**ПРАКТИЧЕСКАЯ РАБОТА**

**СОЗДАНИЕ ПРИЛОЖЕНИЯ С ОСНОВНЫМИ ЭЛЕМЕНТАМИ ВВОДА/ВЫВОДА**

***Цель: получение навыков работы по созданию приложений с основными элементами ввода/вывода.***

Создадим приложение (программы, предназначенные для решения прикладных задач, принято называть приложениями), позволяющее посчитать доход по вкладу (рисунок 1).

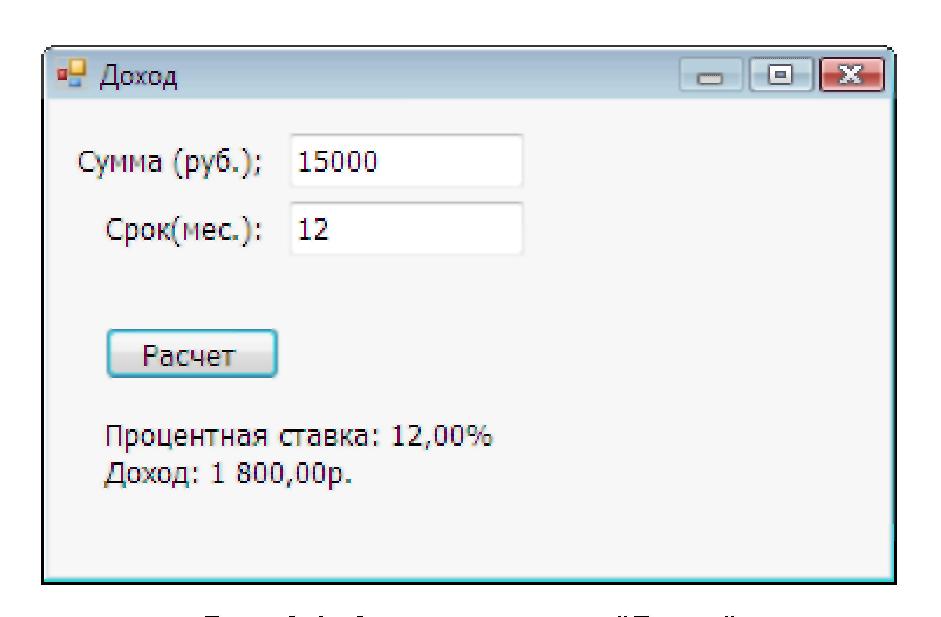


Рисунок 1 - Окно программы "Доход"

Начало работы над проектом

Чтобы начать работу над новым проектом, надо:

В меню File выбрать команду New Project.

В открывшемся окне New Project выбрать тип приложения — Windows Forms Application - Visual C#.

В поле Name ввести имя проекта — profit и нажать кнопку ОК (рисунок 2).

В результате описанных действий в папке временных проектов (по умолчанию это [C:\Users\User\AppData\Local\Temporary](file://C:/Users/Z7ser/AppData/Local/Temporary) Projects) будет создана папка profit, а в ней — проект profit.

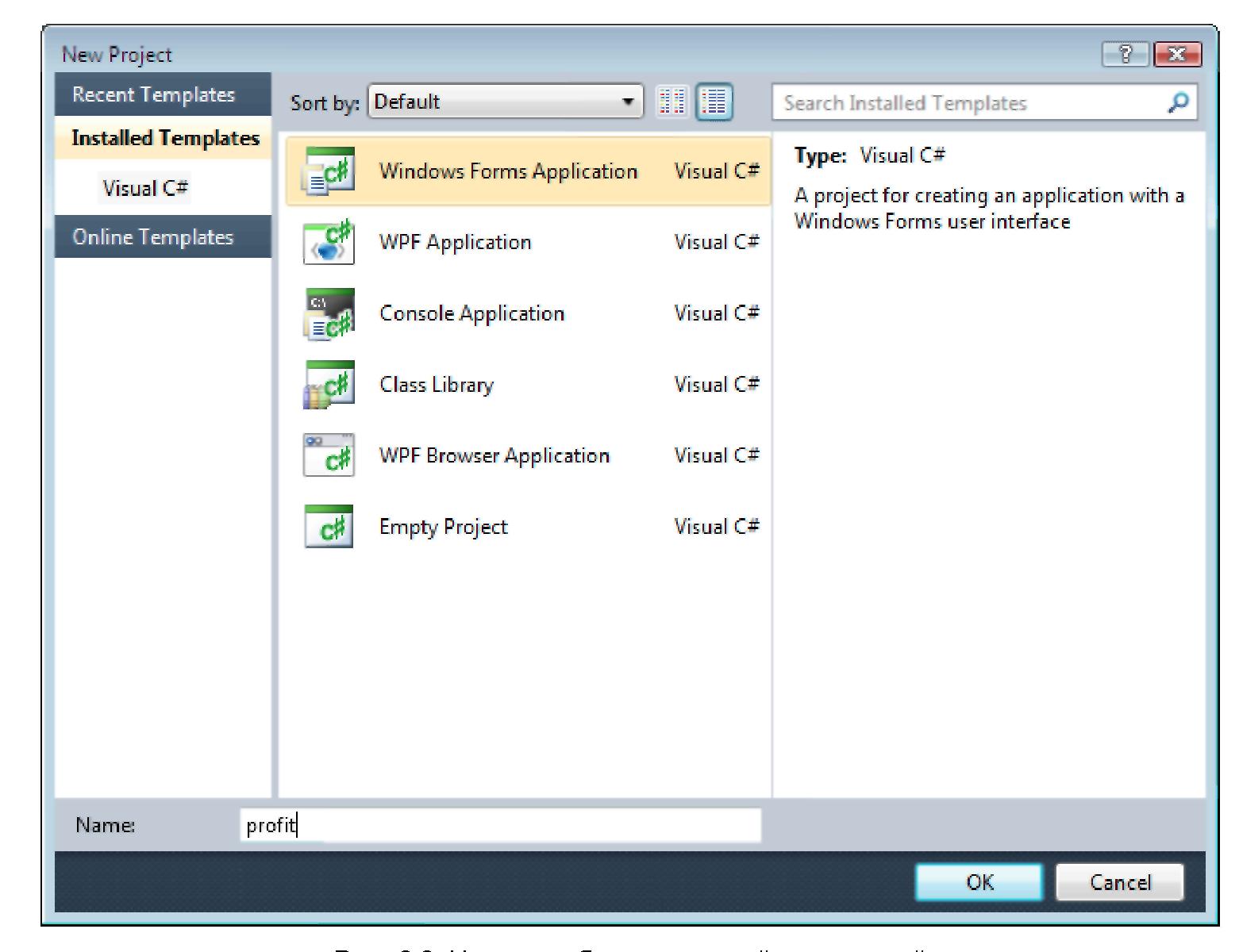


Рисунок 2. Начало работы над новой программой

Форма

Работа над приложением начинается с создания стартовой формы — главного окна программы. Форма создается путем добавления на заготовку формы необходимых компонентов и изменения значений свойств самой формы.

Сначала нужно установить требуемые значения свойств формы, затем — поместить на форму необходимые компоненты (поля ввода информации, командные кнопки, поля отображения текста и др.) и выполнить.

Настройка формы (а также компонентов) осуществляется путем изменения значений свойств. Свойства объекта (формы, компонента) определяют его вид и поведение. Например, свойство Text определяет текст заголовка окна, а свойство startPosition — положение окна в момент появления его на экране.

Основные свойства формы (объекта Form) приведены в таблице 1.

Таблица 1. Свойства формы (объекта Form)

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| Name | Имя формы |
| Text | Текст в заголовке |
| Size | Размер формы. Уточняющее свойство width определяет ширину, свойство Heigh — ВЫСОТУ |
| StartPosition | Положение формы в момент первого появления на экране. Форма может находиться в центре экрана (CenterScreen), в центре родительского окна (CenterParent). Если значение свойства равно Manual, то положение  формы Определяется Значением СВОЙСТВа Location |
| Location | Положение формы на экране. Расстояние от верхней границы формы до верхней границы экрана задает уточняющее свойство Y, расстояние от левой границы формы до левой границы экрана — уточняющее свойство х |
| ForrrBorderStyle | Тип формы (границы). Форма может представлять собой обычное окно  (Sizable), ОКНО фиксированного размера (FixedSingle, Fixed3D), ДИЭЛОГ  (FixedDialog) или окно без кнопок Свернуть и Развернуть  (SizeableToolWindow, FixedToolWindow). ЕСЛИ СВОЙСТВУ ПРИСВОИТЬ Значение None, у окна не будет заголовка и границы |
| ControlBox | Управляет отображением системного меню и кнопок управления окном. Если значение свойства равно False, то в заголовке окна кнопка системного меню, а также кнопки Свернуть, Развернуть, Закрыть не отображаются |
| MaximazeBox | Кнопка Развернуть. Если значение свойства равно False, то находящаяся в заголовке окна кнопка Развернуть недоступна |
| MinimazeBox | Кнопка Свернуть. Если значение свойства равно False, то находящаяся в заголовке окна кнопка Свернуть недоступна |
| Icon | Значок в заголовке окна |
| Font | Шрифт, используемый по умолчанию компонентами, находящимися на поверхности формы. Изменение значения свойства приводит к автоматическому изменению значения свойства Font всех компонентов формы (при условии, что значение свойства компонента не было задано явно) |
| ForeColor | Цвет, наследуемый компонентами формы и используемый ими для отображения текста. Изменение значения свойства приводит к автоматическому изменению соответствующего свойства всех компонентов формы (при условии, что значение свойства Font компонента не было задано явно) |
| BackColor | Цвет фона. Можно задать явно (выбрать на вкладке Custom или Web) или указать элемент цветовой схемы (выбрать на вкладке System) |
| Opacity | Степень прозрачности формы. Форма может быть непрозрачной (100%) или прозрачной. Если значение свойства меньше 100%, то сквозь форму видна поверхность, на которой она отображается |

Для изменения значений свойств объектов используется окно Properties. В левой колонке окна перечислены свойства объекта, выбранного в данный момент, в правой — указаны значения свойств. Имя выбранного объекта отображается в верхней части окна Properties.

Чтобы в заголовке окна отображалось название программы, надо изменить значение свойства Text. Для этого следует щелкнуть левой кнопкой мыши в поле значение свойства Text (в поле появится курсор), ввести в поле редактирования текст ***Доход*** и нажать клавишу <Enter> (рис. 2.3).

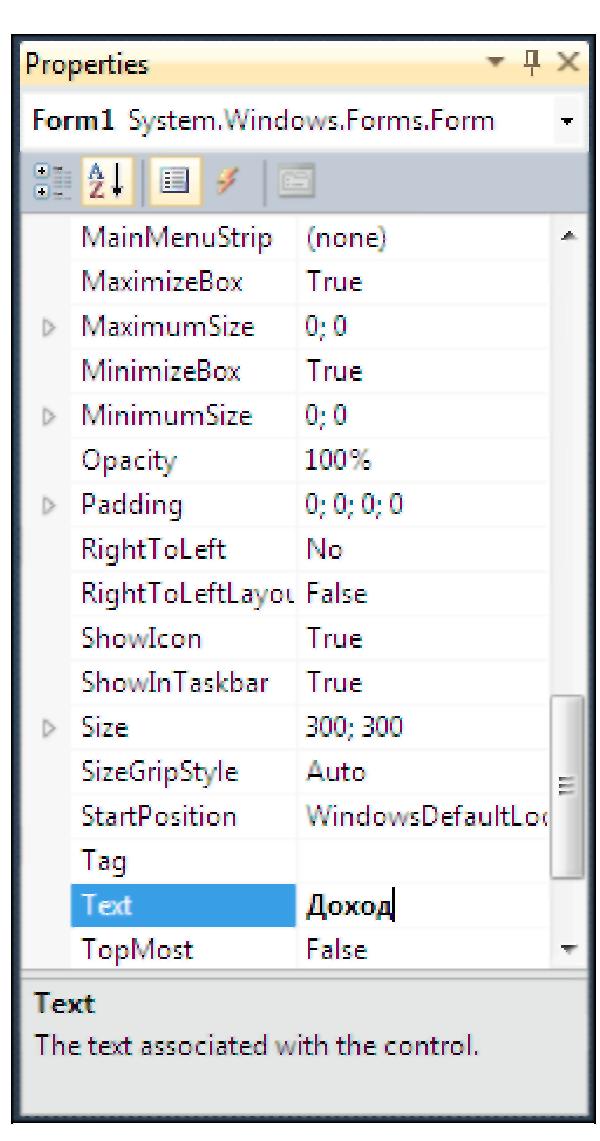


Рисунок 3. Изменение значения свойства Text путем ввода нового значения

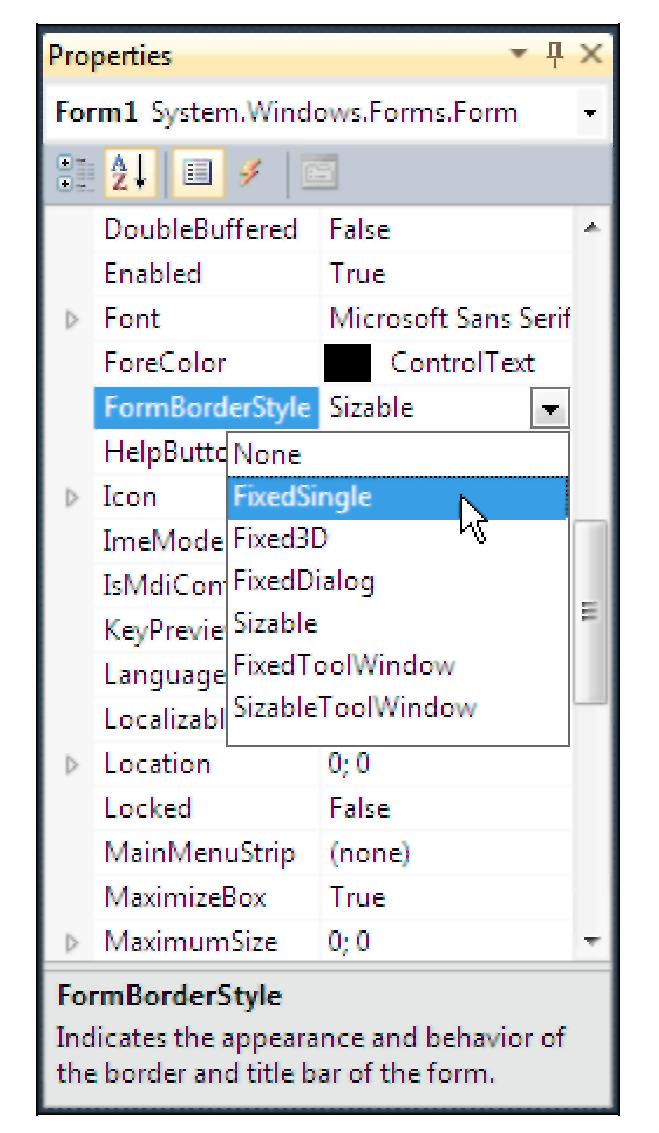


Рисунок 4. Установка значения свойства путем выбора из списка

При выборе некоторых свойств, например FormBorderstyle, справа от текущего значения свойства появляется значок раскрывающегося списка. Очевидно, что значение таких свойств можно задать путем выбора из списка (рисунок.4).

Некоторые свойства являются сложными. Они представляют собой совокупность других (уточняющих) свойств. Например, свойство size, определяющее размер формы, представляет собой совокупность свойств width и Height. Перед именами сложных свойств стоит значок ► , в результате щелчка на котором раскрывается список уточняющих свойств (рисунок 5). Значение уточняющего свойства можно задать (изменить) обычным образом — ввести нужное значение в поле редактирования.

Размер формы можно изменить и с помощью мыши, точно так же, как и любого окна, т. е. путем перемещения границы. По окончании перемещения границы значения свойств width и Height будут соответствовать установленному размеру формы.

В результате выбора некоторых свойств, например Font, в поле значения свойства отображается кнопка, на которой изображены три точки. Это значит, что задать значение свойства можно в дополнительном диалоговом окне, которое появится в результате щелчка на этой кнопке. Например, значение свойства Font можно задать путем ввода значений уточняющих свойств (Name, size, style и др.), а можно воспользоваться стандартным диалоговым окном Шрифт, которое появится в результате щелчка на кнопке с тремя точками (рисунок 6).

