ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ АРХАНГЕЛЬСКОЙ ОБЛАСТИ
«МИРНИНСКИЙ ПРОМЫШЛЕННО-ЭКОНОМИЧЕСКИЙ ТЕХНИКУМ»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКИХ РАБОТ ПО

МДК.02.04 ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ

ПМ.02 ПРИМЕНЕНИЕ МИКРОПРОЦЕССОРНЫХ СИСТЕМ, УСТАНОВКА И НАСТРОЙКА ПЕРИФЕРИЙНОГО ОБОРУДОВАНИЯ

для специальности: 09.02.01 Компьютерные системы и комплексы

2022 г.

Методические рекомендации для МДК.02.04 Программирование на языке высокого уровня разработаны для выполнения практических работ в среде программирования C# и составлены в соответствии с рабочей программой МДК и учебным планом по специальности 09.02.01 «Компьютерные системы и комплексы».

Организация-разработчик: государственное бюджетное профессиональное образовательное учреждение Архангельской области «Мирнинский промышленно-экономический техникум»

Разработчики:
Кузнецова С.П., заведующий дневным отделением;

ОДОБРЕНЫ цикловой комиссией дисциплин специальностей 09.02.01 и 13.02.11	Составлены в соответствии с требованиями ФГОС по специальности среднего профессионального образования 09.02.01 «Компьютерные системы и комплексы» и учебным планом
Председатель цикловой комиссии В.И.Письменник	Заместитель директора техникума по учебной работе М.Н.Венедиктова

СОДЕРЖАНИЕ

1	Создание проекта приложения Windows Forms	3
2	Создание лабиринта	23
3	Создание математической викторины	42
4	Создание игры "Подбери пару!"	64

ПРАКТИЧЕСКАЯ РАБОТА № 1

Тема работы: Создание проекта приложения Windows Forms
Цель работы: разработка приложения для реализации работы с формой

Ход работы:
Первый шаг в создании программы для просмотра изображений это создание проекта приложения Windows Forms.
Создание проекта приложения Windows Forms
1. В меню **Файл** выберите команду **Создать проект**.
2. Если используется не Visual Studio Express, вначале необходимо выбрать язык. В списке**Установленные шаблоны** выберите **C#** или **Visual Basic**.
3. Щелкните значок **Приложение Windows Forms**, а затем введите в качестве имени PictureViewer, затем нажмите кнопку **OK**. Среда Visual Studio автоматически создает решение.
4. В меню **Файл** выберите пункт **Сохранить все**, либо на панели инструментов нажмите кнопку**Сохранить все**, которая показана на рисунке ниже.

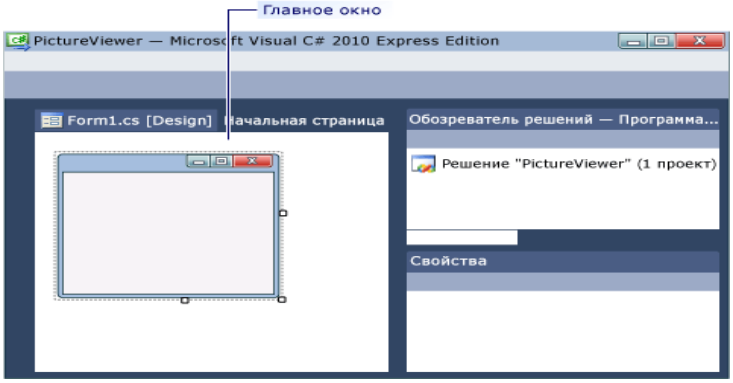
Кнопка панели инструментов "Сохранить все" 

Примечание
Среда Visual Studio сохраняет проект в папке проекта. Интегрированная среда разработки автоматически заполняет имя папки и имя проекта. При использовании среды Visual Studio Express необходимо выполнить шаги 5-7. Для других версий Visual Studio (не Express) проект сохраняется при его первоначальном создании, поэтому шаги 5-7 необязательны.

5. Убедитесь, что проект сохранен во вложенной папке в папке **Мои документы**.
6. Проверьте, что установлен флажок **Создать каталог для решения**.
7. Нажмите кнопку **Сохранить**.

Примечание
При создании проекта интегрированная среда разработки Visual Studio автоматически создает несколько файлов и размещает их в папке. Эти файлы можно изучить с помощью окна **Обозреватель решений**. При первоначальном создании проекта файлы сохраняются во временной папке. В результате нажатия кнопки **Сохранить все** среда интегрированной разработки копирует их в постоянную папку (которая обычно расположена в папке **Мои документы**).

8. Возможно вы уже обратили внимание, что слово *решение* и слово *проект* имеют различные значения, которые объясняются в этом руководстве позднее.
9. На рисунке ниже показано, как должно выглядеть окно интегрированной среды разработки.
10. Окно интегрированной среды разработки



11. Если ваше окно не выглядит так, как показано на рисунке выше, в меню **Окно** выберите пункт**Сброс макета окон**. Если какое-то из окон отсутствует, в меню **Вид** выберите пункт **Окно свойств** или **Обозреватель решений**. Если есть другое открытое окно, нажмите кнопку**Закрыть** (x) в правом верхнем углу.
12. Посмотрите на рисунок. От левого верхнего угла и по часовой стрелки на рисунке показано следующее.

- **Главное окно**. В этом окне выполняется большая часть работы. В нем выполняется работа с формами и редактирование кода. Сейчас в нем представлена форма в редакторе форм. В верхней части окна расположены две вкладки — вкладка **Домашняя страница** и вкладка **Form1.cs [Design]** (в Visual Basic вместо расширения .cs)
- Окно **Обозреватель решений**. В этом окне отображаются все файлы решения. Если выделить файл, изменяются сведения в окне **Свойства**. Если дважды щелкнуть файл кода (с расширением .cs в Visual C#), открывается файл кода или конструктор для файла кода.
- Окно **Свойства**. В этом окне выполняется изменение свойств элементов, которые выделены в других окнах.

> **Примечание**
>
> Обратите внимание, что в верхней строке окна **Обозреватель решений** отображена надпись**Решение "PictureViewer" (1 проект)**. Интегрированная среда разработки создала решение, решение может содержать несколько проектов. Сейчас будет выполняться работа с решениями, которые содержат один проект.

Запуск программы

При создании нового решения фактически выполняется построение программы, которая в последующем запускается. Она пока ничего не делает — просто открывает пустое окно, у которого в строке заголовка окна надпись **Form1**. Но, как очевидно, она выполняется.

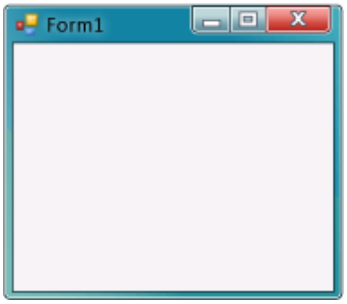
Для запуска программы

1. Нажмите клавишу F5 или нажмите кнопку панели инструментов **Начать отладку**, которая показана на рисунке ниже.

Кнопка панели инструментов "Начать отладку"

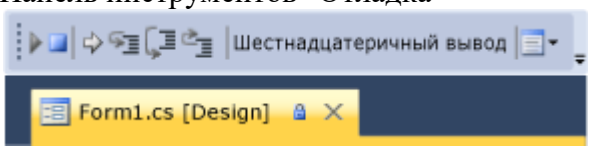
2. Интегрированная среда разработки запускает программу и открывается окно. На рисунке ниже показана программа, для которой только что было выполнено построение. Программа выполняется и скоро она будет дополнена.

Выполнение программы приложения Windows Form



3. Вернитесь в интегрированную среду разработки и посмотрите на новую панель инструментов.

Панель инструментов "Отладка"



4. Нажмите квадратную кнопку **Остановить отладку** или в меню **Отладка** выберите пункт**Остановить отладку**. Программа останавливает выполнение, окно закрывается. Для остановки отладки можно просто закрыть окно.

> **Примечание**
>
> Когда производится запуск программы внутри интегрированной среды разработки, это называется *отладкой*, так как это обычно выполняется для отслеживания и исправления ошибок. Это настоящая программа, ее можно выполнить также, как любую другую программу.

Настройка свойств формы

Далее окно **Свойства** используется для изменения внешнего вида формы.

Настройка свойств формы

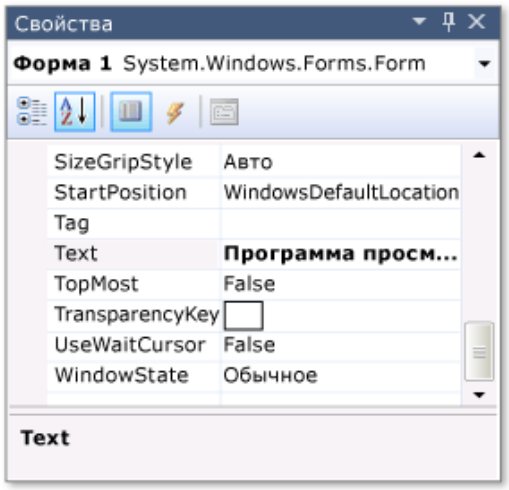
1. Убедитесь, что вы смотрите на конструктор Windows Forms. В интегрированной среде разработки перейдите на вкладку **Form1.cs [Design]**

2. Чтобы выделите форму, щелкните в любом ее месте. Посмотрите на окно **Свойства**. Теперь в нем должны отображаться свойства формы. У формы есть различные свойства. Например, можно установить цвет переднего плана и фона, текст заголовка, который отображается в верхней части формы, размер формы и другие свойства.

> **Примечание**
>
> Если окно **Свойства** не открылось, остановите программу. Для этого нажмите квадратную кнопку **Остановить отладку** (или просто закройте окно).

3. После того как форма выделена, прокрутите до конца содержимое окна **Свойства** и найдите свойство **Text**. Выделите **Text**, введите "Программа просмотра изображений", затем нажмите клавишу ВВОД. Теперь форма в заголовке окна должна содержать текст**Программа просмотра изображений**. Окно **Свойства** должно выглядеть так, как показано на рисунке ниже.

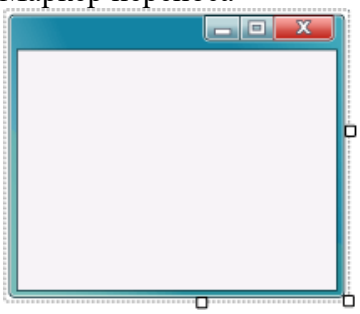
Окно "Свойства"



> **Примечание**
>
> Свойства можно упорядочить по категориям или в алфавитном порядке. Переключение между двумя этими представлениями можно делать с помощью кнопок в окне **Свойства**. В этом руководстве свойства легче находить в представлении, в котором свойства представлены в алфавитном порядке.

4. Вернитесь к конструктору Windows Forms. Щелкните нижний правый маркер переноса формы, который представляет собой небольшой белый квадрат в нижнем правом углу формы и показан на рисунке ниже.

Маркер переноса



Перетащите его, чтобы изменить размер формы — она должна стать шире и немного выше.

5. Посмотрите в окно **Свойства** и обратите внимание, что изменилось значение свойства **Size**.Свойство **Size** меняется каждый раз при изменении формы. Перетащите маркер переноса, чтобы форма имела размер около 550, 350. Такой размер вполне подходит для данного проекта.

6. Снова выполните программу. Нажмите клавишу F5 или нажмите кнопку панели инструментов**Начать отладку**, которая показана на рисунке ниже.

Кнопка панели инструментов "Начать отладку"

Как и ранее, интегрированная среда разработки выполняет построение программы и запускает ее, открывается окно.

7. Перед переходом к следующему шагу, остановите программу, так как интегрированная среда разработки не позволяет изменять программу при ее выполнении.

Создание макета формы с помощью элемента управления TableLayoutPanel

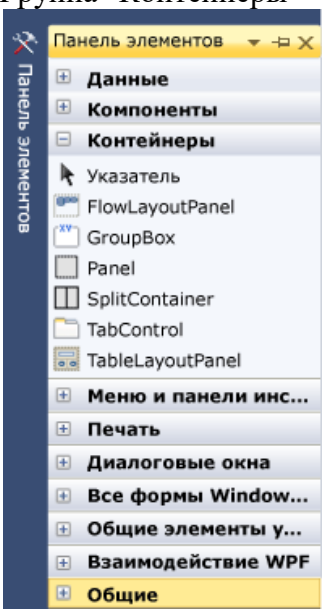
На этом шаге выполняется добавление в форму элемента управления **TableLayoutPanel**.

Создание макета формы с помощью элемента управления TableLayoutPanel

1. Перейдите в конструктор Windows Forms. Посмотрите на левую сторону формы и найдите вкладку **Панель элементов**. Наведите указатель на вкладку **Панель элементов** и задержите над ней. Появится "Панель элементов" (или в меню **Вид** выберите пункт **Панель элементов**).

2. Щелкните по знаку плюс рядом с группой **Контейнеры**, чтобы открыть ее как на рисунке ниже.

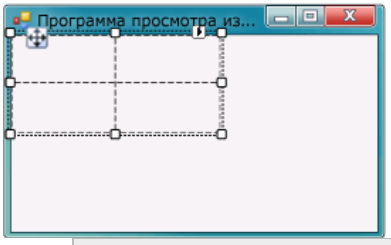
Группа "Контейнеры"



3. В форму можно добавить такие элементы управления, как кнопки, флажки и метки. Дважды щелкните элемент управления **TableLayoutPanel** в панели элементов. В результате этого действия

интегрированная среда разработки добавляет в форму элемент управления**TableLayoutPanel**, как показано на рисунке ниже.

Элемент управления TableLayoutPanel



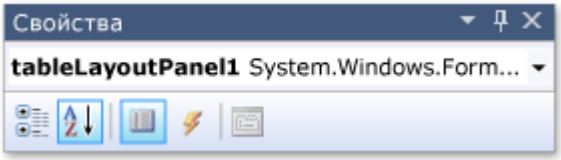
Примечание
После того, как добавляется элемент управления TableLayoutPanel, если внутри формы появляется окно с заголовком **Задачи TableLayoutPanel**, чтобы закрыть его, щелкните в любом месте внутри формы. Далее в этом руководстве приводятся сведения об этом окне.

Примечание
Обратите внимание, как разворачивается панель элементов, чтобы закрыть форму при нажатии на ее вкладку, и закрывается, после щелчка за ее пределами. Это функция автоматического скрытия интегрированной среды разработки. Ее можно включить или выключить для любого из окон, с помощью нажатия значка канцелярской кнопки в правом верхнем углу окна, чтобы включить автоматическое скрытие и закрепление на месте.Появляется значок канцелярской кнопки, как показано на рисунке ниже.

Значок канцелярской кнопки

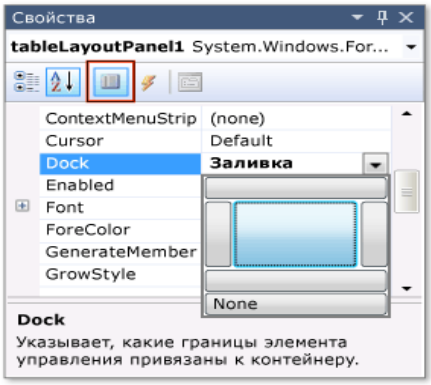
4. Убедитесь, что выделен элемент управления **TableLayoutPanel** — для этого щелкните по нему.Чтобы проверить, что элемент управления выделен, необходимо посмотреть в раскрывающийся список в верхней части окна **Свойства**, как показано на рисунке ниже.

Окно "Свойства", в котором показан элемент управления TableLayoutPanel



5. Селектор элементов управления представляет раскрывающийся список в верхней части окна**Свойства**. В данном пример он показывает, что выделен элемент управления с именем**tableLayoutPanel1**. Элементы управления можно выделять с помощью щелчка в конструкторе Windows Forms или выбирая их в селекторе элементов управления. Теперь, когда выделен элемент управления **TableLayoutPanel** найдите свойство **Dock** и щелкните **Dock**, значение которого должно быть равным **None**. Обратите внимание, что рядом со значением появляется стрелка раскрывающегося списка. Нажмите стрелку, затем нажмите кнопку **Заполнение**(большая кнопка по середине), как показано на рисунке ниже.

Окно "Свойства", в котором нажата кнопка "Заполнение"



6. После того, как у элемента управления TableLayoutPanel свойству **Dock** присвоено значение **Fill**, панель заполняет всю форму. Если снова изменить размер формы, элемент управления TableLayoutPanel останется закрепленным и сам изменит свой размер для заполнения формы.

Примечание
Элемент управления TableLayoutPanel работает как таблица в Microsoft Office Word — он содержит строки и столбцы и отдельная ячейка может занимать несколько строк и столбцов.Каждая ячейка может содержать один элемент управления (например, кнопку, флажок или метку). Этот элемент управления TableLayoutPanel будет содержать элемент управления**PictureBox**, который займет всю верхнюю строку, элемент управления **CheckBox** в левой нижней ячейке, и четыре элемента управления **Button** в правой нижней ячейке.

Примечание
Хотя было сказано, что каждая ячейка может содержать только один элемент управления, нижняя правая ячейка содержит четыре элемента управления **Button**. Это так, потому что можно разместить элемент управления в ячейке, которая содержит другие элементы управления. Такой тип элементов управления называется "контейнер", и элемент управления TableLayoutPanel является контейнером. Далее в этом руководстве приводятся сведения об этом типе элементов.

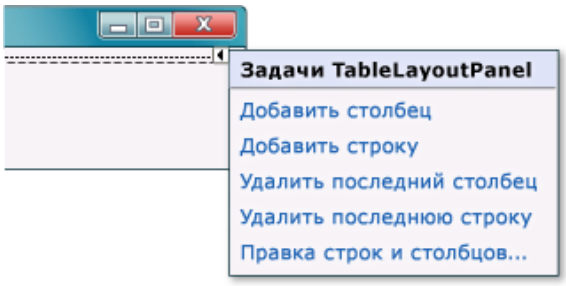
7. В данный момент элемент управления TableLayoutPanel содержит две одинаковые по размеру строки и два одинаковых по размеру столбца. Нужно изменить их размер, чтобы верхняя строка и правый столбец были намного больше. В конструкторе Windows Forms выберите элемент управления TableLayoutPanel. В правом верхнем углу расположена маленькая кнопка с черным треугольником, как показано на рисунке ниже.

Кнопка с треугольником

Это кнопка указывает, что элемент управления содержит задачи, которые помогут автоматически задать его свойства.

8. Чтобы отобразить список задач элемента управления, как показано на рисунке ниже, нажмите треугольник.

Задачи элемента управления TableLayoutPanel



9. Щелкните задачу **Изменить строки и столбцы**, чтобы открыть окно **Стили столбцов и строк**.Щелкните **Column1**, убедитесь, что нажата кнопка **Проценты**, установите его размер равным 15 процентам — введите **15** в поле **Проценты**. (это элемент управления **NumericUpDown**, который далее будет использоваться в этом руководстве). Щелкните **Column2** и задайте значение 85 процентов. Пока не нажимайте кнопку **ОК**, так как окно будет закрыто. (но если это было сделано, его можно открыть повторно с помощью списка задач).
10. В раскрывающемся списке **Показать** в верхней части окна выберите **Строки**. Задайте для **Row1**значение 90 процентов, а для **Row2** 10 процентов.
11. Нажмите кнопку **ОК**. Элемент управления TableLayoutPanel теперь должен содержать большую верхнюю строку, маленькую нижнюю строку, маленький левый столбец и большой правый столбец. Изменить размер строк и столбцов в элементе управления TableLayoutPanel можно с помощью перетаскивания их границ.

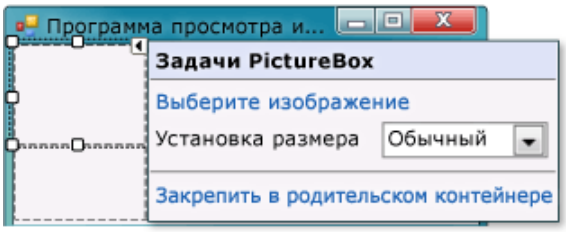
Добавление элементов управления в форму

На данном шаге производится добавление на форму элемента управления **PictureBox** и элемента управления **CheckBox**. Затем на форму добавляются кнопки.

Добавление элементов управления в форму

1. Перейдите к панели элементов и разверните **Стандартные элементы управления**. В результате этого действия отображается большая часть стандартных элементов управления, которые можно увидеть в формах.
2. Дважды щелкните элемент управления **PictureBox**. Интегрированная среда разработки добавляет элемент управления **PictureBox** в форму. Так как TableLayoutPanel размещена для заполнения формы, интегрированная среда разработки добавляет элемент управления**PictureBox** в первую пустую ячейку.
3. Чтобы отобразить список задач, щелкните черный треугольник на новом элементе управления**PictureBox**, как показано на рисунке ниже.

Задачи элемента управления PictureBox



> **Примечание**
>
> Если на TableLayoutPanel случайно был добавлен элемент управления неправильного типа, то его можно удалить. Щелкните правой кнопкой мыши элемент управления, затем выберите в меню пункт **Удалить**. Для удаления элемента с формы также можно выбрать пункт **Отменить**в меню **Правка**.

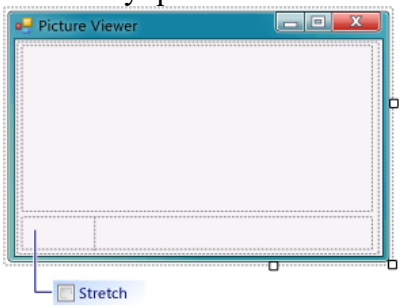
4. Щелкните элемент **Закрепление в родительском контейнере**. В результате этого действия у элемента управления PictureBox свойство **Dock** принимает значение **Fill**. Чтобы это увидеть,

выделите элемент управления **PictureBox**, перейдите к окну **Свойства** и убедитесь, что свойство **Dock** имеет значение **Fill**.

5. Сделайте так, чтобы элемент управления PictureBox занимал два столбца с помощью его свойства **ColumnSpan**. Выделите элемент управления **PictureBox** и установите его свойству**ColumnSpan** значение **2**. Также необходимо, чтобы когда элемент управления PictureBox был пустым, отображалась пустая рамка. Установите для его свойства **BorderStyle** значение **Fixed3D**.

