3. В окне **Создать проект** выполните поиск по фразе **Windows Forms**. Затем в списке Тип проекта выберите **Рабочий стол**.

4. Выберите шаблон **Приложение Windows Forms (.NET Framework)** для C# или Visual Basic, а затем нажмите **Далее.**

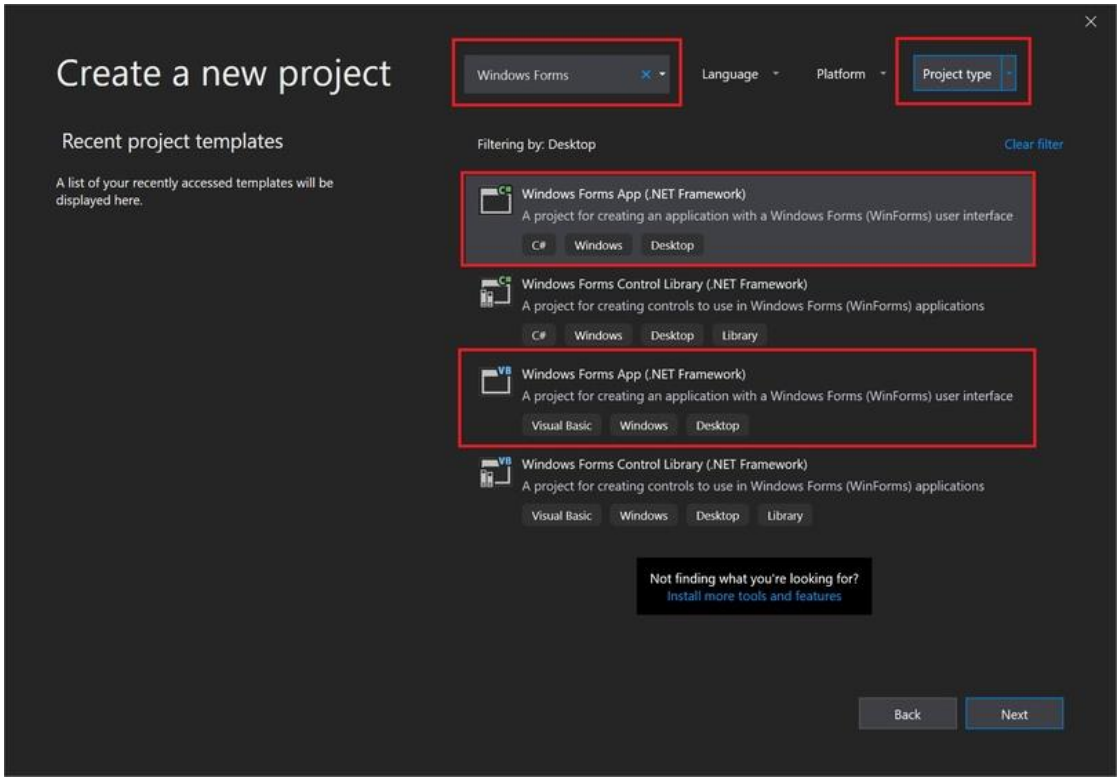

Рисунок №2. Выбор формы

UPD. Если шаблон Приложение Windows Forms (.NET Framework) отсутствует, его можно установить из окна Создание проекта. В сообщении Не нашли то, что искали? выберите ссылку Установка других средств и компонентов.

После этого выберите пункт Разработка классических приложений .NET в Visual Studio Installer.

Рисунок 2.1. Выбор .NET

В Visual Studio Installer выберите Изменить. Вам может быть предложено сохранить результаты работы. Выберите Продолжить, чтобы установить рабочую нагрузку.

5. В окне Настроить новый проект назовите проект MathQuiz, а затем выберите Создать.

Visual Studio создает решение для приложения. Решение является контейнером для всех проектов и файлов, необходимых приложению.

Задание свойств формы

Когда вы выберете шаблон и зададите имя файла, Visual Studio откроет форму. В этом разделе показано, как изменить некоторые свойства формы.

1. В проекте выберите конструктор Windows Forms. На вкладке конструктора для C# считывается файл Form1.cs [Design] , а для Visual Basic — Form1.vb [Design] .

2. Выберите форму Form1.

3. В окне Свойства теперь отображаются свойства формы. Это окно обычно находится в правом нижнем углу окна Visual Studio. Если окно Свойства не отображается, выберите **Вид** > **Окно свойств**.

4. В окне **Свойства** найдите свойство **Text**. В зависимости от того, как отсортирован список, может потребоваться прокрутить вниз. Введите значение **Математический тест** для значения **Текст** и нажмите клавишу **ВВОД**. Форма теперь содержит текст "Математический тест" в заголовке окна.

UPD. Свойства можно отображать по категориям или по алфавиту. Для переключения между свойствами в окне Свойства используйте соответствующие кнопки.

5. Измените размер формы на 500 пикселей в ширину и 400 пикселей в высоту. Можно изменить размер формы, перетаскивая ее края или маркер перетаскивания, пока в окне **Свойства** не отобразится верное значение **Размер**. Этот маркер представляет собой небольшой белый квадрат в правом нижнем углу формы. Для изменения размера формы также можно использовать свойство **Size**.

6. Измените значение свойства **FormBorderStyle** на **Fixed3D**, а свойству **MaximizeBox** установите значение **False**. Эти значения не позволят игрокам изменять размеры формы.

Создание поля "Оставшееся время"

Математический тест содержит поле в правом верхнем углу. В этом поле отображается, сколько секунд осталось. В этом разделе показано, как использовать метку для этого поля.

1. В левой части интегрированной среды разработки Visual Studio выберите вкладку **Панель элементов**. Если вы ее не видите, выберите пункт **Представление(вид)** > **Панель элементов** в строке меню или воспользуйтесь комбинацией клавиш **CTRL+ALT+X.**

2. Выберите элемент управления **Label** в окне Панель элементов, а затем перетащите его на форму.

3. В поле **Свойства** задайте следующие свойства метки:

3.1 Задайте для параметра (**Name**) значение **timeLabel**.

3.2 Измените значение свойства **AutoSize** на **False**, чтобы можно было изменить размер поля.

