Strings - Grade 10

Information Technology

8-18-2021

Table of Contents

FUNDAMENTALS

Strings can be thought of as letters from a-z (uppercase and lowercase inclusive). They can also contain numbers and characters (!@#$ ..etc). Strings are made up of individual characters.

DECLARE A STRING

Here we create a string variable called sName. Declare is another way of saying create. The sName variable can be used inside of the procedure it was declared in.

```
sName : string;  // Declaring a String Variable
```

ASSIGN A STRING

Assignment is putting a value inside of the of the string variable. You can only assign a value to a variable, after it has been declared(created). Values that are assigned to string variables must be inside of matching apostrophes `'myStr'`

```
sName := 'Donovan';   // Assignment of A String Variable
```

EXAMPLES OF STRINGS

The below examples are all strings. Even though '123' is a three-digit number, the fact that the variable sNum is declared as a string means that is how Delphi will treat it.

```
var sName,sCode,sChar,sNum,sReal : String;
sName := 'Milan';
sCode := '123Pass';
sChar := '!@#abc';
sNum  := '123';
sReal := '1.23';
```

CONCATENATION

Concatenating means joining two or more strings.

```
var sName, sSurname,sFullName : String;
begin
  sName := 'A.C';
  sSurname := ' Milan';
  sFullName := sName + sSurname;  // A.C Milan
end;
```

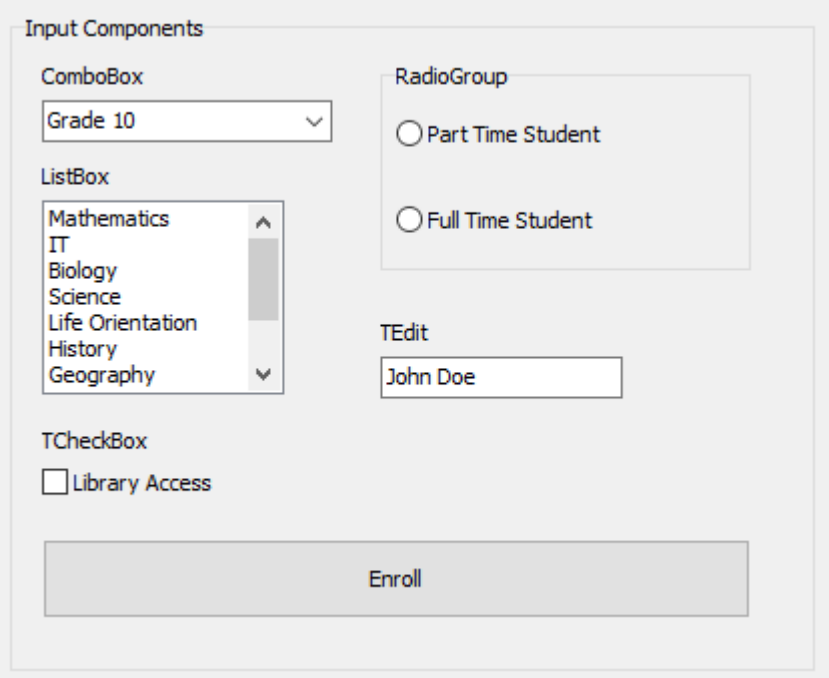
INPUT AND OUTPUT

A person using a Delphi program is normally faced with two options. One he *inputs* some data into the program, or two he receives some *output* from the program. Think about a calculator, you input a 1 then choose + then press 1 again. This is the input, when you press =, you see 2 which is the output.

INPUT

Input is something that user needs to enter. Like his name, or two numbers to work out the sum between them. Input is normally received from:

- TEdit (where the user would type something in)
- TComboBox/TListBox/TRadioBox (where user selects some option)
- TCheckBox
- TInputBox




```
procedure TForm1.btnEnrollClick(Sender: TObject);
var sGrade, sSubject, sLibrary, sStudent, sName : String;
begin
  sGrade := cmbGrade.Items[cmbGrade.ItemIndex];       // gets the selected item combobox
  sSubject := lstSubject.Items[lstSubject.ItemIndex];  // gets the selected item listbox
  sStudent := rgpStudent.Items[rgpStudent.ItemIndex]; // get the selected item radiogroup
  sName := edtName.Text;                               // get the text from tEdit
  if chkLibrary.Checked then
    redOutput.Lines.Add('Library Access Granted');     // determine if checkbox is checked
end;
```

