Занятие 79 практическое

ИНСТРУКЦИОННАЯ КАРТА № 38

на выполнение практического занятия по дисциплине

"Основы программирования и баз данных"

для студентов специальности 09.02.02 Компьютерные сети

Тема: Работа со строками

Цель работы: освоить использование процедур и функций работы со строками.

Норма времени: 2 часа.

После выполненных работ студент должен

знать: понятие символьной строки, основные процедуры и функции обработки строк

уметь: создавать в среде Lazarus приложения для обработки строк

Оснащение рабочего места: ПК, инструкционные карты, конспект.

Литература:

1. Иллюстрированный самоучитель по Lazarus для начинающих (в электронном виде, CHM).

Вводный инструктаж

Следует помнить, что Delphi и Lazarus – это среды программирования, ориентированные на язык программирования Object Pascal. Различаясь в деталях реализации, обе среды основываются на одном стандарте языка.

Символ - это графическое изображение буквы, цифры, арифметического знака, знака препинания или какого-либо другого знака, отвечающее какому-либо стандарту кодировки символов. **Строка** - это цепочка символов.

Каждому символу соответствует его номер в кодовой таблице символов.

Типы строк в Object Pascal

Наиболее значительное различие между Delphi и Lazarus заключается в работе со строками. В обеих системах основным строковым типом является **`String`** (англ. string - строка). Однако в Delphi для строк используется кодировка ANSI (кодовая страница cp1251), а в Lazarus – кодировка **UTF-8**.

В Object Pascal вынужденно уживаются два стандарта работы со строковыми данными. Тип данных **`string`** в Lazarus – это строка в кодировке UTF-8. По умолчанию в проекте указана директива

```
{$mode objfpc}{$H+}
```

Эта директива указывает, что длина строки не ограничена. Переменная типа **`string`** хранит физический *адрес* строки и *количество занимаемых ею байт*. Строка может также рассматриваться как массив символов.

Кроме того, существует тип **`PChar`**: строка в стиле C/C++ с нулевым символом в конце. Такие строки применяются в основном при обращении к функциям API Windows. Переменная типа **`PChar`** – это *указатель* на начало строки. Преобразование типа **`String`** в тип **`PChar`** выполняет функция

```
function PChar(S: String): PChar;
```

Непосредственно сами строки Object Pascal поддерживают единственную операцию, так называемую **операцию конкатенации**, то есть присоединения. С помощью операции конкатенации одна строка присоединяется к другой:

```
var S, S1, S2: String;
begin
  S1:='Привет из '; S2:='Прибрежного';
  S:=S1+S2;
end;
```

Результирующая строка **`S`** равна **`'Привет из Прибрежного'`**.

Следует различать тип данных **`char`** (одиночный символ) и **`String`** (строка символов). Если символ относится к неанглийскому алфавиту, то он занимает ДВА байта.

Стандартные процедуры и функции работы со строками

Lazarus поддерживает два класса процедур и функций для работы со строками. Первый используется для работы со строками в кодировке cp1251, а второй – с UTF8-строками. Имена процедур и функций второго класса отличаются добавлением префикса UTF8. Например, длина строки **`Length()`** и **`UTF8Length()`**. Для подключения UTF8-функций в предложение **`uses`** следует добавить модуль **`LCLProc`**:

```
uses
  Classes, SysUtils, FileUtil, Forms, Controls,
Graphics, Dialogs, StdCtrls, LCLProc;
```

Далее для простоты будут описаны функции обработки ANSI-строк, так как их действие одинаково.

Длина строки, то есть количество символов в строке, возвращается встроенной функцией:

```
function Length(S: String): Integer;
```

Для строк `'Hello!'` и `'Привет'` функция `Length` возвратит значения 6 и 12, а функция `UTF8Length` возвратит 6 и 6.

функция `Concat(sl, s2, s3)`

Возвращает последовательное соединение строк. Эквивалентна оператору `sl+s2+s3`

Работа со строками в Lazarus требует от программиста знания определенного перечня специальных функций.

