KTU
COM 2001: Object Oriented Programming

Assignment 3: A "Student Appointment" Program

Purpose

The purpose of this assignment is to help you learn or review (1) the fundamentals of the C++ programming language, (2) the inheritance and composition concepts, and (3) how to create user space menu based applications.

Rules

Make sure you do coding of your program on your own. You can ask for help from your friends to understand the problem, figure out how to tackle the problem, but you may never copy someones code and present it as your own. If this is detected, you will receive a caution, your code will not be marked. If you copied your work from a friend of yours, you will both receive a caution and your work will not be marked.

Background

It is necessary for the students to regularly meet their supervisors/lecturers to ask questions about their work. Naturally, the lecturers are busy people and can not be found in their rooms most of the time. Hence, it is necessary to create an application that can help students and the lecturers to meet. For this you are required to code a program that can help students take appointments from the lecturers.

The Task

You are required to code a lecturer appointment system for students. In this system, student will have number, name, surname, department, start date, email and (mobile)phone properties. Lecturers will have employee number, name, surname, academic chair, email, phone properties. The appointment will have student, lecturer, date, start and finish times. The relevant information about the students, lecturers and appointments should be read from the relevant files during the program start. The new information input to the program should be save to the files and kept in these files. The student number and employment numbers are unique and can be used to distinguish individuals. The user interface should allow the user to add, list, delete and update students, lecturers and appointments respectively. It should be possible to search for appointments and make sure that the appointments do not overlap for the same users. Furthermore, each users appointments for a given interval should be printed on the screen. In this work you are expected to take advantage of composition and inheritance properties of OOP design.

Functionality

A typical execution of your program from the shell might look like this:

```
$>./randevuSistemi ogrenci.txt akademisyen.txt randevu.txt
```

Step 5: Create a `readme` File

Use `emacs/vi/gedit` to create a text file named `readme` (not `readme.txt`, or `README`, or `Readme`, etc.) that contains:

- Your name, student ID, and the assignment number.
- A description of whatever help (if any) you received from others while doing the assignment, and the names of any individuals with whom you collaborated, as prescribed by the course Policy web page.
- (Optionally) An indication of how much time you spent doing the assignment.
- (Optionally) Your assessment of the assignment: Did it help you to learn? What did it help you to learn? Do you have any suggestions for improvement? Etc.
- (Optionally) Any information that will help us to grade your work in the most favorable light. In particular you should describe all known bugs.

Descriptions of your code should not be in the `readme` file. Instead they should be integrated into your code as comments.

Your `readme` file should be a plain text file. Don't create your `readme` file using Microsoft Word, Hangul (HWP) or any other word processor.

Step 6: Submit

Your submission should include your **project** file, the files that `gcc` generated from it, and your `readme` file. You can do that using moodle platform.

Grading

We will grade your work on two kinds of quality: quality from *the user's* point of view, and quality from *the programmer's* point of view.

From the user's point of view, a program has quality if it behaves as it should.

From the programmer's point of view, a program has quality if it is well styled and thereby easy to maintain. For this assignment we will pay particular attention to rules 1-24. These additional rules apply:

- **Names**: You should use a clear and consistent style for variable and function names. One example of such a style is to prefix each variable name with characters that indicate its type. For example, the prefix `c` might indicate that the variable is of type `char`, `i` might indicate `int`, `pc` might mean `char*`, `ui` might mean `unsigned int`, etc. But it is fine to use another style -- a style that does not include the type of a variable in its name -- as long as the result is a clear and readable program.

- **Comments**: Each source code file should begin with a comment that includes your name, the number of the assignment, and the name of the file.
- **Comments**: Each function -- especially the `main` function -- should begin with a comment that describes *what the function does* from the point of view of the caller. (The comment should not describe *how the function works*.) It should do so by *explicitly* referring to the function's parameters and return value. The comment also should state what, if anything, the function reads from the standard input stream or any other stream, and what, if anything, the function writes to the standard output stream, the standard error stream, or any other stream. Finally, the function's comment should state which global variables the function uses or affects. In short, a function's comments should describe the flow of data into and out of the function.
- **Function modularity**: Your program should not consist of one large `main` function. Instead your program should consist of multiple small functions, each of which performs a single well-defined task.
- **Line lengths**: Limit line lengths in your source code to 72 characters. Doing so allows us to print your work in two columns, thus saving paper.

Special Notes

Sample file formats should be as follows:

ogrenci.txt
123456 Ferdi Besli Bilgisayar 2014 ferdi@abc.com +905051234567 ….

akademisyen.txt
1234 Engin Jurnal Elektronik engin@abc.com +905051234567 Prof.Dr. ….

randevu.txt
123456 1234 12.12.2018 16:00 16:30 ….

KTU

BIL2001: Nesne Yönelimli Programlama

Ödev 3: "Öğrenci Randevu Sistemi" Programı

Amaç

Ödevin amacı (1) C++ programlama dilinin temellerini, (2) miras ve oluşum (inheritance and composition) kavramlarını, (3) menü tabanlı uygulama geliştirmeyi öğrenmeye ve anlamaya yardımcı olmaktır.

Kurallar

Kaynak kodu tek başına kodlamanız beklenmektedir. Problemi anlamak, çözümlemek ve algoritma geliştirmek için arkadaşlarınızdan yardım alabilirsiniz, ancak kaynak kodu başkalarından asla kopyalamayın ve kendi kodunuz gibi sunmayın. Bu tespit edilirse uyarı alırsınız ve kaynak kodunuza 0 puan verilir. Eğer sınıf arkadaşlarınızdan birinin kodunu kopyalarsanız ikiniz de uyarı alırsınız ve ödeviniz 0 puan olarak değerlendirilir.

