

Background

This project is intended to verify certain skills that you should have from CS 1044. This project is not intended to be extremely difficult, but it should give a flavor for the complexity and style of projects that will be assigned in CS 1704. Some of the restrictions placed on this project are simply for purposes of testing requisite knowledge. I am fully aware that there are better ways to implement this project.

For this project, you will implement a scheduling system. You will be given events that take place sometime during one year. The year is unknown until the program runs and therefore all calculations involving when a day occurs in the given year must be computed using:

$$s = (y - 1) + [(y - 1) / 4] - [(y - 1) / 100] + [(y - 1) / 400] + d$$

Day of Week = $s \bmod 7$

y is the year.

d is the sum of days from 1 January of year y up to the date to find (inclusive).

[...] means take the integral value of ...

$A \bmod B$ means take the remainder of A / B .

Day of Week is **Sunday** (if remainder is 0)

Day of Week is **Monday** (if remainder is 1)

Day of Week is **Tuesday** (if remainder is 2)

Day of Week is **Wednesday** (if remainder is 3)

Day of Week is **Thursday** (if remainder is 4)

Day of Week is **Friday** (if remainder is 5)

Day of Week is **Saturday** (if remainder is 6)

Taken from: <http://css.engineering.uiowa.edu/~eng2/Examples/F00/dayofweekcalc.html>

For ease of implementation, we will ignore any complications that leap years may bring up.

I also suggest you come up with a good way to calculate d , the sum of days from 1 January until the date you want to schedule.

Details

You will be given the following commands to implement:

load	will look for and load a file called "Calendar.ini" if it exists
save	will save all scheduled events in a file called "Calendar.ini"
add	will add the given event to the schedule, as long as it doesn't conflict with an already scheduled event
find	will search the schedule for events matching the given description
remove	will remove the given event from the schedule
print	will either print the event information at the given date and time , the given month, or the entire year

`clear` This will clear the current calendar of all its scheduled events

An event will have the following format:

`<Month><whitespace><Date><whitespace><Year><whitespace><Beginning Time><whitespace><Duration><whitespace><Description>`

`<Beginning Time>` is a number between 1 and 24. We will not mess with the minutes.

`<Duration>` is a number indicating the number of hours an event will last. It will not go over day boundaries.

`<Description>` is a string describing the event.

You will need to implement a structure that will contain the following fields: Month, Date, Year, Beginning Time, Duration, Description.

Please notice that there is no field for the day of the week or ending time. Also notice that is a structure and not a class.

The main data structure for this project is a 3 dimensional array. The first dimension should be for the months in the year. The second dimension should be for the days in each month. The final dimension should be for the hours in each day. The data type for this array is to be the same type as your structure.

Additionally, you should use at least one enumerated type. A month enum is suggested, but not required.

You should write functions whenever appropriate and you should not repeat code. For example, you should have a generic print month function that takes a given month and prints out the calendar for that month.

All input should come from standard in and all output should go to standard out. Your program should run until it encounters an end of file marker.

Detailed Description of Commands

`load` – This command should try and open the file called “Calendar.ini”. If it finds the file, it “loads” all of the previously scheduled events into the calendar. If it doesn’t find the file, it should issue some sort of reasonable statement indicating that it was not found. This command will only be issued at the beginning of program execution or after a clear command has been issued.

`save` – This command will take all scheduled events and save them to the file called “Calendar.ini”. The format with which the events are stored in the files is up to you. As long as you are consistent and use a format that is reasonable. The file is not to be binary.

`add` – This command takes the event that is to be scheduled and tries to schedule it. If there is no conflict with any previously scheduled events, then the event is scheduled and a message indicating the success completion of the add command should be issued. If there is

a conflict the event is discarded and a message indicating the conflict is issued. The format for the event is given above.

`find` – This command searches the entire calendar looking for the given event description. If an event is found with a matching description, the event's date, time and duration are outputted. If no events are found a message indicating that no events were found with the matching description. The format for the find command will be
`find<whitespace><description>`

`remove` – This command will use the same format as the add command. If the event is found at the given location, then it is removed. A message indicating the successful removing of the event should be issued. If the event is not found a message indicating that it is not present should be issued.

`print` – This command has three versions. Print with the keyword All followed by an integer, which prints the entire calendar. Print with a month keyword followed by a zero, which prints the given month. Print with a month and date specified, which prints that particular day.

For the printing of the entire year or just one month, the print out should look like a grid of numbers. The numbers are the dates on the calendar. Before each month you should print a header with the name of the month and one for the days of the week. If an event has been scheduled for a particular day, then an "X" should be printed instead of the date. For printing just one day, if the day has events scheduled, you should print each event including its date, time, duration and description. If there are no events scheduled, then you should print an appropriate message.

More on the details for what the months should look like will be forth coming.

`clear` – This command removes all currently scheduled events from the calendar. A message indicating the successful complete of this command should be issued.

Additionally, when a command is given it is to be echoed back to the screen after it is given. Between each command there is to be some sort of divider to separate one command and its output from another.

Restrictions

There are some restrictions placed on this assignment for assessment purposes.

You are not to use any classes.

You are not to use any dynamic data.

No where is there to be a reference to which day of the week a date occurs. Any calculation of days of the week for a given date is to done using the calculation given above.

You must use a struct for the event information. The struct may not have any more fields than those given above.

You must use at least one enum.

You must use a 3-D array of your struct.
You must use one switch statement.

Software Engineering Considerations

Like all programs you write, you will use all of the good naming practices, in code comments, function headers, etc. Additionally, by January 28, 2003, you will submit to the curator a Design Document indicating your design for this project. You are to use the symbols that I went over in class on the second day of lecture. I will post a Word document containing these symbols that you may use. Additionally, you may also use MS Visio to produce this design document. This document should be well thought out and should include all functions and their interfaces. No hand written design documents will be accepted. This must be computer generated.

Submitting

When you submit your final project, you will submit a zip file to the Curator. This project will be archived and the TAs will review your project. In the zip file as well should be an updated design documents to indicate any changes from the original design. This final design should closely conform to your final product. You will have 4 submissions for this first project.

You can submit at <http://eags.cs.vt.edu:8080/curator/EagsCurator>

Pledge

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program:

```
// On my honor:  
//  
// - I have not discussed the C++ language code in my program with  
//   anyone other than my instructor or the teaching assistants  
//   assigned to this course.  
//  
// - I have not used C++ language code obtained from another student,  
//   or any other unauthorized source, either modified or unmodified.  
//  
// - If any C++ language code or documentation used in my program  
//   was obtained from another source, such as a text book or course  
//   notes, that has been clearly noted with a proper citation in  
//   the comments of my program.  
//  
// - I have not designed this program in such a way as to defeat or  
//   interfere with the normal operation of the Curator System.  
//  
// <Student Name>
```

Failure to include this pledge in a submission is a violation of the Honor Code.