Imperial College of Science, Technology and Medicine
Department of Computing

M.Sc. C++ Programming – Unassessed Exercise No. 1

Issued: Friday 16 October 2015

Objective

The exercise is to write a two argument, integer-valued function `substring_position(...)` which searches to see if its first (string) argument is a substring of its second (string) argument. If it finds an instance of the substring, then it returns the start position of the *first* instance of the substring in the string, otherwise it returns -1. For example:

CALL	SHOULD RETURN
`substring_position("this", "this is a simple exercise")`	0
`substring_position("is", "this is a simple exercise")`	2
`substring_position("is a", "this is a simple exercise")`	5
`substring_position("is an", "this is a simple exercise")`	-1
`substring_position("exercise", "this is a simple exercise")`	17
`substring_position("simple exercise", "this is a simple")`	-1
`substring_position("", "this is a simple exercise")`	0
`substring_position("", "")`	0

The function declaration should be placed in a header file called `substring.h` (along with any other function declarations you might decide to use in your answer), and the function implementation should be placed in an implementation file `substring.cpp` (along with any other function implementations). The header and implementation files should be tested using the test program in the file `main.cpp`, which can be found on the web at the URL `http://www.doc.ic.ac.uk/~wjk/C++Intro/substring/main.cpp`.

Hints And Suggestions

Try to complete the exercise without using any of the standard string library functions (in particular, the use of `strstr` is strongly discouraged!). You may find it useful to decompose the task as follows:

1. Write a Boolean valued function `is_prefix(...)` which has two string arguments (and possibly other parameters as well). `is_prefix(...)` returns `True` if its first string argument (or perhaps some part of its first string argument specified by other parameters) is a prefix of its second string argument (or perhaps some part of the second string argument specified by other parameters).

   You might decide to write a recursive definition for `is_prefix(...)`, since, for example, to check that "`indent`" is a prefix of "`indentation`" it is sufficient to check that both strings start with the same letter, and that "`ndent`" is a prefix of "`ndentation`".

2. Test the function `is_prefix(...)`. When satisfied that it is correct, use it in the (recursive or iterative) definition of `substring_position(...)`.

See if you can:

- Define `substring_position(...)` and `is_prefix(...)` using pointer arithmetic instead of array subscripting for running through the strings, and
- define your function `substring_position(...)` recursively, and
- define the function `is_prefix(...)` recursively using exactly two parameters.

However, it is recommended that you first answer the question in whatever way you find most natural before spending too much effort on these finer points.

A challenge for fun

See if you can provide a one-line iterative function `substring_position2(...)` which uses the string library function `strstr` to perform the same task as `substring_position(...)`. Your function definition body (inside the curly braces) should contain only one semicolon, should not make use of any preprocessor symbols (i.e. no `#define`!) or global variables, and should not invoke any function other than `strstr`. Can you equal or better the current record of just 29 characters for the function body (not counting the braces)?