Datorstrukturer och algoritmer

Laboration 1 Part I

Morgan Nyman

mornym-9@student.ltu.se

**Table of Contents**:

* Problem specification
* User manual
* System description
* The Limits of the Solution
* Problems and reflections

**Problem specification**

Part I of the Lab was to measure the time complexities of the vector versus the linked-list on four different methods, to add- or remove a student from database and to add or remove courses from the student’s database, respectively. To analyze which is better to use when the operations become plenty.

**User manual**

When the executable (.exe) runs you should see a Console window popup somewhere on-screen displaying the operations of part I numbered 1 through 6 where the operations will be the following:

1. Add student to record.

2. Delete student from record.

3. Add course to student.

4. Delete course from student.

5. Print all.

6. Exit.

Where the respective operation for the vector- and linked list-database will run one after the other. Starting with the vector-based list, if you want you could disable one or the other if the test of the operations becomes too impractical. Just open ‘Laboration 1\Part I\main.cpp’ in any text-editor. Comment out line 399 to disable the vector-database, or line 400 for linked-list. Commenting out code is done by preceding a line with to forward slashes (‘//’).

**System description**

At the beginning of main.cpp a std::vector<Student> is defined containing all students of that database and an object LinkedListStudent is also defined. A student-object contains an ID, and Name which is given through the constructor. A std::vector<Course> is defined alongside a LinkedListCourse to all the courses. Each student of the databases has a list with courses. A Course-object contains a Course ID, Course name, Course grade and Course credit, which is set with the definition, and cannot be altered with setters. Setters are method that chance private values of a class-object post-declartion. Meaning after the object has been declared.

**The Limits of the Solution**

The argument input via the console prompt are limited by whitespace and enter, if either are inputed the next text you type will be part of the other argument, inputs also need to be of the correct type every time or the program will fail. This can be extended upon by using printf\_s() instead of std::cin.

**Problems and reflections**

The time spent optimizing the runtime of the legal character filter should be minimal and not altered on numerous occasions, give room for many errors. The Report needed more time than it got, so the conclusion of that is to manage the schedule better. Maybe use a Trello sheet.

**Test runs**

The times chosen are the quickest ones cause the other ones are pruned to have been affected by other processes. Meaning shortest time would be equivalent with least interference. And the slowest of time containing the most interference were to be disposed.

|  |  |  |
| --- | --- | --- |
| Operation | Vector List (µs) | Linked List (µs) |
| Add 10 students | 494 | 176 |
| Add 100 students | 4,678 | 1,826 |
| Add 1000 students | 37,184 | 17,421 |
| - | - | - |
| Remove 10 students | 4,678 | 122 |
| Remove 100 students | 18,136 | 1,127 |
| Remove 1000 students | 1,483,918 | 12,514 |
| - | - | - |
| Add 10 Courses | 316 | 94 |
| Add 100 Courses | 3,206 | 1,465 |
| Add 1000 Courses | 27,459 | 58,048 |
| - | - | - |
| Delete 10 Courses | 154 | 45 |
| Delete 100 Courses | 4,599 | 870 |
| Delete 1000 Courses | 804,138 | 34,692 |