6. Добавьте в форму элемент управления **CheckBox**. Дважды щелкните элемент **CheckBox** на панели элементов, чтобы интегрированная среда разработки добавила новый элемент управления **CheckBox** в следующую свободную ячейку таблицы. Так как элемент управления PictureBox занимает первые две ячейки, элемент управления **CheckBox** добавляется в нижнюю левую ячейку. Выберите новый элемент управления **CheckBox** и установите для свойства **Text**значение **Stretch**, как показано на рисунке ниже.

Элемент управления TextBox со свойством Stretch



7. На панели элементов перейдите к группе **Контейнеры** (там где был получен элемент управления **TableLayoutPanel**) и дважды щелкните элемент управления **FlowLayoutPanel**, чтобы добавить новый элемент управления в последнюю ячейку в элементе управления PictureBox. Затем закрепите его в родительском контейнере (выберите в списке задач**Закрепить в родительском контейнере** или установите для его свойства **Dock** значение **Fill**).

Примечание
Элемент управления FlowLayoutPanel является контейнером, который размещает другие элементы управления аккуратно по строкам и в определенном порядке. Когда изменяется размер элемента управления FlowLayoutPanel, если он содержит достаточно места для размещения всех компонентов в одной строке, то он размещает их именно таким образом. В противном случае он размещает их по строкам одну над другой. Элемент управления FlowLayoutPanel будет использоваться для размещения четырех кнопок.

Добавление кнопок

1. Выберите добавленный элемент FlowLayoutPanel. Перейдите к группе **Стандартные элементы управления** на панели элементов. Чтобы добавить кнопку с названием **button1** на элемент управления FlowLayoutPanel, дважды щелкните по значку **Button**. Чтобы добавить другую кнопку, повторите это действие. Среда интегрированной разработки определяет, что уже существует кнопка с именем **button1** и называет следующую кнопку как **button2**.

Примечание
В Visual Basic имена кнопок начинаются с заглавной буквы, поэтому имя **button1** будет как**Button1**, имя **button2** будет как **Button2** и так далее.

2. Обычно другие кнопки добавляются при помощи панели элементов. В этот раз выделите**button2**, затем в меню **Правка** выберите команду **Копировать** (или нажмите сочетание клавиш CTRL+C). Чтобы вставить копию кнопки, в меню **Правка** выберите пункт **Вставить** (или

нажмите сочетание клавиш CTRL+V). Повторите вставку еще раз. Среда интегрированной разработки добавила кнопки **button3** и **button4**.

Примечание
Любой элемент управления можно копировать и вставлять. Среда интегрированной разработки именует и размещает новые элементы управления логическим образом. Если вставка элемента управления выполняется в контейнер, среда интегрированной разработки выбирает следующую логическую область для размещения.

3. Выделите первую кнопку и установите для ее свойства **Text** значение **Показать рисунок**. Затем установите для свойства **Text** следующих трех кнопок значения **Очистить рисунок**, **Установить цвет фона**, **Закрыть**.
4. Следующий шаг это установка размера кнопок и размещение их таким образом, чтобы они были выровнены по левой стороне панели. Выделите элемент управления FlowLayoutPanel и посмотрите свойство **FlowDirection**. Измените его значение на **RightToLeft**. После этого действия кнопки должны сами выровняться по правой стороне ячейки и изменить свой порядок таким образом, чтобы кнопка **Показать рисунок** располагалась с правой стороны.

Примечание
Если кнопки по-прежнему остаются в неправильном порядке, можно перетащить кнопки вокруг элемента управления FlowLayoutPanel для расположения их в произвольном порядке.Можно щелкнуть по одной из кнопок и перетащить ее слева направо.

5. Чтобы выделить кнопку **Закрыть**, щелкните по ней. Удерживая клавишу CTRL, щелкните по трем другим кнопкам, чтобы все они были выделены. При выделенных кнопках перейдите к окну**Свойства** и прокрутите его вверх до свойства **AutoSize**. Это свойство указывает кнопке автоматически изменять свой размер так, чтобы весь текст мог разместиться на ней. Задайте значение **true**. Кнопки теперь должны иметь соответствующий размер и быть расположены в правильном порядке. (пока выделены все четыре кнопки, можно одновременно изменить все четыре свойства **AutoSize**). На следующем рисунке показаны эти четыре кнопки.

Программа просмотра изображений с четырьмя кнопками

6. Теперь снова запустите программу, чтобы увидеть обновленную компоновку формы. Нажатия на кнопки и установка флажков не дает результата, но скоро это заработает.

Присвоение имен элементам управления "Кнопка"

В форме существует только один элемент управления PictureBox. Когда он был добавлен, интегрированная среда разработки автоматически присвоила ему имя **pictureBox1**. Существует только один элемент управления CheckBox с именем **checkBox1**. Скоро будет написан некоторый код. В этом коде будет обращение к элементам управления CheckBox и PictureBox. Так как существуют только по одному экземпляру каждого компонента, то становится ясно, что означает упоминание имен**pictureBox1** или **checkBox1** в коде.

Примечание

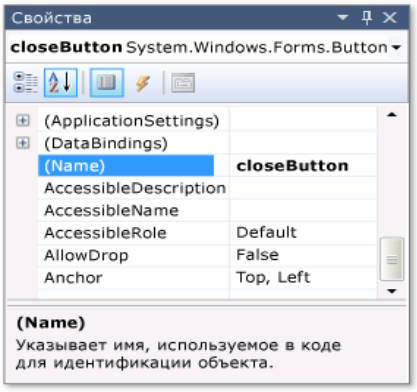
В Visual Basic по умолчанию первая буква любого имени элемента управления является заглавной, поэтому у элементов управления имена **PictureBox1**, **CheckBox1** и так далее.

В форме есть четыре кнопки. Интегрированная среда разработки назвала их как **button1**, **button2**,**button3** и **button4**. Только по их текущему имени нельзя узнать, какая кнопка является кнопкой**Закрыть**, а какая кнопкой **Показать рисунок**. Вот почему рекомендуется давать имена элементам управления "Кнопка".

Присвоение имен элементам управления "Кнопка"

1. Нажмите кнопку **Закрыть**. Если до сих пор выделены все кнопки, для отмены выделения нажмите клавишу ESC. Прокрутите содержимое окна **Свойства**, пока не появиться свойство**(Name)**. Свойство **(Name)** расположено в верхней части, когда свойства расположены в алфавитном порядке. Измените имя на **closeButton**, как показано на рисунке ниже.

Окно "Свойства" с именем closeButton

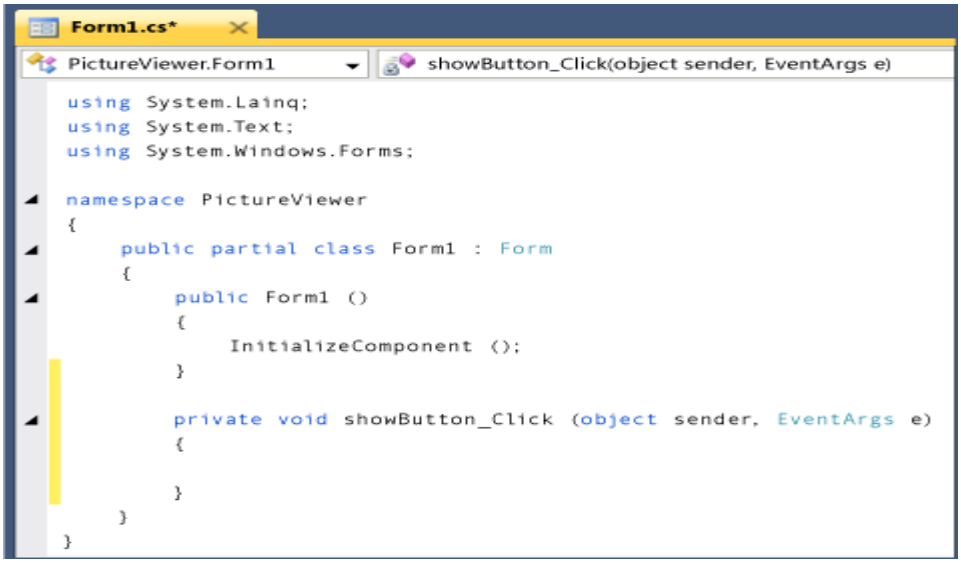


Примечание

Если попробовать изменить имя кнопки на **closeButton**, с пробелом между словами close и Button, среда интегрированной разработки отобразит сообщение об ошибке — "Недопустимое значение свойства". Пробелы (а также несколько других символов) запрещено использовать в именах элементов управления.

2. Переименуйте другие три кнопки как **backgroundButton**, **clearButton**, **showButton**. Имена можно проверить в раскрывающемся списке селектора элемента управления в окне **Свойства**.Отобразятся новые имена кнопок.
3. Дважды щелкните кнопку **Показать рисунок** в конструкторе Windows Forms. В результате этого действия интегрированная среда разработки откроет в главном окне новую вкладку **Form1.cs**, как показано на рисунке ниже.

Вкладка Form1.cs с кодом Visual C#



4. Обратите внимание на эту часть кода.

C#

```
private void showButton_Click(object sender, EventArgs e)
{
}
```

Необходимо найти метод с именем **showButton_Click()**. Интегрированная среда разработки добавила его при нажатии кнопки **showButton**. Этот метод содержит код, который выполняется, когда выполняется нажатие кнопки **Показать рисунок**.

> **Примечание**
>
> В этом руководстве код Visual Basic, который создается автоматически, был упрощен путем удаления всего, что находилось в круглых скобках (). Когда это происходит, одинаковый код можно удалить. Программа будет работать в любом случае. В оставшейся части руководства весь автоматически созданный код был упрощен везде, где это было возможным.

5. Перейдите на вкладку конструктора (вкладка **Form1.cs [Design]**в Visual C# и дважды щелкните кнопку **Очистить рисунок**. Повторите это действие для двух оставшихся кнопок. Каждый раз при этом действии среда интегрированной разработки добавляет в код формы новый метод.
6. Чтобы добавить еще один метод, дважды щелкните элемент управления **CheckBox** в конструкторе форм Windows Forms, чтобы интегрированная среда разработки создала метод**checkBox1_CheckedChanged()**. Этот метод вызывается каждый раз, когда пользователь устанавливает или снимает флажок.

> **Примечание**
>
> При работе с программой необходимо часто переключаться между редактором кода и конструктором Windows Forms. Среда интегрированной разработки упрощает передвижение по проекту. Используйте **Обозреватель решений**, чтобы открыть конструктор Windows Forms с помощью двойного щелчка по **Form1.cs** в Visual C#

7. Ниже показан новый код, который представлен в редакторе кода.
8. C#

```
9.
10.     private void clearButton_Click(object sender, EventArgs e)
11.     {
12.     }
13.
14.     private void backgroundButton_Click(object sender, EventArgs e)
15.     {
16.     }
17.
18.     private void closeButton_Click(object sender, EventArgs e)
19.     {
20.     }
21.
22.     private void checkBox1_CheckedChanged(object sender, EventArgs e)
23.     {
24.     }
25.
26.
```

Примечание

Пять методов, которые были добавлены, называются *обработчики событий*, так как программа вызывает их каждый раз, когда возникает событие (например, пользователь нажимает кнопку или устанавливает флажок).
При двойном щелчке элемента управления в интегрированной среде разработки, она добавляет метод обработчика событий для этого элемента управления. Например, при двойном щелчке кнопки интегрированная среда разработки добавляет обработчик события Click, который вызывается каждый раз, когда пользователь нажимает кнопку. Если дважды щелкнуть флажок, интегрированная среда разработки добавляет обработчик события CheckedChanged, который вызывается каждый раз, когда пользователь устанавливает или снимает флажок.
После добавления обработчика событий для элемента управления, к нему можно вернуться в любой момент из конструктора Windows Forms с помощью двойного щелчка по элементу управления.

Примечание

Имена являются важными при выполнении построения программы, и методы (включая обработчики событий) могут иметь любые имена, которые нужны. При добавлении обработчика событий с помощью интегрированной среды разработки, она выбирает имя на основе имени элемента управления и обрабатываемого события. Например, событие Click для кнопки с именем **showButton** вызывает метод обработчика событий **showButton_Click()**.Также обычно после имени метода добавляются открывающая и закрывающая круглые скобки, чтобы было ясно, какой метод рассматривается.

Добавление компонентов Dialog в форму

С целью подготовки настройки диалогового окна **Открыть файл** и диалогового окна **Цвет** (для выбора цвета фона), на данном шаге выполняется добавление в форму компонента **OpenFileDialog** и компонента **ColorDialog**.

В некотором смысле компонент похож на элемент управления. Для добавления компонента в форму используется панель элементов, его свойства настраиваются в окне **Свойства**. Но в отличие от элемента управления, добавление в форму компонента не добавляет в форму элемент, который может видеть пользователь. Вместо этого, компонент предоставляет определенное поведение, которое можно включать в коде. Имеется компонент, который открывает диалоговое окно **Открыть файл**.

Добавление компонентов диалогового окна в форму

1. Перейдите в конструктор Windows Forms и откройте в панели элементов группу **Диалоговые окна**.

> **Примечание**
>
> Группа **Диалоговые окна** в панели элементов содержит компоненты, которые открывают множество полезных диалоговых окон. Эти диалоговые окна могут использоваться для открытия и сохранения файлов, просмотра папок, выбора шрифтов и цветов. В этом проекте используется два компонента диалоговых окон — **OpenFileDialog** и **ColorDialog**.

2. Чтобы добавить в форму компонент с именем **openFileDialog1** дважды щелкните компонент**OpenFileDialog**. Для добавления в форму компонента с именем **colorDialog1** дважды щелкните в панели элементов компонент **ColorDialog**. (такой компонент используется в следующем шаге руководства). В нижней части конструктора Windows Forms должно быть поле серого цвета, которое содержит значок для каждого из двух добавленных компонентов диалоговых окон, как показано на рисунке ниже.

Компоненты диалоговых окон

3. Перейдите в конструктор Windows Forms и щелкните по значку **openFileDialog1** в поле серого цвета в нижней части конструктора. Установите значения двух свойств.
   - Для свойства **Filter** введите следующее значение (можно копировать и вставить его): JPEG Files (*.jpg)|*.jpg|PNG Files (*.png)|*.png|BMP Files (*.bmp)|*.bmp|All files (*.*)|*.*
   - Установите для свойства **Title** следующее значение "Выбор файла изображения".

> **Примечание**
>
> Чтобы посмотреть пример диалогового окна **Открыть файл** в другом приложении, откройте программу "Блокнот" или Paint, в меню **Файл** выберите пункт **Открыть**. Обратите внимание на раскрывающийся список **Типы файлов** в нижней части окна. Только что было настроено свойство **Filter** у компонента **OpenFileDialog**. Также, обратите внимание как выделены полужирным шрифтом свойства **Title** и **Filter** в окне **Свойства**. Таким образом интегрированная среда разработки показывает свойства, у которых были изменены их значения по умолчанию.

Написание кода для обработчика событий кнопки "Показать рисунок"

На этом шаге выполняется создание кнопки **Показать рисунок**, которая работает следующим образом.

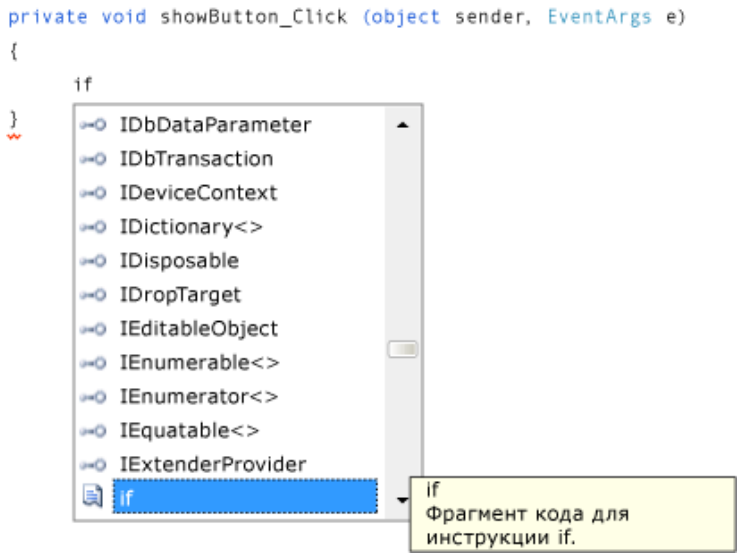
- Когда пользователь нажимает эту кнопку, программа открывает диалоговое окно **Открыть файл**.
- Если пользователь выбирает файл рисунка, программа показывает этот рисунок в элементе управления PictureBox.

Среда IDE содержит мощное средство IntelliSense, которое помогает в написании кода. По мере написания кода, среда IDE открывает поле, в котором содержатся предлагаемые завершения для

частей вводимых слов. Она пытается определить, что требуется сделать далее и автоматически переходит к последнему выбранному элементу из списка. Для перемещения по списку можно использовать клавиши со стрелками ВВЕРХ или ВНИЗ или можно продолжать вводить буквы, чтобы сузить выбор.Когда появится необходимый элемент, для его выбора нажмите клавишу TAB. Либо можно проигнорировать предложение, если оно не требуется.

Написание кода для обработчика событий кнопки "Показать рисунок"

1. Перейдите к конструктору Windows Forms и дважды щелкните кнопку **Показать рисунок**. Среда интегрированной разработки немедленно переключается на конструктор кода и перемещает курсор внутрь метода **showButton_Click()**, который был добавлен ранее.
2. Введите **i** в пустой строке между двумя фигурными скобками { } (в Visual Basic введите пустую строку между Private Sub… and End Sub). Откроется окно **IntelliSense**, как показано на рисунке ниже. IntelliSense с кодом Visual C#



3. Окно **IntelliSense** должно выделить слово **if**. (в противном случае введите в нижнем регистре **f**для этого). Обратите внимание, каким образом желтая подсказка рядом с окном **IntelliSense**показывает **Фрагмент кода для инструкции if**. (в Visual Basic подсказка также указывает такой фрагмент, но немного с другим содержимым). Необходимо использовать фрагмент. Для вставки**if** в код нажмите клавишу TAB. Затем снова нажмите клавишу TAB, чтобы использовать фрагмент**if**. (если пользователь щелкнет кнопкой мыши где-то еще, то окно **IntelliSense** исчезнет; нажмите клавишу BACKSPACE, чтобы удалить **i** и повторно введите эту букву, чтобы снова открыть окно **IntelliSense**).

Код Visual C#

```
private void showButton_Click(object sender, EventArgs e)
{
    if (true)
    {

    }
}
```

4. Далее IntelliSense используется для ввода дополнительного кода для открытия диалогового окна**Открыть файл**. Если пользователь нажимает кнопку **OK**, элемент управления PictureBox загружает выбранный пользователем файл. Следующие действия показывают как ввести код. Хотя представлено множество действий, это просто несколько нажатий клавиш.
1. Начните с выделенным текстом **true** в фрагменте. Введите **op**, чтобы перезаписать его. (в Visual Basic необходимо начинать с первой заглавной буквы, поэтому введите **Op**).
2. Откроется окно **IntelliSense** и отобразит **openFileDialog1**. Чтобы выбрать этот элемент нажмите клавишу TAB. (в Visual Basic он начинается с заглавной буквы, поэтому будет представлен **OpenFileDialog1**; убедитесь, что выделен **OpenFileDialog1**).

3. Введите точку (**.**) Так как точка введена справа после элемента **openFileDialog1**, окно**IntelliSense** открывается с методами и свойствами компонента **OpenFileDialog**. Это те же самые свойства, которые появляются в окне **Свойства** при выделении компонента в конструкторе Windows Forms. Также содержатся методы, которые указывают компоненты выполнить действия (например, открыть диалоговое окно).

Примечание
Окно **IntelliSense** может показывать свойства и методы. Для определения того, что показывается, посмотрите на значок слева. Рядом с каждым методом представлен значок кубика, рядом с каждым свойством представлен значок руки. Также рядом с каждым событием представлен значок с изображением молнии. Ниже представлены эти значки.

4. Значок метода
5. 
6. Значок свойства
7. 
8. Значок события
9. 
10. Начните набирать **ShowDialog** (для IntelliSense регистр значения не имеет). Метод**ShowDialog()** будет открывать диалоговое окно **Открыть файл**. После того, как в окне будет выделен метод **ShowDialog** нажмите клавишу TAB.
11. При использовании метода в элементе управления или компоненте (такое использование называется *вызов метода*) необходимо добавить круглые скобки.Поэтому введите открывающую и закрывающую круглые скобки **()**

Примечание
Методы являются важнейшей частью любой программы. В этом руководстве показано несколько способов использования методов. Можно вызвать метод компонента, чтобы указать ему выполнение некоторых действий, например, как у компонента **OpenFileDialog** вызывается метод **ShowDialog()**. Можно создать собственные методы, чтобы программа выполняла действия, как метод, построение которого выполняется сейчас, выполняет вызов метода **showButton_Click()**, который открывает диалоговое окно и рисунок при нажатием пользователем кнопки.

12. В Visual C# добавьте пробел, затем два знака равенства (==). В Visual Basic добавьте пробел, затем один знак равенства (=). В Visual C# и в Visual Basic используются разные операторы равенства.
13. Добавьте еще один пробел. Как только это будет сделано, откроется другое окно**IntelliSense**. Начните вводить **DialogResult** и нажмите клавишу TAB, чтобы добавить его.

Примечание
При написании кода для вызова метода, в некоторых случаях он возвращает значение. В данном случае у компонента **OpenFileDialog** метод **ShowDialog()**возвращает значение

DialogResult. DialogResult — это специальное значение, которое указывает на событие, которое происходит в диалогом окне. В компоненте**OpenFileDialog** пользователь может нажать кнопку **OK** или **Отмена**, поэтому метод**ShowDialog()** возвращает значение DialogResult.OK или значение DialogResult.Cancel.

14. Чтобы открыть значение DialogResult в окне **IntelliSense** введите точку. Чтобы вставить**OK** введите символ **O** и нажмите клавишу TAB.

Примечание

Первая строка кода должна быть завершена. В Visual C# это выглядит следующим образом.
if (openFileDialog1.ShowDialog() == DialogResult.OK)
В Visual Basic это выглядит следующим образом.
If OpenFileDialog1.ShowDialog() = DialogResult.OK Then

15. Теперь добавьте несколько строк кода. Их можно ввести вручную (или копировать и вставить), однако попробуйте использовать для добавления строк IntelliSense. Чем больше вы знакомы с IntelliSense, тем быстрее можете писать собственный код. Итоговая реализация метода **showButton_Click()** будет выглядеть следующим образом.
C#

```
private void showButton_Click(object sender, EventArgs e)
{
   if (openFileDialog1.ShowDialog() == DialogResult.OK)
   {
      pictureBox1.Load(openFileDialog1.FileName);
   }
}
```

Проверка, комментирование и тестирование кода
Перед добавлением комментариев в код и его проверкой, уделите время на ознакомление с понятиями кода, так как эти понятия будут часто использоваться.

