Universidade Federal do Espírito Santo
Departamento de Informática

1ª Trabalho Prático

Programação II - 2023/1
11 de Maio de 2023

Objetivo

O objetivo deste trabalho é colocar em prática as habilidades de programação na linguagem C adquiridas ao longo do curso. O trabalho foi elaborado para que você exercite diversos conceitos principalmente alocação e manipulação de memória de maneira dinâmica. Como consta no plano de ensino da disciplina, **o trabalho equivale a 30% da nota final do curso**.

Introdução

A extensão universitária é uma atividade complementar à formação acadêmica oferecida pelas universidades e instituições de ensino superior. Ela tem como objetivo estender os conhecimentos gerados no ambiente acadêmico para a sociedade em geral, promovendo a interação entre a instituição de ensino e a comunidade. A UFES possui centenas de projetos de extensão, um deles (que vocês já conhecem) é o Programa de Assistência Dermatológica (PAD-UFES), oferece atendimento gratuito à população Capixaba para tratamento de lesões de pele, desde a triagem até a cirurgia, se necessário. Para digitalizar todo o atendimento dos pacientes atendidos pelo PAD, em 2021 foi criado o PADTech, o núcleo de tecnologia do PAD na qual um dos objetivos é desenvolver e manter um sistema de coleta, armazenamento e análise de todos os dados obtidos durante um atendimento do PAD. Atualmente, o PADTech é um projeto de extensão ligado ao departamento de Informática e que possui bolsistas e voluntários que trabalham na criação do Software de Análise Dermatológica (SADE).

Sendo assim, neste trabalho, sua missão **é implementar um protótipo do SADE** (na qual chamaremos de mini-SADE) que tem como objetivo realizar algumas das tarefas que são necessárias para habilitar um atendimento dos profissionais de saúde

associados ao PAD-UFES. Vale destacar que o trabalho abrange apenas uma pequena parte do sistema que está em produção. Para aqueles que tiverem curiosidade do funcionamento completo, o PADTech funciona como um projeto associado ao Laboratório de Inteligência Artificial em Saúde, no CT-XIII na sala 27.

Descrição

A intenção principal do mini-SADE é gerenciar digitalmente o atendimento de pacientes em uma clínica fictícia. Para isso, vamos definir o seguinte fluxo que o programa deve atender:

- Um paciente precisa ser pré-cadastrado por algum agente externo (ex: um secretário) para realizar uma consulta
- Somente pacientes pré-cadastrados podem realizar consultas
- Durante a consulta, informações clínicas do paciente serão preenchidas
- Uma consulta pode gerar uma ou mais lesões relacionadas a um paciente
- Uma lesão pode ser um caso simples, sem necessidade de um tratamento, ou pode ser encaminhada para crioterapia ou cirurgia
- Ao final da consulta, os dados devem ficar salvos em um banco de dados
- Profissionais de saúde devem ser capazes de buscar um determinado paciente no banco de dados
- Profissionais de saúde podem solicitar a criação de um relatório relacionados ao banco de dados

Portanto, podemos concluir que o mini-SADE deve ser capaz de fornecer as seguintes funcionalidades:

1. Pré-cadastro de pacientes
2. Realização de uma consulta
    a. Atualização de dados clínicos do paciente
    b. Cadastro de lesões do paciente
3. Busca de paciente
4. Criação de relatório

Na sequência, será descrito cada uma destas funcionalidades.

Funcionalidade 1: pré-cadastro de pacientes

A primeira funcionalidade do mini-SADE é o pré-cadastro de pacientes. Nesta etapa, são coletadas informações básicas de um paciente que serão salvas no banco de dados para serem usadas durante uma consulta. **Não é possível realizar uma consulta sem o pré-cadastro do paciente.**

Sendo assim, nesta etapa são coletadas as seguintes informações:

- Nome do paciente
- Cartão do SUS (formato: 000-0000-0000-0000)
- Data de nascimento
- Telefone (formato: (00)00000-0000)
- Gênero (masculino, feminino, outros)

O cartão do SUS é o identificador único do paciente. Portanto, o sistema deve garantir que não seja possível cadastrar mais de um paciente com o mesmo número do SUS. Se houver tentativa de realizar a operação, o sistema deve informar o usuário que não é possível pois já existe paciente cadastrado com o mesmo número.

