# Тема 9

# *Программирование алгоритмов обработки символьных*

# *и строковых данных*

### 4.9.1. Объявление и обработка символов

Управление электронными документами во многих сферах деятельности является очень важной задачей. В связи с этим VB предоставляет различные механизмы как для работы с текстовыми документами, так и для обработки их содержимого. Наиболее общим типом документов является неформатированный текстовый файл, который состоит из слов, букв, цифр и специальных символов различного назначения.

Известно, что наряду с числовыми данными компьютер может хранить и обрабатывать текстовую информацию. Это, например, тексты документов MS Word, текстовые поля форм VS, текстовые поля баз, данных и др.

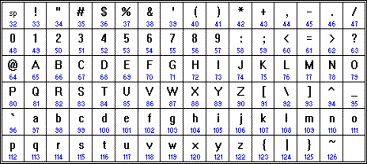
Наиболее часто используемой операцией, которая применяется, например, при упорядочивании данных, является сортировка, т.е. символьные данные сравниваются между собой. В программировании, прежде чем сравнить один символ с другим, он должен быть преобразован в число с помощью таблицы **ASCII** (American Standard Code for Information Interchange – Американский стандартный код для обмена информацией). Поддерживает кодирование 128 буквенно-цифровых символов.

Первые 32 кода базовой таблицы, начиная с нулевого, отданы разработчикам аппаратных средств (в первую очередь производителям компьютеров и печатающих устройств). В этой области размещаются так называемые **управляющие коды**, которым не соответствуют никакие символы языков, и, соответственно, эти коды не выводятся ни на экран, ни на устройства печати, но используются для функций управления (например, возврата каретки или возврата на один символ).

Национальные стандарты кодировочных таблиц включают международную часть кодовой таблицы без изменений, а во второй половине содержат коды национальных алфавитов, символы псевдографики и некоторые математические знаки.

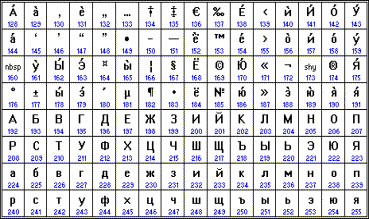
Начиная с кода 32 по код 127 размещены коды символов ***английского алфавита***, ***знаков препинания, цифр, арифметических действий и некоторых вспомогательных символов***. Базовая таблица кодировки ASCII показана в табл. 4.9.1-1.

Таблица 4.9.1-1. Базовая таблица кодировки **ASCII**



Наиболее распространенной в настоящее время является кодировка Windows 1251 (табл. 4.9.1-2) с учетом широкого использования операционных систем и других продуктов этой компании в России.

Таблица 4.9.1-2. Таблица кодировки Windows 1251



В 1991 г. появился новый международный стандарт **Unicode** –   
16-разрядная система кодирования, совместимая с системой ASCII. В Unicode под один символ отводится не один байт, а два, поэтому с его помощью можно закодировать не 256, а 65536 различных символов. Полная спецификация стандарта Unicode охватывает символы различных письменностей **–** латинской, кириллической, греческой, а также языков, использующих иероглифы, например, китайского и японского.

Каждый из символов, который содержится на клавиатуре, имеет свой код ASCII. Специальные «управляющие» символы, такие как табуляция, перевод строки и возврат каретки, имеют коды с 0 по 31, базовые символы клавиатуры имеют коды с 32 по 127. Например, буква a (латинская) в нижнем регистре соответствует ASCII-коду 97, а буква A (тоже латинская) в верхнем регистре соответствует ASCII-коду 65. В результате VB при сортировке или при выполнении других операций считает эти два символа различными. Коды с 128 по 255 в таблице ASCII содержат символы национальных языков, греческие символы, символы псевдографики и некоторые другие.

Для работы с символами в языке VB имеется ряд встроенных функций. Приведем ряд примеров.

Чтобы определить ASCII-код конкретной буквы, можно использовать функцию языка VB **Asc()**. Например, следующий оператор присваивает целочисленной переменной a число 122 (ASCII-код строчной буквы z):

|  |
| --- |
| **Dim a As Short**  **a = Asc("z")** |

С помощью функции **Chr()** можно преобразовать ASCII-код в символ. Например, следующий код присваивает символьной переменной символ **z**:

|  |
| --- |
| **Dim l As Char**  **l = Chr(122)** |

Считается, что символ больше, чем другой символ, если его ASCII-код больше. Например, ASCII-значение буквы **B** больше, чем ASCII-значение буквы **А**, так что выражение **"А" < "В"** истинно, а выражение **"А" > "В"** – ложно.

### 4.9.2. Объявление и инициализация строк

Строковые переменные являются данными типа **String** и должны быть объявлены следующим образом:

|  |
| --- |
| **Dim** *ИмяПеременной* **As String** |

Присвоение значений переменным строкового типа, как и переменным других типов, производится либо при объявлении переменной (при инициализации), либо оператором присваивания:

|  |
| --- |
| **Dim** *ИмяПеременной* **As String = "строка символов"**  *ИмяПеременной = СтроковоеВыражение* |

Где, *СтроковоеВыражение* представляет собой:

строковый литерал – последовательность символов, заключенных в двойные кавычки;

строковую именованную константу либо строковую переменную;

результат операции с участием строковых выражений;

результат любого допустимого выражения, преобразованного в строковый тип.

Строковая именованная константа должна быть обязательно инициализирована при объявлении:

|  |
| --- |
| Const name = "ГОУ ""МТУСИ""" |

Обратите внимание, что если в строку нужно включить двойную кавычку, то после нее ставят еще одну двойную кавычку.

Как известно, во избежание неконтролируемых неявных преобразований типов всегда необходимо использовать режим строгого контроля типов, включаемый оператором **Option Strict On**. Причем для явного преобразования любого числового типа в строку можно использовать функцию **CStr()** или метод **ToString()** класса **Convert**.

|  |
| --- |
| Dim chislo As Double  chislo = 3.14  Dim str1 As String = Convert.ToString(chislo)  Dim str2 As String = CStr(chislo) |

Напомним также, что обратное преобразование из строки в число можно выполнить с помощью функции **Val(), CDbl(), CInt()** и т.д. или с помощью соответствующих методов класса **Convert:Convert.ToDouble(), Convert.ToInt32( )** и т.д.