функция `Concat(sl, s2, s3)` Возвращает последовательное соединение строк. Эквивалентна оператору `sl+s2+s3`
функция `Pos(Substr:String; Str:String): Integer` или `UTF8Pos` Возвращает позицию (индекс) первого вхождения `Substr` в строке `Str`. Если `Substr` нет в `Str`, возвращает 0. Подобна функции `strstr()` в C/C++
функция копирования части строки: функция `Copy(S:String; Index,Count:Integer): String` или `UTF8Copy` Возвращает подстроку строки S, начиная с номера символа, равного Index и содержащую до Count символов. Помимо прочего, эту функцию можно использовать для вывода нецелого числа с нужным количеством цифр после запятой. Для этого, сначала получаем строку из числа формата Real, затем находим функцией Pos позицию запятой в этой строке, прибавляем нужное количество и копируем в результат это количество символов.
функция `Insert(Source:String; var S:String; Index:Integer): Integer` Вставляет строку `Source` в строку `S`, начиная с номера символа, равного `Index`
процедура `Delete(var S: String; Index, Count: Integer)` или `UTF8Delete` Удаляет из строки `S` подстроку, начинающуюся с номера символа, равного `Index`, и содержащую до `Count` символов.
процедура `Val (s, v, code)` Преобразует строку `s` в соответствующее численное представление `v` . Если преобразование успешно, переменная `code` равна нулю.

функция **StringReplace(const S, OldPattern, NewPattern: string; Flags: TReplaceFlags): String**

Заменяет в строке **S** подстроку **OldPattern** на строку **NewPattern** с учётом флага **TReplaceFlags**.

Для работы с функцией нужно создать переменную типа **TReplaceFlags** - это множество, и включить в него одно или оба значения из следующих:

rfReplaceAll - будут заменены все вхождения. Если это значение не будет включено во множество, то будет заменено только первое вхождение;

rfIgnoreCase - замена будет без учёта регистра символов. Если это значение не будет включено во множество, то замена будет чувствительна к регистру символов.

Кроме перечисленных, в Lazarus также есть много других функций. Например:

Процедуры и функции преобразования дат и времени

Функции преобразования строк по различным критериям (изменить регистр и т.д.)

Функции преобразования в числовой формат и обратно

Выполняя вычисления, для ввода данных и отображения результатов используем следующие функции, работающие со строками Lazarus:

Функции преобразования строки в целое число или число с плавающей точкой:

```
StrToInt(S: String): Integer
StrToFloat(S: String): Extended
```

Функции преобразования числа в строку:

```
IntToStr(N: Integer): String
FloatToStr(X: Extended): String
```

Последняя функция может также использоваться для форматированного вывода результата. В следующем примере под число отводится 10 знакомест, из них 4 – после десятичной точки:

```
FloatToStrF(N, ffFixed, 10, 4);
```

Пример создания приложения

Задание:. Во введенной с клавиатуры строке выделить подстроку, находящуюся между первым и вторым двоеточиями. (Как известно, в файле **/etc/passwd** операционной системы Linux поля записей разделены двоеточиями, так что подобное приложение может выделять требуемое поле учетной записи).

Сценарий работы программы: Пользователь вводит строку и щелкает на кнопке «Вычислить». В поле Label отображается один из вариантов: «Строка не содержит двоеточий», «Строка содержит только одно двоеточие» или нужная подстрока.

Возможный вариант формы приведен на рис. 1.

