Universidade Federal do ABC
Programação Estruturada – 3º Quadrimestre 2024
Prof. Carlos da Silva dos Santos

Exercícios de Programação para Entregar

Este documento apresenta o enunciado de dois exercícios de programação que devem ser enviados via Moodle em atividade específica, até **21/12/2024**. Na implementação dos exercícios abaixo, **você pode utilizar trechos de código** de exemplos fornecidos durante o quadrimestre e de exercícios anteriores que você tenha resolvido. Caso você utilize código exemplo fornecido pelo docente durante as aulas de teoria ou laboratório, você deve identificar a origem por meio de comentário no início do arquivo, como no exemplo ilustrativo abaixo:

```
/*
Este arquivo contém trechos de código derivados dos arquivos
xxxx.c e xxxx.h fornecidos na aula de laboratorio de xx/xx/xx
via Moodle.
Este arquivo contém trechos de código apresentados em sala
de aula e que podem ser encontrados no arquivo yyyy.pdf
fornecido via Moodle.
*/
```

O primeiro exercício aborda as habilidades abaixo (**lista A**):

- Declarar tipos e variáveis.
- Utilizar estruturas de controle condicionais.
- Utilizar estruturas de repetição.
- Declarar e utilizar funções.
- Utilizar vetores em programas.

O segundo exercício aborda todas as habilidades listadas acima **e também** as que aparecem na **lista B** abaixo:

- Ler e escrever dados em arquivos.
- Implementar e utilizar busca linear.

- Declarar e utilizar novos tipos de dados.
- Gerenciar memória manualmente utilizando ponteiros.
- Implementar e utilizar listas ligadas simples.
- Implementar e utilizar algoritmos de ordenação.
- Organizar código em módulos (bibliotecas).

Você pode entregar ambos os exercícios ou apenas um deles.

- Caso você entregue somente o primeiro exercício, sua avaliação contemplará apenas a lista A de habilidades.
- Caso você entregue somente o segundo exercício, sua avaliação contemplará a lista A e a lista B de habilidades.
- Caso você entregue ambos exercícios, sua avaliação contemplará a lista A e a lista B de habilidades. Para as habilidades cobertas por ambos exercícios, a avaliação das habilidades vai considerar o melhor caso. Por exemplo, caso você utilize vetores corretamente no exercício 1, isso contará como evidência desta habilidade, mesmo que o exercício 2 contenha erros relativos ao uso de vetores.

Cada exercício pode ser **aceito** ou **não aceito**. Os critérios de aceitação serão listados junto com o enunciado de cada exercício. Caso algum de seus exercícios não seja aceito, você terá a chance de enviar um versão corrigida durante o período de reposição de janeiro/2025. Para quem não conseguir enviar os exercícios até 21/12/2024, um segundo período de envio será aberto também durante a reposição de aulas.

Modo de Entrega

Você deve entregar todos os arquivos relacionados à resolução dos exercícios como um único arquivo compactado com extensão .zip (por favor, não utilize outro formato, como .rar). Depois de descompactado, cada exercício deve aparecer em uma pasta própria. Em cada exercício, deve ser possível compilar o programa-resolução com o comando:

```
> gcc -o programa [todos os arquivos com extensão .c da pasta]
```

Em outras palavras: a pasta de cada exercício deve conter todos os arquivos `.c` ou `.h` relevantes para a compilação; não deve haver nenhuma outra subpasta contendo código necessário para compilar o programa; e a pasta não deve conter arquivos de código além daqueles necessários. Em especial, não deve haver mais de uma função `main` por exercício. A pasta pode conter outros arquivos descritivos sobre o programa ou de entrada/saída.
Em cada exercício, a pasta deve trazer também um arquivo chamado `leiame.txt` contendo:

- Identificação (nome e RA) da/do discente.
- Identificação de qual arquivo contém a função principal (`main`) do programa.
- Instruções para a compilação do programa. Preferencialmente, incluir um comando de compilação que possa ser copiado e colado em uma *shell* para compilar o programa. Caso você opte por entregar um `Makefile`, apresente no arquivo `leiame.txt` os comandos do `make` relevantes.
- Adicione ao arquivo `leiame.txt` qualquer informação adicional que seja relevante para a compilação e execução do programa.

Exercício 1

A *mediana* de uma série ordenada de k números $a_1 \leq a_2 \leq \ldots \leq a_k$, em que k **é ímpar**, é definida como o elemento de índice central, ou seja, $a_{(k+1)/2}$. Em processamento digital de sinais, um *filtro de mediana* de tamanho k é um procedimento que recebe uma série numérica $[x_1, x_2, x_3, \ldots]$ e produz uma saída $[y_1, y_2, y_3, \ldots]$ em que cada y_i é obtido pelo cálculo da mediana sobre uma vizinhança de tamanho k do sinal de entrada. Note que tipicamente o sinal de entrada **não é uma sequência ordenada**. Filtros de mediana são empregados para eliminar elementos *discrepantes* de um sinal (muito baixos ou muito altos), resultando em um sinal de saída suavizado.
Neste exercício, você deve construir um programa que recebe uma sequência de valores inteiros **não negativos** a partir do teclado, terminada por zero. O programa deve produzir como saída uma sequência de valores inteiros obtidos pelo cálculo da mediana dos valores de entrada, usando uma janela de tamanho 3 (ver exemplo ilustrativo abaixo).
A Figura 1 mostra a construção da sequência de saída para a entrada `138, 217, 39, 209, 22, 61`. Uma janela de tamanho 3 é posicionada sobre os

elementos da entrada, a partir do início. A mediana é calculada sobre os três elementos englobados pela janela. O resultado do cálculo é enviado para a saída. A janela é então deslocada em uma posição e o processo é repetido.

