1 The notorious vector example

It happens fairly frequently, in fact basically all the time, that we need to store objects of the same type (say integers) to treat them later on or to represent some higher abstraction like a polynomial.

Moreover in many of these applications, the number of elements to be stored is not known in advance (during compile time), but depends on the current execution (runtime). Consider multiplying two polynomials: Their maximal degree increases and we need more storage for the exponents and coefficients. This however depends on the user input and is therefore not known at compile time.

It is therefore crucial to have a good abstraction of such dynamically sized containers, that "abstract away" the complexity of allocating and releasing memory and are as easy to use as, let's say a Python list. There are several containers in the standard library that solve this issue, like

- *std::list*
- *std::deque*
- *std::vector*.

With each of them having its own pros and cons.

1.1 Requirements

Your implementation must not leak memory and allow access to existing elements. In this TP we constrain ourselves to integers and we therefore want

- **constructor:** Initialises an empty vector
- **destructor:** "Deletes" the object
- **push_back:** Appends a given element to the existing vector
- **pop_back:** Remove the last element from the vector
- **size:** Returns the number of currently stored elements
- **read-access:** Overload the *operator[]* such that it takes an index as argument and returns the value at this position in a read-only manner.
- **read-write-access:** Overload the *operator[]* such that it takes an index as argument and returns the value at this position such that it can be read from and written to.

In addition, the last two functions should throw an error if the user wants to access an element that does not exist ("out-of-bounds-error").

Once you have correctly implemented all member functions, the program should compile. It corresponds to a little benchmark comparing your vector implementation to std::vector.

Reminder build process

Here we have a CMake project. Assuming you are currently in the folder containing the CMakeLists.txt do

1. mkdir build && cd build
2. cmake ..
3. make

This should create an executable called “vectors1”.