Например:

|  |
| --- |
| **Dim chislo As Integer = Convert.ToInt32(TextBox1.Text)**  **Dim dlina As Double = Convert.ToDouble(TextBox2.Text)**  **Dim v As Single = CSng(TextBox3.Text)** |

В **Теме 2** настоящего пособия были подробно изложены средства явных преобразований. Еще раз обращаем внимание на то, что в числовой переменной разделителем целой и дробной частей является точка, а в строке – запятая, в противном случае произойдет ошибка. Это относится ко всем вышеперечисленным функциям и методам, кроме функции **Val()**, которая в качестве разделителя целой и дробной частей принимает только точку. Во избежание ошибок необходимо перед преобразованием строки в число проверять разделитель целой и дробной частей.

### 4.9.3. Средства для ввода/вывода и отображения строк

Простейшим способом отображения в программе текстовых строк, вводимых с клавиатуры, а, следовательно, и из текстового файла, яв­ляется использование объекта текстового поля. Как известно, объекты текстового поля могут иметь любые размеры. Если содержимое текстовой информации не помещается в текстовое поле, к текстовому полю можно добавить полосы прокрутки так, чтобы пользователь смог увидеть весь текст.

Для ввода строковых данных и отображения их на форме могут использоваться известные элементы управления **TextBox**, **ListBox, ComboBox**. Причем для отображения нескольких строк в текстовом поле свойство Multilineустанавливается в значении *True****.*** Для отображения строковой информации на форме, не подлежащей редактированию пользователем, используется элемент управления **Label**. Отображать текстовую информацию в качестве заголовков можно также на кнопках, в заголовке окна формы и т.д. Для отображения используется свойство Text этих элементов. Например,

|  |
| --- |
| **Label1.Text="Введите место рождения"** |

Для того чтобы текст оставался до определенного времени невидимым, его свойству Visible первоначально надо присвоить значение *False***,** а затем, когда нужно это поле отобразить, свойству Visibleприсваивается значение *True***.**

Напомним, что типы данных в VB являются классами и управляются из пространства имен **System**, а классы, в свою очередь, состоят из ***методов*** и ***свойств***.

Значение свойства Text всех элементов управления имеет тип **String**. Поэтому для записи числового значения в свойство Textэлемента управления необходимо произвести явное преобразование в тип **String**, например методом **ToString** соответствующего типа данных или соответствующей встроенной функцией **CStr()**:

|  |
| --- |
| **Dim ЧислоДней As Integer = 31**  **Label1.Tex t = ЧислоДней.ToString**  **Label2.Text = CStr(ЧислоДней)** |

Здесь переменная **ЧислоДней** обрабатывается методом **ToString(),** который принадлежит классу **Integer** или функцией **CStr()**. Для явного преобразования можно также воспользоваться методом класса **Convert**:

|  |
| --- |
| **Label1.Text = Convert.ToString(ЧислоДней)** |

При включенном режиме строгого контроля типов **Option Strict On** отсутствие явного преобразования вызовет ошибку.

Значение, вводимое из поля Textэлементов **TextBox, ListBox,   
ComboBox**в переменные числовых типов, также надо преобразовывать из строки в число. Для этого существуют соответствующие функции VBи методы класса **Convert**:

|  |
| --- |
| **Dim ЧислоДней As Integer = Convert.ToInt32(TextBox1.Text)**  **Dim Длина As Double = Convert.ToDouble(TextBox2.Text)**  **Dim Скорость As Single = CSng(TextBox3.Text)** |

### 4.9.4. Средства Visual Basic

### и Framework.NET для обработки строк

***Средства языка VB для работы со строками.*** В VBсуществует ***операция*** для работы со строками – ***конкатенация*** (**объединение строк**). Она позволяет объединить несколько строк в одну:

|  |
| --- |
| *ТекстоваяПеременная = Строка1 & Строка2 & …* |

*Строка1, Строка2…* – это строковые выражения, состоящие из строковых переменных, строковых констант, строковых литералов и свойств объектов Text**.**

Например:

|  |
| --- |
| **Label1.Text = "Cтудент " & TextBox1.Text & " сдал " & \_**  **СданоЛабораторок(ИндексСтудента) & " лабораторных работ"** |

При сравнении двух строк, которые содержат больше одного символа, VB начинает сравнивать первый символ первой строки с первым символом второй строки, а затем продолжает посимвольное сравнение этих строк до тех пор, пока не найдет различие.

Например, строки **Mike** и **Michael** одинаковы до третьего символа **("k"** и **"с"**). Так как значение ASCII "**k**" больше, чем значение "с", выражение   
**"Mike" > "Michael"** истинно.

Если между строками не найдено различий, то они равны. Если две строки равны на протяжении нескольких символов, но одна строка продолжается, а вторая заканчивается, то более длинная строка больше, чем более короткая. Например, выражение **"ААААА" > "ААА"** истинно.

***Средства работы со строками библиотек Framework.NET класса* String.** Как известно**,** VB является полностью объектно-ориентированным языком, который использует библиотеки классов .NET Framework**.** Это в частности сказывается в том, что строки ***являются не базовыми типами, подобными числовым типам, а объектами класса* String*.*** Этот класс является общим для всех языков платформы .NET Framework. Обращение к строкам производится по тем же правилам, что и к свойствам и методам элементов управления. Например, **TextBox1.Text** возвращает свойство Text объекта **TextBox1,** а метод **ListBox1.Items.Add(**"**Февраль**") добавляет строку "**Февраль**" в список **ListBox1**.

Обращение к свойствам и методам объекта производится по правилу точечной нотации:

|  |
| --- |
| *ИмяОбъекта.ИмяСвойства*  *ИмяОбъекта.ИмяМетода(СписокФактическихПараметров )* |

К наиболее частым действиям со строками, кроме конкатенации, можно отнести следующие действия:

* выделение части строки;
* поиск в строке по образцу;
* замена по образцу;
* изменение регистра символов (большие – маленькие буквы);
* удаление лишних пробелов;
* добавление выравнивающих полей.

Для выполнения вышеперечисленных действий в библиотеки .NET Framework существует большое число методов. Рассмотрим некоторые из них более подробно.

***Выделение части строки***. Метод **Substring()** возвращает часть строки заданной длины, начиная с указанной позиции в строке.

|  |
| --- |
| *ИмяОбъекта.***Substring***(ИндексСтартСимСтроке, КолСимволов)* |

В примере выделяется один символ строки, начиная с пятого:

|  |
| --- |
| **Dim S As String = "Крокодил"**  **Dim ПятыйСимволСтроки As String = S.Substring(4,1)**  *'ПятыйСимволСтроки="о"* |

Если количество возвращаемых символов не указано, возвращается часть строки от стартового символа до конца строки.

