Sistemas de Operação

(Ano letivo de 2025-2026)

Guiões das aulas práticas

script #01 Implementing a simple linked-list in C++ from scratch

---

Summary

- Revision of C/C++ programming
- Implementing a simple linked-list in pure C/C++

Environment

All practical work will be carry out in a Linux environment, with programs being developed in C/C++. Thus, first of all, you must have a Linux distribution installed in your computer. In terms of packages, at least the following or equivalent ones will be necessary: `build-essential`, `glibc-doc`, `manpages-dev`, `doxygen`. If you have Ubuntu, you can execute in the command line

```
sudo apt install build-essential glibc-doc manpages-dev doxygen
```

Introduction

The idea is to implement a simple linked list from scratch, without relying on supporting libraries, like the STL library. Typically, a simple linked list is built based on a data structure, called a node, containing the data itself and a pointer to the next node, that allows to build the list. Often, a proper data structure is also defined to hold the data. In the following 2 exercises, the data is composed of two fields:

- a 32-bit unsigned integer, representing a student number;
- a pointer to a (dynamically allocated) string representing the student's name. Recall that, in the C programming language, a string is implemented as a memory address of a zero-terminated sequence of characters.

In the first exercise, the list is implemented as a library, where every manipulation function has a parameter (a pointer to a node) indicating the list to be processed. In the second exercise, the list is implemented as a singleton, meaning that the manipulation functions do not have a parameter indicating the list to be processed, as there are only one.

The header files in both exercises have comments aimed to `doxygen`, a tool that allows you to produce HTML documentation from them. File `Doxyfile` is configured for that purpose. To generate and visualize that documentation, proceed as follows:

1. In the exercise folder, run command `doxygen`. Folder `html` will be created.
2. Open the `index.html` page inside the `html` folder. A simple way of doing that is executing command `firefox html/index.html &>/dev/null` (you might want to replace `firefox` with your favourite browser).
3. In the browser, a page titled `LinkedList` appears. By pressing tab `File`, a list of files appears. Select the only one there (`linked-list.h`) and enjoy.

Exercises

Exercise 1 *Implementing a linked-list as a library*

The objective of this exercise is to implement a simple linked-list in C++, as a library of functions. Folder `as-library` provides the base code for the implementation.

File `linked-list.h`, the header file, plays the following roles:

- *defines datatype `Student`, which represents the data to be stored in the list,*
- *defines datatype `SllNode`, which represents the node used to implement the list;*
- *declares the signatures of the list manipulation functions.*

File `linked-list.cpp` contains the skeleton of the manipulation functions. File `main.cpp` is the main program which implements a menu driven application. Read these files carefully and try to answer to the following questions.

(a) *What is the purpose of the pattern `#ifndef #define` in `linked-list.h` file? An equivalent, less-portable alternative is the use of `#pragma once`.*

(b) *All of the linked-list module functions have an `SllNode*` first argument. Why?*

(c) *Complete both the linked-list module functions and the main program. Follow an incremental approach, choosing just a few functions, implementing them, and testing them, before tackling the next ones. A good starting point would be to choose the insert and print functions. Leave function `sllLoad` for last. For your tests, you may need to edit the main program.*

Exercise 2 *Implementing a linked-list as a singleton*

The objective of this exercise is to implement a singleton simple linked-list in C++. Folder `as-singleton` provides the base code for the implementation.

File `linked-list.h`, the header file, just declares the signatures of the list manipulation functions. File `linked-list.cpp` plays the following roles:

- *defines datatype `Student`, which represents the data to be stored in the list;*
- *defines datatype `SllNode`, which represents the node used to implement the list;*
- *defines a module variable (`list`), which is the head of the unique linked list;*
- *contains the skeleton of the manipulation functions.*

Read these files carefully and try to answer to the following questions.

(a) *What is the purpose of the pattern `#ifndef #define` in `linked-list.h` file? An equivalent, less-portable alternative is the use of `#pragma once`.*

(b) *The support data structures (`Student` and `SllNode`) are declared in the .cpp file, not in the header file. Why?*

(c) *Variable `list` is defined as static. What is the consequence of this?*

(d) *Complete both the linked-list module functions and the main program. Follow an incremental approach, choosing just a few functions, implementing them, and testing them, before tackling the next ones. A good starting point would be to choose the insert and print functions. Leave function `sllLoad` for last. For your tests, you may need to edit the main program.*
