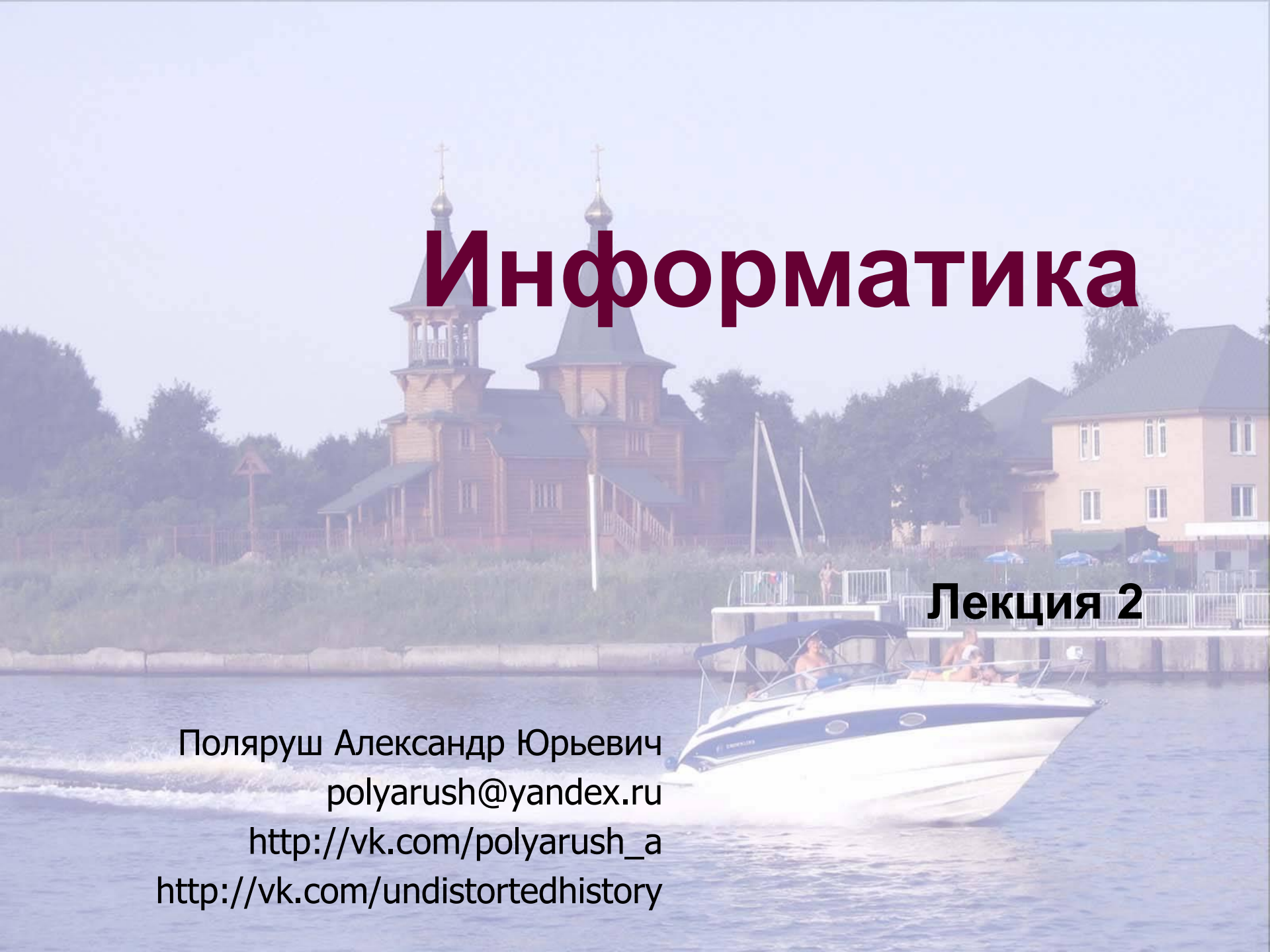


Информатика

Лекция 2

Поляруш Александр Юрьевич
polyarush@yandex.ru
http://vk.com/polyarush_a
http://vk.com/undistortedhistory

План лекции

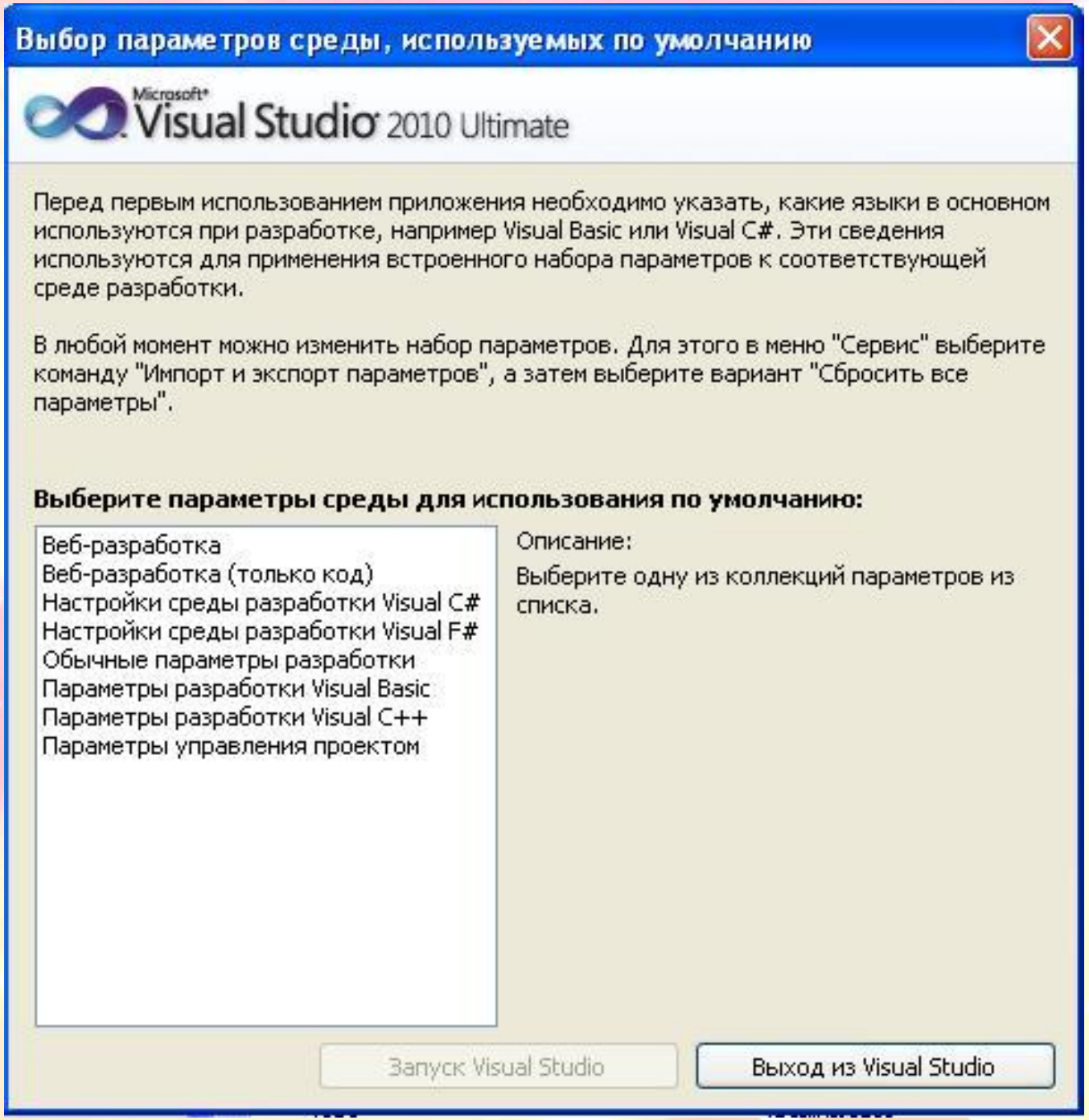
1. Этапы решения задачи на компьютере.
2. Понятие алгоритма. Свойства алгоритма.
3. Константы и переменные.
4. Понятие типов данных.
5. Арифметические выражения.
6. Стандартные функции.
7. Линейный алгоритм.
8. Алгоритм ветвления.
9. Циклический алгоритм.
10. Массивы данных.
11. Символьные данные.
12. Модульное программирование.

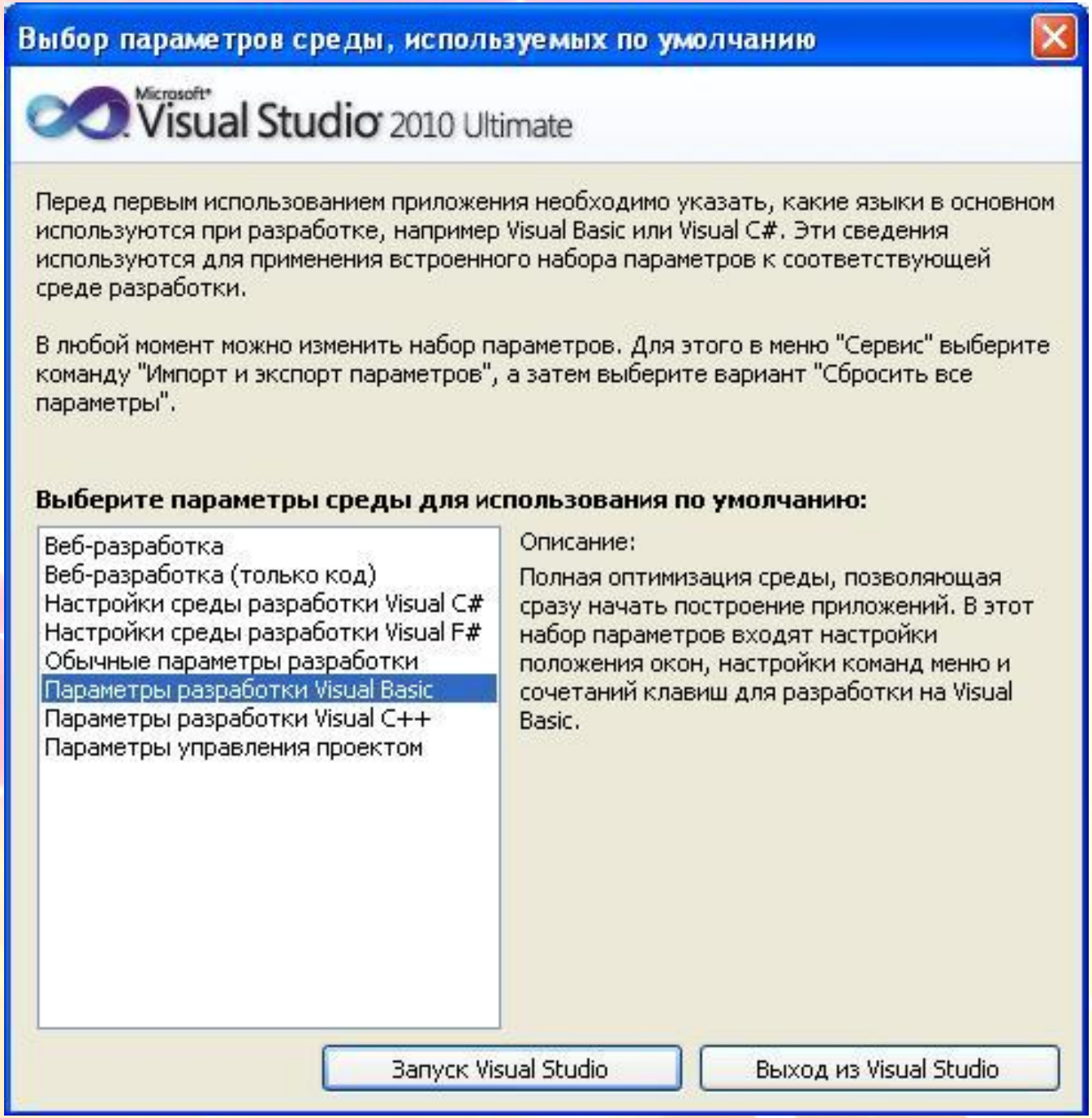
Где взять?

http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-basic-express не в моде совместимости

http://www.dreamspark.ru/ для студентов бесплатно

http://www.microsoft.com/downloads/ru-ru/details.aspx?FamilyID=f81412a2-d48e-4040-9b32-27eaf771c5db&displaylang=ru


Выбор параметров среды, используемых по умолчанию
Microsoft Visual Studio 2010 Ultimate
Перед первым использованием приложения необходимо указать, какие языки в основном используются при разработке, например Visual Basic или Visual C#. Эти сведения используются для применения встроенного набора параметров к соответствующей среде разработки.
В любой момент можно изменить набор параметров. Для этого в меню "Сервис" выберите команду "Импорт и экспорт параметров", а затем выберите вариант "Сбросить все параметры".
Выберите параметры среды для использования по умолчанию:
Веб-разработка
Веб-разработка (только код)
Настройки среды разработки Visual C#
Настройки среды разработки Visual F#
Обычные параметры разработки
Параметры разработки Visual Basic
Параметры разработки Visual C++
Параметры управления проектом
Описание:
Выберите одну из коллекций параметров из списка.
Запуск Visual Studio
Выход из Visual Studio


Выбор параметров среды, используемых по умолчанию
Microsoft Visual Studio 2010 Ultimate
Перед первым использованием приложения необходимо указать, какие языки в основном используются при разработке, например Visual Basic или Visual C#. Эти сведения используются для применения встроенного набора параметров к соответствующей среде разработки.
В любой момент можно изменить набор параметров. Для этого в меню "Сервис" выберите команду "Импорт и экспорт параметров", а затем выберите вариант "Сбросить все параметры".
Выберите параметры среды для использования по умолчанию:
Веб-разработка
Веб-разработка (только код)
Настройки среды разработки Visual C#
Настройки среды разработки Visual F#
Обычные параметры разработки
Параметры разработки Visual Basic
Параметры разработки Visual C++
Параметры управления проектом
Описание:
Полная оптимизация среды, позволяющая сразу начать построение приложений. В этот набор параметров входят настройки положения окон, настройки команд меню и сочетаний клавиш для разработки на Visual Basic.
Запуск Visual Studio
Выход из Visual Studio

Start Page

Toolbox

Microsoft® Visual Studio® 2010 Premium

Connect To Team Foundation Server

New Project...

Open Project...

Recent Projects

- WindowsApplication-stroka
- Ris_Form
- Ris_Figur
- Draw Shapes
- Zoom In
- Zoom In
- Draw Shapes
- Moving Icon
- Moving Icon

Close page after project load

Show page on startup

Get Started | Guidance and Resources | Latest News

Welcome | Windows | Web | Cloud | Office | SharePoint | Data

What's New in Visual Studio 2010

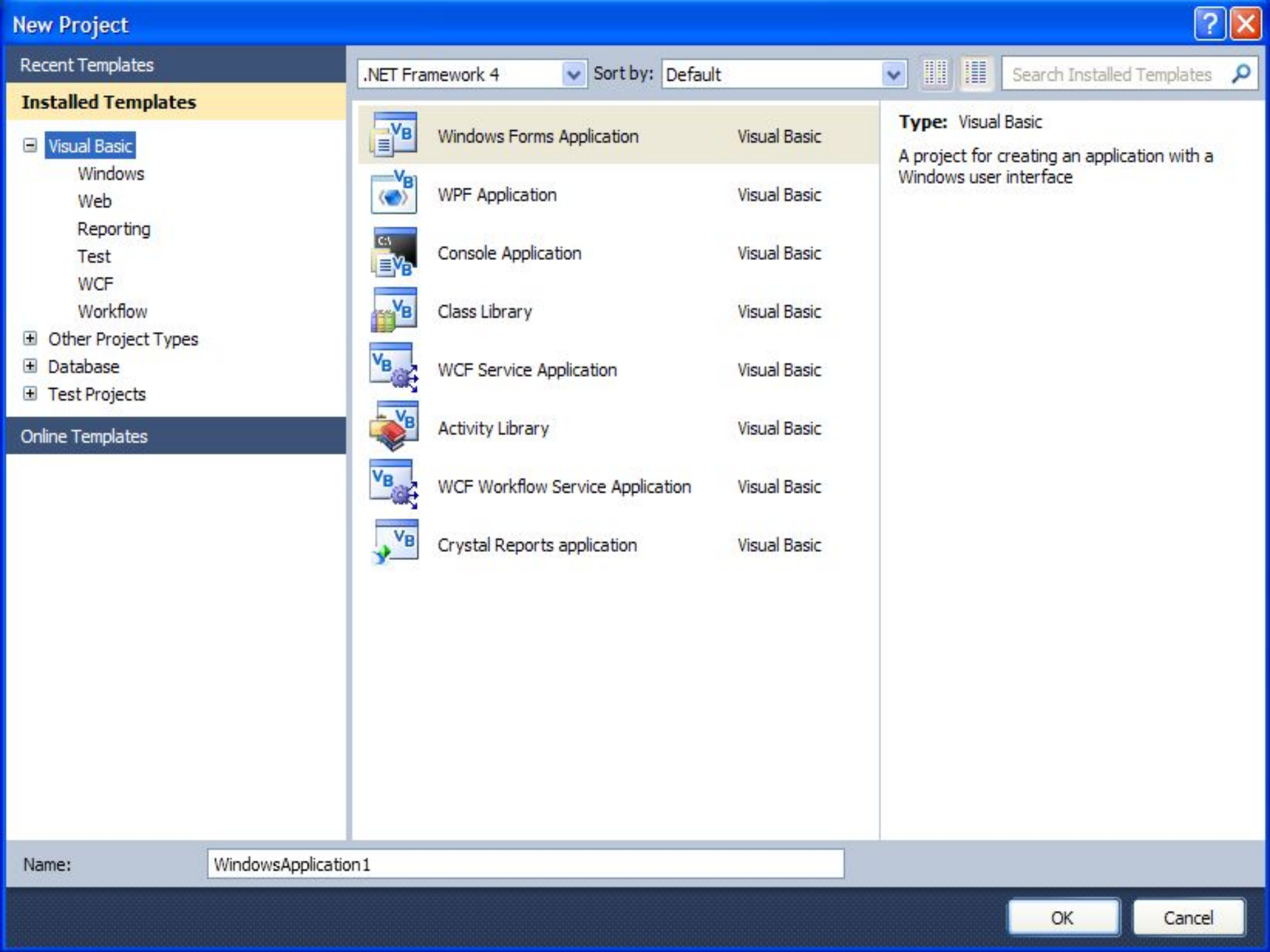
Learn about the new features included in this release.

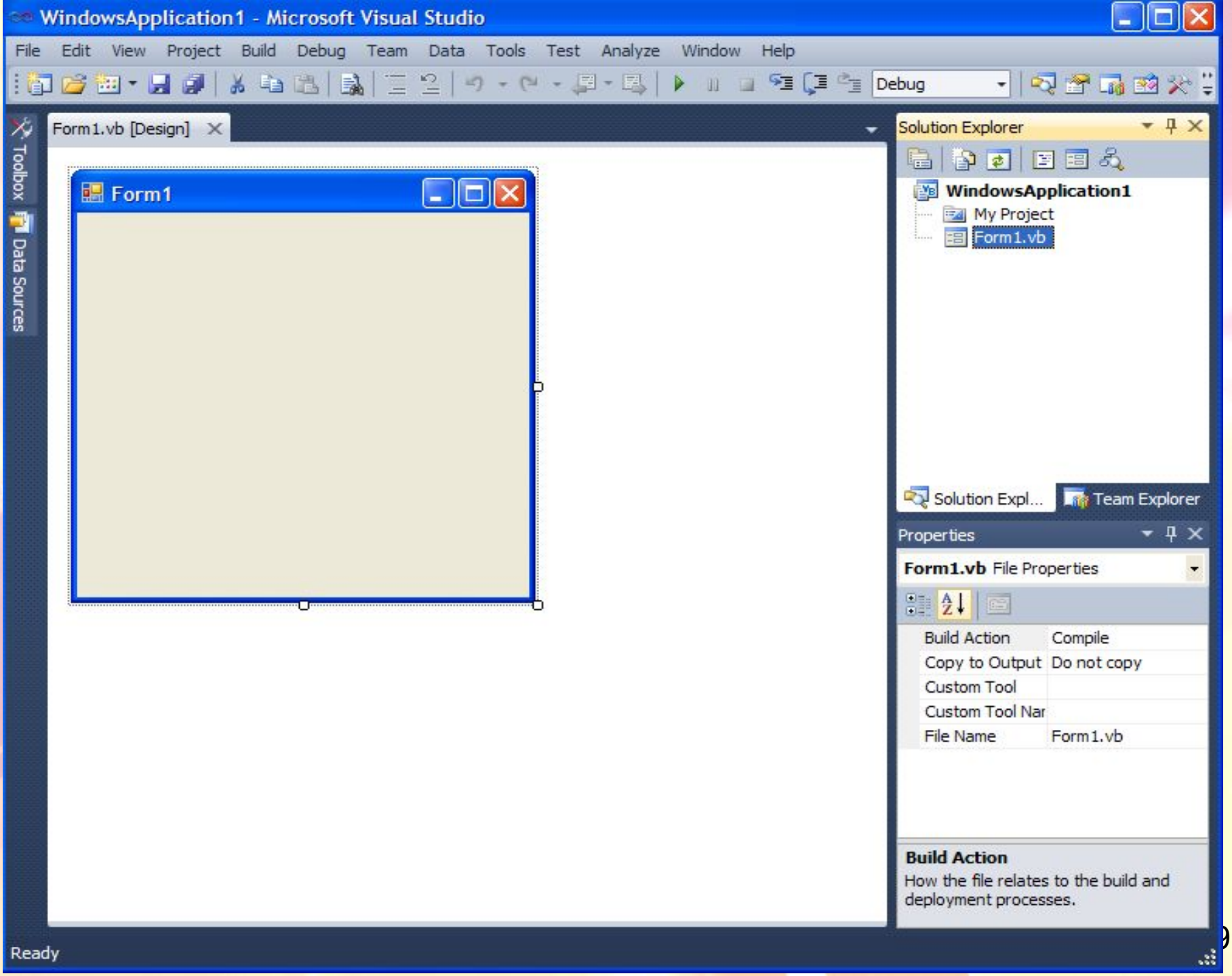
Visual Studio 2010 Overview
What's New in .NET Framework 4
What's New in Visual Basic
Customize the Visual Studio Start Page