3.3 Измените значение свойства **BorderStyle** на **FixedSingle** для отрисовки линии вокруг поля.

3.4 Задайте для свойства **Size** значение **200, 30**.

3.5 Выберите свойство **Text**, а затем щелкните клавишу **BACKSPACE**, чтобы очистить значение **Text**.

3.6 Щелкните знак плюса (+) рядом со свойством **Font**, а затем установите для свойства **Size** значение **15,75**.

4. Переместите метку в правый верхний угол формы. Используйте синие направляющие линии для размещения элемента управления в форме.

5. Добавьте еще один элемент управления Label из панели элементов и установите для его размера шрифта значение 15,75.

6. Задайте свойству **Text** значение **Оставшееся время**.

7. Переместите метку так, чтобы она находилась левее метки **timeLabel**.

Рисунок 3. Интерфейс программы

Добавление элементов управления для примера на сложение

Первая часть теста — пример на сложение. В этом разделе показано, как использовать метки для отображения этого примера.

1. Добавьте элемент управления **Label** из **области элементов** в форму.

2. В поле **Свойства** задайте следующие свойства **метки**:

2.1 Задайте для текста значение **?** (вопросительный знак).

2.2 Задайте для параметра **AutoSize** значение **False.**

2.3 Задайте для свойства **Size** значение **60, 50**.

2.4 Установите размер **шрифта** равным **18**.

2.5 Установите для свойства **TextAlign** значение **MiddleCenter**.

2.6 Установите для свойства **Location** значение **50, 75**, чтобы поместить элемент управления в нужное место в форме.

2.7 Установите для свойства **(Name)** значение **plusLeftLabel.**

3. В форме выберите созданную метку **plusLeftLabel.** Скопируйте метку, выбрав пункт **Изменить > Копировать** или **CTRL+C**.

4. Вставьте метку в форму три раза, выбрав пункт **Изменить > Вставить** или **CTRL+V** три раза.

5. Разместите три новые метки так, чтобы они располагались в ряд справа от метки **plusLeftLabel**.

6. Установите для свойства **Text** второй метки значение **+** (знак плюса).

7. Установите для свойства (**Name**) третьей метки значение **plusRightLabel.**

8. Установите для свойства **Text** четвертой метки значение **=** (знак равенства).

9. Добавьте элемент управления **NumericUpDown** из области элементов в форму. Подробнее этот вид элементов управления мы рассмотрим позже.

10. В поле **Свойства** задайте следующие свойства метки **NumericUpDown**:

10.1 Установите размер **шрифта** равным **18**.

10.2 Установите **ширину 100**.

10.3 Задайте для параметра (**Name**) значение **sum**.

11. Выровняйте элемент управления **NumericUpDown** по элементам управления **Label** для задачи на сложение.

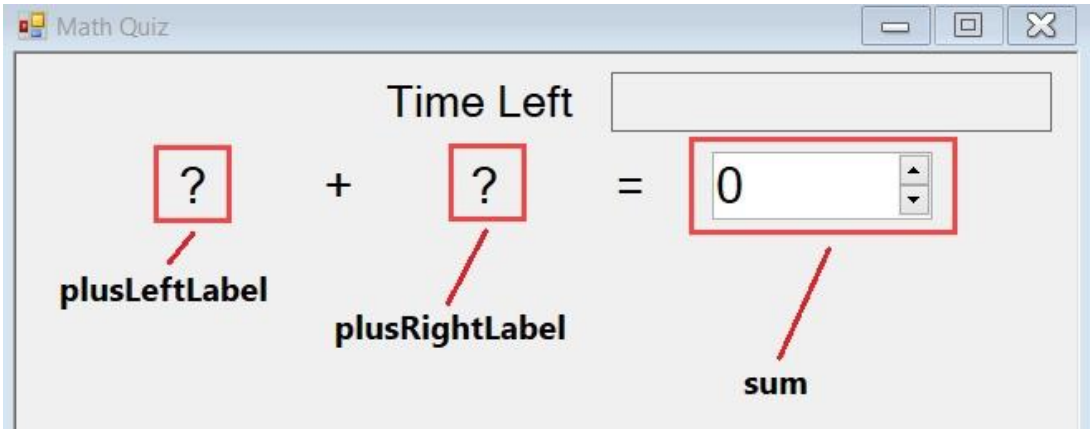

Рисунок 4. Интерфейс программы

Добавление элементов управления для примеров на вычитание, умножение и деление

Затем добавьте метки в форму для оставшихся арифметических примеров.

1. Скопируйте четыре элемента управления **Label** и элемент управления **NumericUpDown**, который был создан для примеров на сложение. Вставьте их в форму.

2. Переместите новые элементы управления на следующую строку под элементами управления для сложения.

3. В поле **Свойства** задайте следующие свойства для новых элементов управления:

3.1 Задайте для свойства (**Name**) первой метки со знаком вопроса значение **minusLeftLabel**.

3.2 Установите для свойства **Text** второй метки значение - (знак минуса).

3.3 Задайте для свойства (**Name**) второй метки со знаком вопроса значение **minusRightLabel**.

3.4 Задайте для свойства (**Name**) элемента управления **the NumericUpDown** значение **difference**.

4. Скопируйте элементы управления для сложения и вставьте их в форму еще два раза.

5. Для третьей строки:

5.1 Задайте для свойства (**Name**) первой метки со знаком вопроса значение **timesLeftLabel.**

5.2 Установите для свойства **Text** второй метки значение × (знак умножения).

5.3 Задайте для свойства (**Name**) второй метки со знаком вопроса значение **timesRightLabel.**

5.4 Задайте для свойства (**Name**) элемента управления **NumericUpDown** значение **product.**

6. Для четвертой строки:

6.1 Задайте для свойства (**Name**) первой метки со знаком вопроса значение **dividedLeftLabel**.

6.2 Установите для свойства **Text** второй метки значение ÷ (знак деления).

6.3 Задайте для свойства (**Name**) второй метки со знаком вопроса значение **dividedRightLabel**.

