Reminding C: Function Pointers

Sebastien Varrette, Frederic Pinel and Pascal Bouvry
<Firstname.Lastname@uni.lu>

Version 1.0

Exercise 1 *Views*

In this exercise, we will process arrays of strings. An example of such an array is:

```
char *data[] = {
   "This is a line. It can be very long...",
   "This too. Imagine it is also quite long...",
   "# but this is a comment",
   "Line again. Long too.",
   "# comment 2",
   NULL
};
```

Assume the array is not known at compile time, for example: it could be read from a file. By convention, the last entry in the array, `NULL`, is a marker for the end-of-data. We want to build different *views* of this data. A view is a subset of the data. It is typically read-only.

A *filter* is a predicate (a function that returns true or false) used to create the view. All strings for which the predicate is true compose the view.
We suppose that each string can be of any length. As we want to create many views, each view should not copy the data, but rather *point* to the strings in the array.

Finally, we want to be able to execute an arbitrary function over all strings of the views.

1. Propose a structure to represent a view. For example: a linked list of pointers to strings in the data set.
2. Write a `create_view()` function, which creates a view on the data. The view is a list of elements that satisfy the predicate. The data set and predicate are parameters of this `create_view()` function. The `create_view()` function should return the created view.
3. Write a `del_view()` function, which frees all resources related to a view.
4. Write a `iter_view()` function, which will execute a function `f()` over each string of a view. This function `f()` is given as a parameter to `iter_view()`. The view is also a parameter. The function `f()` takes at least one parameter: the element of the view (here, a string). By convention, it must return void, and must not modify the element. As an application: use `iter_view()` to display on `stdout` the contents of a view.
5. Apply these functions to the creation of several views, show their contents, and delete them. For example: create a view that holds non-commented lines (lines not starting with a `#`). Print the line length of the items of a view.