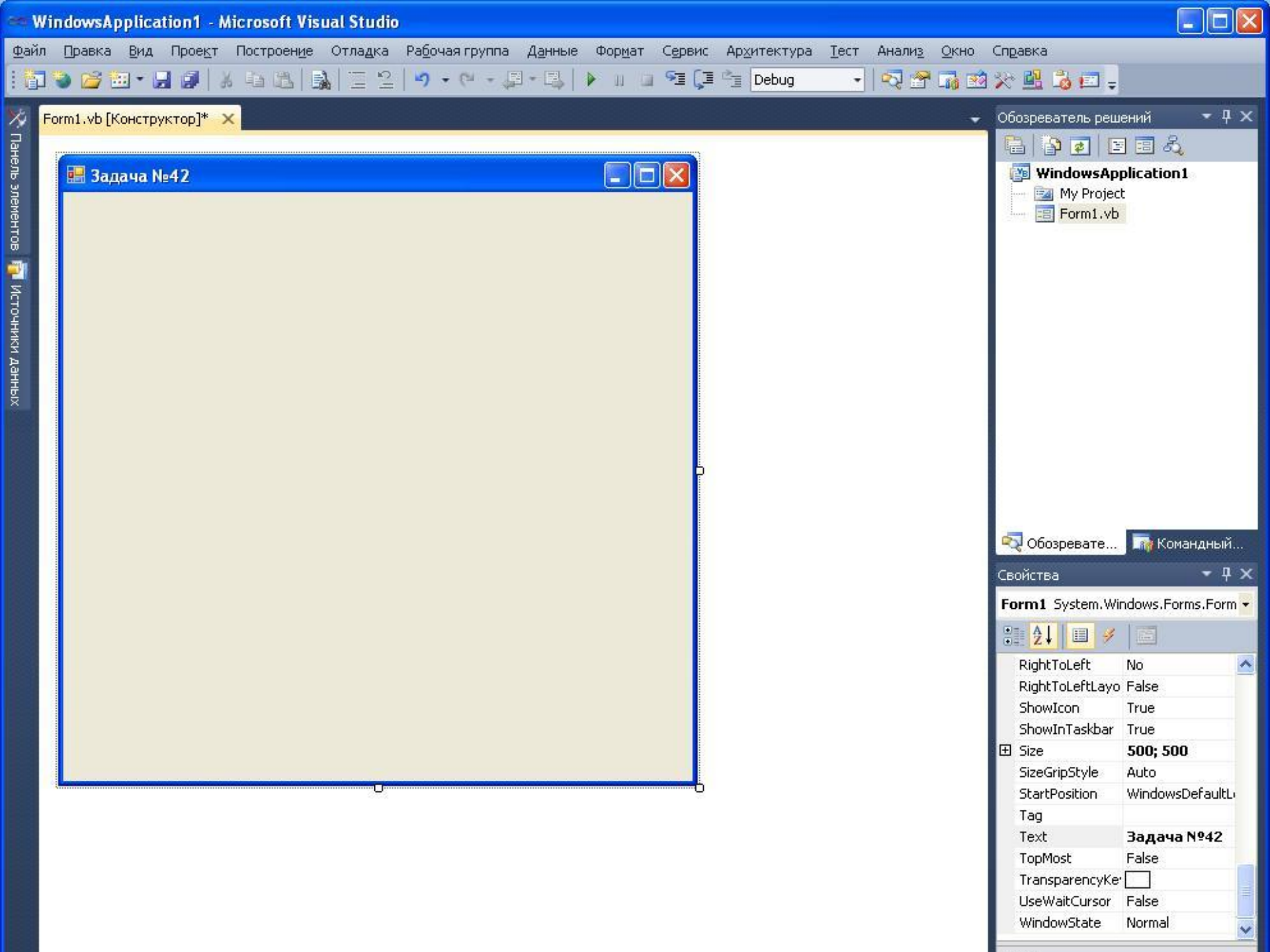
Creating Applications with Visual Studio

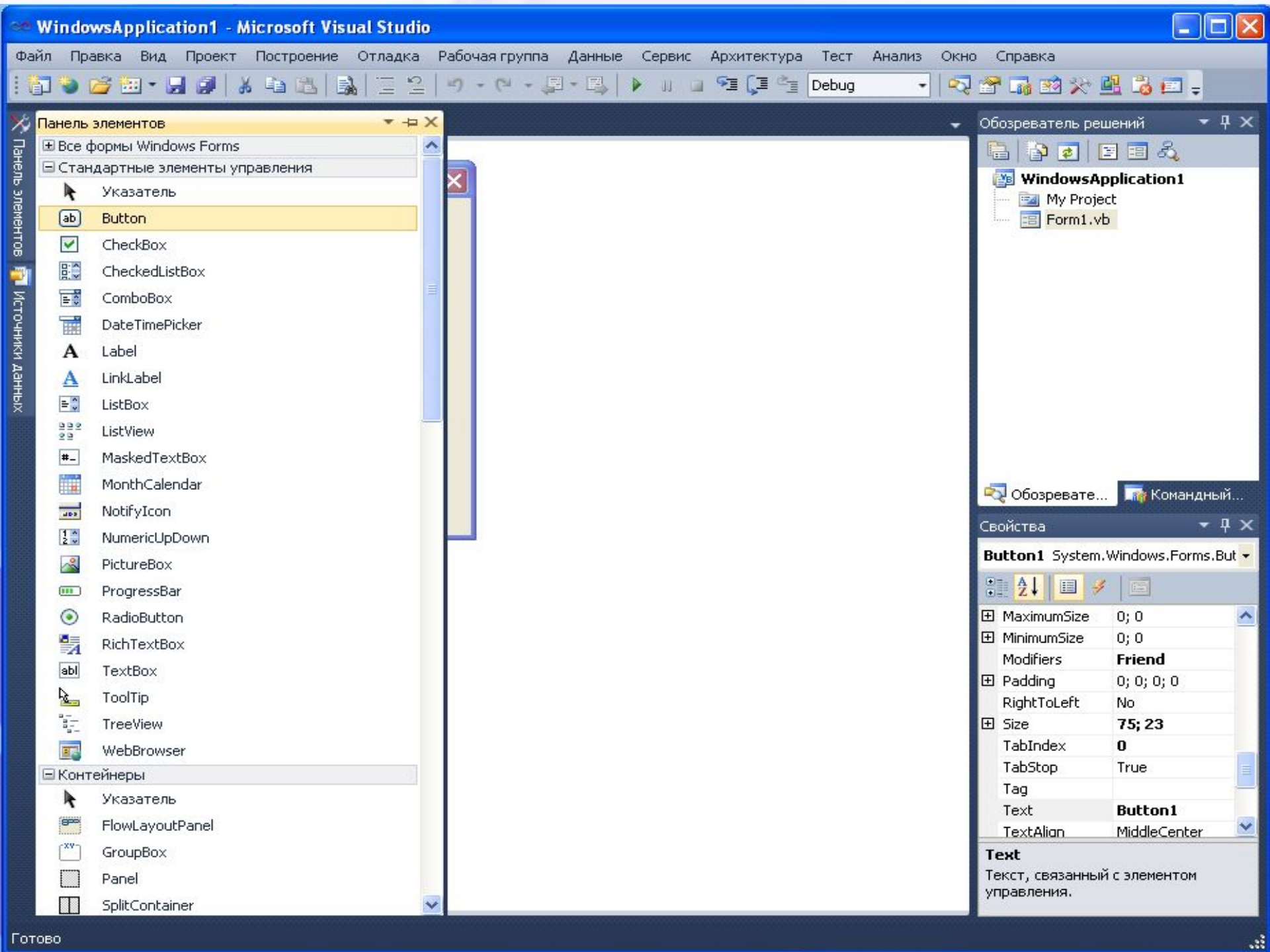
Extending Visual Studio

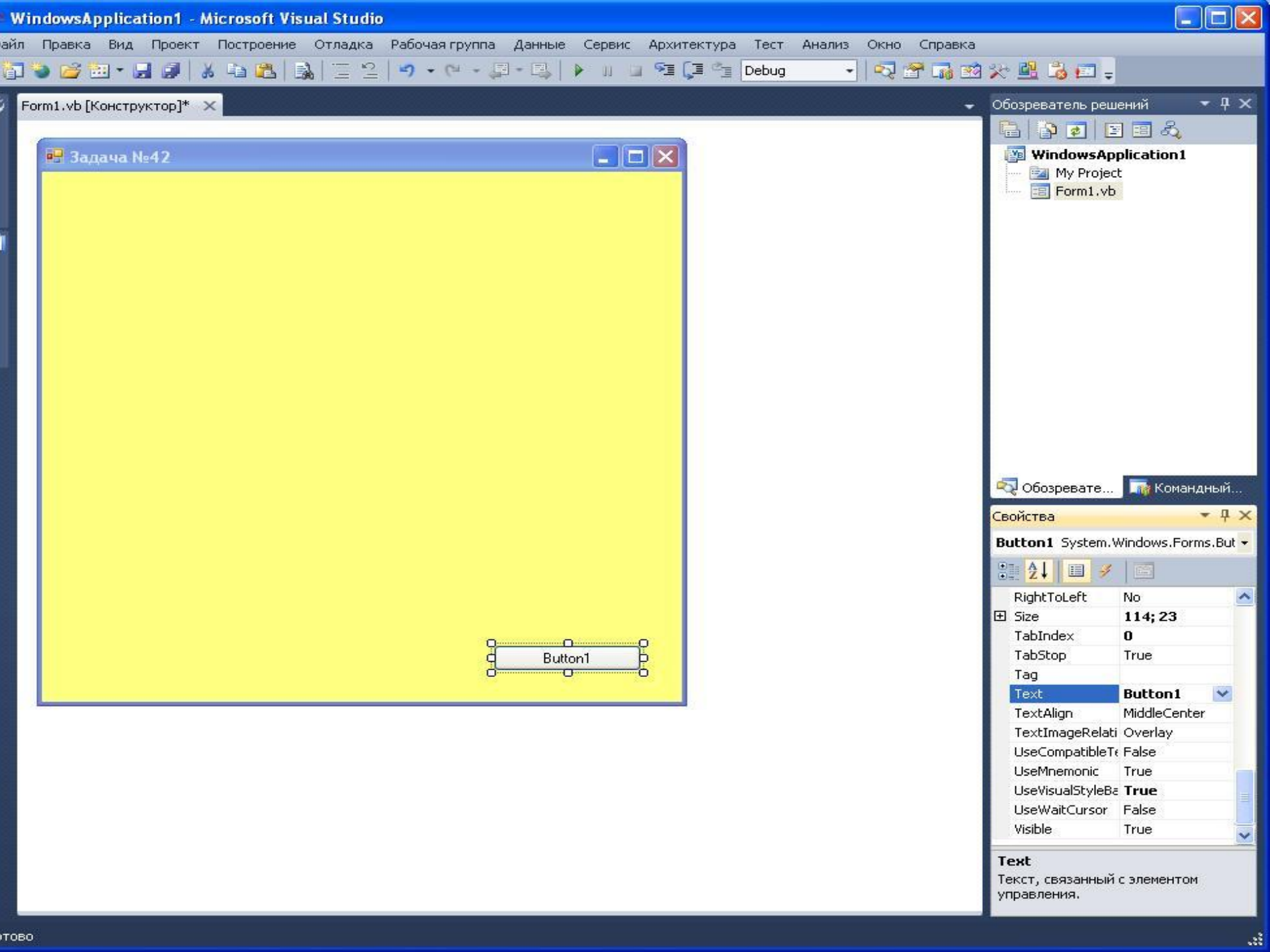
Community and Learning Resources

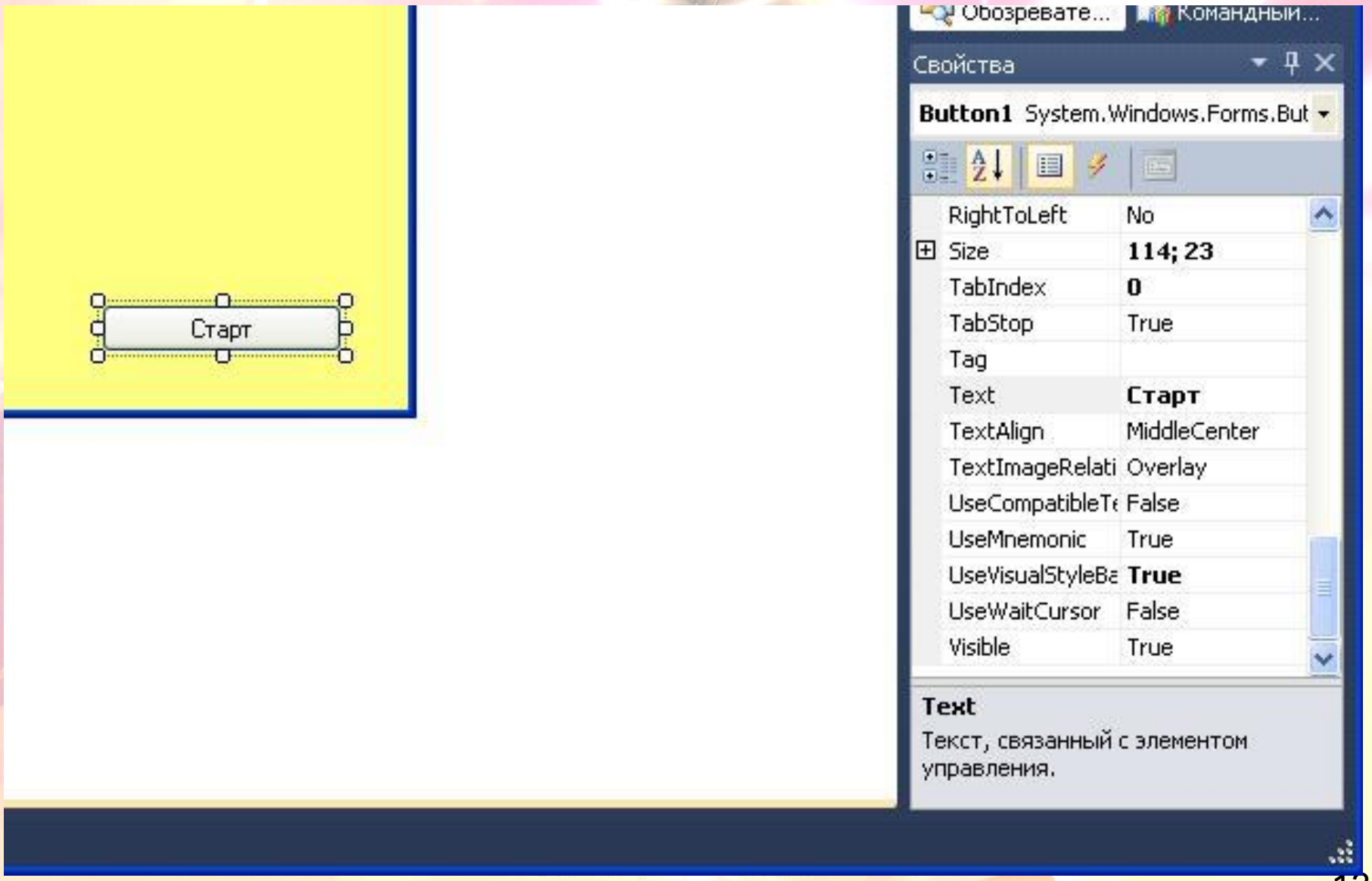

New Project
Recent Templates
Installed Templates
Visual Basic
Windows
Web
Reporting
Test
WCF
Workflow
Other Project Types
Database
Test Projects
Online Templates
.NET Framework 4
Sort by:
Default
Search Installed Templates
Windows Forms Application
Visual Basic
WPF Application
Visual Basic
Console Application
Visual Basic
Class Library
Visual Basic
WCF Service Application
Visual Basic
Activity Library
Visual Basic
WCF Workflow Service Application
Visual Basic
Crystal Reports application
Visual Basic
Type: Visual Basic
A project for creating an application with a Windows user interface
Name:
WindowsApplication1
OK
Cancel

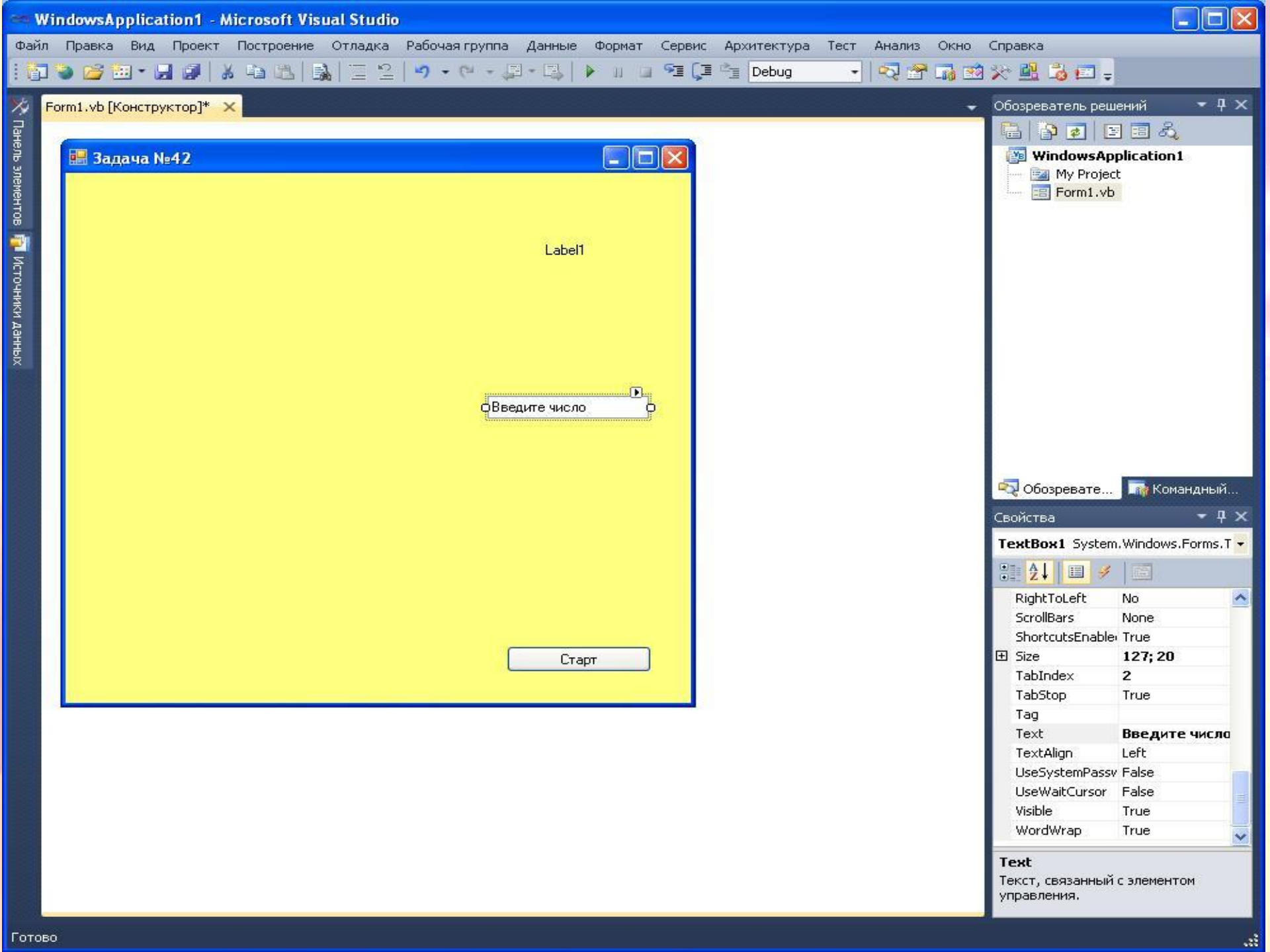

WindowsApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Test Analyze Window Help
Debug
Toolbox
Data Sources
Form1.vb [Design]
Form1
Solution Explorer
WindowsApplication1
My Project
Form1.vb
Solution Expl...
Team Explorer
Properties
Form1.vb File Properties
Build Action Compile
Copy to Output Do not copy
Custom Tool
Custom Tool Nar
File Name Form1.vb
Build Action
How the file relates to the build and deployment processes.
Ready

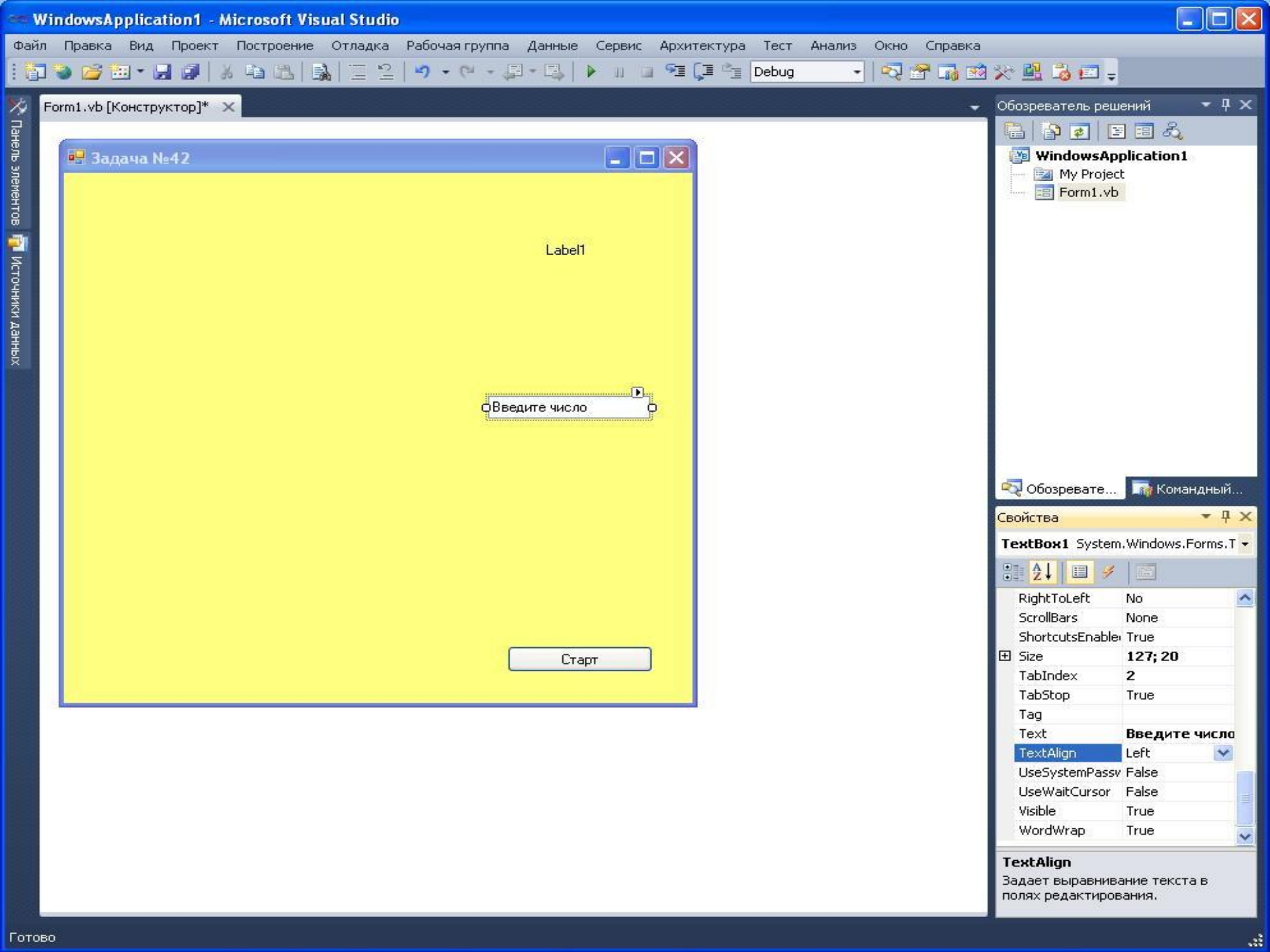

WindowsApplication1 - Microsoft Visual Studio
Файл Правка Вид Проект Построение Отладка Рабочая группа Данные Формат Сервис Архитектура Тест Анализ Окно Справка
Debug
Form1.vb [Конструктор]*
Задача №42
Панель элементов
Источники данных
Обозреватель решений
WindowsApplication1
My Project
Form1.vb
Обозревате...
Командный...
Свойства
Form1 System.Windows.Forms.Form
RightToLeft No
RightToLeftLayo False
ShowIcon True
ShowInTaskbar True
Size 500; 500
SizeGripStyle Auto
StartPosition WindowsDefaultL
Tag
Text Задача №42
TopMost False
TransparencyKe
UseWaitCursor False
WindowState Normal

