Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estruturas de Dados 2020/1
Prof. Luiz Chaimowicz, Raquel O. Prates
Monitores: Álisson R. Svaigen e Michele B. Pinheiro

Trabalho Prático 1: O problema da frota intergaláctica do novo imperador

Valor: 10 pontos
Data de entrega: 24 de setembro de 2020

Introdução

Num universo cinematográfico paralelo da saga Star Wars, o Império derrotou a Aliança Rebelde de uma vez por todas, dizimando-os. Darth Vader, por sua vez, enfurecido com as ordens do imperador Palpatine, rebelou-se contra ele, o destruindo e tornando-se o novo imperador. Neste novo contexto, o imperador Vader segue insaciável por conquistar novos povos da galáxia, impondo o poder militar da sua frota intergaláctica de batalha.

No entanto, Vader anda descontente com os seus generais, pois a sua frota tem se apresentado desorganizada no campo de batalha e, assim, o imperador não está conseguindo planejar e gerenciar seus ataques. Os generais, por sua vez, informaram que isso ocorre dado o motivo de que a frota está muito grande, não havendo mais como comandá-la sem o auxílio de ferramentas computacionais.

O general Falura, braço direito do novo imperador, comentou que há um(a) grande especialista no desenvolvimento de ferramentas computacionais, e que poderia resolver a situação. Esse especialista é você, caro(a) aluno(a)! Vader gostou da ideia, e lhe enviou os detalhes do problema a ser resolvido. Vamos lá, não decepcione o novo imperador!

Detalhes do problema

O objetivo deste trabalho é praticar os conceitos relacionados à estruturas de dados elementares para resolver o problema apresentado a seguir: o imperador Vader deseja saber e controlar as naves que compõem a sua frota intergaláctica de batalha. Para tal, ele necessita de um sistema computacional que as organize em três estruturas, listadas a seguir:

- **Preparação para a batalha**: antes de um combate, as naves são posicionadas pelo imperador de acordo com a sua **aptidão** para a batalha. Só que Vader tem uma peculiaridade: ele posiciona as naves em ordem inversa. Ou seja, **a primeira nave** que ele informará para o sistema é a **menos apta** e, consequentemente, **deve ser a última a entrar, de fato, em batalha**. Da mesma maneira, **a última nave** informada **deve ser a primeira a entrar em batalha**;
- **Combate**: o imperador necessita de um controle para as naves que entrarão em combate. As naves entram em batalha com um comando de Vader. Dado o comando para entrar em combate, a nave selecionada **deve ser a mais apta que está aguardando**, baseado no posicionamento armazenado na estrutura de preparação para a batalha;
- **Nave avariada**: Como em toda batalha, as naves podem ser avariadas. Uma vez que seja reportada que a nave sofreu avaria, ela deve ser excluída da estrutura de naves em combate e ser incluída na estrutura de avaria. Uma vez avariada, a nave tentará ser consertada pela equipe de manutenção do imperador. É importante frisar que, dadas duas naves, *nave1* e *nave2*, se a *nave1* foi avariada antes da *nave2*, então a *nave1* **será consertada antes** da *nave2*. Ou seja: as naves avariadas primeiro são consertadas primeiro. Uma vez que a equipe de manutenção avisa que uma nave foi consertada, a nave passa a aguardar a ordem do imperador para entrar novamente em combate, tendo preferência sobre todas as outras que já estão aguardando.

Entrada e Saída

Entrada e saída. Neste trabalho, a entrada será a padrão do sistema (`stdin`). A saída também será a padrão do sistema (`stdout`). A primeira linha da entrada é composta por um valor inteiro **N** ($0 < \mathbf{N} \leq 5000$), que indica o total de naves da frota do imperador. A seguir, haverão **N** linhas compostas por números inteiros, que representam um identificador de cada nave. Após, a entrada é composta por várias linhas, cada qual contendo um inteiro **I**, que indica uma ação a ser realizada no sistema, cujos significados são descritos a seguir:

- **0**: indica que o imperador deseja enviar a nave mais apta das que estão aguardando para o combate. Ao ser enviada, a mensagem "nave K em combate" deve ser impressa na saída, na qual K é o identificador da nave;
- **X, tal que X é um identificador de uma nave**: indica que a nave de identificador X, que estava em combate, foi avariada. Ao ser avariada, a mensagem "nave K avariada" deve ser impressa na saída, na qual K é o identificador da nave;
- **-1**: indica que a equipe de manutenção informou que uma nave avariada foi consertada. Lembre-se que a nave consertada é sempre a com maior prioridade, aquela que primeiro chegou avariada. Ao ser consertada, a mensagem "nave K consertada" deve ser impressa na saída, na qual K é o identificador da nave. Lembre-se que a nave consertada deve ser levada novamente para a estrutura de preparação de batalha, tendo preferência sobre as que já estão lá;
- **-2**: indica que o imperador deseja obter uma impressão do identificador de todas as naves aguardando para entrar em combate, da mais apta para a menos apta. Cada nave deve ser impressa em uma linha. A impressão deve ocorrer da nave mais apta para a menos apta;
- **-3**: indica que o imperador deseja obter uma impressão do identificador de todas as naves avariadas que estão aguardando para serem consertadas. Cada nave deve ser impressa em uma linha. A impressão deve ocorrer da nave com maior prioridade para ser consertada até a de menor prioridade.

O final da entrada é indicado por `EOF` (`CTRL + D` no terminal).

Informações adicionais. Para este problema, você pode assumir que o sistema é coeso e os seguintes itens são sempre verdade:

- Todas as naves possuem identificadores **distintos**;
- **Não haverá** um identificador cujo valor seja o mesmo de uma instrução reservada do sistema, a saber: 0, -1, -2, -3;
- **Não haverá** um comando de nave avariada com identificador X, tal que X não tenha entrado em combate;
- **Não haverá** comandos de naves para entrar em combate caso não haja mais naves aguardando;
- **Não haverá** comandos de naves consertadas caso não existam naves avariadas;
- O imperador **não pedirá** uma impressão das naves aguardando para entrar em combate caso não existam naves em aguardo.
- O imperador **não pedirá** uma impressão das naves em avaria caso não existam naves avariadas.

Exemplo

Exemplo de Entrada	Exemplo de Saída
3	
5874	
3256	
8412	
0	nave 8412 em combate
-2	3256
	5874
0	nave 3256 em combate
3256	nave 3256 avariada
8412	nave 8412 avariada
-3	3256
	8412
-1	nave 3256 consertada
-2	3256
	5874
0	nave 3256 em combate

Exemplo passo-a-passo

Para ilustrar o funcionamento do sistema, vamos explanar o exemplo de entrada e saída, apresentado na seção anterior. Nesse exemplo, o imperador indica que três naves estão inicialmente aguardando para entrar em combate: 5874, 3256 e 8412. Lembre-se: ele informa da nave menos apta para a mais apta. A inserção das naves na estrutura implementada deve ter um comportamento semelhante ao mostrado na Figura 1.

Figura 1: Naves sendo inseridas na estrutura de aguardo de combate

A próxima instrução requerida pelo imperador foi a de que a nave mais apta vá para o combate (*instrução 0*). Dessa maneira, a estrutura de naves em combate deve receber a nave 8412. As estruturas envolvidas devem se comportar da maneira como indica a Figura 2. A seguinte mensagem deve ser impressa na saída padrão do sistema.

```
nave 8412 em combate
```

