Documentação para o Primeiro Trabalho Pratico de AEDS 2

Nome: Igor Henrique da Silva Dias

Matricula: 2014144192

1. **Introdução**

A manipulação de filas e pilhas é um dos aspectos importantes do curso de Algoritmos e Estrutura de Dados 2, juntamente com a criação de tipos abstratos de dados (TAD) que contem funções para a implementação de algum programa.

O Objetivo desse trabalho é exercitar o uso de TAD's para manipular filas e pilhas. Deverão ser implementadas funções para a manipulação de filas e pilhas. Esse TAD será testado em um programa que simula uma lanchonete usando filas de clientes e pilhas de bandejas para calcular o tempo médio de permanência de um cliente nesse ambiente.

Espera-se que com isso treinar e praticar conceitos de programação sobre TAD's e funções para manipulações de estrutura de dados.

2. **Implementação**

Para a implementação do trabalho foram criados TAD's contendo funções para manipulação de filas e pilhas, ambas implementada usando alocação dinâmica, e funções para o uso da implementação da lanchonete.

Foram criados três TAD's. Um para manipular filas, outro para manipular pilhas e outro com funções para serem usadas como o funcionamento da lanchonete.

No TAD fila foram criadas as seguintes funções:

void FFVazia(TFila *): cria uma fila vazia.

int FVazia(TFila): verifica se a fila passada no parâmetro está vazia.

void Enfileira(TItemCliente, TFila *): adiciona um item na fila. No caso seria um cliente.

void Desenfileira(TFila *, TItemCliente *): retira um item da fila. Também é um cliente

void liberaFila(TFila *): desaloca a fila que foi alocada dinamicamente.

No TAD pilha fora criadas as seguintes funções:

void FPVazia(TPilha *): cria uma pilha vazia.

int PVazia(TPilha): verifica se a pilha passada no paramento está vazia.

void Empilha(TItemBandeja, TPilha *): insere um item na pilha.

void Desempilha(TPilha *, TItemBandeja *): retira um item da pilha.

int Tamanho(TPilha): retorna o tamanho da pilha.

void liberaPilha(TPilha *): desaloca a pilha que foi alocada dinamicamente.

No TAD lanchonete foram criadas as seguintes funções:

void preencheCliente(TItemCliente *, int , int): preenche um tipo cliente com os dados do mesmo (cliente foi identificado com um numero) e preenche o dado da hora de chegada do cliente na lanchonete.

void saidaCliente(TItemCliente *, int): preenche o cliente com o dado da hora de saída do cliente da lanchonete.

int tempoAtendimentoCliente(TItemCliente *): calcula o tempo que cada cliente passou na lanchonete.

void adicionaDoisClientesNaFila(TFila *, int *, int *): essa função adiciona 2 clientes na fila do caixa a cada período de tempo, como especificado na descrição do trabalho.

void entraNaFila(TItemCliente, TFila *): adiciona um cliente na fila.

void saiDaFila(TFila *, TItemCliente *): retira um cliente da fila.

int pilhaBandejasVazia(TPilha): verifica se a pilha de bandejas está vazia.

int pilhaBandejasCheia(TPilha): verifica se a pilha de bandejas está cheia.

void preencheBandeja(TItemBandeja *, int): preenche cada bandeja adicionada na pilha com um numero para sua identificação.

void retiraBandeja(TPilha *, TItemBandeja *): retira uma bandeja da pilha.

void adicionaBandeja(TPilha *, TItemBandeja *, int *): adiciona uma bandeja na pilha.

void vaiParaPosicao(TFila *, TItemCliente *): essa função retira o cliente de alguma fila e o coloca em uma posição de ação. (exemplo: cliente1 vai para o caixa e o cliente2 vai pegar a bandeja)

void saiDaPosicao(TItemCliente, TFila *): essa função retira o cliente de alguma posição de ação e o coloca em alguma fila.

void preenchePosicaoVazio(TItemCliente *): se a posição de ação não vá receber nenhum cliente, essa função preenche a posição com vazio para identificar se há algum cliente ou não.

int posicaoVazia(TItemCliente): verifica se a posição de ação se encontra vazia.