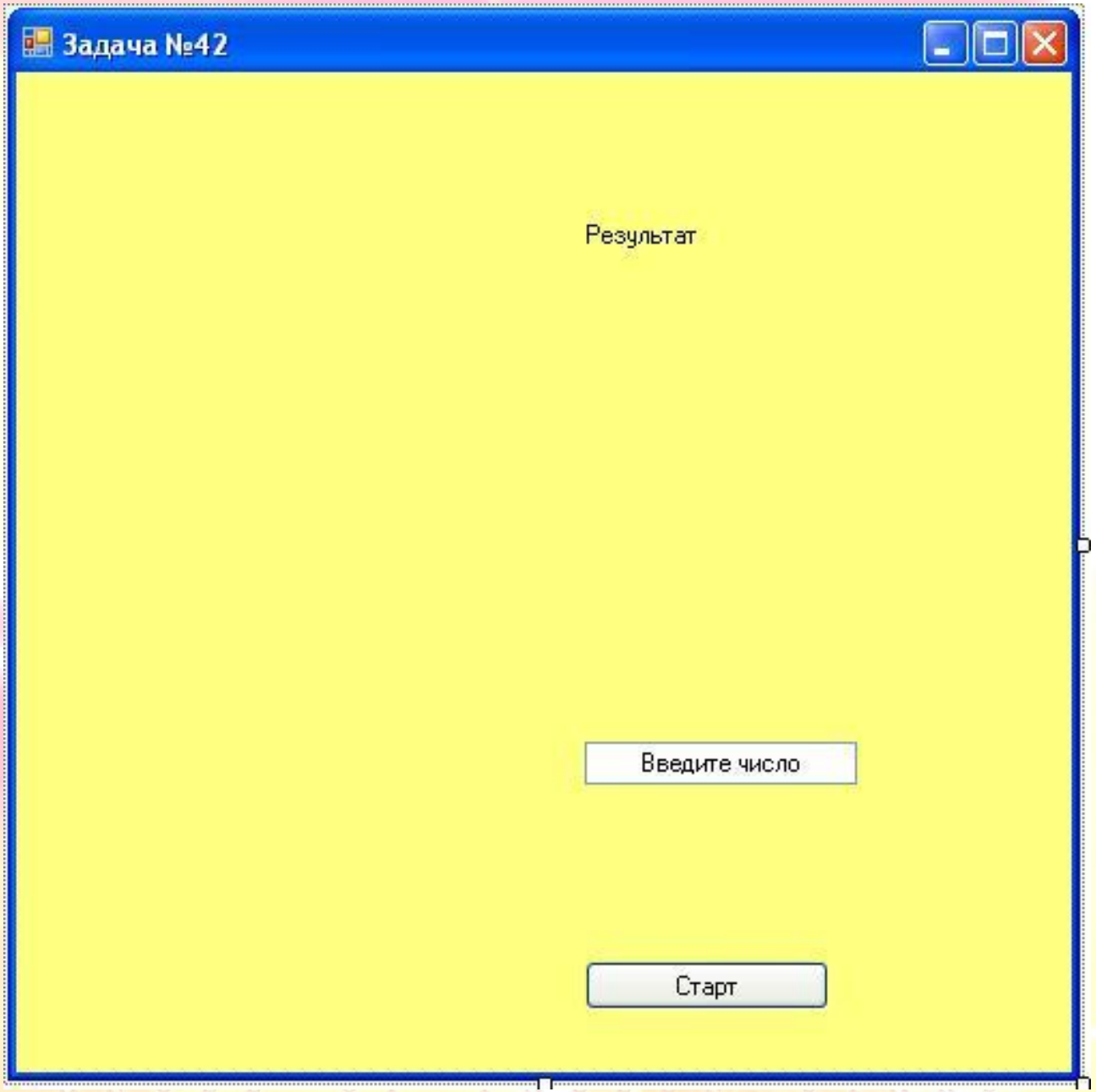

WindowsApplication1 - Microsoft Visual Studio
Файл Правка Вид Проект Построение Отладка Рабочая группа Данные Сервис Архитектура Тест Анализ Окно Справка
Debug
Панель элементов
Все формы Windows Forms
Стандартные элементы управления
Указатель
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
ToolTip
TreeView
WebBrowser
Контейнеры
Указатель
FlowLayoutPanel
GroupBox
Panel
SplitContainer
Обозреватель решений
WindowsApplication1
My Project
Form1.vb
Обозревате...
Командный...
Свойства
Button1 System.Windows.Forms.But
MaximumSize 0; 0
MinimumSize 0; 0
Modifiers Friend
Padding 0; 0; 0; 0
RightToLeft No
Size 75; 23
TabIndex 0
TabStop True
Tag
Text Button1
TextAlign MiddleCenter
Text
Текст, связанный с элементом управления.
Готово

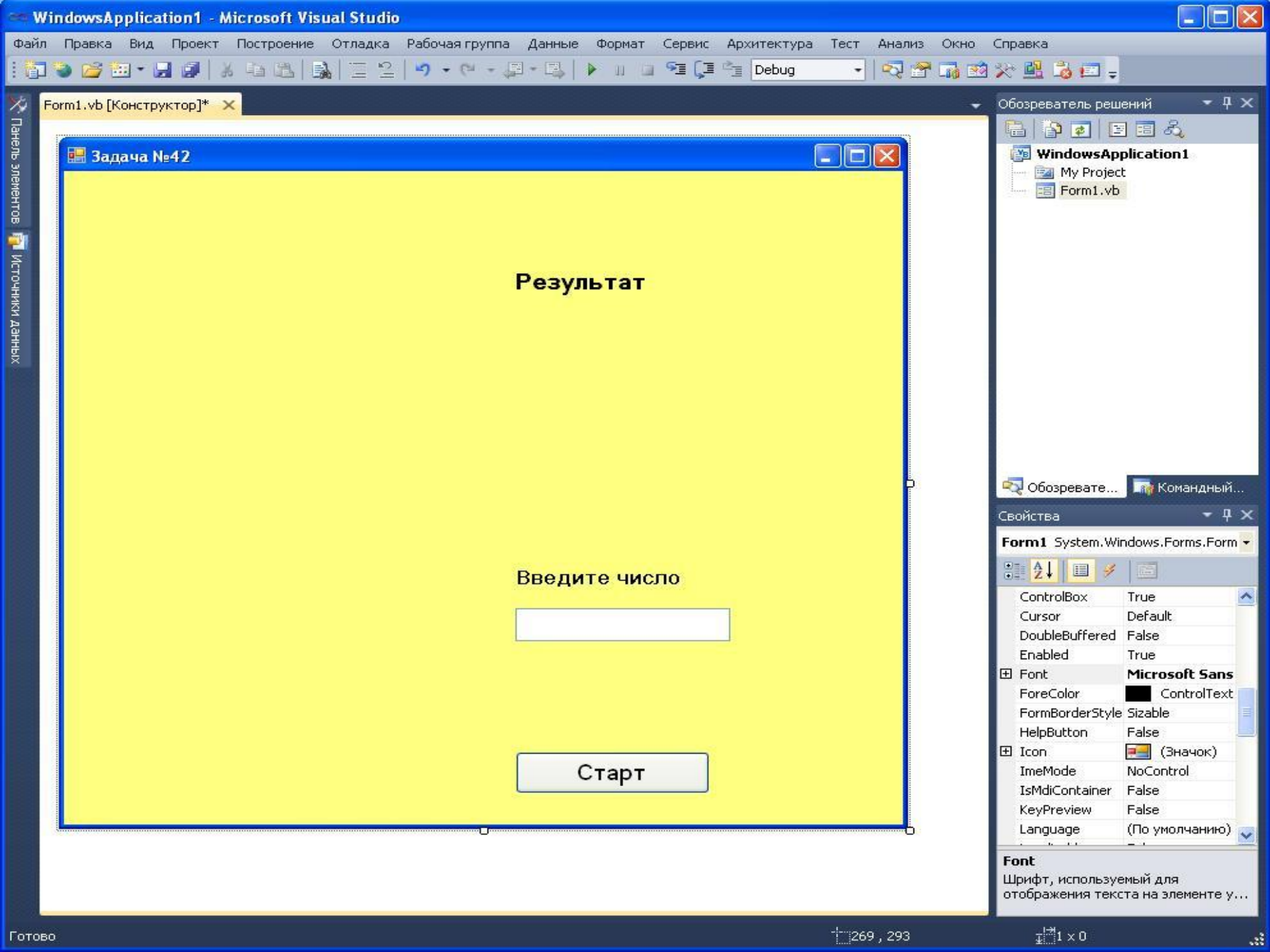

WindowsApplication1 - Microsoft Visual Studio
Файл Правка Вид Проект Построение Отладка Рабочая группа Данные Сервис Архитектура Тест Анализ Окно Справка
Debug
Form1.vb [Конструктор]*
Задача №42
Button1
Обозреватель решений
WindowsApplication1
My Project
Form1.vb
Обозревате...
Командный...
Свойства
Button1 System.Windows.Forms.But
RightToLeft No
Size 114; 23
TabIndex 0
TabStop True
Tag
Text Button1
TextAlign MiddleCenter
TextImageRelati Overlay
UseCompatibleTe False
UseMnemonic True
UseVisualStyleBa True
UseWaitCursor False
Visible True
Text
Текст, связанный с элементом управления.
Готово

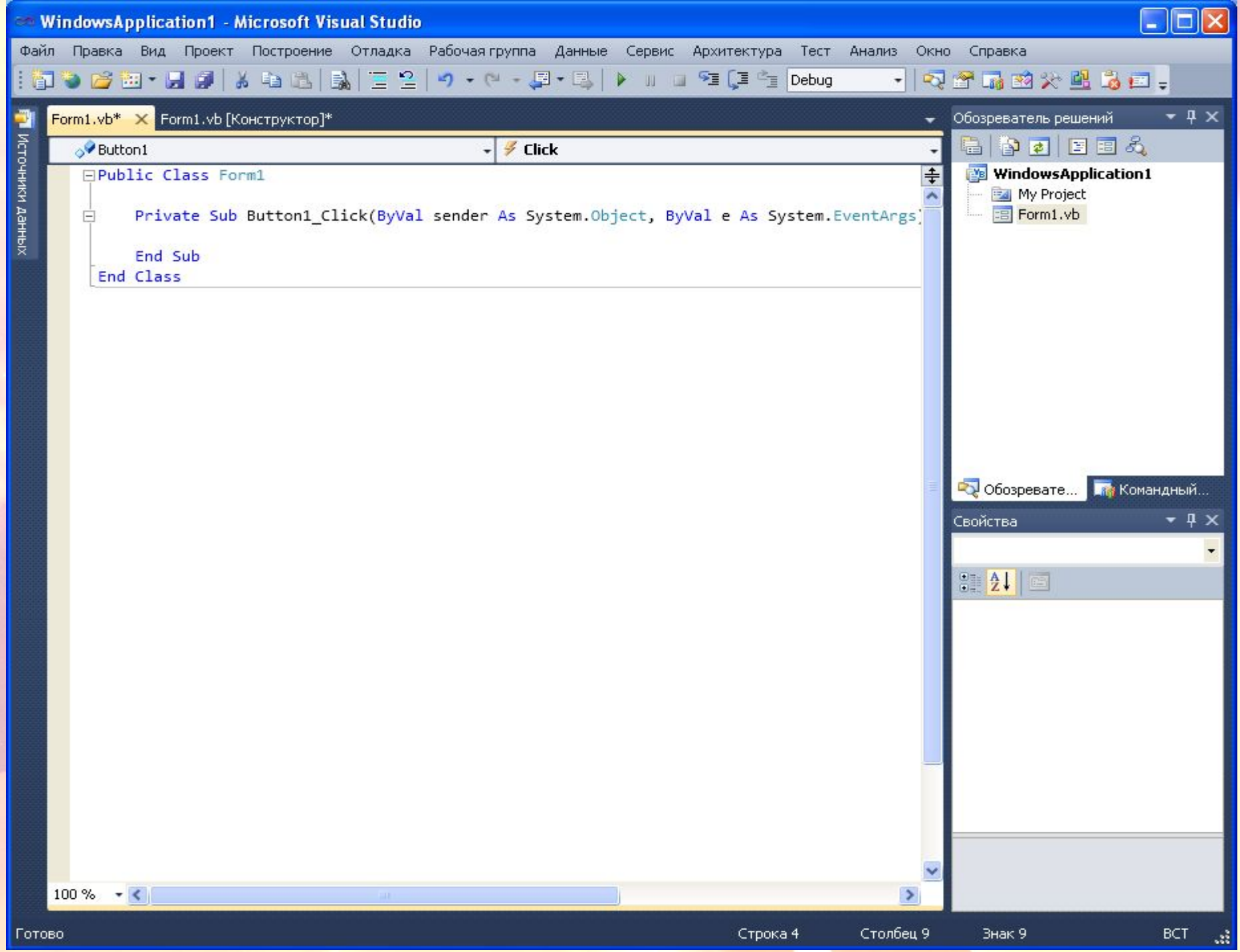

Старт
Обозревате...
Командный...
Свойства
Button1 System.Windows.Forms.But
RightToLeft No
Size 114; 23
TabIndex 0
TabStop True
Tag
Text Старт
TextAlign MiddleCenter
TextImageRelati Overlay
UseCompatibleTe False
UseMnemonic True
UseVisualStyleBa True
UseWaitCursor False
Visible True
Text
Текст, связанный с элементом управления.


WindowsApplication1 - Microsoft Visual Studio
Файл Правка Вид Проект Построение Отладка Рабочая группа Данные Формат Сервис Архитектура Тест Анализ Окно Справка
Debug
Form1.vb [Конструктор]*
Задача №42
Label1
Введите число
Старт
Обозреватель решений
WindowsApplication1
My Project
Form1.vb
Обозревате...
Командный...
Свойства
TextBox1 System.Windows.Forms.T
RightToLeft No
ScrollBars None
ShortcutsEnable True
Size 127; 20
TabIndex 2
TabStop True
Tag
Text Введите число
TextAlign Left
UseSystemPassv False
UseWaitCursor False
Visible True
WordWrap True
Text
Текст, связанный с элементом управления.
Готово


WindowsApplication1 - Microsoft Visual Studio
Файл Правка Вид Проект Построение Отладка Рабочая группа Данные Сервис Архитектура Тест Анализ Окно Справка
Debug
Form1.vb [Конструктор]*
Задача №42
Label1
Введите число
Старт
Обозреватель решений
WindowsApplication1
My Project
Form1.vb
Обозревате...
Командный...
Свойства
TextBox1 System.Windows.Forms.T
RightToLeft No
ScrollBars None
ShortcutsEnable True
Size 127; 20
TabIndex 2
TabStop True
Tag
Text Введите число
TextAlign Left
UseSystemPassv False
UseWaitCursor False
Visible True
WordWrap True
TextAlign
Задает выравнивание текста в полях редактирования.
Готово


Задача №42
Результат
Введите число
Старт


WindowsApplication1 - Microsoft Visual Studio
Файл Правка Вид Проект Построение Отладка Рабочая группа Данные Формат Сервис Архитектура Тест Анализ Окно Справка
Debug
Form1.vb [Конструктор]*
Задача №42
Результат
Введите число
Старт
Обозреватель решений
WindowsApplication1
My Project
Form1.vb
Обозревате...
Командный...
Свойства
Form1 System.Windows.Forms.Form
ControlBox True
Cursor Default
DoubleBuffered False
Enabled True
Font Microsoft Sans
ForeColor ControlText
FormBorderStyle Sizable
HelpButton False
Icon (Значок)
ImeMode NoControl
IsMdiContainer False
KeyPreview False
Language (По умолчанию)
Font
Шрифт, используемый для отображения текста на элементе у...
Готово
269 , 293
1 x 0

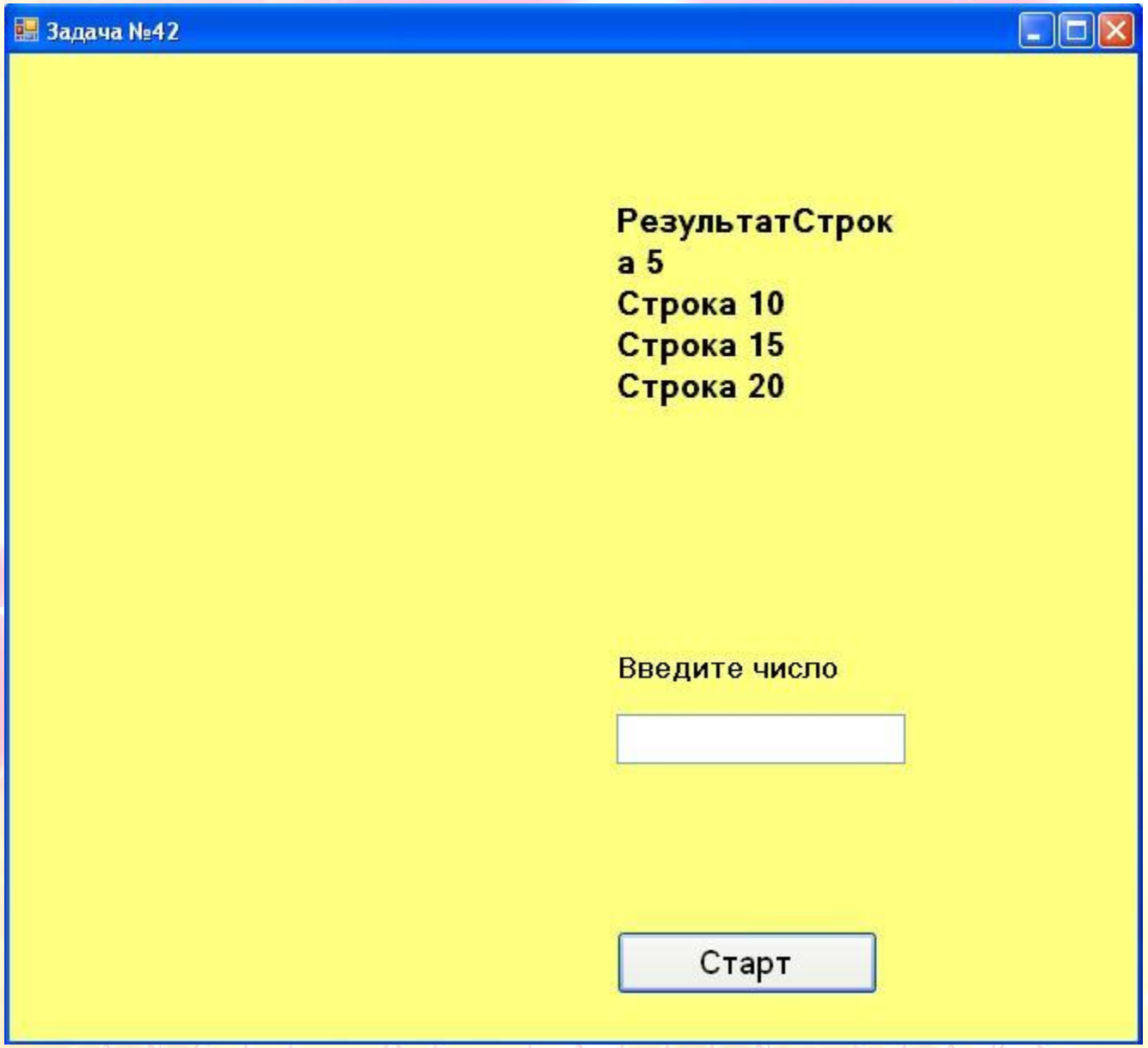

WindowsApplication1 - Microsoft Visual Studio
Файл Правка Вид Проект Построение Отладка Рабочая группа Данные Сервис Архитектура Тест Анализ Окно Справка
Debug
Источники данных
Form1.vb* Form1.vb [Конструктор]*
Button1
Click
Public Class Form1
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
End Sub
End Class
100 %
Обозреватель решений
WindowsApplication1
My Project
Form1.vb
Обозревате...
Командный...
Свойства
Готово
Строка 4
Столбец 9
Знак 9
ВСТ