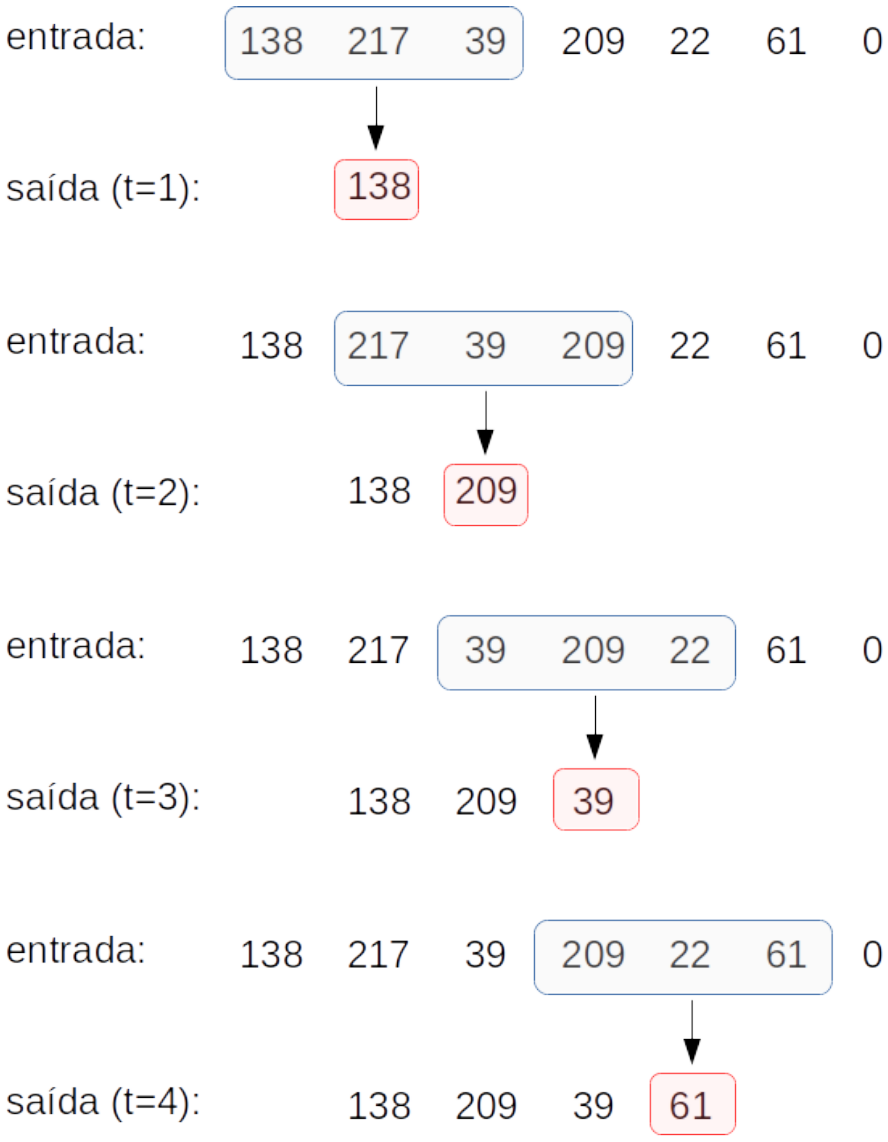

Figura 1: Ilustração do cálculo da mediana em quatro instantes de tempo consecutivos ($t = 1$ até $t = 4$). O retângulo azul representa a janela que é deslocada sobre a entrada. A cada instante, a janela é posicionada sobre uma sequência de três elementos da entrada. O valor mediano é transportado para a saída. A janela é deslocada e o processo se repete.

O programa entregue deve atender aos seguintes pontos:

- Seu programa deve ler dados a partir do teclado e apresentar o resultado na saída padrão.

- A sequência de entrada é encerrada com um valor nulo (zero).
- Você pode supor que a sequência de entrada sempre terá pelo menos três elementos não nulos.
- Se a sequência de entrada tiver n elementos, a sequência de saída terá comprimento $n - 2$.
- Seu programa deve imprimir cada elemento da sequência de saída em sua própria linha. Use a instrução `printf("%d\n", mediana);` para imprimir cada valor de saída, em que `mediana` é uma variável inteira que guarda o valor a ser impresso.