- При двойном щелчке по кнопке **Показать рисунок** в конструкторе Windows Forms интегрированная среда разработки автоматически добавляла *метод* в код программы.
- Методами называют порядок организации кода — это то, каким образом группируется код.
- Большую часть времени метод выполняет небольшое количество действий в определенном порядке, например, метод **showButton_Click()** показывает диалоговое окно и затем загружает рисунок.
- Метод состоит из *операторов*. Считайте, что метод это способ объединения операторов.
- Когда метод выполняется или *вызывается*, операторы в методе выполняются по порядку, один за другим, начиная с первого.

Ниже приведен пример оператора.
C#

```
pictureBox1.Load(openFileDialog1.FileName);
pictureBox1.Load(openFileDialog1.FileName)
```

Операторы это то, что указывает программам выполнять действия.В Visual C# оператор всегда заканчивается точкой с запятой.В Visual Basic конец строки это конец оператора.(в Visual Basic точка с запятой не нужна). Предыдущий оператор указывал элементу управления **PictureBox**загрузить файл, который пользователь выбрал в компоненте **OpenFileDialog**. Затем в код добавляется комментарий. Комментарий это заметка, которая не влияет на выполнение программы. Она позволяет облегчить другим понимание назначения кода. В Visual C# строка

обозначается как комментарий с помощью двух косых черт (//). В Visual Basic строка обозначается как комментарий с помощью одинарной кавычки(').
После добавления комментария программа проверяется. Только что выполнено построение программы, которая работает. Хотя программа еще не завершена, но она уже может загружать рисунок.
Добавление комментариев
1. Добавьте следующее.

```
private void showButton_Click(object sender, EventArgs e)
{
   // Show the Open File dialog. If the user clicks OK, load the
   // picture that the user chose.
   if (openFileDialog1.ShowDialog() == DialogResult.OK)
   {
      pictureBox1.Load(openFileDialog1.FileName);
   }
}
```

Примечание

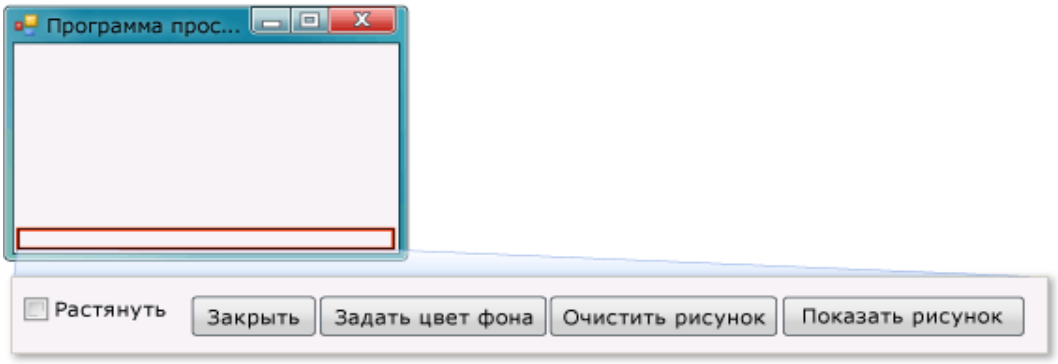
Обработчик события Click кнопки **showButton** завершен и он работает. Написание кода было начато с оператора **if**. Оператор **if** как-будто говорит программе: "Проверь это условие. Если оно выполняется, выполни эти действия". В данном случае программе указывается открыть диалоговое окно **Открыть файл** и, если пользователь выбирает файл и нажимает кнопку **ОК**, загрузить файл в элемент управления PictureBox.

Примечание

Интегрированная среда разработки построена так, чтобы облегчить написание кода и*фрагменты кода* являются одним из этих способов. Фрагмент представляет собой ярлык, который разворачивается в небольшой блок кода.
Посмотреть все фрагменты можно с помощью выбора пункта **Диспетчер фрагментов кода**в меню **Сервис**. Фрагмент **if** расположен в разделе **Шаблоны кода**, внутри вложенной папки**Условия и циклы**. Этот диспетчер можно использовать для просмотра существующих фрагментов и для добавления собственных фрагментов.
Чтобы активировать фрагмент при наборе кода, введите его и нажмите клавишу TAB. В окне**IntelliSense** появится множество фрагментов, поэтому нажмите клавишу TAB дважды — первый раз, чтобы выделить фрагмент в окне **IntelliSense**, второй раз, чтобы указать интегрированной среде разработки использовать фрагмент. (IntelliSense поддерживает фрагмент **if**, но не фрагмент **ifelse**).

2. Перед выполнением программы, сохраните программу с помощью нажатия кнопки панели инструментов **Сохранить все**, которая показана на рисунке ниже.
Кнопка "Сохранить все"

Другой способ сохранения программы — в меню **Файл** выберите пункт **Сохранить все**.Рекомендуется выполнять сохранение от начала разработки и как можно чаще.
При выполнении программа должна выглядеть, как показано на рисунке ниже.
Программа просмотра изображений



Проверка программы

1. Нажмите клавишу F5 или нажмите кнопку панели инструментов **Начать отладку**.
2. Чтобы выполнить код, который был написан только что, нажмите кнопку **Показать рисунок**.Сначала программа открывает диалоговое окно **Открыть файл**. Проверьте, что в нижней части диалогового окна в раскрывающемся списке **Типы файлов** появились фильтры. Затем перейдите к рисунку и откройте его. Обычно образцы рисунков, которые поставляются вместе с операционной системой Windows, можно найти в папке **Мои документы** во вложенной папке**Мои рисунки\Образцы рисунков**.
3. Загрузите рисунок и он появится в элемент управления PictureBox. Затем попробуйте изменить размер формы. Так как элемент управления PictureBox закреплен внутри элемента управления TableLayoutPanel, который сам закреплен внутри формы, область картинки будет сама изменять размер, таким образом, что по ширине она будет как форма и заполнит 90 процентов формы.Поэтому используются контейнеры TableLayoutPanel и FlowLayoutPanel — они сохраняют правильные размеры формы, когда пользователь меняет ее размер.

Создание дополнительных кнопок и флажка

Теперь можно завершить другие четыре метода. Можно копировать и вставить этот код, но для получения дополнительных навыков введите код и используйте IntelliSense.

Примечание
Рекомендация — всегда снабжайте код комментариями. Комментарии — это сведения для человека, который читает код, необходимы для того, чтобы сделать код понятным. Содержимое в строке комментария игнорируется программой. В Visual C# строка комментария начинается с двух символов косой черты (//), в Visual Basic строка комментария начинается с одного знака одинарной кавычки (').

Создание дополнительных кнопок и флажка

- Добавьте следующий код.

C#

```
private void clearButton_Click(object sender, EventArgs e)
{
   // Clear the picture.
   pictureBox1.Image = null;
}

private void backgroundButton_Click(object sender, EventArgs e)
{
   // Show the color dialog box. If the user clicks OK, change the
   // PictureBox control's background to the color the user chose.
   if (colorDialog1.ShowDialog() == DialogResult.OK)
      pictureBox1.BackColor = colorDialog1.Color;
}
```

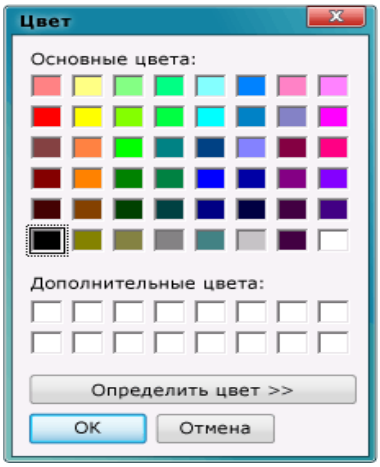
```
private void closeButton_Click(object sender, EventArgs e)
{
   // Close the form.
   this.Close();
}

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
   // If the user selects the Stretch check box,
   // change the PictureBox's
   // SizeMode property to "Stretch". If the user clears
   // the check box, change it to "Normal".
   if (checkBox1.Checked)
      pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
   else
      pictureBox1.SizeMode = PictureBoxSizeMode.Normal;
}
```

Запуск программы и изучение других функций

Разработка программа завершена и она готова для выполнения. Можно запустить программу и настроить цвета фона. В качестве дополнительного занятия, попробуйте улучшить программу с помощью изменения цвета формы, настройки кнопок и флажков, изменения свойств формы.

Запуск программы и настройка цвета фона

1. Нажмите клавишу F5 или нажмите кнопку панели инструментов **Начать отладку**.
2. Перед открытием изображения, нажмите кнопку **Установить цвет фона**. Откроется диалоговое окно **Цвет**.



Диалоговое окно "Цвет"

3. Выберите цвет заднего фона для элемента управления PictureBox. Внимательно просмотрите метод **backgroundButton_Click()**, чтобы понять, как он работает.

Примечание
Можно загрузить рисунок из Интернета, путем вставки URL-адреса в диалоговое окно**Открыть файл**. Попробуйте найти изображение с прозрачным фоном, таким образом, чтобы был показан цвет фона.

4. Убедитесь, что он очищен с помощью нажатия на кнопку **Очистить рисунок**. Затем завершите работу программы, для этого нажмите кнопку **Закрыть**.

Изучение других функций

- Измените цвет формы и кнопок с помощью свойства **BackColor**.
- Настройте кнопки и флажки с помощью свойств **Font** и **ForeColor**.
- Измените свойства формы **FormBorderStyle** и **ControlBox**.

- Используйте свойства **AcceptButton** и **CancelButton**, чтобы выполнялось автоматическое нажатие кнопок при нажатии пользователем клавиши ВВОД или клавиши ESC. Сделайте так, чтобы программа открывала диалоговое окно **Открыть файл** при нажатии пользователем клавиши ВВОД и закрывала окно при нажатии клавиши ESC.

ПРАКТИЧЕСКАЯ РАБОТА № 2

Тема работы: Создание лабиринта

Цель работы: разработка приложения для реализации игры Лабиринт

Ход работы:

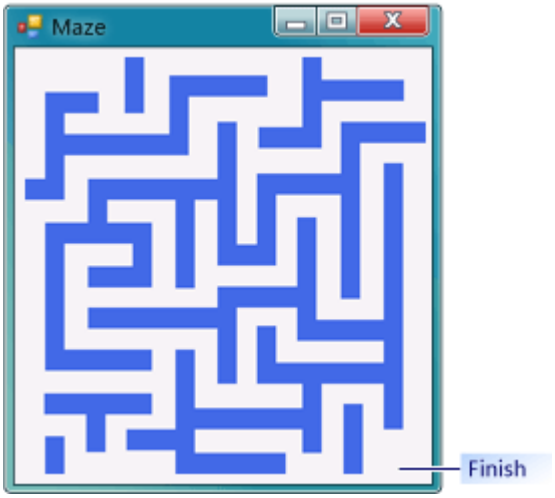
В этом руководстве выполняется создание игры "Лабиринт", в которой пользователь должен переместить указатель мыши от старта к финишу и не коснуться при этом стен лабиринта. Вы научитесь:

- Создавать макет формы с помощью контейнера Panel.
- Создавать лабиринт с помощью элементов управления Label.
- Создавать код для отображения окна сообщений.
- Настраивать обработчики событий для событий мыши.
- Воспроизводить в программе звуке.
- Упорядочивать код с помощью классов.

Описание игры в "Лабиринт". Указатель мыши появляется в левом верхнем углу лабиринта. Пользователь проходит по лабиринту, аккуратно, стараясь не задеть стены указателем мыши. Если указатель касается стены, он автоматически возвращается в исходную позицию. Если указатель достигает метки Финиш в конце лабиринта, отображается окно сообщений с поздравлением и игра заканчивается.

В результате ваша программа будет выглядеть так, как показано на следующем рисунке.

Игра, которую вы создадите в этом учебном руководстве



Примечание

В этом учебном руководстве показаны примеры, как на Visual C#, так и на Visual Basic, поэтому обратите внимание на информацию, которая относится к используемому вами языку программирования.

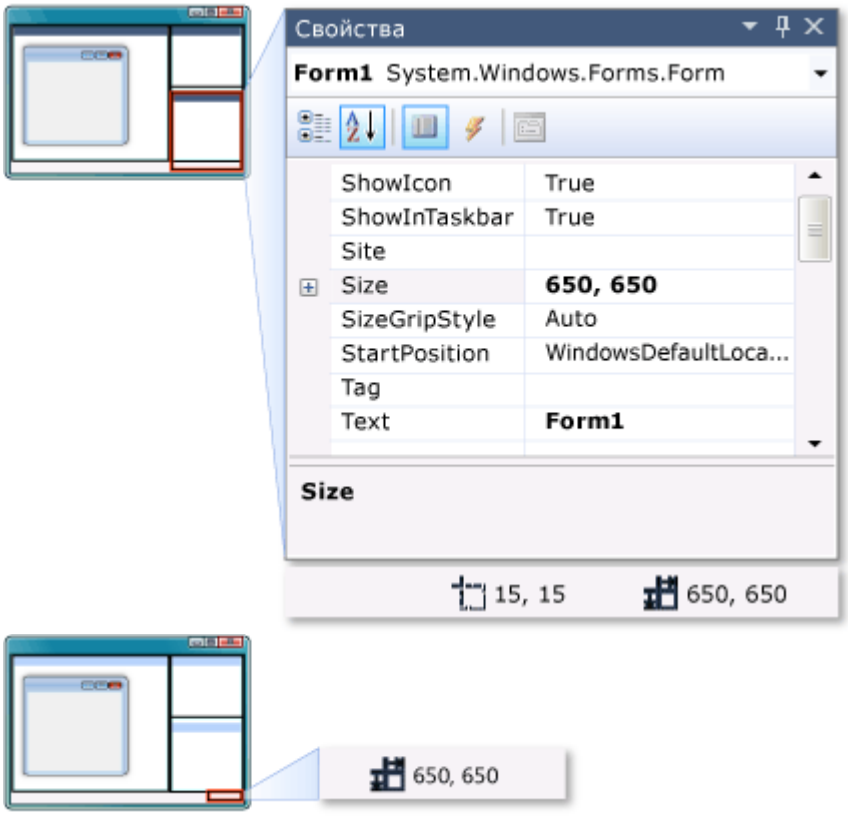
Создание проекта и добавление в форму контейнера Panel

Первым шагом создания игры "Лабиринт" является создание проекта и добавление в форму контейнера Panel.

Создание проекта и добавление контейнера Panel

1. В меню Файл выберите команду Создать проект.
2. Если используется не Visual Studio Express, вначале необходимо выбрать язык. В списке Установленные шаблоны выберите C# или Visual Basic.
3. Щелкните значок Приложение Windows Forms, а затем введите в качестве имени Maze.
4. Задайте свойства формы.
1. Измените размер формы, используя для этого указатель для перетаскивания правого нижнего угла. Следите за правым нижним углом интегрированной среды разработки. В строке состояния отображается размер формы. Перетаскивайте уголок формы с помощью указателя, пока размер формы не составит 650 пикселей. Можно построить лабиринт большего или меньшего размера, поэтому сделайте форму такого размера, какой нужен.

Размер в строке состояния



2. После того как у формы будет нужный размер, установите свойству Текст значение Лабиринт.
3. Так как пользователь не может изменять размер формы, установите для свойства FormBorderStyle значение Fixed3D.
4. Отключите кнопку Развернуть в заголовке окна. Для этого установите для свойства MaximizeBox значение False.

Теперь у формы заданный размер и пользователь не может ее развернуть.

Примечание

При создании новой формы по умолчанию задано два способа изменения ее размера пользователем. Пользователь может перетащить стороны или углы формы, либо может нажать кнопку Развернуть, чтобы развернуть форму. Если необходимо, чтобы пользователь не мог изменить размер формы, отключите обе эти возможности. Установка для свойства FormBorderStyle значения любого заданного стиля предупреждает изменение пользователем размера формы, но пользователь по-прежнему может нажать кнопку Развернуть. Поэтому также необходимо отключить свойство MaximizeBox.

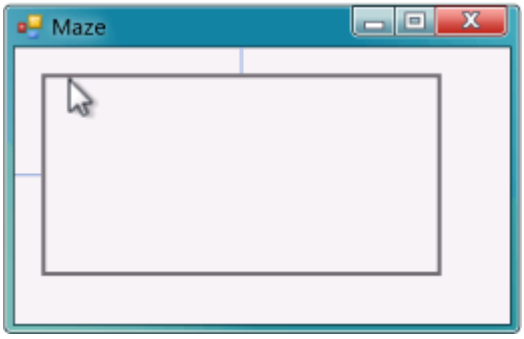
Далее необходимо создать игровое поле, на котором будет построен лабиринт. Для этого используется элемент управления Panel. Панель это контейнерный элемент управления, который позволяет размещать группы элементов управления. В отличие от других контейнеров (например, контейнера TableLayoutPanel и контейнера FlowLayoutPanel) панель не выполняет переупорядочивание элементов управления, которые содержит. Это дает свободу действий при расположении элементов управления в нужных местах, но в отличие от элементов управления TableLayoutPanel или FlowLayoutPanel, панель не рекомендуется использовать, когда пользователь может изменять размеры окна.

5. Перейдите в группу Контейнеры на панели элементов и дважды щелкните элемент управления Panel, чтобы добавить элемент управления в форму. Когда панель выделена, то в ее левом верхнем углу должен быть значок маркера перемещения, который показан на рисунке ниже.

Маркер перемещения

6. Перетащите панель на небольшое расстояние от левого верхнего угла формы. По мере того, как выполняется перетаскивание, можно наблюдать полезную функцию интегрированной среды разработки — как только панель оказывается на определенном расстоянии от верхней части или левой границы формы, она привязывается к расположению и между границей панели и границей формы появляется синяя линия-разделитель. Это можно использовать для выравнивания панели, чтобы ее границы были точно на одном расстоянии от границы формы. Как только появятся верхняя и левая синие линии-разделители, отпустите кнопку мыши, чтобы разместить панель на месте. На рисунке ниже показаны синие линии-разделители.

Синие линии-разделители



Перетаскивайте нижний правый маркер переноса, пока панель не разместится в правой и нижней области.

7. Так как необходимо, чтобы пользователь видел границу лабиринта, необходимо сделать ее видимой. Выделите панель и установите для свойства BorderStyle значение Fixed3D.
8. Сохраните проект. Для этого нажмите на кнопку панели инструментов Сохранить все, которая показана на рисунке ниже.

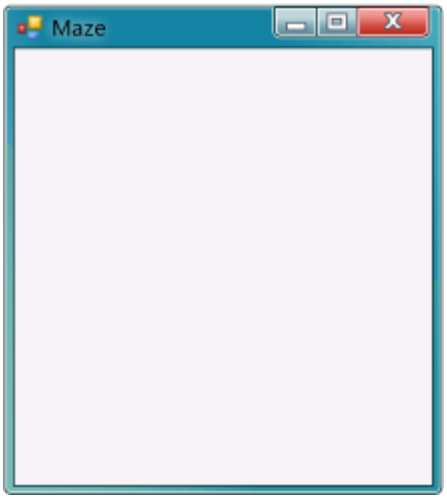
Кнопка "Сохранить все"

9. Чтобы выполнить программу, нажмите кнопку F5 или нажмите кнопку панели инструментов Начать отладку, которая показана на рисунке ниже.

Кнопка панели инструментов "Начать отладку"

При выполнении форма должна выглядеть так, как показано на рисунке ниже.

Начальная форма игры "Лабиринт"



10. Перед переходом к следующему шагу руководства остановите программу. Для этого закройте форму или нажмите кнопку панели инструментов Остановить отладку на панели инструментов Отладка. Пока программа выполняется интегрированная среда разработки остается в режиме только для чтения.

Построение лабиринта с помощью элементов управления Label

Теперь пришло время для создания лабиринта. Лабиринт создается путем добавления в форму множества элементов управления Label. Обычно метка используется для отображения текста. Но в этом проекте метка используется для отображения в форме цветного прямоугольника, который будет стеной лабиринта.

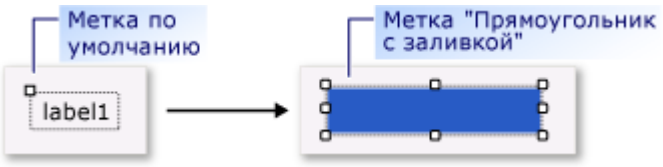
Построение лабиринта с помощью элементов управления Label

1. В конструкторе Windows Forms перейдите на панели элементов в группу Стандартные элементы управления и дважды щелкните по элементу управления Label, чтобы интегрированная среда разработки добавила элемент управления в форму.

2. Установите несколько свойств, чтобы элемент управления Label стал прямоугольником, размер у которого можно менять.

- Задайте свойству AutoSize значение False.
- Установите для свойства BackColor нужный цвет. (в этом руководстве выбран цвет RoyalBlue на вкладке Веб-цвета).
- Измените значение свойства Text, чтобы оно было пустым — выделите текст label1 и удалите его.

Элемент управления Label как закрашенный прямоугольник



Элемент управления Label теперь должен быть как закрашенный прямоугольник.

Примечание

Это может показаться необычным, так как элемент управления Label предполагается использовать как метку. В этом случае элемент управления Label используется как блок для рисования, потому что это работает. Важной частью программирования является распознавание инструмента в имеющемся наборе (в данном случае таким набором является панель элементов интегрированной среды разработки) для выполнения работы, даже если это работа, для которой этой инструмент изначально не предназначен.

