

Lab 1: Dynamic Array

Implement in C++ the given **container** (ADT) using a given representation and a **dynamic array** as a data structure. You are not allowed to use the *vector* from STL or from any other library.

Obs:

- Since your implementation will use dynamic allocation, it is a good practice to implement a destructor, copy constructor and assignment operator as well (even if they are not on the interface).
 - You are allowed to use the function *memcpy* and *realloc*, but you should know that it is not safe to use *memcpy* and *realloc* on memory that was allocated with *new*. Also, if the memory location contains objects, undefined behavior can occur. Your implementation might still work with these functions, but it is not a good practice to use them.
1. **ADT Dynamic Array** – sequential representation, using a dynamic one-dimensional array as data structure.
 2. **ADT Matrix** – represented as a sparse matrix, using a dynamic array of triples <line, column, value> (value $\neq 0$), ordered lexicographically considering the <line, column> of every element.
 3. **ADT Matrix** – represented as a sparse matrix, compressed sparse column representation using dynamic arrays.
 4. **ADT Matrix** – represented as a sparse matrix, compressed sparse line representation using dynamic arrays.
 5. **ADT Bag** – represented using <element, frequency> pairs (or two dynamic arrays).
For example, the bag [5, 10, -1, 2, 3, 10, 5, 5, -5] will be represented as [(5,3), (10, 2), (-1, 1), (2, 1), (3, 1), (-5, 1)]
 6. **ADT Bag** – represented as a dynamic array of frequencies.
For example, the bag [5, 10, -1, 2, 3, 10, 5, 5, -5] will be represented as [1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 3, 0, 0, 0, 0, 2], built in the following way:
 - the interval of values [-5, 10] is translated into the interval [0, 16] of positions
 - at position 0 we have the frequency of -5 (minimum element), on position 1 we have the frequency of -4, ..., on position 15 we have the frequency of 10 (maximum element).
 7. **ADT Bag** – represented as a dynamic array of unique elements (U) and a dynamic array of positions (P) in U of the elements from the Bag.
For example, the bag [5, 10, -1, 2, 3, 10, 5, 5, -5] will be represented as:
U = [5, 10, -1, 2, 3, -5]
P = [0, 1, 2, 3, 4, 1, 0, 0, 5]
 8. **ADT SortedBag** – having elements of type **TComp**, represented using <element, frequency> pairs, sorted using a relation on the elements.

9. **ADT SortedBag** – having elements of type **TComp**, sorted using a relation on the elements.
10. **ADT SortedSet** – having elements of type **TComp**, sorted using a relation on the elements.
11. **ADT Set** – represented as a dynamic array of elements.
12. **ADT Set** – represented as a dynamic array of Boolean values.
13. **ADT Queue** – represented on a circular dynamic array.
14. **ADT Map** – represented as a dynamic array of <key, value> pairs.
15. **ADT Sorted Map** – represented as a dynamic array of <key, value> pairs, sorted using a relation over the keys.