Для выделения последних символов строки необходимо знать размер строки, который возвращает свойство **Length**.

|  |
| --- |
| **Dim S As String = "Крокодил"**  **Dim L As Integer = S.Length** *'L=8* |

Существует также метод **Right( )**, возвращающий последние символы. Для использования этого метода размер строки знать не нужно:

|  |
| --- |
| **Dim ПослДвеБук As String = Strings.Right(S,2)***'ПослДвеБук = "ил"* |

Необходимо обратить внимание на синтаксис обращения к этому методу. Здесь строковый объект не указывается перед точкой, а является параметром обращения к методу; это – так называемый ***статический метод* класса.**

***Поиск в строке***. Часто нужно искать в строке положение разделителя слов, например, пробела между именем и отчеством. Для поиска заданного символа используется метод **IndexOf( )**. Этот метод возвращает индекс первого вхождения символа в строку:

|  |
| --- |
| **Dim ФИО As String = "Сидоров Иван Петрович"**  **Dim ПервыйПробел As Integer = ФИО.IndexOf(" ")** *'ПервыйПробел=7*  **Dim P As Integer = ФИО.IndexOf("о")** *'P=3* |

Если нужно найти последнее вхождение символа в строке, используется метод **LastIndexOf()**.

|  |
| --- |
| **Dim ПосПроб As Integer = ФИО.LastIndexOf(" ")*'ПосПроб=12***  **P = ФИО.LastIndexOf("о") *'P=17*** |

А что делать, если пробелов или искомых подстрок в строке несколько и требуется разобрать текст по словам? В этом случае используется более общая модификация метода **IndexOf()**.

|  |
| --- |
| **Строка.IndexOf(Символ, СтартовыйИндексПоиска)** |

Например:

|  |
| --- |
| **СледИнПро=ФИО.IndexOf(" ", ПредыдущийИндексПробела)**  **P = ФИО.IndexOf("о",4)** *'P=5*  **P = ФИО.IndexOf("б",4)** *'P=-1 символ "б" не найден* |

Если символ не найден, возвращается значение -1.

Методы, имеющие несколько вариантов списков параметров обращения, называются ***перегруженными методами* (п.3.5)**.

Для полного разбора строки на слова существует метод, превращающий строку в массив слов или других подстрок, разделённых запятыми или какими-то другими разделителями. Синтаксис обращения, следующий:

|  |
| --- |
| *Строка***.Split**(*СимволРазделитель*) |

Метод **Split()** применяется для разбивки исходной строки на подстроки и сохраняет эти подстроки в массиве. Метод **Split()** разделяет строку по границам, определяемым одиночными символами.

Ниже приведен пример разбора на отдельные слова строки, состоящей из нескольких слов:

|  |
| --- |
| **Dim ФИО As String = "Иванов Иван Иванович"**  **Dim Mas ( ) As String = ФИО.Split(" ")**  **Label1.Text = "Фамилия – " & Mas(0)** *'Mas (0) = "Иванов"*  **Label2.Text = "Имя – " & Mas(1)** *'Mas (1) = "Иван"*  **Label3.Text = "Отчество – " & Mas(2)***'Mas (2) = "Иванович"* |

Существует обратный метод **Join()**, который возвращают строку, получаемую объединением нескольких строк, содержащихся в указанном массиве.

|  |
| --- |
| **String.Join**(*СимволРазделитель, ОдномерныйМассив*) *' для метода* |

Метод в качестве символа разделителя может использовать как одиночный символ, так и строку. Если в качестве разделителя задана строка нулевой длины, то все подстроки массива объединяются в одну строку без разделения.

Кроме того, существуют методы, проверяющие окончание строки – **EndsWith( )** и начало строки – **StartsWith( )**:

|  |
| --- |
| **If TextBox1.Text.EndsWith("ич") Then**  **Обращение = "Дорогой "**  **ElseIf TextBox1.Text.EndsWith("на")**  **Обращение = "Дорогая "**  **Else**  **Обращение = "Товарищ "**  **End If** |

***Замена символов***. Для замены символов используется метод   
**Replace()** c указанием заменяемых и заменяющих символов:

|  |
| --- |
| *Строка***.Replace**(*ИсходнаяПодстрока, НоваяПодстрока*) |

Этот метод возвращает новую подстроку, созданную на основе исходной строки путём указанной замены. Например,

|  |
| --- |
| **Dim Был As String = "Дорогой "**  **Dim Станет As String = Был.Replace("ой", "ая" )** |

Если возвращаемое значение присваивается исходной строке, всё равно образуется новая строка, а старая с этим именем становится недоступной и автоматически удаляется «***службой уборки мусора***», встроенной в .NET Framework.

***Удаление символов*.** Для удаления символов используется метод   
**Remove()**. Он удаляет заданное количество символов, начиная с заданного.

|  |
| --- |
| *Строка*.**Remove**(*НачальнаяПозиция, КоличествоУдаляемыхСимволов*) |

|  |
| --- |
| **Dim Оценка As String = "неудовл."**  **Оценка = Оценка.Remove(0,2)** |

***Изменение регистра символов***. Используемые для этого методы возвращают копию строки, преобразованную к указанному регистру. Метод **ToUpper()** создаёт строку, преобразованную к верхнему регистру, метод **ToLower()** – к нижнему регистру. Параметров эти методы не требуют.

|  |
| --- |
| **Dim ВсёБолБуквами As String = ИсходнаяСтрока.ToUpper()**  **Dim ВсёМалБуквами As String = ИсходнаяСтрока.ToLower()** |

***Удаление лишних пробелов****.* Пробелы в начале или конце строки могут появляться из-за привычки части пользователей нажимать на <Space> где надо и не надо. Лишние пробелы влияют на результат операции сравнения. Метод **TrimStart()** удаляет пробелы в начале строки, перед первым непробельным символом, **TrimEnd()** удаляет пробелы в конце строки, а **Trim()** удаляет пробелы и в конце, и в начале строки.

