04: Descriptors

Exercise Instructions

Goal

The two parts of this module cover working with descriptors. Through various examples, you learn their individual differences as well as a few of the available manipulation functions.

Introduction

Working with text in various ways is one of the basics of application development. Due to efficiency and memory requirements, Symbian OS provides its own types for strings, called descriptors. They differentiate very well between various ways of storing the text (heap, stack or ROM) and are very powerful to use thanks to the class hierarchy – however, you will have to spend some time to get used to them.

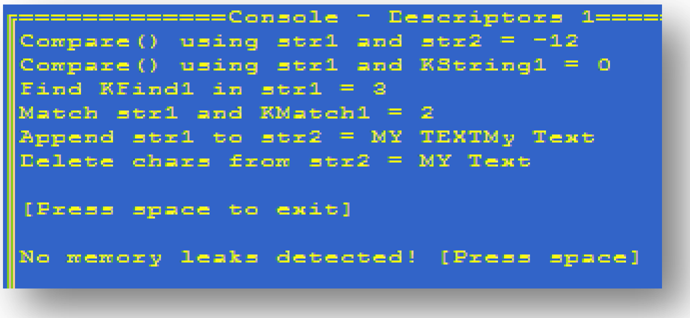
Structure of this Exercise

To keep a clear structure in this exercise, it has been split up into two parts. While the first part covers working with a modifiable descriptor, the second part compares various different types and some different use cases. The general structure of the exercises is identical to the framework of the previous modules.

Contents of part 1: Basic Descriptors & Manipulation Functions

- Literals and Stack-based modifiable strings
- Comparing
- Finding
- Matching
- Copying
- Appending
- Deleting

The final output of part 1 should look like this:

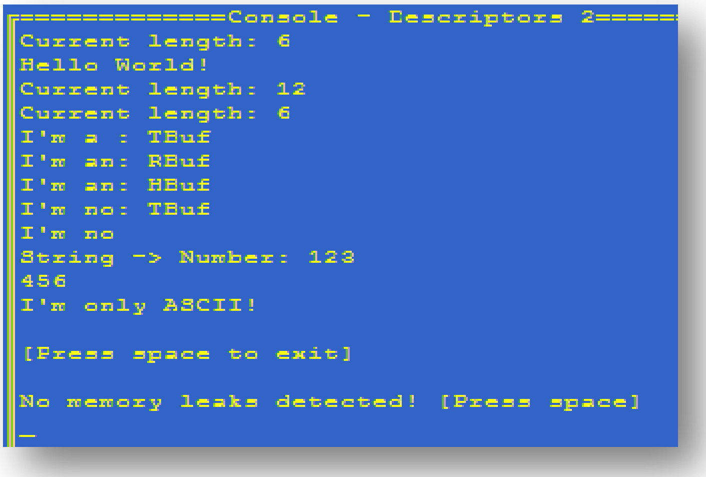


```
=============Console - Descriptors 1====
Compare() using str1 and str2 = -12
Compare() using str1 and KString1 = 0
Find KFind1 in str1 = 3
Match str1 and KMatch1 = 2
Append str1 to str2 = MY TEXTMy Text
Delete chars from str2 = MY Text

 [Press space to exit]

No memory leaks detected! [Press space]
```

Contents of part 2: Advanced Descriptor Usage

- Using the `RBuf`
- Function parameters and return types
- `TPtr` and `TPtrC`
- String to/from number conversion
- Unicode conversion

The final output of part 2 should look like this:



```
==============Console - Descriptors 2======
Current length: 6
Hello World!
Current length: 12
Current length: 6
I'm a : TBuf
I'm an: RBuf
I'm an: HBuf
I'm no: TBuf
I'm no
String -> Number: 123
456
I'm only ASCII!

[Press space to exit]

No memory leaks detected! [Press space]
_
```

Detailed Description of Part 1

Initialization

The first steps of this part contain defining several literals that will be used throughout the exercise as well as two `TBuf` variables.

Compare

Some basic experiments with the "compare" functionality of descriptors, which is basically identical to standard C.

Find

The find function allows getting the position of a text (needle) in a larger text (haystack).

Match

Matching is more powerful than finding and even allows to use wildcards like `*` and `?`.

Copy

In this section, the copy functionality is presented, which even exists in some specialized variants which can make the text uppercase while copying.

Append

Simple appending of descriptors.

Delete

Allows deleting parts of the contents of a descriptor.

Detailed Description of Part 2

Initialization

Pre-written fixed string literals that will be used in this exercise.

Using RBuf

The `RBuf` is a new descriptor type that is available since Symbian OS 8. It is more comfortable to use than the `HBufC*` and follows the usage concepts of R type classes. As most examples still use the `HBufC*`, this section also includes how to create an `RBuf` based on an `HBufC*` instance.

Function parameters and return types

Passing descriptors correctly to functions and returning new descriptors if they were created by the function is very important, as it’s easy to create memory leaks or problems, e.g. by incorrect handling of the cleanup stack.

TPtr and TPtrC

These descriptor types are very useful in a few specialized situations, e.g. they allow pointing to substrings of a larger text.

String to/from number conversion

From time to time, you will need to covert a descriptor to a number or the other way round, e.g. when reading / writing data from / to files.

`TLex` is a very powerful class that can also be used for converting text to numbers – while this task might sound easy, it is actually quite difficult considering the influence of the locale, with different methods for textually representing the comma of a floating point value, …

Unicode Conversion

The required method for Unicode conversion depends on the source and required target data. In this example, you have to use one of the simpler methods. In most cases it is sufficient to convert 8 bit descriptors, which you get from files and sockets, to 16 bit descriptors – which are normally used in current versions of Symbian OS.