Funcionalidade 2: realização da consulta

Um profissional de saúde deve ser capaz de realizar uma consulta de um paciente pré-cadastrado. Para isso, a consulta inicia com o profissional digitando um número de cartão do SUS. Após digitar o número, pode ocorrer duas situações:

1. **O número do SUS <u>não</u> está cadastrado**: neste caso, a consulta não será iniciada e uma mensagem deve ser exibida na tela informado que o paciente precisa ser pré-cadastrado. O fluxo do programa deve voltar para o menu de início (o menu será descrito na próxima seção)
2. **O número do SUS está cadastrado:** neste caso, o sistema deve exibir na tela o nome e a idade do paciente e a consulta pode ser iniciada.

Na primeira etapa da consulta o profissional coleta as seguintes informações do paciente:

- Se o paciente possui diabetes
- Se o paciente é fumante
- Se possui alergia a algum medicamento. Se sim, informar o nome qual.
- Se o paciente possui histórico de câncer
- Qual o tipo de pele do paciente
    - Valores possíveis: (I, II, III, IV, V, VI)

Coletada essas informações, o profissional deve ser capaz de cadastrar uma ou mais lesões relacionadas ao paciente. Uma lesão possui os seguintes atributos:

- Rótulo (L1, L2, L3, ..., Ln)
    - Deve ser incrementado automaticamente à medida que as lesões são adicionadas ao paciente
- Diagnóstico clínico
    - Valores possíveis: Carcinoma Basocelular, Carcinoma Espinocelular, Melanoma, Ceratose Actínica, Nevo e Ceratose Seborréica

- Região do corpo
    - Valores possíveis: Face, Orelha, Couro cabeludo, Peitoral, Dorso, Abdome, Braço, Antebraço, Mão, Coxa, Canela e Pé,
- Tamanho (em milímetros)
- Se a lesão deve ser enviada para cirurgia
- Se a lesão deve ser enviada para crioterapia

Ao final do cadastro de todas as lesões, o atendimento deve ser encerrado. Todo atendimento deve ser registrado em uma pasta de logs. Um log nada mais é do que um arquivo com nome `log_<id>` (exemplo: `log_1`) que armazena o número do cartão do SUS do paciente e os rótulos das lesões criadas no atendimento. Exemplo de arquivo de log:

```
141-5263-5248-8987
L1
L2
L3
```

O número do `id` do log deve ser gerado automaticamente pelo sistema de acordo com o número de atendimentos. Informações sobre onde salvar o documento serão descritas na seção **Funcionamento do Programa**.

Funcionalidade 3: busca de paciente

O sistema mini-SADE deve permitir que o usuário busque por um paciente cadastrado na base de dados. A busca é realizada através do cartão do SUS. Se o paciente **não** existir, o sistema deve exibir uma mensagem na tela informando que o mesmo não está presente na base dados. Por outro lado, se existir, o sistema deve exibir na tela uma mensagem falando que encontrou e que gerou o arquivo de resultado de busca.

O arquivo de resultado de busca nada mais é do que um documento com nome `busca_<numero_sus>` (exemplo: `busca_141-5263-5248-8987`) que contém todas as informações do paciente (exceto o SUS, que já está no nome do arquivo), incluindo todas as lesões que ele possui. Um exemplo de documento é mostrado a seguir:

```
NOME: CASSIAN ANDOR
DATA DE NASCIMENTO: 19/11/1990 (32 ANOS)
GENERO: MASCULINO
TELEFONE: (27)99898-3131
```

```
DIABETES: SIM
FUMANTE: NAO
ALERGIA A MEDICAMENTO: NAO
HISTORICO DE CANCER: NAO
TIPO DE PELE: III

LESOES:
TOTAL: 3
ENVIADA PARA CIRURGIA: 1
ENVIADA PARA CRIOTERAPIA: 1

DESCRICAO DAS LESOES:
L1 - CARCINOMA ESPINOCELULAR - FACE - 30MM - ENVIADA PARA CIRURGIA
L2 - CERATOSE ACTINICA - ANTEBRACO - 90MM - ENVIADA PARA CRIOTERAPIA
L3 - CERATOSE SEBORREICA - FACE - 40MM
```

Cada busca deve resultar em um documento. Se o documento já existir, ele deve ser sobrescrito. Se a informação não foi preenchida, deve ser impresso uma string vazia. Informações sobre onde salvar o documento serão descritas na seção **Funcionamento do Programa**.

