Mr Long Summary ON STRING HANDLING

Understanding strings

Strings

- Data type used to store text
- Size (in bytes) is dependent on how it is declared.
    - Uses ONE byte for the length of the text (characters in the string) and the rest is used to store the characters (ONE byte for each character).

Examples:

String Declaration	Description
var sReport : **string** ;	Unlimited number of characters. Memory is allocated to the string as needed.
var sName : **string[30]** ;	String can contain 30 characters. 30 + 1 (for the length of the string) = 31 bytes will be reserved to store this string.
var sUsername : **ShortString** ;	*ShortString* can contain up to 255 characters.

- Strings are written between single quotes:
  Example: sName := '*Bruce*' ;
- An *empty string* is a string which contains 0 characters. It is indicated by two single quotes with nothing between them (NOT even a space).
  Example: sName := '' ;

- Each character in a string is allocated a position where 1 is the first character in the string.

  Example:
  sTemp := '*Hello World*' ;

1	2	3	4	5	6	7	8	9	10	11
H	e	l	l	o		W	o	r	l	D

    - *H* is the first character at position 1.
    - *D* is the last character at position 11.
    - *o* is the character at position 5.
    - *Space character* is at position 6.

- Each character in a string can be referenced by using the string's variable name followed by the position number of the character in square brackets.
- Examples:

String Declaration	Description
sName := sTemp**[1]** ;	The first character in ***sTemp*** is stored in the ***sName*** string.
sName := sTemp**[8]** ;	The eighth character in ***sTemp*** is stored in the ***sName*** string.
iNum := Random(6) + 1 ; sName := sTemp**[iNum]** ;	***iNum*** is a random number from 1 to 6. A random character (the value of ***iNum***) in ***sTemp*** is stored in the ***sName*** string.

String Functions

- A function is called with specific parameters and returns ONE value
- String functions do not all return string values, some return other data types, but the are used when handling strings.

NOTE: For the examples below, we will refer to sTemp := '*Hello World*' ;

1	2	3	4	5	6	7	8	9	10	11
H	e	l	l	o		W	o	r	l	D

Length function

- Requires ONE string parameter
- Returns an integer value that represents the number of characters in the given string.

Code	Description
iNum := **length**(sTemp) ;	iNum = 11 (Number of characters in the sTemp string)

for i := 1 to **length**(sTemp) do
begin
  memDisplay.Lines.Add(sTemp[i]) ;
end ;
	Loop from 1 to the length of the ***sTemp*** string (11) and display each character of ***sTemp***.

Pos function

- Requires TWO string parameters: a *substring* and a *string*
- Returns an integer value that represents the position of the first letter of the *substring* in the *string* (what position is the *substring* contained in the *string*).

Code	Description
iNum := **pos**('lo' , sTemp) ;	iNum = 4 (Position of the substring "lo" in sTemp.)
iNum := **pos**('l' , sTemp) ;	iNum = 3 (Position of the substring "l" in sTemp.) ***NOTE:*** *If multiple occurrences, result will be the first occurrence of the substring.*
iNum := **pos**('word' , sTemp) ;	iNum = 0 (Position of the substring "word" in sTemp.) ***NOTE:*** *If no occurrence of substring, then zero is returned.*
iNum := **pos**('world' , sTemp) ;	iNum = 0 (Position of the substring "world" in sTemp.) ***NOTE:*** *Must match case as well.* ***World*** *and* ***world*** *are not the same string (capital first letter vs small first letter).*

Copy function

- Requires ONE string parameter, followed by TWO integer parameters (first representing starting position and the second representing the length)
- Returns a string value that is a copy of all the characters in the string parameter starting from position of starting position integer parameter and copies the number of characters of the length integer parameter.
  NOTE: It is NOT from the starting position integer to the second integer, but from the starting position, for the second integer number of character.

Code	Description
sExtra := **copy**(sTemp, 3, 5) ;	sExtra = '*llo W*' (Copies from position 3, for 5 characters: character 3, 4, 5, 6 and 7)
sExtra := **copy**(sTemp, 7, 10) ;	sExtra = '*World*' (Copies from position 7, for 10 characters) ***NOTE:*** *If the number of characters to copy goes beyond the end of the string, it will only copy the characters until the end of the string.*
sExtra := **copy**(sTemp, 2, 2) ;	sExtra = '*el*' (Copies from position 2, for 2 characters: character 2 & 3)
sExtra := **copy**(sTemp, 1, pos(' ', sTemp) ;	sExtra = '*Hello* ' (Copies from position 1, for the number of characters until the position of the first space in the ***sTemp*** string) ***NOTE:*** *The space is included in the copy.*

Uppercase and Lowercase function

- Requires ONE string parameter.
- Returns a string value that is a copy of the string parameter but in all capital letters (with Uppercase) or in all small letters (with Lowercase).