***Средства работы со строками библиотек Framework.NET класса* StringBuilder*.*** Любой объект класса **String** является неизменяемым, и его значение никогда не изменяется после создания строки. Это означает, что при каждом примене­нии метода, который изменяет значение строки, в действительности создается но­вый объект типа **String**. Например, выражение: **S = S.Insert(3, "1234")** не изменяет исходную строку в памяти. Вместо этого метод **Insert()** создает новый объект типа **String**, который затем присваивается объектной переменной **S**. Исход­ный строковый объект в памяти, в конце концов, будет уничтожен вовремя следу­ющей сборки мусора, если только на него не указывает другая переменная. Схема выделения памяти **.**NET обеспечивает, что этот механизм добавляет относительно небольшие непроизводительные издержки, тем не менее, слишком большое коли­чество операций выделения и освобождения памяти можетухудшить эффектив­ность приложения. Объект типа **Text.StringBuilder** предлагает решение этой про­блемы. Причем при использовании объектов класса **StringBuilder** необходимо подключить библиотеку **System.Text (Imports System.Text)**.

Объект типа **System.Text.StringBuilder** можно представить в виде буфера, который содержит строку, способную увеличиваться в размерах от нуля символов и до теку­щей вместимости этого буфера. До тех пор, пока эта вместимость не превышена, строка строится в буфере, и память не выделяется и не освобождается. Если строка становится больше текущей вместимости буфера, объект класса **StringBuilder** создает буфер большего размера. Используемый по умолчанию буфер изначально содержит 16 символов, но это значение можно изменить, присвоив другое значение вместимости в конструкторе **StringBuilder()** или присвоив новое значение свойству Capacit*y*.

|  |
| --- |
| *'Создание объекта StringBuilder*  *'с начальной вместимостью 1000 символов*  **Dim sb As New StringBuilder(1000)** |

Строку, хранящуюся в объекте **StringBuilder**, можно обрабатывать с помощью нескольких методов, большая часть которых имеет такое же имя и работает так же, как методы, определенные в классе **String.** Например, методы **Insert(), Remove()** и **Replace()**. Чаще всего строку внутри объекта **StringBuilder** строят с помощью метода этого объекта **Append()**, который принимает аргумент любого типа и добавляет его к текущей внутренней строке:

|  |
| --- |
| *'Создание списка первых 100 целых чисел, разделенных запятыми*  **For n As Integer = 1 То 100**  *'Обратите внимание на то, что два метода Append исполняются быстрее,*  *'чем один метод Append, аргумент которого–конкатенация n истроки ","*  **sb.Append(n) : sb.Append(",")**  **Next n**  **sb.Insert(0, "Список чисел:")** *'Вставляет строку в начало буфера*  **TextBox1.Text = sb.ToString()** *' Список чисел: 1,2,3,4,5,6,...*  **TextBox2.Text = CStr(sb.Length)** *' Длина строки равна 309* |

Свойство **Length** возвращает текущую длину внутренней строки.

Также имеется метод **AppendFormat()**, который позволяет задать строку форматирования во многом так же, как и метод **String.Format()**, и метод **AppendLine()**, который добавляет в буфер обмена **StringBuilder** строку и используемый по умолчанию символ конца строки:

|  |
| --- |
| **For n As Integer = 1 То 100**  **sb.AppendLine(CStr(n))**  **Next n** |

В результате этого фрагмента программы получится объект-строка из чисел, причем каждое число будет записываться с новой строки. Так как технически содержимое объекта **StringBuilder** является не строкой, а символьным буфером, то в нужный момент (например, для вывода результата или присвоения) этот объект необходимо преобразовать в строку методом **ToString()**.

**Пример 4.9.4-1. Создать проект, который вводит с клавиатуры строки a и b, отображает их на форме, а затем обрабатывает, удаляя из строки a все символы, входящие в строку b и расположенные на нечетных позициях строки b, результат обработки необходимо отобразить на форме.**

Форма интерфейса пользователя приведена на рис. 4.9.4-1.

Схема алгоритма обработки представлена на рис. 4.9-4-2.

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | |  |

*Рис. 4.9.4-1. Форма проекта* ***Пример 4.9.4-1*** *Рис. 4.9.4-2.Схема алгоритма*

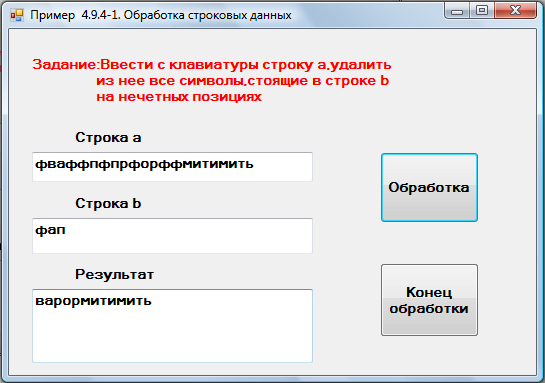
*процедуры* **Обработка()**

**Примера 4.9.4-1**

Программный код проекта представлен на рис. 4.9.4-3. Он содержит процедуры **vvod(), vivod()** и **Обработка()**.

|  |
| --- |
| **Public Class Form1**  *'Процедура вывода результата в TextBox*  **Sub vivod(ByVal z As String, ByRef t As TextBox)**  **If z.Length <> 0 Then**  **t.Text = z**  **Else**  **t.Text = "Все удалено"**  **End If**  **End Sub**    *'Функция ввода исходных данных из TextBox*  **Function vvod(ByVal T As TextBox) As String**  **Return T.Text**  **End Function**    **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) Handles Button1.Click**  **Dim Str1, Str2 As String**  **Str1 = vvod(TextBox1) : Str2 = vvod(TextBox2)**  **Обработка(Str2, Str1) : vivod(Str1, TextBox3)**  **End Sub**  **Private Sub Button2\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) Handles Button2.Click**  **End**  **End Sub**  *'Процедура решения задачи удаления символов из строки*  **Sub Обработка(ByVal b As String, ByRef a As String)**  **Dim d, m, i, j As Short**  **m = CShort(a.Length)** *'длина строки a*  **d = CShort(b.Length)** *'длина строки b*  **For i = 0 To CShort(d - 1) Step 2**  *'перебор нечетных симв. строки b*  **j = 0**  **Do While j < m** *'перебор всех символов строки a*  **If a.Substring(j, 1) = b.Substring(i, 1) Then**  **a = a.Remove(j, 1)***'пока симв.совпад. удаляем из a*  **m = CShort(m - 1)** *'уменьшаем длину строки a*  **Else**  **j = CShort(j + 1)** *'переход к след.символу строки a*  **End If**  **Loop**  **Next i**  **End Sub**  **End Class** |

*Рис. 4.9.4-3. Программный код проекта* **Пример 9.4-1**



*Рис. 4.9.4-5. Результаты выполнения проекта* **Пример 9.4-1**

### 

### 4.9.5 Базовые алгоритмы обработки строк и примеры их программирования

Функции и методы для работы со строками помогли познакомится с возможностями обработки строк и символов в VB. При решении задач следует научиться находить необходимые функции, понимая их предназначение, а также использовать в совокупности простые приемы обработки строк (назовем их базовыми алгоритмами), среди которых можно выделить следующие:

Определение количества символов в строке при заданных условиях (Пример 4.9.5-1).