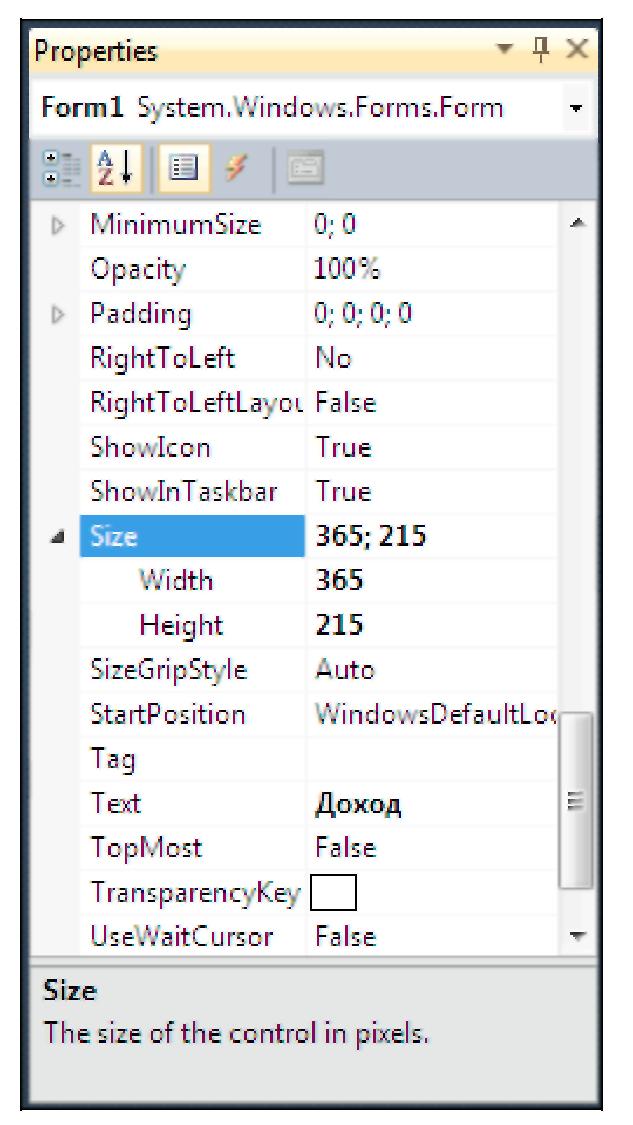


Рисунок 5. Изменение значения уточняющего свойства

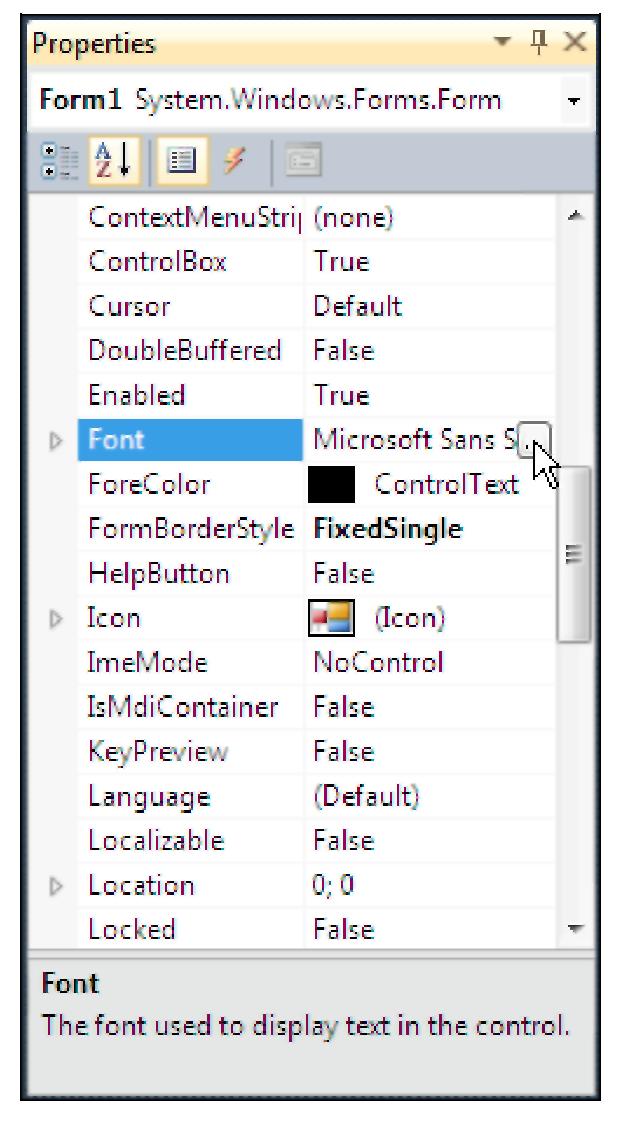


Рисунок 6. Чтобы задать свойства шрифта, щелкните на кнопке с тремя точками

В таблице 2 приведены значения свойств формы программы "Доход". Значения остальных свойств формы оставлены без изменения и поэтому в таблице не представлены. Обратите внимание, в именах некоторых свойств есть точка. Это значит, что это значение уточняющего свойства.

Таблица 2. Значения свойств стартовой формы

|  |  |  |
| --- | --- | --- |
| ***Свойство*** | ***Значение*** | ***Комментарий*** |
| Text | Доход |  |
| Size.Width | 365 |  |
| Size.Height | 215 |  |
| FonrBorderStyle | FixedSingle | Тонкая граница формы. Во время работы программы пользователь не сможет изменить размер окна путем захвата и перемещения его границы |
| Свойство | Значение | Комментарий |
| StartPosition | CenterScreen | Окно программы появится в центре экрана |
| MaximizeBox | False | В заголовке окна не отображать кнопку Развернуть |
| Font. Name | Tahoma |  |
| Font.Size | 10 |  |

После того как будут установлены значения свойств формы, она должна выглядеть так, как показано на рисунке 7. Теперь на форму надо добавить компоненты.

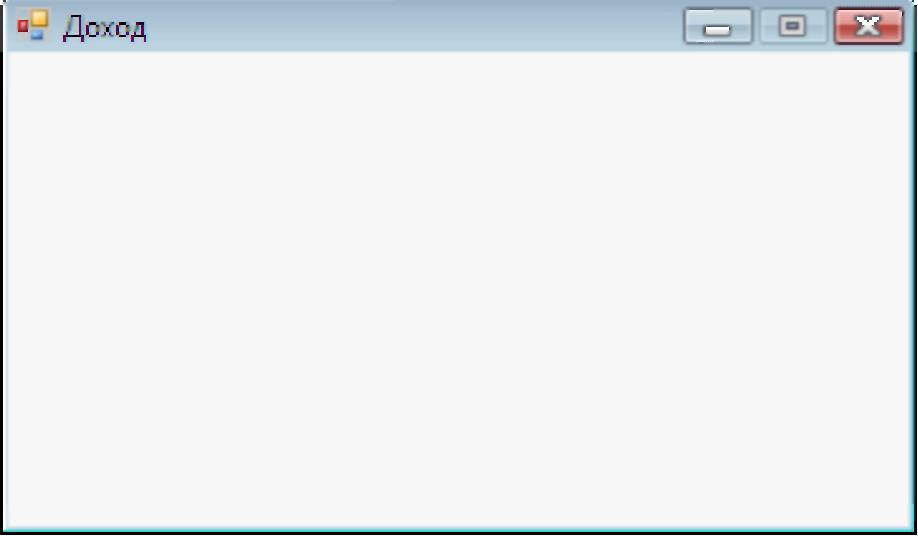


Рисунок 7. Форма после изменения значений ее свойств

**Компоненты**

Поля ввода/редактирования, поля отображения текста, командные кнопки, списки, переключатели и другие элементы, обеспечивающие взаимодействие пользователя с программой, называют компонентами пользовательского интерфейса. Они находятся в окне Tollbox на вкладке Common Controls.

Программа "Доход" должна получить от пользователя исходные данные — сумму и срок вклада. Ввод данных с клавиатуры обеспечивает компонент TextBox. Таким образом, на форму разрабатываемого приложения нужно поместить два компонента TextBox.

Чтобы на форму добавить компонент TextBox, надо:

В палитре компонентов (окно Toolbox) раскрыть вкладку Common Controls.

Сделать щелчок на значке компонента TextBox (рисунок 8).

Установить указатель мыши в ту точку формы, в которой должен быть левый верхний угол компонента, и сделать щелчок левой кнопкой мыши.

В результате на форме появляется поле ввода/редактирования — компонент

TextBox (рисунок 9).

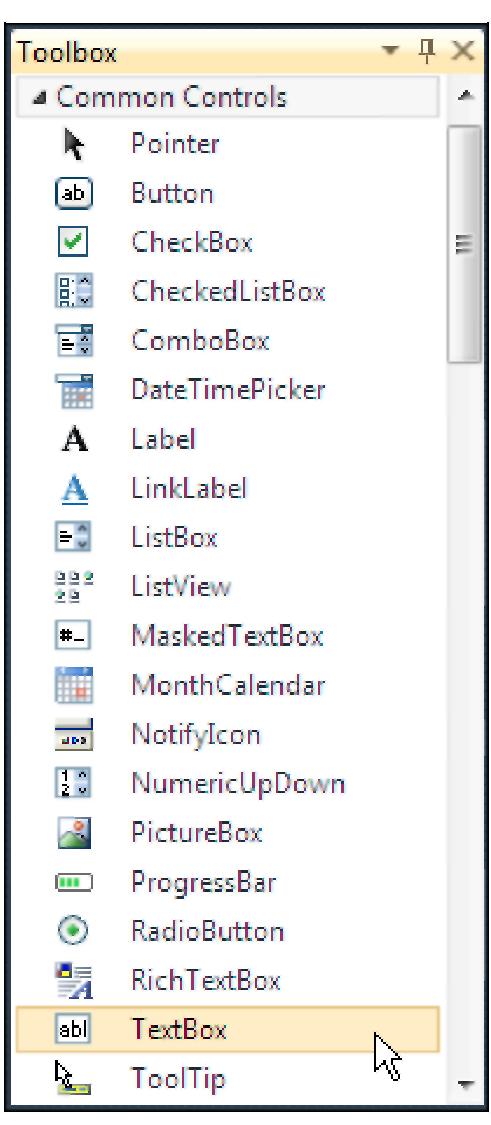


Рисунок 8. Выбор компонента в палитре (компонент TextBox — поле редактирования)

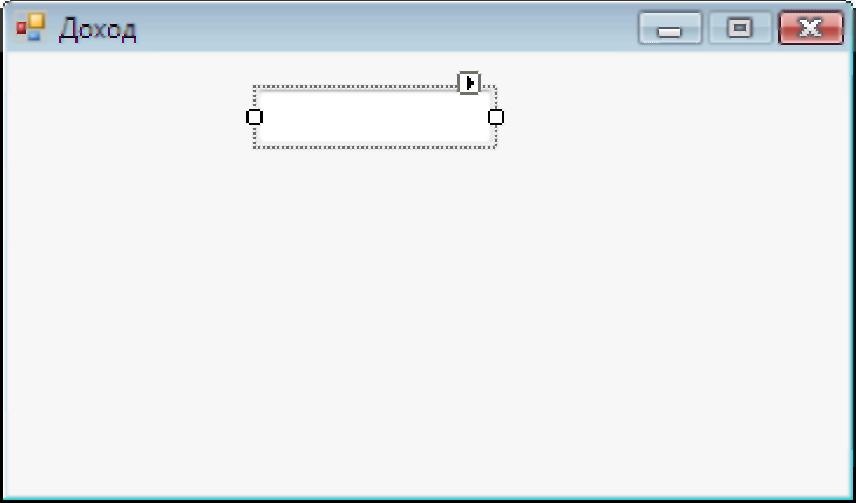


Рисунок 9. Результат добавления на форму компонента TextBox

Каждому добавленному компоненту среда разработки присваивает имя, которое состоит из названия компонента и его порядкового номера. Например, первый добавленный на форму компонент TextBox получает ИМЯ textBox1, второй — textBox2. Программист путем изменения значения свойства Name может поменять имя компонента. Однако в простых программах имена компонентов, как правило, не меняют.

Основные свойства компонента TextBox приведены в таблице 3.

Таблица 3. Свойства компонента TextBox

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| Name | Имя компонента. Используется для доступа к компоненту и его свойствам |
| Text | Текст, который находится в поле редактирования |
| Location | Положение компонента на поверхности формы |
| Size | Размер компонента |
| Font | Шрифт, используемый для отображения текста в поле компонента |
| ForeColor | Цвет текста, находящегося в поле компонента |
| BackColor | Цвет фона поля компонента |
| Borderstyle | Вид рамки (границы) компонента. Граница компонента может быть обычной  (Fixed3D), ТОНКОЙ (FixedSingle) ИЛИ отсутствовать (None) |
| TextAlign | Способ выравнивания текста в поле компонента. Текст в поле компонента может быть прижат клевой границе компонента (Left), правой (Right) или  НаХОДИТЬСЯ ПО Центру (Center) |
| MaxLength | Максимальное количество символов, которое можно ввести в поле компонента |
| Multiline | Разрешает (True) или запрещает (False) ввод нескольких строк текста |
| Readonly | Разрешает (True) или запрещает (False) редактирование отображаемого текста |
| Lines | Массив строк, элементы которого содержат текст, находящийся в поле редактирования, если компонент находится в режиме MultiLine. Доступ к строке осуществляется по номеру. Строки нумеруются с нуля |
| ScrollBars | Задает отображаемые полосы прокрутки: Horizontal — горизонтальная; Vertical — вертикальная; Both — горизонтальная и вертикальная; None — не отображать |

На рисунок 10 приведен вид формы программы "Доход" после добавления двух полей ввода/редактирования. Один из компонентов выбран — выделен рамкой. Свойства именно этого (выбранного) компонента отображаются в окне Properties. Чтобы увидеть и, если надо, изменить свойства другого компонента, нужно этот компонент выбрать — щелкнуть левой кнопкой мыши на изображении компонента в форме или выбрать его имя в раскрывающемся списке, который находится в верхней части окна Properties (рисунок 11).

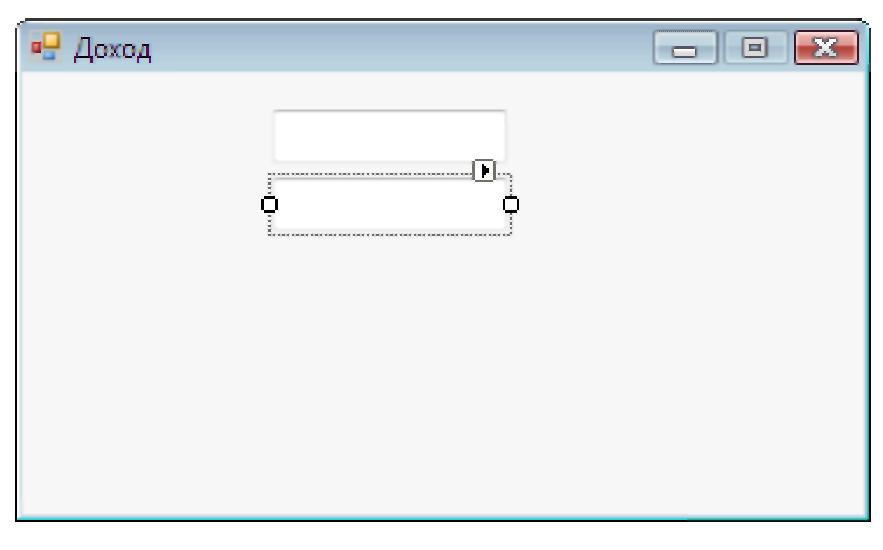


Рисунок 10. Форма с двумя компонентами TextBox

Значения свойств компонента, определяющих размер и положение компонента на поверхности формы, можно изменить с помощью мыши.

Чтобы изменить положение компонента, необходимо установить курсор мыши на его изображение, нажать левую кнопку мыши и, удерживая ее нажатой, переместить компонент в нужную точку формы (рисунок 12).

Для того чтобы изменить размер компонента, необходимо сделать щелчок на его изображении (в результате чего компонент будет выделен), установить указатель мыши на один из маркеров, помечающих границу компонента, нажать левую кнопку мыши и, удерживая ее нажатой, изменить положение границы компонента (рисунок 13).