6.4 Задайте для свойства (**Name**) элемента управления **NumericUpDown** значение **quotient.**

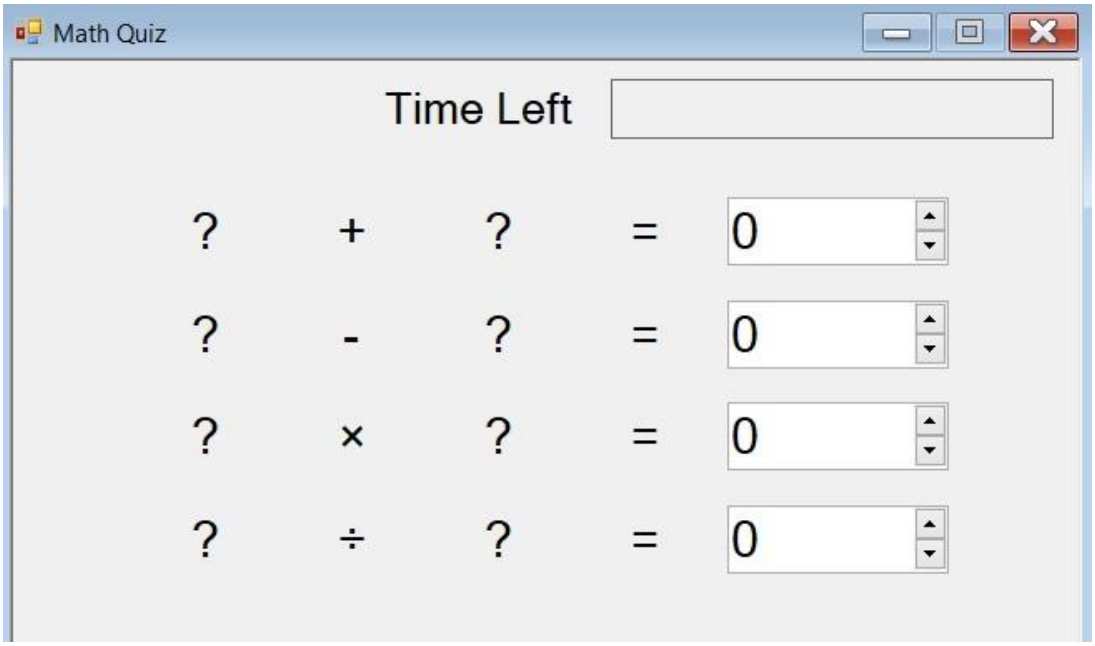

Рисунок 5. Интерфейс программы

ДОБАВЛЕНИЕ КНОПКИ ЗАПУСКА И ЗАДАНИЕ ПОРЯДКА ПЕРЕХОДА ПО КЛАВИШЕ TAB

В этом разделе показано, как добавить кнопку запуска. Также необходимо указать последовательность перехода для элементов управления. Этот порядок определяет, как в тесте осуществляется переход от одного элемента управления к другому с помощью клавиши TAB.

1. Добавьте элемент управления **Button** из области элементов в форму.

2. В поле **Свойства** задайте следующие свойства для элемента управления **Button**:

2.1 Задайте для свойства (**Name**) значение **startButton**.

2.2 Задайте свойству **Text** значение **Начать тест**.

2.3 Установите размер **шрифта** равным **14**.

2.4 Установите для свойства **AutoSize** значение **True**, которое вызывает автоматическое изменение размера кнопки в зависимости от размера текста.

2.5 Установите для **TabIndex** значение **0**. При этом значении кнопка запуска первой появляется в фокусе.

3. Разместите кнопку по центру в нижней части формы.

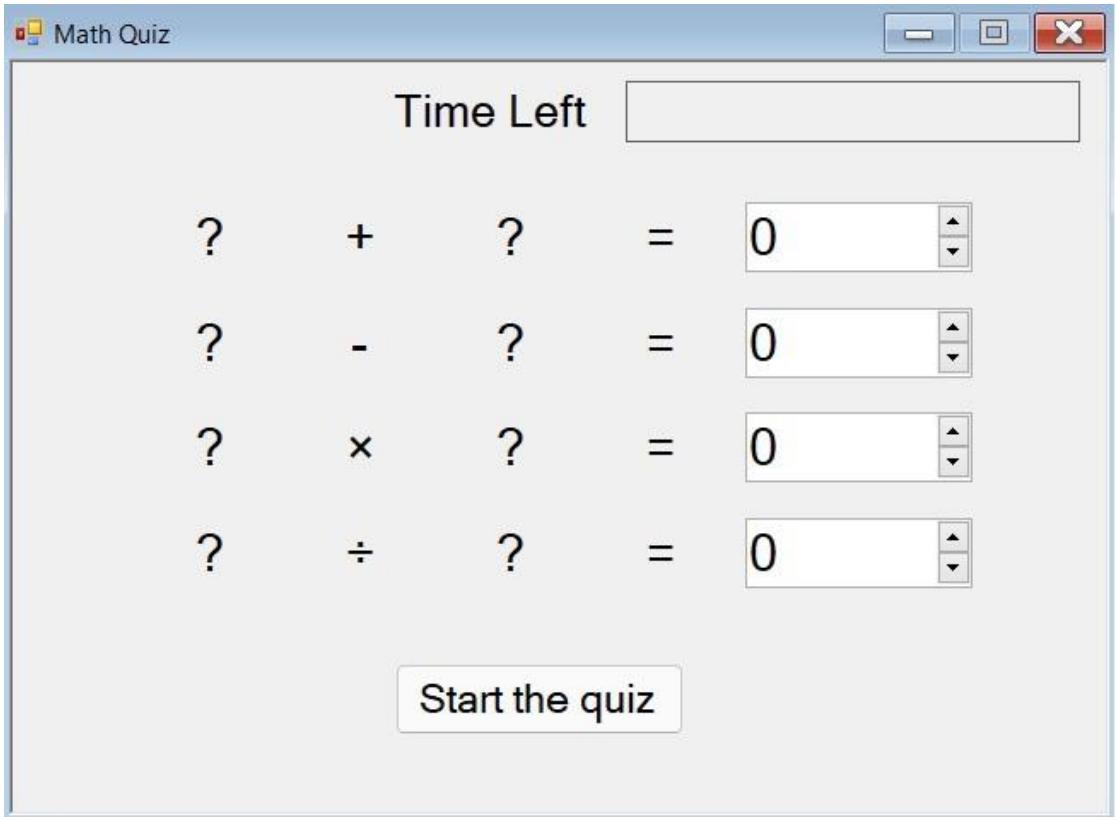

Рисунок 6. Интерфейс программы

1. В поле Свойства задайте свойство **TabIndex** для каждого элемента управления **NumericUpDown**:

1.1 Установите для свойства **TabIndex** элемента управления **NumericUpDown sum** значение **1**.

