



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Прикладная математика»

Программирование в Delphi: строковый тип данных

Методические указания к лабораторной работе № 5
по курсам «Информатика», «Алгоритмические языки
и программирование»

Автор
Е.Н. Ладоса, Д.С. Цымбалов, О.В. Яценко,
Е.В. Бочарова

Ростов-на-Дону, 2018

Аннотация

Описаны средства и способы обработки текстовой информации в Object Pascal. Целью работы ставится освоение студентами технологий представления и преобразования строковых данных в среде Delphi. Предназначены для студентов всех специальностей факультета «Информатика и вычислительная техника».

Автор

Доцент, к.т.н.
Ладоса Е.Н.

Старший преподаватель кафедры
«Электроника и электротехника»
Цымбалов Д.С.

Доцент, к.ф.-м.н.
Яценко О.В.

Студент ДГТУ
Бочарова Е.В.



Цель работы

Цели работы – изучить **строковые типы данных**. Отдельно рассмотрены строковые операторы и функции работы с ними.

Строка, строковые операторы и встроенные строковые функции

Строка – это набор символов произвольной длины, которая относится к структурированным типам данных. Строковые константы обозначаются апострофами. Ниже приведена таблица 1 типов строк в Delphi.

Таблица 1

Строковые типы в Object Pascal

Тип строки	Максимальная длина	Требования к памяти	Используется для	Нулевой
ShortString	255	от 2 байт до 256 байт	обратной совместности	нет
AnsiString	$\sim 2^{31}$ (~2Гб)	от 4 байт до 2 Гб	однобайтных, двухбайтных и многобайтных символов ANSI	есть
String	~2 Гб	от 2 байт до 2 Гб	символов ANSI или Unicode	есть или нет
WideString	$\sim 2^{30}$ (~1Гб)	от 4 байт до 2 Гб	символов Unicode, в серверах COM и интерфейсах	есть

В следующей строке кода переменной `today` присваивается значение Четверг:

```
today := 'Четверг'
```

Операция **конкатенации**, обозначаемая символом `+` – единственная строковая операция Object Pascal. Конкатенация объединяет строки в указанном порядке слева направо. Таким образом, оператор `+` может выполнять различные операции в зависимости от типа операндов. Такие операторы называются **перегружаемыми**. В следующем фрагменте кода выполняется конкатенация строк:

```
weekendDays := 'Суббота' + ' и ' + 'Воскресенье';
```

После выполнения этого кода переменная `weekendDays` содержит строку Суббота и Воскресенье.

В предыдущем примере, как вы заметили, апострофы не являются частью строки. Но что делать, если нужно использовать строку, содержащую апостро-

фы? Следующий фрагмент кода является неправильным:

```
weekendDays := 'Суббота и Воскресенье';
```

Обрабатывая эту строку кода, компилятор сгенерирует сообщение о **синтаксической ошибке**. Компилятор Delphi не сможет понять структуру этого оператора. Он видит, что за первым апострофом непосредственно следует второй, и считает это пустой строкой, т.е. строкой, не содержащей символов. Затем оператор встречает слово *Суббота* и не знает, Как его понять, поскольку строка уже закончилась. К счастью, существует несколько способов включения апострофа в состав строки. Внутри строки компилятор считает последовательность из двух апострофов не признаком окончания строки, а одним апострофом, входящим в состав строки. В следующем фрагменте кода переменной *weekendDays* присваивается значение 'Суббота и Воскресенье':

```
weekendDays := '''Суббота и Воскресенье''';
```

Первый и последний апострофы обозначают конец строки. Каждая пара апострофов внутри строки рассматривается компилятором как один апостроф.

Другой способ включения апострофов в состав строки – использование числовых значений символов. В компьютере с каждым символом ассоциировано числовое значение **ASCII** (American Standard Code for Information Interchange – Американский стандартный код для обмена информацией). В операционной системе Windows на экран выводятся не все символы ASCII, в ней используется более ограниченный набор символов **ANSI** (American National Standards Institute – Национальный институт стандартизации США). Числовые значения большинства символов ASCII и ANSI совпадают.

Для работы с символами ASCII в Object Pascal есть две встроенные функции:

- *Chr* (*n*) – возвращает символ, числовое значение ASCII которого равно *n*;
- *Ord* (*СИМВОЛ*) – возвращает значение ASCII символа *СИМВОЛ*.

