Universidade Federal de Minas Gerais

Sistemas de Informação

Algoritmos e Estruturas de Dados II

TP 1 – Cantina do ICEX

Nome: Ewerton Silva Santos
Matricula: 2016058140

Documentação

Introdução - Simulando a Cantina do ICEX

O programa simula a cantina do ICEX num período de 4 horas (240 minutos), não possui entrada, apenas uma saída padrão que se refere ao tempo médio de atendimento e ao número de usuários atendidos. O programa usa duas Tad's, uma que é usada para inserir os usuários em uma fila, outra usada para inserir bandejas em uma pilha. O conteúdo principal é executado dentro de um loop de 240 ciclos, a cada interação do loop 2 usuários são inseridos na fila do caixa. O primeiro usuário da fila do caixa é removido a cada ciclo e enviado para última posição da fila de bandejas. Já para a fila de bandejas o primeiro usuário é removido e enviado para uma variável chamada **flag_comida,** que recebe as informações do usuarios removido. O valor da variável **flag_comida** é atribuído a variável **comida_pri**, que se refere ao primeiro tipo de comida. Nesta faze final do loop, a cada interação o usuário que se serve do quarto tipo de comida é removido da cantina, o usuário que se serve do terceiro tipo de comida vai se servir do quarto tipo, ou seja todos usuarios andam uma posição, removendo o ultimo e deixando a primeira posição vazia, para que um novo usuário possa se servir.

A linguagem utilizada para o desenvolvimento foi a linguagem C, o compilador utilizado foi o **GCC Mingw(4.9.3).**

Pasta do TP

A pasta **cantina_simulador** contém:

- **main.exe** que executa a simulação do problema;
- **main.c** que corresponde ao arquivo principal do programa;
- **band.h** contém os protótipos de funções e definição de tipos usados para manipular as bandejas;
- **band.c** contem a implementação das funções que manipulam as bandejas;
- **filas.h** contém os protótipos de funções e definição de tipos usados para manipular as filas;

- **filas.c** contem a implementação das funções que manipulam as bandejas;

Desenvolvimento

A solução utilizada para resolver o problema foi simples, usando apenas as TAD's do tipo pilha e do tipo fila é possível resolver. É utilizado duas estruturas do tipo fila, uma para o caixa(**fila_caixa**) e outra para pegar bandejas(**fila_band**), estas filas contam apenas com funções básicas: criação de uma fila vazia (**FFVazia**), verificação de fila vazia(**FVazia**), inserção de dados na fila(**Enfileira**) e remoção de dados da fila(**Desenfileira**).

Para a única estrutura de pilha, que é usada para a pilha de bandejas (**p_bandeja**), usamos só as funções básicas: criação de uma pilha vazia (**FPVazia**), verificação de pilha vazia (**Vazia**), inserção de dados na pilha (**Empilha**), remoção de dados da fila (**Desempilha**), verifica o tamanho da pilha (**Tamanho**),

Dentro do loop principal a cada 12 interações do ciclo até 10 bandejas são inseridas na estrutura pilha bandeja, sempre respeitando o máximo de 30 bandejas. Definido o número de bandejas é executado um loop **for** de até 10 interações para inserir bandejas na pilha.

Quando o laço principal é iniciado, o tempo atual (**tempo_atual)** é atribuído para um espaço da fila, que corresponde a um usuários. O único dado gerada é o tempo de entrada do usuário e todas as operações realizadas são em cima deste valor, que e enviado de estrutura para estrutura. Quando o usuário chega ao quarto tipo de comida, ou seja, seu atendimento chegou ao fim, é feita a diferença do tempo de saída com o tempo de entrada que é somada a variável **soma_tempo,** também é acrescido em 1 a variável **num_users**(número de usuarios atendidos).

Ao final do código é feita uma varredura sobre os tipos de comida e as duas filhas, verificando o tempo em que usuarios não atendidos ficaram na fila, esse tempo é somado a variável **soma_tempo** e **num_users** é acrescido em mais 1. Ao final ocorre três outputs: output da variável **soma_tempo** (é a soma do tempo de todos usuarios que ficaram na fila); output da variável **media_tempo** (é a divisão da variável **soma_tempo** por **num_users**, essa divisão nos retorna o tempo médio de atendimento); output da variável **num_users** que se refere ao número de usuarios atendidos;

Filas.c

O estrutura filas.c é composta por 7 funções:

- ***void Desenfileira(TFila *Fila, USUARIO *Itemm);*** que remove os usuarios de uma fila e guarda seus dados em uma variável de flag, tem função de complexidade f(6), ordem O(1);
- ***void Enfileira(USUARIO x, TFila *Fila);***//criar um novo usuário em uma fila, tem função de complexidade f(4), ordem O(1);
- ***int FVazia(TFila Fila);***//retorna se a fila esta vazia ou não, tem função de complexidade f(1), ordem O(1);
- ***void FFVazia(TFila *Fila);*** //cria um nova fila, tem função de complexidade f(3), ordem O(1);
- ***USUARIO insere_val_users(USUARIO comida, int x);***//função usada para inserir dados em um usuário, tem função de complexidade f(4), ordem O(1);
- ***int retorna_entrada(USUARIO comida);*** //retorna o tempo de entrada de um tipo **USUARIO**, tem função de complexidade f(1), ordem O(1);
- ***int destroi_fila (TFila *Fila); //*** que desaloca a memoria usada para as filas, complexidade f(n), ordem O(n), depende do número de usuarios inseridos;

Toda sua estrutura é a mesma de uma simples fila dinâmica, por este motivo não existe a necessidade de se aprofundar na sua implementação. O Tipo **USUARIO**, que é o tipo aonde os dados dos usuarios são armazenados possui apenas um tipo de campo, que é o campo do tempo de entrada.

Band.c

O tipo band.c é composta por 6 funções:

- ***void FPVazia(TPilha *Pilha);*** //criar uma nova pilha, tem função de complexidade f(4), ordem O(1);
- ***int Vazia(TPilha Pilha);***//verifica se a pilha esta vazia, tem função de complexidade f(1), ordem O(1);
- ***void Empilha(TItem x, TPilha *Pilha);*** //insere uma bandeja na pilha, tem função de complexidade f(5), ordem O(1);
- ***void Desempilha(TPilha *Pilha);*** // remove um elemento da pilha, tem função de complexidade f(5), ordem O(1);
- ***int Tamanho(TPilha Pilha);*** //verifica o tamanho da pilha de bandeja, função de complexidade f(1), ordem O(1); ;
- ***TItem emiplha_bandeja(TItem n_band, int x);***// permite reinserir uma nova quantidade de bandejas na pilha, tem função de complexidade f(1), ordem O(1);
- ***int destroi_pilha(TPilha *Pilha);*** // desaloca a memória usada com a pilha de bandejas, tem , tem função de complexidade f(n), ordem O(n), pois depende do número de bandejas que foi inserido;
-

Toda sua estrutura é a mesma de uma simples pilha dinâmica, por este motivo não existe a necessidade de se aprofundar na sua implementação. O Tipo **TItem**, que é o tipo usado para as bandejas possui apenas uma variável do tipo **CHAVE,** apenas para guardar um valor de identificação da bandeja, que e totalmente desprezível para a execução e resultado final do código.

Resultados: Casos de teste

No caso de teste padrão (1 fila de bandeja,1 caixa, 1 fila de bandejas, com máximo de 30, 1 pilha de bandejas e 1 estação para servir comida), tem-se um tempo de atendimento médio de é 62.68 minutos, número de clientes atendidos é 220, somo total dos tempos de atendimento dos usuários é 13790.

No caso de teste 2(2 filas de bandeja, 1 fila de bandejas, com máximo de 30, 1 pilha de bandejas e 1 estação para servir comida), tem-se um tempo de atendimento médio de é 55.14 minutos, número de clientes atendidos é 110, somo total dos tempos de atendimento dos usuários é 6065. Neste caso temos a metade dos atendimentos, pois temos o dobro de usuarios no ciclo, reduzindo a quantidade de bandejas pela metade.

No caso de teste 3(1 fila de bandeja, 2 filas a de bandejas, com máximo de 40, 1 pilha de bandejas e 1 estação para servir comida), tem-se um tempo de atendimento médio de é 62.20 minutos, número de clientes atendidos é 230, somo total dos tempos de atendimento dos usuários é 14305. Neste caso temos a metade dos atendimentos, pois temos o dobro de usuarios no ciclo, reduzindo a quantidade de bandejas pela metade.

Conclusão

Podemos concluir então que o tempo de atendimento médio é diretamente ligado ao número pessoas que são atendidas no caixa, ao número de bandejas e principalmente ao número de locais para servir os 4 tipos de comida, sendo este o maior gargalo de atendimento. O segundo gargalo mais relevante é número de bandejas, já que muitos usuarios aumentam seu tempo de espera por esperar as bandejas serem repostas.

A configuração ideal seria análoga o caso 2, pois existe uma proporção mais harmônica na relação entre caixas, filas, pilhas e estações de atendimento, tendo tempo de atendimento médio de 55.14.