Assignment 9: Sorting the music database

Course 'Imperative Programming' (IPC031)

Make sure to start working on the assignment **before the lab** session, otherwise you will be too late for the deadline

1 Background

In the lecture we discussed four sorting algorithms: insertion sort, selection sort, bubble sort, and heap sort. In this assignment you implement these algorithms on vector-of-El, with `El` the type synonym for `Track` as defined in assignment 8. In order to accomplish this, an ordering relation (`<` and `==`) on `Track` values needs to be implemented (in the bonus assignment you experiment with other variations of the ordering relation). Note that the other comparison operators (`>`, `<=`, `>=`) are defined using `<` and `==` in "`main.cpp`".

2 Learning objectives

After doing this assignment you are able to:

- Use the insertion, selection, bubble, and heap sort sorting algorithms in practice.

3 Assignment

The provided "`main.cpp`" contains the definitions and functions that have been discussed in the lecture. For the functions that need to be implemented the function signatures and pre- and post-conditions are included.

Part 1: Comparing tracks by CD

In the music database "`Tracks.txt`" fields are interpreted as ASCII-texts. As a result, the tracks can be ordered in an unusual way. For example, track 10 of a CD occurs before track 2 of the same CD, as for the strings `"10"` and `"2"`, the ASCII ordering `"10" < "2"` holds.
Define the comparison operators `<` and `==` for `Track` values such that tracks are compared by interpreting their members differently (artist as a string, title of CD as a string, year of recording as a number, track number as a number) and by comparing them following their order relation! As a result, in the example above, in the same CD (same artist, title of CD, year of recording), track number 2 will precede track number 10 (because `2 < 10` in the ordering of integers). As `>`, `<=`, `>=` are based on `<` and `==`, the definition of those operators need not be adjusted.

Part 2: Sorting algorithms

Extend "`main.cpp`" with a vector-of-El (with `El` the type synonym of `Track`) implementation of the sorting algorithms insertion sort, selection sort, bubble sort, and heap sort. Check for each sorting algorithm that the result of sorting satisfies the condition set in Part 1 (within the same CD, track 2 precedes track 10).

Important

Implementing the sorting algorithms of Part 2 can be time consuming. The last algorithm, heap sort, is especially challenging compared to the other algorithms. We therefore urge you to start working on this assignment before the lab session. This allows you to ask questions regarding heap sort during the lab session, which otherwise may not be possible for the simple reason of running out of time.

4 Products

As product-to-deliver you upload to Brightspace:

- "`main.cpp`" that you have created with solutions for each part of the assignment.
- "`main_test.cpp`" that has been extended with non-trivial unit tests for each new function that you have developed in "`main.cpp`".

Deadline

Lab assignment: Friday, November 17, 2023, 23:59h