Замена или вставка символов в строке (Пример 4.9.5-2).

Подсчет количества фрагментов (Пример 4.9.5-3).

Формирование строки из слов при заданных условиях исходной строки (Пример 4.9.5-4).

Данные базовые алгоритмы можно использовать не только для решения учебных задач, но и для разработки текстовых редакторов, программного обеспечения, предназначенного для верстки текста, и т.д.

**Пример 4.9.5-1. Разработать процедуру-Function, которая подсчитывает, сколько раз заданный символ встречается в исходной строке.**

Программный код приведен на рис. 4.9.5-1.

|  |
| --- |
| **Function Pr951(ByVal row As String, \_**  **ByVal simvol As String) As Integer**  **Dim kol, i As Integer**  **Dim tmp As String**  **kol = 0**  **For i = 0 To row.Length - 1**  **tmp = row.Substring(i,1)**  **If tmp = simvol Then**  **kol = kol + 1**  **End If**  **Next**  **Return kol**  **End Function**  **Private Sub Button1\_Click(…)**  **Dim r As String**  **Dim s As Char**  **s = CChar(InputBox("Введите символ")) : r = TextBox1.Text**  **Label2.Text = "Кол.найденных символов в строке=" & Pr951(r, s)**  **End Sub** |

*Рис. 4.9.5-1 Программный код функции* **Pr941()**

Примера 4.9.5-1

Параметрами функции **Pr951()** являются строка **row** и переменная **simvol**. В цикле происходит выделение символа с помощью метода **Substring()**. Параметр **row** – наша исходная строка, **i** – параметр цикла, который будет изменяться в цикле от нуля (начало строки) до конца строки – за это отвечает метод **Length**. Cчетчик **kol = kol + 1** обеспечивает подсчет количества заданного символа.

**Пример 4.9.5-2. Разработать процедуру-Function, которая заменяет пробелы в исходной строке заданным символом.**

Параметрами функции **Pr952()** являются строка **row** и переменная **sim**. В методе **Replace()** используем в качестве параметров исходную строку **row**, в качестве заменяемой подстроки используем пробел и подстроку, на которую заменяется подстрока – **sim**. Цель применения операции логического отрицания **Not** очевидна, т.к. оператор **row = row.Replace(" ", sim)** будет выполняться тогда, когда логическое выражение будет иметь значение **True**. Программный код приведен на рис. 4.9.5-2.

|  |
| --- |
| **Function Pr952(ByVal row As String, ByVal simvol As Char) As String**  **Dim zamena As Boolean**  **If Not zamena Then**  **row = row.Replace(" ", simvol)**  **Else**  **row = row.Replace(simvol, " ")**  **End If**  **zamena = Not zamena**  **Return row**  **End Function**  **Private Sub Button1\_Click(…)**  **Dim r As String, s As Char**  **r = TextBox1.Text**  **s = CChar(InputBox("Введите символ"))**  **Label2.Text = "Результирующая строка: " & Pr952(r, s)**  **End Sub** |

*Рис. 4.9.5-2. Программный код процедуры* **Pr952() Примера 4.9.5-2**

Пример 4.9.5-3. Разработать процедуру-Function, которая определяет, сколько раз подстрока **row\_1** входит в исходную строку **row**.

Функция **Pr953()** использует метод **IndexOf()**, который возвращает позицию подстроки **row\_1** в исходной строке **row**, при этом счетчик будет увеличиваться на единицу при каждом нахождении подстроки. В противном случае произойдет досрочный выход из цикла с помощью оператора **Exit Do**.

Программный код приведен на рис. 4.9.5-3.

|  |
| --- |
| **Function Pr953(ByVal row As String, ByVal row\_1 As String) \_**  **As Integer**  **Dim i, kol, x As Integer**  **i = 0**  **Do**  **x = row.IndexOf(row\_1, i))**  **If x >= 0 Then**  **i = x + row\_1.Length**  **kol = kol + 1**  **Else**  **Exit Do**  **End If**  **Loop**  **Return kol**  **End Function**  **Private Sub Button1\_Click(…)**  **Dim r, r\_1 As String**  **r = TextBox1.Text**  **r\_1 = InputBox("Введите подстроку")**  **Label2.Text = "Всего вхождений подстроки" & \_**  **r\_1 & "в строку " & Pr953(r, r\_1)**  **End Sub** |

*Рис. 4.9.5-3. Программный код процедуры* **Pr953()**

**Примера 4.9.5-3**

**Пример 4.9.5-4. Разработать процедуру-Function, которая находит в заданной строке все слова, состоящие только из цифр, и записывает найденные слова в новую строку через пробел. Если таких слов нет, то результатом процедуры должна быть строка с сообщением.**

**Заданная произвольная строка символов, в которой группу символов, разделенную с одной или с обеих сторон одним или несколькими пробелами и не содержащую внутри себя пробелов, называют словом.**

|  |
| --- |
| **Sub Pr954(ByRef s As String)**  **s = s.Trim**  *'цикл замены двух пробелов одним*  **Do While s.IndexOf(Space(2)) >= 0**  **s = s.Replace(Space(2), Space(1))**  **Loop**  **End Sub**  **Function Reshenie(ByVal s As String) As String**  **Dim Rez As String = ""**  **Dim i, j, d, k As Integer**  **Dim slovo As String**  **Dim mas() As String = s.Split(CChar(Space(1)))**  *'разбиение строку на массив слов*  **For i = 0 To mas.Length - 1**  **slovo = mas(i)**  **k = 0** *'количество цифр в слове*  **d = slovo.Length** *'длина слова*  **For j = 0 To d - 1** *'цикл по одному слову*  **If slovo.Substring(j, 1) >= "0" And \_**  **slovo.Substring(j, 1) <= "9" Then**  **k = k + 1**  **End If**  **Next**  **If k = d Then**  **Rez = Rez + slovo + Space(1)**  **End If**  **Next**  **If Rez = "" Then**  **Rez = "Таких слов нет"**  **End If**  **Return Rez**  **End Function**  Рис. 4.9.5-4. Программный код процедур **Pr954()** и **Reshenie()**  **Пример 4.9.4-4** |

Решение задачи реализует процедура-функция **Reshenie().** В ней после вызова процедуры **Pr954()** создается строковый массив слов **mas()** с помощью метода **Split()**, которая разбивает строку на массив по признаку одного пробела.