Например, *Chr* (65) возвращает символ *A*, а *Ord* ('*A*') возвращает значение 65. Таким образом, числовое значение символа *A* равно 65.

Числовое значение ASCII символа "апостроф" равно 39, поэтому строку предыдущего примера можно записать в виде

```
weekendDays := Chr(39) + 'Суббота и Воскресенье' + Chr(39);
```

Этот оператор присваивания эквивалентен предыдущему: он присваивает переменной *weekendDays* значение 'Суббота и Воскресенье'.

Кроме функции *Chr* () и операции конкатенации, для включения в строку символа в Object Pascal можно воспользоваться **управляющей последовательностью**. Она состоит из символа #, за которым следует целое число в диапазоне от 0 до 255, обозначающее символ ASCII. Например, оператор



```
myString := #72#101#108#108#111;
```

эквивалентен оператору

```
myString := 'Hello';
```

Еще один пример: оператор

```
weekendDays := #39'Суббота и Воскресенье'#39;
```

эквивалентен и оператору

```
weekendDays := ''Суббота и Воскресенье'';
```

и оператору

```
weekendDays := Chr(39) + 'Суббота и Воскресенье' + Chr(39);
```

Обратите внимание: управляющая последовательность в составе строки не работает. Она управляет только работой компилятора. Например, если переменную `myString := 'Hel#108o'` вывести на экран, то мы увидим не `Hello`, а `Hel#108o`.

Язык Object Pascal поддерживает также типы данных и функции, необходимые для работы с **набором символов Unicode**, в котором каждый символ представлен двумя байтами. Строка Unicode состоит из последовательности двухбайтовых символов. Преимуществом набора Unicode является то, что он содержит символы всех языков мира. Это существенно облегчает создание приложений, которые будут использоваться в других странах. Символы и строки Unicode иногда называют **широкими символами** и **широкими строками**. Первые 256 символов Unicode совпадают с символами набора ANSI. При необходимости вы можете обратиться к справочной системе Delphi, в которой представлена полная информация о символах Unicode.

Для обработки строк Object Pascal предоставляет довольно много встроенных процедур, благодаря которым он является одним из лучших языков для работы со строками. В таблице 2 перечислены некоторые наиболее полезные встроенные строковые функции и процедуры.

Таблица 2

Встроенные строковые функции

Функция	Назначение
<code>CompareStr (const S1, S2: string): Integer;</code>	Сравнивает две строки с учетом регистра
<code>CompareText (const S1, S2: string): Integer;</code>	Сравнивает две строки по числовым значениям символов без учета регистра
<code>Concat (S1[, S2, ..., SN]: string): string;</code>	Конкатенирует две (или более) строки в одну
<code>Copy (St: String; Index, Count: Integer): string;</code>	Возвращает подстроку заданной строки
<code>IsDelimiter (const Delimiters, S: string; Index: Integer): Boolean;</code>	Определяет, является ли указанный символ строки символом-разделителем
<code>LastDelimiter (const Delimiters, S: string): Integer;</code>	Возвращает позицию последнего символа-разделителя в строке

Функция	Назначение
<code>Length(S) : Integer;</code>	Возвращает количество символов в строке
<code>LowerCase (const S: string): string;</code>	Возвращает копию строки, заменив все буквы верхнего регистра соответствующими буквами нижнего регистра
<code>Pos (SubSt, St: String): Integer;</code>	Возвращает позицию первого символа в подстроке, содержащегося в заданной строке
<code>QuotedStr (const S: string): string;</code>	Возвращает строку, выделенную апострофами (т.е. добавляет апострофы в конец и в начало строки)
<code>StringOfChar (Ch: Char; Count: Integer): string;</code>	Возвращает строку с указанным количеством повторяющихся символов
<code>StringReplace(const S, OldPattern, NewPattern: string; Flags: TReplaceFlags): string;</code>	Возвращает строку с заменой всех вхождений одной подстроки другой подстрокой
<code>Trim(const S: string): string;</code>	Удаляет из строки пробелы, расположенные в начале и в конце строки, и управляющие символы
<code>TrimLeft(const S: string): string;</code>	Удаляет из строки пробелы, расположенные в ее начале, и управляющие символы
<code>TrimRight(const S: string): string;</code>	Удаляет из строки пробелы, расположенные в ее конце, и управляющие символы
<code>Uppercase(const S: string): string;</code>	Возвращает копию строки с заменой всех букв нижнего регистра соответствующими буквами верхнего регистра
<code>WrapText(const Line, BreakStr: string; nBreakChars: TSysCharSet; MaxCol: Integer): string;</code>	Разбивает строку на несколько строк, имеющих указанный размер
Процедура	Назначение
<code>Delete (BufPos: longint; DelLen: longint)</code>	Удаляет подстроку из строки
<code>Insert (BufPos: longint; Text: string);</code>	Вставляет подстроку в строку, начиная с указанной позиции
<code>SetLength (var S; NewLength: Integer);</code>	Устанавливает длину строки
<code>SetString (var s: string; buffer: PChar; len: Integer)</code>	Устанавливает содержимое и длину строки
<code>Str(X [: Width [: Decimals]]; var S);</code>	Преобразует числовую переменную в строку
<code>Val(S; var V; var Code: Integer);</code>	Преобразует строку в числовую переменную

