DD2387 Programsystemkonstruktion med C++
Lab 2: The Progressive
21st of november 2015

Introduction

By finishing the assignments present in this document, students will effectively gather extra credit that will affect the student's grade upon finishing the course.

General Requirements

- The *General Requirements* listed in *Lab 2: The Essentials*.
- During the oral presentation of each assignment, the questions associated with such shall be answered verbally.

Submitting and presenting your work

Every assignment requires an oral presentation where you, and your potential partner, shall be able to explain and answer questions regarding your solutions.

Some assignments require you to submit code for automatic testing, which will verify that your implementation is correct in relation to the requirements set forth by the assignment in question.

To submit an implementation for automatic testing, open up a web browser and point it to `https://kth.kattis.com`, then;

- authenticate using your KTH-id, if this is the first time you are using *Kattis* you must register for the service after signing in, also;
- make sure that you register as a student taking cprog15, before you try to submit any of your solutions.

Contents

1 The Progressive (assignments for extra credit)

1.1 Relate, move and reccur 6p

Add the following functionality.

Write the member-function `bool move_event(const Date & from, const Date & to, std::string event)` which can move events in the calendar. If the event was moved succesfully, return `true`. If the event does not exist return `false`

Write the member-function `bool add_related_event(const Date & rel_date, int days, std::string rel_event, std::string new_event)`. An event can relate to an existing event by specifying number of days apart. If the existing event is moved/deleted, the related event should also be moved/deleted. Example, a dinner is scheduled in the calendar. A related event "shop groceries for dinner" is added the day before the dinner. If the dinner is brought forward two days, the related event should also be brought forward and still happen the day before the dinner. A new related event can relate to an existing related event thus forming a chain of related events. Example, "plan what to cook for dinner" the day before "shop groceries for dinner". If the dinner is moved, the two related events are moved as well. An event that already has a related event can not be related to again (the event chain is a single linked list, not a tree).

Write a member-function that adds recurring events. Example, "new years eve" 31st december each year. It shall be possible to specify the amount of times the event reccurs, default is infinite amount of times.

Write the member-function `print_events(const Date & start_date, const Date & end_date)` that writes all of the events between (and including) the dates specified. Use ical format.

Write a member-function that removes all instances of a recurring event.

1.1.1 Additional requirements

- The assignment only specifies what should happen if the head of the event chain is moved. If any other event in the related event chain is moved it is up to you to specify the behavior. Do motivate your design decision.
- If any event in a related event chain is removed, all related events to that event shall be removed. Example, a related event chain of four dinners. If the third dinner is removed, the fourth one will also be removed.
- If any collision occur (the same event already exists) when moving a related event chain, the move should fail and the calendar remain as before.
- A single recurring event may be removed without affecting the other recurring events.
- Adding a related event to a recurring event is not specified in this assignment. Moving a recurring event is not specified in this assignment. Do make a design decision and motivate.

- Be aware so you do not accidentally overwrite your event chain. A related event chain of three dinners two days apart ("dinner"->"dinner"->"dinner") might overwrite each other if moving the event chain two days.
- Show adequate testing of your code. Both correct and errornous behavior.

1.2 Calendar and time (5p)

Implement a calendar that handles events of different durations.

To add an event, in addition to the event name, you will need the start and stop time of the event in question. Removing an event only requires its name and start time (since there can be no overlapping events).

1.2.1 Requirements

- Any two events may not overlap. There shall be no overlap in the calendar. The overlap algorithm must be reasonably efficient.
- It shall be possible to remove an event without giving the start time if it is the only kind of event that day. Example, lunch may be removed without specifying time if there is only one lunch-event that day.
- Printing the calendar in ical-format shoult use DTSTART and DTEND with timestamps.
- Show adequate testing of your code. Both correct and errornous behavior.