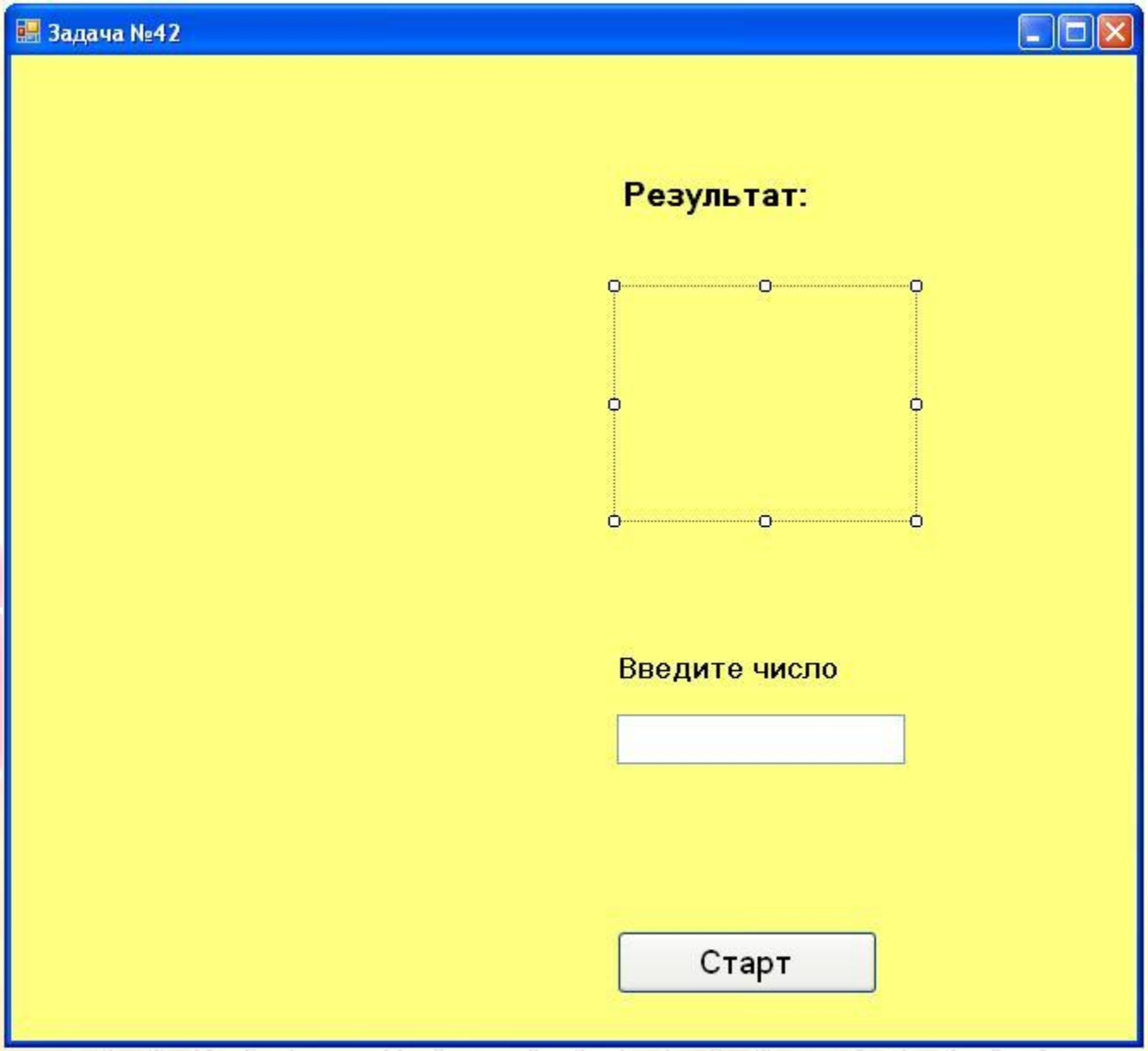
Кнопка, которая изменяет свойство текст объекта Label1

```
Private Sub Button1_Click(ByVal sender As
   System.Object, ByVal e As System.EventArgs) Handles
   Button1.Click

    Label1.Text ="Привет всем !"

End Sub
```

```
Public Class Form1
  Private Sub Button1_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs) Handles
    Button1.Click
    Dim i As Integer
    Dim Wrap As String
    Wrap = Chr(13) & Chr(10)
    For i = 5 To 25 Step 5
    Label1.Text = Label1.Text & "Строка " & i & Wrap
    Next i
  End Sub
End Class
```

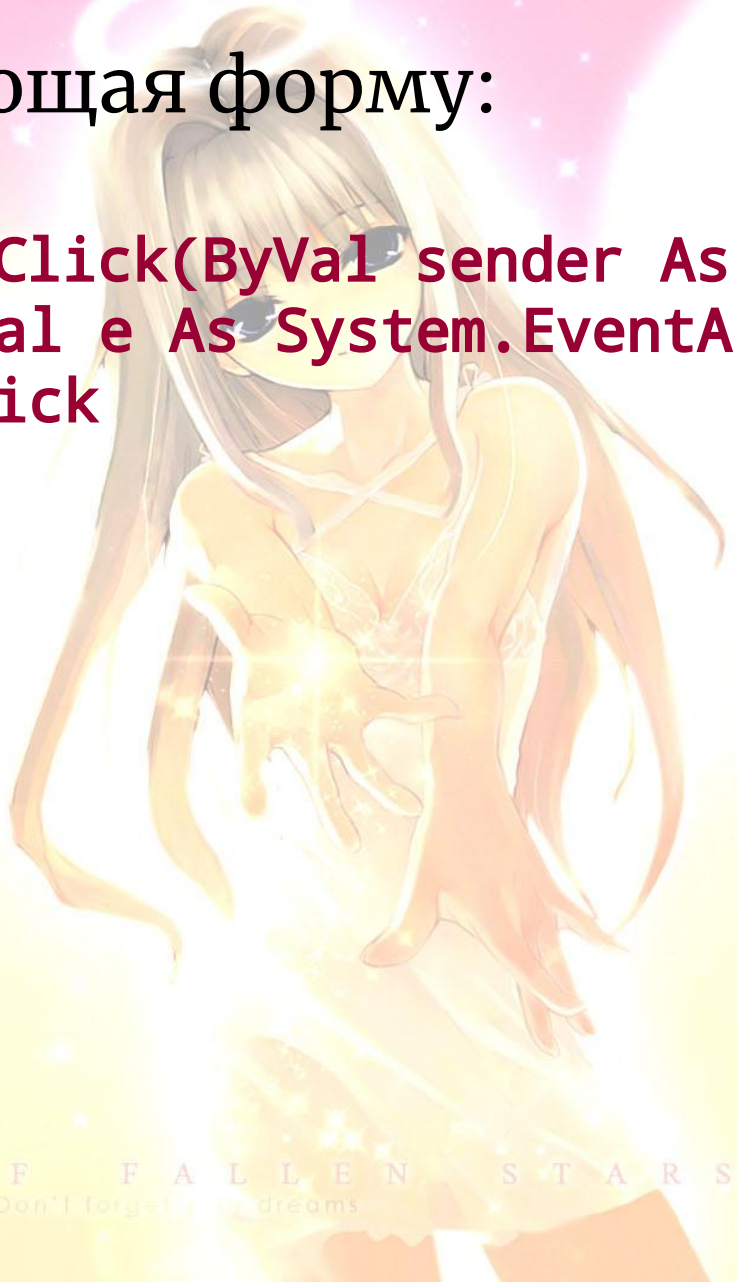
vbCrLf


Задача №42
РезультатСтрок
а 5
Строка 10
Строка 15
Строка 20
Введите число
Старт


Задача №42
Результат:
Введите число
Старт

Кнопка, закрывающая форму:

```
Private Sub Button1_Click(ByVal sender As
  System.Object, ByVal e As System.EventArgs)
  Handles Button1.Click

   Me.Close()

End Sub
```

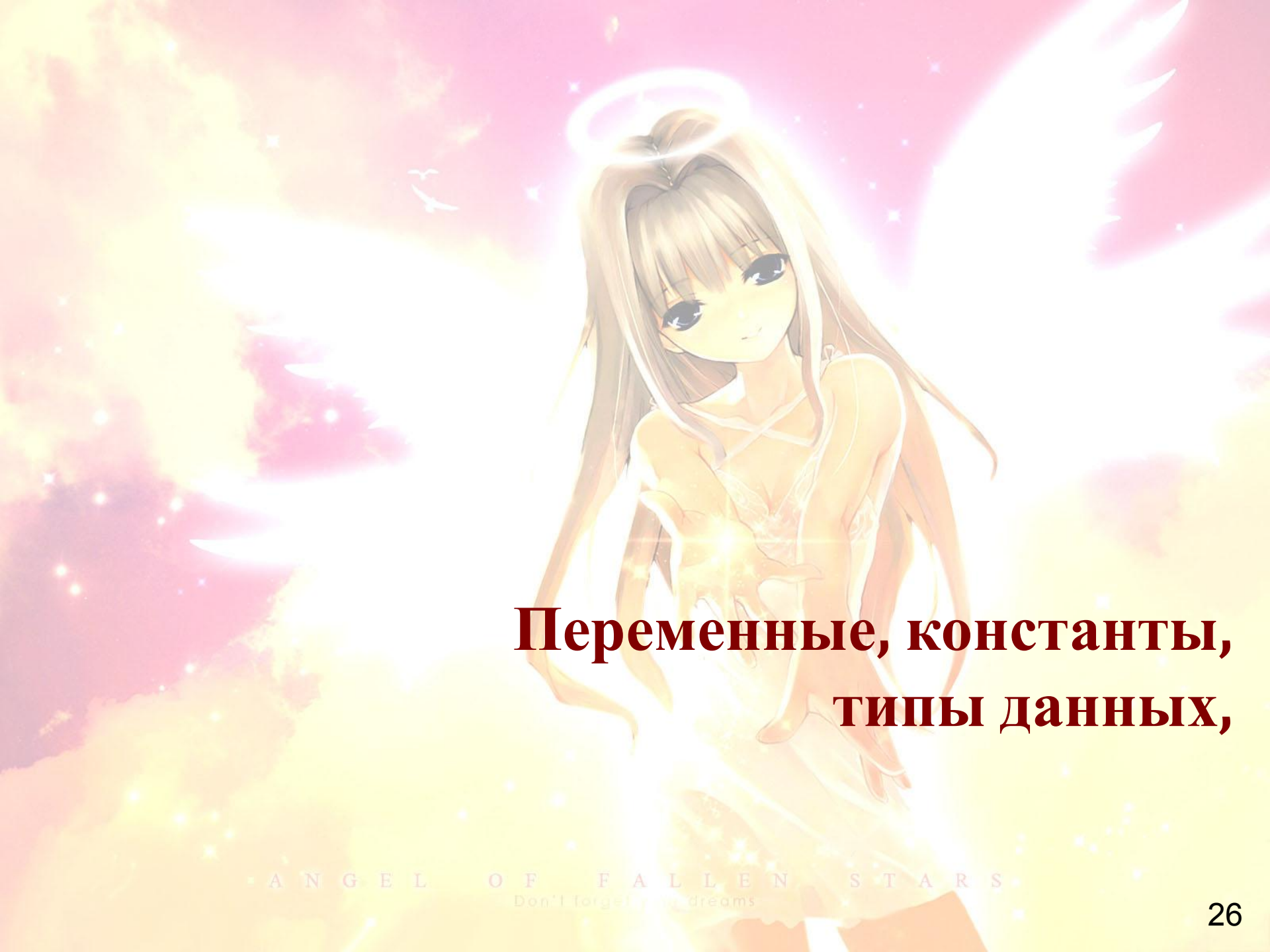


Языки программирования могут быть разделены на компилируемые и интерпретируемые

Программа на компилируемом языке при помощи специальной программы компилятора преобразуется (компилируется) в набор инструкций для данного типа процессора (машинный код) и далее записывается в исполняемый файл, который может быть запущен на выполнение как отдельная программа. Другими словами, компилятор переводит программу с языка высокого уровня на низкоуровневый язык, понятный процессору.

Если программа написана на интерпретируемом языке, то интерпретатор непосредственно выполняет (интерпретирует) ее текст без предварительного перевода. При

Во многих ранних версиях BASIC был *интерпретируемым* языком. Каждая строка перед выполнением интерпретировалась, что вызывало медленную обработку кода. В большинстве современных вариантов языка BASIC программа компилируется, и в результате выполнение программы значительно ускорилось, а перемещаемость программы улучшилась.



Переменные, константы, типы данных,

Переменные

Данные в компьютер поступают в различных видах. В пределах данного курса мы будем использовать числовую, символьную (текстовую, строковую, литерную) и логическую информацию.

Эти данные запоминаются в ячейках памяти компьютера. Ячейки могут состоять из 1, 2, 4, 8 и т.д. байт. 1 байт содержит 8 бит. 1 бит – это минимальная единица памяти компьютера.

Переменная – это именованная ячейка памяти, в которой хранятся данные. В процессе выполнения программы эти данные могут меняться.

Правила выбора имен переменных:

Может иметь длину до 255 символов.

Имя может начинаться только с буквы.

Нельзя использовать пробелы, точки и другие знаки препинания, специальные символы, кроме _.

Должно быть уникальным.

Не должно совпадать с ключевыми словами Visual Basic.

Переменные

Таким образом,

имя переменной – это комбинация букв, цифр, _. Начинается обязательно с буквы и имеет длину до 255 символов.

Регистр символов компилятором игнорируется. То есть, например, MyVariable и myvariable – это одна и та переменная, то есть одна и та же ячейка памяти.

Имена переменных нужно выбирать таким образом, чтобы легко было потом вспомнить для каких целей служила та или иная переменная.

Для задания (описания, объявления) переменной используется следующая синтаксическая конструкция:

Dim *имя_переменной* As *тип_данных* = начальное значение переменой.

Пример: **Dim iA As Integer = 10** – это описание целочисленной переменной с инициализацие начальным значением 10. Начальное значение можно не задавать. По умолчанию начальное значение переменой на Visual Basic равно нулю

Константы

Константа – это величина, которая не меняется в ходе выполнения программы.

Описание константы на языке Visual Basic:

Const *имя_константы* As *тип_данных* = значение *константы*

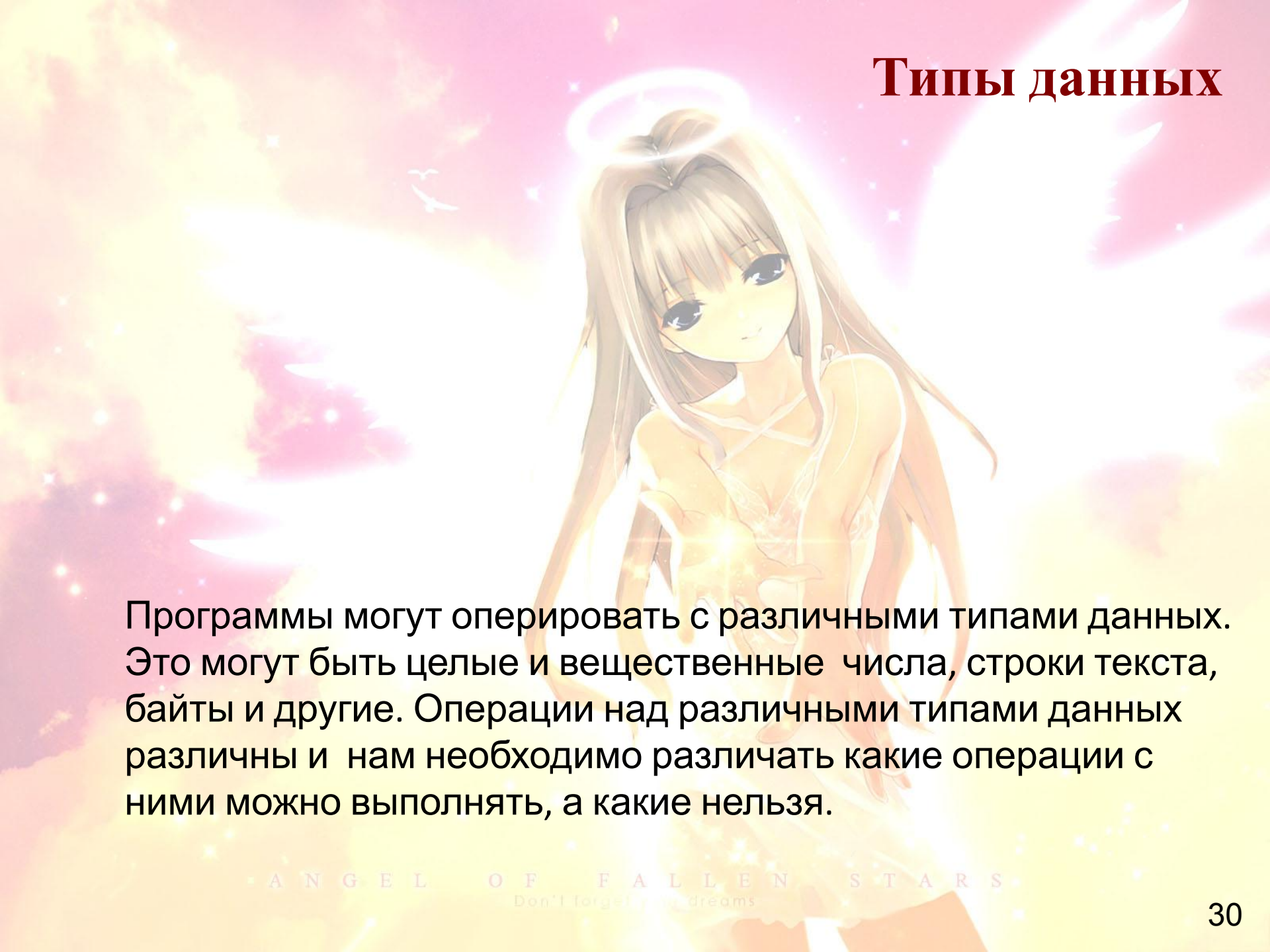
Пример:

```
Const PI As Single = 3.141593
```

И тогда в программе можно писать:

`L = 2 * PI * R` вместо `L = 2 * 3.141593 *R`

Типы данных



Программы могут оперировать с различными типами данных. Это могут быть целые и вещественные числа, строки текста, байты и другие. Операции над различными типами данных различны и нам необходимо различать какие операции с ними можно выполнять, а какие нельзя.

Типы данных	*Описание*	*Диапазон допустимых значений*
1. Short	Короткое целое. 2-х байтовое целое число со знаком.	-32 768 — +32 768
2. Integer	Целое. 4-х байтовое целое число со знаком.	-2 147 483 648 — +2 147 483 647
3. Long	Целое. 8-х байтовое целое число со знаком	-9 223 372 036 854 775 808 — +9 223 372 036 854 775 807
4. Single	4-х байтовое вещественное число	Для отрицательных значений: -3. 402 823 E+38 – 1. 401 298 E-45 Для положительных значений: +1. 401 298 E-45 – 3.402 823 E+38
5. Double	8-и байтовое вещественное число	Для отрицательных значений: -1. 79 769 313 486 232 E+308 – -4. 94 065 645 841 247 E-324 Для положительных значений: +4. 94 065 645 841 247 E-324 – +1. 79 769 313 486 232 E+308
6. Boolean	Логический	True и False
7. Char	Один символ формата Unicode, т.е. 2 байта на один символ.	0 … 32 768
8. String	Строка постоянной длины	От одного символа до 2-х миллионов символов приблизительно
9. Object	Объект	Любое значение любого типа

Логический тип данных

Переменные логического типа принимают только два значения - *true (истина)* и *false (ложь).*

true (истина) и *false (ложь)* – это логические константы.

Для них определены следующие логические операции:

Операция отрицания Not (НЕ).

Операция And (И)

Операция Or (ИЛИ)

Операция Xor (Исключающее ИЛИ)

C= Ф and G

Таблица истинности – это таблица значений, принимаемых переменными в результате данной логической операции.

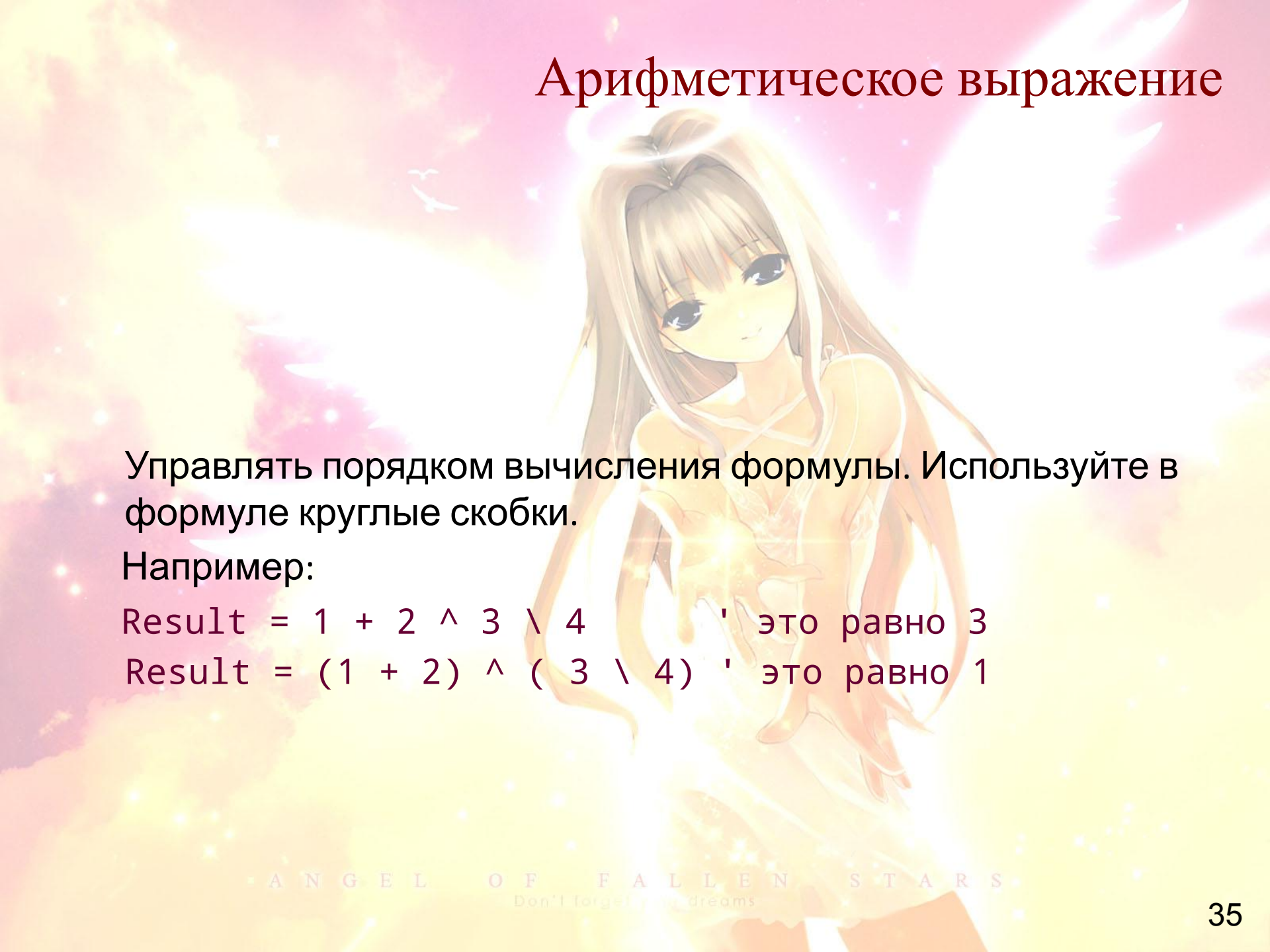
Знак операции	Операнд 1	Опреанд 2	Результат
Not	False		True
	True		False
And	False	False	False
	False	True	False
	True	False	False
	True	True	True
Or	False	False	False
	False	True	True
	True	False	True
	True	True	True
Xor	False	False	False
	False	True	True
	True	False	True
	True	True	False

Арифметическое выражение

Visual Basic содержит следующие операторы:

Оператор	Описание
+	Сложение
-	Вычитание
*	Умножение
/	Деление
\	Целочисленное деление (без остатка)
Mod	Остаток от деления
^	Возведение в степень
&	Объединение (конкатенация) строк

Арифметическое выражение



Управлять порядком вычисления формулы. Используйте в формуле круглые скобки.