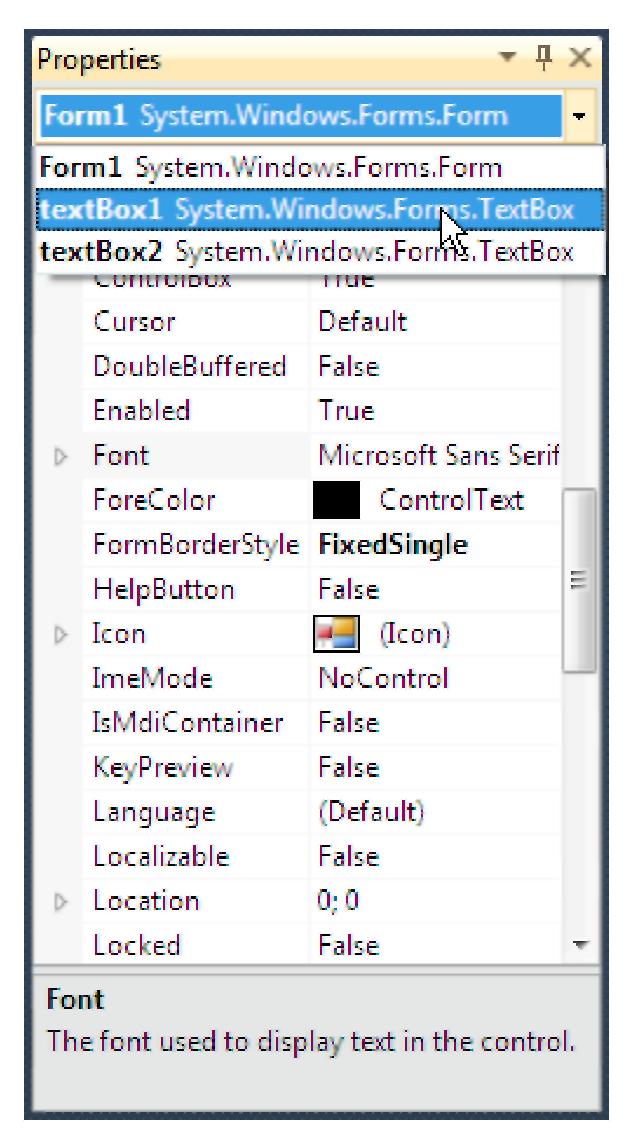


Рисунок 11. Выбор компонента в окне Properties

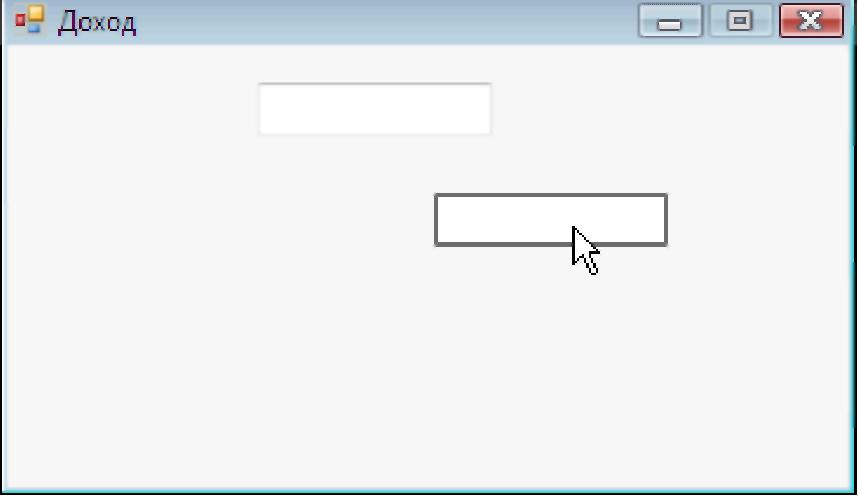


Рисунок 12. Изменение положения компонента

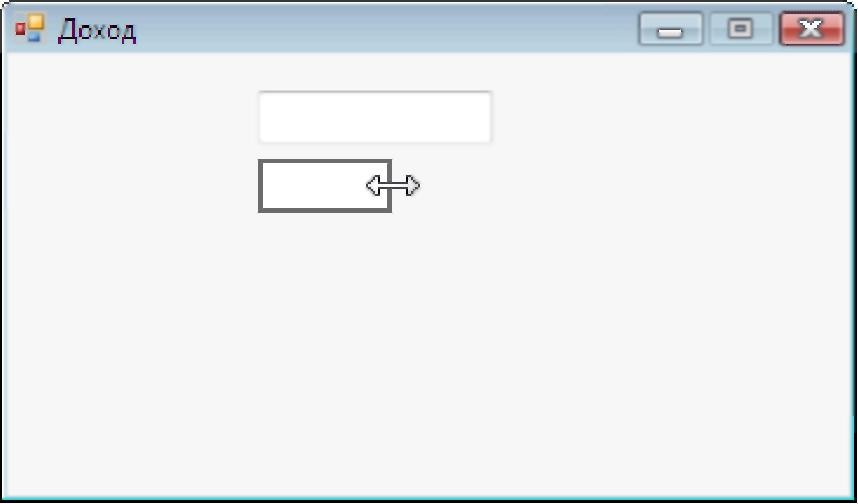


Рисунок 13. Изменение размера компонента

В таблице 4 приведены значения свойств компонентов textBoxl и textBox2 (прочерк показывает, что значением свойства Text является пустая строка). Значения остальных свойств компонентов оставлены без изменения и поэтому в таблице не показаны. Компонент textBoxl предназначен для ввода суммы вклада, textBox2 — срока. Так как значения свойства Font компонентов TextBox не были изменены, то во время работы программы текст в полях редактирования будет отображаться шрифтом, заданным для формы. Компоненты TextBox, как и другие компоненты, находящиеся на форме, наследуют значение свойства Font формы (если значение свойства Font компонента не было задано явно). Поэтому если изменить значение свойства Font формы, автоматически изменятся значения свойств Font компонентов, находящихся на форме. Если требуется, чтобы текст в поле компонента отображался другим шрифтом, нужно явно задать значение свойства Font этого компонента.

Форма программы "Доход" после настройки компонентов TextBox приведена на рисунке 14.

Таблица 4. Значения свойств компонентов TextBox

|  |  |  |
| --- | --- | --- |
| **Компонент** | **Свойство** | **Значение** |
| textBoxl | Location.X | 107 |
| Location.Y | 16 |
| Size.Width | 100 |
| Size.Height | 23 |
| Text | - |
| Tablndex | 0 |
| textBox2 | Location.X | 107 |
| Location.Y | 45 |
| Size.Width | 57 |
| Size.Height | 23 |
| Text | - |
| Tablndex | 1 |

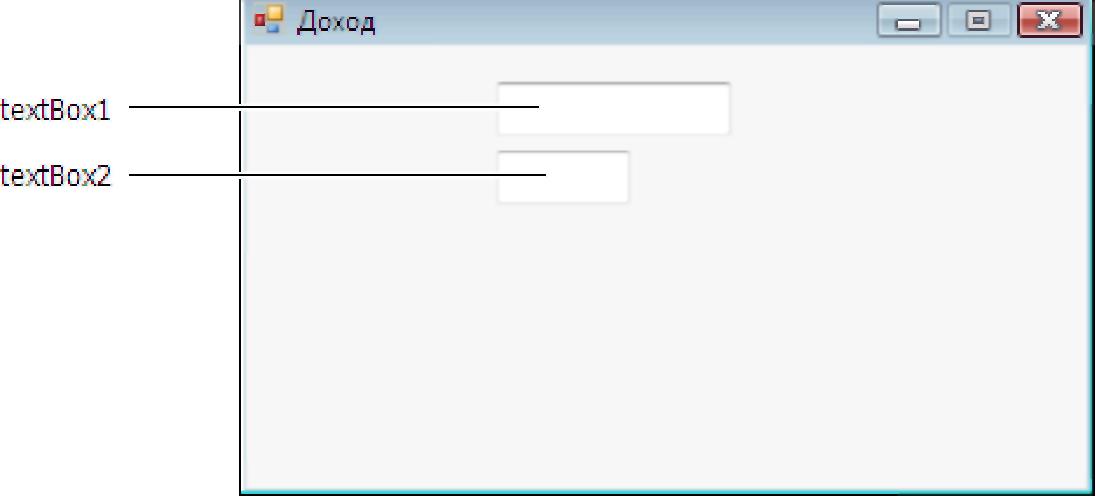


Рисунок 14. Форма после настройки компонентов TextBox

Отображение текста на поверхности формы (подсказок, результата расчета) обеспечивает компонент Label. В окне программы "Доход" текст отображается слева от полей ввода/редактирования (информация о назначении полей). Результат расчета также отображается в окне программы. Поэтому в форму надо добавить три компонента Label (рисунок 15).

Добавляется компонент Label на форму точно так же, как и поле редактирования (компонент TextBox).

Основные свойства компонента Label приведены в таблице 5.

На форму разрабатываемого приложения надо добавить три компонента Label. В полях labell и label2 отображается информация о назначении полей ввода, поле label3 используется для вывода результата расчета.

Значения свойств компонентов Label приведены в таблице 6.

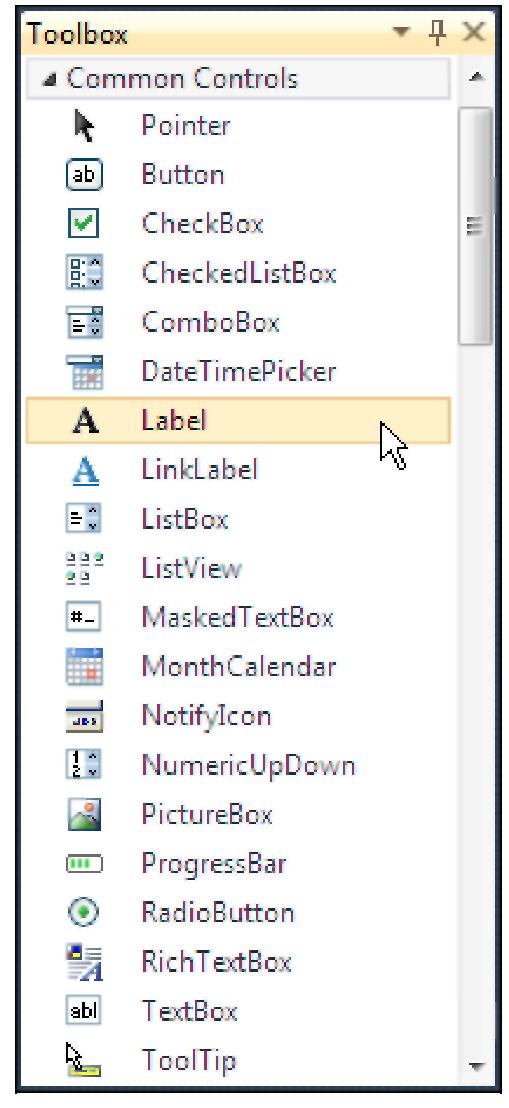


Рисунок 15. Компонент Label — поле отображения текста

Таблица 5. Свойства компонента Label

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| Name | Имя компонента. Используется в программе для доступа к свойствам компонента |
| Text | Отображаемый текст |
| Location | Положение компонента на поверхности формы |
| AutoSize | Признак автоматического изменения размера компонента. Если значение свойства равно True, то при изменении значения свойства Text (или Font) автоматически изменяется размер компонента |
| Size | Размер компонента (области отображения текста). Определяет (если значение свойства AutoSize равно False) размер компонента (области отображения текста) |
| Font | Шрифт, используемый для отображения текста |
| ForeColor | Цвет текста, отображаемого в поле компонента |
| BackColor | Цвет закраски области вывода текста |
| TextAlign | Способ выравнивания (расположения) текста в поле компонента. Всего существует девять способов расположения текста. На практике наиболее часто используют выравнивание по левой верхней границе (TopLeft), посередине (TopCenter) И ПО центру (MiddleCenter) |

Таблица 6. Значения свойств компонентов Label

|  |  |  |
| --- | --- | --- |
| Компонент | Свойство | Значение |
| label1 | Location.X | 13 |
|  | Location.Y | 19 |
|  | AutoSize | False |
|  | Sise.Width | 88 |
|  | Size.Height | 20 |
|  | Text | Сумма (руб.): |
|  | TextAlign | MiddleRight |
| label2 | Location.X | 13 |
|  | Location.Y | 45 |
|  | AutoSize | False |
|  | Sise.Width | 88 |
|  | Size.Height | 20 |
|  | Text | Срок (мес.): |
|  | TextAlign | MiddleRight |
| label3 | Location.X | 23 |
|  | Location.Y | 122 |
|  | AutoSize | False |
|  | Sise.Width | 299 |
|  | Size.Height | 50 |
|  | Text | " |

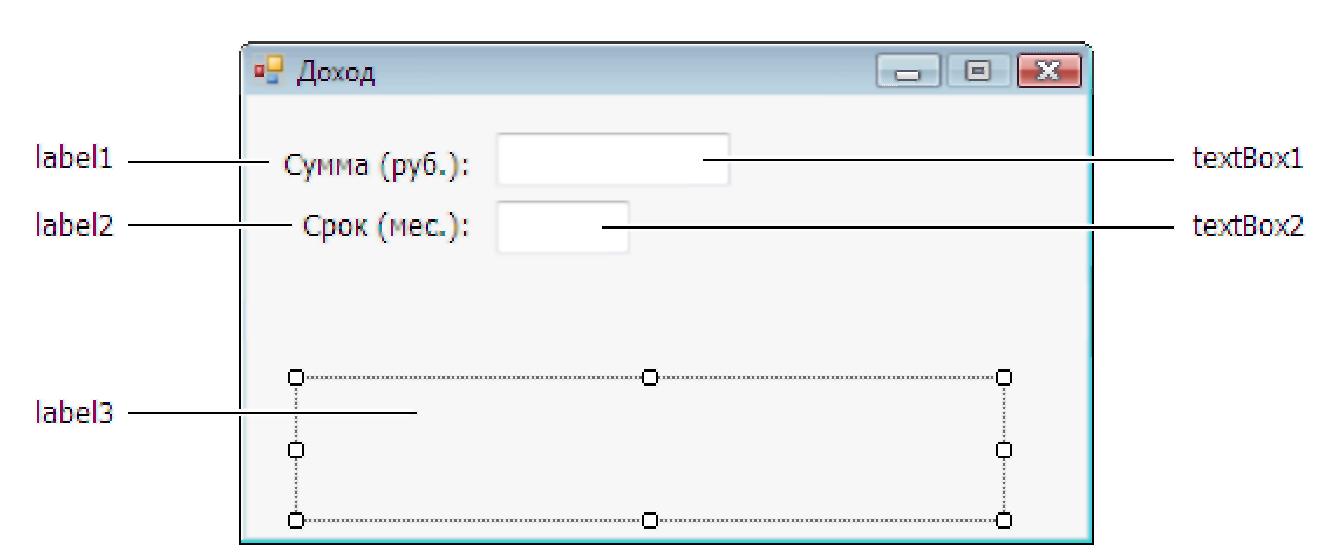
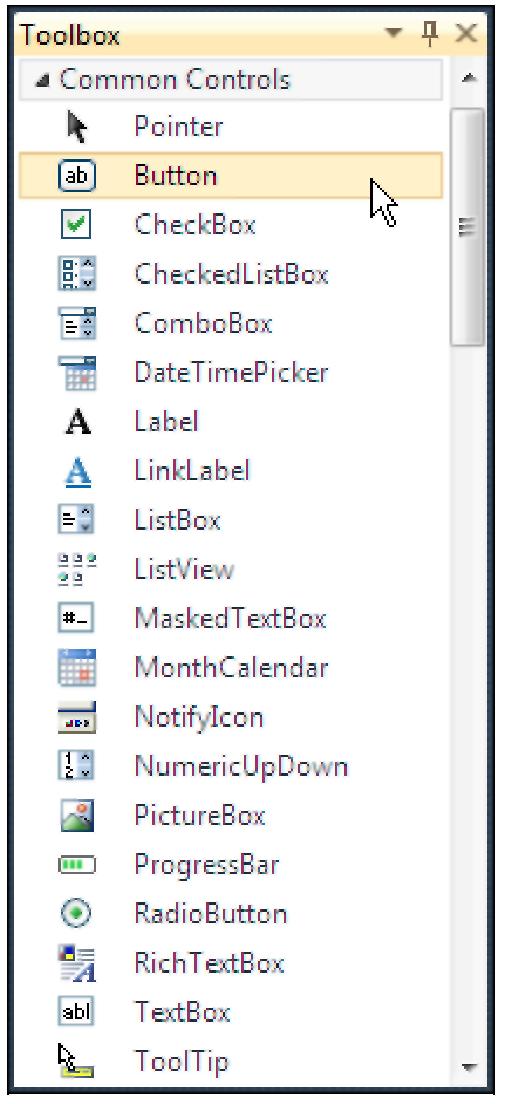


Рисунок 16. Вид формы после настройки полей отображения текста

После настройки компонентов Label форма разрабатываемого приложения должна выглядеть так, как показано на рисунке 16.



Рисунке 17. Командная кнопка — компонент Button

Последнее, что надо сделать на этапе создания формы, — добавить на форму командную кнопку Расчет. Назначение этой кнопки очевидно.

Командная кнопка, компонент Button (рисунок 17), добавляется на форму точно так же, как и другие компоненты. Значок компонента Button находится на вкладке Common Controls. Основные свойства компонента Button приведены в таблице 7.

Таблица 7. Свойства компонента Button

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| Name | Имя компонента. Используется для доступа к компоненту и его свойствам |
| Text | Текст на кнопке |
| TextAlign | Положение текста на кнопке. Текст может располагаться в центре кнопки  (MiddleCenter), быть прижат к левой (middleleft) или Правой  (MiddleRight) границе. Можно задать и другие способы размещения  надписи (TopLeft, TopCenter, TopRight, BottomLeft, BottomCenter, BottomRight) |
| FlatStyle | СТИЛЬ. Кнопка МОЖеТ быть Стандартной (Standard), ПЛОСКОЙ (Flat) ИЛИ "ВСПЛЫВаЮЩеЙ" (Popup) |
| Location | Положение кнопки на поверхности формы. Уточняющее свойство х определяет расстояние от левой границы кнопки до левой границы формы, уточняющее свойство Y — от верхней границы кнопки до верхней границы клиентской области формы (нижней границы заголовка) |
| Size | Размер кнопки |
| Enabled | Признак доступности кнопки. Кнопка доступна, если значение свойства равно True, и недоступна, если значение свойства равно False (в этом случае нажать кнопку нельзя, событие click в результате щелчка на ней не возникает) |
| Visible | Позволяет скрыть кнопку (False) или сделать ее видимой (True) |
| Cursor | Вид указателя мыши при позиционировании указателя на кнопке |
| Image | Картинка на поверхности кнопки. Рекомендуется использовать gif-файл, в котором определен прозрачный цвет |
| ImageAlign | Положение картинки на кнопке. Картинка может располагаться в центре  (middlecenter), быть прижат к левой (middleleft) или правой  (middleright) границе. можно задать и другие способы размещения  картинки на кнопке (topleft, topcenter, topright, bottomleft, bottomcenter, bottomright) |
| ImageList | Набор изображений, из которых может быть выбрано то, которое будет отображаться на поверхности кнопки. представляет собой объект типа imagelist. чтобы задать значение свойства, в форму приложения нужно  добавить компонент imagelist |
| ImageIndex | Номер (индекс) изображения из набора ImageList, которое отображается на кнопке |
| ToolTip | Подсказка, появляющаяся рядом с указателем мыши при его позиционировании на кнопке. Чтобы свойство стало доступно, в форму приложения  НУЖНО Добавить компонент tooltip |

После того как на форму будут добавлены кнопки, нужно выполнить их настройку. Значения свойств компонентов Button приведены в табл. 2.8, окончательный вид формы показан на рисунке 18.