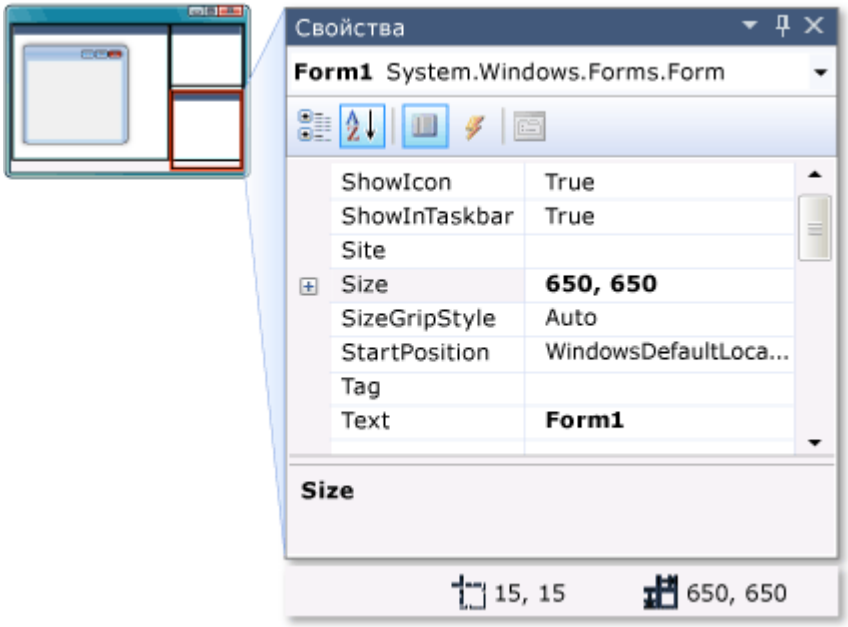
3. Проявите творчество при создании игры "Лабиринт". Копируйте элемент управления Label — выделите его, затем в меню Правка выберите пункт Копировать (или нажмите сочетание клавиш Ctrl+C). Затем вставьте его несколько раз. Нажмите сочетание клавиш CTRL+V или в меню Правка выберите пункт Вставить. В результате этого действия появятся несколько горизонтальных стен лабиринта. Возьмите одну из стен и измените ее размер таким образом, чтобы она стала высокой и узкой. Скопируйте и вставьте ее несколько раз, чтобы создать вертикальные стены.
4. Чтобы создать лабиринт, разместите метки на панели. Не нужно делать проходы слишком узкими, иначе игра будет слишком сложной. Оставьте больше места в левом верхнем углу, откуда игрок начнет прохождение лабиринта.

Примечание

Как упоминалась ранее, размер формы, когда выполняется изменение ее размера, отображается в строке состояния интегрированной среды разработки. Тоже самое происходит, когда выполняется изменение размера элемента управления Label или любого другого. При необходимости это можно использовать, для обеспечения гарантии, что все стены лабиринта имеют одинаковый размер.

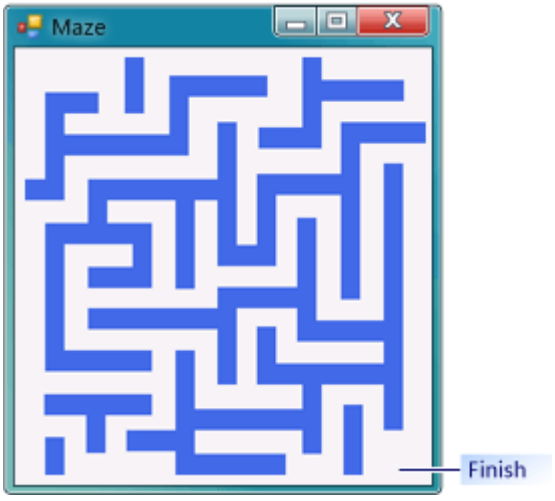
Линии выравнивания интегрированной среды разработки, которые использовались для размещения панели, также помогают при размещении стен лабиринта. Для более точного размещения текущего выделенного элемента управления можно использовать клавиши со стрелками. На рисунке ниже представлен размер в строке состояния.

5. Размер в строке состояния
6.



7. После создания макета лабиринта перейдите на панели элементов в группу Стандартные элементы управления и еще раз дважды щелкните элемент управления Label. Чтобы присвоить этому элементу управления имя finishLabel используйте строку (Name) в окне Свойства, а также измените значение его свойства Text на Финиш.
8. Перетащите новый элемент управления Label с текстом Финиш в конец лабиринта. Это цель, которую должен достичь пользователь.
9. Сохраните проект, а затем снова запустите программу. На рисунке ниже представлен пример завершенной формы игры "Лабиринт". Созданный вами лабиринт может выглядеть иначе.

Завершенная форма игры "Лабиринт"



Завершение игры

Чтобы создать конец игры, необходимо сделать рабочим элемент управления Label с текстом Финиш. Это выполняется путем добавления обработчика событий для события MouseEnter элемента управления Label.

Примечание

Если было изучено "Учебное руководство 1. Создание приложения для просмотра рисунков", то должно быть известно, что такое обработчик событий. Большая часть элементов управления содержит различные события, которые они могут вызывать. Программа просмотра изображений использует у элемента управления Button событие Click и у элемента управления CheckBox событие CheckChanged. В этом руководстве у элемента управления Label используется событие MouseEnter, которое возникает каждый раз, когда указатель мыши входит в элемент управления. Элемент управления Label содержит четыре десятка событий. Большая их часть имеет интуитивно понятные имена, например, DoubleClick, Resize, TextChanged. Далее в этом руководстве приводится список имен событий.

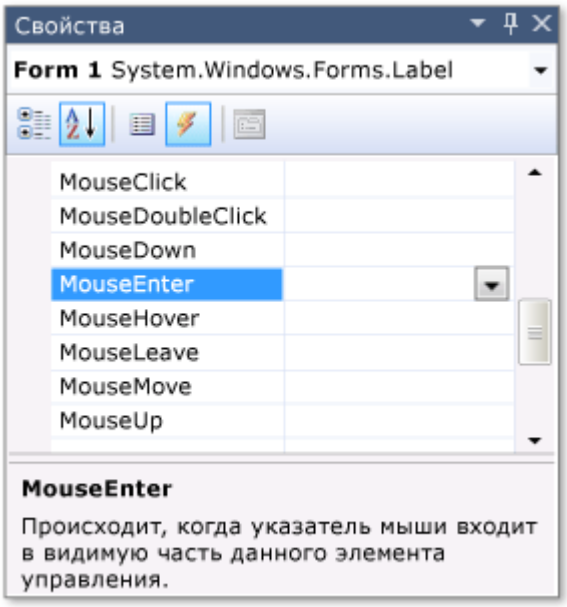
Завершении игры

1. Выберите элемент управления finishLabel, затем щелкните значок Событие (выглядит как изображением молнии) в верхней части окна Свойства. При его нажатии в окне вместо свойств отображаются события элемента управления. Чтобы вернуться к списку свойств, щелкните по значку Свойства. Теперь сделайте так, чтобы в окне Свойства отображались все события для элемента управления finishLabel. Прокрутите вниз до события MouseEnter. Упомянутые ранее значки и событие MouseEnter показаны на рисунке ниже.

Значок события 

Значок свойства 

Событие MouseEnter



2. Дважды щелкните слово MouseEnter. В результате этого действия интегрированная среда разработки автоматически добавляет метод обработчика событий в форму и показывает его в редакторе кода. Код представлен ниже.

C#

```
private void finishLabel_MouseEnter(object sender, EventArgs e)
{

}
```

Метод обработчика событий выполняется каждый раз, когда указатель мыши входит в метку.

3. Необходимо, чтобы программа открывало окно сообщений, в котором показывалось сообщение "Congratulations!" (Поздравляем!), а затем программа закрывалась. Для этого добавьте строки кода, которые представлены ниже (включая комментарий).

C#

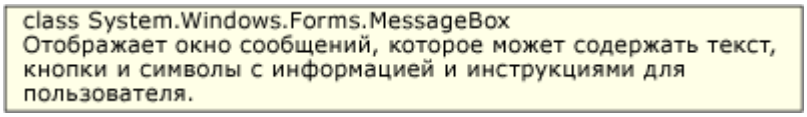
```
private void finishLabel_MouseEnter(object sender, EventArgs e)
{
   // Show a congratulatory MessageBox, then close the form.
   MessageBox.Show("Congratulations!");
   Close();
}
```

Примечание

Метод **finishLabel_MouseEnter()** содержит два оператора. Первый оператор вызывает метод с именем **Show()**, открывающий окно сообщений, в котором содержится текст, размещенный внутри кавычек.

4. Дополнительные сведения о происходящем можно получить путем изучения кода с помощью интегрированной среды разработки. Расположите указатель мыши поверх слова MessageBox. Должна отобразиться следующая подсказка.

Подсказка



Примечание

Интегрированная среда разработки показывает, что существует класс с именем **System.Windows.Forms.MessageBox**, и метод **Show()**, который вызывается внутри этого класса. Не требуется полного понимания, чтобы создать работающее окно сообщений, однако рекомендуется ознакомиться с дополнительными сведениями.

По второму оператору — каждая форма содержит встроенный метод с именем **Close()**, вызов которого приводит к закрытию формы. Некоторые программы содержат несколько окон, между которыми может переключаться пользователь. При работе с такой программой, он закрывает текущее окно, но оставляет оставшиеся окна программы работать. (например, если существует несколько одновременно открытых документов Microsoft Office Word, закрытие

окна документа приводит к закрытию этого документа, но Office Word остается открытым). Одна в программе такой, какая разрабатывается в этом руководстве, существует только одно окно. Закрытие этого окна приводит к прекращению выполнения, поэтому закрытие формы приводит к закрытию программы.

5. Сохраните и выполните программу. Поместите указатель мыши поверх метки Финиш. Это должно привести к появлению сообщения, затем программа закрывается.

Добавление метода для перезапуска игры

Было показано, как интегрированная среда разработки автоматически добавила в программу метод обработчика события. Также можно написать методы и добавить их в код. Многие программисты затрачивают большую часть своего времени на добавление собственных методов.

Примечание

Написание своих собственных методов рекомендуется, когда существует набор инструкций, которые нужно выполнить множество раз в различных местах. Это происходит часто при написании программ.

Например, при создании этой программы лабиринта, когда необходимо при запуске программы автоматически изменять положение указателя мыши в левый верхний угол панели. Когда пользователь перемещает указатель в стену необходимо, чтобы указатель перемещался на исходную позицию. Когда пользователь перемещает указатель за пределы игрового поля и возвращает его обратно необходимо, чтобы указатель снова перемещался на исходную позицию.

Перемещение указателя на исходную позицию можно реализовать с помощью трех строк кода. Однако если будет не нужно писать три одинаковые строки кода в нескольких различных местах программы, это сэкономит время. Если разместить эти три строки кода в методе, например, в методе с именем **MoveToStart()**, нужно будет написать их только один раз. Затем каждый раз, когда нужно переместить указатель назад в левый верхний угол панели, нужно просто вызвать метод **MoveToStart()**.

Добавление метода для перезапуска игры

1. Перейдите к коду формы. Для этого щелкните правой кнопкой мыши Form1.cs в Обозревателе решений и выберите в меню пункт Перейти к коду.
2. В коде должен быть представлен добавленный метод **finishLabel_MouseEnter()**. Сразу после этого метода добавьте новый метод **MoveToStart()**.

C#

```
private void MoveToStart()
{
   Point startingPoint = panel1.Location;
   startingPoint.Offset(10, 10);
   Cursor.Position = PointToScreen(startingPoint);
}
```

3. Существует специальный тип комментариев, которые можно добавлять перед любым методом. Интегрированная среда разработки поможет добавить его. Поместите курсор на строку перед новым методом. В Visual C# добавьте три знака косой черты (///). В Visual Basic добавьте три знака одинарных кавычек (''). Интегрированная среда разработки автоматически добавит следующий текст.

C#

```
private void MoveToStart()
{
   Point startingPoint = panel1.Location;
   startingPoint.Offset(10, 10);
   Cursor.Position = PointToScreen(startingPoint);
}
```

4. В строке между двумя тегами summary добавьте следующий комментарий. После нажатия клавиши ВВОД интегрированная среда разработки, в зависимости от языка программирования, добавляет три знака косой черты (///) или три знака одинарных кавычек (''), поэтому комментарий можно продолжить.

C#

```
// <summary>
/// Move the pointer to a point 10 pixels down and to the right
/// of the starting point in the upper-left corner of the maze.
/// </summary>
```

Примечание

Только что был добавлен XML-комментарий. Как уже упоминалось ранее интегрированная среда разработки показывает сведения в подсказке, если задержать указатель над словом MessageBox. Интегрированная среда разработки автоматически выводит сведения для разработанных методов. Все, что помещается в XML-комментарий отображается в подсказке интегрированной среды разработки, а также в окне IntelliSense. Это рекомендуется делать в программе, которая содержит множество методов. Также, если разместить стену, которая расположена на 10 пикселей вниз и вправо от левого верхнего угла панели, то в коде можно изменить (10, 10). Попробуйте с различными числами, пока не найдете начальную позицию указателя, которая подходит для лабиринта.

5. После добавления метода, его необходимо вызвать. Так как необходимо, чтобы программа перемещала указатель в начальную позицию сразу при запуске программы, необходимо вызвать метод непосредственно при запуске формы. Для Visual C# найдите в форме кода следующий метод.

C#

```
public Form1()
{
   InitializeComponent();
}
```

Это специальный метод, который называется конструктором. Он выполняется один раз при создании формы. Сейчас все что он делает, это вызов метода с именем **InitializeComponent()**. Нужно будет добавить строку к нему, чтобы вызвать только что написанный метод **MoveToStart()**. Перед продолжением рассмотрите, что добавить в программу, чтоб заставить ее вызвать метод **MoveToStart()** сразу после того, как она вызывает метод **InitializeComponent()**.

Примечание

Метод **InitializeComponent()** в конструкторе формы, это метод, который был написан интегрированной средой разработки. Он добавляет все элементы управления и компоненты в форму и устанавливает их свойства. Каждый раз, когда меняются свойства формы или ее элементов управления, интегрированная среда разработки изменяет этот метод. Его содержимое можно посмотреть, открыв файл Form1.Designer.cs из Обозревателя решений. Изменять содержимое метода **InitializeComponent()** не нужно. Интегрированная среда разработки следит за этим на основе формы, которая создается в представлении конструирования.

6. Добавьте вызов метода **MoveToStart()** сразу после вызова метода **InitializeComponent()**. Код формы должен выглядеть так, как показано в следующем примере.

C#

```
namespace Maze
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
                  MoveToStart();
      }

      private void finishLabel_MouseEnter(object sender, EventArgs e)
      {
         // Show a congratulatory MessageBox, then close the form.
         MessageBox.Show("Congratulations!");
         Close();
      }

      /// <summary>
      /// Move the pointer to a point 10 pixels down and to the right
      /// of the starting point in the upper-left corner of the maze.
      /// </summary>
      private void MoveToStart()
      {
         Point startingPoint = panel1.Location;
         startingPoint.Offset(10, 10);
         Cursor.Position = PointToScreen(startingPoint);
      }
   }
}
```

Обратите внимание на вызов метода **MoveToStart()** сразу после вызова метода **InitializeComponent()**. Если используется Visual C# помните, что конец строки завершается точкой с запятой (;), в противном случае не выполняется построение программы.

7. Сохраните и выполните программу. Как только программа запускается, указатель должен быть автоматически перемещен немного вниз и в правую часть левого верхнего угла панели.

Добавление обработчика событий MouseEnter для каждой стенки

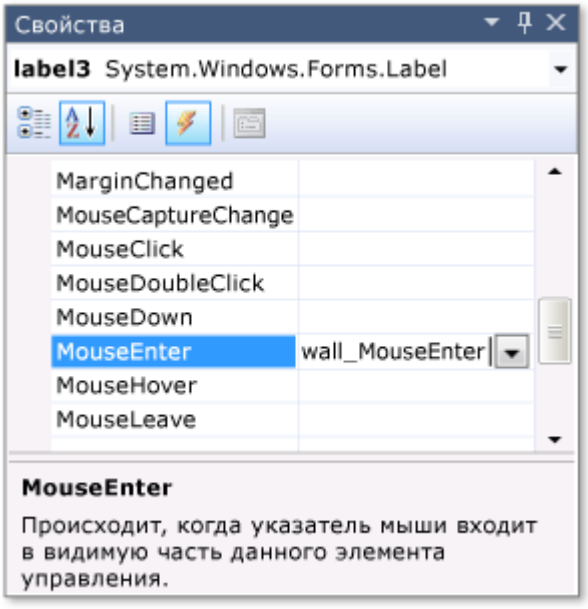
Игра "Лабиринт" будет более трудной и интересной если каждый раз, когда указатель касается стены, возвращать указатель мыши пользователя назад к начальной позиции. Перед тем, как читать дальше, подумайте, как это можно сделать.

Добавление обработчика событий MouseEnter для каждой стенки

1. Перейдите в конструктор Windows Forms и выделите любую из вновь добавленных стен.
2. Перейдите в окно Свойства и щелкните по значку Событие, чтобы отобразить события для этой стены. Прокрутите вниз до события MouseEnter. Вместо двойного щелчка по нему введите текст wall_MouseEnter, затем нажмите клавишу ВВОД. Появится значок Событие и окно Свойства, как показано на рисунке ниже.

Значок события 

Окно свойств, отображающее событие MouseEnter



Примечание

Если ввести имя события непосредственно в таблицу событий в окне Свойства, это прямо указывает интегрированной среде разработки создать обработчик события с этим именем и присоединить его к событию элемента управления. Часто необходимо, чтобы интегрированная среда разработки выбирала имена событий, так как имена являются

логичными, использование имен облегчает другим чтение и понимание кода. Когда интегрированная среда разработки выбирает имя для обработчика событий, она использует имя элемента управления и имя события. В данном случае имена стен по умолчанию не менялись, и они выглядят как label4, label18, label25 и так далее. Поэтому, если щелкнуть по стене с именем label12, интегрированная среда разработки назовет обработчик событий как label12_MouseEnter. Путем ввода имени wall_MouseEnter, выбирается более подходящее имя. Это особенно важно при использовании одного обработчика событий для нескольких элементов управления, что будет сделано далее в этом руководстве.

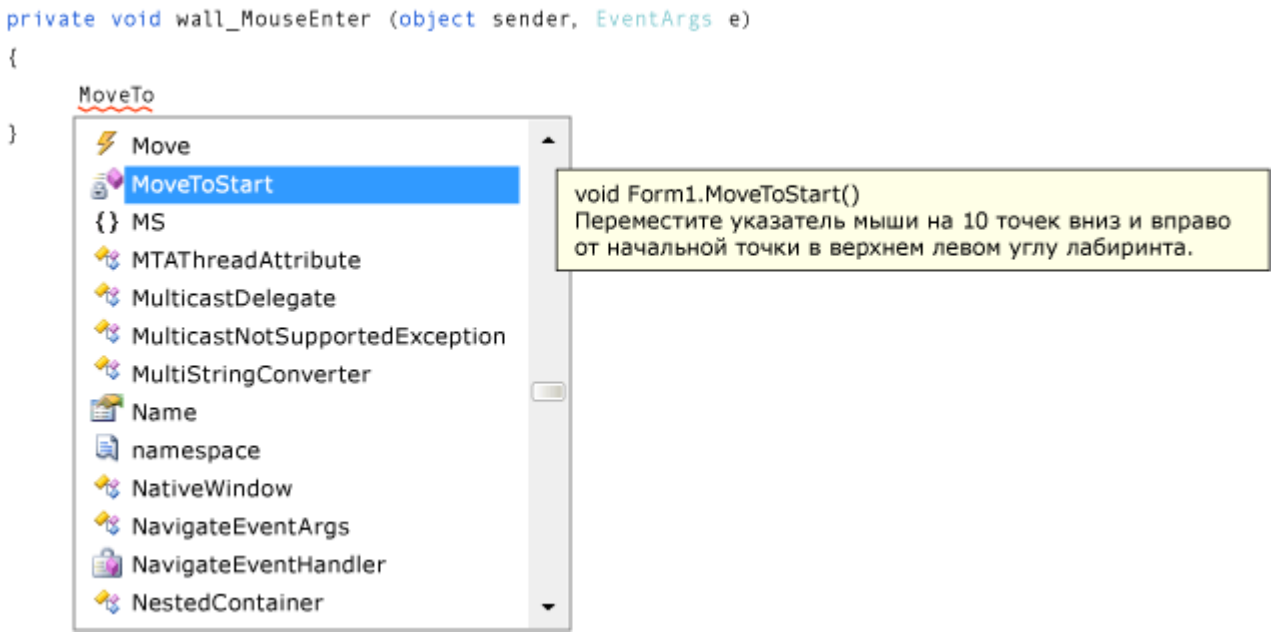
3. После нажатия клавиши ВВОД интегрированная среда разработки добавляет новый обработчик событий и присоединяет его к событию этой стены MouseEnter. В редакторе кода должен появиться вновь добавленный код, как показано на рисунке ниже. В Visual Basic указанная метка может не быть меткой с именем Label8, как показано в коде.

C#

```
private void wall_MouseEnter(object sender, EventArgs e)
{

}
```

4. Затем добавьте вызов метода **MoveToStart()** вместе с комментарием, который поясняет метод. Начните с перехода к методу и добавлением оператора **MoveToStart()**. Откроется окно IntelliSense, содержание которого показано на рисунке ниже.

Окно IntelliSense



При добавлении метода **MoveToStart()**, интегрированная среда разработки добавила его в окно IntelliSense. Добавленный XML-комментарий отображается в подсказке. Это полезно, когда в программе существует множество методов.

5. Чтобы указать IntelliSense завершить имя метода, нажмите клавишу TAB. При написании кода Visual C# code, не забывайте добавлять точку с запятой (;) в конце оператора. Затем добавьте комментарий перед инструкцией. Код должен выглядеть так, как показано ниже. В Visual Basic указанная метка может не быть меткой с именем Label8, как показано в коде.

C#

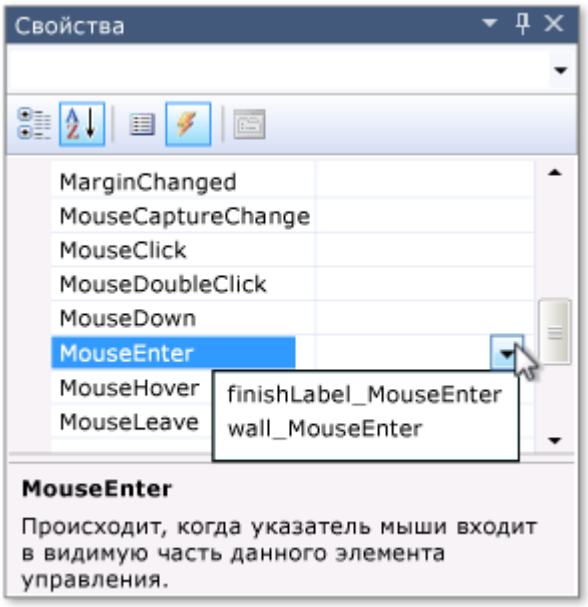
```
private void wall_MouseEnter(object sender, EventArgs e)
{
   // When the mouse pointer hits a wall or enters the panel,
   // call the MoveToStart() method.
   MoveToStart();
}
```

6. Сохраните и выполните программу. Переместите указатель мыши над стеной, к которой присоединен обработчик событий. (если не помните стену, перемещайте указатель над каждой стеной, пока не найдете нужную). Как только указатель коснется стены, он будет отправлен назад к начальной позиции.

