Scheduler - Task

Scheduler

Write a scheduler application which runs tasks, each at a specific (absolute) time since boot-up.
Currently, the scheduler will be implemented as a **single-threaded app**, meaning that when it finishes to run all tasks, the program ends.

It will also be a **non-preemptive scheduler**, meaning that each task runs until it is (based on its current cycle) done. You can just use simple prints to emulate each task on interest.

Each **task** should hold the **number of cycles** it should run, and an **interval for the next time** it should be executed in. In other words: Each task "requests" to schedule itself in a certain (absolute) time.

The Tasks should implement the following interface:

```
struct ITask
{
    virtual void run() = 0;
    virtual unsigned long getNextRunPeriod() = 0; // Absolute time since
boot-up (in milliseconds)
};
```

If getNextRunPeriod() returns a value which is greater than zero, then it should be rescheduled for another period.

You should write the following classes:

```
class Scheduler {};    // Runs the tasks
class Time {};    // Accessory for time calculations
```

You should write the API classes and implement them.

Things to consider:

1. Which container will you use in the Scheduler and why?
2. What type of item should be kept in the container?
3. What should the Scheduler do "between" tasks?

Class Time

Class Time calculates times, some of the API suggested:

```
void now() // change the time to now (i.e. absolute current time)
operator<=
operator<
operator>=
operator>
operator+
```

Implement this Scheduler based on a simple **round-robin** policy.

Advanced (I)

Consider the requested time of each task. Maybe you can order them in an optimized manner?

Advanced (II)

Add the possibility to add tasks "on the fly".