2.1 Programa principal

O programa principal cria varias variáveis de tipos inteiros, tipos TFila e TPilha, e variáveis do tipo TItemBandeja e TItemCliente (esses são tipos de itens que são usados nos TAD's). Após isso ele inicia as filas, pilhas e preenche os itens. Em seguida ele inicia um loop que contem vários comandos de execução para simular o movimento de clientes na lanchonete. Ele itera esse loop até não ter mais clientes na lanchonete. Nesse loop são usadas as funções criadas simulando cada passo que deve ser feito na lanchonete, como mover um cliente da fila para o caixa, do caixa para fila de bandejas, entre outras funções. Nesse loop o programa só utiliza as funções criadas, não acessa diretamente a estrutura dos dados.

2.2 Organização do código

O código foi dividido em sete arquivos. são eles: main.c, fila.c, fila.h, pilha.c, pilha.h, lanchonete.c e lanchonete.h.

O programa recebe o valor de tempo que haverá clientes entrando na fila para retirar sua ficha. Esse tempo foi definido em 240 medidas de tempo, que são equivalentes a 4 horas de funcionamento. Como não ficou claro na descrição do trabalho, o programa continua rondando até não haver mais clientes na lanchonete, e após isso é calculado o tempo médio gasto dos clientes para serem atendidos.

Para compilar o código foi usado a seguinte linha de comando:

Gcc main.c fila.c pilha.c lanchonete.c -o programa

Para executar o programa foi usado a seguinte linha de comando:

./programa

Ao final do programa ele imprime na tela o tempo medio que os clientes ficaram na lanchonete, a quantidade de clientes atendidos e o tempo total que o programa rodou do momento que o primeiro cliente chegou a lanchonete até o momento que o ultimo cliente saiu da lanchonete.

3. Analise de complexidade

As principais funções dos TAD's Fila e Pilha são executadas em tempo O(1), pois nelas não há laço, há só comandos definidos, só nas funções de liberar a memoria alocada que o tempo varia dependendo do tamanho da fila ou da pilha, mas no programa ela é só usada para liberar a fila quando já não há mais elementos, pois todos os clientes já não se encontram mais na lanchonete. A função de liberar a memoria das pilhas também varia de acordo com o tamanho da pilha, mas no programa essa entrada tem um valor finito menor que 30, então o custo também é baixo e pode ser considerado como O(1).

O programa principal é o que tem o maior custo. Ele usa um loop que varia dependendo da quantidade de tempo que a lanchonete será usada. Ele tem a complexidade O(n) multiplicado por uma variável que dependerá dos padrões da lanchonete, como quantidade de bandejas adicionadas a pilha, qual a distancia do tempo que elas são adicionadas a pilha.

No final o programa tem complexidade O(n), pois é a função que domina assintoticamente as outras funções.

No TAD lanchonete as funções também são básicas, todas usam comandos sequenciais, sem haver loops ou usam funções dos outros dois TAD's. Por isso o custo é fixo em O(1) para todas as funções.

4. Testes

O código foi compilado usando o GCC em um computador rodando **Ubuntu 16.04 LTS 64-bit**, com um processador Intel I5 de 1.7GHz e 4GB de memoria.

Foram realizados alguns testes com valores de tempos diferentes e quantidade de bandejas diferentes que era adicionado a pilha para verificar se o código rodaria como esperado.

5. **Conclusão**

A implementação do programa correu sem problemas, mas teve um tempo demorado de implementação pela dificuldade de manipular ponteiros. Essa foi a principal dificuldade encontrada no trabalho.

Referencias

Foram usados os slides das aulas apresentadas pelo professor e os exercícios das listas que foram passadas.

Anexos

Listagem dos programas

Main.c

Fila.c

Fila.h

Pilha.c

Pilha.h

Lanchonete.c

Lanchonete.h