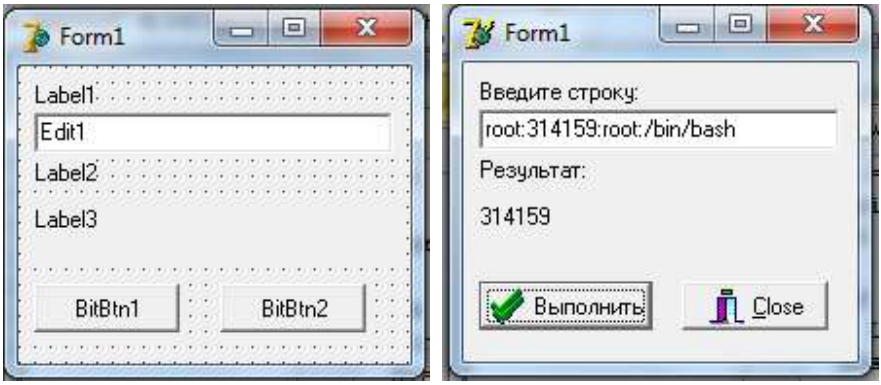

Рис. 1. Вид формы на этапах проектирования и выполнения

Компонент `BitBtn` расположен на странице Additional Палитры Компонентов и представляет собой разновидность стандартной кнопки Button. Его отличительная особенность – наличие растрового изображения на поверхности кнопки, которое определяется свойством **`Glyph`**. Кроме того, имеется свойство **`Kind`**, которое задает одну из 11 стандартных разновидностей кнопок. Кнопка bkClose закрывает главное окно и завершает работу программы.

Для кнопки `BitBtn1` выбираем **`Kind=bkAll`**, **`Caption=«Выполнить»`**. Для `BitBtn2` выбираем **`Kind=bkClose`**.

Возможны два варианта алгоритма выполнения работы.

Вариант 1. Используя функцию **`Pos`**, находим позицию первого двоеточия в строке – **`pos1`**. Затем процедурой **`Delete`** удалим из строки символы с первого до **`pos1`**. После этого снова ищем двоеточие в оставшейся части строки.

```
procedure TForm1.BitBtn1Click(Sender: TObject);
var
  str1, str2: string;
  pos1, pos2: integer;
  len, i: integer;
begin
    str1:=Edit1.Text;

  // вариант 1: использование встроенных функций
  pos1:=Pos(':',str1);
  str2:=str1;
  Delete(str2,1,pos1);
  pos2:=Pos(':',str2);
```

Вариант 2. Анализируем строку как массив символов. Вначале находим длину строки функцией **`Length`** и просматриваем строку, пока очередной символ строки не окажется равным '**`:`**'. Запоминаем номер **`pos1`**. Далее

просматриваем строку, пока не снова не встретится двоеточие, и запоминаем позицию в **pos2**.

```
  // вариант 2: анализ строки как массива символов
  pos1:=0; pos2:=0; i:=1;
  len:=Length(str1);
  while ((str1[i]<>':') and (i<=len)) do i:=i+1;
  if i<=len then pos1:=i;
  i:=i+1;
  while ((str1[i]<>':') and (i<=len)) do i:=i+1;
  if i<=len then pos2:=i-pos1;
```

Итоговая обработка одинакова для обоих алгоритмов:

```
  if pos1=0 then
    Label3.Caption:='Строка не содержит двоеточий'
  else
    if pos2=0 then
      Label3.Caption:='Строка содержит только одно двоеточие'
    else begin
      str2:=Copy(str1,pos1+1,pos2-1);
      Label3.Caption:=str2;
    end;
end;
```

НЕОБХОДИМО ОТРАБОТАТЬ ОБА ВАРИАНТА ПРИМЕРА.

Задания для самостоятельного выполнения

1. Во введённой строке удалите все символы, стоящие на нечётных местах.

2. Выясните, какая из букв, первая или последняя, встречается во введённой строке чаще?

3. В строке, введённой с клавиатуры, удалите все лишние пробелы.

Указание. Создайте вторую строку, в которую записывать результат.

4. Во введённой строке подсчитайте количество символов `'*'` и символов `'!'`.

5. Во введённой строке подсчитайте общее число вхождений символов `'+', '-', '*'`.

6. Во введенной строке определить количество вхождений заданной подстроки.

Указание. Рекомендуется использовать алгоритм, приведенный в варианте 1.

7. В строке, введённой с клавиатуры, заменить все X на Y.

8. Введённую с клавиатуры строку S1 записать в обратном порядке в строку S2. Строку S2 вывести на экран.

9. Во введённой строке каждую цифру заменить на следующую по порядку цифру. Цифру 9 заменить на цифру 0.

10. Во введённой строке все '123' заменить на '45'.

11. Определить и вывести на экран длину самого большого слова во введённой строке. Считать, что слова разделяются только пробелами

Дополнительные задания

1. Дан текст, содержащий цифры, латинские и русские буквы. Подсчитать сумму цифр, встречающихся в тексте.