OUTPUT

Output in Delphi is handled by a few components, like a:

- **RichEdit**
- **TEdit**
- **ShowMessage**

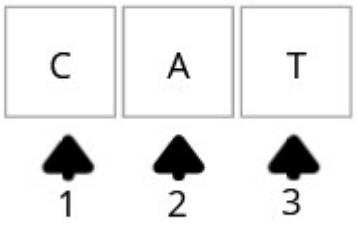
Its what the users sees. The above three components, all require the String Data Type. Attempting to put anything other than a string inside of them, will result in a syntax error.

```
var sName, sSurname,sFullName : String;
begin
  sName := 'A.C';
  sSurname := ' Milan';
  sFullName := sName + sSurname;

 // outputting a string with different components

  redOutput.Lines.Add(sFullName);   // RichEdit
  ShowMessage(sFullName);           // ShowMesage
  edtOutput.Text := sFullName;      // TEdit
  lblOutput.Caption := sFullName;   // TLabel
end;
```

ACCESSING A STRING CHARACTERS

Consider the string 'CAT'. It consists of three characters. To access each of the characters individually, we would need to specify their position.

```
var sStr : String;
begin
  sStr := 'CAT';
  redOutput.Lines.Add(sStr[1]); // 'C'
  redOutput.Lines.Add(sStr[2]); // 'A'
  redOutput.Lines.Add(sStr[3]); // 'T'
end;
```



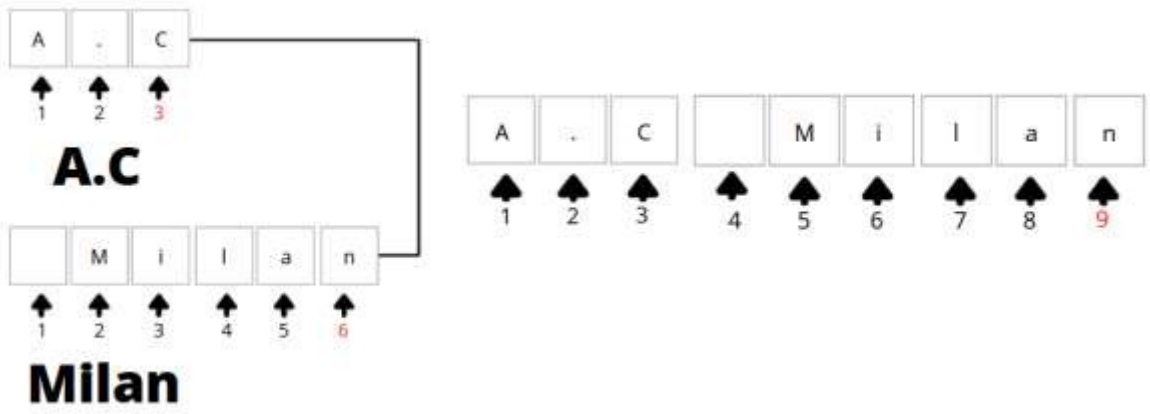
STRING FUNCTIONS

There are various string functions in Delphi, that allow us to do more with strings.

LENGTH FUNCTION

Counts how many characters are inside of a string and returns an **integer**.

```
var sName, sSurName, sFullName : String;
begin
  sName := 'A.C';
  sSurname := ' Milan'; // There is a whitespace before the M
  sFullName := sName + sSurname;  // A.C Milan

  redOutput.Lines.Add('length(sName): ' + intToStr(length(sName))); // 3
  redOutput.Lines.Add('length(sSurname): ' + intToStr(length(sSurname))); // 6
  redOutput.Lines.Add('length(sFullName): '  + intToStr(length(sFullName))); // 9
end;
```



UPCASE

Converts a single character from a string, into uppercase – returns a single character.

```
var sName, sSurName, sFullName : String;
begin
  sName := 'A.C';
  sSurname := ' Milan';
  sFullName := sName + sSurname;  // A.C Milan

  redOutput.Lines.Add(UpCase(sFullName[9])); // N
end;
```