Как видно из таблицы 2, функция `QuotedStr()` предоставляет еще один способ размещения апострофов вокруг строки. С ее помощью фрагмент кода, присваивающий строке `weekendDays` значение 'Суббота и Воскресенье' можно записать так:



```
weekendDays := QuotedStr('Суббота и Воскресенье');
```

Работа со строками

Если заранее известно, что количество символов строковой переменной не будет превышать определенную величину, то эту переменную можно объявить как короткую строку. **Короткие строки** используются, если длина строки не превышает 255 символов. Синтаксис объявления короткой строки имеет вид

```
var  
    имя_строки:    String[n];
```

Обратите внимание: переменная *имя_строки* занимает $n+1$ байт памяти (от 0 до n), причем байты от 1 до n содержат символы строки, а нулевой байт – размер строки. Поскольку максимальное число, которое можно хранить в байте, равно 255, то ясно, что n должно быть меньше или равно 255. Тип данных `ShortString` эквивалентен типу `String[255]`. Типы `String[n]` и `ShortString` введены главным образом для совместимости с предыдущими версиями Delphi и Turbo Pascal. Приведем пример использования короткой строки:

```
var  
    myString: String[10];  
begin  
    myString := 'Это пример строки';  
    writeln(myString);  
    readln;  
end;
```

Если выполнить этот код, то в окне консольного приложения появится строка `Это пример`. Куда подевались остальные символы?

Тип данных `String` эквивалентен типу `AnsiString`. Оба они являются типами длинной строки, длина которой ограничена только объемом доступной памяти. Перепишем предыдущий пример с использованием **длинной строки**:

```
var  
    myString: String;  
begin  
    myString := 'Это пример строки';  
    writeln(myString);  
    readln;  
end;
```

При выполнении этого кода в консольном окне появляется строка `Это пример строки`.

Тип `String` был первым строковым типом данных в Turbo Pascal. Первоначально этот тип был реализован как короткая строка. В режиме компилятора Delphi `{H+}` или `{LONGSTRINGS on}`, установленном по умолчанию, тип данных `String` определен как эквивалентный типу `AnsiString`. Режим `{H-}`

или { \$LONGSTRINGS OFF} полезен при работе с кодами, унаследованными от старых версий Delphi или Turbo Pascal, в которых по умолчанию используются короткие строки. С помощью этой директивы компилятора можно также изменить определение типа String локально, только для старого фрагмента кода. При этом в остальном коде тип String будет определен как в режиме { \$H+}. Другой способ использования старого кода – изменить объявления String на String[255] или на ShortString, сделав таким образом код независимым от режима компилятора.

Строка фактически является **массивом** символов. Массив – это индексированный набор элементов одного и того же типа (который называется базовым типом). С каждым символом строки ассоциировано уникальное значение индекса. На рис. 1 показана структура строковой переменной как массива символов.

Предположим, переменной myString типа String присвоено значение
myString := 'Это пример строки';

Тогда структура массива myString имеет вид:

Индекс	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
myString	Э	т	о		п	р	и	м	е	р			с	т	р	о	к	и

Рис. 1. Внутренняя структура переменной myString

Обратиться к отдельному символу внутри строковой переменной можно с помощью выражения *имя_строки*[*i*], где *i* – индекс символа, на который вы хотите сослаться. Значение индекса первого символа строковой переменной равно 1, второго – 2 и т.д. Обратите внимание: выражение *имя_строки*[*i*] имеет символьный тип, поэтому его значение можно сохранить в любой переменной символьного или строкового типа. Кроме того, выражение *имя_строки*[*i*] можно использовать в операторе присваивания или в качестве параметра подпрограммы. Используйте его в следующем упражнении:

```

program Project2;
{$APPTYPE CONSOLE}
uses
    SysUtils;
var
    myString:    String;
    myChar:     String;
    aChar:      Char;
begin
    myString := 'This is an example';

    myChar := myString[13];
    myChar := Uppercase(myChar);
    aChar := myChar[1];
    myString[13] := aChar;
    writeln(myString);
    writeln('');
    writeln('Hit <Enter> to exit');
    readln;
end.