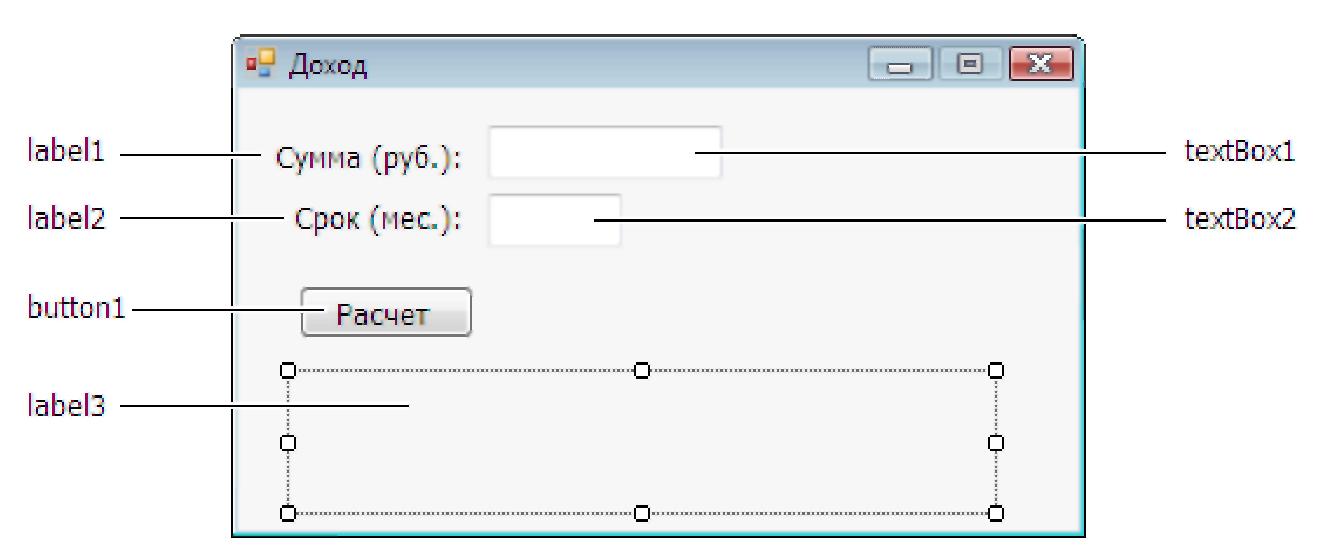


Рисунок18. Окончательный вид формы программы "Доход"

Таблица 8. Значения свойств компонента buttonl

|  |  |
| --- | --- |
| **Свойство** | **Значение** |
| Location.X | 26 |
| Location.Y | 84 |
| Size.Width | 75 |
| Size.Height | 23 |
| Text | Расчет |

Завершив работу по созданию формы, можно приступить к программированию — созданию процедур обработки событий.

**Событие**

Вид формы программы "Доход" подсказывает, как работает программа. Очевидно, что пользователь должен ввести в поля редактирования исходные данные и сделать щелчок на кнопке Расчет. Щелчок на изображении командной кнопки — это пример того, что называется событием.

Событие (event) — это то, что происходит во время работы программы. Например, щелчок кнопкой мыши — это событие click, двойной щелчок мышью — событие DblClick.

В таблице 9 приведены некоторые события, возникающие в результате действий пользователя.

Таблица 9. События

|  |  |
| --- | --- |
| Событие | Описание |
| Click | Щелчок кнопкой мыши |
| Doubleclick | Двойной щелчок кнопкой мыши |
| MouseDown | Нажатие кнопки мыши |
| MouseUp | Отпускание нажатой кнопки мыши |
| MouseMove | Перемещение указателя мыши |
| KeyPress | Нажатие клавиши |
| KeyDowi | Нажатие клавиши. События KeyDown и KeyPress — это чередующиеся, повторяющиеся события, которые происходят до тех пор, пока не будет отпущена удерживаемая клавиша (в этот момент происходит событие KeyUp) |
| KeyUp | Отпускание нажатой клавиши |
| TextChanged | Признак, указывающий, изменился ли текст, находящийся в поле редактирования (изменилось значение свойства Text) |
| Load | Загрузка формы. Функция обработки этого события обычно используется для инициализации переменных, выполнения подготовительных действий |
| Paint | Событие происходит при появлении окна на экране в начале работы программы, после появления части окна, которая, например, была закрыта другим окном |
| Enter | Получение фокуса элементом управления |
| Leave | Потеря фокуса элементом управления |

Следует понимать, что одни и те же действия, но выполненные над разными объектами, вызывают разные события. Например, нажатие клавиши (событие KeyPress) в поле ввода/редактирования Сумма и нажатие клавиши (также событие KeyPress) в поле Срок — это два разных события.

**Функция обработки события**

Реакцией на событие должно быть какое-либо действие. В Visual C# реакция на событие реализуется как функция обработки события. Таким образом, для того чтобы программа в ответ на действия пользователя выполняла некоторую работу, программист должен написать функцию (метод) обработки соответствующего события.

Процесс создания функции обработки события рассмотрим на примере обработки события click для кнопки Расчет.

Чтобы создать функцию обработки события, сначала надо выбрать компонент, для которого создается функция обработки события. Для этого в окне конструктора формы надо сделать щелчок левой кнопкой мыши на нужном компоненте. Затем в окне Properties щелчком на кнопке Events (рисунок 19) нужно открыть вкладку Events.

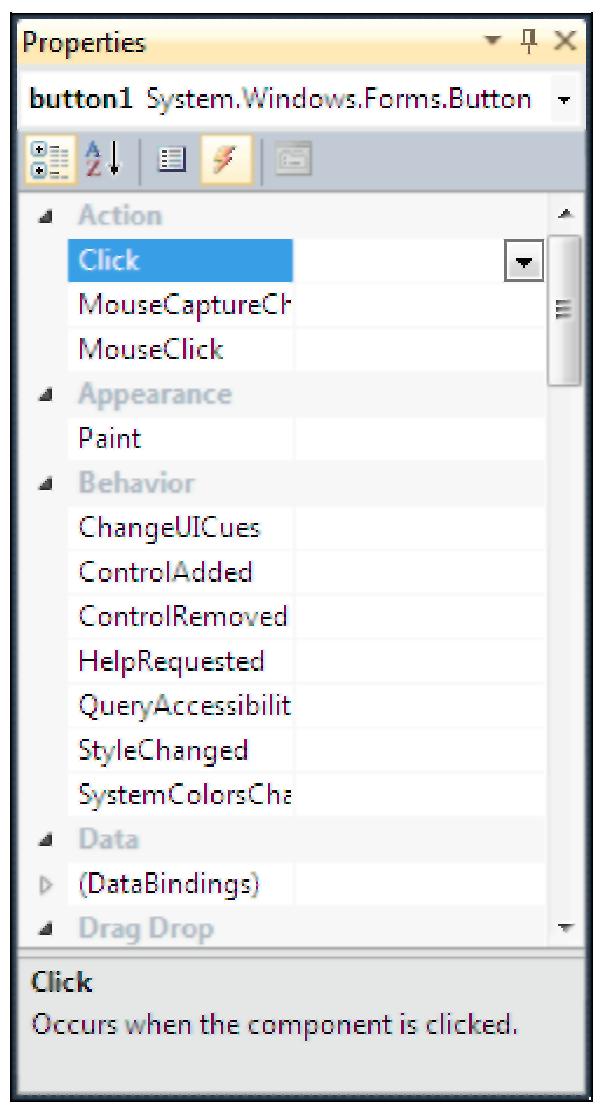


Рисунок 19. Кнопка Events



Рисунок 20. На вкладке Events перечислены события, которые может воспринимать компонент

В левой колонке вкладки Events (рисунок 20) перечислены события, которые может воспринимать выбранный компонент. Строго говоря, на вкладке Events указаны не события, а свойства, значением которых являются имена функций обработки соответствующих событий.

Для того чтобы создать функцию обработки события, нужно на вкладке Events выбрать событие (сделать щелчок мышью на имени события), в поле значения

свойства ввести имя функции обработки события (рисунок 21) и нажать клавишу <Enter>.

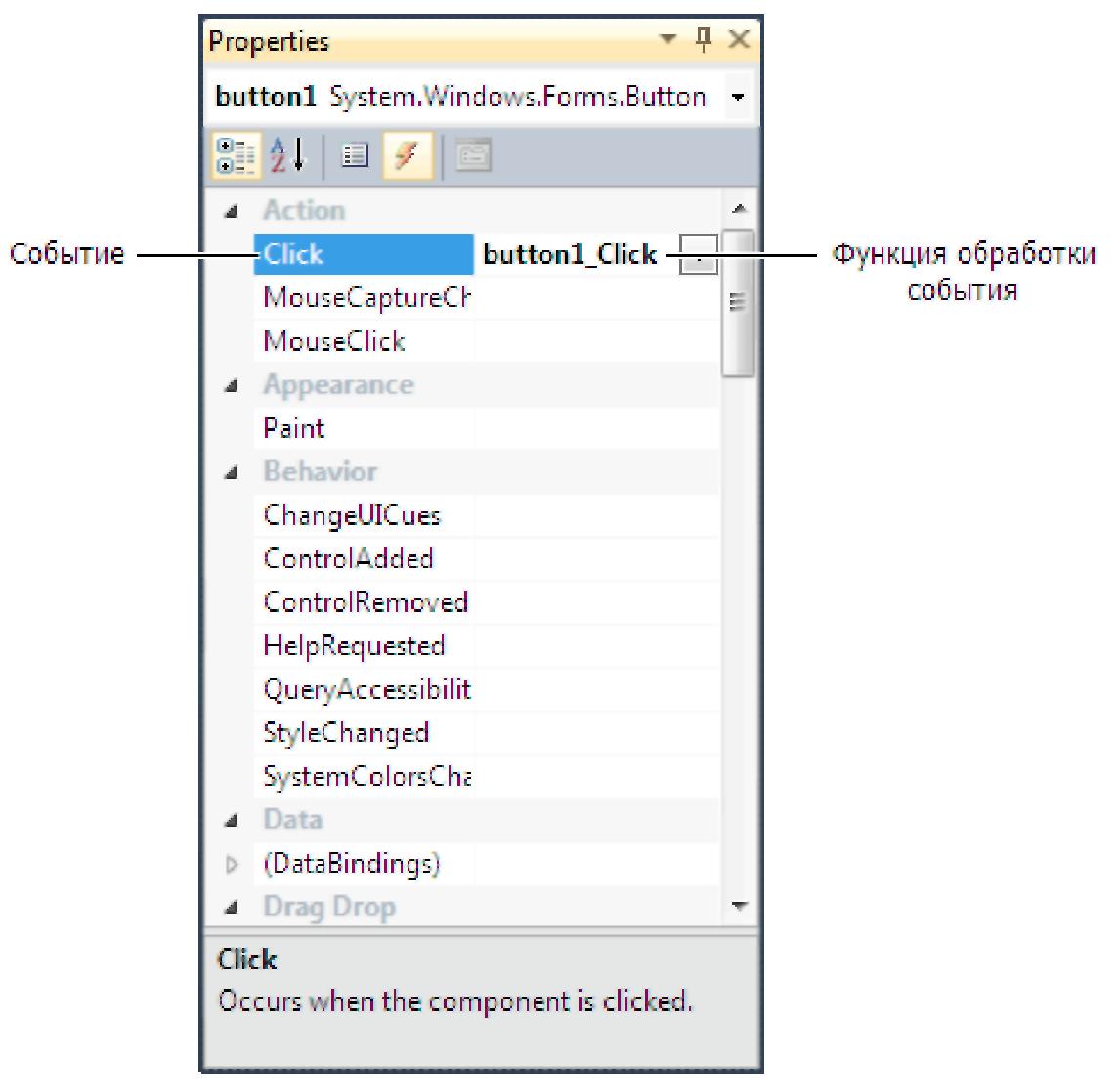
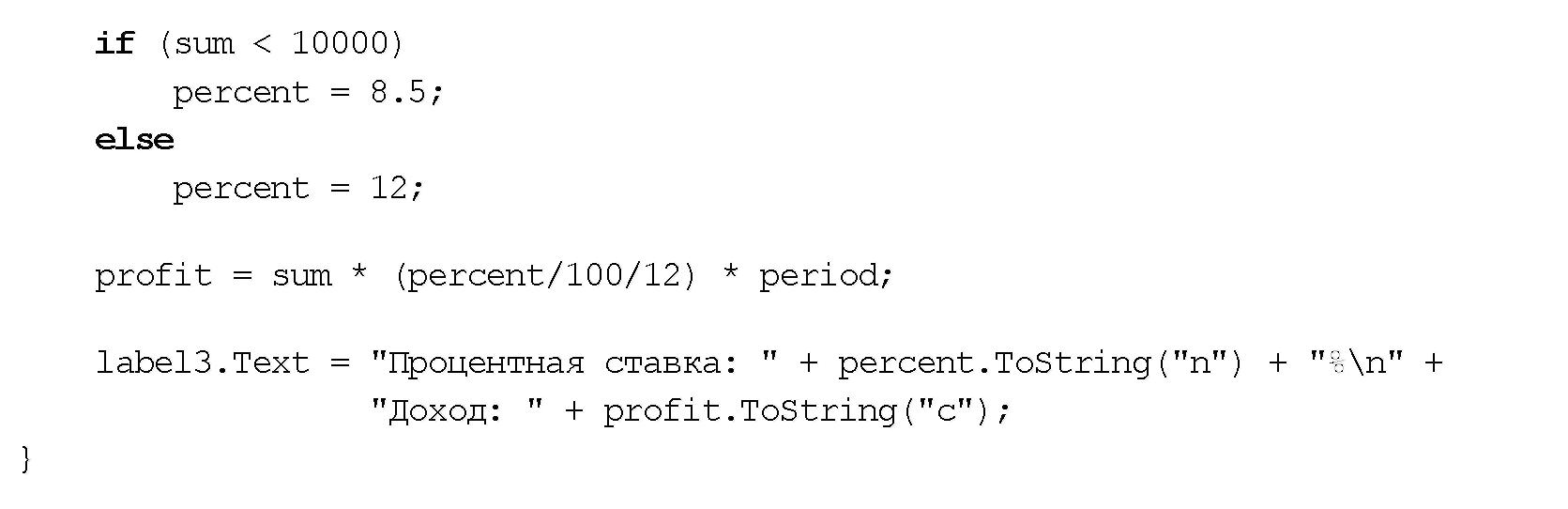


Рисунок 21. Рядом с именем события надо ввести имя функции обработки события

В результате этих действий в модуль формы (cs-файл) будет добавлена функция (метод класса формы) обработки события и станет доступным окно редактора кода (рисунок 22), в котором можно набирать инструкции, реализующие функцию обработки события.

Функция обработки события click для кнопки Расчет (buttonl) приведена в листинге 2.1.





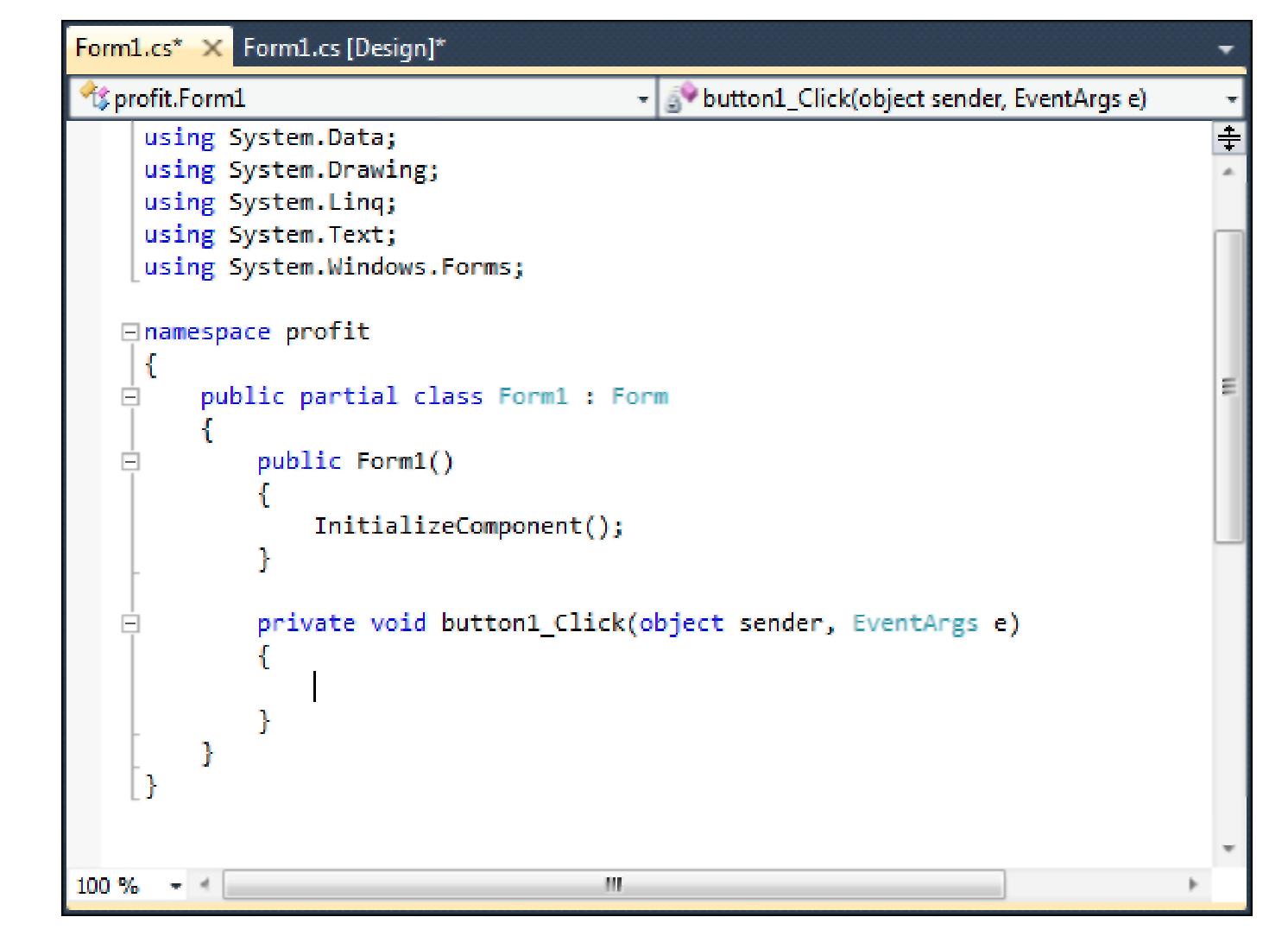


Рисунок 22. Шаблон функции (метода) обработки события

Функция buttonlciick вычисляет доход по вкладу и выводит результат расчета в поле компонента label3. Исходные данные (сумма и срок вклада) вводятся из полей редактирования textBoxl и textBox2 путем обращения к свойству Text. Значением свойства Text является строка, которая находится в поле редактирования. Свойство Text строкового типа, поэтому для преобразования строк в числа используются принадлежащие пространству имен system, convert функции ToDouble и Toint32. Следует обратить внимание, что функция ToDouble возвращает результат только в том случае, если строка, переданная ей в качестве параметра, является изображением дробного числа, что предполагает использование запятой в качестве десятичного разделителя (при стандартной для России настройке операционной системы). Аналогично, параметр функции Toint32 должен представлять собой строку, являющуюся изображением целого числа.