UPPERCASE

Converts the entire string into uppercase – returns a string.

```
procedure TForm1.Button1Click(Sender: TObject);
var sName, sSurName, sFullName : String;
begin
  sName := 'A.C';
  sSurname := ' Milan';
  sFullName := sName + sSurname;  // A.C Milan

  redOutput.Lines.Add(UpperCase(sFullName)); // A.C MILAN
end;
```

LOWERCASE

Converts the entire string into lowercase – returns a string.

```
procedure TForm1.Button1Click(Sender: TObject);
var sName, sSurName, sFullName : String;
begin
  sName := 'A.C';
  sSurname := ' Milan';
  sFullName := sName + sSurname;  // A.C Milan

  redOutput.Lines.Add(LowerCase(sFullName)); // a.c milan
end;
```

COPY

Extracts a piece of a given string – returns a string.

copy(string,startIndex,count)

- **string** : the string you want to copy from
- **startIndex** : the position to start copying from
- **count**: the number of characters to copy

This function is used create a substring of a given string. To understand what this means better, consider this example which extracts pieces from the alphabet.

```
procedure TForm1.Button1Click(Sender: TObject);
var sAlphabet : string;
begin
  sAlphabet := 'abcdefghijklmnopqrstuvwxyz';
  redOutput.Lines.Add(copy(sAlphabet,1,3));  // abc
  redOutput.Lines.Add(copy(sAlphabet,24,3));  // xyz
  redOutput.Lines.Add(copy(sAlphabet,4,4));  // defg
end;
```

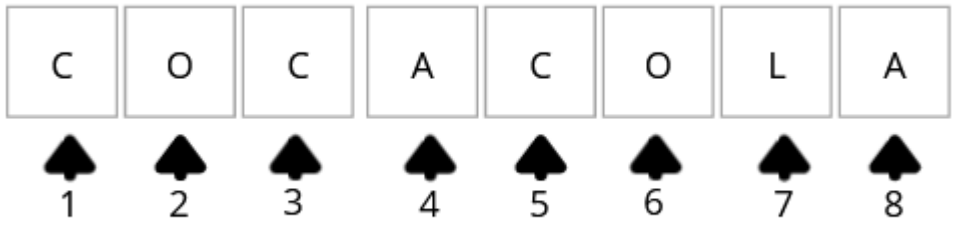
EXAMPLE

Consider this example, of the soft drink ‘CocaCola’.

```
var sSoftDrink : string;
begin
  sSoftDrink := 'CocaCola';
end;
```

We can see that the sSoftDrink variable contains 8 characters and can use the length function to confirm this.

```
procedure TForm1.Button1Click(Sender: TObject);
var sSoftDrink : string;
begin
  sSoftDrink := 'CocaCola';
  redOutput.Lines.Add(inttostr(length(sSoftDrink)));   // 8
end;
```



To separate the sSoftDrink variable into two words ‘Coca’ and ‘Cola’ we can use the string function copy.

```
procedure TForm1.Button1Click(Sender: TObject);
var sSoftDrink,sFirst,sSecond : string;
begin
  sSoftDrink := 'CocaCola';
  redOutput.Lines.Add(inttostr(length(sSoftDrink)));   // 8

  sFirst := copy(sSoftDrink,1,4); // Coca
  sSecond := copy(sSoftDrink,5,4); // Cola

  redOutput.Lines.Add('sFirst = ' + sFirst);
  redOutput.Lines.Add('sSecond = ' + sSecond);
end;
```

DELETE

Deletes a part of a given string. Take note that it modifies the string that it is working on.

delete(string,startIndex,count)

- **string** : the string you want to delete from
- **startIndex** : the position to start deleting from
- **count**: the number of characters to delete

In the below example, we delete the second part of the 'CocaCola' string.

```
var sSoftDrink: string;
begin
  sSoftDrink := 'CocaCola';
  delete(sSoftDrink,1,4);
  redOutput.Lines.Add(sSoftDrink); // Cola
end;
```