1.2 Установите для свойства **TabIndex** элемента управления **NumericUpDown difference** значение **2**.

1.3 Установите для свойства **TabIndex** элемента управления **NumericUpDown product** значение 3.

1.4 Установите для свойства **TabIndex** элемента управления **NumericUpDown quotient** значение 4.

ЗАПУСТИТЕ ПРИЛОЖЕНИЕ.

Арифметические примеры пока не работают в тесте. Но вы по-прежнему можете запустить приложение, чтобы проверить, работает ли значение TabIndex должным образом.

1. Для сохранения приложения используйте один из указанных ниже методов:

1.1 Нажмите клавиши **CTRL+SHIFT+S**.

1.2 В строке меню выберите **Файл > Сохранить все**.

1.3 На панели инструментов нажмите кнопку **Сохранить все**.

2. Для запуска приложения используйте один из следующих методов:

2.1 Нажмите клавишу **F5**.

2.2 В строке меню выберите **Отладка > Начать отладку**.

2.3 На панели инструментов нажмите кнопку **Запустить**.

3. Дважды щелкните клавишу **TAB**, чтобы увидеть, как фокус перемещается от одного элемента управления к другому.

СОЗДАНИЕ ЗАДАЧИ НА СЛОЖЕНИЕ СЛУЧАЙНЫХ ЧИСЕЛ

1. В проекте **Visual Studio** выберите конструктор **Windows Forms**.

2. Выберите форму **Form1**.

3. В строке меню выберите **Вид>Код.** Откроется файл **Form1.cs** , отображая код, стоящий за формой.

4. Создайте объект **Random**, добавив инструкцию **new** в начале кода.

```
public partial class Form1 : Form
{
    // Create a Random object called randomizer
    // to generate random numbers.
    Random randomizer = new Random();
```

Рисунок 7. Объект Random

С помощью таких инструкций **new** можно создавать кнопки, метки, панели, **OpenFileDialogs, ColorDialogs, SoundPlayers, Randoms** и даже формы. Эти элементы называются объектами.

При запуске программы запускается форма. Код программной части создает случайный объект и называет его **randomizer**.

Для теста требуются переменные для хранения случайных чисел, создаваемых для каждого арифметического примера. Перед использованием переменных их необходимо объявить, т. е. указать их имена и типы данных.

Добавьте в форму **две целочисленные переменные** и назовите их **addend1** и **addend2.**

Upd. Целочисленная переменная в C# называется int. В переменных этого типа можно хранить положительные и отрицательные числа в диапазоне от -2147483648 до 2147483647, причем это могут быть только целые числа, без десятичных знаков.

Для добавления целочисленной переменной используется синтаксис, похожий на тот, с помощью которого вы добавили объект **Random**, как показано в следующем коде.

```
// Create a Random object called randomizer
// to generate random numbers.
Random randomizer = new Random();
// These integer variables store the numbers
// for the addition problem.
int addend1;
int addend2;
```

Рисунок 8. Integer

Добавьте метод с именем **StartTheQuiz()**. Этот метод использует метод объекта **Random Next()** для создания случайных чисел для меток. **StartTheQuiz()** в конечном итоге заполнит все примеры, а затем запустит таймер, поэтому добавьте эти сведения в общий комментарий. Функция должна выглядеть примерно следующим образом.

```
public void StartTheQuiz()
{
    // Fill in the addition problem.
    // Generate two random numbers to add.
    // Store the values in the variables 'addend1' and 'addend2'.
    addend1 = randomizer.Next(51);
    addend2 = randomizer.Next(51);
    // Convert the two randomly generated numbers
    // into strings so that they can be displayed
    // in the label controls.
    plusLeftLabel.Text = addend1.ToString();
    plusRightLabel.Text = addend2.ToString();
    // 'sum' is the name of the NumericUpDown control.
    // This step makes sure its value is zero before
    // adding any values to it.
    sum.Value = 0;
}
```

Рисунок 9. Code

При использовании метода **Next()** с объектом **Random**, например при вызове **randomizer.Next(51)**, возвращается случайное число меньше 51 (от 0 до 50). Этот код вызывает **randomizer.Next(51**), чтобы два случайных числа складывались и получался ответ от 0 до 100.

Более внимательно ознакомимся с этими операторами.

```
plusLeftLabel.Text = addend1.ToString();
plusRightLabel.Text = addend2.ToString();
```

Рисунок 10. Code

Операторы задают свойства **Text** двух меток — **plusLeftLabel** и **plusRightLabel** — так, чтобы они отображали два случайных числа. Элементы управления **Label** отображают значения в текстовом формате, а в коде строки содержат текст. Каждый метод целого числа **ToString()** преобразует целое число в текст, который может отображаться в метке.

СОЗДАНИЕ СЛУЧАЙНЫХ ПРИМЕРОВ НА ВЫЧИТАНИЕ, УМНОЖЕНИЕ И ДЕЛЕНИЕ

Следующим шагом является объявление переменных и предоставление случайных значений для других арифметических примеров.