Arkaplan (Senaryo)

Öğrencilerin ders veren akademisyenler ve/veya danışmanları ile düzenli olarak görüşmeleri önemlidir. Elbette ki akademisyenler yoğun kişiler ve çoğu zaman kendi odalarında bulunamıyorlar. Bu nedenle öğrencilerin akademisyenlerden randevu alabileceği bir yazılım geliştirmek çok önemli. Bunun için sizden öğrencilerin akademisyenlerden randevu alabilecekleri bir programı kodlamanız istenmektedir.

İş Tanımı

Sizden bir üniversitede öğrencilerin akademisyenlerden randevu almasını sağlayan bir sistem geliştirmeniz istenmektedir. Öğrencinin numara, ad, soyad, bölüm, kayıt yılı, e-posta, telefon bilgileri tutulacaktır. Akademisyenin sicil numarasi, ad, soyad, anabilim dalı, e-posta, telefon, ünvan bilgileri tutulacaktir. Randevunun da öğrenci, akademisyen, tarih, başlangıç saati, bitiş saati bilgileri olmalidir. Program başlangıcında ogrenci, akademisyen ve randevu bilgileri dosya[lar] dan okunmali ve program çıkışında dosya[lar] a kaydedilmelidir. Dosya örnekleri aşağıda açıklanmaktadır. Öğrenci numara bilgisi öğrenci için, sicil numarasi bilgisi akademisyen için benzersiz bir bilgidir. Tasarlanan bir konsol ekran menüsü sayesinde sisteme yeni öğrenci ve akademisyen eklenebilmelidir. Randevu için hem öğrencinin hem de akademisyenin randevuları sorgulanmalı ve ayni kisi[ler]e ait takvim çakışması engellenmelidir. Ayrica öğrenci ve akademisyenlerin randevu programı ekrana çıktı olarak verebilmelidir. Sınıf tasarımları yapilirken miras ve olusum (inheritance and composition) mekanizmaları kullanılmalı, DRY (don't repeat

yourself=kendini tekrarlama) ile nesne yönelimli programlama prensipleri ihmal edilmemelidir.

Kullanım

Yazacağınız programın girdileri 3 adet dosya olacaktır. Tipik bir program çalıştırma komutu ve parametreler aşağıdaki şekilde olabilir.

```
$>./randevuSistemi ogrenci.txt akademisyen.txt randevu.txt
```

Kaynak Kod Yazımı ve Gönderimi

- Kaynak kod yazarken kaynak kodun en başına açıklama satırı şeklinde, ad, soyad, numara bilgileri yazılmalıdır.
- Gerekli yerlere açıklama satırları eklenmeli ve kodun okunabilirliğini sağlamak için girintilere ve değişken isimlerine dikkat edilmelidir.
- Ayrıca kaynak kodun görsel açıdan hizalı olmasına dikkat edilmelidir.
- Kodu gönderirken tüm kodları tek bir cpp ya da txt dosyasına yazarak ödev yükleme arayüzüne yüklenmelidir.
- Ödev savunmaları esnasında öğrenciden bu dosyadaki kaynak kodlar kullanılarak çalıştırılabilir duruma getirilmesi ve gerekirse değişiklik yapması istenebilecektir.

Puanlama

Çalışmanız kullanıcı ve programcı olmak üzere iki açıdan değerlendirilecektir.

Kullanıcı açısından değerlendirilirken programın davranışları ve çalışması test edilecektir.

Programcı açısından test edilirken kaynak kodun yazma stili ve bakım kolaylığı göz önüne alınacaktır. Bu açıdan aşağıdaki kurallara uyulması beklenmektedir.

- **İsimlendirme**: Açık ve tutarlı değişken, sınıf ve fonksiyon isimlendirme stili kullanılmalıdır. Örneğin her değişkenden önce türünü ifade eden bir karakter kullanılabilir. char için c, unsigned int için ui kullanılabilir. Ayrıca bakınız: hungarian, camel case, PEP8
- **Açıklamalar**: Her kaynak kod dosya adı, öğrenci ismi, numarası ve dosya düzenleme tarihi (sürüm numarası) gibi bilgileri en başta açıklama satırları şeklinde içermelidir. Her fonksiyonun -- özellikle main() fonksiyonu -- ne iş yaptığı açıklama satırı olarak yazılmalıdır. Parametre ve geri dönüş değerleri hakkında yorum satırları eklenmelidir.
- **Modüler Programlama**: Program çok büyük bir main fonksiyonu içermemelidir. Bunun yerine daha kısa ve belirli bir iş yapan birden çok fonksiyona bölünmelidir. Fonksiyon parametre sayısı makul sayıda olmalıdır. Aynı durum sınıf tasarımında da dikkate alınmalıdır.
- **Satır Uzunlukları**: Kaynak dosyada bir satır uzunluğu 72 karakter ile sınırlı tutulmaldır.
- **OOP**: Kaynak koddaki tasarımda OOP kavramları dikkate alınmalıdır.

Diğer Notlar

Örnek dosya içerikleri aşağıdaki gibi olmalıdır:

ogrenci.txt
123456 Ferdi Besli Bilgisayar 2014 ferdi@abc.com +905051234567 ….

akademisyen.txt
1234 Engin Jurnal Elektronik engin@abc.com +905051234567 Prof.Dr. ….

randevu.txt
123456 1234 12.12.2018 16:00 16:30 ….