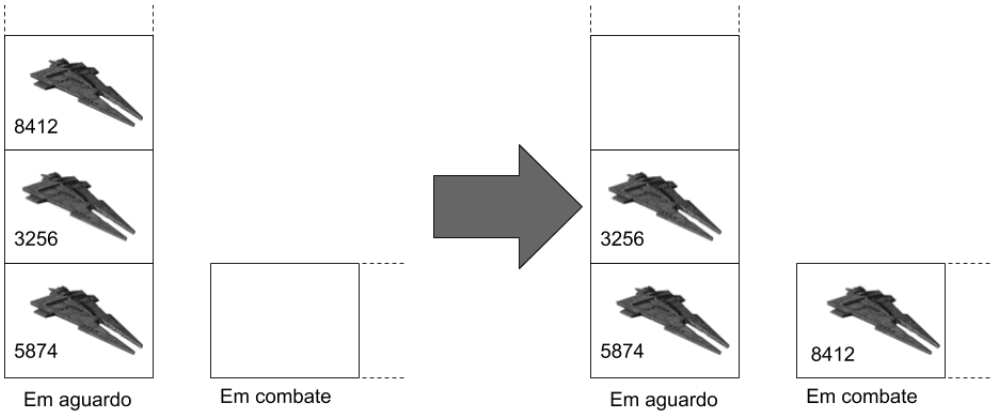

Figura 2: Nave 8412 entrando em combate

A próxima instrução solicitada é a de impressão do identificador de todas as naves aguardando para entrar em combate (*instrução -2*). Como a impressão deve ocorrer da de maior aptidão para menor aptidão, a impressão na saída padrão será:

```
3256
5874
```

Na sequência, uma nova instrução de nave a ir para combate é enviada. Assim, a nave mais apta, de identificador 3256, entra em combate. A Figura 3 ilustra o comportamento das estruturas envolvidas. Deve ser impresso na saída padrão a seguinte mensagem:

```
nave 3256 em combate
```

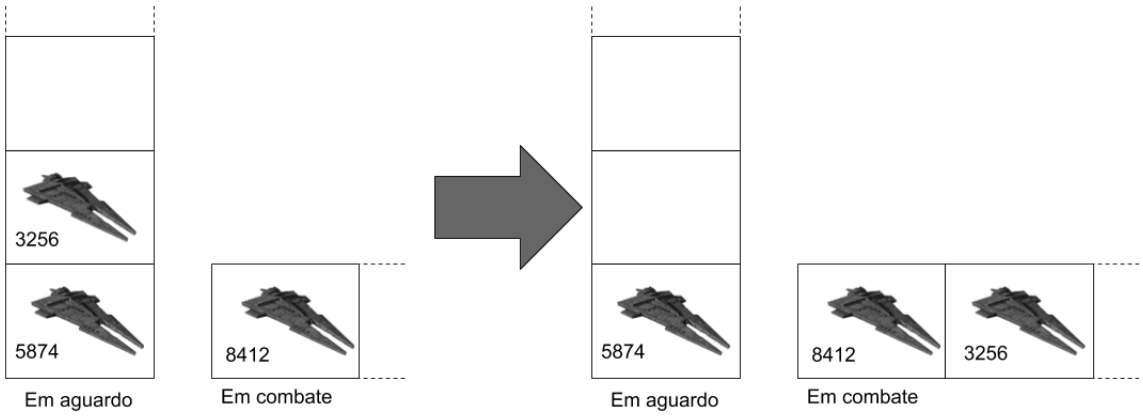

Figura 3: Nave 3256 entrando em combate

Após, é dada uma instrução com o identificador 3256, indicando que essa nave foi avariada. Assim, essa nave deve ser removida da estrutura de combate e ser inserida na estrutura de naves avariadas. A Figura 4 ilustra o comportamento esperado para essas estruturas, referente à essa instrução. A mensagem a seguir deve ser impressa na saída padrão do sistema:

```
nave 3256 avariada
```

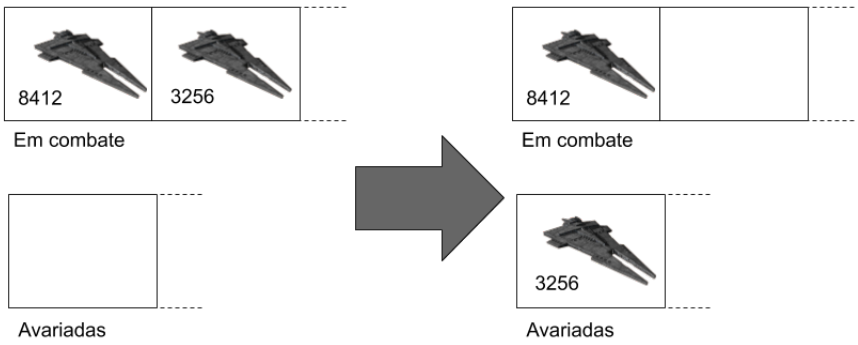

Figura 4: Nave 3256 avariada

Na sequencia, é informado que a nave 8412 foi avariada. Deve-se então realizar manipulações nas estruturas semelhante à instrução anterior, conforme mostra a Figura 5. A seguinte mensagem deve ser impressa na saída padrão do sistema:

```
nave 8412 avariada
```