Далее необходимо сделать тоже самое для оставшихся стен. Можно написать такой же обработчик событий MouseEnter для каждой из стен. Однако этот процесс будет длительным, приведет к появлению множества строк одинакового кода в программе и будет сложным для изменений. Интегрированная среда разработки предоставляет простой способ для соединения одного обработчика событий со всеми стенами.

7. Перейдите в конструктор Windows Forms. В меню Правка выберите пункт Выделить все.
8. Удерживайте клавишу CTRL, затем щелкните метку **Финиш**, чтобы снять выделение. В результате на панели должны остаться выделенными только стены.
9. В окне Свойства перейдите в таблицу событий. Прокрутите содержимое окна к событию MouseEnter и щелкните поле ввода рядом с ним. В нем должна быть стрелка раскрывающегося списка. Если щелкнуть по стрелке, должен появиться список всех обработчиков событий в программе, которые можно выбрать для этого события. В данном случае в списке должен быть обработчик событий finishLabel_MouseEnter, который был добавлен ранее, и обработчик событий wall_MouseEnter, который был написан только что, как показано на рисунке ниже.

Событие MouseEnter с обработчиками событий



10. Выделите обработчик событий wall_MouseEnter. Если выбрать неправильный обработчик событий либо случайно добавить новый обработчик событий, можно снова выделить все стены и панель, а затем выбрать нужный метод.
11. Теперь игра в лабиринт должна быть более интересной. Попробуйте сохранить и запустить ее. Если указатель наткнется на стену или указатель будет перемещен за пределы лабиринта и возвращен обратно, программа должна автоматически перемещать указатель в начальную позицию лабиринта.

Добавление объекта SoundPlayer

Далее добавим в игру "Лабиринт" звук. Один из звуков должен воспроизводится, когда пользователь касается стены и возвращается назад к начальной позиции. Другой звук должен воспроизводится, когда пользователь выигрывает. На данном шаге добавляется звук, который воспроизводится, когда указатель мыши касается стены. Хотя это может показаться сложным, требуется всего лишь несколько строк кода.

Добавление объекта класса SoundPlayer для шума

1. Начните добавление объекта класса SoundPlayer в код формы перед конструктором.

C#

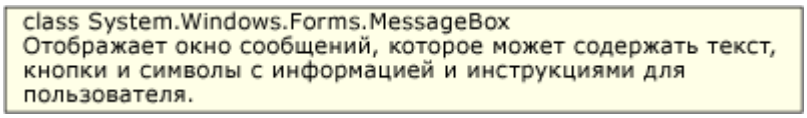
```
public partial class Form1 : Form
{
   // This SoundPlayer plays a sound whenever the player hits a wall.
   System.Media.SoundPlayer startSoundPlayer = new
System.Media.SoundPlayer(@"C:\Windows\Media\chord.wav");

   public Form1()
   {
      InitializeComponent();
      MoveToStart();
   }
```

Примечание

Первая строка (public partial class Form1 : Form) ранее встречалась несколько раз. Она является важной, так как содержит ключевое слово class. Ключевое слово class встречается много раз, так как класс это основной стандартный блок любой программы.

2. Ранее указатель мыши помещался над словом MessageBox в операторе MessageBox.Show("Congratulations!");, чтобы интегрированная среда разработки открыла подсказку. Повторите это еще раз и внимательно посмотрите на первую строку, которая приведена ниже.

Подсказка



Примечание

Ключевое слово class появляется в первой строке. Оно встречается часто, так как код организован в классы следующим образом — программа содержит классы, каждый класс содержит методы, каждый метод содержит операторы. Существует множество встроенных классов, например, **MessageBox**. Класс **MessageBox** содержит метод, который вызывается как **Show()** и будучи вызванным, он выполняет операторы, которые открывают окно сообщения. Также уже выполнялась работа с классами **Button**, **Label**, **Panel**. При установке значений свойств этих классов производилась работа с другой стороной классов — класс, как и метод, может содержать свойства; установка этих свойств приводит к выполнению классом операторов, которые меняют поведение.

Как можно догадаться, класс **SoundPlayer** — это класс, который воспроизводит звук. При создании объекта SoundPlayer вместе с ключевым словом new, он загружает звук из файла, который может быть воспроизведен с помощью метода **Play()**. Этот объект SoundPlayer будет использован для воспроизведения звука "Аккорд" операционной системы Windows, когда игрок начинает новую игру или когда указатель касается стены и игрок заново начинает игру. (поэтому у метода имя startSoundPlayer).

3. Если необходимо использовать другие звуки, замените путь между знаками кавычек (C:\Windows\Media\chord.wav) на путь к нужному звуковому файлу.

При создании формы с помощью конструктора Windows Forms при создании собственных классов используется помощь интегрированной среды разработки, в этом случае у класса будет имя **Form1**. При добавлении строки кода перед конструктором, в форму добавляется новый объект SoundPlayer, точно также, как ранее добавлялись кнопка или метка. Оператор расположен за пределами метода, поэтому доступ к объекту SoundPlayer может быть выполнен из нескольких методов. Вот почему необходимо поместить новый оператор внутри кода формы, но вне ее методов. Ему дается имя startSoundPlayer, таким же образом назван один из элементов управления **Label** — finishLabel.

После добавления оператора для создания нового объекта SoundPlayer и его вызова startSoundPlayer, он появляется в окне IntelliSense, также как метки, кнопки и другие элементы управления.

Это может показаться сложным, но это похоже на то, что выполнялось ранее в интегрированной среде разработки. Например, при использовании панели элементов интегрированной среды разработки для добавления кнопки или метки в форму, интегрированная среда разработки автоматически добавляет строки кода, которые используются для создания новой кнопки или новой метки. Сейчас выполнялось тоже самое, только на этот раз создается объект SoundPlayer. (второй объект SoundPlayer создается в следующем шаге руководства).

Добавление в форму кода для воспроизведения звуков

Теперь можно добавить второй компонент SoundPlayer, а затем добавить метод, для каждого компонента SoundPlayer.

Воспроизведение звуков

1. Начните с добавления второго компонента SoundPlayer для воспроизведения звука Tada операционной системы Windows. Игра будет воспроизводить этот звук, когда игрок достигает метки Финиш.

C#

```
public partial class Form1 : Form
{
   // This SoundPlayer plays a sound whenever the player hits a wall.
   System.Media.SoundPlayer startSoundPlayer = new
System.Media.SoundPlayer(@"C:\Windows\Media\chord.wav");

   // This SoundPlayer plays a sound when the player finishes the game.
   System.Media.SoundPlayer finishSoundPlayer = new
System.Media.SoundPlayer(@"C:\Windows\Media\tada.wav");

   public Form1()
   {
      InitializeComponent();
      MoveToStart();
   }
```

2. Оба компонента SoundPlayer теперь добавлены в форму. Добавьте метод **Play()**, чтобы вызывать компонент SoundPlayer для воспроизведения звуков в соответствующее время. Необходимо, чтобы звук воспроизводился, когда пользователь касается стены. Поэтому добавьте оператор startSoundPlayer.Play(); в метод **MoveToStart()**. Не забудьте обновить комментарий. Итоговая реализация метода будет выглядеть следующим образом.

C#

```
/// <summary>
/// Play a sound, then move the mouse pointer to a point 10 pixels down and to
/// the right of the starting point in the upper-left corner of the maze.
/// </summary>
private void MoveToStart()
{
   startSoundPlayer.Play();
```

```
    Point startingPoint = panel1.Location;
    startingPoint.Offset(10, 10);
    Cursor.Position = PointToScreen(startingPoint);
}
```

3. Добавьте оператор finishSoundPlayer.Play();в обработчик события MouseEnter метки Финиш. Не забудьте обновить комментарий, так как код изменяется, как показано в примере ниже.

C#

```
private void finishLabel_MouseEnter(object sender, EventArgs e)
{
    // Play a sound, show a congratulatory MessageBox, then close the form.
    finishSoundPlayer.Play();
    MessageBox.Show("Congratulations!");
    Close();
}
```

Запуск программы и изучение других функций

Разработка программа завершена и она готова для выполнения. Программу можно запустить и протестировать. Для закрепления навыков попробуйте изменить цвета и звуки.

Для запуска программы

1. Сохраните программу, затем запустите ее.
2. Убедитесь, что указатель мыши расположен в начале лабиринта.
3. Переместите указатель мыши по лабиринту. Заденьте стену и проверьте, что воспроизводится звук и указатель возвращается в исходную позицию.
4. Переместите указатель мыши за пределы лабиринта. Затем переместите указатель мыши назад на панель и проверьте, что указатель мыши возвращается к исходной позиции.

> **Примечание**
>
> При проверке необходимо убедиться, что все функции программы работают. Необходимо проверить, что обработчик событий MouseEnter элемента управления Label с надписью Финиш воспроизводит звук Tada, открывает окно сообщений с поздравлением и закрывает игру. Чтобы отменить необходимость прохождения всего лабиринта, можно временно выключить обработчик событий MouseEnter для элемента управления Panel. Таким образом можно переместить указатель мыши за пределы лабиринта и затем поместить его над элементом управления с текстом Финиш, при этом он не будет возвращен на исходную позицию.

5. Выделите элемент управления Panel, затем перейдите в таблицу событий в окне Свойства. Прокрутите содержимое окна вниз до событие MouseEnter и выделите имя события.
6. Чтобы удалить имя обработчика событий, нажмите клавишу DELETE, затем нажмите клавишу ВВОД. Интегрированная среда разработки автоматически отключает обработчик событий от панели. Обработчик событий для стен по-прежнему подключен, но можно переместить указатель мыши за пределы лабиринта, чтобы добраться до элемента управления Label с текстом Финиш в нижней части формы.

7. Сохраните и запустите программу. Проверьте, что элемент управления Label с текстом Финиш воспроизводит звук, отображается окно сообщений, игра закрывается. После проверки этого, включите обработчик событий MouseEnter элемента управления Panel. Выделите его, перейдите в окно Свойства, прокрутите содержимое окна вниз к строке события MouseEnter, выберите в раскрывающемся списке обработчик событий wall_MouseEnter.

Изучение других функций

- Замените звуки в игре на звуки, которые нравятся больше.
- Настройте их таким образом, чтобы игра воспроизводила звук, когда указатель мыши касается стены, но при запуске программы звук не воспроизводился.
- Вместо закрытия программы, когда игрок выигрывает, сделайте так, чтобы указатель возвращался на исходную позицию.
- Измените некоторые цвета стен, сделайте так, чтобы программа воспроизводила различные звуки для стен с разным цветом.

ПРАКТИЧЕСКАЯ РАБОТА № 4

Тема работы: Создание математической викторины

Цель работы: разработка приложения для проверки знаний арифметических действий над числами

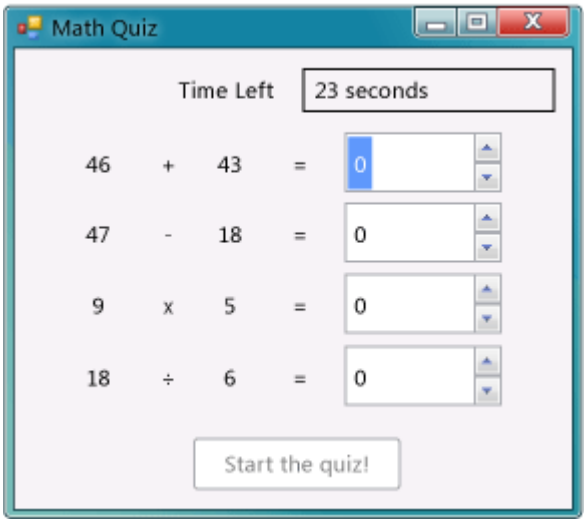
Ход работы:

В этом руководстве выполняется создание ограниченной по времени математической головоломки, в которой игрок должен решить четыре арифметические задачи со случайными числами за определенное время. Вы научитесь:

- Создавать случайные числа с помощью класса **Random**.
- Вызывать события с помощью элемента управления Timer.
- Управлять выполнением программы с помощью операторов if else.
- Выполнять основные арифметические операции.

В результате ваша программа будет выглядеть так, как показано на следующем рисунке.

Игра, которую вы создадите в этом учебном руководстве



Создание проекта и добавление в форму элементов управления Label

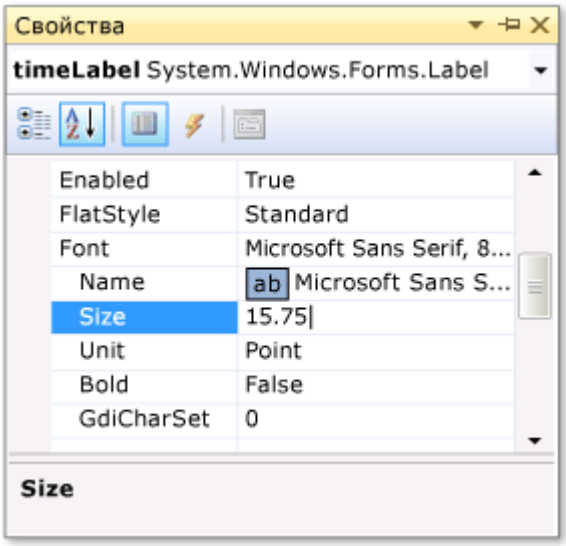
Первым шагом создания математической головоломки является создание проекта и добавление элементов управления Label в форму.

Создание проекта и добавление в форму элементов управления Label

1. В меню Файл выберите команду Создать проект.
2. Если используется не Visual Studio Express, вначале необходимо выбрать язык. В списке Установленные шаблоны выберите C# или Visual Basic.
3. Щелкните значок Приложение Windows Forms, а затем введите в качестве имени "Математическая головоломка".
4. Задайте свойства формы.
   1. Измените свойство Text формы на Математическая головоломка.

2. Измените размер на 550 пикселей по ширине и 550 пикселей по высоте с помощью свойства Size или путем перетаскивания границы, пока не увидите требуемый размер в нижнем левом углу интегрированной среды разработки (IDE).
3. Чтобы предупредить изменение пользователями размера формы измените значение свойства FormBorderStyle на Fixed3D, а значение свойства MaximizeBox на False.
5. Перетащите элемент управления Label с панели инструментов и задайте его свойства.
1. Измените значение свойства (Name) на timeLabel. Эта метка появляется в качестве поля в правом верхнем углу формы, на котором показан обратный отсчет количества секунд для головоломки.
2. Измените значение свойства AutoSize на False, чтобы можно было самостоятельно установить размер поля.
3. Измените значение свойства BorderStyle на FixedSingle для отрисовки линии вокруг поля.
4. Установите значение свойства Size равным 200, 30.
5. Перетащите метку в правый верхний угол формы, пока не появится синие линии-разделители.
6. Очистите свойство Text, для этого щелкните Text в окне Свойства и нажмите клавишу BACKSPACE.
7. Измените размер шрифта на 15.75. Щелкните по значку плюс рядом со свойством Font в окне Свойства, в котором представлено несколько свойств, включая свойство Size, как показано на рисунке ниже.

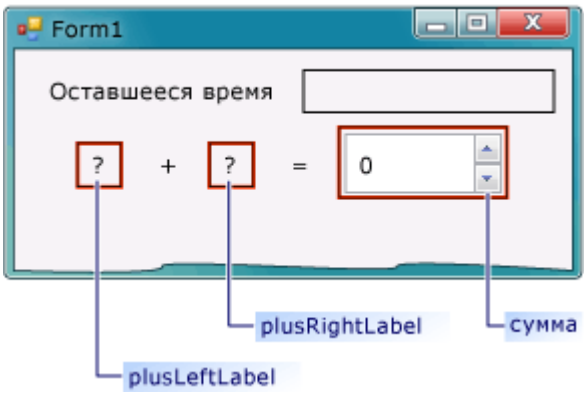
Окно свойств, в котором показан размер шрифта



6. Затем перетащите другой элемент управления Label с панели инструментов и задайте его свойства.
1. Измените размер шрифта на 15.75.
2. Задайте свойству Text значение равное Оставшееся время.
3. Перетащите ее, выравнивая по левой стороне метки timeLabel.
7. Сейчас добавьте элементы управления для задачи сложения. Перетащите элемент управления Label с панели инструментов и задайте его свойства.
1. Установите для свойства Text значение ? (вопросительный знак).
2. Задайте свойству AutoSize значение False.
3. Установите значение свойства Size равным 60, 50.
4. Измените размер шрифта на 18.
5. Измените значение свойства TextAlign на MiddleCenter.

6. Измените значение свойства Location на 75, 75 для размещения его на форме.
7. Измените значение свойства (Name) на plusLeftLabel.
8. Выделите метку plusLeftLabel и скопируйте ее. (нажмите сочетание клавиш CTRL+C или в меню Правка выберите пункт Копировать). Затем выполните следующие действия.
1. Вставьте ее три раза. (нажмите сочетание клавиш CTRL+V или в меню Правка выберите пункт Вставить).
2. Расположите три новых метки таким образом, чтобы поля размещались на строке справа от метки plusLeftLabel, используя линии-разделители для их разделения и выравнивания.
3. Измените у второй метки свойство Text на значение + (знак плюс).
4. Измените у третьей метки свойство(Name) на значение plusRightLabel.
5. Измените у четвертой метки свойство Text на значение = (знак равенства).
9. Перетащите элемент управления NumericUpDown control с панели элементов и затем выполните следующие действия.
1. Измените размер шрифта на 18, затем сделайте его уже, чтобы его ширина была равной 100.
2. Перетаскивайте его, пока он не станет вровень с элементами управления Label для задачи сложения.
3. Измените значение свойства (Name) на sum. (дополнительные сведения об элементе управления NumericUpDown приводятся далее). У головоломки сейчас есть первая строка, которая показана на рисунке ниже.

Первая строка математической головоломки



10. Выделите все пять элементов управления в задаче сложения (четыре элемента управления Label и элемент управления NumericUpDown) и скопируйте их. Затем выполните следующие действия.
1. Вставьте элементы управления, в результате на форму должны добавиться пять новых элементов управления.
2. Элементы управления по-прежнему будут выделены, поэтому можно щелкнуть по одному элементу управления и перетащить элементы управления в новое расположение так, чтобы они были выровнены под элементами управления сложения. Используйте линии разделители для обеспечения достаточного расстояния между двумя строками.
3. Измените у второй метки свойство Text на значение – (знак минус).
4. Назовите первую метку с вопросительным знаком как minusLeftLabel.
5. Назовите вторую метку с вопросительным знаком как minusRightLabel.
6. Назовите элемент управления NumericUpDown как difference.
11. Вставьте пять элементов управления еще два раза и затем выполните следующие действия.

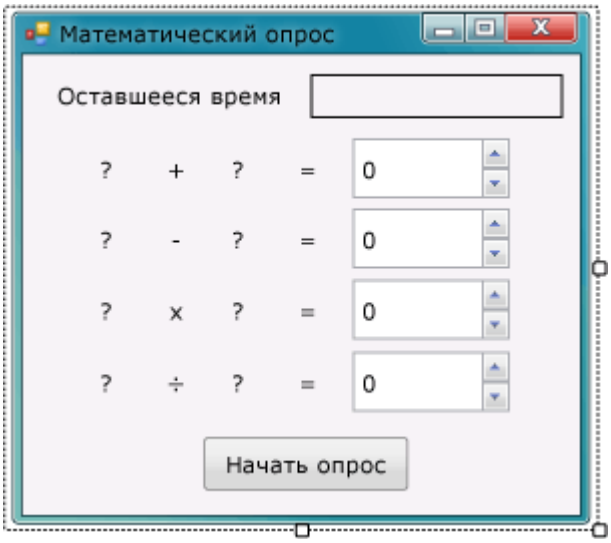
1. В третьей строке назовите первую метку как timesLeftLabel, измените у второй метки значение свойства Text на × (знак умножения), назовите третью метку как timesRightLabel, назовите элемент управления NumericUpDown как product.
2. В четвертой строке назовите первую метку как dividedLeftLabel, измените у второй метки значение свойства Text на ÷ (знак деления), назовите третью метку как dividedRightLabel, назовите элемент управления NumericUpDown как quotient.

Примечание

Знак умножения × и знак деления ÷ можно скопировать из этого руководства и вставить их в интегрированную среду разработки.

12. На форме нужен еще один элемент управления — кнопка для запуска головоломки. Перетащите элемент управления Button с панели инструментов и задайте его свойства.
1. Задайте для свойства (Name) значение startButton.
2. Задайте свойству Text значение Запуск головоломки.
3. Установите размер шрифта равным 14.
4. Установите для свойства AutoSize значение True, которое вызывает автоматическое изменение размера кнопки в зависимости от размера текста.
5. Перетащите кнопку вниз формы и разместите ее по центру.
13. В заключение выделите элемент управления startButton и выполните следующие действия.
1. Установите для свойства TabIndex значение 1.
2. Выделите элемент управления суммы NumericUpDown.
3. Установите для свойства TabIndex значение 2.
4. Установите значения для других элементов управления NumericUpDown. Для элемента управления разностью установите свойству TabIndex значение 3, для элемента управления произведением установите свойству TabIndex значение 4, для элемента управления частным установите свойству TabIndex значение 5. В результате форма должна выглядеть как на рисунке ниже.

Исходная форма математической головоломки



Примечание

Назначением свойства TabIndex является установка порядка элементов управления при нажатии пользователем клавиши TAB. Откройте любое диалоговое окно (например, в меню

Файл выберите пункт Открыть) и нажмите клавишу TAB несколько раз. Наблюдайте за тем, как курсор перемещается от одного элемента управления к другому при нажатии клавиши TAB. Когда форма была разработана изначально, программист установил этот порядок.

14. Чтобы посмотреть, как работает свойство TabIndex сохраните и запустите программу, затем несколько раз нажмите клавишу TAB.

Создание задачи на сложение случайных чисел

Для головоломки нужны математические задачи. Если головоломка повторяет одинаковые задачи, это не интересно, поэтому необходимо включить случайные числа. Был добавлен метод с именем **StartTheQuiz()**, который заполняет описание задач и запускает таймер обратного отсчета. На этом шаге добавляется задача случайного сложения. Другие математические задачи и таймер обратного отсчета добавляются в следующих шагах этого руководства.