In the below example, we delete the first part of the 'CocaCola' string.

```
procedure TForm1.Button1Click(Sender: TObject);
var sSoftDrink: string;
begin
  sSoftDrink := 'CocaCola';
  delete(sSoftDrink,5,4);
  redOutput.Lines.Add(sSoftDrink); // Coca
end;
```

POS

Pos is short for position, and locates the position of a string inside of another of another string – returns an integer. It will return 0, if the string is not found.

pos(stringToSearchFor,stringToSearchFrom);

- stringToSearchFor : is the string you want to find
- stringToSearchFrom : is the string you need to search from

In the Coca-Cola example, we can search for the character 'a' :

```
var sSoftDrink: string;
    iPos : integer;
begin
  sSoftDrink := 'CocaCola';
  iPos := pos('a',sSoftDrink);
  redOutput.Lines.Add(inttostr(iPos)); // 4
end;
```

We see that the output is the value 4. Now its worthwhile to mention that there are two a character inside of the sSoftDrink variable. One at position 4, and the other at position 8. Therefore, pos returns the first occurrence it finds.

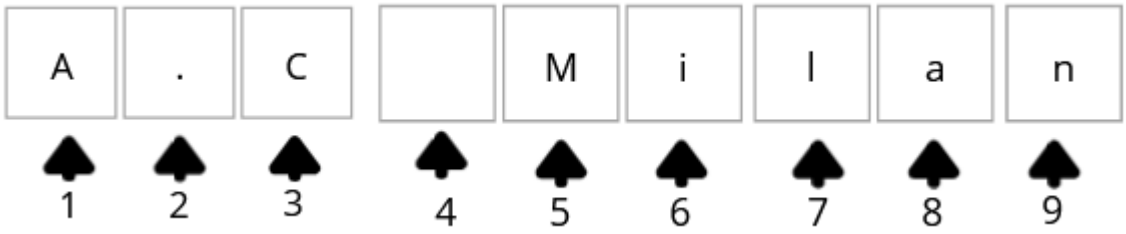
In this example, we search for the substring ‘la’. The value returned by pos is 7, which is the position of ‘l’ in the sSoftDrink varaible.

```
var sSoftDrink: string;
   iPos : integer;
begin
 sSoftDrink := 'CocaCola';
 iPos := pos('la',sSoftDrink);
 redOutput.Lines.Add(inttostr(iPos)); // 7
end;
```

EXAMPLE

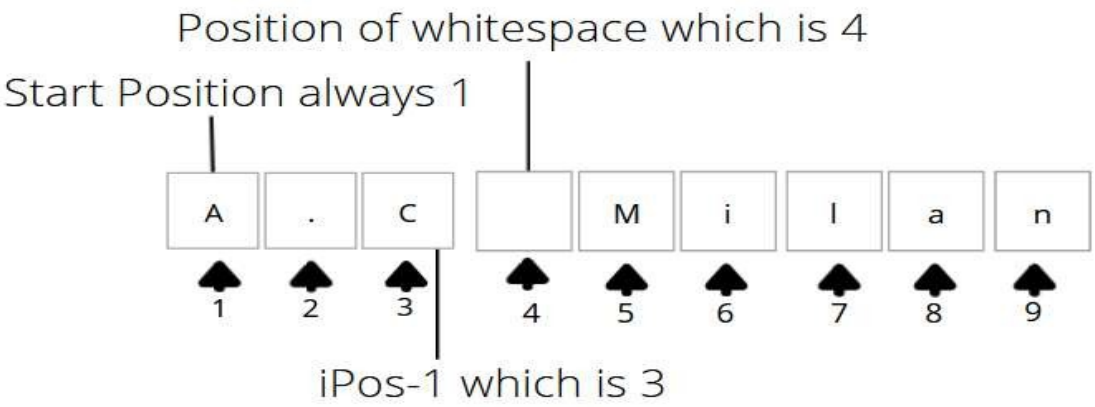
Consider this example, where we have a sFullName variable, containing a user’s name and surname, which are separated by a whitespace.

```
var sName, sSurname, sFullName : string;
begin
 sFullName := 'A.C Milan';
end;
```



To separate this full name into the name and surname:

1. Locate the whitespace
2. Use copy to isolate the name and surname.



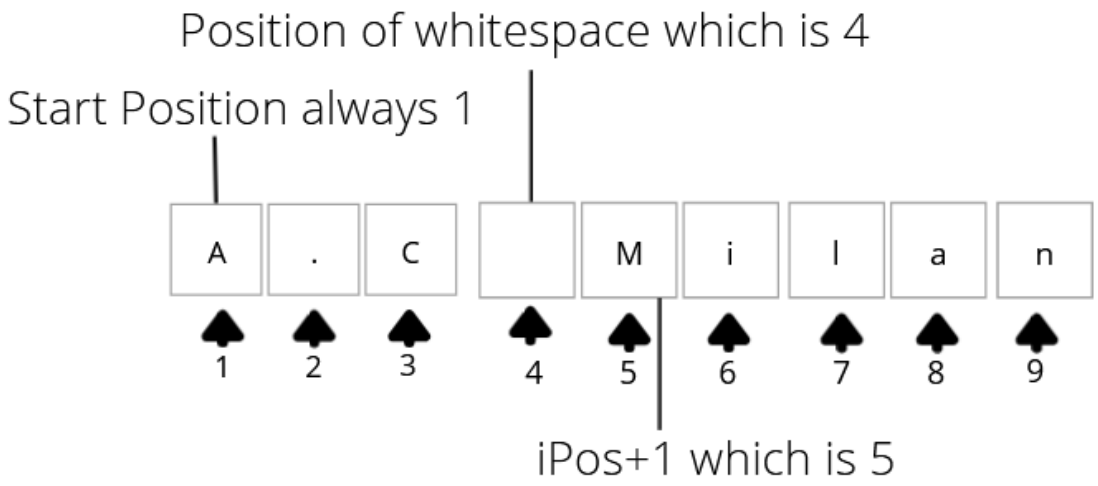
First we locate the whitespace:

```
var sName, sSurname, sFullName : string;
   iPos : integer;
begin
 sFullName := 'A.C Milan';
 iPos = pos(' ',sFullName); // 4
end;
```

Second, we isolate the name and surname. Now that we know where the whitespace is located, we can copy from the start of the string(which is always position 1), to iPos-1.

```
var sName, sSurname, sFullName : string;
   iPos : integer;
begin
 sFullName := 'A.C Milan';
 iPos := pos(' ',sFullName);
 sName := copy(sFullName,1,iPos-1);

 redOutput.Lines.Add('sName = ' + sName);  // A.C
end;
```

We use the same technique for the surname. The starting position changes to iPos+1:



```
var sName, sSurname, sFullName : string;
   iPos : integer;
begin
 sFullName := 'A.C Milan';
 iPos := pos(' ',sFullName);
 sName := copy(sFullName,1,iPos-1);
 sSurname := copy(sFullName,iPos+1); // notice here there is only two
 //parameters. If you leave out the 'count' from copy, it copies until
 // the end of the string

 redOutput.Lines.Add('sName = ' + sName);  // A.C
 redOutput.Lines.Add('sSurname = ' + sSurname);  // Milan
end;
```

IN

Checks if an element exists within a set.

```
var cChar : char;
begin
  cChar := 'a';
  if cChar IN['A','B','c','!','F'] then
   redOutput.Lines.Add(cChar + ' found')
  else
   redOutput.Lines.Add(cChar + ' not found');
  // 'a' not found in this example
end;
```

EXAMPLE

In this example, we make use of IN operator to determine if how many vowels a given string contains.

```
var sString : string;
   iCount,i : integer;
begin
   sString := 'The big brown fox';
   iCount := 0; // initialize iCount to 0 which will count how many vowels

   // loop through the string
   for i := 1 to length(sString) do // loop start at beginning of string go to end
   begin
     // sString[i] is the individual characters inside of the string
     if sString[i] IN ['a','e','i','o','u','A','E','I','O','U'] then
       iCount := iCount + 1;
   end;
   // Once the loop has finished print the number of vowels if any
   redOutput.Lines.Add('Number of vowels: ' + inttostr(iCount)); // 4
end;
```