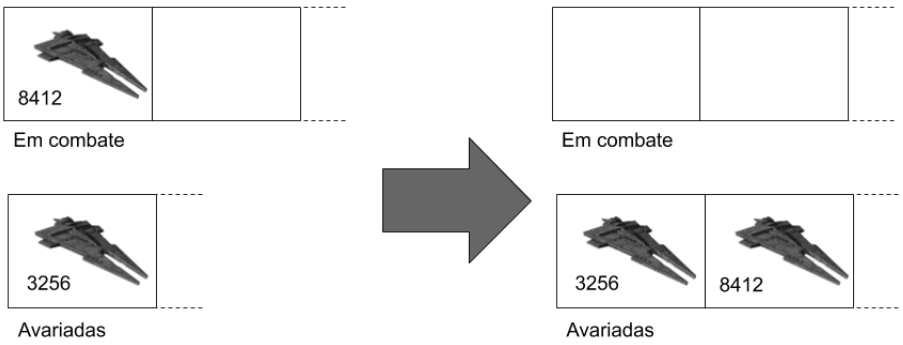

Figura 5: Nave 8412 avariada

Após, é dada a instrução de impressão do identificador de todas as naves avariadas (*instrução -3*). Lembrando que a impressão deve ocorrer da nave que será consertada primeiro até a que será consertada por último, deve ser impresso na saída o que segue:

```
3256
8412
```

A próxima instrução é relacionada ao aviso de que uma nave foi consertada (*instrução -1*). Dessa maneira, a nave 3256, que é a que deveria ser consertada primeiro, deve sair da estrutura de naves avariadas e novamente ser inserida na estrutura de naves aguardando para entrar em combate. Lembre-se que a nave consertada tem maior aptidão do que todas as outras que estão aguardando. A Figura 6 ilustra o comportamento das estruturas. Na saída padrão do sistema, deve ser impresso o que segue:

```
nave 3256 consertada
```

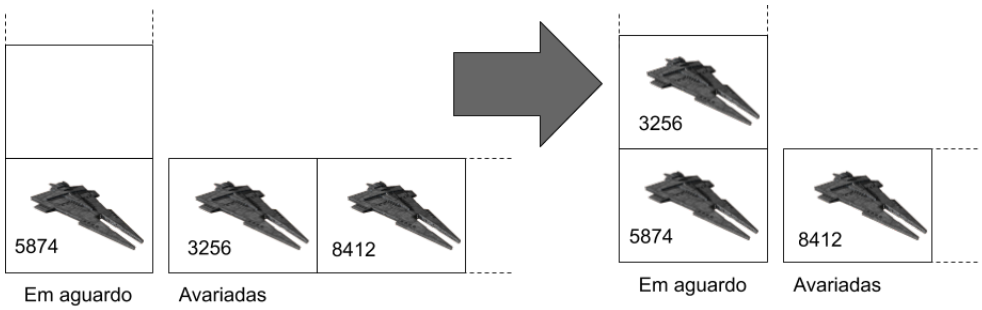

Figura 6: Nave 3256 consertada

Após, pede-se novamente que sejam impressas as naves aguardando para entrar em combate (*instrução -2*). Assim, deve ser impresso na saída padrão conforme segue:

```
3256
5874
```

Por fim, a última instrução dada na entrada é uma instrução para a nave mais apta entrar em combate (*operação 0*). Dessa maneira, a nave 3256 deve entrar em combate. A Figura 7 ilustra o comportamento das estruturas envolvidas. Na saída padrão, deve ser impresso:

```
nave 3256 em combate
```