A listagem abaixo demonstra um exemplo de entrada fornecida para o programa via teclado:

```
10
20
7
12
5
0
```

A listagem demonstra a saída esperada para a sequência de entrada anterior:

```
10
12
7
```

Via Moodle, você receberá pares de arquivo entrada/saída que poderá usar para testar seu programa via redirecionamento de entrada e saída.

Critérios de aceitação:

- O código enviado deve estar sintaticamente correto e compilar sem erros.
- As instruções de compilação devem aparecer no arquivo `leiame.txt` entregue junto com o exercício.
- O cálculo da mediana deve ser implementado por meio de uma função específica, que devolve o valor da mediana dos parâmetros de entrada. Você não pode implementar o algoritmo para calcular a mediana diretamente na função `main`.
- Após a leitura a partir do teclado, os dados de entrada devem ser mantidos em um vetor (note que somente três valores de entrada precisam ser guardados simultaneamente).

- O programa deve ler os dados de entrada a partir do teclado.
- O programa deve imprimir cada elemento da sequência de saída em uma linha própria (dois elementos não podem ser impressos na mesma linha).
- O programa deve calcular corretamente a sequência de saída, de acordo com a especificação detalhada acima.

Exercício 2

Neste exercício você vai emular parte da funcionalidade de uma agenda de contatos. Seu programa vai operar em duas fases. Na primeira fase, o programa vai realizar operações de inserção (I) ou remoção (R) sobre uma lista de números de telefone, lidos a partir de um arquivo. O programa deve executar as operações, mantendo em memória uma lista encadeada de números de telefone. Na segunda fase, o programa deve copiar os números de telefone contidos na lista para um vetor alocado de maneira dinâmica (função `malloc`) e ordená-los. Em seguida, o programa deve escrever os números de telefone, de maneira ordenada, em um arquivo de saída.

Cada número de telefone é representado por um DDD (dois dígitos) e o número propriamente dito (8 dígitos). Para ordenar os números de telefone, você deve considerar primeiro o DDD (por exemplo, todos os telefones com DDD 12 devem aparecer antes dos telefones de DDD 22). Quando dois telefones tiverem o mesmo DDD, eles devem ser ordenados pela ordem lexicográfica ("de dicionário") do número de telefone (p.ex. "44442229" precede "44443330"). Os nomes do arquivo de entrada e do arquivo de saída serão fornecidos pelo usuário, via teclado.

Você pode utilizar o código fornecido durante as aulas como ponto de partida para implementar a lista encadeada e o algoritmo de ordenação. Você pode escolher o método de ordenação, entre aqueles discutidos durante as aulas.

O arquivo de entrada do programa tem o seguinte formato: a primeira linha contem a quantidade de operações, como um número inteiro. A partir da segunda linha, são listadas as operações. A primeira linha da operação contém apenas um caractere especificando o tipo da operação, `I` para inserção e `R` para remoção. A segunda linha da operação contém o DDD do telefone como uma sequência de 2 dígitos. A terceira e última linha da operação contém o número do telefone, como uma sequência de 8 dígitos.

O arquivo de saída do programa tem o seguinte formato: a primeira linha

contem a quantidade de telefones, como um número inteiro. A partir da segunda linha, são listados os telefones. A primeira linha do telefone contém o DDD como uma sequência de 2 dígitos. A segunda linha do telefone contém o número, como uma sequência de 8 dígitos.
A listagem abaixo demonstra um exemplo de arquivo de entrada:

```
4
I
11
55550000
I
11
22223333
R
11
55550000
I
12
22226060
```

A listagem demonstra o arquivo de saída esperado para o arquivo de entrada anterior:

```
2
11
22223333
12
22226060
```

Via Moodle, você receberá exemplos de arquivos de entrada/saída para avaliar seu programa.

Critérios de aceitação:

- O código enviado deve estar sintaticamente correto e compilar sem erros.
- As instruções de compilação devem aparecer no arquivo `leiame.txt` entregue junto com o exercício.
- O programa deve receber os nomes do arquivo de entrada e do arquivo de saída a partir do teclado.
- O programa deve ler os dados a partir do arquivo de entrada e escrever o resultado no arquivo de saída, de acordo com a especificação anterior.

- O programa deve usar uma lista encadeada para manter os dados lidos a partir do arquivo de entrada e executar as operações de inserção e remoção.
- O programa deve usar um vetor alocado de maneira dinâmica para ordenar os números de telefone, após a fase de inserção/remoção de dados.
- Toda memória alocada de maneira dinâmica deve ser liberada (função `free`) antes do final do programa.
- O algoritmo de ordenação deve ser implementado como uma função específica (isto é, você não deve implementar o algoritmo de ordenação na função `main`).
- As funções para criar e gerenciar a lista encadeada devem ser implementadas em uma biblioteca separada do arquivo que contém a função `main`. Esta biblioteca deve ser implementada com separação entre cabeçalho (arquivo .h), contendo as declarações principais, e implementação (arquivo .c).
- O código entregue deve conter as implementações do método de ordenação e da lista encadeada. Você não pode usar bibliotecas externas para estas funcionalidades.