Далее в цикле по всем элементам массива рассматривается каждое слово и во внутреннем цикле каждый символ рассматриваемого слова с помощью метода **Substring()** сравнивается с символами-цифрами и подсчитывается количество цифр в слове. Если это количество цифр в слове равно длине слова, значит рассматриваемое слово состоит только из цифр, и, следовательно, оно добавляется в новую результатирующую строку **Rez**. Если после выполнения внешнего цикла результирующая строка **Rez** пуста (т.е. не содержит слов), то в нее записывается сообщение **"Таких слов нет".**

### 4.9.6. Тестовые задания

Символьным (строковым) выражением является

последовательность букв русского или латинского алфавита, цифр или других символов, взятая в кавычки

последовательность букв русского или латинского алфавита

последовательность букв русского алфавита

последовательность букв русского или латинского алфавита, цифр или других символов, начинающаяся с латинской буквы и взятая в кавычки

Ошибочной записью символьной переменной является

с$

b

c1

нет верного ответа

Подсчитать количество символов в символьном выражении позволяет функция

Len( )

Left( )

Asc( )

Mid ( )

Операции, допустимые для строковых данных

>, < , = , < = , > = , <> , + , &

+ , = , &

=

нет верного ответа

Метод a.Substring(0, n)

возвращает n первых символов строки a

возвращает n последних символов строки a

не применяется к строкам

находит символ, стоящий на позиции n в строке a

ничего не делает

Метод a.Substring(n, m)

вырезает из строки a , начиная с n-ого, m символов

вырезает из строки a , начиная с m-ого , n cимволов,

не применяется к строкам

нет верного ответа

Функция Space(n)

возвращает строку из n пробелов

возвращает строку из n первых символов

возвращает строку из n последних символов

удаляет n начальных пробелов

Метод a.ToUpper

заменяет строчные буквы строки a на прописные

заменяет прописные буквы на строчные

удаляет начальные пробелы строки

удаляет конечные пробелы строки

Метод a.ToLower

заменяет прописные буквы строки aна строчные

заменяет строчные буквы на прописные

удаляет начальные пробелы строкиa

удаляет конечные пробелы строки a

Метод a.TrimStart

удаляет начальные пробелы строки a

заменяет прописные буквы строки a на строчные

заменяет строчные буквы строки a на прописные

удаляет конечные пробелы строки

нет верного ответа

Метод a.TrimEnd

удаляет конечные пробелы строки

заменяет прописные буквы на строчные

заменяет строчные буквы на прописные

удаляет начальные пробелы строки

нет верного ответа

Метод a.Indexof(b, n)

начиная с n-й позиции строки a, определяет позицию вхождения в нее строки b

начиная с n-й позиции строки b, определяет позицию вхождения в нее строки a

заменяет n символов строки a на n символов строки b

Функция Chr(n)

возвращает символ, код ASCII которого равен числу n

возвращает строку из n пробелов

возвращает строку из n первых символов

возвращает строку из n последних символов

Функция Asc(a)

возвращает ASCII код первого символа строки a

возвращает количество символов в строке a

преобразует число в строку a

не применяется к строкам

Функция Val(a)

преобразует строку a в число, если это возможно

возвращает длину строки

определяет ASCII код строки

не применяется к строкам

Определить результат выполнения фрагмента программы

|  |
| --- |
| **Dim a As String = "стереопроигрыватель"**  **TextBox1.Text = a.Substring(0, 6)** |

стерео

O

нет верного ответа

проигрыватель

В результате выполнения фрагмента программы

|  |
| --- |
| **A ="КОЗАК"**  **a.Replace("О","А")**  **TextBox1.Text= a** |

вторая буква будет заменена на букву «А»

будет вырезана буква «А»

будет вырезана вторая буква слова

Функция Ucase (a)…

заменяет строчные буквы на прописные cтроки а

заменяет прописные буквы на строчные

удаляет начальные пробелы строки

удаляет конечные пробелы строки

Функция Ltrim (A)…

удаляет начальные пробелы строки A

заменяет прописные буквы на строчные

заменяет строчные буквы на прописные

удаляет конечные пробелы строки

нет верного ответа

Функция Rtrim (a)…

удаляет конечные пробелы строки а

заменяет прописные буквы на строчные

заменяет строчные буквы на прописные

удаляет начальные пробелы строки

## 4.9.7. Лабораторная работа по теме «Программирование алгоритмов обработки символьных и строковых данных»

**Цель работы:** изучить средства языка программирования при обработке строковых данных, приемы совместной обработки строковой и числовой информации, функции, методы и элементы управления, предназначенные для работы со строковыми и символьными данными.

### Вопросы, подлежащие изучению

1. Способы объявления строковых переменных.
2. Инициализация строк.
3. Классы и методы .NET Framework для работы со строками.
4. Методы преобразования строки в число и наоборот.
5. Конкатенация строк.
6. Элементы управления для работы со строками.

### Общее задание на разработку проекта

1. ***Изучите вопросы программирование алгоритмов обработки символьных и строковых данных*** *(Тема 9).*
2. ***Выберите вариант задания*** *из табл. 4.9.7-1.*
3. ***Разработайте графический интерфейс*** *пользователя.*
4. ***Разработайте схемы алгоритмов процедур*** *пользователя в соответствии с индивидуальным заданием, предварительно проведя формализацию.*
5. ***Напишите программный код процедур*** *в соответствии с разработанными схемами алгоритмов.*
6. ***Разработайте проект приложения****, решающий поставленную задачу, который состоит из интерфейса пользователя и соответствующего программного кода. Все пользовательские процедуры должны находиться в модуле формы. Обмен данными между пользовательскими процедурами должен осуществляться через параметры, без использования глобальных переменных.*
7. ***Выполните приложение и получите результат.***
8. ***Докажите правильность результата.***