Например:

```
Result = 1 + 2 ^ 3 \ 4        ' это равно 3
Result = (1 + 2) ^ ( 3 \ 4) ' это равно 1
```

Оператор присвоения

A=B

A=A+1

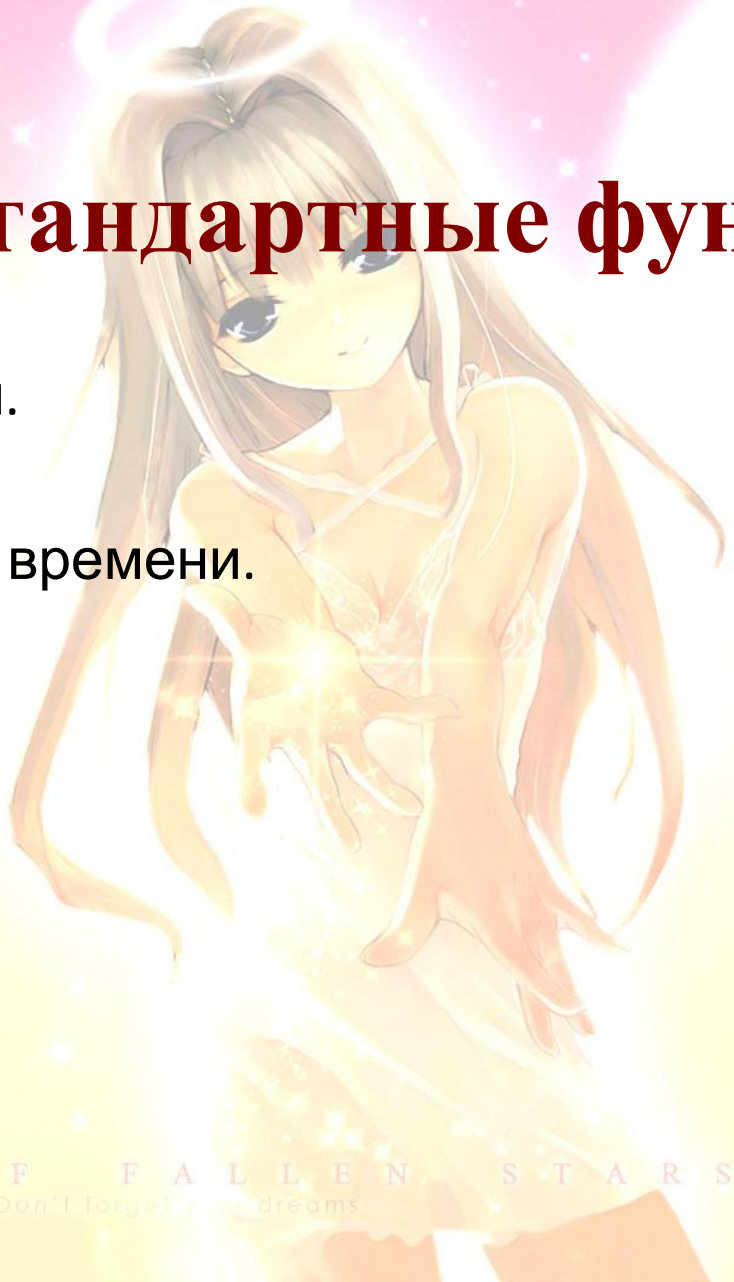
Стандартные функции

Математические функции.

Строковые функции.

Функции обработки дат и времени.

Функции…



Функция Math.Abs()

Функция Math.Abs() – возвращает модуль числа (абсолютное значение), заданного в качестве параметра функции.

Примеры использования:

```
Dim H As Single
H = Math.Abs(25)    '--- Результат будет 25
H = Math.Abs(-25)   '---Результат будет 25
Label1.text= H
```

Функция Math.Sqrt()

Функция Math.Sqrt() – предназначена для вычисления квадратного корня числа, заданного в качестве параметра функции.

Примеры использования:

```
Dim H As Single
H = Math.Sqrt(25)    '---Результат будет 5
Label1.text= H


Dim Hypotenuse As Single
X=3
Y=5
Hypotenuse = Sqrt(x ^ 2 + y ^ 2)
Label1.text= Hypotenuse
```

Функция Math.Sin()

Функция Math.Sin() – предназначена для вычисления синуса заданного угла (в радианах).

Пример использования:

```
Dim H As Single
H = Sin(1.57)    '---Результат будет 1
Label1.text= H
```

Функция Math.Cos()

Функция Math.Cos() – предназначена для вычисления косинуса заданного угла (в радианах).

Пример использования:

```
Dim H As Single
H = Cos (1.57)    '---Результат будет 0
Label1.text= H
```

Функция Math.Exp()

Функция Math.Exp() – предназначена для вычисления числа e = 2.71828182, возведенного в заданную в качестве аргумента степень.

$y=e^x$

Пример использования:

```
Msgbox Math.Exp(1) '---Результат будет 2.71828182.
```

Функция Math.Ceiling()

Функция Math.Ceiling() – возвращает наименьшее целое число, большее или равное заданному в качестве аргумента числу.

Ceiling [‘si:liŋ] – потолок, верхний предел.

Примеры использования:

```
Msgbox Math.Ceiling(2.4)  ‘---Результат будет 3.
Msgbox Math.Ceiling(-2.4) ‘---Результат будет -2.
```

Функция Math.Floor()

Функция Math.Floor() – возвращает наибольшее целое число, меньшее или равное заданному в качестве аргумента числу.

Floor [flo:] – нижний предел.

Примеры использования:

```
Msgbox Math.Floor (2.4)      '---Результат будет 2.
Msgbox Math.Floor(-2.4)      '---Результат будет -3
```

Функция Math.Round()

Функция Math.Round() – служит для округления числа, заданного в качестве аргумента, до ближайшего целого числа.

Round [raund] – круглый.

Примеры использования:

```
Msgbox Math.Round(2.4)      '---Результат будет 2.
Msgbox Math.Round(2.7)      '---Результат будет 3
```

Функция Math.Log()

Функция Math.Log() – применяется для вычисления натурального логарифма числа, заданного в качестве аргумента.

Пример использования:

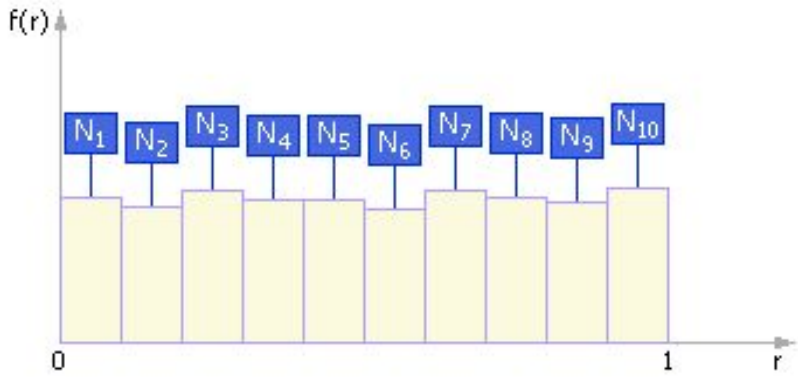
```
Msgbox Math.Log(2)     '---Результат будет
```

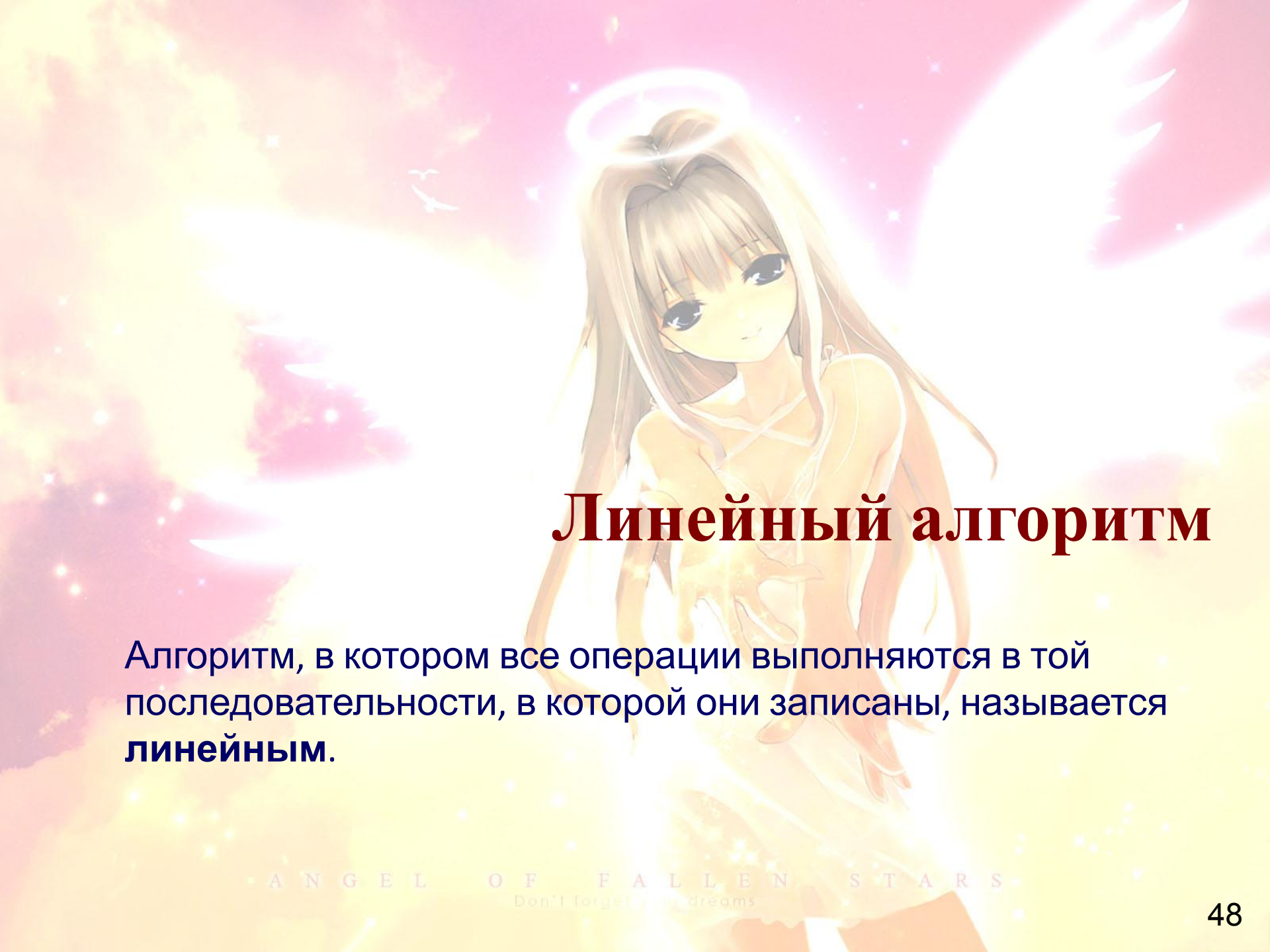
Функция rnd()

Функция rnd() – применяется для генерации случайных чисел. Она генерирует случайные вещественные числа в диапазоне 0.0 – 1.0

ГСЧ в качестве источника случайных чисел используют специальным образом составленные таблицы, содержащие проверенные некоррелированные, то есть никак не зависящие друг от друга, цифры.

```
Dim a As Integer
a = int(rnd()*100)
Label1.Text = a
```





Линейный алгоритм

Алгоритм, в котором все операции выполняются в той последовательности, в которой они записаны, называется **линейным**.

Линейный алгоритм.

Разберем решение задачи на линейный алгоритм на простом примере (пройдем все этапы решения задачи на компьютере):

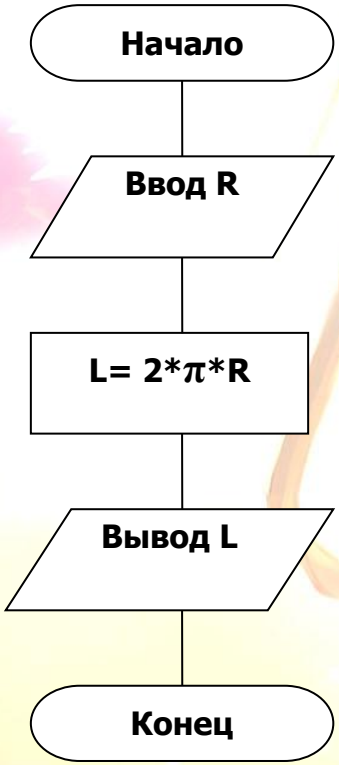
Постановка задачи. *Задан радиус окружности R. Найти длину окружности.*

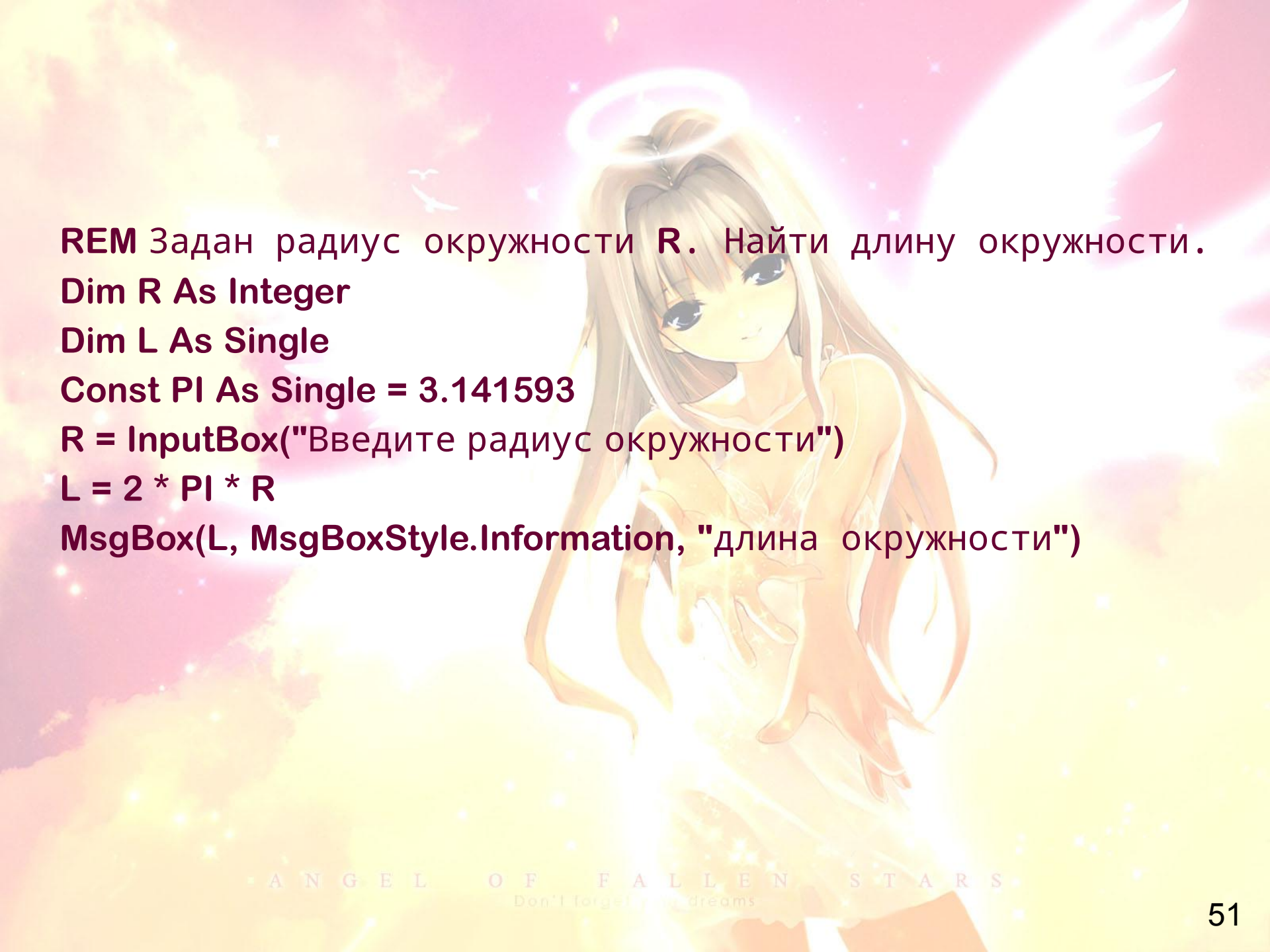
Разработка алгоритма решения задачи.

– Словесная запись алгоритма:
  - Задать численное значение радиуса окружности **R**.
  - Вычислить значение длины окружности по формуле

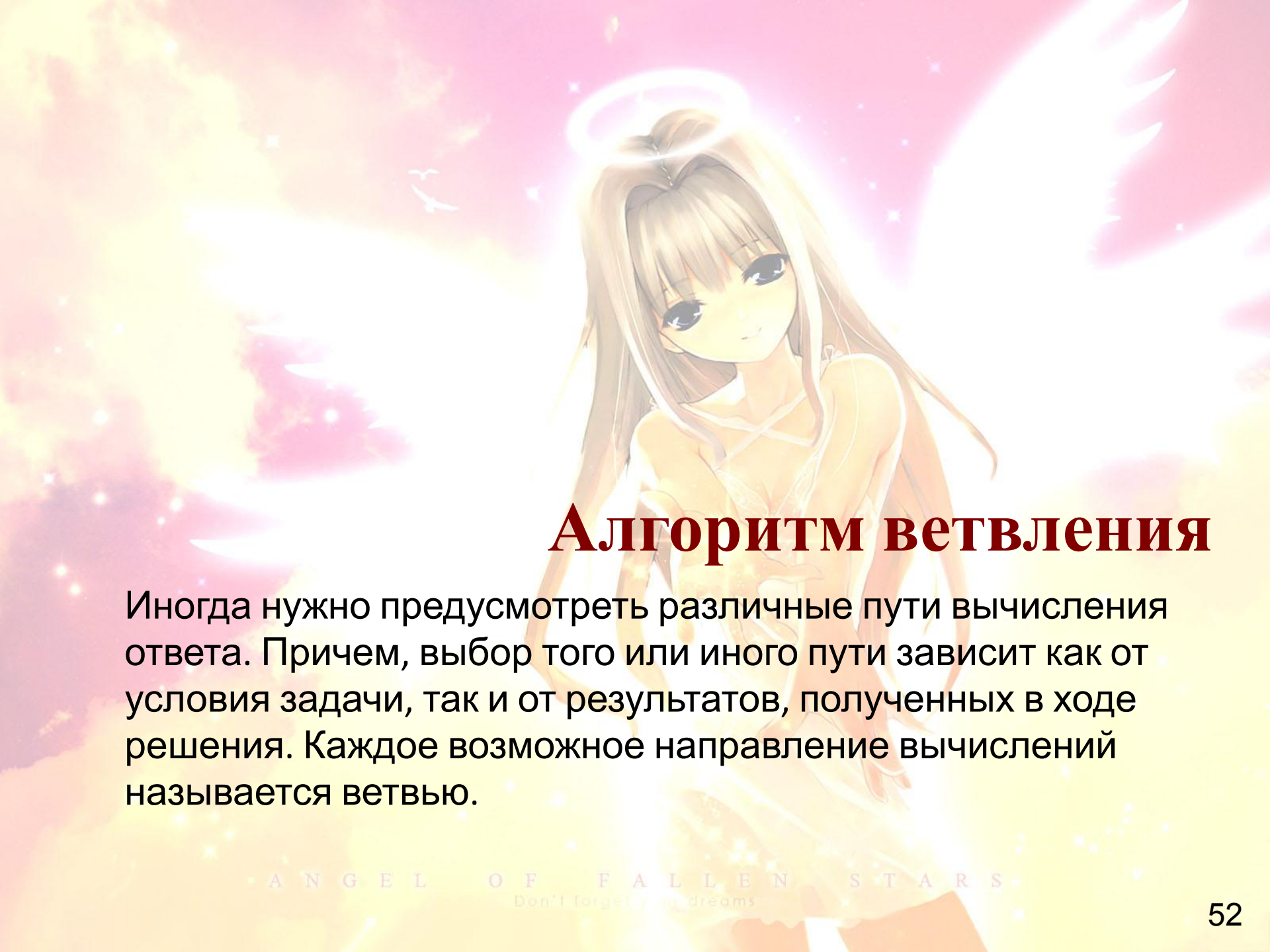
    L = 2*π*R
  - Напечатать значение L

Описание алгоритма с помощью блок-схемы:





```
REM Задан радиус окружности R. Найти длину окружности.
Dim R As Integer
Dim L As Single
Const PI As Single = 3.141593
R = InputBox("Введите радиус окружности")
L = 2 * PI * R
MsgBox(L, MsgBoxStyle.Information, "длина окружности")
```



Алгоритм ветвления

Иногда нужно предусмотреть различные пути вычисления ответа. Причем, выбор того или иного пути зависит как от условия задачи, так и от результатов, полученных в ходе решения. Каждое возможное направление вычислений называется ветвью.