Другие функции преобразования строк приведены в таблице 10.

Таблица 10. Функции преобразования строк

|  |  |
| --- | --- |
| **Функция** | **Значение** |
| ToSingle(s), ToDouble(s) | Дробное Single, Double |
| ToByte(s), ToIntl6(s), ToInt32(s), ToInt64 (s), | Целое типа Byte, Intl6, Int32, Int64 |
| ToUIntl6(s), ToUInt32(s), ToUInt64(s) | Целое типа uintl6, uint32, uint64 |

**Пространство имен**

Концепция пространства имен является развитием концепции модулей. Пространство имен позволяет избежать конфликта имен, дает программисту свободу в выборе идентификаторов. Так, например, при объявлении функции можно не заботиться об уникальности ее имени, достаточно объявить эту функцию в новом пространстве имен.

В приведенной в листинге 2.1 функции для преобразования строки в дробное число используется функция ToDouble. Она принадлежит пространству имен System. Convert, на что указывает префикс перед именем функции (строго говоря, функция ToDouble — это метод объекта Convert, который принадлежит пространству имен System).

Пространство имен (namespace) — это контейнер (модуль), который предоставляет программе, использующей этот модуль, свои объекты (типы, функции, константы и т.д.). Например, пространство имен System.windows.Forms содержит объекты Label, TextBox, Button И др.

Каждый объект является элементом какого-либо пространства имен. Например, поле редактирования, объект типа TextBox, является элементом или, как принято говорить, принадлежит пространству имен System.Windows.Forms.

Пространства имен, которые использует программа, указывается в инструкции using. Например, в начале модуля формы (cs-файл) есть ссылки на пространства имен System, System.Windows.Forms, System.Drawing И др.

Для того чтобы получить доступ к объекту пространства имен (например, методу или константе), следует перед именем объекта указать идентификатор пространства имен, которому принадлежит объект, разделив идентификатор и имя объекта точкой. Например, инструкция

*n = System.Convert.ToSingle(TextBoxl.Text);*

показывает, что для преобразования строки в число используется метод ToSingle объекта Convert, который принадлежит пространству имен System

Вычисленные значения процентной ставки и величины дохода выводятся в поле label3 путем присваивания значения свойству Text. Для преобразования дробного числа в строку (свойство Text строкового типа) используется функция (метод) Tostring. Параметр метода Tostring задает формат строки-результата: "с" — финансовый (от англ. currency); "n" — числовой (от англ. number). Следует обратить внимание, что при использовании финансового формата после числового значения выводится обозначение денежной единицы (в соответствии с настройкой операционной системы). В таблице 11 приведены возможные форматы представления числовой информации.

Таблица 11. Форматы представления чисел

|  |  |  |
| --- | --- | --- |
| **Параметр функции**  **ToString** | **Формат** | **Пример** |
| "с" | Currency— финансовый (денежный). Используется для представления денежных величин. Обозначение денежной единицы, разделитель групп разрядов, способ отображения отрицательных чисел определяют соответствующие настройки операционной системы | 55 055,28 р. |
| "е" | Scientific (exponential) — научный. Используется для представления очень маленьких или очень больших чисел. Разделитель целой и дробной частей числа задается в настройках операционной системы | 5.50528+Е004 |
| "f" | Fixed— число с фиксированным десятичным разделителем. Используется для представления дробных чисел. Количество цифр дробной части, способ отображения отрицательных чисел определяют соответствующие настройки операционной системы | 55 055,28 |
| "п" | Nurrber — числовой. Используется для представления дробных чисел. Количество цифр дробной части, символ-разделитель групп разрядов, способ отображения отрицательных чисел определяют соответствующие настройки операционной системы | 55 055,28 |
| "д" | General—универсальный формат. Похож на Nurrber, но разряды не разделены на группы | 55055,275 |
| "г" | Roundtrip — без округления. В отличие от формата N, этот формат не выполняет округления (количество цифр дробной части зависит от значения числа) | 55 055,2775 |

**Структура проекта**

Проект представляет собой совокупность файлов, которые компилятор использует для создания выполняемого файла. Структура проекта отображается в окне Solution Explorer (рисунок 23).

Основными элементами проекта являются:

* главный модуль приложения (файл Program.css);
* модули форм.

**Главный модуль**

В главном модуле находится функция Main, с которой начинается выполнение программы. Функция Main создает стартовую форму (имя класса стартовой формы

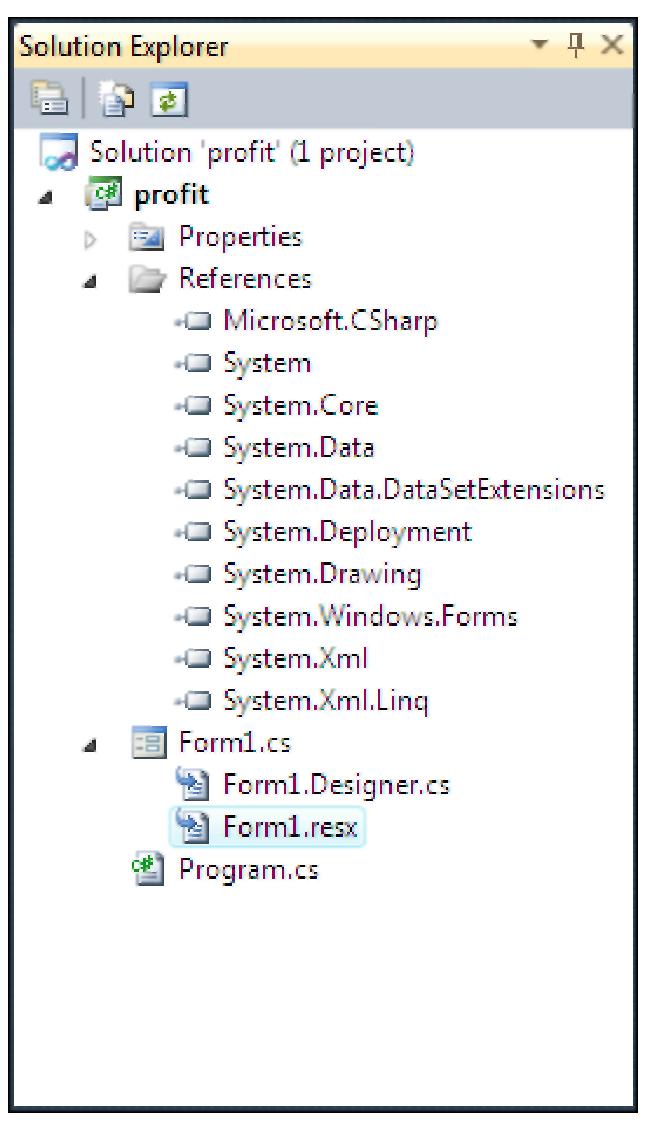
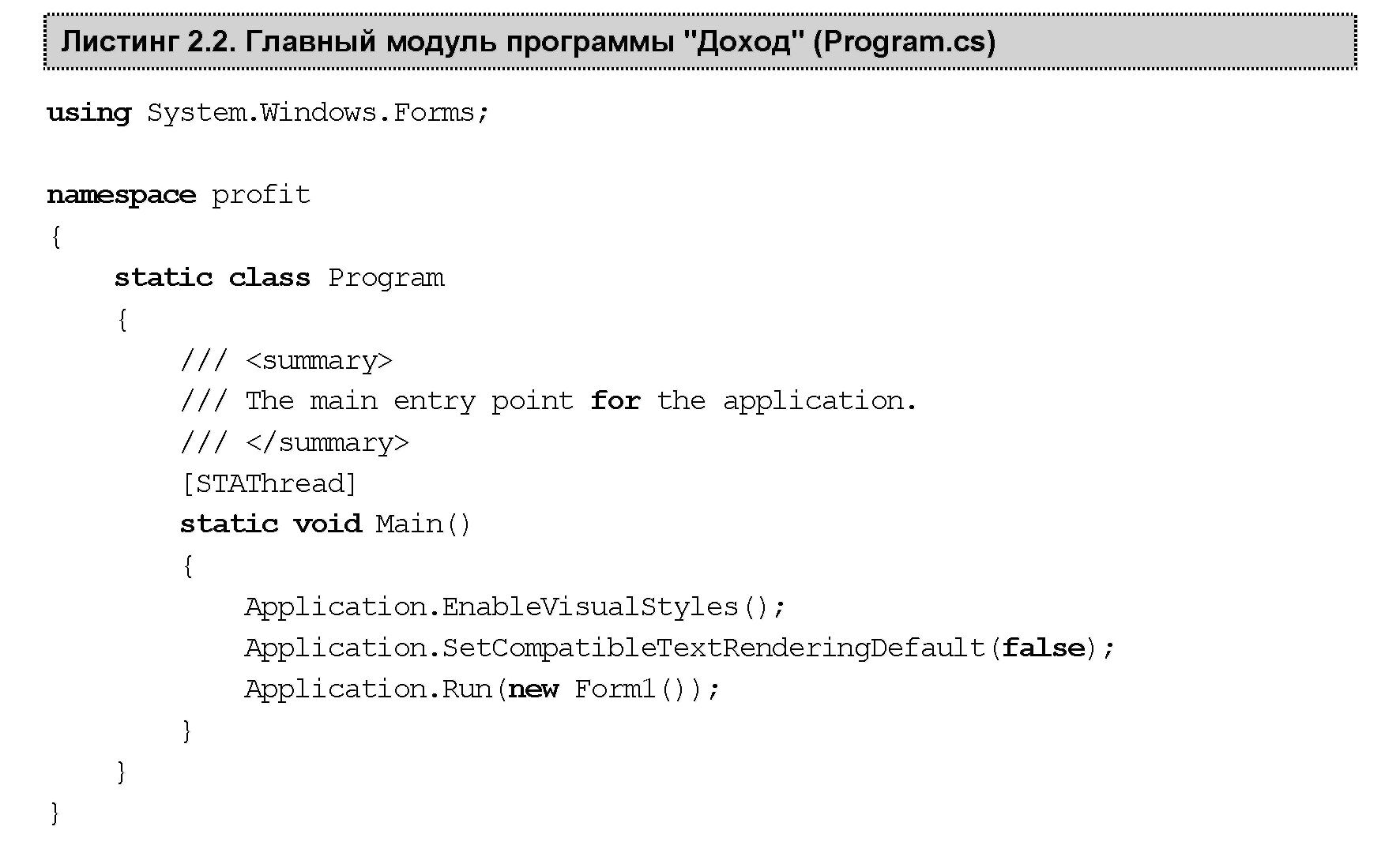


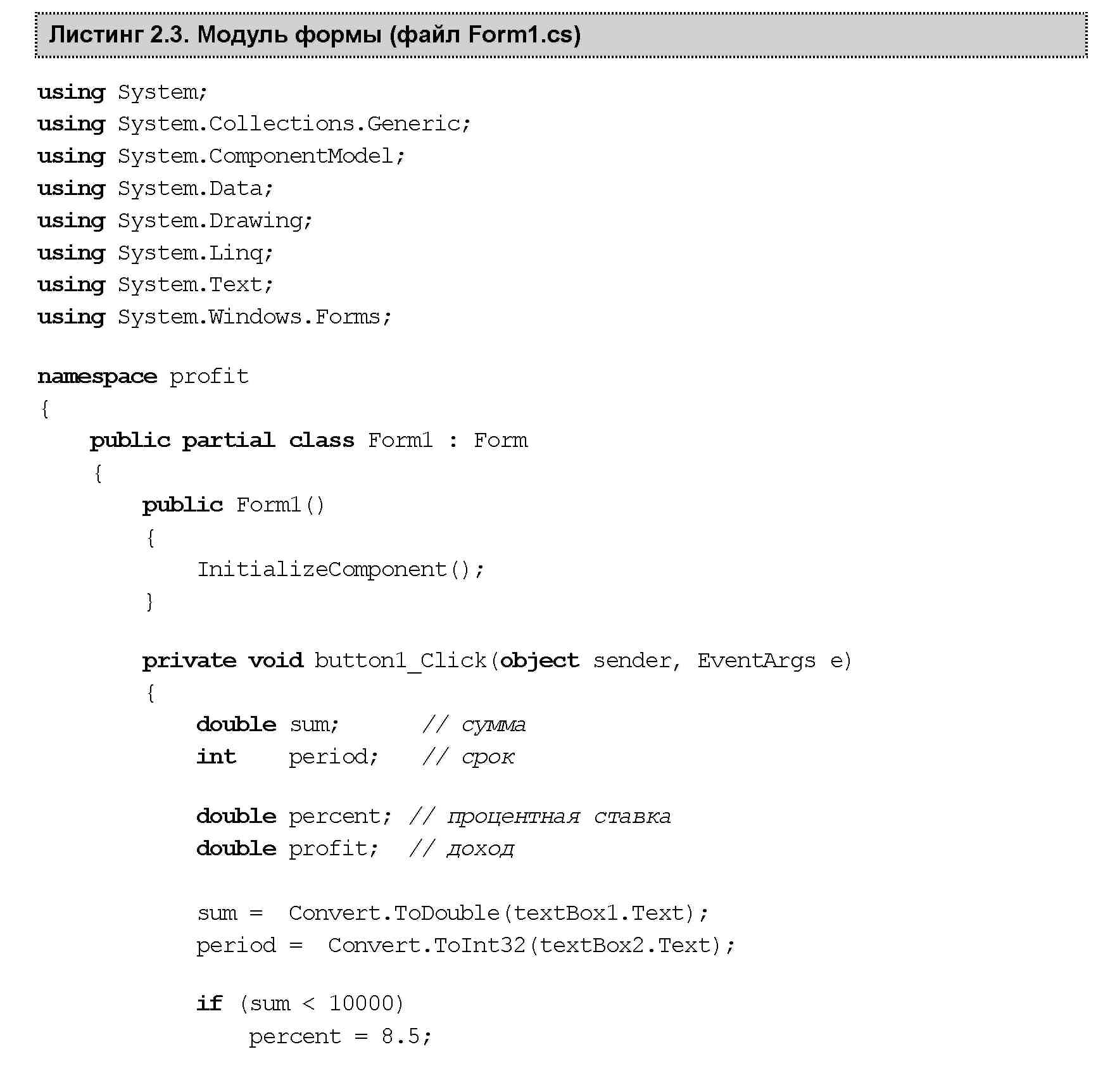
Рисунок 23. Структура проекта отображается в окне Solution Explorer

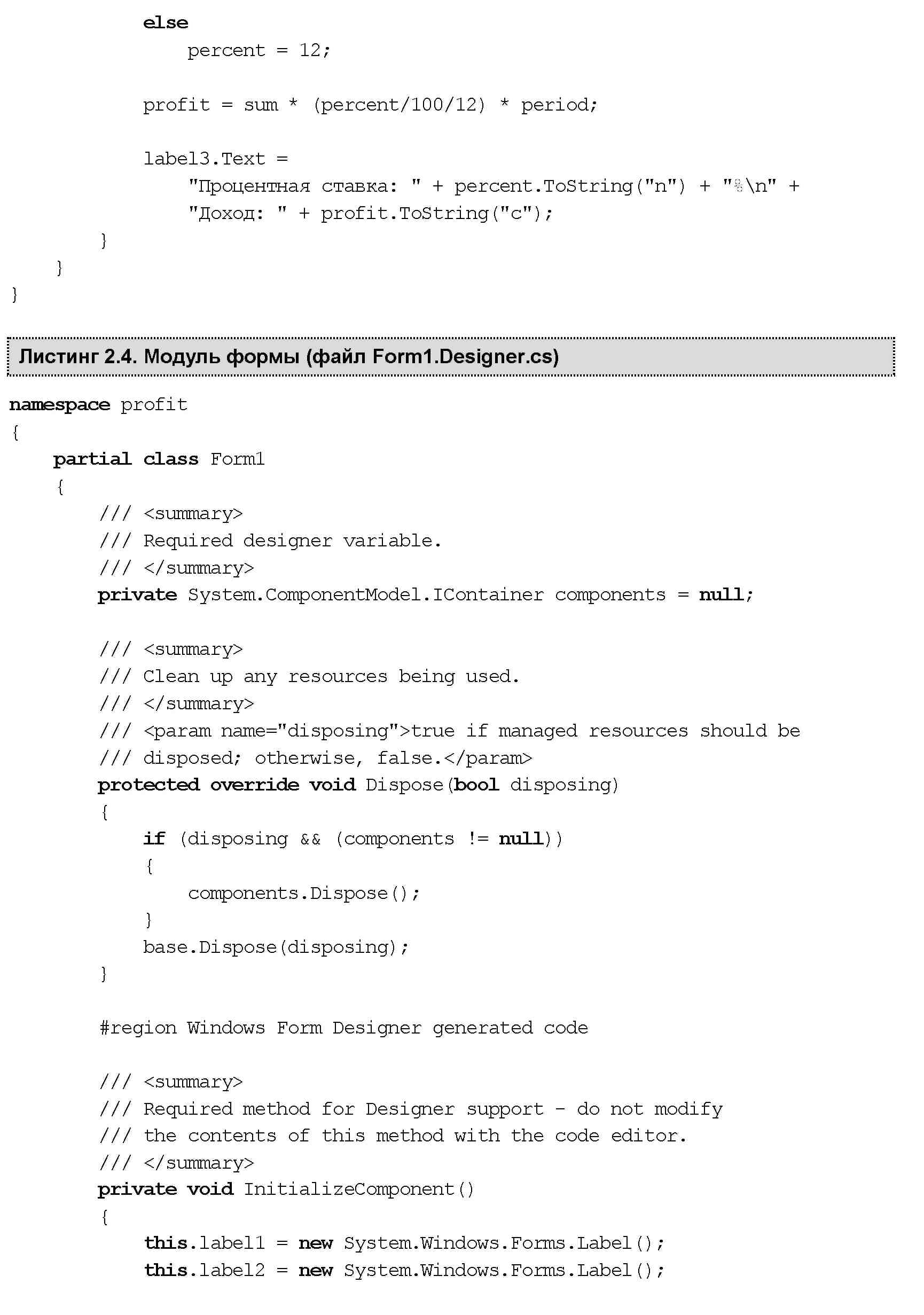
указывается в качестве параметра метода Run), в результате чего на экране появляется окно программы. Главный модуль программы "Доход" приведен в листинге 2.2.