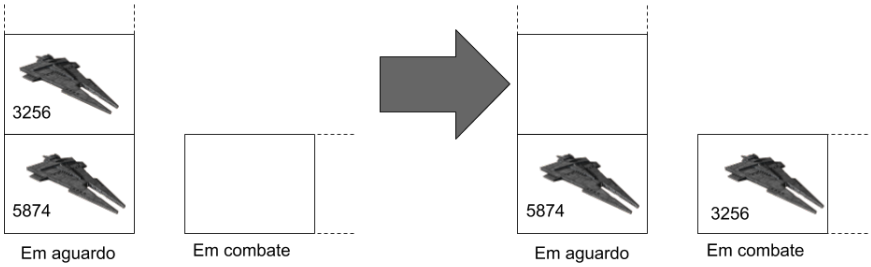

Figura 7: Nave 3256 entra novamente em combate

Entregáveis

Código-fonte. A implementação poderá ser feita utilizando as linguagens C ou C++. Não será permitido o uso da *Standard Library* do C++ ou de bibliotecas externas que implementem as estruturas de dados ou os algoritmos. **A implementação das estruturas e algoritmos utilizados neste trabalho deve ser sua**. A utilização de *Makefile*[1] é **obrigatória** para este trabalho.

Aplique boas práticas de programação e organize seu código-fonte em arquivos, classes e funções de acordo com o significado de cada parte. A separação de responsabilidades é um dos princípios da engenharia de software: cada função deve realizar apenas uma tarefa e cada classe deve conter apenas métodos condizentes com sua semântica.

Documentação. A documentação de seu programa **deverá** estar em formato **PDF**, **seguir** o modelo de trabalhos acadêmicos da SBC (que pode ser encontrado online[2]) e ser **sucinta, não ultrapassando 10 páginas**.

[1] `https://opensource.com/article/18/8/what-how-makefile`

[2] `http://www.sbc.org.br/documentos-da-sbc/summary/169-templates-para-artigos-e-capitulos-de-livros/878-modelosparapublicaodeartigos`

A documentação deverá conter **todos** os seguintes tópicos:

- Cabeçalho. Título do trabalho, nome e número de matrícula do autor.
- Introdução. Apresentação do problema abordado e visão geral sobre o funcionamento do programa.
- Implementação. Descrição sobre a implementação do programa. Devem ser detalhados o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, compilador utilizado, bem como decisões tomadas relativas aos casos e detalhes que porventura estejam omissos no enunciado.
- Instruções de compilação e execução. Instruções de como compilar e executar o programa.
- Análise de complexidade. Estudo da **complexidade de tempo e espaço** do algoritmo de melhor e pior caso desenvolvido utilizando o formalismo da notação assintótica.
- Conclusão. Resumo do trabalho realizado, conclusões gerais sobre os resultados e eventuais dificuldades ou considerações sobre seu desenvolvimento.
- Bibliografia. Fontes consultadas para realização do trabalho.

O código-fonte e a documentação devem ser organizados como demonstrado pela árvore de diretórios na Figura 8. O diretório raiz deve ser nomeado de acordo seu **nome e último sobrenome**, separado por *underscore*, por exemplo, o trabalho de "Kristoff das Neves Björgman" seria entregue em um diretório chamado `kristoff_bjorgman`. Este diretório principal deverá conter um subdiretório chamado `src`, que por sua vez conterá os códigos (`.cpp, .c, .h, .hpp`) na estrutura de diretórios desejada. A documentação **em formato PDF** deverá ser incluída no diretório raiz do trabalho. Evite o uso de caracteres especiais, acentos e espaços na nomeação de arquivos e diretórios.

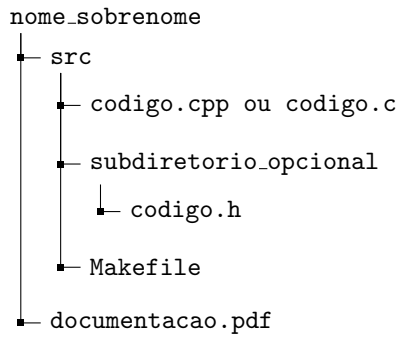

Figura 8: Estrutura de diretórios do entregável do TP1

O diretório deverá ser submetido em um único arquivo **'nome_sobrenome.zip'**, (onde nome e sobrenome seguem as mesmas diretrizes para o nome do diretório, explicado acima) através do Moodle da disciplina até as **23:59** do dia **24 de setembro de 2020**.