Funcionalidade 4: criação de relatório

A última funcionalidade que o mini-SADE deve disponibilizar é a criação de um relatório dos dados coletados ao longo dos atendimentos. Este relatório é consolidação geral do banco de dados e deve fornecer as seguintes informações:

- Número total de pacientes **que possui pelo menos um atendimento**
- Média e desvio padrão da idade dos pacientes
    - Utilizando somente a parte inteira
- Distribuição percentual dos pacientes por gênero
    - Na ordem: feminino, masculino e outros
- Tamanho médio e desvio padrão das lesões
    - Utilizando somente a parte inteira
- Número total de lesões
- Número total de cirurgias
    - Incluindo porcentagem em relação ao número total
- Número total de crioterapias
    - Incluindo porcentagem em relação ao número total
- Distribuição absoluta e percentual dos diagnósticos clínico
    - Utilizando ordenação decrescente de acordo com o número absoluto
    - Se houver empate, usar ordem alfabética

Um exemplo do relatório é mostrado a seguir:

```
NUMERO TOTAL DE PACIENTES ATENDIDOS: 50
IDADE MEDIA: 61 +- 5 ANOS
DISTRIBUICAO POR GENERO:
- FEMININO: 45.52%
- MASCULINO: 53.48%
- OUTROS: 1.00%
TAMANHO MEDIO DAS LESOES: 18 +- 5 MM
NUMERO TOTAL DE LESOES: 90
NUMERO TOTAL DE CIRURGIAS: 47 (52.22%)
NUMERO TOTAL DE CRIOTERAPIA: 30 (33.33%)
DISTRIBUICAO POR DIAGNOSTICO:
- CERATOSE ACTINICA: 40 (44.44%)
- CARCINOMA BASOCELULAR: 15 (16.66%)
- CARCINOMA ESPINOCELULAR: 15 (16.66%)
- NEVO: 10 (11.11%)
- CERATOSE SEBORREICA: 8 (8,88%)
- MELANOMA: 2 (2,22%)
```

O nome do relatório deve se chamar `relatorio_final` e informações sobre onde salvar o documento serão descritas na seção **Funcionamento do Programa**.

Fluxo do programa

Ao iniciar seu programa o mesmo deve oferecer um menu que solicita ao usuário qual operação ele deseja executar. Você é livre para escolher o layout deste menu, desde que respeite os caracteres que representam cada ação. Sendo assim, o primeiro menu deve apresentar as seguintes opções:

```
######################### MENU INICIAL ############################
Escolha uma opcao:
- Pre-cadastrar um paciente (P ou p)
- Iniciar um atendimento (A ou a)
- Buscar um paciente (B ou b)
- Gerar relatorio (R ou r)
- Finalizar programa (F ou f)
###################################################################
```

Ao escolher a opção de pré-cadastro, o programa deve ler da entrada padrão todos os dados desta etapa de acordo com a mesma sequência fornecida na seção de Funcionalidade 1. A mesma lógica deve ser seguida para a opção de iniciar um

atendimento. Porém, neste caso, o atendimento pode receber mais de uma lesão. Sendo assim, **o atendimento é encerrado se o usuário digitar o caractere E ou e.** Feito isso, o fluxo do programa deve retornar para o menu inicial.

Para buscar o paciente e gerar o relatório, o programa deve escrever os arquivos solicitados nas seções que descrevem estas funcionalidades. Sempre que a funcionalidade encerrar, o programa deve voltar para o menu inicial. Obviamente, ao selecionar a opção de finalização, o programa deve ser encerrado e toda memória alocada será liberada.

Funcionamento do programa

Seu programa deve ser escrito na linguagem C e **obrigatoriamente**, a função principal deve estar em um arquivo chamado `trab1.c`. Você pode criar quantos TADs e bibliotecas julgar ser necessário. Por conta disso, você deve fornecer o arquivo `makefile` do seu programa. Para mais informações sobre este arquivo, leia este tutorial. Seu `makefile` deve gerar um executável para Linux de nome `trab1`.

Após compilado, seu programa deve receber um argumento de entrada através da linha de comando (para mais informações, leia este tutorial). Este argumento vai direcionar a saída é o caminho para uma pasta de saída, local na qual seu programa deve escrever os arquivos solicitados em cada uma das funcionalidades. Exemplo de chamada do programa:

```
./trab1 /home/user/saida
```

Isso significa que todos os arquivos de saída do seu programa devem ser escritos dentro do diretório `/home/user/saida`. Isso é obrigatório para que a correção funcione corretamente.