1. Добавьте целочисленные переменные для оставшихся примеров в форму после переменных с примером на сложение. Код должен выглядеть так, как показано ниже.

```
public partial class Form1 : Form
{
    // Create a Random object called randomizer
    // to generate random numbers.
    Random randomizer = new Random();
    // These integer variables store the numbers
    // for the addition problem.
    int addend1;
    int addend2;
    // These integer variables store the numbers
    // for the subtraction problem.
    int minuend;
    int subtrahend;
    // These integer variables store the numbers
    // for the multiplication problem.
    int multiplicand;
    int multiplier;
    // These integer variables store the numbers
    // for the division problem.
    int dividend;
    int divisor;
```

Рисунок 11. Code

2. Измените метод **StartTheQuiz(),** добавив следующий код, начиная с комментария "Заполните пример на вычитание".

```
public void StartTheQuiz()
{
    // Fill in the addition problem.
    // Generate two random numbers to add.
    // Store the values in the variables 'addend1' and 'addend2'.
    addend1 = randomizer.Next(51);
    addend2 = randomizer.Next(51);
    // Convert the two randomly generated numbers
    // into strings so that they can be displayed
    // in the label controls.
    plusLeftLabel.Text = addend1.ToString();
    plusRightLabel.Text = addend2.ToString();
    // 'sum' is the name of the NumericUpDown control.
    // This step makes sure its value is zero before
    // adding any values to it.
    sum.Value = 0;
    // Fill in the subtraction problem.
    minuend = randomizer.Next(1, 101);
    subtrahend = randomizer.Next(1, minuend);
    minusLeftLabel.Text = minuend.ToString();
    minusRightLabel.Text = subtrahend.ToString();
    difference.Value = 0;
    // Fill in the multiplication problem.
    multiplicand = randomizer.Next(2, 11);
    multiplier = randomizer.Next(2, 11);
    timesLeftLabel.Text = multiplicand.ToString();
    timesRightLabel.Text = multiplier.ToString();
    product.Value = 0;
    // Fill in the division problem.
    divisor = randomizer.Next(2, 11);
    int temporaryQuotient = randomizer.Next(2, 11);
    dividend = divisor * temporaryQuotient;
    dividedLeftLabel.Text = dividend.ToString();
    dividedRightLabel.Text = divisor.ToString();
    quotient.Value = 0;
```

Рисунок 11. Code

Upd. Комментарии не надо переписывать (все что //)

В этом коде метод **Next()** класса **Random** используется несколько иначе, чем в примере на сложение. Когда методу **Next()** передается два значения, он выбирает случайное число, которое больше первого значения или равно ему и меньше второго значения.

С помощью метода **Next()** с двумя аргументами можно убедиться, что у примера на вычитания положительный ответ, ответ в примере на умножение не превышает 100, а ответ в примере на деление не является дробным.

ДОБАВЛЕНИЕ ОБРАБОТЧИКА СОБЫТИЙ ДЛЯ КНОПКИ ЗАПУСКА

В этом разделе вы добавите код для запуска теста при выборе кнопки запуска. Код, который выполняется в ответ на событие, например нажатие кнопки, называется обработчиком событий.

1. В конструкторе **Windows Forms** дважды щелкните кнопку **Начать тест** или выберите ее, а затем нажмите клавишу **ВВОД**. Отобразится код формы и новый метод. Эти действия добавляют обработчик событий **Click** к кнопке запуска. Когда игрок нажмет на эту кнопку, приложение запустит код, который будет добавлен в этот новый метод.

2. Добавьте следующие два оператора, чтобы обработчик событий начал тест.

```
private void startButton_Click(object sender, EventArgs e)
{
    StartTheQuiz();
    startButton.Enabled = false;
}
```

Рисунок 12. Code

Первый оператор вызывает новый метод **StartTheQuiz()**. Второй оператор устанавливает свойству **Enabled** элемента управления **startButton** значение **false**, чтобы игрок не мог нажать кнопку в процессе теста.

ЗАПУСТИТЕ ПРИЛОЖЕНИЕ.

1. Сохраните код.

2. Запустите приложение и выберите **Начать тест**. Отображаются случайные арифметические примеры, как показано на следующем снимке экрана.

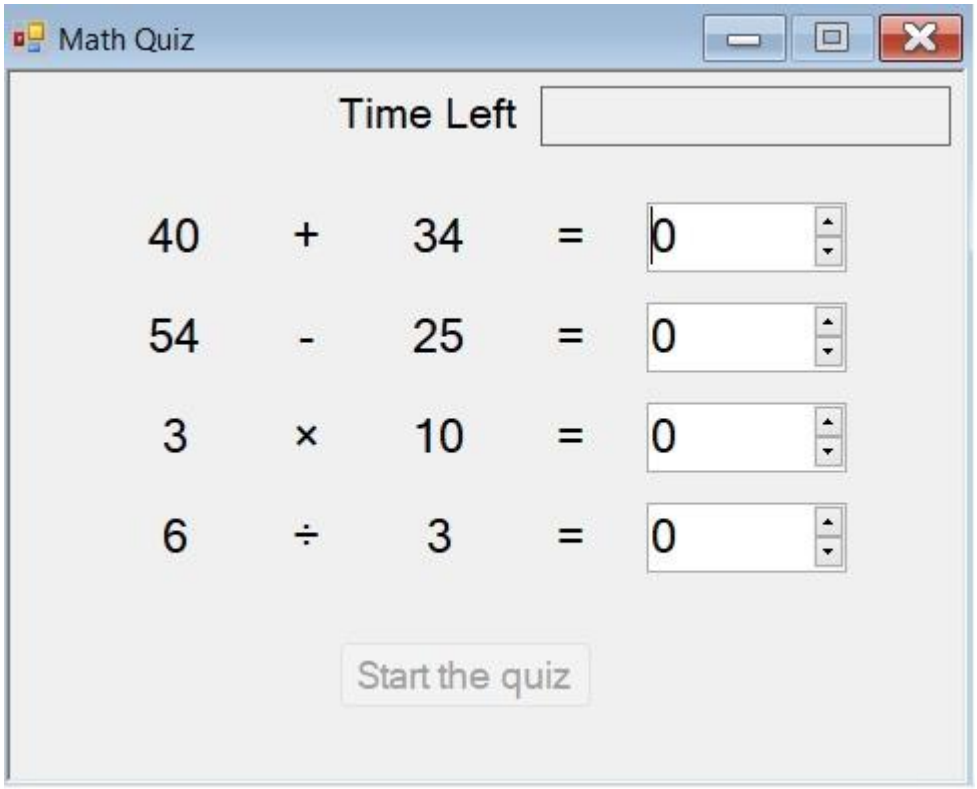

Рисунок 13. Интерфейс программы

Добавление элемента управления Timer в приложение WinForms с математическим тестом

Для наблюдения за временем в ходе теста используется компонент "Таймер". Кроме того, требуется переменная для хранения оставшегося времени.