```


Какой текст появится в результате выполнения этого кода?

К сожалению, функции `UpperCase()` и `LowerCase()` выполняют преобразование только латинских букв набора ASCII. Русские буквы они оставляют неизменными.

Контрольные задания

1. Дана символьная строка `S`. Скопировать посимвольно ее в другую символьную строку.
2. Дана символьная строка `S`. Составить алгоритм для выделения подстроки в исходной строке. В качестве входных параметров используются: номер символа, с которого начинается подстрока в строке, и длина подстроки.
3. Дана символьная строка `S`. Подсчитать сколько раз в заданной строке встречаются символы 'а' и '1'.
4. Дана символьная строка `S`. Скопировать первые `N` символов из исходной строки в новую. `N` вводится с клавиатуры. Если `N` больше длины строки, то строка копируется целиком.
5. Дана символьная строка `S`. В этой строке все символы '7' и '!' заменить на символы '?'.
6. Даны две символьные строки `S1` и `S2`. Получить третью строку, как объединение двух строк следующим способом: если `S1="123"` и `S2="abc"`, то `S3="1a2b3c"`.
7. Дана символьная строка. Составить алгоритм, который будет осуществлять поиск введенной подстроки в исходной строке.

Контрольные вопросы

1. Как можно объявить величину строкового типа?
2. К каким типам данных относятся строки?
3. Какова максимально возможная длина строки?
4. С величиной какого типа данных совместим по присваиванию отдельный символ строки?
5. Расскажите об операциях, которые можно выполнять над строковыми величинами.
6. Расскажите о функциях, определенных для величин строкового типа.
7. Расскажите о процедурах, определенных для величин строкового типа.
8. Как осуществляется доступ к отдельному символу строки?
9. Что такое конкатенация строк?
10. Поясните применение управляющих последовательностей.
11. Что такое короткая строка и когда она применяется, чем отличие длинной строки от короткой?



12. Какая функция (процедура) является аналогом операции конкатенация строк (+) при работе со строками?

Задачи для самостоятельного выполнения

1. Написать программу, которая после ввода с клавиатуры числа (в диапазоне от 1 до 999), формировавала бы строковую переменную следующей формы, например 98 рублей или 31 рубль.
2. Написать программу, которая после ввода с клавиатуры фамилии, имени, отчества определяет пол.
3. С клавиатуры последовательно вводятся строки. Строки могут являться либо правильно записанными числами (не обязательно целыми и возможно со знаком), либо последовательностью букв. Нужно посчитать отдельно суммы целых и вещественных введенных чисел и сконкатинировать строки, состоящие из букв. Процесс ввода новых строк прервать, как только пользователь введет строку, состоящую и из букв и из цифр.
4. С клавиатуры вводится строка, состоящая из слов, отделенных друг от друга пробелами (пробелов может быть несколько). Напечатать только те слова, в которых первая буква встречается еще раз.
5. С клавиатуры вводится строка, состоящая из слов, разделенных пробелами (пробелов может быть несколько). Инвертировать (т. е. записать наоборот) все четные слова в этой строке.
6. Разработайте программу, с помощью которой пользователь вводит свое имя и фамилию, а затем программа выводит в одной строке фамилию, запятую и имя.
7. Разработайте программу, с помощью которой пользователь вводит фамилию, имя и отчество, а затем программа выводит его инициалы.

Список использованной литературы

1. *Фаронов В.В.* Delphi 3. Учебный курс. М.: «Нолидж», 1998. 400 с.
2. *Галисеев Г.В.* Программирование в среде Delphi 8 for .NET. М.: Издательский дом «Вильямс», 2004. 304 с.
3. *Павловска Т.А.* Паскаль. Программирование на языке высокого уровня. СПб.: Питер, 2003. 393 с.



4. Абрамов С.А. и др. Задачи по программированию. М.: Наука, 1988. 224 с.