Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Estrutura de Dados

Trabalho Prático 1

O problema da frota intergaláctica do novo imperador

Vitor Hugo Lacerda Lana 2018072301

Professor: Luiz Chaimowicz
Professora: Raquel Prates

24 de Setembro de 2020

1 Introdução

O trabalho documentado abaixo, foi desenvolvido a partir da necessidade de entregar para nosso grande imperador, um sistema para organizar as naves de combates do Império Galático, assim tendo uma visão mais clara sobre quais estão em combate, quais sofreram avarias e quais estão prontas para entrar em combate.

O software do trabalho foi desenvolvido utilizando a linguagem C++ e utilizando o compilador g++.

2 Implementação

Para o desenvolvimento do projeto foram utilizados dois Tipos Abstratos de Dados (TAD), e uma Estrutura que possui o identificador de cada nave, essas duas estruturas foram escolhidas pelo seu padrão de comportamento se adequar as exigências do projeto.

2.1 As Estrutura

A estrutura criada foi bem simples, como o imperador não precisava de muitas informações das naves para serem salvas, apenas qual o identificador numérico, mas caso seja necessário pode ser adicionado outras informações sem problemas.

```
struct Nave
{
    int _ident;
    Nave *prox = NULL;
};
```

2.2 A Pilha

Devido necessidade do projeto, onde o imperador insere as naves na ordem das menos aptas para as mais aptas, foi escolhido como TAD para a estrutura de preparação de combate uma Pilha, essa estrutura segue esse mesmo padrão de comportamento, sendo o primeiro item a Entrar o ultimo item a sair. Baseando nesse comportamento, foram implementadas quatro métodos, para que utilizando deles, possamos entregar para o imperador as informações necessários.O custo computacional dessa pilha no pior dos casos será O(2n) quando o método de imprimir reverso é chamado e para as operações de inserir e remover a mesma terá custo O(1).

Pilha
*topo: Nave
addPilha(int) : void remPilha() : int impPilha() : void impPilhaInv() : void ∼Pilha()

2.2.1 addPilha(int): void

A função de adicionar elementos na pilha, a mesma recebe como parâmetro um inteiro que é o identificador da nave, a função adiciona esse identificador na pilha e seguindo o comportamento padrão da estrutura.

2.2.2 remPilha(): int

A função realiza a remoção do último elemento adicionado na pilha, quando é chamada e retorna um inteiro contendo o valor do identificador.

2.2.3 impPilha(): void

A função quando é chamada imprime os elementos que estão armazenados na pilha de cima para baixo, então o último adicionado vai ser o primeiro a ser impresso e o primeiro elemento da pilha vai ser o último a ser impresso.

2.2.4 impPilhaInv(): void

A função quando é chamada, imprime a pilha na ordem inversa da função normal, para isso ela utiliza de uma Fila (falaremos mais adiante dessa função), adicionando os elementos da pilha em uma fila, realizando a impressão dessa fila e logo após a impressão chama o método destrutor dessa fila.

2.3 A Fila

Seguindo as necessidade do projeto, foi necessário também realizar o desenvolvimento de uma classe de Fila para a estrutura combate e de manutenção, essa estrutura segue o padrão de comportamento FIFO, sendo o primeiro item a Entrar o primeiro item a Sair. Baseando nesse comportamento, foram implementadas cinco métodos, assim utilizando deles para entregar para o imperador os dados necessários. O custo computacional dessa fila no pior dos casos será O(2n) quando o método de imprimir reverso é chamado e para as operações de inserir e remover a mesma terá custo O(1).

Fila
*topo: Nave
addFila(int) : void remFila() : int impFila() : void impFilaInv() : void busRemFila(int) : void ∼Fila()

2.3.1 addFila(int): void

A função de adicionar elementos na fila, a mesma recebe como parâmetro um inteiro que é o identificador da nave, a função adiciona esse identificador na fila e seguindo o comportamento padrão da estrutura.

2.3.2 remFila(): int

A função realiza a remoção do primeiro elemento adicionado na fila, quando é chamada e retorna um inteiro contendo o valor do identificador.

2.3.3 impFila(): void

A função quando é chamada imprime os elementos que estão armazenados na fila do primeiro para o último, então o primeiro adicionado vai ser o primeiro a ser impresso e o último elemento da fila vai ser o último a ser impresso.

2.3.4 impFilaInv(): void

A função quando é chamada, imprime a fila na ordem inversa da função normal, para isso ela utiliza de uma Pilha , adicionando os elementos da fila em uma Pilha, realizando a impressão dessa pilha e logo após a impressão chama o método destrutor dessa pilha.

2.3.5 busRemFila(): void

A função quando é chamada recebe um valor inteiro, que é o identificador da nave que deseja remover. Realiza a busca desse identificador na fila, faz a remoção dele dessa fila e o adiciona, na fila de manutenção.

2.4 Lógica Principal

A lógica do sistema foi feita no arquivo main.cc, foi utilizado uma estrutura de repetição para receber todos os identificadores das naves e armazenar na pilha de combate, após todas as naves serem inseridas, entramos em outro laço de repetição, nesse laço, recebemos os comandos para adicionar a nave em combate, adicionar na fila de combate, remover da fila de combate ou remover da fila de manutenção.

A implementação dessas funções foram feitas utilizando a lógica básica em junção as estruturas implementas anteriormente.

3 Instruções Compilação

Junto aos arquivos do programa segue um arquivo Makefile configurado para compilar os códigos do programa, e realizar os testes do mesmo.

Para realizar a compilação e os testes basta utilizando um sistema Linux (Utilizei para testes Ubuntu 18.04, rodando WSL), entrar no terminal dentro da pasta **src**, e digital o comando **make**. O programa irá compilar e executar todos os testes.

Abaixo o retorno do sistema após os comandos de **make** e **make test**

```
codespace:~/workspace/ed/2020/tp1/vitor_lana/src$ make
g++  -Wall -Wextra main.cc Pilha.o Fila.o -o ./tp1
codespace:~/workspace/ed/2020/tp1/vitor_lana/src$ make test
Test 0 passed
Test 1 passed
Test 2 passed
Test 3 passed
Test 4 passed
Test 5 passed
```

4 Conclusão

O trabalho foi de grande importância para desenvolvimento e utilização dos conceitos estudados de TADs funções recursivas e outros assuntos vistos durante a matéria de estrutura de dados.

1 2 3 4 5

[1] https://pt.wikipedia.org/wiki/Tipo_abstrato_de_dado
[2] https://www.ime.usp.br/~pf/algoritmos/aulas/lista.html
[3] https://pt.wikipedia.org/wiki/FIFO
[4] https://en.wikipedia.org/wiki/Stack_(abstract_data_type)
[5] https://en.wikipedia.org/wiki/Galactic_Empire_(Star_Wars)