Запись оператора ветвления на языке Visual Basic:

```
If условие Then оператор_1

If условие Then оператор_1 Else оператор_2

If условие Then
 опер_1
 опер_2
    EndIf
```

```
If условие Then
   опер_1
   опер_2
   …
Else
   опер_11
   опер_12
   …
EndIf
```

Операторы сравнения

Оператор сравнения	Значение
=	Равно
<>	Не равно
>	Больше, чем
<	Меньше, чем
>=	Больше или равно
<=	Меньше или равно

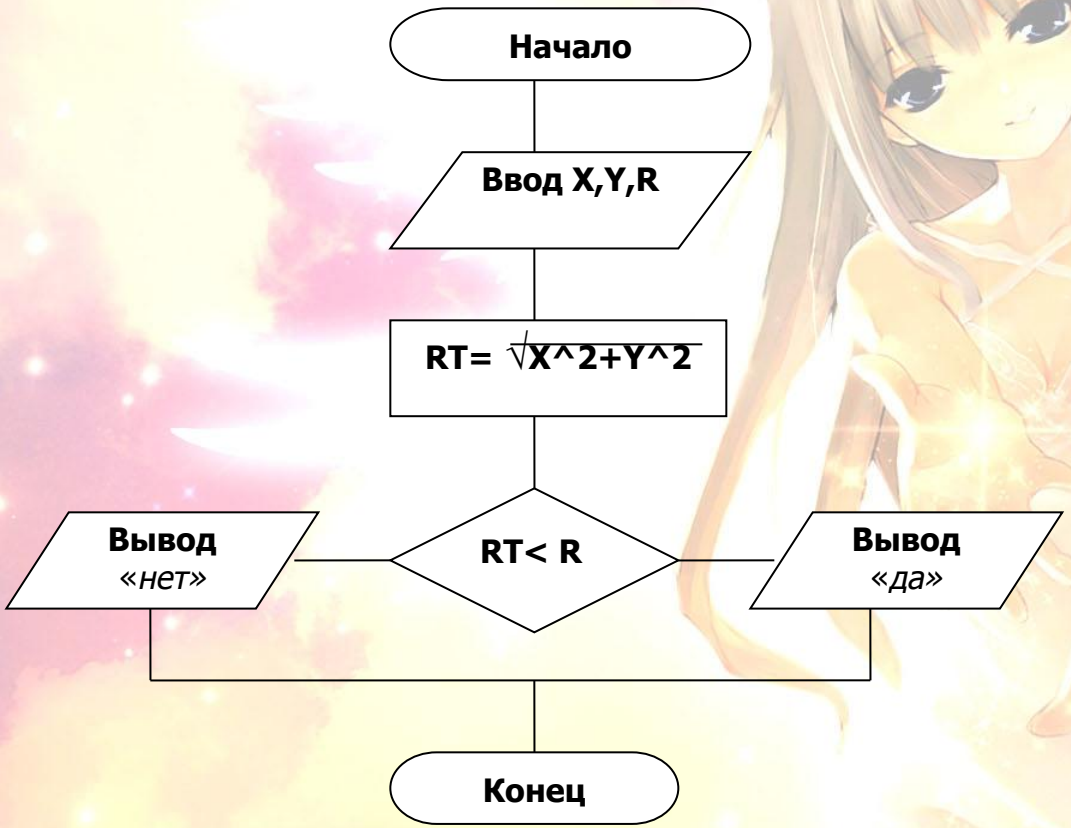
Задача на алгоритм ветвления

1. Постановка задачи. Определить попадает ли точка с координатами T(X,Y) в круг радиуса R. Центр окружности совпадает с началом системы координат. Программа должна ответить «*да*» или «*нет*».

2.Разработка алгоритма:

– Словесное описание алгоритма:
  - Задать координаты точки X,Y.
  - Задать радиус окружности R.
  - Вычислить расстояние от центра окружности до точки RT по формуле **RT= √X^2+Y^2**.
  - Если RT < R, то напечатать, что «*да*», иначе напечатать, что «*нет*».

Задача на алгоритм ветвления



Задача на алгоритм ветвления

3. Ввод текста программы в компьютер и ее отладка

```
REM Определить попадает ли точка с координатами T(X,Y)
в круг радиуса R.
REM Центр окружности совпадает с началом системы
координат.
REM Программа должна ответить «да» или «нет».
      Dim X, Y, R As Integer
      Dim RT As Single
      X = InputBox("Введите X-координату точки")
      Y = InputBox("Введите Y-координату точки")
      R = InputBox("Введите радиус окружности")
      RT = Math.Sqrt(X ^ 2 + Y ^ 2)
      If RT < R Then MsgBox("Да") Else MsgBox("Нет")
```

4. Тестирование программы.

5. Использование программы.

Оператор выбора Select Case

Select Case переменная

Case значение1 операторы программы, исполняемые, если переменная содержит значение 1

Case значение 2 операторы программы, исполняемые, если переменная содержит значение 2

Case значение 3 операторы программы, исполняемые, если переменная содержит значение 3

…

Case Else операторы программы, исполняемые, если совпадения не найдено"

End Select

Вместо слова *переменная* можно использовать переменную, арифметическое выражение, логическое выражение, свойство объекта и т.д.

```
REM Определить статус ученика в зависимости от
   полученной отметки
Dim a As Integer
a = InputBox("Введите оценку в виде числа")
Select Case a
    Case 1, 2
        MsgBox("Двоечник", MsgBoxStyle.Information)
    Case 3
        MsgBox("Троечник", MsgBoxStyle.Information)
    Case 4
        MsgBox("Хорошист", MsgBoxStyle.Information)
    Case 5
        MsgBox("Отличник", MsgBoxStyle.Information)
    Case Else
         MsgBox("Ошибка", MsgBoxStyle.Information)
End Select
```

```
Dim Age As Integer
Age = 18
Select Case Age
Case 16 MsgBox("Теперь вы можете водить машину!")
Case 18 MsgBox("Теперь вы можете участвовать в выборах!")
Case 21 MsgBox("Теперь вы можете пить вино. ")
Case 65 MsgBox("Теперь вы можете жить в свое удовольствие!")
End Select
```

```
Dim Age As Integer
Age = 25
Select Case Age
Case 16
Label1.Text = "Теперь вы можете водить машину!"
Case 18 Label1.Text = "Теперь вы можете участвовать
    в выборах!"
Case 21 Label1.Text = "Теперь вы можете пить вино."
Case 65 Label1.Text = "Теперь вы можете жить в свое
    удовольствие!"
Case Else Label1.Text = "Вы в полном расцвете сил!"
End Select
```

```
Select Case Age
Case Is < 13
Label1.Text = "Радуйся своему детству!"
Case 13 To 19
Label1.Text = "Радуйся своей молодости!"
Case 21 Label1.Text = "Пей вино с друзьями."
Case Is > 100
Label1.Text = "Прекрасно выглядишь!"
Case Else
Label1.Text = "Вы в полном расцвете сил."
End Select
```

Операторы сравнения позволяют включить в структуру `Select Case` диапазон значений. Для этого служат такие операторы сравнения Visual Basic, как

`=, <>, >, <, >= ,<=.`

Чтобы использовать операторы сравнения, вы должны включить в выражение ключевое слово `Is` или ключевое слово `To`, определяющие производимое сравнение. Ключевое слово `Is` говорит компилятору о том, что требуется сравнить переменную с выражением, указанным после ключевого слова `Is`. Ключевое слово `To` определяет диапазон значений.

Циклический алгоритм

Многократное повторение одних и тех же операций называется циклом.

Набор команд (операций), которые повторяются многократно, составляют тело цикла.

Циклический алгоритм

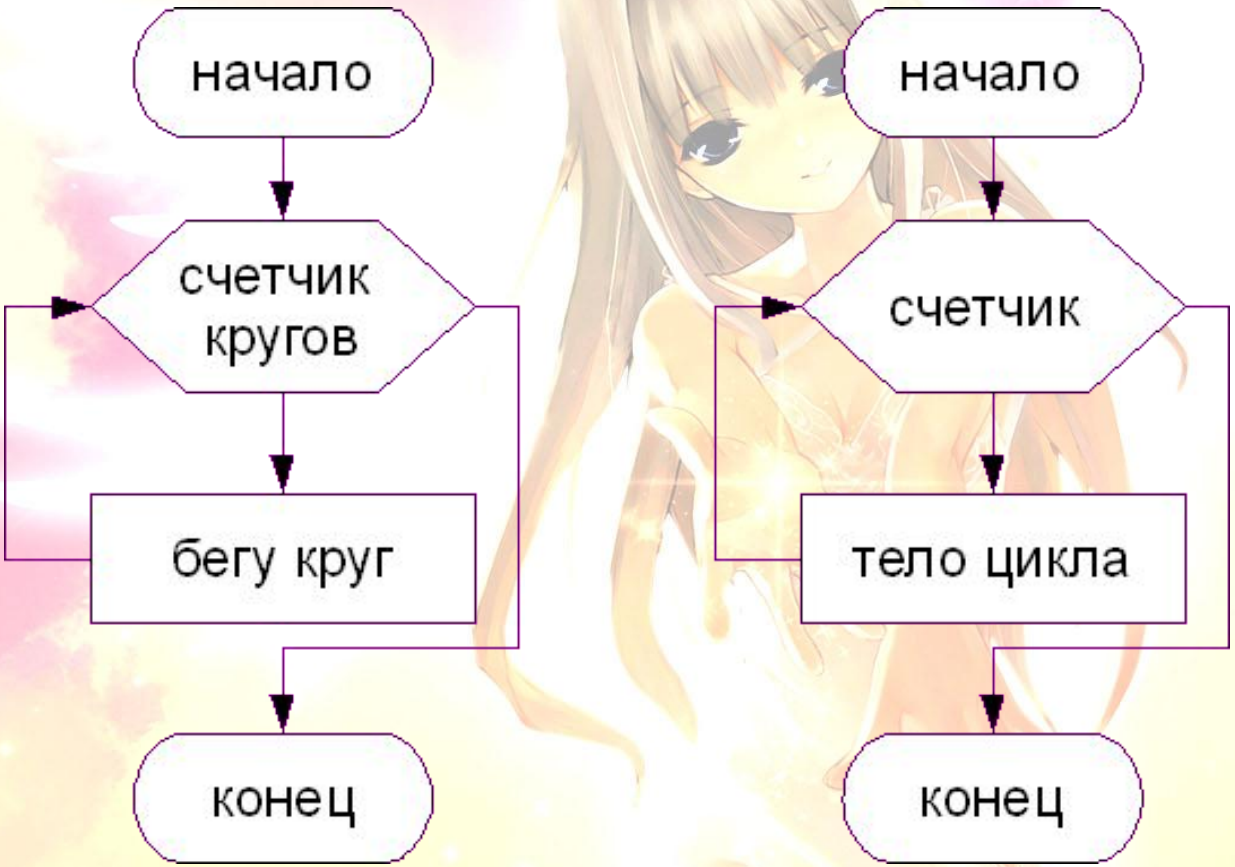
Постановка задачи.

Посчитать сколько калорий потратит бегун на стадионе за 15 кругов, если на первом круге он тратит 20ккал, а на каждом следующем на 5% больше.

Словесная запись алгоритма:

1) задать начальное условие.
2) задать приращение и способ его добавления
3) задать цикл и условие выхода из него.

Циклический алгоритм



Оператор цикла For…Next

С помощью цикла For…Next вы можете выполнять группу операторов программы в процедуре события или модуле кода заданное число раз. Этот подход полезен в том случае, если вы выполняете несколько связанных вычислений, работаете с элементами на экране или обрабатываете несколько фрагментов данных, введенных пользователем. Цикл For…Next является на самом деле просто кратким способом записать длинный список операторов программы. Так как каждая группа операторов в таком списке будет делать одно и то же, Visual Basic позволяет определить одну такую группу операторов и выполнить ее столько раз, сколько вы захотите.

Оператор цикла For…Next

Синтаксис цикла For…Next имеет следующий вид:

For переменная = начальное_значение To конечное_ значение

повторяемые операторы

Next [переменная]

В этом описании синтаксиса For, To и Next - это обязательные ключевые слова, оператор присвоения (=) также обязателен. Вы должны заменить переменная именем числовой переменной, которая хранит текущее значение счетчика циклов (переменная после Next указывать необязательно), а начальное_значение и конечное_значение заменить числовыми значениями, представляющими начальную и конечную точки цикла. (Заметьте, что вы должны объявить переменную до того, как станете использовать ее в операторе For…Next.) Строка или строки между операторами For и Next - это операторы, которые повторяются при каждом исполнении цикла.

Оператор цикла For…Next

Например, следующий цикл For…Next воспроизводит в виде быстрой последовательности четыре звуковых сигнала:

```
Dim i As Integer
For i = 1 To 4
Beep()
Next i
```

Этот цикл эквивалентен написанию в процедуре оператора Beep четыре раза. Компилятор воспринимает его точно так же, как и

```
Beep()
Beep()
Beep()
Beep()
```

Оператор цикла For…Next

Переменная-счетчик `i` в цикле `For…Next` может стать мощным инструментом вашей программы. При некотором воображении вы можете использовать ее для создания в ваших циклах нескольких полезных последовательностей чисел. Чтобы создать цикл с шаблоном счетчика, отличным от 1, 2, 3, 4 и т.д., вы можете указать в цикле различные начальные значения, а затем использовать ключевое слово `Step` для увеличения счетчика на различные интервалы.

Оператор цикла For…Next

Например, код

```
Dim i As Integer
Dim Wrap As String
Wrap = Chr(13) & Chr(10)
For i = 5 To 25 Step 5
TextBox1.Text = TextBox1.Text & "Строка " & i & Wrap
Next i
```

отобразит в текстовом поле следующую последовательность номеров строк:

```
Строка 5
Строка 10
Строка 15
Строка 20
Строка 25
```

Оператор цикла For…Next

Если вы объявите i как переменную с плавающей точкой одинарной или двойной точности, вы сможете указать в цикле десятичные значения. Например, цикл For…Next

```
Dim i As Single
Dim Wrap As String
Wrap = Chr(13) & Chr(10)
For i = 1 To 2.5 Step 0.5
TextBox1.Text = TextBox1.Text & "Строка " & i & Wrap
Next i
```

отобразит в текстовом поле следующие номера строк:

```
Строка 1
Строка 1.5
Строка 2
Строка 2.5
```

Оператор Exit For

Большинство циклов `For…Next` выполняются до конца без каких-либо проблем, но иногда бывает нужно остановить работу цикла `For…Next` "досрочно" при выполнении некоторого условия. Такую возможность предоставляет использование оператора `Exit For` - специального оператора для досрочного завершения выполнения цикла `For…Next` и передачи управления на первый оператор, стоящий после этого цикла.

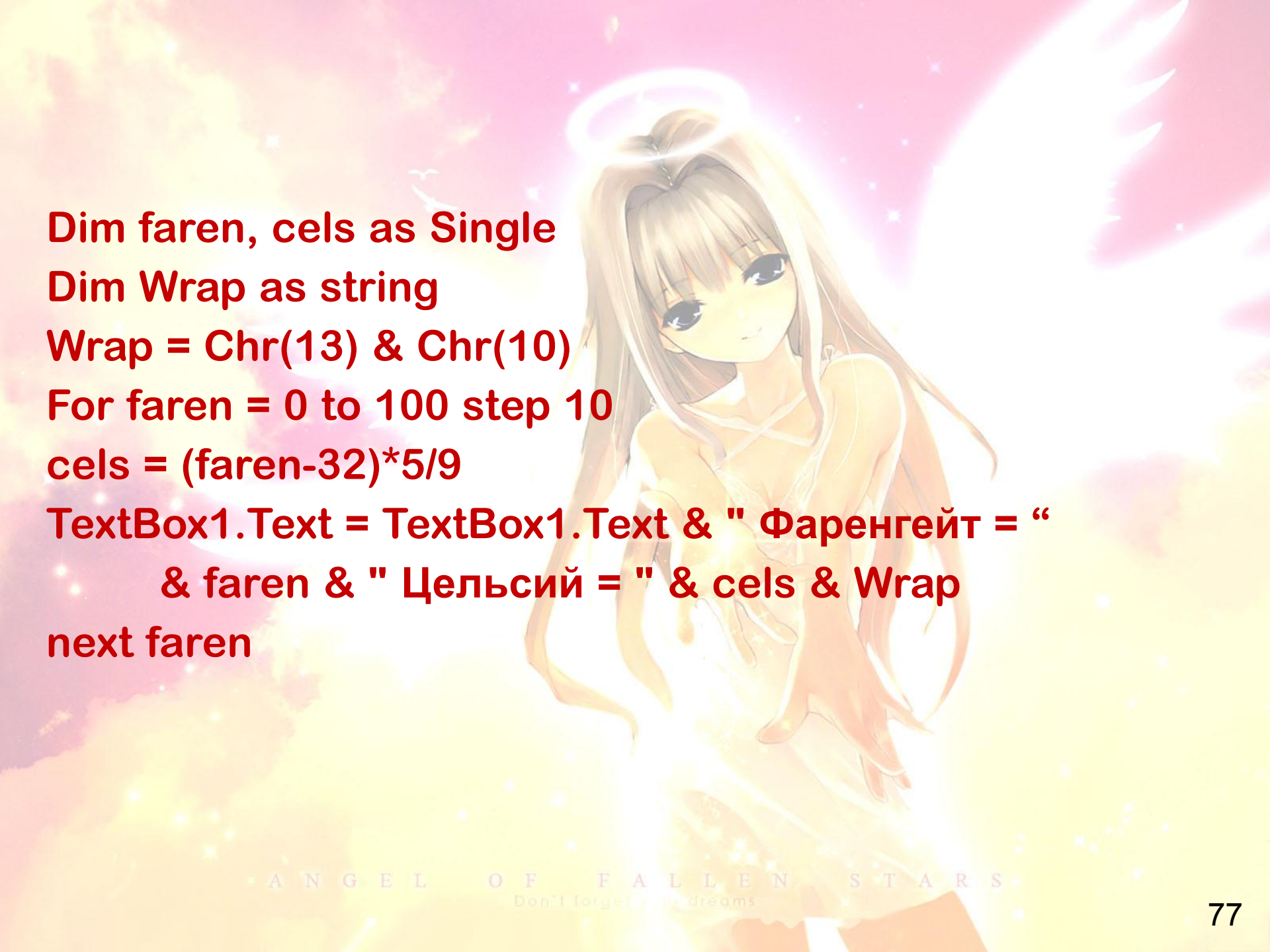
Например, следующий цикл `For…Next` запрашивает у пользователя 10 имен и отображает их одно за другим в текстовом поле до тех пор, пока пользователь не введет слово "Готово":

```
Dim i As Integer
Dim InpName As String
For i = 1 To 10
InpName = InputBox("Введите ваше имя или наберите Готово для выхода.")
If InpName = "Готово"  Then Exit For
 Label1.Text = "Вас зовут " & InpName
Next i
```

задача

Создать таблицу перевода значений температуры по Фаренгейту в интервале от 0 до 100 градусов в значения температуры по Цельсию с шагом 10 градусов, используя формулу:

$$t^c = 5/9 * (t^f - 32)$$



```
Dim faren, cels as Single
Dim Wrap as string
Wrap = Chr(13) & Chr(10)
For faren = 0 to 100 step 10
cels = (faren-32)*5/9
TextBox1.Text = TextBox1.Text & " Фаренгейт = “
        & faren & " Цельсий = " & cels & Wrap
next faren
```

Задача о бегуне

Посчитать сколько калорий потратит бегун на стадионе за 15 кругов, если на первом круге он тратит 20ккал, а на каждом следующем на 5% больше.

```
Dim sum, proc as Single
Dim i As Integer
Const rashod1 as Single = 20.0
proc = rashod1 *0.05
sum = rashod1
rashod = rashod1
For i = 2 To 15
sum = sum + rashod + proc
rashod = rashod + proc
Next i
MsgBox("Всего калорий = " & sum)
```

Цикл Do

В качестве альтернативы циклу `For…Next` можно написать цикл `Do`, который исполняет группу операторов до тех пор, пока некоторое условие не станет равно `True`. Циклы `Do` ценны тем, что зачастую вы не сможете заранее узнать, сколько раз цикл должен повторяться. Например, вы можете позволить пользователю вводить имена в базу данных до тех пор, пока пользователь не введет в поле ввода слово "Готово". В этом случае можно использовать цикл `Do`, повторяющийся до тех пор, пока не будет введено слово "Готово".

Цикл Do

Цикл Do имеет несколько форматов, зависящих от того, где и как вычисляется условие цикла. Чаще других встречается такой синтаксис:

Do While условие

блок выполняемых операторов

Loop

while – остановись, когда условие нарушится

Until – остановись, когда условие выполнится

Например, следующий цикл Do будет запрашивать у пользователя данные и отображать их в текстовом поле до тех пор, пока не будет введено слово "Готово":

```
Dim InpName As String
Do While InpName <> "Готово"
InpName = InputBox("Введите ваше имя или наберите  Готово для выхода.")
If InpName <> "Готово" Then
Label1.Text = InpName
Loop
```

Условный оператор в этом цикле имеет вид
`InpName <> "Готово",`
и компилятор Visual Basic транслирует его в значение "повторять цикл до тех пор, пока переменная `InpName` не будет содержать слова Готово". Это приводит к одному интересному моменту, касающемуся циклов `Do`: если условие в начале цикла не равно `True` при первом выполнении оператора `Do`, то цикл `Do` никогда не исполняется. Таким образом, если строковая переменная `InpName` содержит значение "Готово" до начала цикла (возможно, в результате более раннего присвоения значения в процедуре события), Visual Basic пропустит весь цикл и продолжит выполнение со строки, расположенной вслед за ключевым словом `Loop`.

Если вы хотите, чтобы в программе цикл всегда выполнялся хотя бы один раз, поместите проверку условия в конце цикла. Например, цикл

```
Dim InpName As String
Do
InpName = InputBox("Введите ваше имя или наберите Готово для выхода.")
If InpName <> " Готово" Then
TextBox1.Text = InpName
Loop While InpName <> " Готово"
```

Это пример цикла с постусловием.

Предотвращение бесконечных циклов

Из-за существующей структуры циклов Do, очень важно проектировать проверки условий таким образом, чтобы каждый цикл имел реальную точку выхода. Если проверка условий выполнения цикла никогда не даст в результате False, цикл будет выполняться бесконечно и ваша программа перестанет реагировать на ввод. Рассмотрим следующий пример:

```
Dim Number as Double
Do
Number = InputBox("Введите число для возведения
в квадрат. Введите -1 для выхода.")
Number = Number * Number
TextBox1.Text = Number
Loop While Number >= 0
```

В этом цикле пользователь вводит одно число за другим, а программа возводит число в квадрат и отображает результат в текстовом поле.

К сожалению, пользователь, закончив работу, не сможет выйти, так как представленное условие выхода не работает. Когда пользователь вводит -1, программа возводит его в квадрат, и переменной `Number` присваивается значение 1. (Проблему можно решить, поставив другое условие выхода.) Отслеживание бесконечных циклов при написании циклов `Do` очень важно. К счастью, если вы тщательно тестируете программу, то их довольно легко обнаружить.

Использование в циклах Do ключевого слова Until

Циклы Do, с которыми вы до сих пор работали, для выполнения группы операторов до тех пор, пока условие равно True, использовали ключевое слово While.

