01: Symbian OS Types and Declarations

Exercise Instructions

Goal

The goal of the first exercise is to get to know the basic data types of Symbian OS as well as the console mode of the Symbian OS emulator.

Introduction

If you know how powerful the UI of S60 and UIQ is, it might be a bit boring to start with the console interface of the emulators. However, this has got several advantages:

- ***Time:*** The start-up time is a lot faster than if the emulators have to load the full UI. Especially at the beginning this is very important, as it's a quick way to try several alternatives for different tasks.
- ***Complexity:*** Even a basic "*Hello World*" GUI application does already require several classes and many definitions. For getting to know the general style of Symbian OS development, it's better to start with the console. Development is like with standard C / C++.
- ***Independence:*** The code presented for console applications will work for both S60 and UIQ, so the training is more generic and not targeted to specific environments right from the beginning.

Structure of this Exercise

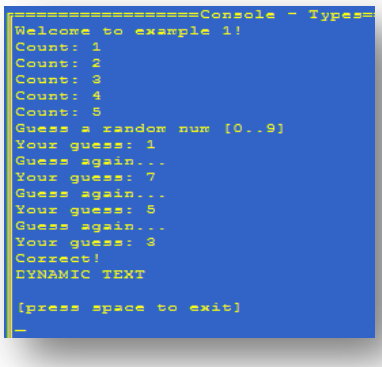
The entry point of the application is the `E32Main()`-function, which will catch and output any errors (= leaves) that the code might produce. No changes have to be made here.

The `MainL()`-function is split up into several different sections:

- ***Initialization:*** Creates the console window for text output.
- ***Using TInt, text output:*** Some basic experiments with `TInt` variables and text output.
- ***Random numbers, TBool and TChar:*** Lets the user guess a random number, while working with some more basic data types of Symbian OS.
- ***R Classes:*** How to use an R type class, in this case the `RBuf`.
- ***Cleanup:*** Waits for user input and deletes the console.

C type classes will be examined more closely in the memory management modules, as their usage requires knowledge of several other concepts like the cleanup stack or two-phase construction.

The final output should look like this:



Detailed Descriptions

Initialization

In this section of the training exercise, the text output console is created. This is a C type class and therefore has to be created on the heap. Further instructions on how to handle C type classes will be presented in the memory management modules. No edits have to be done here.

Using TInt, text output

First, this section explains how to format and print text to the console. This works like the `printf()`-function of standard C.

To start off easy, a simple for-loop has to be created, which outputs numbers from 1 to 5, using a `TInt` as counting variable. The current value should be printed using formatted text.

Random numbers, TBool and TChar

This section is the main part of this exercise and contains a short number guessing game. The computer thinks of a random number from 0..9. The user now has to guess it by entering a number.

The input is handled in a `do` / `while`-loop. A single character is read from the keyboard, which is then converted to its numeric value using one of the functions provided by the `TChar` data type. Next, the application checks if the user has guessed the correct number.

This example could be extended to inform the user if his guess was too high or too low. Also, the number of tries could be counted.

R Classes

In this section, basic use of R type classes is demonstrated. More detailed explanations of the RBuf descriptor are presented in the “Descriptors”-module.

The `RBuf`-class manages data on the heap. It is important to know that these classes have to be cleaned up by using `Close()` or `Reset()` (depending on the class).

Cleanup

Waits for the space character to ensure the user can still read the output. Then, the console is deleted. No edits have to be done here.

Glossary

You might encounter the following definitions, which will not be familiar to you as of now. The following table lists them, along with a short description:

Term	Description
RBuf	One of the Symbian OS descriptors, available since Symbian OS 8. Descriptors are the Symbian OS way of C++ strings.
_LIT	Defines a literal = a fixed descriptor (string) that's stored directly in the compiled application.
Cleanup Stack	Allows safe deletion of heap-based objects when used as local variables in a function and an error (leave) occurs before the delete-statement can be reached. Also applies to R type classes which open a connection to an external service.
Leaves, TRAP	The exception handling mechanism of Symbian OS. Exceptions are called "Leaves", they can be caught by using the TRAP / TRAPD-macro.