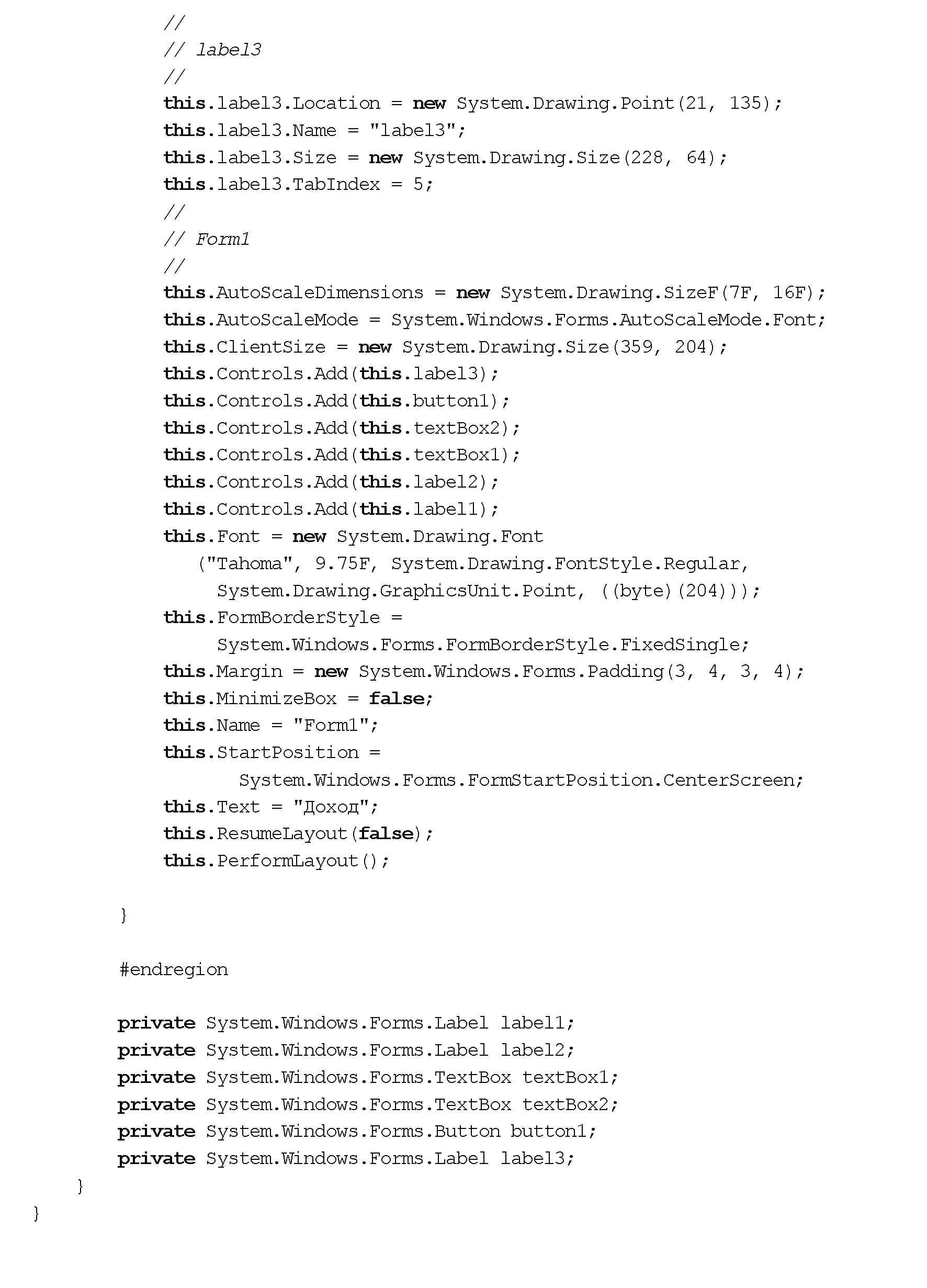
**Модуль формы**

Модуль формы содержит объявление класса формы. Физически модуль формы разделен на два файла: Forml.cs и Forml.Designer.cs (листинги 2.3 и 2.4 соответственно). В файле Forml.cs находятся функции (методы класса формы) обработки событий формы и ее компонентов. В файле Forml.Designer.cs (чтобы его увидеть, надо в окне Solution Explorer сделать двойной щелчок на имени файла) находится объявление класса формы, в том числе сформированная дизайнером формы функция initializeComponent, обеспечивающая создание и настройку компонентов. Следует обратить внимание на секцию Windows Form Designer generated code (секция — фрагмент кода, находящийся между директивами #region и #endregion). В ней находится функция initializeComponent, обеспечивающая непосредственно создание и инициализацию формы и компонентов.









**Сохранение проекта**

Как было сказано раньше, в момент создания проекта среда разработки в папке временных проектов (по умолчанию это C:\Users\User\AppData\Local\Temporary Projects, где User— имя пользователя в системе) создает каталог проекта. Чтобы программист мог работать с программой в дальнейшем, проект надо сохранить явно. Для этого в меню File надо выбрать команду Save All и в появившемся окне Save Project нажать кнопку Save (рис. 2.24).

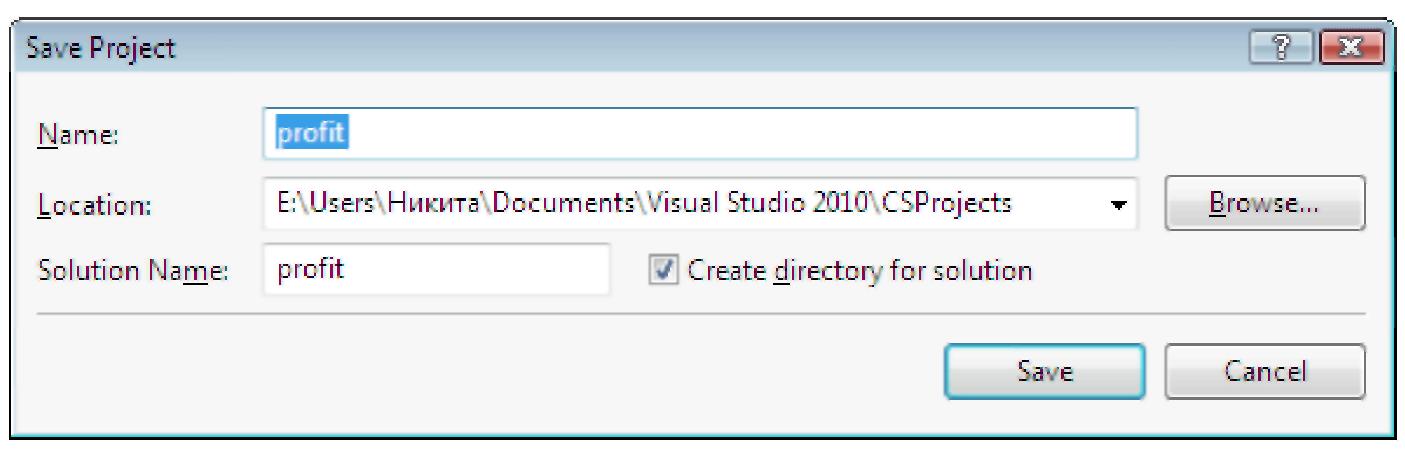


Рисунок 24. Сохранение проекта

Следует обратить внимание, что в момент сохранения проекта в папке, имя которой указано в поле Location, для сохраняемого проекта будет создана новая папка (если установлен флажок Create directory for solution).

**Компиляция**

Процесс преобразования исходной программы в выполняемую называется компиляцией или построением (build). Укрупненно процесс построения программы можно представить как последовательность двух этапов: компиляция и компоновка. На этапе компиляции выполняется перевод исходной программы (модулей) в некоторое внутреннее представление. На этапе компоновки — объединение модулей в единую программу.

Процесс построения программы активизируется в результате выбора в меню Debug команды Build solution, а также в результате запуска программы из среды разработки (меню Debug, команда Start Debugging), если с момента последней компиляции в программу были внесены изменения.

Результат компиляции отражается в окне Error List. Если в программе нет ошибок, то по завершении процесса компиляции окно Error List выглядит так, как показано на рис. 2.25.

Если в процессе построения в программе обнаруживаются ошибки, то в окне Error List (рисунок 26) выводится их список. Чтобы перейти к фрагменту кода, содержащего ошибку, надо сделать двойной щелчок левой кнопкой мыши в строке сообщения об этой ошибке.

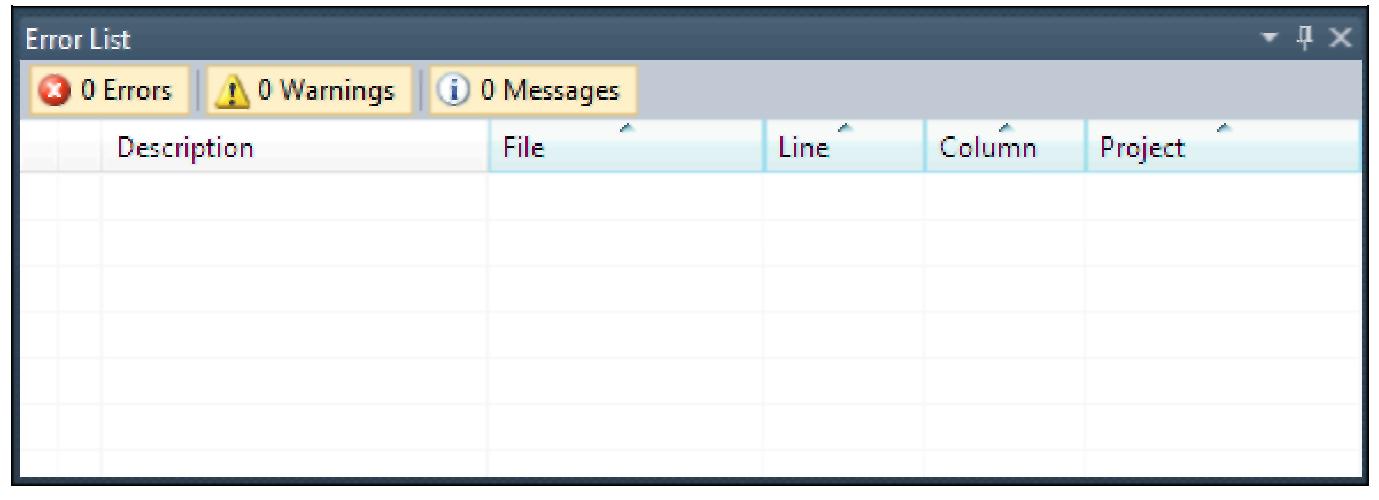


Рисунок 25. Результат построения (в программе ошибок нет)

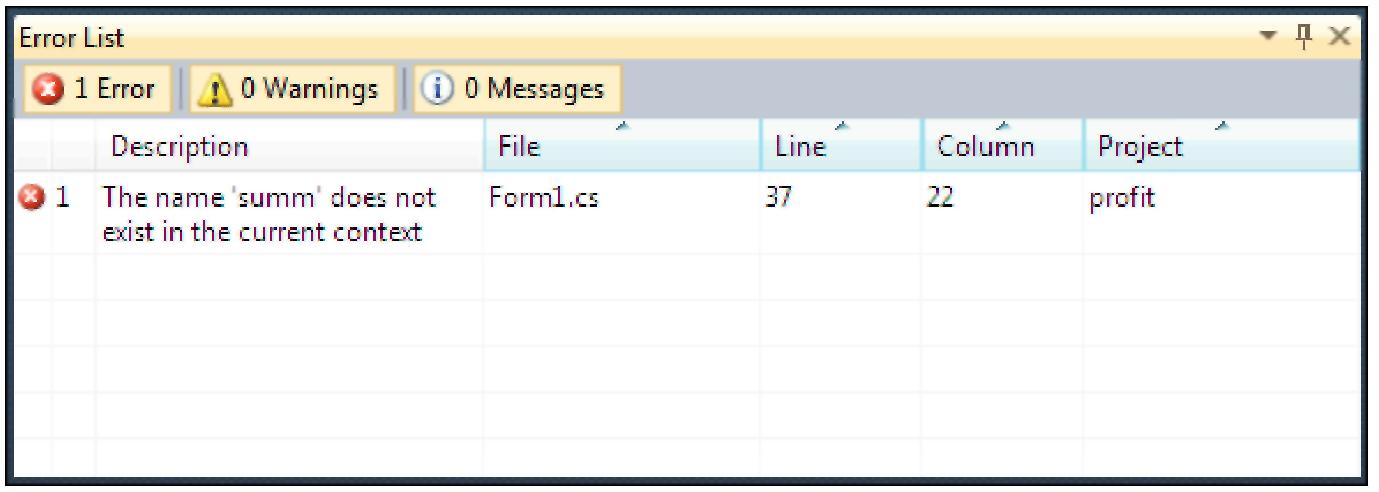


Рисунок 26. Результат построения (в программе есть ошибка)

**Ошибки**

Компилятор генерирует выполняемую программу (ехе-файл) только в том случае, если в исходной программе (в тексте) нет ошибок.

Если в программе есть ошибки, то программист должен их устранить. Процесс устранения ошибок носит итерационный характер. Обычно сначала устраняются наиболее очевидные ошибки, например, объявляются необъявленные переменные, затем, после выполнения повторной компиляции, — остальные.

В таблице 12 приведены сообщения компилятора о типичных ошибках.

Таблица 12. Сообщения компилятора об ошибках

|  |  |
| --- | --- |
| **Сообщение компилятора** | **Вероятная причина ошибки** |
| The name идентификатор does not exist in the current context  (В текущем контексте имя не существует) | 1. Используемая в программе переменная не объявлена.  2. Ошибка при записи имени переменной. Напри мер, объявлена переменная sum, а в тексте про граммы написано: Sum |
| Cannot implicitly convert type typel to type2  (Невозможно преобразовать значение типа typel в значение типа type2)  Пример: Cannot implicitly convert type 'double' to 'int' | В инструкции присваивания тип выражения не соответствует типу переменной, которой присваивается значение. Например, если переменные п и m целого типа, то инструкция п = т/12 неверная, т. к. выражение т/12 дробное |
| Use of unassigned local variable  (Используется локальная переменная, которой не присвоено начальное значение) | В программе нет инструкции, присваивающей переменной начальное значение |
| Пространство имен does not contain definition for Идентификатор  (Пространство имен Пространство имен  не содержит определение идентификатора Идентификатор) | Неправильно (например, не в том регистре) записано имя пространства имен или идентификатор (например, имя функции) ему принадлежащий. Пример:  System. Convert.toint32(textBoxl.Text) —  неправильно записано имя функции. Должно быть  ToInt32 |
| ';' expected  (ожидается символ "точка с запятой") | После инструкции нет символа "точка с запятой" |

Следует обратить внимание на то, что компилятор языка С# различает прописные и строчные буквы. Запись имени переменной или функции "не в том регистре" — типичная причина ошибок в программе.

**Предупреждения**

В программе могут быть не только ошибки, но и неточности. Например, инструкция присваивания целой переменной дробного значения формально является верной. Присвоить значение переменной можно, но что делать с дробной частью? Отбросить или округлить? Что хотел сделать программист, записав эту инструкцию?

При обнаружении в программе неточностей компилятор выводит предупреждения — Warnings. Например, при обнаружении не объявленной, но не используемой переменной выводится сообщение: unreferenced local variable. Действительно, зачем объявлять переменную и не использовать ее?

В таблице 13 приведены предупреждения и подсказки компилятора о типичных неточностях в программе.

Таблица 13. Предупреждения и подсказки компилятора

|  |  |
| --- | --- |
| **Сообщение** | **Причина** |
| The variable ... is declared but never used | Переменная объявлена, но не используется |
| The variable ... is assigned but its value is never used | Переменной присвоено значение, но оно не используется |

**Запуск программы**

Пробный запуск программы можно выполнить из Visual Studio, не завершая работу со средой разработки. Для этого в меню Debug надо выбрать команду Start Debugging. Можно также сделать щелчок на находящейся в панели инструментов Debug кнопке Start Debugging (рисунок 27) или нажать клавишу <F5>.

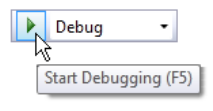


Рисунок 27 Чтобы запустить программу, сделайте щелчок на кнопке Start Debugging

**Исключения**

Ошибки, возникающие во время работы программы, называют исключениями. В большинстве случаев причиной исключений (exception) являются неверные данные. Например, если в поле Сумма окна программы "Доход" ввести, скажем, 100.50 и сделать щелчок на кнопке Расчет, то на экране появится окно (рисунок 28) с сообщением о возникновении исключения FormatException: "Input string was not in correct format" ("Неверный формат введенной строки"). Кроме этого, среда разработки делает активным окно редактора кода, в котором выделяется инструкция программы, при выполнении которой произошла ошибка (возникло исключение).