2. Программа должна учитывать авиарейсы из города Прибрежное. Каждая строка содержит следующие данные:

```
0578Симферополь     15:4512345.48
```

номер рейса (целое число, 4 цифры); пункт назначения (строка длиной до 15 символов); время отправления (строка длиной 5 символов в формате "ЧЧ:ММ"); стоимость билета (число в формате 99999.99.

Составить программу, выделяющую из заданной строки пункт назначения и время отправления

3. Дана строка, состоящая из групп нулей и единиц. Каждая группа отделяется друг от друга одним или несколькими пробелами. Найти количество групп с пятью символами.

4. Определить и вывести на экран количество слов во введённой строке. Считать, что слова разделяются только одним или несколькими пробелами.

5. Дан текст. Составить программу проверки правильности написания сочетаний «жи»-«ши», «ча»-«ща», «чу»-«щу». Исправить ошибки.

6. Дана непустая последовательность непустых слов из латинских букв; соседние слова отделены друг от друга запятой, за последним словом – точка. Вывести текст, составленный из первых символов всех слов текста.

Содержание отчета о занятии

1. Эскиз окна приложения (формы)
2. Таблицы свойств формы и свойств компонентов.
3. Тексты программ-обработчиков событий.

Контрольные вопросы

1. Что такое строка?
2. Как объявляются переменные для хранения строк?
3. Какова максимальная длина строки?
4. Чем отличается тип Char от типа String?
5. Объясните механизм хранения строк.
6. Опишите функции для работы со строками.
7. Опишите процедуры для работы со строками.
8. Как обратиться к отдельному элементу строки?

Инструкционную карту составил преподаватель: Дубик Н.А.
Рассмотрено и одобрено на заседании цикловой комиссии
общетехнических и специальных дисциплин
Протокол № _____ от «_____» _____ 2017 г.
Председатель цикловой комиссии _____ Ю.Ф. Тулова

Дополнительные материалы и примеры выполнения заданий

Все задания выполнены для кодировки ср1251. Необходимо самостоятельно переделать их для работы со строками UTF8.

Задание 1: Во введённой строке удалить все символы, стоящие на нечётных местах

Для составления алгоритма решения рассмотрим пример. Пусть строка состоит из 7 символов: **`s:='1234567'`**. Удалим первый символ, тогда остаток строки равен: **`s:='234567'`**. В полученной строке надо удалить второй символ (3), после этого остаток строки равен **`s:='24567'`**. Таким образом, каждый раз в строке, оставшейся после удаления **`i`**-го символа, необходимо удалить **`(i+1)`**-ый символ:

вначале строка равна **`s:='1234567'`**

`i = 1` удаляем **`1`** символ, остаток **`s:='234567'`** **`i = 2`** удаляем **`2`** символ, остаток **`s:='24567'`**

`i = 3` удаляем **`3`** символ, остаток **`s:='2467'`** **`i = 4`** удаляем **`4`** символ, остаток **`s:='23456'`**

Количество повторений цикла **`n`** определяется так: если длина строки (количество символов) четная, то число повторений равно n div 2, иначе оно равно (n div 2)+1.

```
program pr31_01;
{ Во введённой строке удалить все символы, стоящие на нечёт-
ных местах }
var
  s: string; i,n: integer;
begin
  write('Введите строку:'); readln(s);  writeln('введенная
строка:',s);
  n:=length(s);
  if (n mod 2)=0 then n:=n div 2 else n:=(n div 2)+1;
  for i:=1 to n do delete(s,i,1);
  writeln('результат:',s);
  readln;
end.
```

Задание 2: выяснить, какая буква, первая или последняя, встречается во введённой строке чаще

Строка рассматривается как массив символов и просматривается от второго до предпоследнего символа.

```
program pr31_02;
{ выяснить, какая буква, первая  или  последняя, встречается
во введённой строке чаще }
var
  s: string[80];  c,c1,c2: char;  i,len,n1,n2: integer;
begin
  write('введите строку:'); readln(s);
```

```
  len:=length(s); c1:=s[1]; c2:=s[len];
  n1:=1; n2:=1;
  for i:=2 to len-1 do begin
    c:=s[i];
    if c=c1 then inc(n1); if c=c2 then inc(n2);
  end;
  writeln('первый символ встречен ',n1, ' раз, второй -
',n2);
  readln;
end.
```

