Trabalho de Estrutura de Dados

IFMG - Formiga

Denise Ferreira Garcia Rezende

Aluno: Alef Faria Silva

Curso: Bacharelado em Ciência da Computação

Período: 3°

Abstração do porto

O trabalho teve como objetivo desenvolver um programa que controla ações de um porto. Após a leitura do documento do trabalho, o aluno chegou na seguinte abstração:

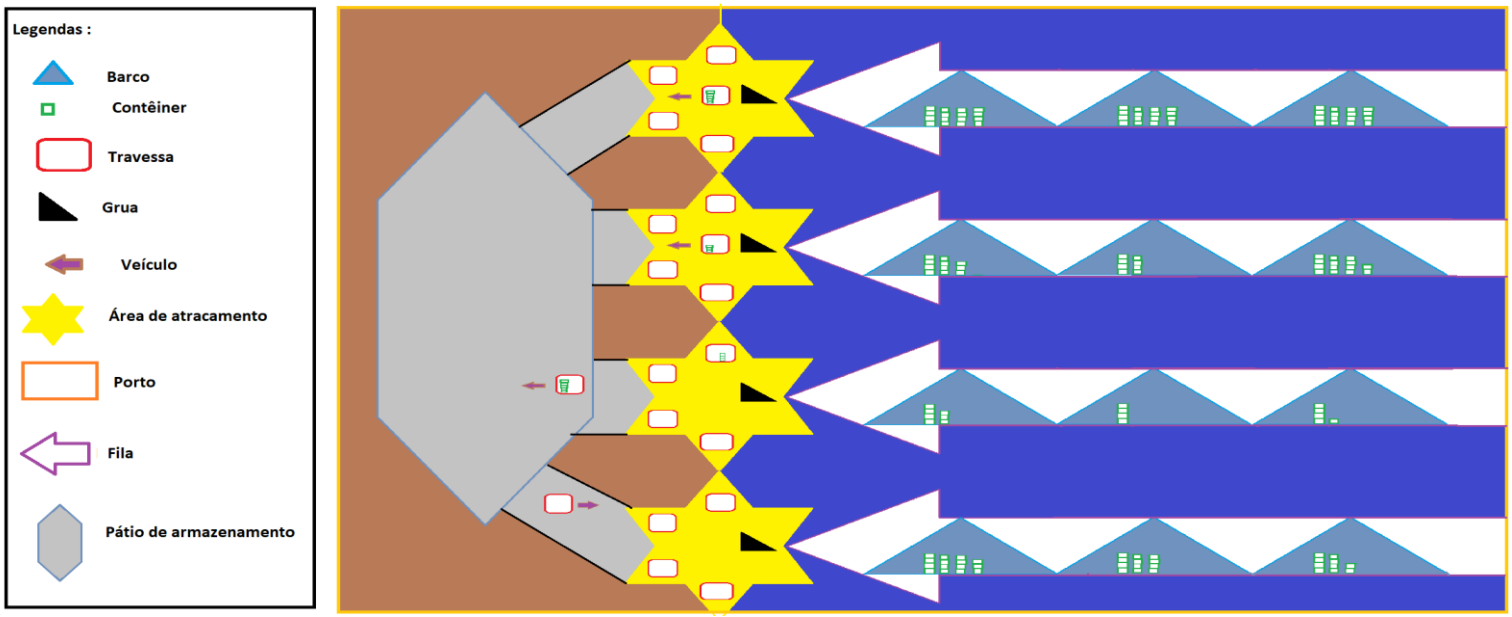

Figura 1- Abstração do Porto

Estruturas de dados utilizadas

Para cumprir os requisitos especificados no trabalho foram usados os seguintes tipos abstratos de dados:

- struct conteiner (Representa um contêiner):
  - Atributos:
    - int id; (Representa o Código da struct)

    - Estrutura de dado:
        - Pilha (Armazena contêineres seguindo o padrão de pilha)
- struct navio (representa um navio):
    - Atributos:
        - int id; (Representa o Código da struct)
        - Pilha *pi[4]; (Vetor de ponteiro que aponta para pilha de struct de conteiner)
        - int tempoDeEspera; (Representa o tempo que o navio passou na fila do atracadouro)
        - int qtdConteiner; (Representa a quantidade de contêineres que o navio possui)
    - Estrutura de dado:
        - Fila (Armazena navios seguindo o padrão de fila)
- struct atracadouro (Representa um atracadouro):
    - Atributos:
        - int id; (Representa o Código da struct)
        - int carro; (Variável de controle para calcular se o veículo está desempilhando uma travessa)
        - int qtdViagem; (Variável que conta quantas vezes o carro foi usado)
        - int pilhaDesempilhando; (Variável que armazena o index da pilha que está desempilhando)
        - Pilha *pi[5]; (Vetor de ponteiro que aponta para a pilha de struct de conteiner)
        - Fila *fi; (Ponteiro de fila de navios)
- Fila:

```
typedef struct fila Fila;

struct fila {
        struct elemento2 *inicio;
        struct elemento2 * final;
};

 struct elemento2 {
        struct navio navios;
        struct elemento2 *prox;
};
```

```
typedef struct elemento2 Elem2;
```

