Descrição Trabalho Porto

Nome: Higor Gabriel Lino Silva
Matrícula: 0070308

As seguintes TADs foram utilizadas para ser feito este trabalho:

```c
typedef struct TipoCelula *TipoApontador;

typedef struct TipoNavio {
    int id;
    int quantidadeTempoDesc;
    TipoPilha PilhasContainers[4];
    int quantidadeContainers;
} TipoNavio;

typedef struct TipoCelula {
    TipoNavio Navio;
    TipoApontador Prox;
} TipoCelula;

typedef struct TipoFila {
    TipoApontador Frente, Tras;
} TipoFila;

void FFVazia(TipoFila *Fila);
int FVazia(TipoFila Fila);
void Enfileira(TipoNavio *navio, TipoFila *fila);
void Desenfileira(TipoFila *Fila, TipoNavio *Navio);
int TamanhoFila(TipoFila Fila);
```

```c
typedef int TipoApontadorP;

typedef struct TipoContainer{
    int numeroContainer;
}TipoContainer;

typedef struct TipoPilha{
    TipoContainer Container[10];
    TipoApontadorP Topo;
}TipoPilha;

typedef struct TipoPilhaT{
    TipoContainer Container[10];
    TipoApontadorP Topo;
}TipoPilhaT;

typedef struct TipoTravessa{
    TipoPilhaT Containers;
    int quantidadeContainers;
    int situation; //0 = disponivel 1 = desempilhando 2 = cheia
}TipoTravessa;

void FazPilhaVazia(TipoPilha *pilha);
void FazPilhaVaziaT(TipoPilhaT *pilha);
int PVazia(TipoPilha pilha);
int PVaziaT(TipoPilhaT pilha);
void Empilha(TipoContainer x, TipoPilha *Pilha);
void EmpilhaT(TipoContainer x, TipoPilhaT *Pilha);
TipoContainer Desempilha(TipoPilha *Pilha, TipoContainer *Container);
TipoContainer DesempilhaT(TipoPilhaT *Pilha, TipoContainer *Container);
int Tamanho(TipoPilha Pilha);
```

Note que no caso da pilha foram feitas funções diferentes para a pilha das travessas (FazPilhaVaziaT, PVaziaT, EmpilhaT e DesempilhaT) e para as pilhas de contêineres dos navios.

Funcionamento do porto

Cada iteração do laço while principal na main equivale à 1 unidade de tempo

A cada unidade de tempo são gerados de 0 a 3 navios com 4 pilhas de contêineres em cada navio, cada navio só pode ter uma quantidade máxima de 16 containers, logo cada pilha de contêineres dos navios fica com no máximo 4 contêineres.

```
TipoNavio gerarNavio(int navioID) {
    TipoNavio navio;
    navio.id = navioID;
    navio.quantidadeTempoDesc = 0;
    navio.quantidadeContainers = 0;

    int quantidadeContainers = gerarNumeroAleatorio( min: 4,  max: 16); // Gera quantidade de contêineres aleatória

    // Divide a quantidade de contêineres igualmente entre as 4 pilhas
    int quantidadeContainersPorPilha = quantidadeContainers / 4;

    for (int i = 0; i < 4; i++) {
        TipoPilha pilha;
        FazPilhaVazia( pilha: &pilha);

        for (int j = 0; j < quantidadeContainersPorPilha; j++) {
            TipoContainer container;
            container.numeroContainer = j + 1;
            Empilha( x: container,  Pilha: &pilha);
            navio.quantidadeContainers++;
        }
        navio.PilhasContainers[i] = pilha;
    }

    // Imprime informações do navio gerado
    printf( format: "Navio gerado: ID %d, Quantidade de Containers: %d\n", navio.id, navio.quantidadeContainers);

    return navio;
}
```

Existem 4 áreas de atracamento onde cada uma delas existe uma fila de navios, os navios que são gerados são distribuídos uniformemente nessas 4 filas levando em consideração a quantidade de contêineres para que o tempo de espera seja o menor possível.

```c
void enfileirarNavioUniforme(TipoNavio* navio, TipoFila* fila1, TipoFila* fila2, TipoFila* fila3, TipoFila* fila4) {
    int quantidadeTotalFila1 = somarQuantidadeContainersFila( fila: *fila1);
    int quantidadeTotalFila2 = somarQuantidadeContainersFila( fila: *fila2);
    int quantidadeTotalFila3 = somarQuantidadeContainersFila( fila: *fila3);
    int quantidadeTotalFila4 = somarQuantidadeContainersFila( fila: *fila4);

    int menorQuantidadeTotal = quantidadeTotalFila1;
    if (quantidadeTotalFila2 < menorQuantidadeTotal)
        menorQuantidadeTotal = quantidadeTotalFila2;
    if (quantidadeTotalFila3 < menorQuantidadeTotal)
        menorQuantidadeTotal = quantidadeTotalFila3;
    if (quantidadeTotalFila4 < menorQuantidadeTotal)
        menorQuantidadeTotal = quantidadeTotalFila4;

    if (quantidadeTotalFila1 == menorQuantidadeTotal) {
        Enfileira( navio: &(*navio),  fila: fila1); // Passa o endereço do ponteiro navio
        printf( format: "Navio enfileirado na Fila 1\n");
    } else if (quantidadeTotalFila2 == menorQuantidadeTotal) {
        Enfileira( navio: &(*navio),  fila: fila2); // Passa o endereço do ponteiro navio
        printf( format: "Navio enfileirado na Fila 2\n");
    } else if (quantidadeTotalFila3 == menorQuantidadeTotal) {
        Enfileira( navio: &(*navio),  fila: fila3); // Passa o endereço do ponteiro navio
        printf( format: "Navio enfileirado na Fila 3\n");
    } else if (quantidadeTotalFila4 == menorQuantidadeTotal) {
        Enfileira( navio: &(*navio),  fila: fila4); // Passa o endereço do ponteiro navio
        printf( format: "Navio enfileirado na Fila 4\n");
    }
}
```

A parte de desempilhamento dos contêineres funciona da seguinte forma: a grua irá desempilhar um dos contêineres e irá empilhá-lo na travessa, tal travessa que quando chegar em 5 contêineres irá ser levada para o pátio para ser esvaziada e só voltará a estar disponível novamente depois de 2 unidades de tempo, enquanto está sendo esvaziada a grua irá utilizar a quinta travessa que também tem uma capacidade de 5 contêineres.

Informações importantes:

No arquivo main.c foi definido uma constante `"lIMITE_TEMPO"` que serve para não deixar o programa executando para sempre, você pode definir quantas unidades de tempo quer fazer a simulação apenas alterando esta constante, por padrão irá fazer 1000 iterações e irá parar a simulação.