Code	Description
sExtra := **Uppercase**(sTemp) ;	sExtra = '*HELLO WORLD*' (Copies ***sTemp*** in capital letters)
sExtra := **Lowercase**(sTemp) ;	sExtra = '*hello world*' (Copies ***sTemp*** in small letters)
sTemp := Uppercase(***sTemp***) ;	***sTemp*** is replaced with itself but in capital letters.

Concat function

- Requires at least TWO string parameters but can take in many.
- Returns all the given string parameters as ONE string.
  NOTE: It is a replacement for using the + symbol when combining strings

Code	Description
sExtra := **Concat**(sTemp, ' ', 'Goodbye') ;	sExtra = '*Hello World Goodbye*' (Combines ***sTemp***, *space* string and '*Goodbye'* string into one string)

String Procedures

- A procedure is simply called with specific parameters.
- Examples below will continue to use the ***sTemp*** string (***sTemp*** := '*Hello World*').

Insert procedure

- Requires TWO string parameters (first representing the *subString* that will be inserted and the second representing the string that the *subString* will be inserted into), followed by ONE integer parameter.
- The substring string parameter will be inserted into the second string parameter starting at the position specified by the integer parameter.

Code	Description
Insert ('YES', sTemp, 3) ;	sTemp = '*HeYESllo World*' (inserts the string '*YES*' into ***sTemp*** starting at position 3)
Insert ('YES' , sTemp, 15) ;	sTemp = '*Hello WorldYES*' (inserts the string '*YES*' into ***sTemp*** starting at position 15 but because ***sTemp*** is only 11 characters in length, it simply inserts it at the end of ***sTemp***)

Delete procedure

- Requires ONE string parameter, followed by TWO integer parameters (first representing starting position and the second representing the length)
- Characters in the string parameter will be removed starting from starting position integer parameter and will remove the number of characters of the length integer parameter.
  NOTE: It is NOT from the starting position integer to the second integer, but from the starting position, for the second integer number of character.

Code	Description
Delete (sTemp, 3, 5) ;	sTemp = '*Heorld*' (deletes ***sTemp*** from position 3, for 5 characters: character 3, 4, 5, 6 and 7)
Delete (sTemp, 9, 10) ;	sTemp = '*Hello Wo*' (deletes ***sTemp*** from position 9, for 10 characters) ***NOTE:*** *If the number of characters to delete goes beyond the end of the string, it will only delete the characters until the end of the string.*
Delete (sTemp, 1, pos(' ', sTemp) ;	sTemp = '*World* ' (deletes ***sTemp*** from position 1, for the number of characters until the position of the first space in the ***sTemp*** string)

Character functions

- These functions deal with char data types (characters or strings of length 1)

Upcase function

- Requires ONE char (character) parameter.
- Returns the given char parameter value but in capital letters.

Code	Description
cLetter := **Upcase**(sTemp[3]) ;	cLetter = '*L*' (Copies ***sTemp***'s third letter but in capital letters)
sTemp[1] := Upcase(***sTemp[1]***) ;	The first character in ***sTemp*** is converted to uppercase.

NOTE: In computers, each character is represented by an ASCII code. ASCII codes are used to represent text when storing the numerical code in a byte.

Ord function

- Requires ONE char (character) parameter.
- Returns the ASCII code of the character parameter.

Code	Description
iNum := **Ord**('B') ;	iNum = 66 (ASCII Code of a capital B is 66)
iNum := **Ord**('b') ;	iNum = 98 (ASCII Code of a lowercase B is 98)
iNum := **Ord**('?') ;	iNum = 63 (ASCII Code of a question mark is 63)

Chr function

- Requires ONE integer parameter.
- Returns the character (char) associated with the ASCII code integer parameter.