A entrada de dados do programa será realizada utilizando a entrada padrão. Isso significa que a leitura pode ser realizada tanto via teclado quanto usando o terminal do Linux com o comando **<** . Exemplo de chamada do programa:

```
./trab1 /home/user/saida < dados_entrada
```

De acordo com as especificações das funcionalidades, seu programa deve ser capaz de gerar três tipos de documentos:

- `log_<id>`
- `busca_<numero_sus>`

- `relatorio_final`

Dentro do diretório de saída informada como argumento de entrada do programa, deve ser criado uma pasta para cada tipo de documento a ser gerado. Os documentos de log serão salvos em uma pasta chamada `logs`. Os documentos de busca em uma pasta chamada `buscas`. E o relatório final dentro de uma pasta chamada `relatorio`. Sendo assim, considerando que a pasta de saída seja /home/user/saida, um exemplo do diretório final do programa é ilustrado na sequência:

- `/home/user/saida/logs`
  - `log_1`
  - `log_2`
  - `log_3`
- `/home/user/saida/buscas`
  - `busca_137-8569-8974-8256`
  - `busca_112-9654-6325-7854`
- `/home/user/saida/relatorio`
  - `relatorio_final`

Regras gerais

Para todas as etapas do trabalho, considere as seguintes regras:

- Nenhuma entrada dados nos casos de teste vai possuir acentos ou caracteres especiais. Por exemplo, João será sempre Joao.
- Em todos os arquivos de saída você deve escrever sempre com letras maiúsculas. Por exemplo, se o paciente se chama Joao, na saída dele deve constar como JOAO.
- Todos os arquivos de entrada vão seguir sempre o mesmo padrão, incluindo o nome dos arquivos.
- Para dados de ponto flutuante, use sempre precisão simples. Para a escrita destes dados, use apenas duas casas decimais.
- **Não** é permitido o uso de bibliotecas externas além das que já foram trabalhadas em sala de aula, ou seja, `stdio.h`, `stdlib.h`, `string.h`, `time.h`, etc
- Para calcular a idade dos pacientes considere a data de referência como sendo 11/05/2023 (dia em que o trabalho foi divulgado)
- Os dados devem ser alocados na memória na medida que aparecem para ser cadastrados. Isso deve ser feito com **alocação dinâmica**. Trabalhos que utilizem alocação estática para este fim, serão desconsiderados
- É obrigatório o uso de pelo menos um TAD totalmente encapsulado (opaco). Lembre-se, quanto mais modularizado, mais fácil vai ser a modificação para

inclusão da(s) nova(s) features que deverão ser implementadas em sala de aula
- Liberar a memória alocada é de sua responsabilidade. O seu programa será testado utilizando o Valgrind. Haverá descontos significativos da nota para erros e vazamentos de memória apontados pela ferramenta
- **Possíveis problemas na descrição do trabalho serão solucionados e comunicados via Classroom. Don't Panic!**

Correção do trabalho

O seu trabalho será corrigido de maneira automática em um terminal Linux. Seu programa será compilado usando o `makefile` que você deve disponibilizar. Na sequência ele será executado da seguinte forma:

```
./trab1 /home/user/saida < dados_entrada
```

Após a execução, os arquivos gerados dentro do diretório de saída (informado como parâmetro) serão comparados com as respostas reais. **É muito importante que você siga rigorosamente as instruções para que o teste não falhe.**

Além do teste automático, seu código será avaliado utilizando os seguintes critérios:
- Organização
- Modularização
- Criação de TADs
- Gerenciamento e manipulação adequados da memória

Em outras palavras, não basta apenas acertar todos os testes é necessário cumprir todos os pré-requisitos desta especificação.

A correção possui duas etapas. Na primeira, seu código será testado e avaliado segundo os critérios já estabelecidos. A segunda etapa é a implementação de novas funcionalidades que serão informadas no dia da entrega do trabalho e deverá ser realizada obrigatoriamente em sala de aula. A intenção desta etapa é verificar o seu conhecimento para com o código e forçar a implementação modularizada e organizada da solução. A nota final será a média ponderada entre as duas etapas. Além disso, todos os trabalhos serão testados em relação a plágio, que terão tolerância zero.

Se você não possui o sistema operacional Linux, **garanta que seu código funcione nos computadores do Labgrad**. Qualquer discrepância de resultado na correção por

conta de Sistema Operacional, os computadores do Labgrad serão utilizados para solucionar o problema.

Prazo e submissão

O trabalho deve ser submetido até **23h:59min do dia 04 de junho de 2022** utilizando um repositório na organização do Github da disciplina. Você deve criar esse repositório **privado** seguindo o seguinte padrão: `trab-1-<seu_nome>-<seu-sobrenome>`. Exemplo: `trab-1-andre-pacheco`.

Se porventura você atrasar a entrega do trabalho, será descontado 40% da nota final a cada dia de atraso.