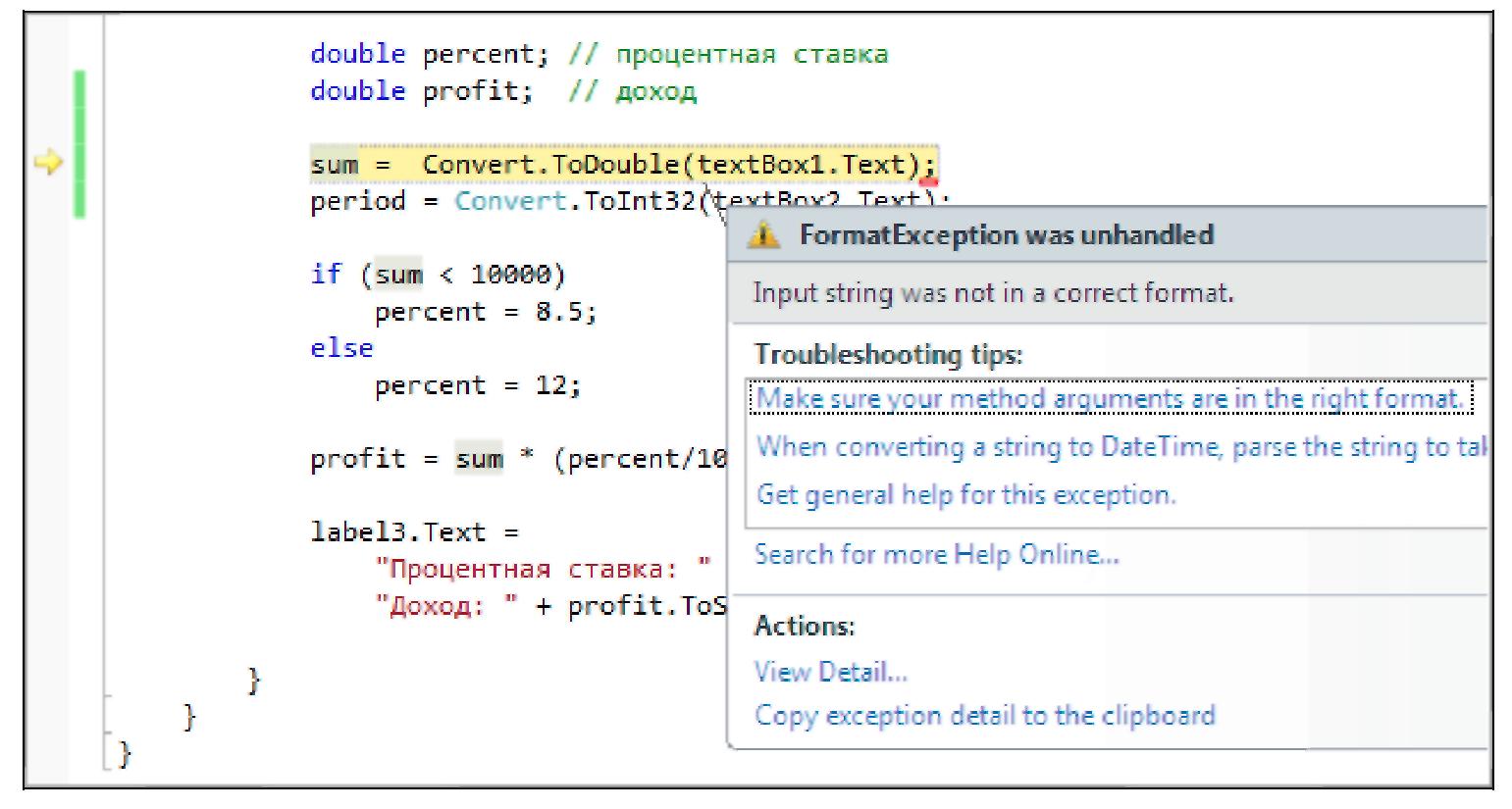


Рисунок 28 Пример сообщения об исключении — ошибке, произошедшей во время работы программы (программа запущена из среды разработки)

Причина возникновения исключения в рассматриваемом примере в следующем. Преобразование строки в число выполняет функция ToDouble. Эта функция работает правильно, если ее параметром действительно является строковое представление дробного числа, что при стандартной для России настройке операционной системы предполагает использование в качестве десятичного разделителя запятой. В рассматриваемом примере строка 100.50 не является строковым представлением дробного числа, т. к. в качестве десятичного разделителя указана точка, и, поэтому, возникает исключение FormatException— ошибка формата. Исключение "ошибка формата" произойдет и в том случае, если в поле Срок будет введено дробное значение. Причина— попытка преобразовать в целое значение строку, которая не является изображением целого числа. Это же исключение произойдет и в том случае, если какое-либо из полей ввода оставить незаполненным.

Для того чтобы остановить программу, во время работы которой возникло исключение, надо в меню Debug выбрать команду Stop Debugging или нажать комбинацию клавиш <Shift>+<F5>.

Если программа запущена из операционной системы (из Windows), то при возникновении исключения так же, как и в случае запуска программы из среды разработки, выводится сообщение об ошибке (рисунок 29). Чтобы остановить работу программы, надо нажать кнопку Quit. Щелчок на кнопке Continue разрешает продолжить выполнение программы, несмотря на возникшую ошибку.

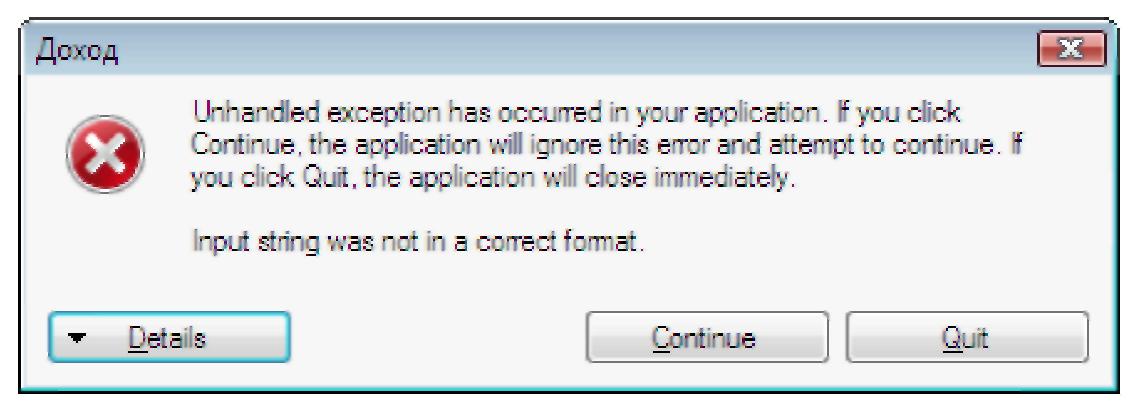


Рисунок 29. Пример сообщения о возникновении исключения (программа запущена из операционной системы)

**Обработка исключения**

По умолчанию обработку исключений берет на себя автоматически добавляемый в выполняемую программу код, который обеспечивает вывод сообщения об ошибке и завершение работы программы, при выполнении которой возникло исключение. Вместе с тем программист может поместить в программу код, который выполнит обработку исключения.

В простейшем случае инструкция обработки исключения в общем виде выглядит так:

try {

// Здесь инструкции, при выполнении которых

// мажет возникнуть исключение }

catch {Тип Исключения )

{

// Здесь инструкции обработки исключения

}

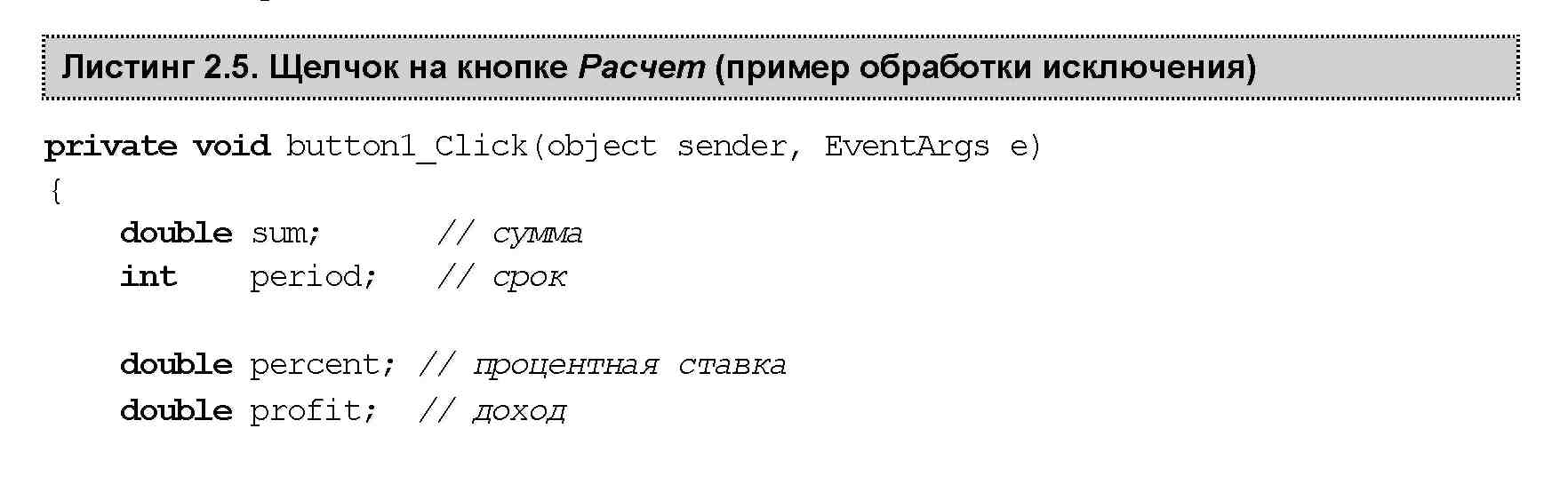
Ключевое слово try указывает, что далее следуют инструкции, при выполнении которых возможно возникновение исключений, и что обработку этих исключений берет на себя программа. Слово catch обозначает начало секции обработки исключений. После слова catch указывается тип исключения, обработку которого берет на себя программа. Далее следуют инструкции, обеспечивающие обработку исключения. Нужно обратить внимание на то, что инструкции секции try, следующие за той, при выполнении которой возникло исключение, после обработки исключения не выполняются.

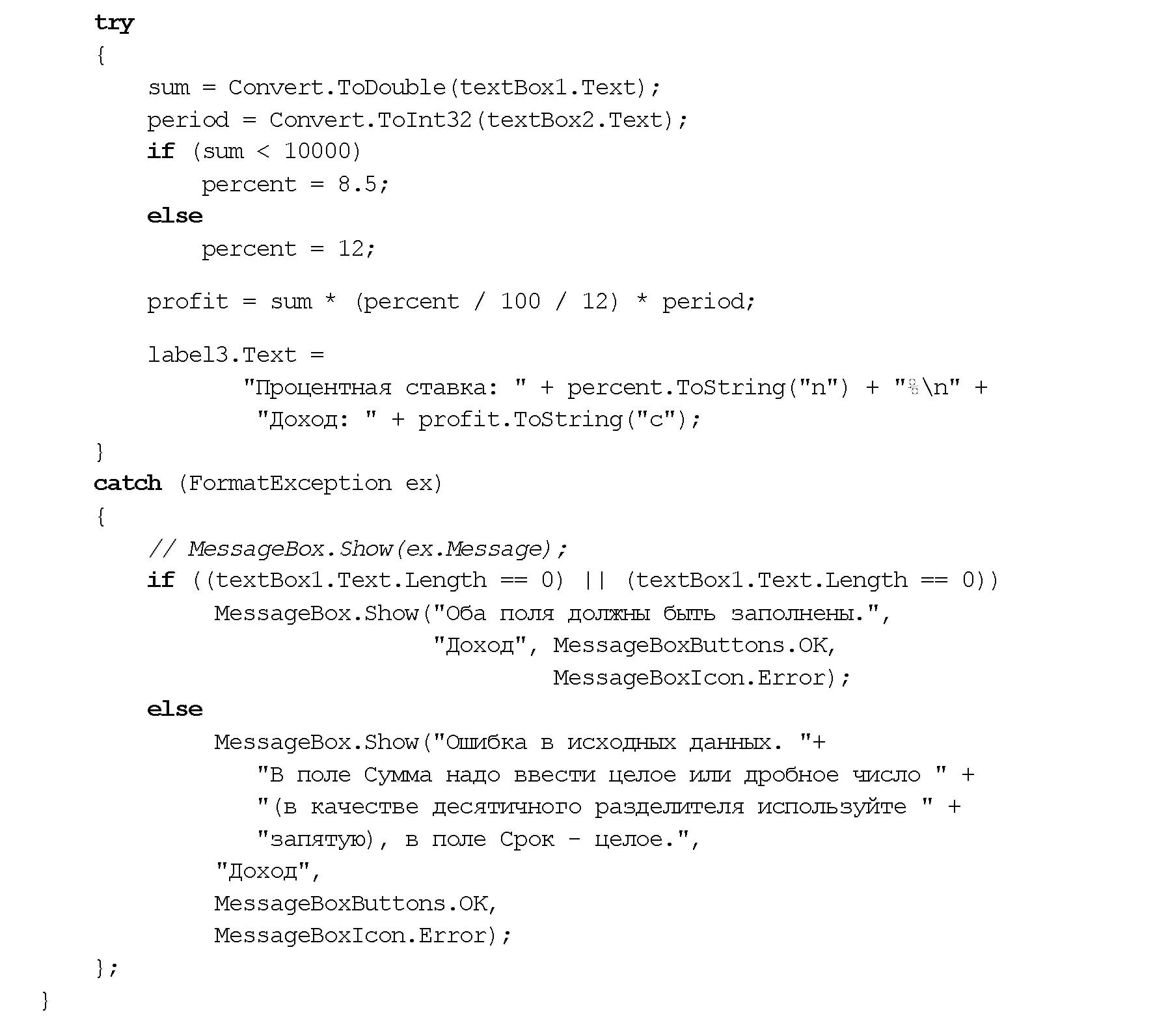
В таблице 14 перечислены некоторые из возможных исключений и указаны вероятные причины их возникновения.

Таблица 14. Типичные исключения

|  |  |
| --- | --- |
| **Исключение** | **Возникает** |
| FormatException —  ошибка формата (преобразования) | При выполнении преобразования, если преобразуемая величина не может быть приведена к требуемому типу. Наиболее часто возникает при преобразовании строки символов в число |
| IndexCutOfRangeException—  выход значения индекса за допустимые границы | При обращении к несуществующему элементу массива |
| ArgumentCutOfRangeException —  выход значения аргумента за допустимые границы | При обращении к несуществующему элементу данных, например, при выполнении операций со строками |
| OverflowException—  переполнение | Если результат выполнения операции выходит за границы допустимого диапазона, а также при выполнении операции деления, если делитель равен нулю |

В качестве примера обработки исключения в листинге 2.5 приведена функция обработки события click для кнопки Расчет программы "Доход". При возникновении исключения FormatException программа определяет причину (какое из полей формы не заполнено или содержит неверные данные) и выводит соответствующее сообщение (рисунок 30).





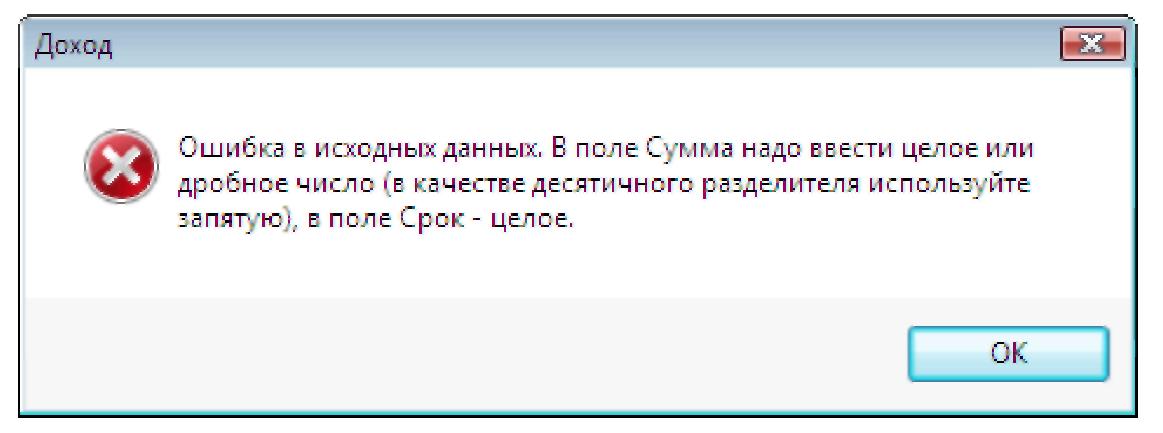


Рисунок 30. Пример сообщения об ошибке

В приведенной функции обработки события click для вывода сообщения о неверных данных используется функция MessageBox.show, инструкция вызова которой в общем виде выглядит так:

г = MessageBox.show{Сообщение, Заголовок, Кнопки, ТипСообщения, КнопкаПоУмолчанию)

где:

* сообщение — текст сообщения;
* заголовок — текст в заголовке окна сообщения;
* кнопки— кнопки, отображаемые в окне сообщения (таблице 15);
* тип— тип сообщения. Сообщение может быть информационным, предупреждающим или сообщением об ошибке. Каждому типу сообщения соответствует значок (таблице 16);
* кнопкаПоУмолчанию— порядковый номер кнопки, на которой находится фокус при появлении окна сообщения на экране (таблице 17).

Значение (тип system.windows.Forms.DiaiogResuit), возвращаемое функцией MessageBox.show, позволяет определить, какая кнопка была нажата пользователем для завершения диалога (таблице 18). Если в окне сообщения отображается одна кнопка (очевидно, что в этом случае не нужно проверять, какую кнопку нажал пользователь), то функцию MessageBox.show можно вызвать как процедуру.