Considerações Finais

Algumas considerações finais importantes:

- **Preste bastante atenção nos detalhes da especificação**. Cada detalhe ignorado acarretará em perda de pontos.

- O que será avaliado no trabalho:

  Boas práticas de programação: se o código está bem organizado e identado, com comentários explicativos, possui variáveis com nomes intuitivos, modularizado, etc.

  Implementação correta dos algoritmos: se as estruturas abstratas de dados foram implementadas de forma correta e resolvem o problema aqui descrito.

  Conteúdo da documentação: se todo o conteúdo necessário está presente, reflete o que foi implementado e está escrito de forma coerente e coesa.

- Após submeter no Moodle seu arquivo **'.zip'**, faça o download dele e certifique-se que não está corrompido. Não será dada segunda chance de submissão para arquivos corrompidos.

- Em caso de dúvidas, **não hesite em perguntar** no Fórum de Discussão no Moodle ou procurar os monitores da disciplina – estamos aqui para ajudar!

- **PLÁGIO É CRIME**: caso utilize códigos disponíveis online ou em livros, **referencie** (inclua comentários no código fonte e descreva a situação na documentação). Trabalhos onde o plágio for identificado serão devidamente penalizados: o aluno terá seu trabalho anulado e as devidas providências administrativas serão tomadas. Discussões sobre o trabalho entre colegas são encorajadas, porém compartilhamento de código ou texto é plágio e as regras acima também se aplicam.

- Em caso de atraso na entrega, serão descontados $2^d - 1$ pontos, onde d é o número de dias (corridos) de atraso arredondado para cima.

- Comece o trabalho o mais cedo possível. Você nunca terá tanto tempo pra fazê-lo se começar agora!

Bom trabalho!

Apêndices

A Dicas para a documentação

O objetivo desta seção é apresentar algumas dicas para auxiliar na redação da documentação do trabalho prático.

1. **Sobre *Screenshots***: ao incluir *screenshots* (imagens da tela) em sua documentação, evite utilizar o fundo escuro. Muitas pessoas preferem imprimir documentos para lê-los e imagens com fundo preto dificultam a impressão e visualização. Recomenda-se o uso de fundo branco com caracteres pretos, para *screenshots*. Evite incluir trechos de código e/ou pseudocódigos utilizando *screenshots*. Leia abaixo algumas dicas de como incluir código e pseudocódigo em seu texto.

2. **Sobre códigos e pseudocódigos**: Ao incluir este tipo de texto em sua documentação procure usar a ferramenta adequada para que a formatação fique a melhor possível. Existem várias ferramentas para LaTeX, como o `minted`[3], o `lstlisting`[4], e o `algorithm2e`[5] (para pseudocódigos), e algumas para Google Docs (`Code Blocks`[6], Code Pretty[7]).

3. **Sobre URLs e referências**: evite utilizar URLs da internet como referências. Geralmente URLs são incluídas como notas de rodapé. Para isto basta utilizar o comando `\footnote{\url{}}` no LaTeX, ou ativar a opção nota de rodapé[8] no Google Docs/MS Word.

4. **Evite o Ctrl+C/Ctrl+V**: encoraja-se a modularização de código, porém a documentação é única e só serve para um trabalho prático. Reuso de documentação é auto-plágio[9]!

[3] `https://www.overleaf.com/learn/latex/Code_Highlighting_with_minted`
[4] `https://www.overleaf.com/learn/latex/Code_listing`
[5] `https://en.wikibooks.org/wiki/LaTeX/Algorithms`
[6] `https://gsuite.google.com/marketplace/app/code_blocks/100740430168`
[7] `https://chrome.google.com/webstore/detail/code-pretty/igjbncgfgnfpbnifnnlcmjfbnidkndnh?hl=en`
[8] `https://support.office.com/en-ie/article/insert-footnotes-and-endnotes-61f3fb1a-4717-414c-9a8f-015a5f3ff4cb`
[9] `https://blog.scielo.org/blog/2013/11/11/etica-editorial-e-o-problema-do-autoplagio/#.XORgbdtKg5k`