В руководстве 2 было создано несколько компонентов SoundPlayer для игры "Лабиринт". Тоже самое делается для математической головоломки, за исключением того, что вместо класса **SoundPlayer** используется класс **Random**.

Создание задачи на сложение случайных чисел

1. Создайте объект **Random** с помощью оператора new, как показано в примере ниже.

C#

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

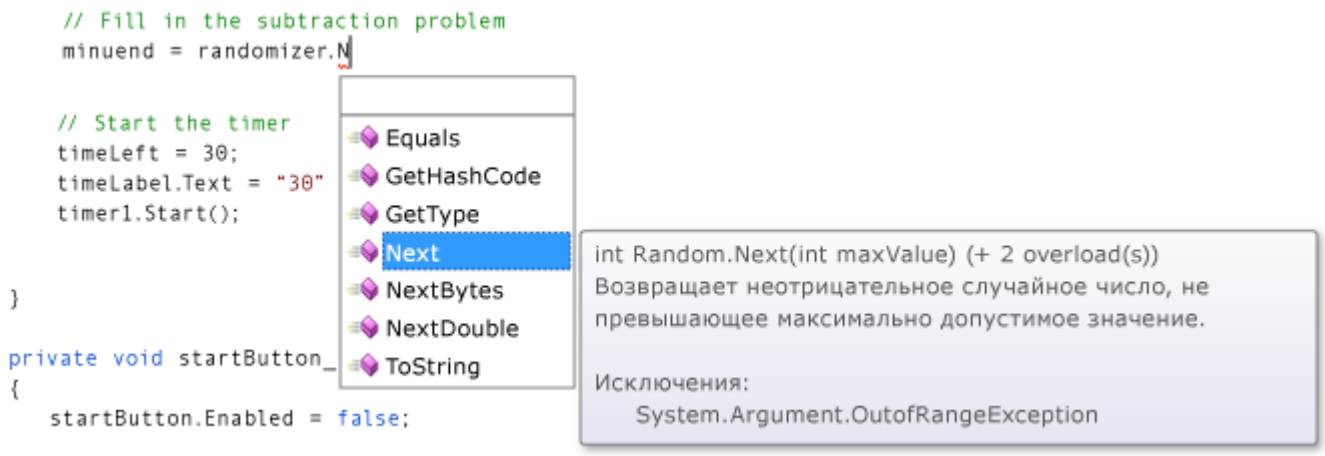
Сейчас в форму был добавлен объект **Random** с именем randomizer.

Примечание

В руководстве по игре "Лабиринт" было создано два компонента SoundPlayer с использованием оператора new. Сейчас объект был создан таким же образом. Единственное отличие от SoundPlayer состоит в том, что **Random** это не компонент и не элемент управления, поэтому его нельзя вызвать по такому имени. Он называется объектом. Вероятно вы слышали ранее слово "объект". Дополнительные сведения о том, что это значит приводятся в следующих нескольких руководствах. Сейчас все что нужно знать это то, что когда программа использует оператор new для создания кнопок, меток, панелей, компонентов OpenFileDialog, ColorDialog, SoundPlayer, Random и даже форм, элемент, который создается, называется объектом. В следующих руководствах более подробно рассказывается о работе объектов.

2. Сейчас в начале формы создается новый объект **Random** и ему присваивается имя randomizer. Как и для компонента SoundPlayer, если перейти в метод и начать вводить randomizer, а затем точку (.), откроется окно IntelliSense, в котором будут показаны все методы объекта **Random**, которые можно вызвать. Метод **Next()** используется следующим образом.

Метод Next



При вызове метода **random.Next(50)** возвращается случайное число, которое меньше 50 (от 0 до 49).

3. Вскоре будет создан метод для проверки ответов, поэтому программе нужно запомнить числа, которые она выбрала для задач. Добавьте в форму целое число (int в C# или Integer в Visual Basic) с именем addend1 и целое число int (Integer) с именем addend2 (также, как только что был добавлен объект **Random** с именем randomizer), как показано в примере ниже.

C#

```
// Create a Random object to generate random numbers.
Random randomizer = new Random();

// These ints will store the numbers
// for the addition problem.
int addend1;
int addend2;
```

Примечание

Целое число int (Integer) используется для хранения положительного или отрицательного значения числа. Оно может содержать любое целое число из диапазона от -2 147 483 648 до 2 147 483 647. Оно может хранить только целые числа, но не дроби.

4. Далее добавьте метод с именем **StartTheQuiz()**, который использует у объекта **Random** метод **Next()** для выбора двух чисел и размещения их в метках. В конечном итоге он заполнит все задачи и затем запустит таймер, поэтому добавьте комментарий. Код должен выглядеть следующим образом.

C#

```
/// <summary>
/// Start the quiz by filling in all of the problems
/// and starting the timer.
/// </summary>
public void StartTheQuiz()
```

```
{
   // Fill in the addition problem.
   addend1 = randomizer.Next(51);
   addend2 = randomizer.Next(51);

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

   sum.Value = 0;
}
```

Примечание

Обратите внимание, что вызывается метод **randomizer.Next(51)**. Используется число 51, а не 50, так как два числа добавляются к ответу, который находится в диапазоне от 0 до 100. Если в метод **Next()** передать число 50, то выбор выполняется из диапазона от 0 до 49, поэтому максимальный возможный ответ равен 98, а не 100. После того, как в методе выполняются два оператора, каждое из двух целых чисел int (Integer), addend1 и addend2 хранит случайное число в диапазоне от 0 до 50.

Более внимательно ознакомимся с этими операторами.

C#

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

Операторы устанавливают значение свойств Текст у двух меток сложения — plusLeftLabel и plusRightLabel, поэтому эти метки отображают два случайных числа. Нужно использовать метод **ToString()** для преобразования целых чисел в текст (в программировании string (строка) означает текст), так как элементы управления Label могут отображать только текст, а не числа.

5. Нужно, чтобы кнопка Пуск запускала головоломку, поэтому перейдите в конструктор Windows Forms и дважды щелкните по кнопке, чтобы добавить обработчик событий Click. Затем добавьте два следующих оператора.

C#

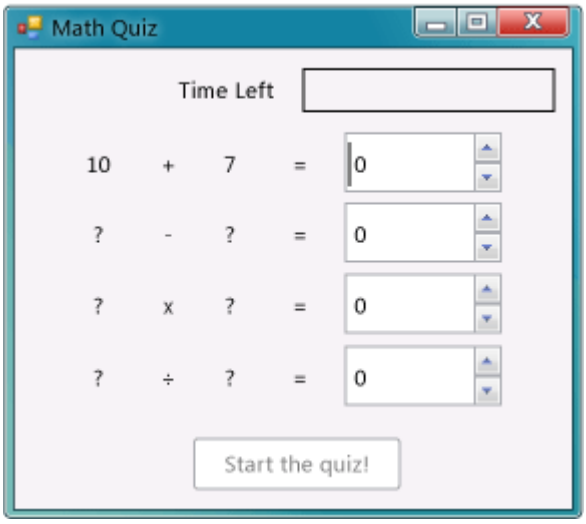
```
private void startButton_Click(object sender, EventArgs e)
{
   startButton.Enabled = false;
   StartTheQuiz();
}
```

Известно, что делает второй оператор. Он вызывает новый метод **StartTheQuiz()**. Первый оператор устанавливает у элемента управления startButton свойство Enabled значение равное False. Это отключает кнопку, поэтому пользователь не может ее нажать. Поэтому пользователь может нажать

на кнопку Пуск только один раз. После этого кнопка отображается затененной и недоступной, пользователь должен завершить головоломку до истечения времени (или закрыть программу).

6. Сохраните и выполните программу. Нажмите кнопку Пуск. Должна появиться задача сложения случайных чисел, как показано на рисунке ниже.

Задача сложения случайных чисел



Добавление таймера с обратным отсчетом

Так как головоломка ограничена по времени, будет добавлен таймер обратного отсчета. Программа должна отслеживать количество секунд, которое осталось до завершения игры.

Добавление таймера с обратным отсчетом

1. Добавьте целое число int (Integer) с именем timeLeft также, как это выполнялось ранее. Код должен выглядеть так, как показано ниже.

C#

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

   // These ints will store the numbers
   // for the addition problem.
   int addend1;
   int addend2;

   // This int will keep track of the time left.
   int timeLeft;
```

2. Теперь необходимо что-то, что фактически выполняет отсчет, например, таймер. Перейдите в конструктор Windows Forms и перетащите в форму с панели элементов (категория Компоненты)

элемент управления Timer. Элемент управления будет расположен в серой области в нижней части конструктора Windows Forms.

3. Щелкните по только что добавленному значку timer1 и установите для свойства Interval значение равное 1000. Это вызывает событие Tick каждую секунду. Чтобы добавить обработчик событий Tick, дважды щелкните по значку. Интегрированная среда разработки переключается на редактор кода и переходит к новому методу обработчика событий. Добавьте следующие операторы.

C#

```
private void timer1_Tick(object sender, EventArgs e)
{
   if (timeLeft > 0)
   {
      // 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;
      startButton.Enabled = true;
   }
}
```

В зависимости от указанного значения, таймер каждую секунду проверяет, истекло ли время путем проверки, больше ли значение целого числа int (Integer) с именем timeLeft чем 0. Если больше, то время еще осталось. Сначала таймер вычитает 1 из значения переменной timeLeft, затем он обновляет у элемента управления timeLabel свойство Text, чтобы показать пользователю, сколько секунд осталось.

Если времени не осталось, таймер останавливается и меняется текст в элементе управления timeLabel, чтобы он показывал Time's up! (Время вышло!). Появляется окно сообщения, говорящее пользователю, что игра закончена. Показывается ответ — в данном случае результат сложения значений переменной addend1 и переменной addend2. Свойство Enabled у элемента управления startButton имеет значение true, чтобы кнопка была снова доступна. С помощью этого пользователь может начать игру повторно.

Только что был добавлен оператор if else, в котором описывается, как программа принимает решение . Оператор if else представлен ниже.

C#

```
if (something your program will check)
{
   // statements that will get executed
   // if the thing that the program checked is true
```

```
}
else
{
   // statements that will get executed
   // if the thing that the program checked is NOT true
}
```

Внимательно посмотрите в добавленном операторе на блок else, в котором показан ответ на задачу вычитание.

C#

```
sum.Value = addend1 + addend2;
```

Как известно addend1 + addend2 выполняет сложение двух чисел вместе.Первая часть (sum.Value) использует свойство Value у элемента управления NumericUpDown, для отображения правильного ответа.Свойство Value также используется позднее, когда необходимо проверить ответы для головоломки.

Элемент управления NumericUpDown упрощает для пользователя ввод чисел, поэтому этот элемент управления используется для ответов на математические задачи.Так как все возможные ответы, это числа из диапазона от 0 до 100, остаются значения свойств Minimum и Maximum по умолчанию равные 0 и 100.Это приводит к тому, что пользователь в таком элементе управления может ввести только значения из диапазона от 0 до 100.Так как ответами могут быть только целые числа, значение свойства DecimalPlaces устанавливается равным 0 — это значит, что пользователь не может вводить дробные числа.Если необходимо разрешить пользователю вводить число 3.141, но не число 3.1415, можно установить свойству DecimalPlaces значение равное 3.

4. Добавьте три строки в конец метода **StartTheQuiz()**, чтобы код выглядел так, как показано ниже.

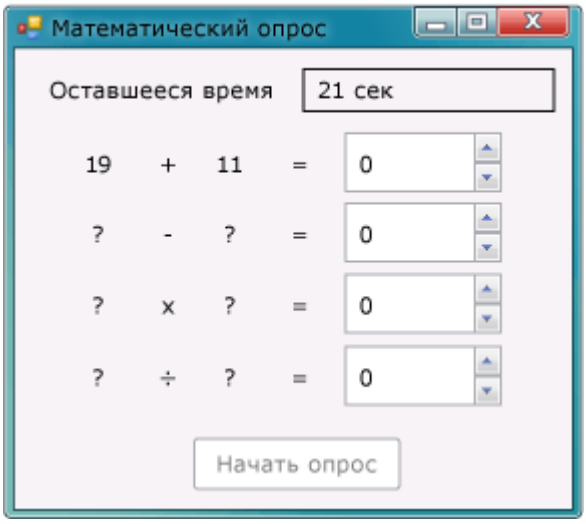
C#

```
/// <summary>
/// Start the quiz by filling in all of the problems
/// and starting the timer.
/// </summary>
public void StartTheQuiz()
{
   // Fill in the addition problem.
   addend1 = randomizer.Next(51);
   addend2 = randomizer.Next(51);
   plusLeftLabel.Text = addend1.ToString();
   plusRightLabel.Text = addend2.ToString();
   sum.Value = 0;
   // Start the timer.
   timeLeft = 30;
   timeLabel.Text = "30 seconds";
   timer1.Start();
}
```

Теперь при запуске головоломки он устанавливает целому числу int (Integer) с именем timeLeft значение 30, и изменяет у элемента управления timeLabel свойство Text на значение равное 30 секундам. Затем, чтобы начать обратный отсчет, он вызывает у элемента управления Timer метод **Start()** (ответ он пока не проверяет — это будет сделано позже).

5. Сохраните и выполните программу. При нажатии кнопки Пуск таймер должен начать обратный отсчет. Когда время истечет, игра закончится и появится ответ. На рисунке ниже показана головоломка в процессе игры.

Математическая головоломка в процессе игры



Добавление метода CheckTheAnswer()

Игра должна проверять, дает ли пользователь правильный ответ. К счастью, написание методов, которые выполняют простые вычисления, например, метода **CheckTheAnswer()**, не является сложной задачей.

Примечание

Для тех, кто разрабатывает игру на Visual Basic необходимо отметить, что поскольку метод возвращает значение, вместо обычного ключевого слова Sub будет использоваться ключевое слово Function. Это объясняется просто — процедуры не возвращают значения, в отличие от функций.

Добавление метода CheckTheAnswer()

1. Добавьте метод **CheckTheAnswer()**, который выполняет сложение переменной addend1 и переменной addend2 и выполняет проверку, что сумма равна значению sum элемента управления NumericUpDown. Если значение суммы равно значению sum метод возвращает значение true; в противном случае — false. Код должен выглядеть так, как показано ниже.

C#

```
/// <summary>
/// Check the answer to see if the user got everything right.
/// </summary>
/// <returns>True if the answer's correct, false otherwise.</returns>
```

```
private bool CheckTheAnswer()
{
   if (addend1 + addend2 == sum.Value)
      return true;
   else
      return false;
}
```

Программа должна вызвать этот метод для проверки, правильный ли ответ дал пользователь. Это выполняется путем добавления оператора if else. Оператор выглядит следующим образом.

C#

```
if (CheckTheAnswer())
{
    // statements that will get executed
    // if the answer is correct
}
else if (timeLeft > 0)
{
    // statements that will get executed
    // if there's still time left on the timer
}
else
{
    // statements that will get executed if the timer ran out
}
```

2. Затем выполняется изменение обработчика событий Tick для проверки ответа. Новый обработчик с проверкой ответа должен содержать следующее.

C#

```
private void timer1_Tick(object sender, EventArgs e)
{
   if (CheckTheAnswer())
   {
      // If 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)
   {
      // Decrease the time left by one second and display
      // the new time left by updating the Time Left label.
      timeLeft--;
      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;
        startButton.Enabled = true;
    }
}
```

Теперь если обработчик события таймера обнаруживает, что пользователь дал правильный ответ, обработчик событий останавливает таймер, показывает сообщение с поздравлением и делает снова доступной кнопку Пуск.

3. Сохраните и выполните программу. Начните игру, введите правильный ответ для задачи на сложение.

Примечание

При вводе ответа можно заметить некую странность в поведении элемента управления NumericUpDown. Если начать ввод ответа, не выделив при этом ответ полностью, нуль остается, его необходимо удалить вручную. Такое странное поведение будет исправлено в этом руководстве позднее.

4. При вводе правильного ответа должно открыться окно сообщений, кнопка Пуск должна стать доступной, таймер должен остановиться. Нажмите снова кнопку Пуск и убедитесь в этом.

Добавление обработчиков событий входа для элементов управления NumericUpDown

Возможно вы обратили внимание на некую странность при вводе чисел в элементе управления NumericUpDown. Чтобы исправить это, добавьте обработчик события Enter.

Просмотр поведения элемента управления NumericUpDown

1. Выполните программу и запустите игру. Сумма в элементе управления NumericUpDown должна содержать мигающий курсор рядом с 0 (нулем).
2. Введите 3 и появится значение 30. Введите 5 и появится значение 350, однако секунду спустя оно изменится на 100.

Перед исправлением ошибки подумайте, почему она возникла. Подумайте, почему не исчезает 0 при вводе значения 3. Подумайте, почему 350 меняется на 100 и почему возникает задержка при изменении.

Примечание

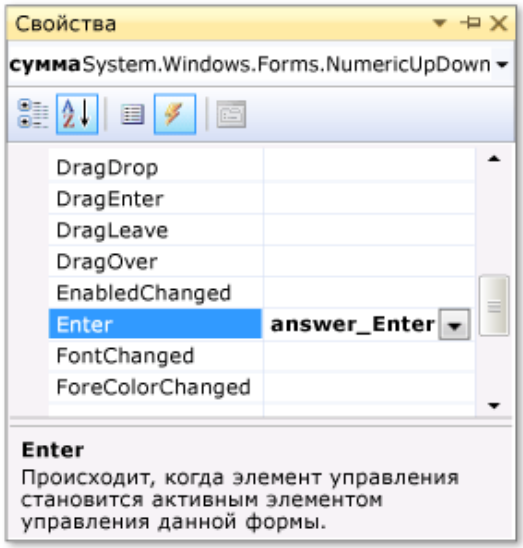
Хотя такое поведение кажется странным, этому есть объяснение. При нажатии на кнопку Пуск, ее свойство Enabled имеет значение False, кнопка отображается серым цветом и недоступна. Программа ищет элемент управления со следующим наименьшим значением

свойства TabIndex — сумма в элементе управления NumericUpDown — и меняет фокус к этому элементу управления. При использовании клавиши TAB для перехода к элементу управления NumericUpDown, происходит автоматическое размещение курсора в начале элемента управления, это приводит к тому, что вводимые числа отображаются слева, а не справа. При вводе числа, которое больше значения свойства MaximumValue, которое равно 100, выполняется замена введенного числа на максимальное значение.

Добавление обработчика событий Enter для элемента управления NumericUpDown

1. Чтобы предупредить такое странное поведение и сделать программу более удобной в использовании, добавьте обработчик событий для события Enter у всех элементов управления NumericUpDown. Чтобы добавить обработчик события Enter для элемента управления NumericUpDown с именем answer_Enter, используемого для отображения суммы, используйте страницу События в окне Свойства.

Диалоговое окно "Свойства"



Код должен выглядеть следующим образом.

C#

```
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);
   }
}
```

Хотя на первый взгляд он может показаться сложным, его легче понять, если просматривать его шаг за шагом. Сначала посмотрите на верхнюю часть метода — object sender в C# или sender As

System.Object в Visual Basic. Это значит, что внутри метода, когда используется sender, он указывает на элемент управления NumericUpDown у которого возникло событие Enter. Поэтому в первой строке метода указывается, что это не просто объект, а именно элемент управления NumericUpDown. (каждый элемент управления NumericUpDown это объект, но не каждый объект, это элемент управления NumericUpDown). В следующей строке кода выполняется проверка, что answerBox был успешно преобразован из объекта в элемент управления NumericUpDown. Если операция неуспешна, то он будет иметь значение null (C#) или Nothing (Visual Basic). В третьей строке кода выполняется определение длины ответа, который в настоящий момент отображается в элементе управления NumericUpDown. В четвертой строке указывается элементу управления NumericUpDown выбрать ответ. Теперь, когда пользователь переходит к элементу управления, он вызывает это событие, которое приводит к выбору ответа. Как только пользователь начинает ввод данных, предыдущий ответ стирается и заменяется на новый ответ.

2. После размещения обработчика событий перейдите в конструктор Windows Forms и выделите другой элемент управления NumericUpDown. Перейдите на страницу События в диалоговом окне Свойства, прокрутите содержимое окна до события Enter и выберите обработчик событий, который был только что добавлен.
3. Затем проделайте такое же действие для элементов управления NumericUpDown, в которых отображаются значения переменных product (произведения) и quotient (частного).
4. Сохраните и выполните программу. Странное поведение больше не должно появляться.

Добавление задачи на вычитание

Чтобы добавить задачу на вычитание необходимо следующее.

- Хранение значений, которые участвуют в операции вычитания.
- Создание случайных чисел для задачи (а также гарантия, что ответ лежит в диапазоне от 0 до 100).
- Обновление метода, который проверяет ответ таким образом, чтобы этот метод также проверял новую задачу на вычитание.
- Обновление обработчика событий таймера Tick таким образом, чтобы этот обработчик событий заполнял корректный ответ после истечения времени.

Добавление задачи на вычитание

1. Сначала необходимо место для хранения значений, поэтому добавьте два целых числа int (Integer) для задачи на вычитание в форму. Новый код добавляется между целыми числами задачи на сложение и целым числом для таймера. Код должен выглядеть следующим образом.

C#

> **Примечание**
>
> Имена новых целых чисел — minuend и subtrahend — это не термины Visual C# и не термины программирования. Это принятые арифметические обозначения для числа, которое вычитается из другого (subtrahend — вычитаемое) и числа, из которого производится вычитание (minuend — уменьшаемое). Остаток — это уменьшаемое за минусом вычитаемого. Можно использовать другие имена, так как программа не требует определенных имен для целых чисел, элементов управления, компонентов или методов. Существуют определенные правила, например, имена не могут начинаться с цифр. В общем случае можно использовать такие имена, как x1, x2, x3, x4 и так далее. Однако это затруднит чтение кода и сделает практически невозможным отслеживание ошибок. Далее в этом

руководстве для умножения и для деления будут использоваться принятые имена: multiplicand (множимое) × multiplier (множитель) = product (произведение); dividend (делимое) ÷ divisor (делитель) = quotient (частное).

2. Затем, измените метод **StartTheQuiz()** для заполнения задачи на вычитание случайных чисел. Новый код располагается под комментарием "Fill in the subtraction problem" (Заполнение задачи на вычитание). Код должен выглядеть следующим образом.

C#

```
/// <summary>
/// Start the quiz by filling in all of the problems
/// and starting the timer.
/// </summary>
public void StartTheQuiz()
{
    // Fill in the addition problem.
    addend1 = randomizer.Next(51);
    addend2 = randomizer.Next(51);
    plusLeftLabel.Text = addend1.ToString();
    plusRightLabel.Text = addend2.ToString();
    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;

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

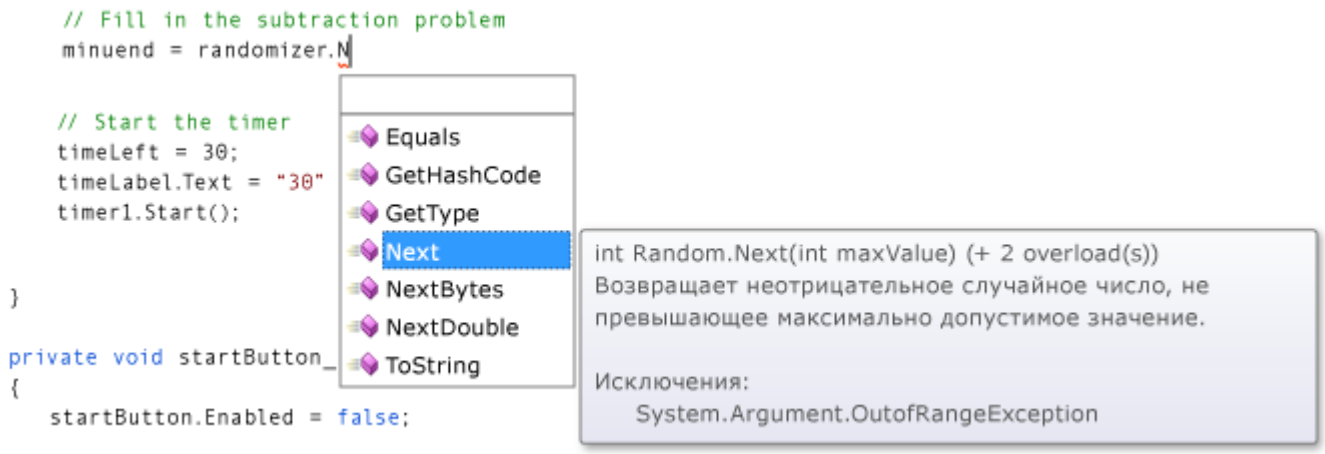
Этот код немного по другому использует у класса **Random** метод **Next()**. Когда ему передается два значения, он выбирает случайное число, которое меньше или равно первого значения и меньше второго значения. В следующей строке кода выполняется выбор случайного числа от 1 до 100 и производится сохранение его в переменной с именем minuend (уменьшаемое).