1. Добавьте целочисленную переменную с именем **timeLeft** так же, как вы объявляли переменные в предыдущем. Разместите объявление **timeLeft** сразу после других объявлений. Код должен выглядеть так, как показано ниже.

```
public partial class Form1 : Form
{
    // Create a Random object called randomizer
    // to generate random numbers.
    Random randomizer = new Random();
    // These integer variables store the numbers
    // for the addition problem.
    int addend1;
    int addend2;
    // These integer variables store the numbers
    // for the subtraction problem.
    int minuend;
    int subtrahend;
    // These integer variables store the numbers
    // for the multiplication problem.
    int multiplicand;
    int multiplier;
    // These integer variables store the numbers
    // for the division problem.
    int dividend;
    int divisor;
    // This integer variable keeps track of the
    // remaining time.
    int timeLeft;
```

Рисунок 14. Интерфейс программы

Upd. Комментарии не надо переписывать (все что //)

В конструкторе **Windows Forms** переместите элемент управления **Timer** из категории **Компоненты** на панели элементов в форму. Элемент управления появляется в серой области в нижней части окна конструктора.

Щелкните в форме только что добавленный значок **timer1** и установите его свойство **Interval** равным **1000**. Поскольку этот интервал измеряется в миллисекундах, при значении 1000 таймер создает событие **Tick** каждую **секунду**.

ПРОВЕРКА ОТВЕТОВ

Поскольку таймер создает событие **Tick** каждую секунду, имеет смысл проверять истекшее время в обработчике событий **Tick**. Также целесообразно

проверять ответы в этом обработчике событий. Если время истекло или ответы указаны правильно, тест должен завершиться.

Перед написанием этого обработчика событий добавьте метод **CheckTheAnswer()**, чтобы определить, верны ли ответы на арифметические примеры. Этот метод должен располагаться в строке с другими методами, например **StartTheQuiz().** Код должен выглядеть так, как показано ниже.

```
private bool CheckTheAnswer()
{
    if ((addend1 + addend2 == sum.Value)
        && (minuend - subtrahend == difference.Value)
        && (multiplicand * multiplier == product.Value)
        && (dividend / divisor == quotient.Value))
        return true;
    else
        return false;
}
```

Рисунок 15. Code

Этот метод определяет ответы на арифметические примеры и сравнивает результаты со значениями в элементах управления **NumericUpDown**. В этом коде:

- Версия на **Visual Basic** использует ключевое слово **Function** вместо обычного ключевого слова **Sub**, потому что этот метод возвращает значение.
- Так как простого способа ввести знак умножения (×) и знак деления (÷) с клавиатуры нет, в языках **C#** и **Visual Basic** используется звездочка (*) для умножения и косая черта (/) для деления.
- В **C# &&** — это **оператор logical and.** Эквивалентный оператор в языке **Visual Basic — AndAlso**. Оператор **logical and** используется для проверки того, **имеет ли значение true** более одного условия. В этом случае, если все значения верны, метод возвращает **значение true.** В противном случае метод возвращает **значение false.**
- **Инструкция if** использует свойство **Value** элемента управления **NumericUpDown** для доступа к текущему значению элемента управления. В

следующем разделе вы используете то же свойство для вывода правильного ответа в каждом элементе управления.

ДОБАВЛЕНИЕ ОБРАБОТЧИКА СОБЫТИЙ В ТАЙМЕР

Теперь, когда у вас есть способ проверить ответы, можно написать код для обработчика событий **Tick**. Этот код выполняется каждую секунду после того, как таймер создаст событие **Tick**. Этот обработчик событий проверяет ответы игрока, вызывая метод **CheckTheAnswer()**. Он также проверяет, сколько времени теста уже истекло.

1. Двойным щелчком выберите в форме элемент управления **Timer** либо выделите его и нажмите клавишу **ВВОД**. Эти действия добавляют обработчик событий **Tick** к таймеру. Откроется редактор кода, в котором отобразится метод обработчика **Tick**.

2. Добавьте в новый метод обработчика событий следующие операторы.

```
private void timer1_Tick(object sender, EventArgs e)
{
    if (CheckTheAnswer())
    {
        // If CheckTheAnswer() returns true, then the user
        // got the answer right. Stop the timer
        // and show a MessageBox.
        timer1.Stop();
        MessageBox.Show("You got all the answers right!",
                        "Congratulations!");
        startButton.Enabled = true;
    }
    else if (timeLeft > 0)
    {
        // If CheckTheAnswer() returns false, keep counting
        // down. Decrease the time left by one second and
        // display the new time left by updating the
        // Time Left label.
        timeLeft = timeLeft - 1;
        timeLabel.Text = timeLeft + " seconds";
    }
    else
    {
        // If the user ran out of time, stop the timer, show
        // a MessageBox, and fill in the answers.
        timer1.Stop();
        timeLabel.Text = "Time's up!";
        MessageBox.Show("You didn't finish in time.", "Sorry!");
        sum.Value = addend1 + addend2;
        difference.Value = minuend - subtrahend;
        product.Value = multiplicand * multiplier;
        quotient.Value = dividend / divisor;
        startButton.Enabled = true;
    }
}
```

Рисунок 16. Code

Upd. Комментарии не надо переписывать (все что //)

Этот метод выполняется каждую секунду теста. Код сначала проверяет значение, которое возвращает **CheckTheAnswer().**

- Если все ответы верны, это значение равно **true**, и тест завершается:
  - Таймер останавливается.
  - Появится поздравительное сообщение.
  - Свойству **Enabled** элемента управления **startButton** устанавливается значение **true**, чтобы игрок мог заново запустить тест.

- Если **CheckTheAnswer()** возвращает **false**, код проверяет значение **timeLeft:**
    - Если эта переменная больше **0**, таймер вычитает **1** из **timeLeft**. Затем он обновляет свойство Text элемента управления **timeLabel,** чтобы показать игроку, сколько осталось секунд.
    - Если времени не остается, таймер останавливается и изменяет текст **timeLabel** на **time's up!** В окне сообщения будет объявлено, что тест закончен, и появятся ответы. Кнопка старта снова станет доступной.

