Pontifícia Universidade Católica de Minas Gerais

PUC Minas

Disciplinas:	Algoritmos e Estruturas de Dados I
Curso:	Ciência de Dados
Prof.:	Gustavo Soares
Entrega:	09/11/2023
Valor:	5 pontos (AEDs I)

ADAPTADO PARA SEMESTRE 2023 02

Autoria da proposta de trabalho:

Profa. Eveline Alonso Veloso e Prof. Roberto Felipe Rocha

Observações:

- O trabalho poderá ser feito em **grupos de até 3 alunos**.
- Cópias de trabalho receberão nota **ZERO**.
- O programa deve ser feito na linguagem de programação C.
- As informações deverão ser armazenadas em arquivo(s) **acesso direto**, portanto, deverá ser feita leitura e escrita em arquivos.
- O trabalho deverá ser entregue pelo Canvas até o dia **04/12/2022** às **23:59 horas**.
- O grupo deve preparar uma apresentação gravada com a participação de todos os seus componentes. Essa apresentação também deverá ser entregue no Canvas e deve demonstrar todas as funcionalidades do *software*.
- Deverá ser entregue o **projeto completo** do programa, a **documentação**, os **arquivos** contendo os testes realizados e a apresentação gravada.
- Em caso de dúvida, entre em contato com seu professor.

Clínica Viva Bem

Viva Bem é uma clínica médica que tem como principal objetivo atender bem aos seus pacientes. Ela está localizada no centro de Belo Horizonte e possui médicos das seguintes especialidades: cardiologia, dermatologia, e clínica médica. Até hoje a Viva Bem faz seus controles de médico, paciente e consulta em planilhas do Excel e cadernos, o que tem gerado diversos problemas de organização. Sem falar que o controle da quantidade de consultas para cada médico não está sendo feito e muitas vezes, consultas são marcadas e depois têm que ser desmarcadas. Diante desse grande problema vivido pela Viva Bem, a clínica resolveu contratar uma empresa desenvolvedora de *software* (vocês). Sendo assim, é necessário compreender a real necessidade da clínica e desenvolver um *software* específico. A seguir é descrito como deverá ser o *software*, bem como suas restrições.

O sistema

Deseja-se cadastrar os pacientes da clínica, os médicos da clínica e as consultas. As informações que devem ser cadastradas são:

- PACIENTE = código, nome, endereço, telefone e data de nascimento
- MEDICO = código, nome, telefone, especialidade
- CONSULTA = código da consulta, data, hora, código do médico, código do paciente

Considere as seguintes restrições: ** *Não se esqueça de sempre validar essas restrições*

- Para cadastrar uma consulta, primeiro o médico e o paciente devem estar cadastrados.
- As consultas devem ser marcadas com intervalos de 30 minutos entre elas.
- Para cada dia podem ser realizadas, no máximo, duas consultas para cada médico.

1. Implemente uma função para cadastrar um paciente. Essa função deve garantir que não haverá mais de um paciente com o mesmo código. Se quiser pode gerar o código do paciente automaticamente.

2. Implemente uma função para cadastrar um médico. Essa função deve garantir que não haverá mais de um médico com o mesmo código. Se quiser pode gerar o código do médico automaticamente.

3. Implemente uma função que cadastre uma consulta. Para cadastrar a consulta, o sistema deve receber o CÓDIGO do paciente que deseja se consultar e o CÓDIGO do médico. Lembre-se que cada médico só pode atender, no máximo, duas consultas por dia.

4. Implemente uma função que permita cancelar uma determinada consulta.

5. Implemente os seguintes relatórios:
   a) Receba uma data e mostre na tela todas as consultas daquele dia.
   b) Receba o nome OU o código de um paciente e mostre suas consultas já realizadas até a data corrente.

6. Implemente uma função que mostre na tela todas as consultas de um determinado médico (a consulta poderá ser realizada pelo nome OU pelo código do médico).

7. Implemente uma função extra, criada pelo grupo. Sejam criativos!

Para fazer esse programa pode ser necessário criar mais funções do que as que estão descritas. Finalmente, faça uma função *main*() que teste o *software* descrito acima. A função *main*() deve exibir um *menu* na tela, com as opções de cadastrar um paciente, um médico e uma consulta. Além disso, permitir realizar as pesquisas. Esse *menu* deve ficar em *loop* até o usuário selecionar a opção SAIR. Além disso, todas as informações deverão ser persistidas/armazenadas em arquivo(s) texto. Portanto, deverá ser feita leitura e escrita em arquivos.

Metodologia

Este é um trabalho interdisciplinar em que você deve planejar, analisar, projetar, implementar e testar uma solução de *software* para o problema apresentado utilizando o Scrum para gerenciar seu progresso.

Inicialmente organize o *backlog* do produto com as funções básicas do sistema. Cada uma das funções será de responsabilidade de um membro do grupo e será desenvolvida em *sprints* de 3 a 4 dias. Seguem algumas sugestões de atividades a serem realizadas nas *sprints*:

1- Definir a assinatura da(s) função(ões). Reflita sobre os parâmetros de entrada e saída da função e comunique aos seus colegas de projeto.

2- Documentar a função indicando seu propósito, e os parâmetros de entrada e saída. O nome da função deve ser escolhido sob o ponto de vista de quem usa a função ou de quem vai chamar a função e deve refletir o que a função faz.

3- Implementar o caso de sucesso da função.

4- Selecionar casos de teste para verificar o funcionamento da função. Um caso de teste deve conter os valores de entrada para a função e a saída esperada.

5- Executar os casos de teste planejados para a função. Inicie fazendo a execução manual de alguns poucos casos de teste. Em seguida implemente a automatização dos testes da função usando a biblioteca *munit*.

6- Criar um relatório de execução de testes que contenha os casos de teste, a saída retornada durante sua execução e uma indicação se a função passou ou não no teste. Isso é feito comparando-se a saída esperada, documentada no caso de teste, com a saída retornada durante a execução da função (esperado x real).

7- Implementar os casos especiais, exceções que possam existir na função. Em seguida, executar os casos de teste anteriores para garantir que as mudanças não quebraram o código anterior que já funcionava. Pense também nos novos casos de teste necessários para a nova versão da função.

O que deve ser entregue para os professores no Canvas

1- A evolução do *backlog* do produto a cada semana. Indique quais tarefas encontravam-se inicialmente no *backlog* do produto, e em qual *sprint* cada tarefa foi alocada, juntamente com seu responsável.

2- A documentação das funcionalidades do *software*.

3- O planejamento dos casos de teste (entradas, procedimento de teste e saídas esperadas), a implementação dos casos de teste automatizados e o relatório de execução dos testes.

4- O código, em C, das funções e do programa principal, juntamente com o projeto completo do *software*.

5- Arquivos contendo dados já incluídos para teste das funcionalidades.

6- Apresentação gravada em vídeo (*pitch*) mostrando todas as funcionalidades do sistema.

Link para a biblioteca munit: https://nemequ.github.io/munit/