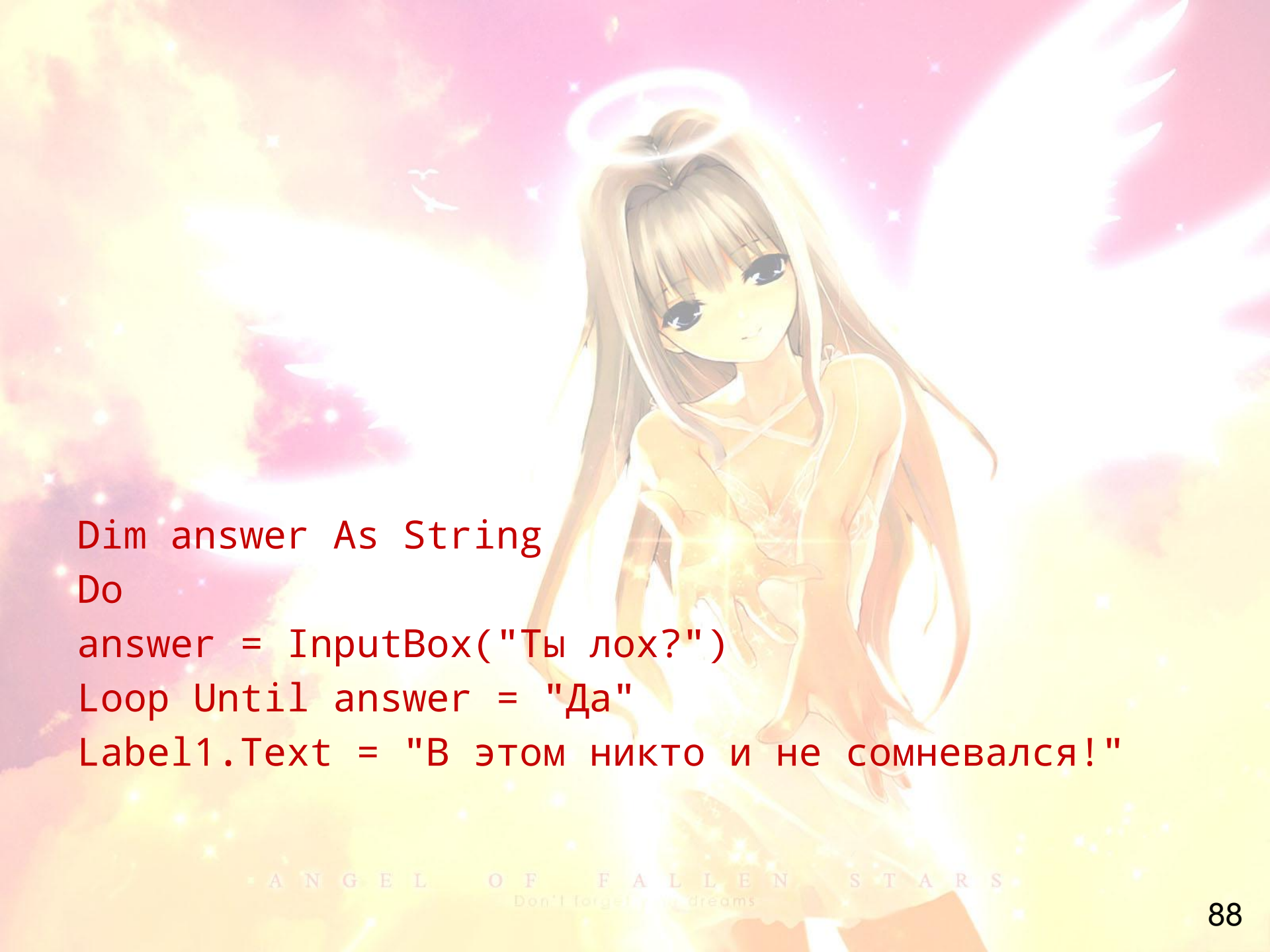
Visual Basic также позволяет использовать в циклах Do ключевое слово Until, которое приводит к повторению цикла до тех пор, пока заданное условие не станет равным True.

Ключевое слово Until можно использовать для проверки условия, как в начале, так и в конце цикла Do, точно так же, как и ключевое слово While.

Например, следующий цикл Do использует ключевое слово Until для повторения цикла до тех пор, пока пользователь не ввел в поле ввода слово "Готово":

```
Dim InpName As String
Do
InpName = InputBox("Введите ваше имя или
наберите Готово для выхода.")
If InpName <> "Готово" Then TextBox1.Text =
InpName
Loop Until InpName = "Готово"
```

Цикл, который использует ключевое слово Until, аналогичен циклу, который использует ключевое слово While, за исключением того, что проверка условия обычно содержит обратный оператор - в данном случае это оператор = (равно) вместо оператора <> (не равно).



```
Dim answer As String
Do
answer = InputBox("Ты лох?")
Loop Until answer = "Да"
Label1.Text = "В этом никто и не сомневался!"
```

Структурированные данные

Для повышения производительности и качества работы необходимо иметь данные, максимально приближенные к реальным аналогам.

Тип данных, позволяющий хранить вместе под одним именем несколько переменных, называется структурированным.

Каждый язык программирования имеет свои структурированные типы.

Рассмотрим структуру, объединяющую элементы одного типа данных, — массив.

Массивы данных

Массивом называют упорядоченную последовательность однотипных данных, обозначенных одним именем.

Имя массива формируется так же, как и имя переменной.

Отдельные величины, составляющие массив данных, называются элементами массива. К каждому элементу массива можно обратиться, указав имя массива и индекс (номер), который показывает положение элемента в массиве.

Описание массива на языке Visual Basic:

```
Dim A(5) As Integer
```

A – имя массива

5 – это размер массива, то есть, сколько элементов в массиве.

A(0)	A(1)	A(2)	A(3)	A(4)	A(5)

Например, если сказано: «задан массив A(10)», это означает, что даны элементы: $a_1\ a_2, \ldots, a_{10}$.

Ввод элементов одномерного массива осуществляется поэлемен тно, в порядке, необходимом для решения конкретной задачи. Обычно, когда требуется ввести весь массив, элементы вводятся в порядке возрастания их индексов.

Пример записи данных в конкретный элемент:

```
A(1)=100
Msgbox (A(1),msgboxStyle.Information,
” Значение первого элемента массива”)
```

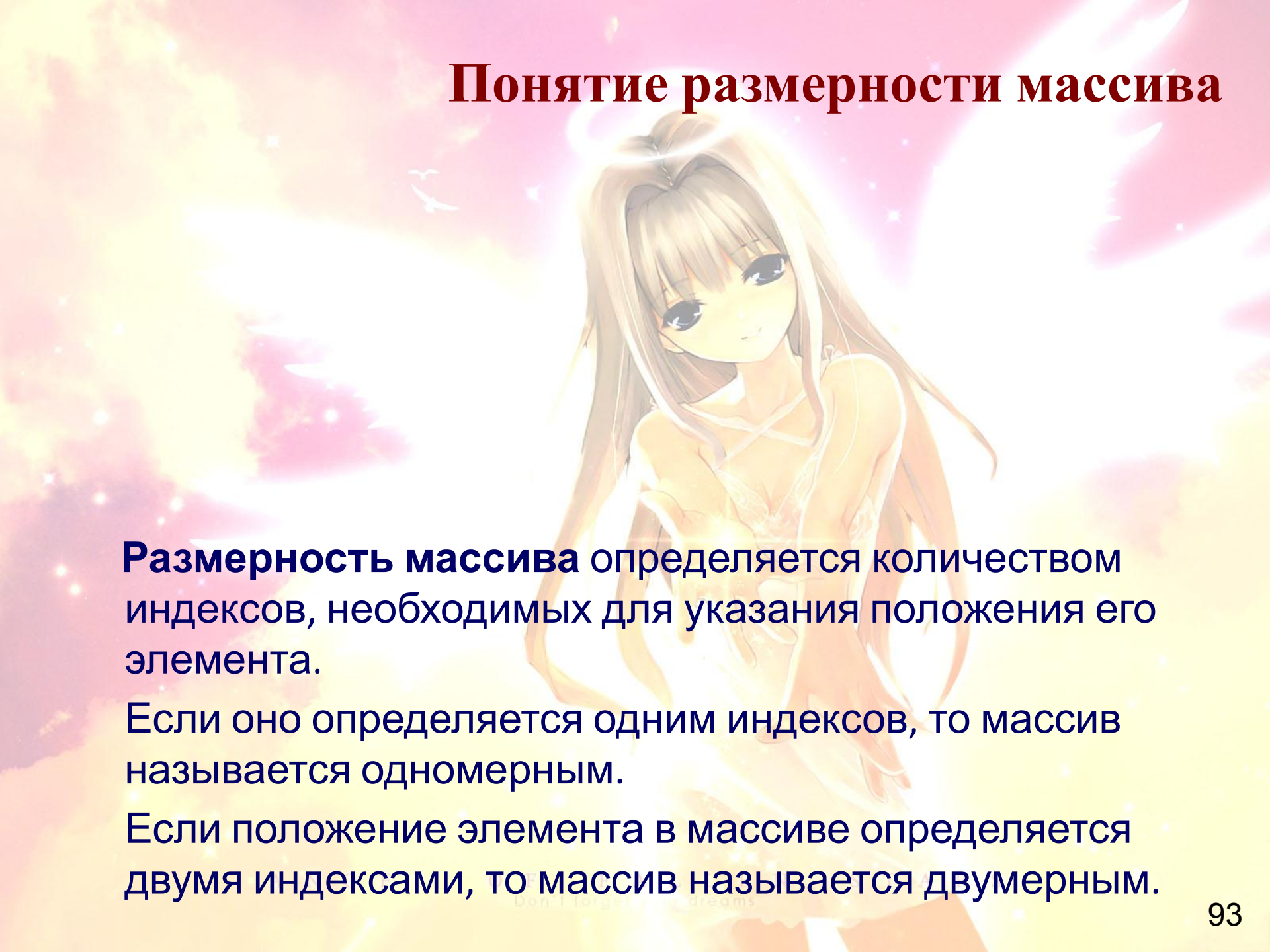
В общем случае во многих языках программирования Массивы бывают статические и динамические.

Статические массивы создаются для хранения определенного числа элементов, и далее в программе их количество больше не меняется.

Динамические массивы в процессе выполнения программы могут изменять свой размер.

Такие массивы удобно использовать, если заранее неизвестен размер массива.

Понятие размерности массива



Размерность массива определяется количеством индексов, необходимых для указания положения его элемента.

Если оно определяется одним индексов, то массив называется одномерным.

Если положение элемента в массиве определяется двумя индексами, то массив называется двумерным.

Задать и вывести на экран элементы одномерного массива.

```
Dim a(5) As Integer
Dim i As Integer
For i = 0 To 5
   a(i) = Val(TextBox1.Text)
    Label1.Text = a(i)
Next i
```

В заданном числовом массиве найти сумму всех элементов.

```
Dim a() As Integer
Dim i, n, Summa As Integer
n = Val(InputBox("Введите размер массива"))
ReDim a(n)
For i = 0 To n
   a(i) = Val(InputBox("Next element:"))
    Label1.Text = a(i)
Next i
Summa = 0
For i = 0 To n
   Summa = Summa + a(i)
Next i
MsgBox(Summa,MsgBoxStyle.Information,"Сумма всех чисел“)
```

В заданном числовом массиве найти максимальный элемент (число).

```
Dim a() As Integer
Dim i, n, max As Integer
n = Val(InputBox("Введите размер массива"))
ReDim a(n)
For i = 0 To n
   a(i) = Val(InputBox("Next element:"))
   MsgBox a(i)
Next i
max = a(0)
For i = 0 To n
   If a(i) > max Then max = a(i)
Next i
MsgBox(max, MsgBoxStyle.Information,
   "Максимальный элемент")
```

Сумма и количество четных элементов (чисел):

```
Dim a() As Integer
Dim i, n, Kol, Summa As Integer
n = Val(InputBox("Введите размер массива"))
ReDim a(n)
For i = 0 To n
a(i) = int(rnd*100)
Next i
  Kol = 0
  Summa = 0
     For i = 0 To n
         If a(i) mod 2 = 0 Then
             Kol = Kol + 1
             Summa = Summa + a(i)
         End If
     Next i
     MsgBox(Kol, MsgBoxStyle.Information, "Количество
четных чисел")
     MsgBox(Summa, MsgBoxStyle.Information, "Сумма
четных чисел")
```

Двумерные массивы

Если положение элемента в массиве определяется двумя индексами (номером по строке и номером по столбцу), то **массив называется двумерным**.

В программе на языке Basic двумерный массив описывается следующим образом:

`Dim A(m,n) as Тип массива`, где `m` – количество строк, `n` – количество столбцов.

Графическое представление двумерного массива

A(0,0)	A(0,1)	A(0,2)	…	A(0,n)
A(1,0)	A(1,1)	A(1,2)	…	A(1,n)
A(2,0)	A(2,1)	A(2,2)	…	A(2,n)
…	…	…	…	…
A(m,0)	A(m,1)	A(m,2)	…	A(m,n)

Заполнение двумерного массива данными:

Можно заполнять по строкам или по столбцам с помощью двух вложенных циклов.

Циклы называются вложенными, если один находится внутри другого, например:

```
For i = 0 To 2
    For j = 0 To 3
  MsgBox(i & j)
    Next j
 Next i
```

Числа напечатаются в следующем порядке:

```
0,0; 0,1; 0,2; 0,3  1,0; 1,1; 1,2; 1,3  2,0; 2,1; 2,2; 2,3
```

Пример на двумерный массив

```
Dim a(,), m, n As Integer
Dim i, j As Integer
m = Val(InputBox("Введите количество строк"))
n = Val(InputBox("Введите количество столбцов"))
ReDim a(m, n)
Label1.Text = ""
For i = 0 To m
  For j = 0 To n
    a(i,j) = Int(Rnd() * 10)
    Label1.Text = Label1.Text & " " & a(i,j)
  Next j
  Label1.Text = Label1.Text & vbCrLf
Next i
```

Символьные данные

Символьные данные иначе называются текстовыми, строковыми, литерными. Под *символьными данными* подразумевается весь набор алфавита языка Basic.

Символьные данные бывают константами и переменными.

Символьная константа – это набор символов, заключенных в двойные кавычки.

Количество символов, содержащихся в символьной константе, называется **длиной символьной константы**.

Символьная переменная, содержащая символьную константу, называется **символьной переменной**.

На языке Visual Basic Net символьная переменная описывается следующим образом:

`Char` — один символ формата Unicode (от 0 до 65535).

`String` — Строка постоянной длины (Длина может быть от 0 до приблизительно двух миллионов символов).

Над символьными данными можно производить операцию *слияния* или *конкатенации*. Она обозначается знаком &. Иногда ее обозначают знаком +, что осталось от предыдущих версий.

Символы в компьютере запоминаются в числовых кодах, то есть каждому символу ставится в соответствие какой-то числовой код. Разработано несколько систем кодирования символов. Самой распространенной оказалась ASCII система. В этой системе для хранения каждого символа отводится 1 байт памяти. А 1 байт может хранить числа от 0 до 255. Таким образом, в ASCII системе можно закодировать 256 символов.

Функция Len() – возвращает количество символов в строке, заданной в качестве аргумента.

Пример использования:

```
Dim n As Integer
n= Len("Я изучаю Visual Basic .Net")
Label1.text= n  '---результат будет 26
```

Функция Mid(s,n1,n2) – возвращает часть строки, согласно аргументам:

s – исходная строка,

n1 – номер символа, с которого начинается выделяемая часть строки,

n2 – количество выделяемых символов.

Пример использования:

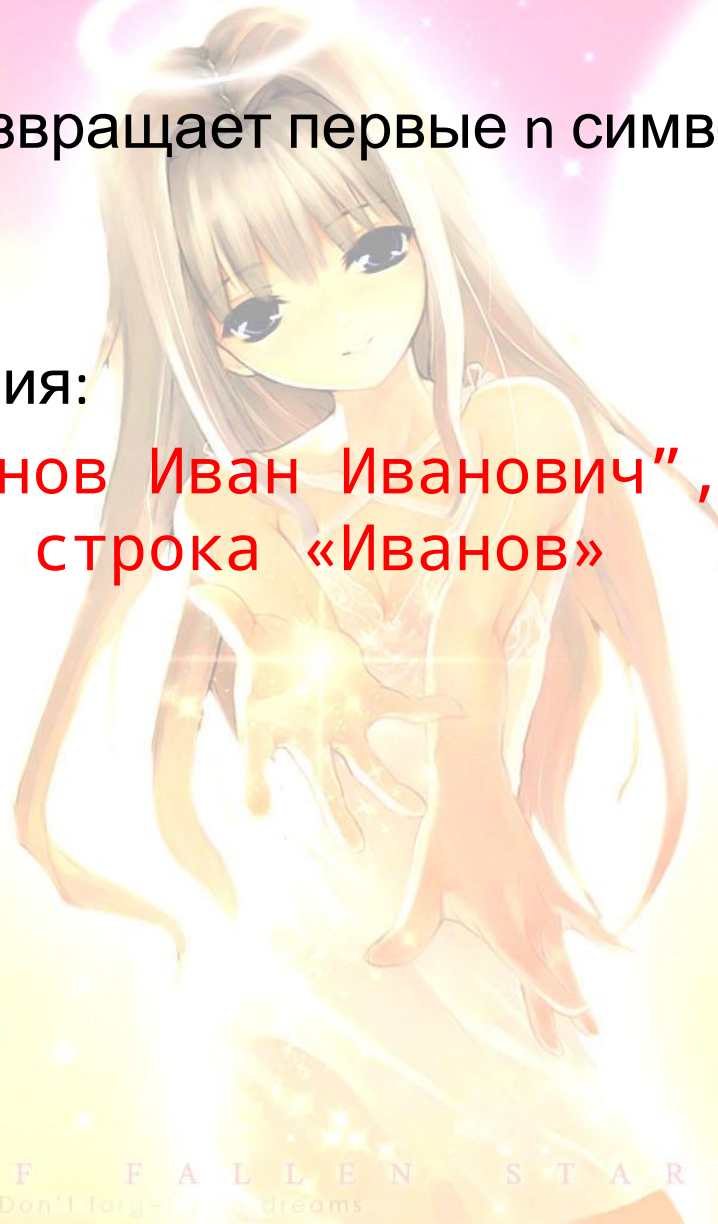
```
Msgbox Mid(“Иванов Петр Сидорович”,8,4)
‘--- слово Петр начинается с восьмого символа и имеет длину 4 символа
```

Результатом выполнения данного примера будет выведение на экран слова Петр

Функция Left(s,n) – возвращает первые n символов строки s.

Пример использования:

```
Msgbox Left(“Иванов Иван Иванович”,6)
‘--- результат - строка «Иванов»
```



Функция Right(s,n) – возвращает последние n символов строки s.

Пример использования:

```
Msgbox Right("Протвино", 4) '--- результат «вино»
```

Функция GetChar(s,n) – возвращает символ строки s с заданным индексом (номером).

Индексация символов в строке, в отличие от массивов, начинается с 1, а не с 0.

Пример использования:

```
Msgbox GetChar(“Дубна”,4) ‘- Результат будет буква «н»
```

Функция InStr(s,s1) – возвращает позицию первого вхождения одной строки (s1) символов в другую (s).

Пример использования:

```
Msgbox Instr("Иванов Иван Иванович","Иван")
'--- Результат будет число 1.
```

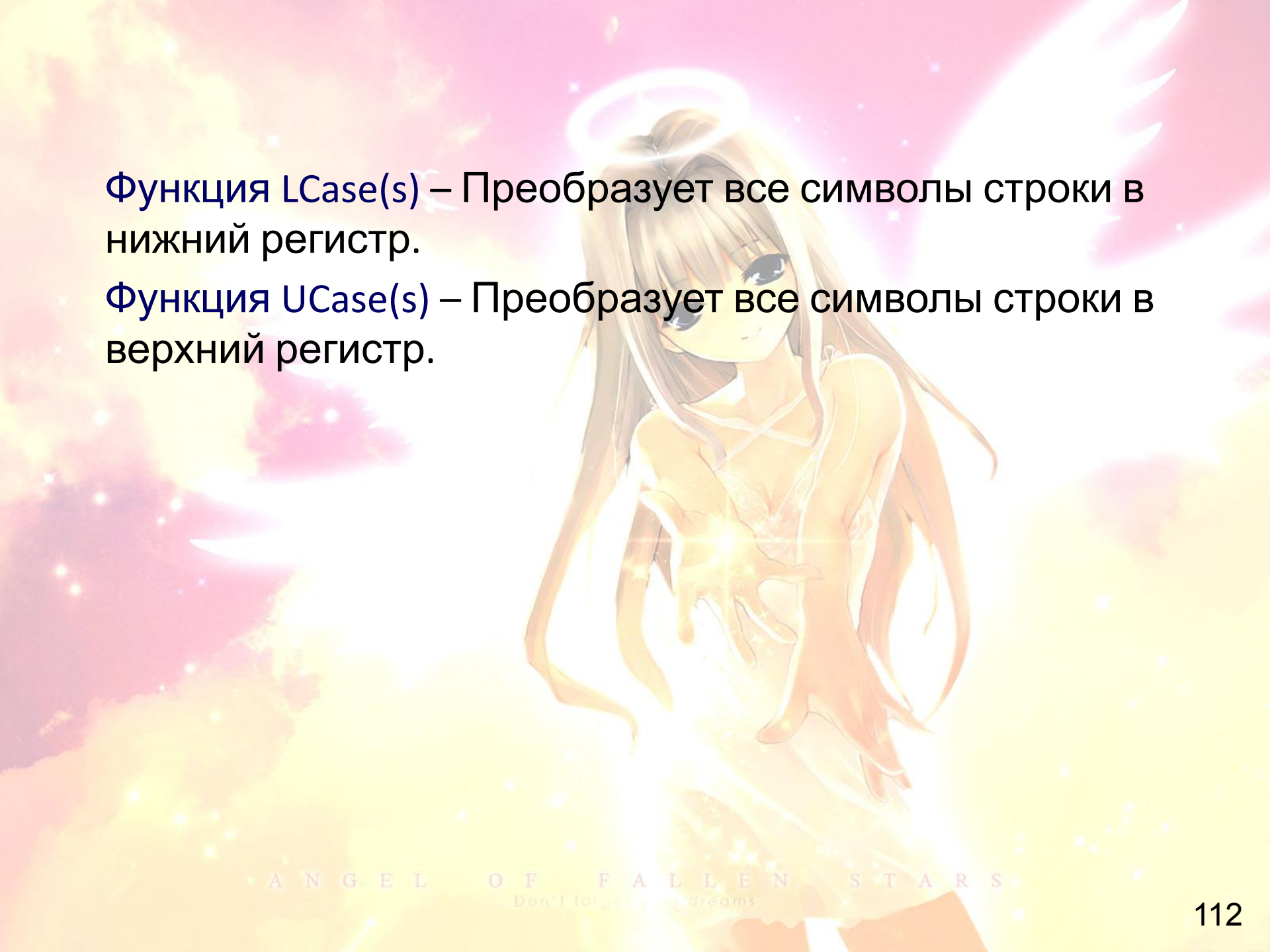
Функция InStrRev(s,s1) – возвращает позицию последненго вхождения одной строки (s1) символов в другую (s).