C#

```
minuend = randomizer.Next(1, 101);
```

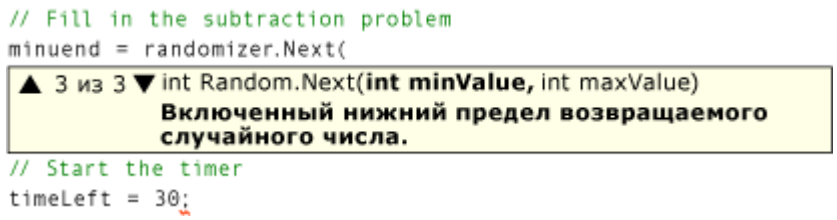
Метод **Next()** класса **Random** может быть вызван несколькими способами. Когда происходит вызов метода несколькими способами, то такой метод называется перегруженным методом, для его изучения можно использовать IntelliSense. Еще раз посмотрите на подсказку окна IntelliSense для метода **Next()**.

Подсказка окна Intellisense



Обратите внимание, что в подсказке указано (+ 2 overload(s)) (+ 2 перегрузки). Это значит, что существует два других способа, с помощью которых можно вызвать метод **Next()**. При вводе нового кода для метода **StartTheQuiz()** можно увидеть дополнительные сведения. Как только будет введено randomizer.Next(,, откроется окно IntelliSense. Чтобы переключиться между перегрузками, которые показаны на рисунке ниже, используйте клавиши СТРЕЛКА ВВЕРХ и СТРЕЛКА ВНИЗ.

Перегрузки в окне Intellisense



Нужна перегрузка, которая показана на рисунке выше, так как она позволяет указать минимальное и максимальное значение.

3. Для проверки правильного ответа для задачи на вычитание, измените метод **CheckTheAnswer()**. Код должен выглядеть следующим образом.

C#

```
/// <summary>
/// Check the answer to see if the user got everything right.
/// </summary>
/// <returns>True if the answer's correct, false otherwise.</returns>
private bool CheckTheAnswer()
{
   if ((addend1 + addend2 == sum.Value)
      && (minuend - subtrahend == difference.Value))
      return true;
   else
      return false;
}
```

Два знака && это оператор logical and (логическое И) в языке Visual C# . Эквивалентный оператор в языке Visual Basic — AndAlso. Это тоже самое, как сказать "Если переменную addend1 прибавить к переменной addend2 это равно значению переменной sum в элементе управления NumericUpDown,

И если из переменной minuend (уменьшаемое) вычесть переменную subtrahend (вычитаемое) это равно значению переменной difference (разность) элемента управления NumericUpDown". Метод **CheckTheAnswer()** только лишь возвращает значение true если задача на сложение решена правильно и задача на вычитание решена правильно.

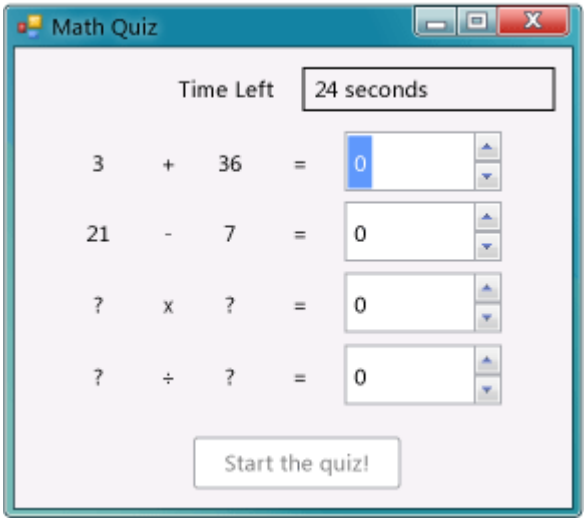
4. Измените последнюю часть обработчика событий таймера Tick таким образом, чтобы этот обработчик событий заполнял корректный ответ после истечения времени. Код должен выглядеть следующим образом.

C#

```
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;
   startButton.Enabled = true;
}
```

5. Сохраните и выполните код. Теперь программа должна содержать задачу на вычитание, как показано на рисунке ниже.

Математическая головоломка с задачей на вычитание



Добавление задач на умножение и деление

Перед тем как начать выполнение этой процедуры, подумайте о том, как будут добавлены задачи на умножение и на деление. Рассмотрим начальный шаг, который предполагает сохранение значений.

Добавление задач на умножение и деление

1. Добавьте в форму четыре целых числа int (Integer). Код должен выглядеть следующим образом.

C#

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

    // These ints will store the numbers for the addition problem.
    int addend1;
    int addend2;

    // These ints will store the numbers for the subtraction problem.
    int minuend;
    int subtrahend;

    // These ints will store the numbers for the multiplication problem.
    int multiplicand;
    int multiplier;

    // These ints will store the numbers for the division problem.
    int dividend;
    int divisor;

    // This int will keep track of the time left.
    int timeLeft;
```

2. Как это делалось ранее, измените метод **StartTheQuiz()** для заполнения задач на умножение и деление случайных чисел. Код должен выглядеть следующим образом.

C#

```
/// <summary>
/// Start the quiz by filling in all of the problems
/// and starting the timer.
/// </summary>
public void StartTheQuiz()
{
    // Fill in the addition problem.
    addend1 = randomizer.Next(51);
    addend2 = randomizer.Next(51);
    plusLeftLabel.Text = addend1.ToString();
    plusRightLabel.Text = addend2.ToString();
    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;

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

3. Измените метод **CheckTheAnswer()** таким образом, чтобы этот метод также проверял задачи на умножение и деление. Код должен выглядеть следующим образом.

C#

```
/// <summary>
/// Check the answer to see if the user got everything right.
/// </summary>
/// <returns>True if the answer's correct, false otherwise.</returns>
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;
}
```

Примечание

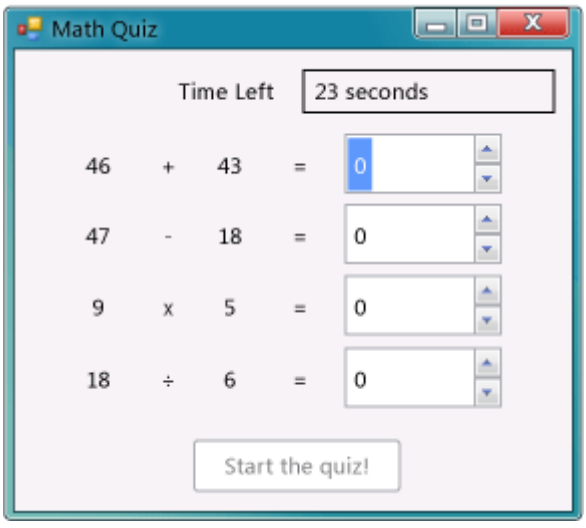
Так как нет простого способа ввести знак умножения (×) и знак деления (÷) с помощью клавиатуры, в языках Visual C# и Visual Basic используется знак звездочка (*) для умножения и знак косой черты (/) для деления.

4. Измените последнюю часть обработчика событий таймера Tick таким образом, чтобы этот обработчик событий заполнял корректный ответ после истечения времени. Код должен выглядеть следующим образом.

C#

```
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;
}
```

5. Сохраните и выполните программу. Теперь необходимо дать ответ на четыре задачи, чтобы решить головоломку, как показано на рисунке ниже. Математическая головоломка с четырьмя задачами



Изучение других функций

Подумайте, как программа создает задачу на деление случайных чисел. А также почему программа не создает задачу, в которой ответ — дробное число. В качестве дополнительного занятия попытайтесь изменить цвет элемента управления timeLabel и дать пользователю подсказку.

Изучение других функций

- Измените цвет элемента управления timeLabel на красный, когда у пользователя остается пять секунд. Для этого измените свойство BackColor у этого элемента управления (timeLabel.BackColor = Color.Red;). Восстановите цвет при завершении игры.
- Предоставьте пользователю подсказку с помощью воспроизведения звука при вводе правильного ответа в один из элементов управления NumericUpDown. (необходимо написать обработчик событий ValueChanged для каждого элемента управления, который срабатывает при каждом изменении пользователем значения элемента управления).

ПРАКТИЧЕСКАЯ РАБОТА № 5

Тема работы: Создание игры "Подбери пару!"

Цель работы: разработка приложения для реализации игры подбери пару

Ход работы:

В этом учебном руководстве вы создадите игру "Подбери пару!", в которой игрок должен подобрать пару скрытым значкам. Вы научитесь:

- хранить объекты, используя объект **List**,
- использовать цикл foreach в C# или цикл For Each в Visual Basic,
- отслеживать состояние формы с помощью ссылочных переменных,
- создавать обработчик событий, который можно использовать для нескольких объектов,
- включать таймер только один раз при запуске.

В результате ваша программа будет выглядеть так, как показано на следующем рисунке.

Игра, которую вы создадите в этом учебном руководстве

Создание проекта и добавление таблицы в форму

Первым шагом создания игры "Подбери пару!" является создание проекта и добавление таблицы в форму.

Создание проекта и добавление таблицы в форму

1. В меню Файл выберите команду Создать проект.
2. Если используется не Visual Studio Express, вначале необходимо выбрать язык. В списке Установленные шаблоны выберите C# или Visual Basic.
3. Щелкните значок Приложение Windows Forms, а затем введите в качестве имени MatchingGame.
4. Задайте свойства формы:
   1. Измените свойство формы Text на Подбери пару!.
   2. Измените размер на 550 пикселей по ширине и 550 пикселей по высоте с помощью свойства Size или путем перетаскивания границы, пока не увидите требуемый размер в нижнем левом углу интегрированной среды разработки (IDE).

5. Перетащите элемент управления TableLayoutPanel с панели инструментов и задайте его свойства:

1. Задайте свойству BackColor значение CornflowerBlue. (В палитре выберите вкладку Веб для просмотра названий цветов.)
2. Задайте свойству Dock значение Fill, щелкнув разворачивающуюся кнопку рядом со свойством, а затем щелкнув большую среднюю кнопку.
3. Нажмите треугольную кнопку в верхнем правом углу TableLayoutPanel для отображения меню задач.
4. Дважды щелкните Добавить строку для добавления двух дополнительных строк, а затем дважды щелкните Добавить столбец для добавления двух дополнительных столбцов.
5. Щелкните Изменить строки и столбцы для открытия окна Стили столбцов и строк. Выберите каждый из столбцов, нажмите кнопку Процент и задайте ширину каждого столбца равную 25 процентам от общей ширины. Затем в раскрывающемся списке в верхней части окна выберите Строки и задайте высоту каждой строки равную 25 процентам. Нажмите кнопку ОК.

Теперь в TableLayoutPanel должно быть шестнадцать одинаковых по размеру квадратных ячеек.

6. Убедитесь, что TableLayoutPanel выбран в редакторе формы. Пока он выбран, откройте панель элементов и дважды щелкните Label для добавления элемента управления Label в верхний левый квадрат. Теперь элемент управления Label должен быть выбран в интегрированной среде разработки. Задайте его свойства:

1. Задайте свойству BackColor значение CornflowerBlue.
2. Задайте свойству AutoSize значение False.
3. Задайте свойству Dock значение Fill.
4. Задайте свойству TextAlign значение MiddleCenter, щелкнув разворачивающуюся кнопку рядом со свойством, а затем щелкнув большую среднюю кнопку.
5. Щелкните свойство Font. Должна появиться кнопка с многоточием.
6. Нажмите кнопку с многоточием и выберите шрифт Webdings, размер 72, жирный.
7. Задайте свойству Text значение равное букве c.

Теперь в верхней левой ячейке TableLayoutPanel должен располагаться большой черный квадрат на синем фоне, который выравнивается по центру.

Примечание

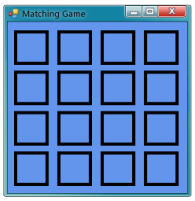
Шрифт Webdings является шрифтом значков, которые поставляются с операционной системой Microsoft Windows. В вашей игре "Подбери пару!" игроку необходимо найти пары одинаковых значков, поэтому используйте этот шрифт для отображения значков. Вместо ввода буквы c в свойство Text попробуйте вводить разные буквы, чтобы увидеть, какие значки отображаются. Восклицательный знак соответствует пауку, прописная буква N — глазу, а запятая — перцу чили.

7. Выберите ваш элемент управления Label и скопируйте его. (Нажмите сочетание клавиш CTRL+C или в меню Правка выберите команду Копировать). Затем вставьте его. (Нажмите сочетание клавиш CTRL+V или в меню Правка выберите команду Вставить). Во второй ячейке TableLayoutPanel появится еще одна метка. Вставьте его снова, и в третьей ячейке появится еще одна метка. Продолжайте вставлять элементы управления Label, пока все ячейки не будут заполнены.

Примечание

Если выполнить команду вставки слишком много раз, интегрированная среда разработки добавит новую строку в TableLayoutPanel, чтобы появилось место для добавления нового элемента управления Label. Можно выполнить откат. Для удаления новой ячейки нажмите сочетание клавиш CTRL+Z или в меню Правка выберите команду Отменить.

8. Теперь ваша форма готова. Она должна быть такой, как показано на следующем рисунке.
9. Исходная форма игры "Подбери пару!"



Добавление случайного объекта и списка значков

Необходимо использовать два оператора new для создания двух объектов и добавления их в форму. Первый является объектом **Random**. Такой же объект использовался в игре "Математическая головоломка". Второй является новым объектом **List**.

Добавление случайного объекта и списка значков

1. Перед добавлением следующего кода для создания списка разберитесь в принципе его работы.

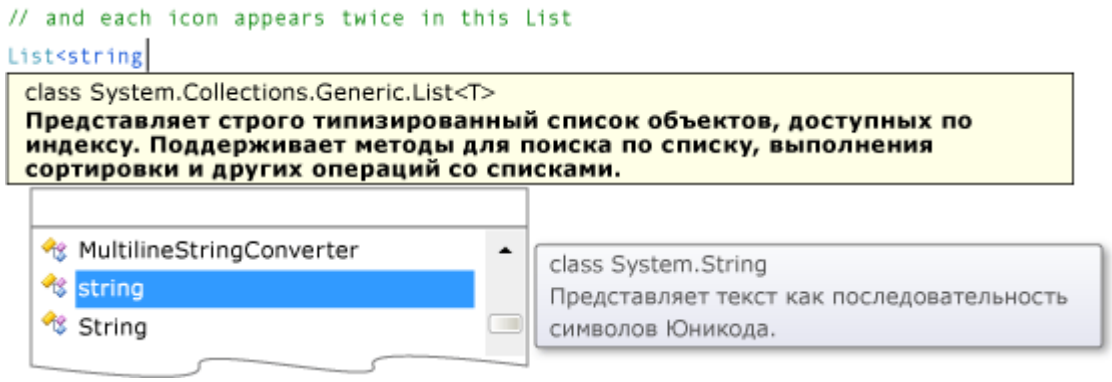
C#

```
public partial class Form1 : Form
{
   // Use this Random object to choose random icons for the squares
   Random random = new Random();

   // Each of these letters is an interesting icon
   // in the Webdings font,
   // and each icon appears twice in this list
   List<string> icons = new List<string>()
   {
      "!", "!", "N", "N", ",", ",", "k", "k",
      "b", "b", "v", "v", "w", "w", "z", "z"
   };
```

2. Перейдите в редактор кода, щелкнув правой кнопкой мыши Form1.cs в Обозревателе решений и выбрав в меню команду Перейти к коду. Начните вводить код, показанный в предыдущем шаге. При написании кода на языке Visual C# убедитесь, что вы помещаете код после открывающей фигурной скобки и сразу после объявления класса (public partial class Form1 : Form). При написании кода на языке Visual Basic поместите код сразу после объявления класса (Public Class Form1).

3. При добавлении объекта **List** внимательно изучите открывшееся окно IntelliSense. Ниже приведен пример на языке Visual C#. (Аналогичный текст появится при добавлении списка в Visual Basic.)

Окно IntelliSense



Примечание

Код проще понять, если рассматривать его небольшими частями. В ваших программах могут использоваться объекты **List** для отслеживания множества элементов. Список может содержать числа, значения true или false, текст и другие объекты. Также объект **List** может содержать другие объекты **List**. Список состоит из элементов, и каждый список содержит только один тип элементов. Поэтому список чисел может содержать только числа. В него нельзя добавить текст. Также нельзя добавлять числа в список значений true или false.

Примечание

При создании объекта **List** с помощью оператора new необходимо указать ему, какие данные будут храниться в этом объекте. Вот почему подсказка в верхней части окна IntelliSense отображает тип элементов списка. Вот что значит код List<string> (в Visual C#) и List(Of String) (в Visual Basic) — это объект **List**, содержащий строки. Строка используется программой для хранения текста, о чем и сообщается в подсказке справа от окна IntelliSense.

4. Рассмотрим, почему в Visual Basic сначала необходимо создать временный массив, а в Visual C# можно создавать список с помощью одного оператора. Это обусловлено тем, что в языке Visual C# имеются инициализаторы коллекции. В Visual Basic 2010 можно использовать инициализатор коллекции. Однако для обеспечения совместимости с предыдущей версией Visual Basic рекомендуется использовать предыдущий код.

Примечание

При использовании инициализатора коллекции с оператором new после создания нового объекта **List** программа заполняет его данными, которые указаны внутри фигурных скобок. В этом случае будет инициализирован список строк с именем icons, содержащий шестнадцать строк. Каждая из этих строк представляет одну букву, которая соответствует значку, отображаемому в метке. Таким образом, игра будет иметь пару восклицательных знаков, пару прописных букв N, пару запятых и т. д. Объект **List** будет содержать шестнадцать строк, по одной для каждой ячейки в TableLayoutPanel.

Примечание

В Visual Basic получится такой же результат, но сначала строки помещаются во временный массив, который затем преобразуется в объект **List**. Массив похож на список, за исключением того, например, что массивы создаются фиксированного размера. Списки можно уменьшать и увеличивать по мере необходимости, что важно для этой программы.

Назначение каждой метке случайного значка

Если игра всегда скрывает одни и те же значки в тех же местах, то она становится неинтересной. Необходимо случайным образом назначать значки элементам управления Label в форме. Для этого добавьте метод **AssignIconsToSquares()**.

Назначение каждой метке случайного значка

1. Перед добавлением следующего кода разберитесь в принципе его работы. В C# есть новое ключевое слово foreach, а в Visual Basic — For Each. (Одна из строк закомментирована по причине, описанной в конце этой процедуры.)

C#

```
/// <summary>
/// Assign each icon from the list of icons to a random square
/// </summary>
private void AssignIconsToSquares()
{
   // The TableLayoutPanel has 16 labels,
   // and the icon list has 16 icons,
   // so an icon is pulled at random from the list
   // and added to each label
   foreach (Control control in tableLayoutPanel1.Controls)
   {
      Label iconLabel = control as Label;
      if (iconLabel != null)
      {
         int randomNumber = random.Next(icons.Count);
         iconLabel.Text = icons[randomNumber];
         // iconLabel.ForeColor = iconLabel.BackColor;
         icons.RemoveAt(randomNumber);
      }
   }
}
```

2. Добавьте метод **AssignIconsToSquares()**, как показано в предыдущем шаге. Можно поместить его сразу же после кода, добавленного в разделе Шаг 2. Добавление случайного объекта и списка значков.

В методе **AssignIconsToSquares()** имеется некоторое нововведение: цикл foreach в C# и For Each в Visual Basic. Цикл foreach можно использовать в любое время для неоднократного выполнения одно и того же действия. В данном случае требуется выполнять одни и те же операторы для каждой метки в TableLayoutPanel, как показано в следующем коде.

C#

```
foreach (Control control in tableLayoutPanel1.Controls)
{
   // The statements you want to execute
   // for each label go here
   // The statements use iconLabel to access
   // each label's properties and methods
}
```

Примечание

Используются имена iconLabel и control, поскольку они являются описательными. Можно изменить эти имена на другие. Это не повлияет на их работу (при условии, что будут изменены имена в каждом операторе внутри фигурных скобок).