- Pilha:

```
struct elemento {
        struct conteiner dados;
        struct elemento *prox;
} elemento;

typedef struct elemento Elem;

typedef struct Pilha {
        struct elemento *topo;
} Pilha;
```

Tanto a pilha como a fila utilizadas são dinâmicas. A escolha para esse modo de implementação se deu em função da geração aleatória de navios e contêineres. A pilha dinâmica evita o trabalho de atribuir os demais index não usados das pilhas estáticas em 0. O que seria necessário para controlar o empilhamento dos contêineres nas travessas. A fila dinâmica evita possíveis erros que pode ocorrer caso o tamanho da fila fosse limitado e uma determinada geração de navios em um tempo excedesse o tamanho disponível na fila.

Detalhes do desenvolvimento

O programa foi desenvolvido no ambiente de desenvolvimento integrado NetBeans, a linguagem usada foi o C, versão C11.

Funções usadas:

- Main: Função onde é criado 4 structs atracadouro e inicializado seus atributos para simular o funcionamento do porto. Possui também o menu do programa onde o usuário decide a quantidade de tempo que será passada, entre 1 e 100, ou digita 0 para sair do programa. Caso a entrada do usuário seja entre 1 e 100 é chamada a função passaTempo.
- passaTempo: Função principal do programa a qual chamara as outras funções que gastam uma unidade de tempo, incrementara a variável tempo passado;
- geraNavioConteiner: Função que gera de 0 a 3 navios e 4 a 16 contêineres por navio, empilha eles, nas respectivas pilhas de cada um. Retorna o número de navios gerados
- enfileiraNavio: Recebe os navios da função geraNavioConteiner e os enfileira em uma das filas do porto

- desempilhaConteiner: Função que desempilha um contêiner dos primeiros navios de cada uma das 4 filas do porto.
- desenfileiraNavios: Função que verifica se o primeiro navio de cada fila está vazio, caso sim, remove ele da fila.
- mostraResultado: Função que gera um relatório do porto.
- delay: Função que gera um pequeno delay dentro do laço de repetição da função passaTempo, para que a geração aleatória funcione melhor.

Resolução de problemas

- **Gerar Navios e contêineres**: Foi usado a função rand() do C para a geração de números aleatórios.
- **Enfileirar navios**: Os novos navios gerados são colocados um por um na fila de menor tamanho, conservando assim a proporção entre as 4 filas.
- **Evitar engarrafamento**: Antes de gerar os novos navios ao se passar o tempo, verifica-se o tamanho das filas, caso todas estejam com o tamanho menor ou igual a 5, os navios são gerados, se não, nesse tempo nenhum navio é gerado.
- **Desempilhar travessas**: Para que o desempilhamento da travessa ocorra em duas unidades de tempo foi criada a variável carro na struct atracadouro, com valor igual a 0, quando o desempilhamento ocorre a variável carro passa a valer –2, sempre que um novo tempo é acionado, verifica-se se o carro vale 0, caso não, soma-se 1 em seu valor. O carro só pode desempilhar uma pilha caso seu valor seja 0, assim e garantido que ele fique 2 unidades de tempo ocupado com o desempilhamento. O atributo pilhaDesempilhando guarda a pilha que está desempilhando, faz-se uma verificação para não empilhar nada na pilha de mesmo index dessa variável, assim garantindo que essa fique sem empilhar por 2 unidades de tempo.
- **Incrementar o tempo dos navios**: Os navios que permanecem nas filas têm seus atributos tempoDeEspera incrementado em 1 a cada passagem de tempo, isso foi feito através de uma função recursiva que recebe o fila.inicio e uma variável int cod, que recebe o id do último elemento da fila. A função incrementa o tempo e, enquanto o id do navio for diferente de cod chama-se novamente a mesma função, passando como parâmetro o elemento.proximo.