Таблица 15. Идентификаторы кнопок

|  |  |
| --- | --- |
| **Значение параметра** | **Кнопки, отображаемые в окне сообщения** |
| MessageBoxButtons.OK | OK |
| MessageBoxButtons.YesNo | Yes, No (Да, Нет) |
| MessageBoxButtons.YesNoCancel | Yes, No, Cancel (Да, Нет, Отменить) |

Таблица 2.16. Тип сообщения

|  |  |  |
| --- | --- | --- |
| Сообщение | Тип сообщения | Значок |
| Warning — Внимание | MessageBoxIcon .Warning |  |
| Error — Ошибка | MessageBoxIcon.Error |  |
| Information — Информация | MessageBoxIcon.Information |  |

Таблица 17. Активная кнопка

|  |  |
| --- | --- |
| Значение параметра | Номер активной кнопки |
| System.Windows .Forms .MessageBoxDefaultButton.Buttonl | l |
| System.Windows.Forms .MessageBoxDefaultButton.Button2 | 2 |
| System.Windows .Forms .MessageBoxDefaultButton.Button3 | 3 |

Таблица 18. Значения функции MessageBox. show

|  |  |
| --- | --- |
| Значение | Нажата кнопка |
| System.Windows.Forms .DialogResult.Yes | Yes |
| System. Windows.Forms.DialogResult.No | No |
| System. Windows.Forms.DialogResult.Cancel | Cancel |

Внесение изменений

Программу "Доход" можно усовершенствовать. Например, сделать так, чтобы в поля редактирования пользователь мог ввести только числа (в поле Сумма — дробное число, в поле Срок — целое), чтобы в результате нажатия клавиши <Enter> в поле Сумма курсор переходил в поле Срок, а при нажатии этой же клавиши в поле Срок становилась активной кнопка Расчет. Кроме этого, можно сделать так, чтобы кнопка Расчет становилась доступной только после ввода данных в оба поля редактирования.

Чтобы внести изменения в программу, нужно открыть соответствующий проект. Для этого надо в меню File выбрать команду Open Project, открыть папку проекта и сделать щелчок на значке файла проекта (рисунок 31).

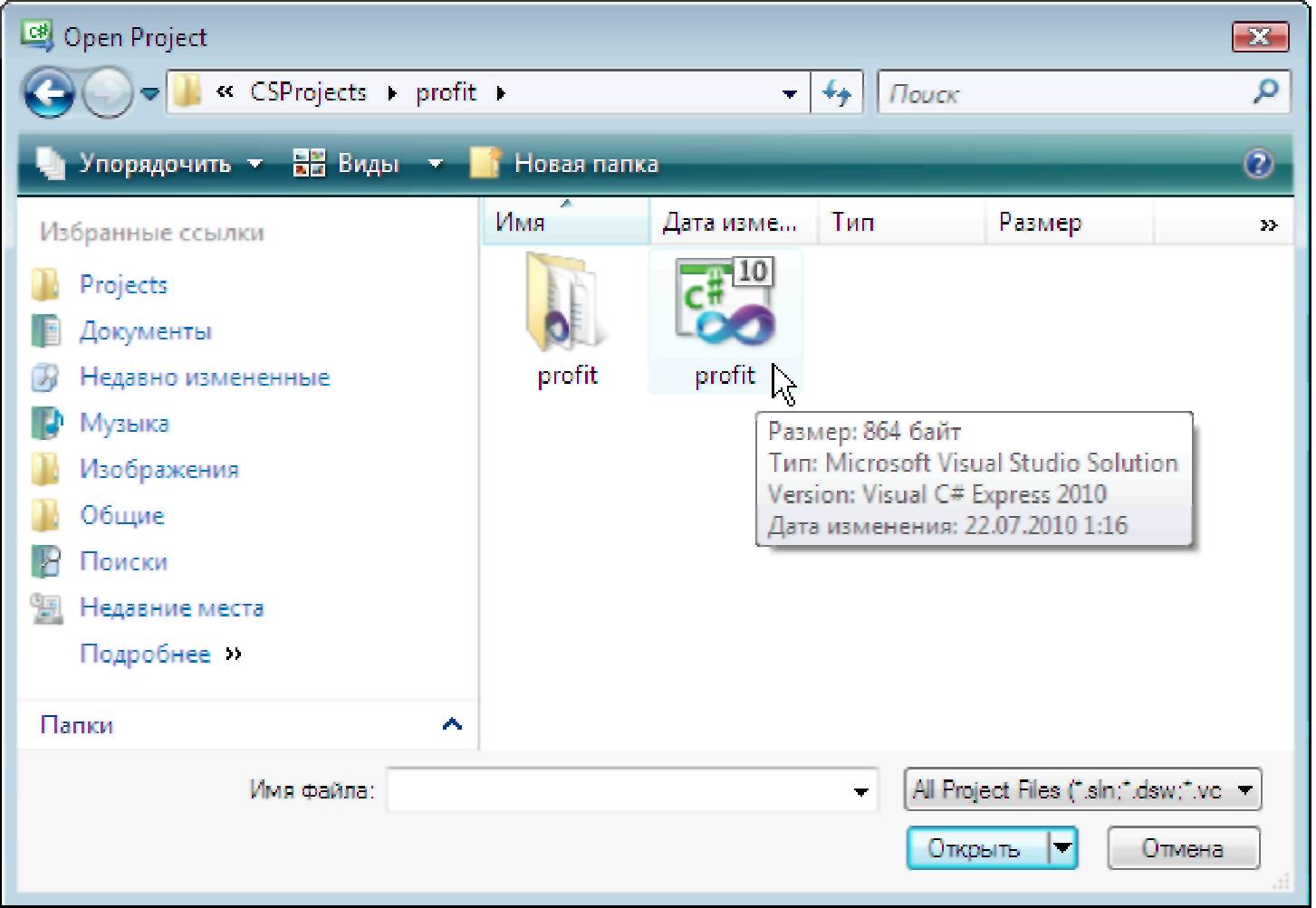


Рисунок 31. Загрузка проекта

Нужный проект для загрузки можно выбрать также из списка проектов, над которыми в последнее время работал программист. Этот список становится доступным в результате выбора в меню File команды Recent Projects and Solutions.

Чтобы программа "Доход" работала так, как было описано ранее, надо создать

функции обработки событий KeyPress И TextChanged ДЛЯ полей редактирования (компонентов textBoxl и textBox2). Функция обработки события KeyPress (для каждого компонента своя) фильтрует символы, вводимые пользователем. Она проверяет символ нажатой клавиши (символ передается в процедуру обработки события через параметр е) и, если символ "запрещен", присваивает значение True свойству Handled. В результате "запрещенный" символ в поле редактирования не появляется. Процедура обработки события TextChanged (событие возникает, если текст, находящийся в поле редактирования, изменился, например, в результате нажатия какой-либо клавиши в поле редактирования) управляет доступностью кнопки Расчет. Она проверяет, есть ли данные в полях редактирования, и, если в каком-либо из полей данных нет, присваивает свойству Enabled кнопки buttonl значение False (тем самым делает кнопку недоступной). Следует обратить внимание, что действие, которое надо выполнить, если изменилось содержимое поля textBoxl, ничем не отличается от действия, которое надо выполнить, если изменилось содержимое поля textBox2. Поэтому обработку события TextChanged для обоих компонентов может выполнить одна функция. Чтобы одна функция могла обрабатывать события разных компонентов, сначала надо создать функцию обработки события для одного компонента, а затем указать эту функцию в качестве обработчика соответствующего события другого компонента (рисунок 32).

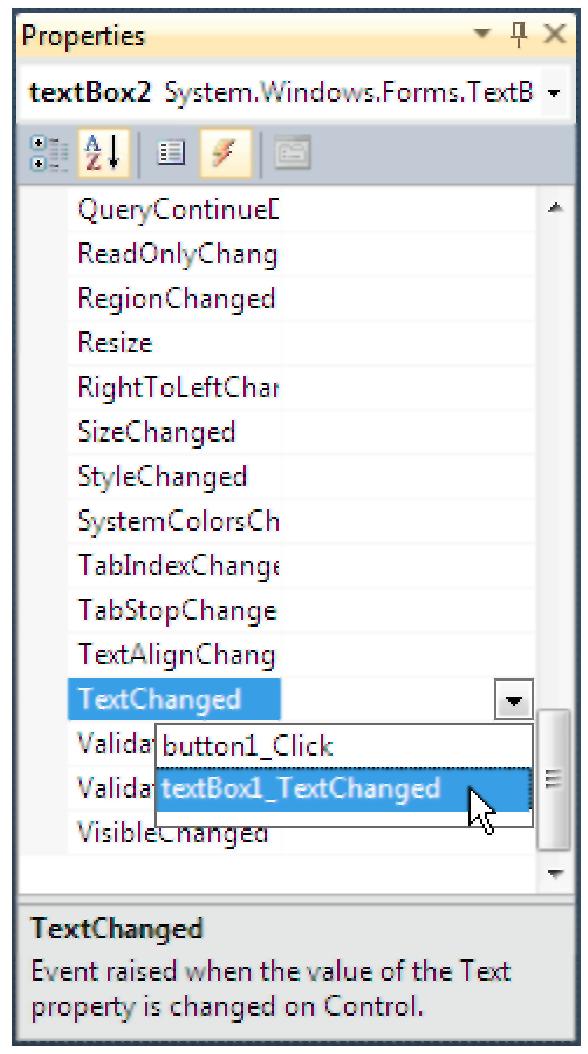
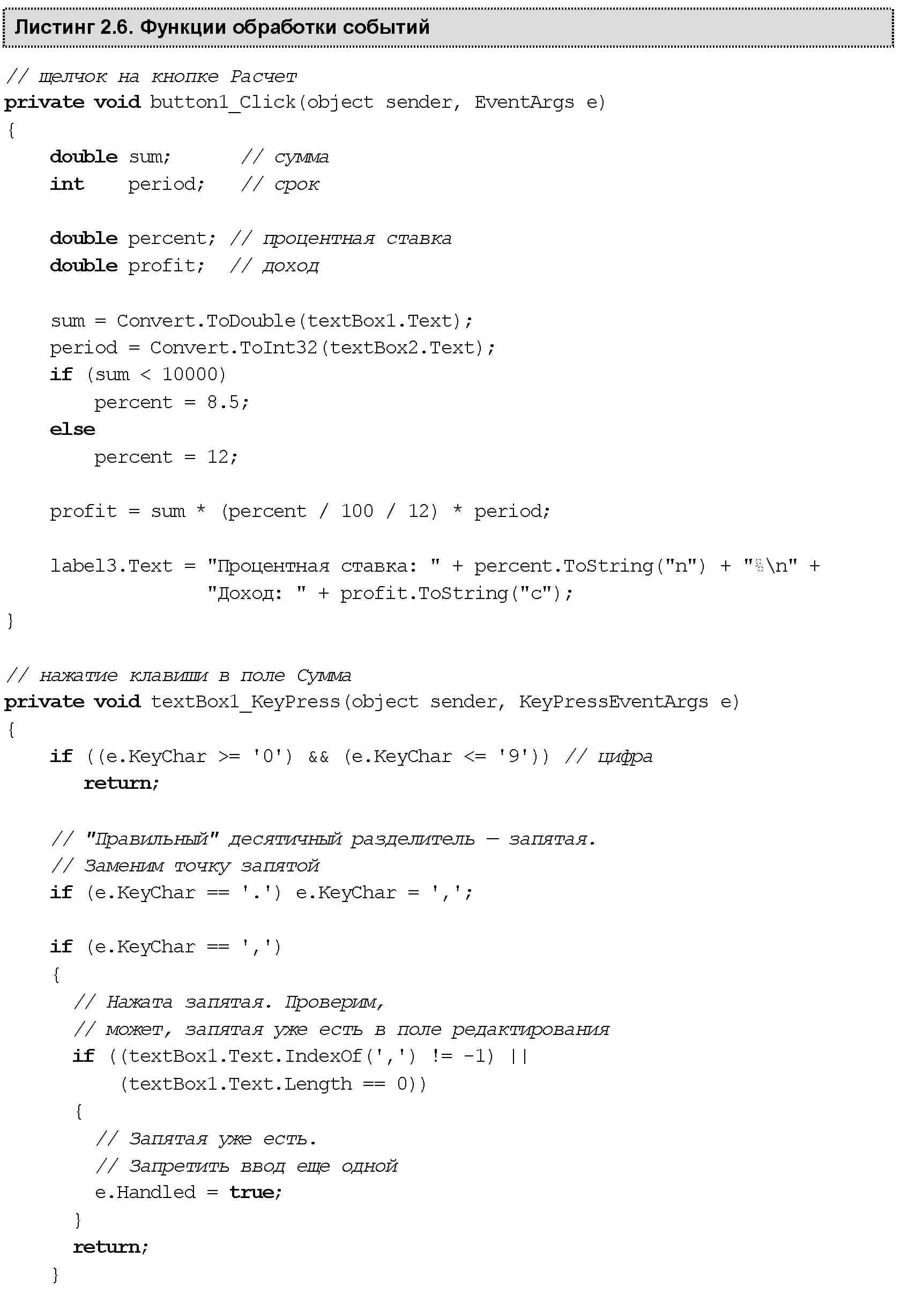


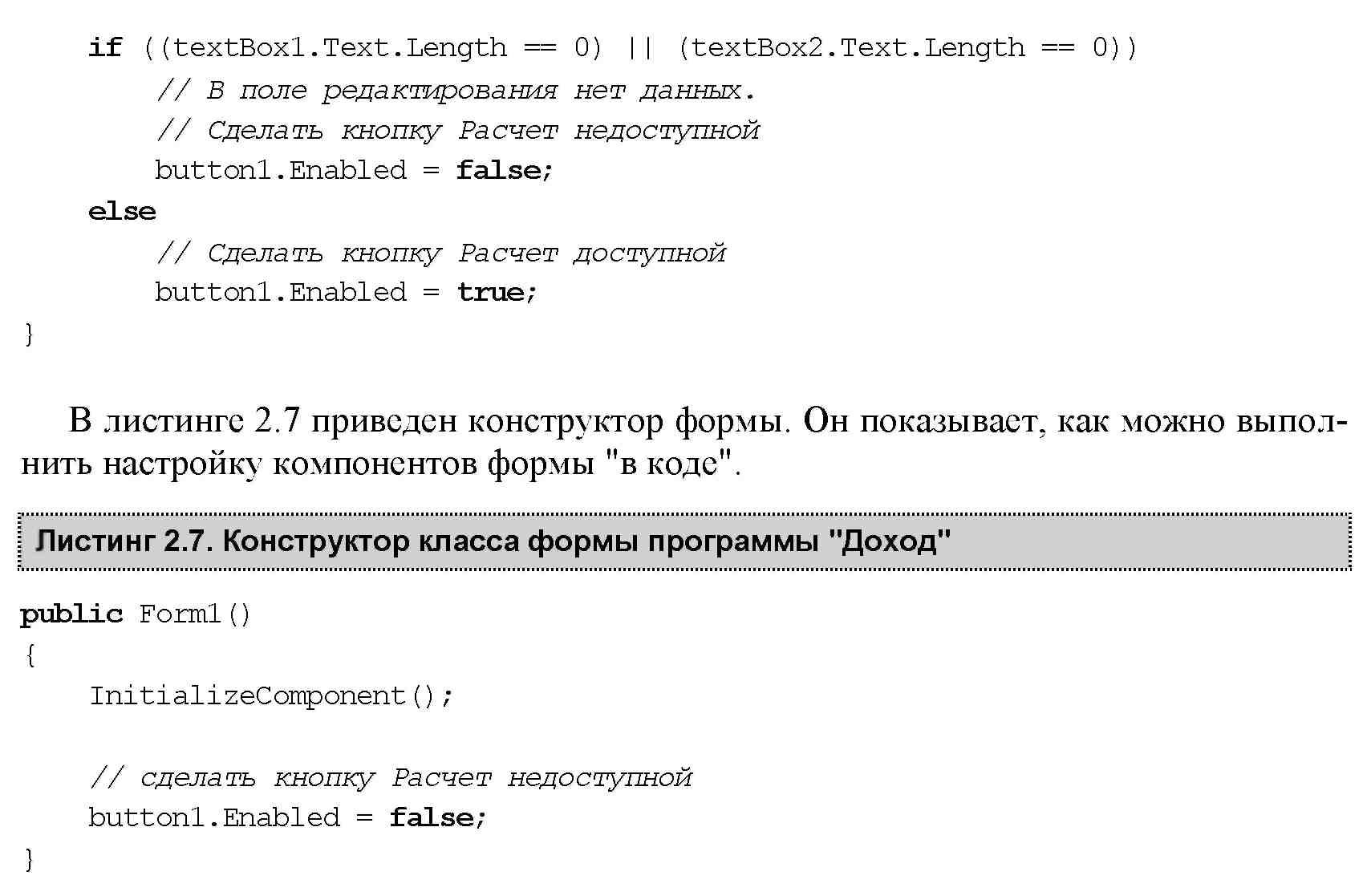
Рисунок 32. Выбор функции обработки события:

* обработку события TextChanged компонента textBox2
* выполняет функция обработки этого же события
* компонента textBoxl

Функции обработки указанных событий приведены в листинге 2.6. Обратите внимание, теперь в функции обработки события click кнопки Расчет нет инструкций обработки исключений. Теперь они не нужны. Исключения не могут возникнуть, потому что пользователь не сможет ввести в поля редактирования неверные данные, а кнопка Расчет становится доступной только после того, как будут заполнены оба поля.







**Завершение работы над проектом**

После того как приложение будет отлажено, можно выполнить его сборку. Для этого в меню Debug надо выбрать команду Build. В результате описанных действий в подкаталоге bin - Release каталога проекта будет создан выполняемый (ехе) файл программы.

Для завершения работы над проектом надо в меню File выбрать команду Close Solution.