Метод **AssignIconsToSquares()** проходит через каждый элемент управления Label в TableLayoutPanel и выполняет одни и те же операторы для каждого из них. Эти операторы запрашивают случайные значки из списка, добавленного в разделе Шаг 2. Добавление случайного объекта и списка значков. (Вот почему в список включено по два значка. Это позволяет назначить пару значков случайным элементам управления Label.)

3. Сразу же после запуска программы необходимо вызвать метод **AssignIconsToSquares()**. При написании кода на языке Visual C# добавьте оператор после вызова метода **InitializeComponent()** в конструкторе Form1. Таким образом форма будет вызывать ваш новый метод для его настройки перед отображением.

C#

```
public Form1()
{
   InitializeComponent();

   AssignIconsToSquares();
}
```

Для Visual Basic сначала добавьте конструктор, а затем добавьте вызов метода в конструктор. Перед только что созданным методом **AssignIconsToSquares()** введите код Public Sub New(). При нажатии клавиши ВВОД для перехода к следующей строке функция IntelliSense должна отобразить следующий код для завершения конструктора.

4. -Сохраните и выполните программу. Должна отобразиться форма со случайными значками, которые назначены каждой метке.
5. Закройте программу, а затем снова запустите ее. Теперь другие значки назначены каждой метке, как показано на следующем рисунке.

Игра "Подбери пару!" со случайными значками

6. Теперь остановите программу и раскомментируйте следующую строку кода внутри цикла foreach.

C#

VB

```
iconLabel.ForeColor = iconLabel.BackColor;
```

7. Нажмите кнопку панели инструментов Сохранить все для сохранения программы и снова запустите ее. Похоже, что значки исчезли. Отображается только голубой фон. Однако значки назначены случайным образом и по-прежнему существуют. Поскольку значки того же цвета, что и фон, они невидимы.

Добавление к каждой метке обработчика событий щелчка мышью

Игра "Подбери пару!" работает следующим образом.

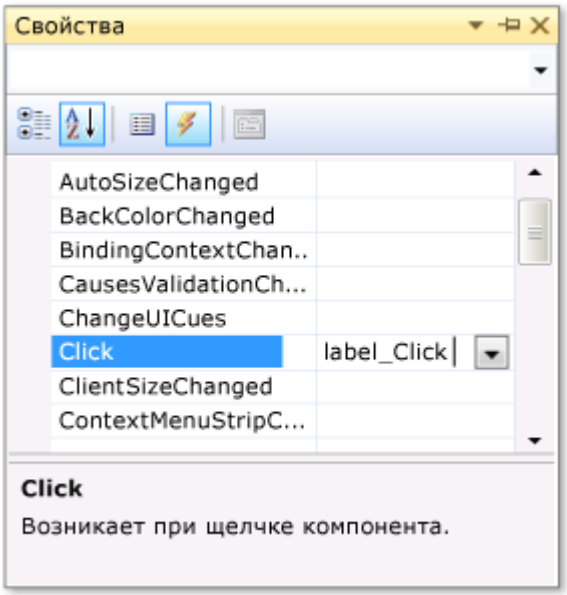
1. Когда игрок щелкает по одному из квадратов со скрытым значком, программа отображает значок игроку, изменяя цвет значка на черный.
2. После этого игрок щелкает другой скрытый значок.
3. Если значки совпадают, они остаются видимыми. Если нет, оба значка снова скрываются.

Чтобы программа работала таким образом, добавьте обработчик событий Click, изменяющий цвет метки, которую щелкнули.

Добавление обработчика событий Click к каждой метке

1. Перейдите к конструктору Windows Forms и щелкните первый элемент управления Label, чтобы выбрать его. Затем, удерживая клавишу CTRL, щелкните каждую из остальных меток для их выбора. Убедитесь, что выбраны все метки.
2. Затем перейдите на страницу События в окне Свойства. Прокрутите вниз до события Click и введите label_Click в поле, как показано на следующем рисунке.

Окно свойств, отображающее событие Click



3. Нажмите клавишу ВВОД. Интегрированная среда разработки добавляет обработчик событий Click, который называется **label_Click()**, в код и подключает его к каждой метке.
4. Добавьте остальную часть кода следующим образом:

C#

VB

```
/// <summary>
/// Every label's Click event is handled by this event handler
/// </summary>
/// <param name="sender">The label that was clicked</param>
/// <param name="e"></param>
private void label_Click(object sender, EventArgs e)
{
   Label clickedLabel = sender as Label;

   if (clickedLabel != null)
   {
      // If the clicked label is black, the player clicked
      // an icon that's already been revealed --
      // ignore the click
      if (clickedLabel.ForeColor == Color.Black)
         return;

      clickedLabel.ForeColor = Color.Black;
   }
}
```

Примечание

Возможно, вы узнали object sender в верхней части обработчика событий из руководства "Создание математической головоломки". Вы подключили различные события Click элемента управления Label к одному методу обработчика событий, поэтому один и тот же метод вызывается независимо от того, какую метку щелкнет пользователь. Методу требуется знать, какую метку щелкнули, поэтому для элемента управления Label используется имя sender. Первая строка метода сообщает программе, что это не просто объект, а элемент управления Label, для доступа к свойствам и методам которого используется имя clickedLabel.

Этот метод сначала проверяет, было ли успешно выполнено преобразование (приведение) clickedLabel из объекта в элемент управления Label. Если операция завершилась неудачей, его значение будет null (C#) или Nothing (Visual Basic) и оставшаяся часть кода метода не будет выполнена. Затем метод с помощью свойства ForeColor проверяет цвет текста метки, которую щелкнули. Если она уже черная, значит значок щелкнули и, следовательно, метод выполнен. (Это и делает оператор возврата. Он сообщает программе, что требуется остановить выполнение метода.) Если значок еще не был щелкнут, его цвет изменится на черный.

5. Сохраните и выполните программу. Вы должны увидеть пустую форму с синим фоном. Щелкните в форме, и один из значков должен отобразиться. Продолжайте щелкать в разных местах формы. При щелчках значков они должны отображаться.

Добавление ссылок на метки

Программе необходимо отслеживать, какой элемент управления Label щелкнул игрок. После щелчка первой метки программа показывает ее значок. После щелчка второй метки программа должна показать оба значка на короткое время, а затем снова их скрыть. Программа будет отслеживать с помощью ссылочных переменных, какой элемент управления Label щелкнули первым, а какой вторым.

Добавление ссылок на метки

1. Добавьте ссылки на метки в свою форму, используя следующий код.

C#

VB

```
public partial class Form1 : Form
{
   // firstClicked points to the first Label control
   // that the player clicks, but it will be null
   // if the player hasn't clicked a label yet
   Label firstClicked = null;

   // secondClicked points to the second Label control
   // that the player clicks
   Label secondClicked = null;
```

Примечание

Ссылочные переменные похожи на операторы, которые вы использовали для добавления объектов (таких как объекты **Timer**, **List** и **Random**) в форму. Однако эти операторы не приводят к появлению в форме двух дополнительных элементов управления Label, поскольку у них отсутствует оператор new. Без оператора new объект не создается. Вот почему firstClicked и secondClicked называются ссылочными переменными — они просто отслеживают (или ссылаются на) объекты **Label**.

Примечание

Когда переменная не отслеживает объект, ей задается специальное значение — null в Visual C# и Nothing в Visual Basic. Поэтому при запуске программы переменным firstClickedи secondClicked задано значение null или Nothing. Это означает, что переменные ничего не отслеживают.

2. Измените свой обработчик событий Click для использования новой ссылочной переменной firstClicked. Удалите последний оператор (clickedLabel.ForeColor = Color.Black;) в методе обработчика событий **label_Click()** и замените его последующим оператором if. (Убедитесь, что вы добавили комментарии и весь код, который находится между фигурными скобками.)

C#

VB

```
/// <summary>
/// Every label's Click event is handled by this event handler
/// </summary>
/// <param name="sender">The label that was clicked</param>
/// <param name="e"></param>
private void label_Click(object sender, EventArgs e)
{
   Label clickedLabel = sender as Label;

   if (clickedLabel != null)
   {
      // If the clicked label is black, the player clicked
      // an icon that's already been revealed --
      // ignore the click
      if (clickedLabel.ForeColor == Color.Black)
         return;

      // If firstClicked is null, this is the first icon
      // in the pair that the player clicked,
      // so set firstClicked to the label that the player
      // clicked, change its color to black, and return
      if (firstClicked == null)
      {
         firstClicked = clickedLabel;
         firstClicked.ForeColor = Color.Black;
```

```
            return;
        }
    }
}
```

3. Сохраните и выполните программу. Щелкните один из элементов управления Label и отобразится его значок.
4. Щелкните следующий элемент управления Label и обратите внимание, что ничего не происходит. Программа уже отслеживает первую метку, которую щелкнул игрок, поэтому firstClicked не равно null в Visual C# или Nothing в Visual Basic. Когда оператор if проверяет, имеет ли переменная firstClicked значение null или Nothing, он обнаруживает, что это не так, и не выполняет операторы в блоке if. Поэтому только первый значок, который щелкнули, становится черным, а другие значки остаются невидимыми, как показано на следующем рисунке.

Игра "Подбери пару!", отображающая один значок

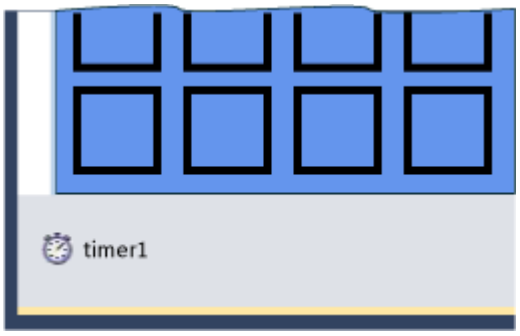
Добавление таймера

Далее, добавьте в игру "Подбери пару!" таймер.

Добавление таймера

1. Перейдите к панели элементов в конструкторе Windows Forms. Дважды щелкните Timer (в категории Компоненты) и добавьте в форму таймер. Его значок появится в сером поле под формой, как показано на следующем рисунке.

Таймер



2. Щелкните значок timer1 для выбора таймера. Задайте свойству Interval значение 750, но оставьте свойству Enabled значение False. Свойство Interval задает таймеру время ожидания между тактами, тем самым оно указывает таймеру подождать три четверти секунды (750 миллисекунд), прежде чем запустить свое первое событие Tick. Вы не хотите запускать таймер при запуске программы. Вместо этого воспользуйтесь методом **Start()** для запуска таймера при нажатии игроком второй метки.
3. Дважды щелкните значок элемента управления Timer в конструкторе Windows Forms для добавления обработчика событий Tick, как показано в следующем коде.

C#

VB

```
/// <summary>
/// This timer is started when the player clicks
/// two icons that don't match,
/// so it counts three quarters of a second
/// and then turns itself off and hides both icons
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void timer1_Tick(object sender, EventArgs e)
{
   // Stop the timer
   timer1.Stop();

   // Hide both icons
   firstClicked.ForeColor = firstClicked.BackColor;
   secondClicked.ForeColor = secondClicked.BackColor;

   // Reset firstClicked and secondClicked
   // so the next time a label is
   // clicked, the program knows it's the first click
   firstClicked = null;
   secondClicked = null;
}
```

Обработчик события Tick выполняет три действия. Сначала он останавливает таймер, вызывая метод **Stop()**. Затем он использует две ссылочные переменные, firstClicked и secondClicked, чтобы

снова сделать невидимыми значки двух меток, которые щелкнул игрок. Наконец, он сбрасывает значения ссылочных переменных firstClicked и secondClicked на null в Visual C# и Nothing в Visual Basic. Это важно, поскольку таким образом программа сбрасывает саму себя. Теперь она не отслеживает какие-либо элементы управления Label и снова готова к первому щелчку игрока.

Примечание

У объекта **Timer** есть метод **Start()**, который запускает таймер, и метод **Stop()**, который останавливает его. Если в окне Свойства свойству таймера Enabled задать значение True, он начнет отсчет сразу же, как только запустится программа. Но если в этом свойстве оставить значение False, отсчет не начнется, пока не будет вызван метод **Start()**.

Примечание

Как правило, таймер запускает событие Tick снова и снова, используя свойство Interval, чтобы определить, сколько миллисекунд ждать между тактами. Обратите внимание на вызов метода таймера **Stop()** внутри события Tick. Таймер переходит в режим однократного действия, поэтому при вызове метода **Start()** он ожидает в течение своего интервала, а затем запускает одно событие Tick.

4. Чтобы увидеть работу нового таймера, перейдите к редактору кода и добавьте следующий код в начало и конец метода обработчика событий **label_Click()**. (Добавляется оператор if в начало метода и три оператора в конец. Остальная часть метода остается неизменной.)

C#

VB

```
/// <summary>
/// Every label's Click event is handled by this event handler
/// </summary>
/// <param name="sender">The label that was clicked</param>
/// <param name="e"></param>
private void label_Click(object sender, EventArgs e)
{
   // The timer is only on after two non-matching
   // icons have been shown to the player,
   // so ignore any clicks if the timer is running
   if (timer1.Enabled == true)
      return;

   Label clickedLabel = sender as Label;

   if (clickedLabel != null)
   {
      // If the clicked label is black, the player clicked
      // an icon that's already been revealed --
      // ignore the click
      if (clickedLabel.ForeColor == Color.Black)
         return;
```

```
        // If firstClicked is null, this is the first icon
        // in the pair that the player clicked,
        // so set firstClicked to the label that the player
        // clicked, change its color to black, and return
        if (firstClicked == null)
        {
            firstClicked = clickedLabel;
            firstClicked.ForeColor = Color.Black;
            return;
        }

        // If the player gets this far, the timer isn't
        // running and firstClicked isn't null,
        // so this must be the second icon the player clicked
        // Set its color to black
        secondClicked = clickedLabel;
        secondClicked.ForeColor = Color.Black;

        // If the player gets this far, the player
        // clicked two different icons, so start the
        // timer (which will wait three quarters of
        // a second, and then hide the icons)
        timer1.Start();
    }
}
```

Код в начале метода проверяет, запущен ли таймер, обращаясь к свойству Enabled. Таким образом, если игрок щелкнет первый и второй элемент управления Label и таймер запуститься, щелчок третьего элемента управления ни к чему не приведет.

Код в конце метода задает ссылочную переменную secondClicked таким образом, чтобы она отслеживала второй элемент управления Label, который щелкнул игрок, и задает этой метке черный цвет значка, чтобы сделать ее видимой. Затем таймер запускается в однократном режиме, то есть ожидает 750 миллисекунд и после этого запускает событие Tick. Далее обработчик событий Tick таймера скрывает два значка и сбрасывает значения ссылочных переменных firstClicked и secondClicked, таким образом форма готова к щелчку игроком другого значка.

5. Сохраните и выполните программу. Щелкните значок, и он станет видимым.
6. Щелкните другой значок. Он появляется на короткое время, а затем оба значка исчезают. Повторите это несколько раз. В форме отслеживаются первый и второй значок, которые вы щелкнули, и используется таймер для паузы перед исчезновением значков.

Отмена исчезновения пар значков

Игра работает хорошо, пока игрок щелкает только пары значков, которые не совпадают. Давайте рассмотрим, что произойдет, когда игрок щелкнет совпадающую пару. Вместо того, чтобы значки исчезали благодаря включению таймера (с помощью метода **Start()**), игра должна сбрасываться и больше не отслеживать метки с помощью ссылочных переменных firstClicked и secondClicked. Но цвета двух меток, которые вы щелкнули, не должны сбрасываться.

Отмена исчезновения пар значков

1. Добавьте следующий оператор if в конец метода обработчика событий **label_Click()** перед оператором, который запускает таймер. Внимательно просмотрите код при добавлении его в программу. Рассмотрим, как работает код.

C#

VB

```
        // If the player gets this far, the timer isn't
        // running and firstClicked isn't null,
        // so this must be the second icon the player clicked
        // Set its color to black
        secondClicked = clickedLabel;
        secondClicked.ForeColor = Color.Black;

        // If the player clicked two matching icons, keep them
        // black and reset firstClicked and secondClicked
        // so the player can click another icon
        if (firstClicked.Text == secondClicked.Text)
        {
            firstClicked = null;
            secondClicked = null;
            return;
        }

        // If the player gets this far, the player
        // clicked two different icons, so start the
        // timer (which will wait three quarters of
        // a second, and then hide the icons)
        timer1.Start();
    }
}
```

Первая строка только что добавленного оператора if проверяет, совпадает ли значок в первой метке, которую щелкнул игрок, со значком во второй метке. Если значки совпадают, программа выполняет три оператора, которые расположены между фигурными скобками в C# или внутри оператора if в Visual Basic. Первые два оператора сбрасывают ссылочные переменные firstClicked и secondClicked, чтобы они не отслеживали какие-либо метки. (Возможно, вы узнали эти два оператора из обработчика событий Tick таймера.) Третий оператор — оператор возврата, который сообщает программе, что требуется пропустить и не выполнять оставшиеся в методе операторы.

При программировании в Visual C# вы могли обратить внимание, что в некоторых местах кода используется одиночный знак равенства (=), а в других — двойной знак равенства (==). Рассмотрим, почему в одних местах используется =, а в других — ==.

Вот хороший пример, показывающий разницу. Внимательно посмотрите на код между скобками в операторе if.

VB

firstClicked.Text = secondClicked.Text

C#

firstClicked.Text == secondClicked.Text

Затем внимательно посмотрите на первый оператор в блоке кода после оператора if.

VB

firstClicked = Nothing

C#

firstClicked = null;

Первый из этих двух операторов проверяет, являются ли два значка одинаковыми.Поскольку два значения сравниваются, программа Visual C# использует оператор равенства ==.Второй оператор, на самом деле, изменяет значение (это называется присваивание), задавая ссылочной переменной firstClicked значение null, чтобы ее сбросить.Вот почему здесь используется оператор присваивания =.В Visual C# для задания значений используется оператор =, а оператор == используется для их сравнения.В Visual Basic оператор = используется для присваивания и сравнения.

2. Сохраните и выполните программу, а затем начните щелкать в форме. Если вы щелкните пару, которая не совпадает, произойдет событие таймера Tick и оба значка исчезнут. Если вы щелкните пару, которая совпадает, выполнится новый оператор if, а оператор возврата укажет методу пропустить код, запускающий таймер, чтобы значки остались видимыми, как показано на следующем рисунке.

Игра "Подбери пару!" с видимыми парами значков

Добавление метода для проверки, выиграл ли игрок

Вы создали интересную игру, но ей требуется еще один элемент. Игра должна заканчиваться победой игрока, поэтому необходимо добавить метод **CheckForWinner()** для проверки, выиграл ли игрок.

Добавление метода для проверки, выиграл ли игрок

1. Добавьте метод **CheckForWinner()** в вашу форму, как показано в следующем коде.

C#

```
/// <summary>
/// Check every icon to see if it is matched, by
/// comparing its foreground color to its background color.
/// If all of the icons are matched, the player wins
/// </summary>
private void CheckForWinner()
{
   // Go through all of the labels in the TableLayoutPanel,
   // checking each one to see if its icon is matched
   foreach (Control control in tableLayoutPanel1.Controls)
   {
      Label iconLabel = control as Label;

      if (iconLabel != null)
      {
         if (iconLabel.ForeColor == iconLabel.BackColor)
            return;
      }
   }

   // If the loop didn’t return, it didn't find
   // any unmatched icons
   // That means the user won. Show a message and close the form
   MessageBox.Show("You matched all the icons!", "Congratulations");
   Close();
}
```

В этом методе используется еще один цикл foreach для C# или цикл For Each для Visual Basic, чтобы пройти по каждой метке в TableLayoutPanel. Он использует оператор равенства (== в Visual C# и = в Visual Basic) для проверки цвета значка каждой метки на соответствие цвету фона. Если цвета совпадают, значок остается невидимым, а значит игрок не подобрал пару оставшимся значкам. В этом случае программа использует оператор return, чтобы пропустить оставшуюся часть метода. Если цикл прошел через все метки без выполнения оператора return, значит всем значкам была подобрана пара. Программа отображает MessageBox, а затем вызывает метод формы **Close()** для завершения игры.

2. После этого обработчик событий Click метки вызывает новый метод **CheckForWinner()**. Убедитесь, что программа проверяет наличие победителя после отображения второго значка, который щелкает игрок. Найдите строку, где задается цвет второму значку, который вы щелкнули, и после этой операции вызовите метод **CheckForWinner()**, как показано в следующем коде.

C#

```
// If the player gets this far, the timer isn't
// running and firstClicked isn't null,
// so this must be the second icon the player clicked
```

```
// Set its color to black
secondClicked = clickedLabel;
secondClicked.ForeColor = Color.Black;
// Check to see if the player won
CheckForWinner();
// If the player clicked two matching icons, keep them
// black and reset firstClicked and secondClicked
// so the player can click another icon
if (firstClicked.Text == secondClicked.Text)
{
    firstClicked = null;
    secondClicked = null;
    return;
}
```

3. Сохраните и выполните программу. Сыграйте в игру и подберите пару всем значкам. Если вы победили, программа отображает MessageBox (как показано на следующем рисунке) и закрывает окно.

Игра "Подбери пару!" с MessageBox

Изучение других функций

Для изучения других функций попробуйте изменить значки и цвета, добавить сетку или добавить звуки. Чтобы сделать игру интереснее, увеличьте игровое поле или измените настройки таймера.

Изучение других функций

- Замените значки и цвета на другие.
- Добавьте сетку, которая будет появляться вокруг значков.
- Добавьте звуки, которые будут воспроизводиться при нахождении игроком пары, отображении двух несовпадающих значков и повторном сокрытии значков программой.
- Сделайте игру труднее, увеличив игровое поле. (Подсказка. Необходимо сделать больше, чем просто добавить строки и столбцы в TableLayoutPanel.)
- Сделайте игру интереснее, скрывая первый значок, если игрок медлителен и не щелкает второй значок в течение отведенного времени.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Парта С. Язык программирования C++. Лекции и упражнения, 6-е изд.: Пер. с англ – СПб.: ООО «Диалектика», 2020 – 1248с.

2 Черпаков И.В. Основы программирования. – М.: Юрайт, 2017.

3 Семакин И.Г. Основы алгоритмизации и программирования. – М.: Академия, 2016.