4th Practical Session
A report has to be written

Mourad Ismail Vincent Danjean Christophe Prud'homme

November 13, 2015

Contents

Abstract

The aim of this practical session is to prove that you are able to make link between an existing C or C++ library with Python's language.

1 General presentation

You will need to give a report (which will be marked) in latex/pdf format and push it in Github before November 27th

2 Simplified Wrapper and Interface Generator (SWIG)

Swig works from an input file like file header (header). However, it provides guidelines for additional swig introduced by the character '%. On her often gives the `.i`. From this file, swig will generate wrappers functions in `C/C++` code as well as in the target language (Python in our case). The python code will call the code wrappers functions that called itself the library code we want to use.

Check that swig is installed on your computer. If it is not the case, you can install it at least locally if you are not root

2.1 Syntax of SWIG's input file

A swig file has often the following structure :

```
%module mymodule

%{
/* part A */
#include "myheader.h"
%}

/* part B */
/* declaration in myheader.h */
int foo; /* global variable */
int bar(int x); /* function */
```

1. What are the differences between declarations in Part A and those in Part B?

Remark: If we want to have the same declarations in both A and B parts, we can use this syntax :

```
%include %{
/* common instructions for both parts */
%}
```

2.2 Elementary Types and Functions

2. Download the tarball file `libtp4.tar.gz` from the SVN repository `svn up`.

   Create a file `tp4.i` for swig that permits to access the functionalities of `tp4` library declared in the first part of `tp4.h` file.

   Complete the `CMakeLists.txt` files to insure a correct compilation (This step can be done at the end of your work. See http://www.swig.org/Doc1.3/Introduction.html for the use of swig with cmake)

3. Write a python program doing the same thing as the C code `example/test1`.

2.3 Structures, pointers and objects

For swig, the non simple types are considered as opaque types handled by pointers.

4. Write a python interface for the whole library

5. Write a python program showing that your able to use/acces to `Vecteurs` class. This program has to do more or less the same thing as the C program `example/test2`. Note that in python we have to call explicitly `Vecteur_destroy` to free memory.

3 (optional)

Write a Graphical User Interface (GUI) for your different codes using the Tkinter or Qt package

4 Online Documentation

- http://www.cmake.org/
- http://www.swig.org/
- http://www.swig.org/Doc2.0/SWIG.html
- http://www.swig.org/Doc2.0/Python.html
- https://docs.python.org/2/library/tkinter.html