Задание 3: во введенной строке заменить каждую точку многоточием

Создается вторая строка, вначале пустая. Обозначим **n** позицию первой встреченной в строке точки (используем функцию Pos). Если **n<>0**, то во вторую строку копируем (функция Copy) символы с первого по **(n-1)** и удаляем из первой строки **n** символов (процедура Delete). Затем снова находим **n**. Цикл повторяем, пока **n>0** (то есть пока в строке есть точки). По окончании цикла добавляем к результату остаток исходной строки.

```
program pr31_03;
{ во введенной строке заменить каждую точку многоточием }
var  s, subs,ins,s2: string;  n: integer;
begin
  write('введите строку: '); readln(s);
  n:=Pos('.',s); s2:='';
  while n>0 do begin
    s2:=s2+copy(s,1,n-1);
    s2:=s2+'...'; write(s2);
    delete(s,1,n); writeln('==>',s);
    n:=pos('.',s);
  end;
  s2:=s2+s; writeln(s2);
  readln;
end.
```

Задание 4: во введённой строке подсчитайте количество символов '*' и символов '!'

Решение аналогично заданию 2. Строка рассматривается как массив символов и просматривается от первого до последнего символа.

```
program pr31_04;
{ Во введённой строке подсчитайте количество символов '*' и
символов '!' }
var  s: string;  i, n1,n2: integer;
begin
  write('введите строку: '); readln(s);
  n1:=0; n2:=0;
  for i:=1 to length(s) do begin
    if s[i]='*' then n1:=n1+1;
    if s[i]='!' then n2:=n2+1;
  end;
```

```
  writeln('символ ''*'' найден ',n1, ' раз,',
   ' символ ''!'' - ',n2,' раз');
  readln;
end.
```

Задание 5. Во введённой строке подсчитайте общее число вхождений символов '+', '-', '*'.

Решение аналогично заданию 4. Строка рассматривается как массив символов и просматривается от первого до последнего символа.

```
program pr31_05;
{ Во введённой строке подсчитайте количество символов '+','-
','*' }
var  s: string;  i, n1,n2,n3: integer;  { количество
вхождений }
begin
  write('введите строку: '); readln(s);
  n1:=0; n2:=0; n3:=0;
  for i:=1 to length(s) do begin
    if s[i]='+' then n1:=n1+1;
    if s[i]='-' then n2:=n2+1;
    if s[i]='*' then n3:=n3+1;
  end;
  writeln;
  writeln('символ ''+'' найден ',n1, ' раз,', ' символ ''-''
- ',n2,' раз',
          ' символ ''*'' - ',n2,' раз' );
  readln;
end.
```

Задание 6. В строке, введённой с клавиатуры заменить все X на Y

Используем представление строки как массива символов

```
program pr31_06;
{ Во введённой строке заменить все 'X' на 'Y' }
var  s: string;  i: integer;
begin
  write('введите строку: '); readln(s);
  for i:=1 to length(s) do
    if s[i]='X' then s[i]:='Y';
  writeln;
  writeln(s);
  readln;
end.
```

Задание 9: подсчитать количество слов в строке

Будем считать, что слова разделены только одним пробелом. Добавим пробел в конец исходной строки, чтобы учесть последнее слово.

Далее ищем позицию первого пробела (p). Если p<>0, то удаляем p символов, начиная с первого. Процесс повторяем, пока p<>0.

```
program pr31_09;
{ подсчитать количество слов в строке }
```

```
var s:string; i,p:integer;
begin
  write('Введи строку: '); readln(s);
  s:=s+' ';
  i:=0;
  p:=pos(' ',s);
  while p<>0 do begin
    delete(s,1,p);
    inc(i);
    p:=pos(' ',s);
  end;
writeln('Найдено ',i,' слов(а)');
end.
```