ЗАПУСК ТАЙМЕРА

Чтобы запустить таймер при запуске теста, добавьте в конец метода StartTheQuiz() три строки, как показано в следующем примере.

```
// Start the timer.
timeLeft = 30;
timeLabel.Text = "30 seconds";
timer1.Start();
```

Рисунок 17. Code

Теперь при запуске теста переменная **timeLeft** устанавливается в значение 30, а свойство Text элемента управления **timeLabel** — 30 секунд. После этого метод **Start()** элемента управления **Timer** начинает обратный отсчет.

ЗАПУСТИТЕ ПРИЛОЖЕНИЕ.

1. Сохраните и выполните программу.

2. Нажмите **Начать тест**. Таймер начинает обратный отсчет. Когда время истечет, тест закончится и появятся ответы.

3. Запустите еще один тест и предоставьте правильные ответы на арифметические примеры. При правильном ответе в течение отведенного времени откроется окно сообщения, кнопка запуска станет доступной, а таймер остановится.

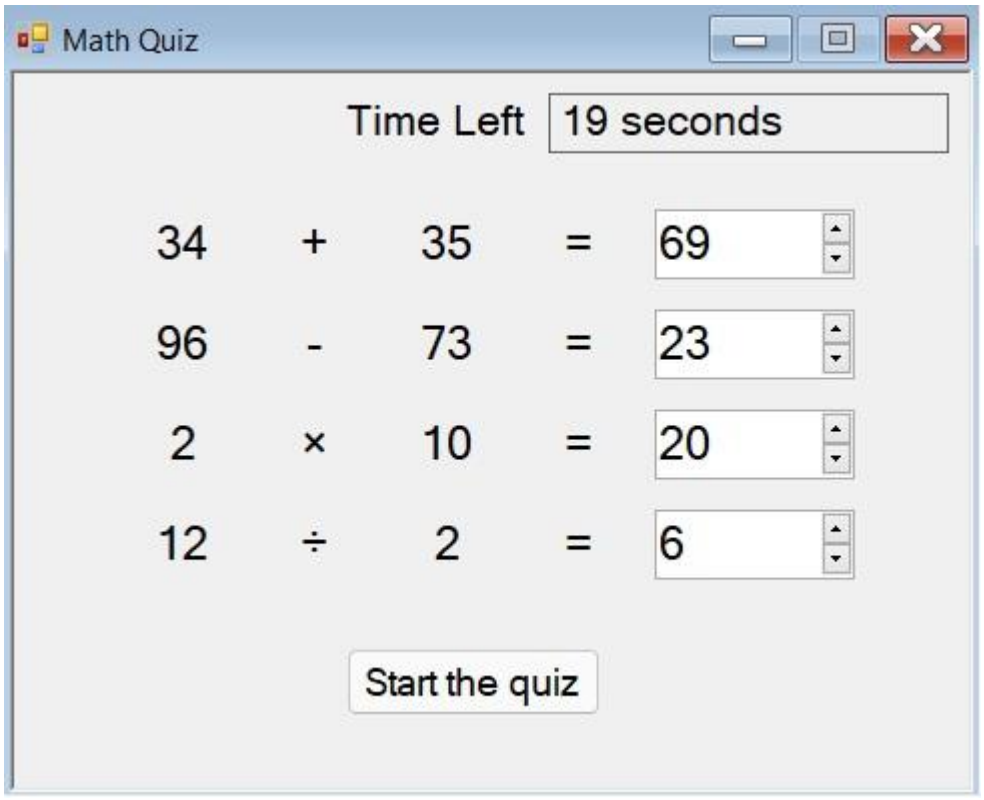

Рисунок 18. Code

ДОБАВЛЕНИЕ ОБРАБОТЧИКОВ СОБЫТИЙ ДЛЯ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ NUMERICUPDOWN

Тест содержит элементы управления **NumericUpDown**, с помощью которых игрок вводит числа. При вводе ответа необходимо либо выбрать значение по умолчанию, либо удалить это значение вручную. Добавив обработчик событий Enter, можно упростить ввод ответов. Этот код будет выделять и удалять текущее значение в каждом элементе управления **NumericUpDown**, как только игрок выберет элемент управления и начнет вводить другое значение.

1. Выберите первый элемент управления **NumericUpDown** в форме. В диалоговом окне Свойства выберите значок События на панели инструментов.

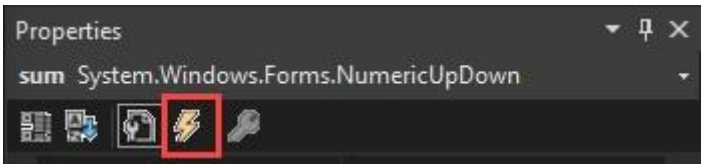

Рисунок 19. Обработчик

На вкладке **События** в диалоговом окне **Свойства** отображаются все события для выбранного в форме элемента, на которые можно реагировать. В этом случае все перечисленные события относятся к элементу управления **NumericUpDown.**

2. Выберите событие **Enter**, введите **answer_Enter** и нажмите клавишу **ВВОД.**

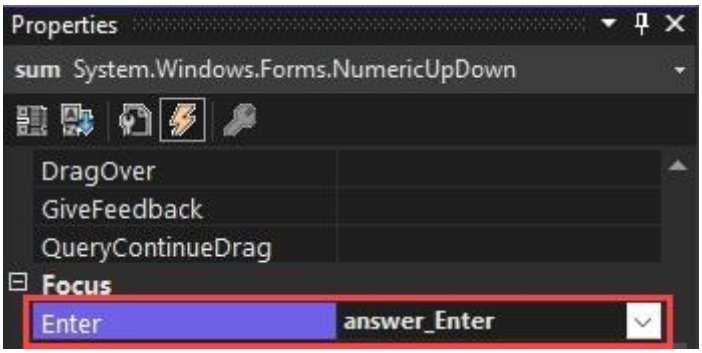

Рисунок 20. Обработчик

Откроется редактор кода, в котором отобразится обработчик события Enter, созданный для элемента управления **NumericUpDown sum.**