### Варианты индивидуальных заданий

Таблица 4.9.7-1

|  |  |
| --- | --- |
| **№** | **Задача** |
| **1)** | В заданной строке определите наибольшее количество цифр, идущих подряд. |
| **2)** | Зашифруйте строку, содержащую русские буквы, пробелы и знаки препинания: замените каждую букву, непосредственно следующей за ней по алфавиту (например, "**я**" на "**а**"), строчную букву преобразуйте в прописную, прописную – в строчную. |
| **3)** | В заданной строке, состоящей из слов, разделенных одним или более пробелами, определите количество слов, начинающихся и заканчивающихся на одну и ту же букву. |
| **4)** | В заданной строке, состоящей из слов, разделенных одним или более пробелами, замените каждое третье слово «мама» словом «мамочка» и подсчитайте количество проведенных замен. |
| **5)** | Из заданной строки удалите все символы, входящие в нее более одного раза. |
| **6)** | Преобразуйте заданную строку, повторив в ней каждый символ, совпадающий с заданным. |
| **7)** | Определите, можно ли из символов заданной строки составить вашу фамилию. |
| **8)** | В заданной строке определите максимальную длину подстроки, состоящей из заданного символа. |
| **9)** | Исключите из заданной строки группы символов, расположенные между круглыми скобками, включая сами скобки. Предполагается, что внутри каждой пары скобок нет других скобок. |
| **10)** | В заданной строке, состоящей из слов, разделенных одним или несколькими пробелами, определить количество слов четной длины, в которых нет буквы «а». |
| **11)** | В заданной строке, состоящей из слов, разделенных одним или несколькими пробелами, определите количество слов, которые содержат ровно три буквы «а». |
| **12)** | В заданной строке, состоящей из слов, разделенных одним или несколькими пробелами, определите самое длинное слово нечетной длины. |
| **13)** | В заданной строке, состоящей из слов, разделенных одним или более пробелами, определите самое короткое слово четной длины. |
| **14)** | Строка содержит дату в формате "**mm-dd-yyyy**", где **mm** – месяц, **dd** – день, **yyyy** – год. Проверьте корректность задания даты и получите строку с датой в формате "**dd.mm.yy**", где **yy** – две последние цифры года. |
| **15)** | Из заданной строки, не содержащей цифр, получите "сжатую" строку, заменив самую длинную подстроку вида "**XXX…XX**" на подстроку вида "**nX**", где **X** – повторяющийся символ, **n** – число его повторений. |
| **16)** | Определите, является ли заданная строка записью целого числа, записью вещественного числа, или строку нельзя преобразовать в число. |
| **17)** | Из всех символов заданной строки получите новую строку, записывая каждый символ исходной строки только один раз. |
| **18)** | В заданной строке подсчитайте сумму цифр. |
| **19)** | В заданной строке, не содержащей букв русского алфавита, найдите процент гласных букв (**a, e, i, o, u, y**). |
| **20)** | Для заданной строки, состоящей из слов, разделенных одним или несколькими пробелами, определите, какое слово встречается раньше: самое короткое или самое длинное. |
| **21)** | В заданной строке, состоящей из слов, разделенных одним или более пробелами, удалите из каждого слова нечетной длины средний символ и подсчитайте их количество. |
| **22)** | В заданной строке, состоящей из слов, разделенных одним или более пробелами, подсчитайте количество симметричных слов, т.е. читающихся одинаково слева направо и справа налево. |
| **23)** | Определите, является ли заданная строка записью вещественного числа с нулевой дробной частью. |
| **24)** | Из строки, содержащей корректно заданную дату в формате "**dd.mm.yy**", где **dd** – день, **mm** – месяц, **yy** – год, получите строку с датой, отстоящей от заданной на **n** дней вперед, в том же формате. |
| **25)** | Определите, расположены ли символы заданной строки в алфавитном порядке. |
| **26)** | Определите, является ли заданная строка записью целого числа. |
| **27)** | Определите позицию **n**-ого с конца вхождения заданного символа в заданную строку символов. При отсутствии **n**-ого вхождения установите номер позиции равным-1. |
| **28)** | Строка состоит из "слов" длиной менее **10** символов, разделенных запятыми. Преобразуйте строку, заменив запятые группами пробелов так, чтобы каждое слово начиналось с позиции **10\*(k - 1) + 1**, где **k** – номер "слова". |
| **29** | В заданной строке, состоящей из слов, разделенных одним или более пробелами, вставьте в середину каждого слова четной длины букву «К» и найдите количество вставок. |
| **30)** | Из строки, содержащей корректно заданную дату в формате "**dd.mm.yy**", где **dd** – день, **mm** – месяц, **yy** – год, получите строку с датой, отстоящей от заданной на **n** дней назад, в том же формате. |

### Содержание отчёта

1. Тема и название лабораторной работы.
2. Фамилия, имя студента, номер группы, номер варианта.
3. Задание на разработку проекта.
4. Формализация и уточнение задания.
5. Формализация и уточнение задания.
6. графический интерфейс пользователя;
7. таблица свойств объектов;
8. схема алгоритма решаемой задачи;
9. программный код проекта.
10. Результаты выполнения проектов.
11. Доказательство правильности работы программы.

### Пример выполнения задания

1. **Тема и название лабораторной работы:**

**Обработка строковых данных:**

Совместная обработка строковой и числовой информации.

1. **Фамилия, имя студента, номер группы, номер варианта:**

Иванов И., БИН1405, вариант 13.

1. **Задание на разработку проекта:**

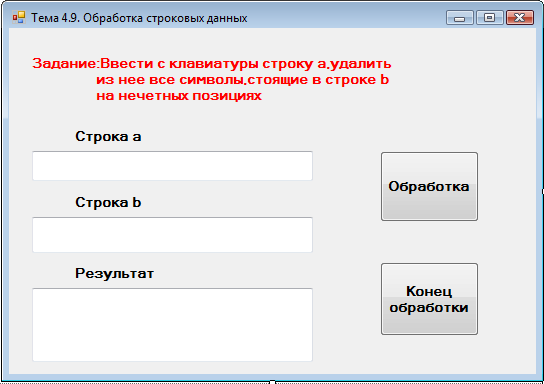
Создайте проект, который реализует задачу ввода с клавиатуры строк **a** и **b**, отображает их на форме, а затем обрабатывает, удаляя из строки **a** все символы, входящие в строку **b** и расположенные на нечетных позициях строки **b**. Результат обработки отобразите на форме.