Пример использования:

```
Msgbox InstrRev(“Иванов Иван Иванович”,“Иван”)
‘--- Результат будет число 13.
```

Функция LCase(s) – Преобразует все символы строки в нижний регистр.

Функция UCase(s) – Преобразует все символы строки в верхний регистр.



Функция LTrim(s) – Удаляет пробелы в начале строки, если они есть

Пример использования:

```
Msgbox LTrim(“  Дубна”)    ‘--- Результат будет “Дубна”
```

Функция RTrim(s) – Удаляет пробелы в конце строки, если они есть

Пример использования:

```
Msgbox RTrim(“Дубна  ”)‘--- Результат будет “Дубна”
```

Функция Trim(s) – Удаляет пробелы в начале и в конце строки, если они есть

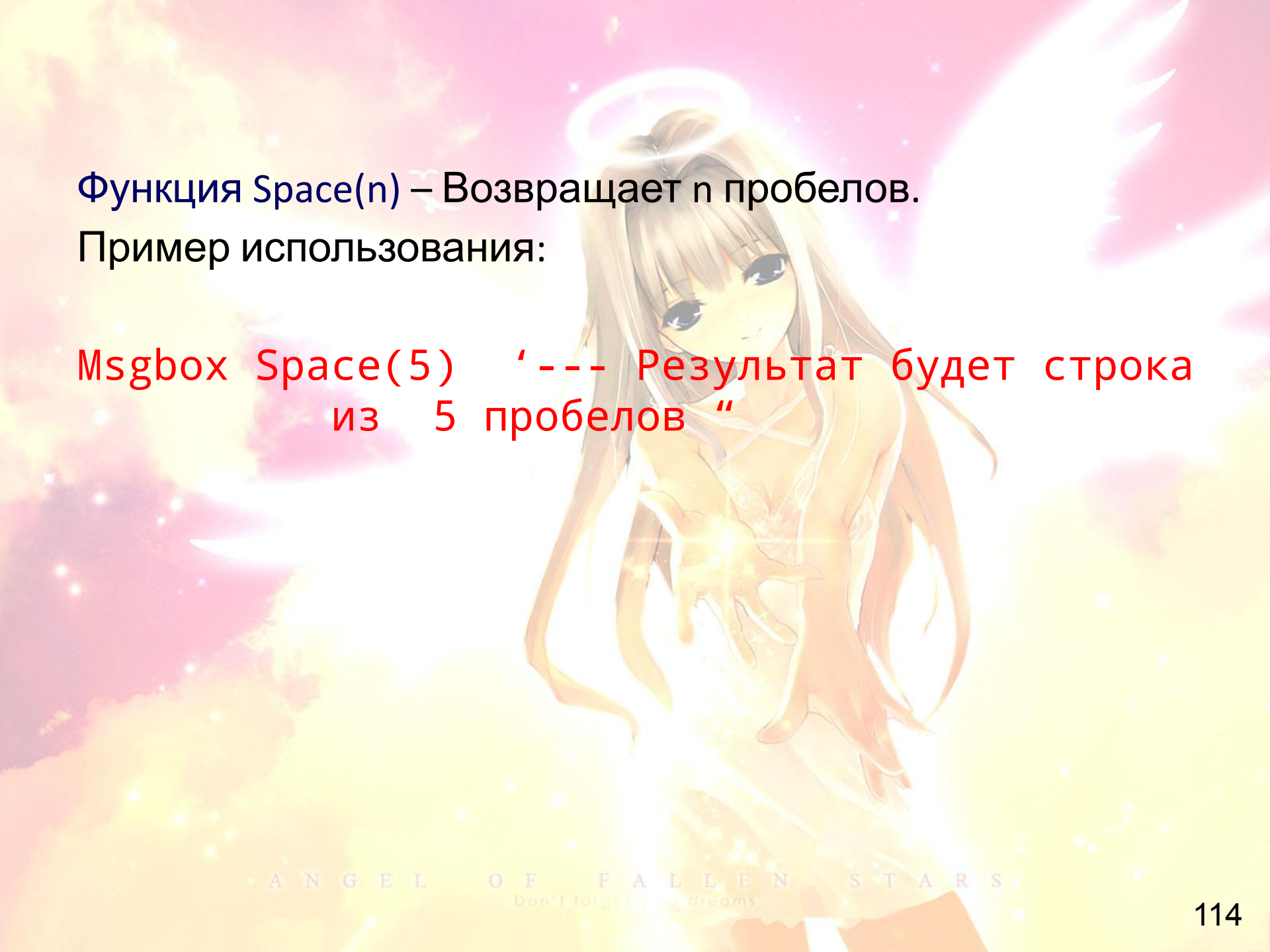
Пример использования:

```
Msgbox Trim(“  Дубна  ”) ‘--- Результат будет “Дубна”
```

Функция Space(n) – Возвращает n пробелов.

Пример использования:

```
Msgbox Space(5)  '--- Результат будет строка
          из  5 пробелов "
```



Функция Replace(s,s1,s2,n1,n2) – заменяет часть заданной строки на указанную строку, согласно заданным аргументам:

S – исходная строка.

S1 – строка, которая будет заменяться.

S2 – строка, на которую будет заменяться строка S1.

N1- позиция символа в s, с которого надо начинать поиск (по умолчанию 1).

N2 – количество замен, которые необходимо выполнить (по умолчанию -1, то есть надо произвести все замены).

Пример использования:

```
Msgbox Replace("Иванов Иван Иванович","Иван", "Петр",1, -1)     '--- Результат будет строка «Петров Петр Петрович».
```

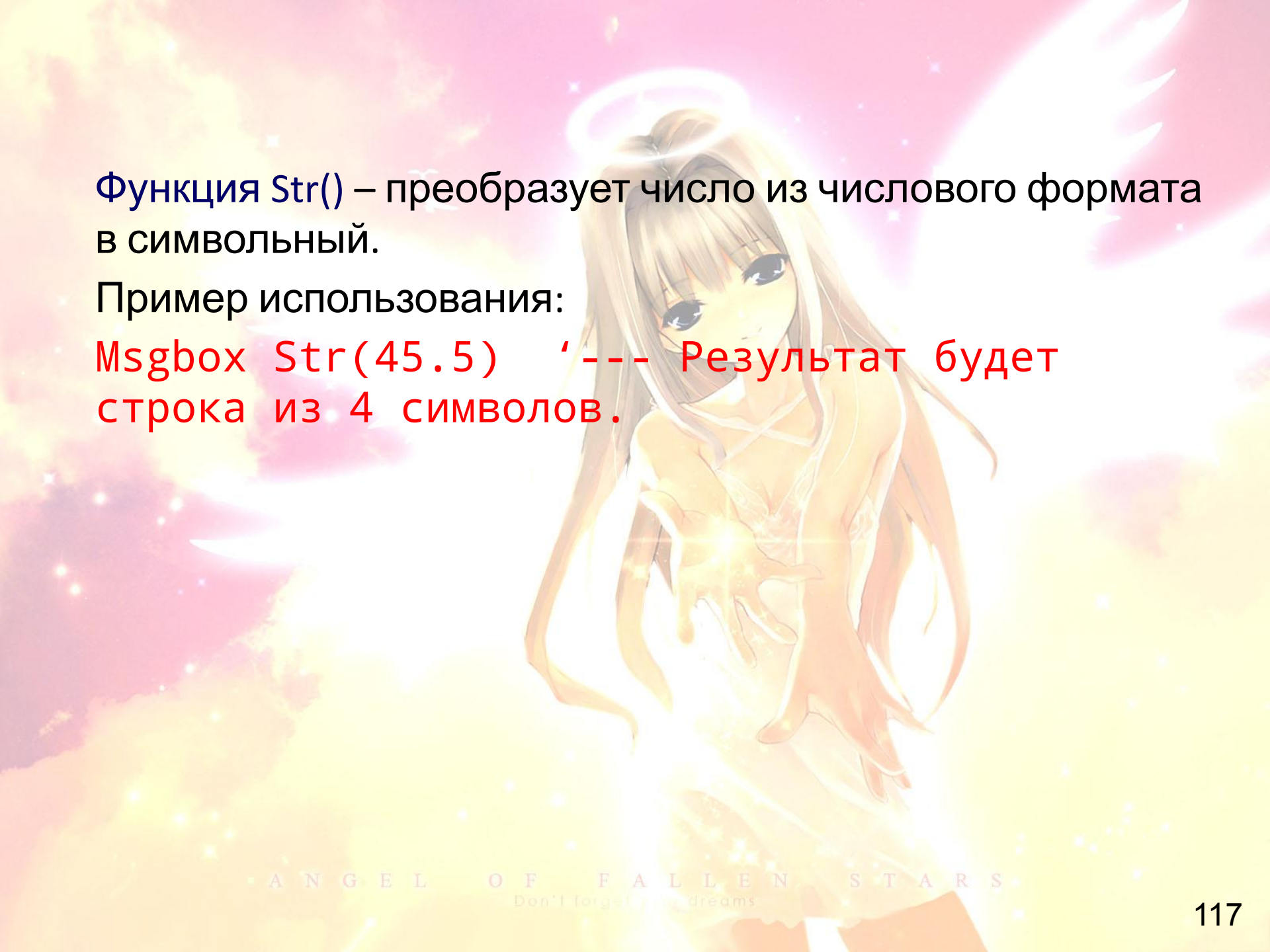
Функция Split() – Возвращает n пробелов – применяется для разбиения строки на различные строки по заданным разделителям. Основной синтаксис:

Split(строка, разделитель, количество разбиений)

По умолчанию разделителем является пробел, количество разделителей не ограничено (-1)

Пример использования:

```
Dim s(2) As String ‘--- Создаем массив из трех элементов
s = Split("Иванов Иван Иванович", " ", -1)
MsgBox(s(0), MsgBoxStyle.Information, "1 слово") ‘--- Иванов
MsgBox(s(1), MsgBoxStyle.Information, "2 слово") ‘--- Иван
MsgBox(s(2), MsgBoxStyle.Information, "3 слово") ‘---Иванович
```



Функция Str() – преобразует число из числового формата в символьный.

Пример использования:

```
Msgbox Str(45.5)  '--- Результат будет строка из 4 символов.
```

Функция Val () – преобразует число из символьного формата в числовой.

Пример использования:

```
Msgbox Val(“45.5”) ‘ Результат будет число 45,5
```

Если задана строка, содержит кроме цифр и другие символы, то за число принимаются все до первого символа, который не может входить в формирование числа.

```
Msgbox Val(“45.5f”) ‘ Результат будет число 45,5
Msgbox Val(f“45.5”) ‘ Результат будет число 0.
```

Модульное программирование

Предположим, что алгоритм решения задачи уже разработан. Прежде чем писать программу, её необходимо *спроектировать*, т.е. выявить взаимосвязанные небольшие части программы *(модули),* дальнейшее уточнение которых может быть продолжено независимо друг от друга. Степень такой модульности определяет гибкость и изменяемость программы.

Существует метод разработки программ – **модульное программирование**, предполагающий разбиение программы на независимые модули. В качестве модульной структуры программы принято использовать древовидную структуру.

Использование модульного программирования предоставляет ряд существенных преимуществ. К ним, в первую очередь, относятся:

возможность использования модулей, которые ранее созданы в других программах, т.е. уменьшается дублирование;

замена модулей на более современные (эффективные).

Таким образом, модульное программирование, в котором применяется принцип “разделяй и властвуй”, является методом «борьбы со сложностью программ».

Процедуры

Модульный подход в Visual Basic реализуется с помощью процедур.

Процедура представляет собой блок программного кода, выполняющий определённую задачу, который представляет собой законченную программу и оформленный специальным образом.

В Visual Basic.Net существует два вида процедур:

Подпрограммы (Sub procedure);

Функции (Function procedure).

Главным отличием *функции от подпрограммы* является то, что функция может возвращать под своим именем какое-либо значение, а подпрограмма – нет.

Подпрограммы в Visual Basic.Net бывают двух типов: обработчики событий и подпрограммы, определённые пользователем. До сих пор мы использовали процедуры обработки событий, которые были связаны с определённым объектом и событием, например, щелчок мышью на кнопке (Click). При наступлении события автоматически выполняется процедура его обработки.

Подпрограммы

Синтаксис, который применяется при определении пользовательской подпрограммы, выглядим следующим образом:

Sub имя_процедуры ([список_параметров])

[операторы]

End Sub

Квадратные скобки [] при описании синтаксиса в языках программирования означают необязательную часть конструкции. В данном случае, параметры могут отсутствовать.

Имя_процедуры – однозначно идентифицирует данную процедуру и используется для ее вызова из основной программы. Имя процедуры должно быть уникальным, то есть не должно совпадать ни с именем какой-либо стандартной функции, ни с именами функций, созданных пользователем.

Список параметров – это перечень типов и имён ячеек памяти, которые подпрограмма использует при хранении переданной ей информации. Если переменных несколько, то они разделяются запятой.

Например, следующая подпрограмма отображает текущую дату в окне сообщений:

```
Sub ShowDate()
MsgBox Date()
End Sub
```

Рассмотрим подпрограмму с двумя параметрами a и b, которая вычисляет сумму чисел a и b. Результат заносится в переменную c и выводится на экран с помощью MsgBox:

```
Sub Add(ByVal a As Integer, ByVal b As Integer)
Dim c As Integer
c =  a + b
MsgBox (c)
End Sub
```

Ключевое слово ByVal, предшествующее имени параметра, означает, что подпрограмма работает не с самой переменной, а с её копией, т.е. в данном случае подпрограмма не может изменить значения переменных a и b.

Вызов подпрограммы можно записать так:

```
Add (7,5)
Call Add (7,5)

или

Dim x, y As Integer
x = 7
y = 5
a= Add (x, y)
```

В обоих случаях на экран будет выводиться число 12.

Функции

Функция – процедура, которая возвращает под своим именем результирующее значение.

До сих пор мы использовали стандартные функции, предоставляемых нам средой программирования Visual Basic.Net. Это и числовые, и символьные функции. Теперь же мы научимся создавать свои собственные функции.

Синтаксис функции:

```
Function имя_функции ([список_параметров]) As тип_результата
[операторы]
End Function
```

Имя_функции и список_параметров имеют тот же смысл, что и для подпрограммы.

Тип_результата – показывает какого типа будет результат выполнения функции, который будет возвращён в программу, вызвавшую данную функцию.

Для создания новой функции или подпрограммы необходимо установить курсор за пределами других функций или подпрограмм и начать вводить заголовок функции или подпрограммы. После ввода заголовка и нажатия клавиши *Enter,* редактор кода Visual Basic.Net автоматически создаст команду завершения функции или подпрограммы, то есть `End Function` или `End Sub`. Все операторы, которые мы напишем внутри, между заголовком и концом, и будут составлять блок функции или подпрограммы.

Для возврата значения из функции используется два способа:

1. оператор `Return`, – может появляться в любом месте программного кода функции и встречаться несколько раз. После выполнения оператора `Return` функция заканчивает работу и управление передаётся вызывающей программе.
2. оператор присваивания результирующего значения имени функции.

Рассмотрим пример функции нахождения максимального значения из трех чисел l, m и n:

```
Function MaxX(ByVal l As Integer, ByVal m As
  Integer, ByVal n As Integer) As Integer
If l > m And l > n Then MaxX = l
If m > l And m > n Then MaxX = m
If n > l And n > m Then MaxX = n
End Function
```

Основная программа, вызывающая данную функцию, может выглядеть следующим образом:

```
Private Sub CommandButton2_Click()
Dim a, b, c, d As Integer
a = 3
b = 4
c = 5
d = MaxX(a, b, c)
MsgBox(d)
End Sub
```

Рассмотрим пример возврата значения из функции с помощью оператора Return. В примере используется операция деления, следовательно, мы должны проверять знаменатель на равенство нулю. Предполагается, что числитель не равен нулю.

```
Function MyDiv(ByVal A1 As Integer, ByVal A2 As
Integer) As Double
'Пример досрочного завершения функции (деление на 0)
     If A2 = 0 Then
       Return 0  'возвращаем 0, если деление на 0
     Else
       Return A1 / A2
     End If
End Function
```

```
Private Sub Butdiv()
        Dim R As Double
        R = MyDiv(7, 2)
        MsgBox(R)
        R = MyDiv(5, 0)
        MsgBox(R)
End Sub
```

– в первом случае

– во втором случае

Передача параметров в процедуры

Каждая переменная имеет значение и уникальный адрес, соответствующий положению переменной в оперативной памяти компьютера.

Visual Basic.Net позволяет передавать в процедуру как значение переменной (передача *по значению*), так и её адрес (передача *по ссылке*).

Выбор метода передачи параметров зависит от того, нужно ли, чтобы процедура имела доступ к переменной и могла изменять её значение.

Передача переменных по значению

При передаче по значению процедура работает с копией параметра, т.е. в данном случае процедура не получает доступ к переменной и не может изменить значения параметров. Перед параметром в списке параметров необходимо добавить ключевое слово `ByVal` (от англ. By value – по значению). Если в процедуре нет необходимости изменять значение переменной, то используйте передачу по значению.

Передача переменных по ссылке

При передаче параметра по ссылке в процедуре передаются как значение, так и адрес переменной, т.е. её положение в оперативной памяти. Таким образом, процедура получает доступ к переменной в памяти. Этот метод используют, когда в процедуре необходимо изменить значение параметра. Перед параметром ставится ключевое слово `ByRef` (от англ. By reference – по ссылке), которое указывает, что в процедуру передаётся адрес переменной.

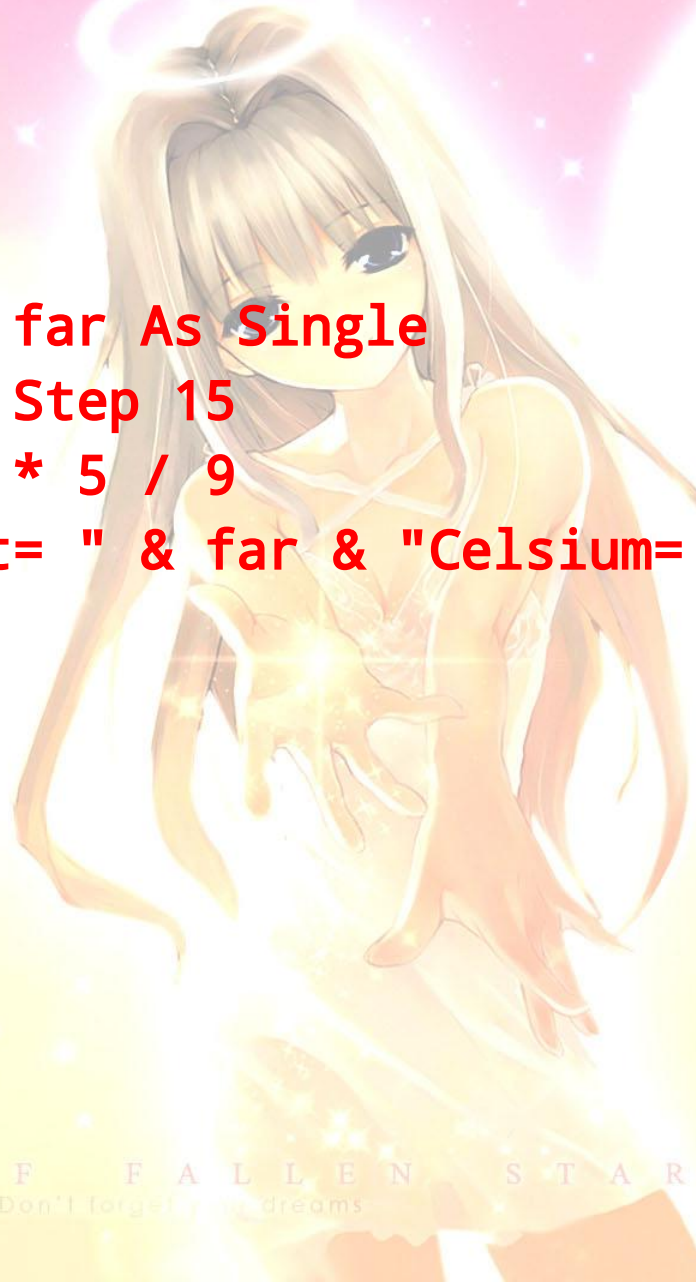
```
Function Add1(ByRef n1 As Integer, ByRef n2
As Integer) As Integer
        Add1 = n1 + n2
        n1 = 0
        n2 = 0
   End Function
```

При выходе из функции параметры n1 и n2 получат значения, равные нулю, т.к. осуществляется передача по ссылке (ByRef).

```
Function Add2(ByVal n1 As Integer, ByRef n2
As Integer) As Integer
   Add2 = n1 + n2
   n1 = 0
   n2 = 0
End Function
```

При выходе из функции параметры n1 получат значение, которое было у него при входе в функцию (не равное нулю, т.к. этот параметр описан по ссылке - ByVal), а n2 получает значение, равное нулю, т.к. осуществляется передача по ссылке (ByRef).

```
Module Module1

  Sub Main()

    Dim cel As Single, far As Single
    For far = 0 To 150 Step 15
      cel = (far - 32) * 5 / 9
      MsgBox("Farengeit= " & far & "Celsium= " & cel)
    Next far

  End Sub

End Module
```



```
Module Module1

    Sub Main()
        Dim a(100) As Integer
        Dim i, max_a As Integer
        max_a = -1
        For i = 1 To 100
          a(i) = Int(Rnd() * 500)
            If a(i) > 100 And a(i) < 250 Then
              If a(i) > max_a Then
                max_a = a(i)
              End If
            End If
        Next
        MsgBox("Максимальное в интервале 100 - 250=" & max_a)
    End Sub

End Module
```

```
Module Module1

    Sub Main()
        REM Тыщук Александр Александрович зад 41,42
        Dim ST As String
        Dim LENGTH, it, countA, countABC As Integer
        countA = 0
        countABC = 0
        ST = InputBox("введите исследуемую строку")
        LENGTH = Len(ST)
        For it = 1 To LENGTH
            If Mid(ST, it, 1) = "a" Then
                countA = countA + 1
                If Mid(ST, it, 3) = "abc" Then
                    countABC = countABC + 1
                End If
            End If
        Next
        MsgBox(countA, MsgBoxStyle.Critical, "число букв A")
        MsgBox(countABC, MsgBoxStyle.Critical, "число сочетаний ABC")
    End Sub

End Module
```

http://www.intuit.ru/department/pl/vbnet/

http://www.intuit.ru/department/pl/vb/

http://msdn.microsoft.com/ru-ru/library/2x7h1hfk.aspx

Объектно-ориентированное программирование в Visual Basic:
http://msdn.microsoft.com/ru-ru/library/b86b82w0%28VS.90%29.aspx