Code	Description
cLetter := **Chr**(70) ;	cLetter = '*F*' (Capital F's ASCII Code is 70)
cLetter := **Chr**(111) ;	cLetter = '*o*' (Lower case O's ASCII Code is 111)
cLetter := **Chr**(55) ;	cLetter = '*7*' (Text 7's ASCII Code is 55)

3 Techniques with String Handling

When dealing with strings there are 3 common types of scenarios that require 3 different techniques:

- Extracting sections or blocks of text from a string
- Traversing a string to check each character
- Recreating or building a new string
  NOTE: There are other types of scenarios, but these are the more common scenarios

Extracting sections or blocks of text from a string

- Involves have a string and you want to extract sections of the text out of the string. For example:
  - Copy the first 5 characters, then the next 3 characters.
  - Extract each word in a string when a space character indicates the end of each word.
  - Comma separated string where each block of data is separated by a comma.
- More complicated examples are when the size of the sections to be extracted are not constant but are indicated a delimiter (comma or hash).
  Example: Surname<*comma*>Name<*comma*>IDCode
  Banner,Bruce,12345

- Suggested technique:
  - Find the position of the first delimiter (comma, hash space, etc)
  - Copy from 1 until the position of the delimiter – 1
    NOTE: You don't want to also copy the delimiter, hence the -1
  - Delete from 1 until the position of the delimiter
    NOTE: Remove the section of string that you have stored to work with what's remaining.
  - Continue this process for each section of data EXCEPT the last remaining section.
  - By deleting each section after storing it's value, you will eventually be left with just the last section of data and NO delimiter. You can simply store the last section of data, as is, in the appropriate variable.

Consider the example of extracting the data from: *Banner,Bruce,12345*

Sample Code – Extracting sections

```
sLine := 'Banner,Bruce,12345' ;
iComma := pos( ',' ,  sLine ) ;          //find position of first delimiter (comma) in sLine
sSurname := Copy( sLine, 1, iComma – 1 ) ;  //copy surname from position 1 to one before comma
Delete( sLine, 1, iComma ) ;             //remove copied section: sLine = Bruce,12345
//repeat the process for Name
iComma := pos( ',' ,  sLine ) ;          //find position of first delimiter (comma) in sLine
sName := Copy( sLine, 1, iComma – 1 ) ;  //copy name from position 1 to one before comma
Delete( sLine, 1, iComma ) ;             //remove copied section: sLine = 12345
//last section is what is left in sLine
iCode := StrToInt( sLine ) ;             //convert because data is integer
```

If more sections needed to be extracted, then repeat the first process until you are left with the final section of data. If string consists of different delimiters (# for first section, $ for second, etc) then find the position of each different delimiter.

Traversing a string to check each character

- Involves having to count or compare each individual character of a string.
  For example:
  - Count the number of M's in a string.
  - Count the number of vowels in a string.

- Suggested technique:
  - Loop from 1 to the length of the string
    Example: ***for*** *K := 1* ***to*** *length(sLine)* ***do***
  - Compare string at position loop variable with what you are counting or required
    Example: ***if*** *sLine[K] = …* ***then***

Consider the example of counting all the vowels in the string ***sLine***.

Sample Code – Traversing string

```
iCount := 0 ;                                  //initialise the vowel counter
for K := 1 to length( sLine ) do               //loop through each character in sLine
begin
  if sLine[ K ] IN [‘a’, ‘e’, ‘i’ , ‘o’ ,’u’ ] then    //check if character is a vowel
  begin
    Inc( iCount ) ;                            //increase vowel counter when vowel found
  end ;  // end of if
end ;  //end of for

showmessage( Inttostr( iCount ) + ‘ vowels in ‘ + sLine ) ;          //display vowel counter
```

*This example would not count capital letter vowel characters. Either add the capital vowels to the set after the IN operator or use if **lowercase**(sLine[K] IN….*

Recreating or building a new string

- Involves having to add onto a string or construct a string.
  For example:
  - Create a comma separate string with a name, surname and ID variables.
  - Create a string that is in the reverse order of characters of a specific string.

- Suggested technique:
  - If a few components to add, you simply use the + operator to add all the different components of the string.
  - If many components are needed, then initialise your string to the empty string, then add each component individually.
    Example: *sNew := ‘’ ;*
    sNew := sNew + sName + ‘,’ ;

Consider the example of creating a reverse version of ***sLine*** (hello = olleh).

Sample Code – Constructing a string

```
sNew := ‘’ ;                                   //initialise the new string to empty string
for K := length( sLine ) downto 1 do           //loop through each character in sLine in reverse order
begin
  sNew := sNew + sLine[ K ] ;                  //add each character individually to sNew
end ;  //end of for
showmessage( sNew ) ;                          //display sNew string which is sLine in reverse
```

Additional Links:

- Youtube video playlist:
  https://www.youtube.com/watch?v=YNCR7RT4t2g&list=PLxAS51iVMjv-0sMKwn0DzTEM6L3HcFL5y
- Google drive resource activities:
  https://tinyurl.com/MLE-G10IT-StringHandling