1. **Формализация и уточнение задания:**

Формализация не требуется

1. **Элементы, разрабатываемого проекта:**
2. **Графический интерфейс пользователя:**

Форма приложения может иметь такой же вид, как на рис. 4.9.7-1.



*Рис. 4.9.7-1. Форма проекта* ***Проект 9***

1. **Таблица свойств объектов:**

Определите, установите и сведите свойства в табл. 4.9.7-2.

Таблица 4.9.7-2

|  |  |  |
| --- | --- | --- |
| **Имя** | **Свойство** | **Значение свойства** |
| **Form1** | Name | *Form1* |
| Text | *Тема 4.9: Обработка строковых данных* |
| **Label1** | Text | *Задание: Ввести с клавиатуры строку, а удалить из нее все символы, стоящие в строке b на нечетных позициях.* |
| ImageAlign | *MiddleCenter* |
| **Label2** | Text | *Строка a* |
| ImageAlign | *MiddleCenter* |
| **Label3** | Text | *Строка b* |
| **Label4** | Text | *Результат* |
| **TextBox1** | Name | *TextBox1* |
| **TextBox2** | Name | *TextBox2* |
| **TextBox3** | Name | *TextBox3* |
| **Button1** | Name | *Button1* |
| Text | *Обработка* |
| **Button2** | Name | *Button2* |
| Text | *Конец обработки* |

1. **Алгоритмы решения задачи:**

Схема алгоритма процедуры **Обработка()** представлена на   
 рис. 4.9.7-2.

|  |  |
| --- | --- |
|  |  |

*Рис. 4.9.7-2. Схема алгоритма процедуры* **Обработка()** *проекта* **Проект 9**

1. **Программный код проекта:**

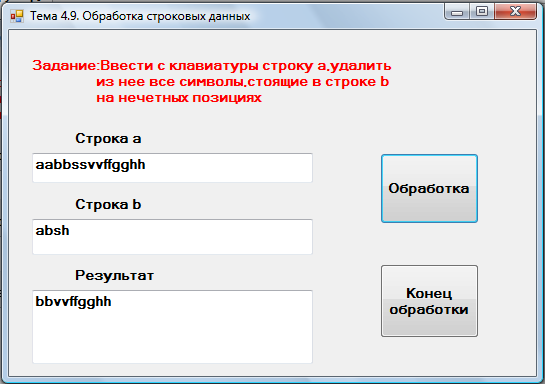
Код программы разработанной задачи приведен на рис. 4.9.7-3

|  |
| --- |
| **Public Class Form1**  *'Функция ввода исходных данных из TextB***ox**  **Function vvod(ByVal T As TextBox) As String**  **Return T.Text**  **End Function**  ***'Процедура вывода результата в TextBox***  **Sub vivod(ByVal z As String, ByRef t As TextBox)**  **If z.Length <> 0 Then**  **t.Text = z**  **Else**  **t.Text = "Все удалено"**  **End If**  **End Sub**  ***'Процедура решения задачи удаления символов из строки***  **Sub Обработка(ByVal b As String, ByRef a As String)**  **Dim d, m, i, j As Short**  **m = CShort(a.Length) 'длина строки a**  **d = CShort(b.Length) 'длина строки b**  **For i = 0 To CShort(d - 1) Step 2**  *'перебор нечетных символов строки b*  **j = 0**  **Do While j < m 'перебор всех символов строки a**  **If a.Substring(j, 1) = b.Substring(i, 1) Then**  **a = a.Remove(j, 1)**  *'пока символы совпадают–удаляем из строки a*  **m = CShort(m - 1)'уменьшаем длину строки a**  **Else**  **j = CShort(j + 1)**  *'переход к следующему символу строки a*  **End If**  **Loop**  **Next i**  **End Sub**    **Private Sub Button1\_Click(sender As Object, \_**  **e As EventArgs) Handles Button1.Click**  **Dim Str1, Str2 As String**  **Str1 = vvod(TextBox1) : Str2 = vvod(TextBox2)**  **Обработка(Str2, Str1)**  **vivod(Str1, TextBox3)**  **End Sub**  **Private Sub Button2\_Click(sender As Object, \_**  **e As EventArgs) Handles Button2.Click**  **End**  **End Sub**  **End Class** |

*Рис. 4.9.7-3. Программный код проекта* **Проект 9**

1. **Результаты выполнения проекта:**

Результаты выполнения проекта приведены на рис. 4.9.7-4.



*Рис. 4.9.7-4.**Результаты выполнения проекта* **Проект 9**

1. **Доказательство правильности работы программы:**

Дана исходная строка **a="aabbssvvffgghh"**. В строке **b="absh"** на нечетных местах стоят символы **"a"** и **"s"**. После их удаления строка **а** должна иметь вид: **"bbvvffgghh".**

Полученный вид строки полностью совпадает с результатом, полученным с использованием программы.

### Контрольные вопросы

1. Как в программах используются переменные строкового типа?
2. Какие возможности предоставляют методы и функции класса **Sysem.String**?
3. Как объявляются переменные строкового типа?
4. Какими способами можно присвоить значение переменной строкового типа?
5. Понятие "преобразование типов". Что такое явное преобразование типов? Что такое неявное преобразование типов?
6. Зачем и как устанавливается режим строгой проверки типов?
7. Как преобразовать число в строку и обратно?
8. Для чего применяется конкатенация строк?
9. Какие способы конкатенации строк вы знаете?
10. Как определить количество символов в строке?
11. Какой индекс имеет первый символ строки?
12. Какие элементы управления используются для ввода и вывода строковой информации?
13. Как определить, имеется ли в строке, введённой пользователем, определённый символ и где он расположен в строке?
14. Как выделить определённую часть строки, введённой пользователем?
15. Как заменить символы в строке?
16. Как удалить символы в строке?
17. Как удалить лишние пробелы в строке?
18. Зачем и как преобразуется регистр строки?
19. Каким образом можно вставить подстроку в заданную строку?
20. Как и какими способами сравниваются строки?
21. Что делает метод **Trim( )**?
22. Что делает метод **Split( )**?
23. Что делает метод **Join( )**?
24. Что такое объект **StringBuilder**, и, какие возможности для работы со строками он предоставляет?
25. Какие классы .NET Framework используются для обработки строк?
26. Какие методы используются для обработки строк?
27. Приведите примеры методов структуры **Char**.
28. Приведите примеры методов класса **String**.
29. Поясните понятие «конструктор».
30. Напишите примеры конструкторов класса **String**.