3. В методе для обработчика событий **answer_Enter** введите следующий код:

```
private void answer_Enter(object sender, EventArgs e)
{
    // Select the whole answer in the NumericUpDown control.
    NumericUpDown answerBox = sender as NumericUpDown;
    if (answerBox != null)
    {
        int lengthOfAnswer = answerBox.Value.ToString().Length;
        answerBox.Select(0, lengthOfAnswer);
    }
}
```

Рисунок 21. Code

В этом коде:

•В первой строке объявляется метод. Он содержит параметр с именем **sender**. В C# параметр имеет значение **object sender.** Этот параметр ссылается на объект, событие которого срабатывает. Он называется отправителем. В данном случае объектом-отправителем является элемент управления **NumericUpDown.**

•Первая строка внутри метода приводит, или преобразует, отправителя из универсального объекта в элемент управления **NumericUpDown**. Эта строка также назначает имя **answerBox** элементу управления **NumericUpDown.** Все элементы управления **NumericUpDown** в форме будут использовать этот метод, а не только элемент управления примера на сложение.

•В следующей строке кода выполняется проверка, что **answerBox** был успешно приведен как элемент управления **NumericUpDown.**

•Первая строка внутри инструкции **if** определяет длину ответа, который в данный момент находится в элементе управления **NumericUpDown.**

•Во второй строке внутри инструкции **if** используется длина ответа для выбора текущего значения в элементе управления.

Когда игрок выберет элемент управления, Visual Studio запускает это событие. Этот код выбирает текущий ответ. Как только игрок начинает вводить другой ответ, текущий ответ удаляется и заменяется новым.

1. В конструкторе **Windows Forms** выберите элемент управления **NumericUpDown** для примера на вычитания.

2. На странице События диалогового окна Свойства найдите событие Enter, а затем в раскрывающемся меню выберите **answer_Enter**. Это обработчик событий, который вы только что добавили.

3. Повторите предыдущие два шага для элементов управления **NumericUpDown** для умножения и деления.

ЗАПУСТИТЕ ПРИЛОЖЕНИЕ.

1. Сохраните и выполните программу.

2. Запустите тест и выберите элемент управления **NumericUpDown**. Существующее значение автоматически выделяется и удаляется, когда вы начинаете вводить другое значение.

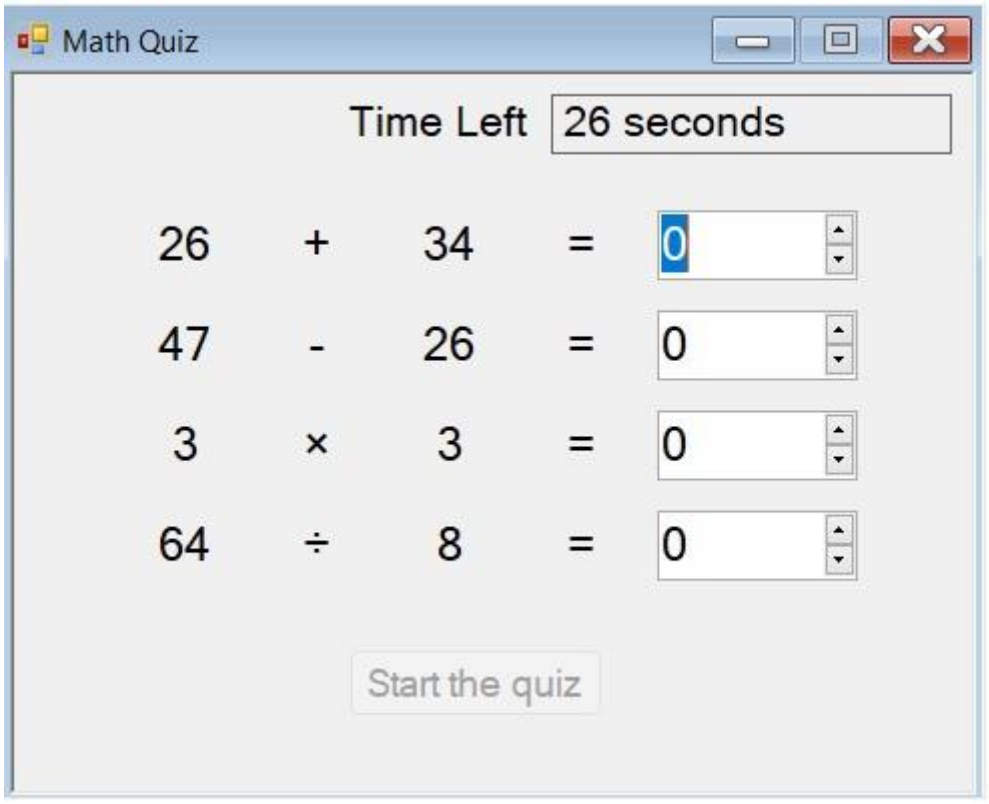

Рисунок 22. Интерфейс

НАСТРОЙКА ТЕСТА

В последней части этого руководства рассматривается несколько способов настройки теста и приводятся дополнительные сведения по изученным вопросам.

Изменение цвета метки

Когда для прохождения теста останется лишь пять секунд, измените цвет элемента управления timeLabel на красный путем задания его свойства BackColor.

```
timeLabel.BackColor = Color.Red;
```

Рисунок 23. Code

Восстановите цвет при завершении игры.

ВОСПРОИЗВЕДЕНИЕ ЗВУКА ПРАВИЛЬНОГО ОТВЕТА (САМОСТОЯТЕЛЬНО)

Дайте игроку подсказку с помощью воспроизведения звука при вводе правильного ответа в элементе управления **NumericUpDown.** Для реализации этой функции напишите обработчик событий для каждого события элемента управления **ValueChanged.** Этот тип события срабатывает всякий раз, когда игрок изменяет значение элемента управления.

В конце не забудьте отправить готовый (рабочий) проект, в свой репозиторий гитхаба (вы с ним уже разбирались), для этой практической работы создайте отдельную ветку назовите которую «Pr_2», проверьте отправился ли ваш проект, и только тогда проверю вашу работу.