Gustavo Rocha Figueiredo Matrícula: 2015086409 UFMG

Sistemas de Informação

Compilar com: *gcc –Wall *.c –o main*

1.Introdução

O objetivo do trabalho é criar um sistema para calcular o tempo médio gasto na fila por cliente em uma lanchonete no intervalo de 4 horas. Nessa lanchonete há uma fila para o caixa e uma fila para pegar a bandeja. A cada 12 minutos 10 bandejas são repostas na pilha inicial de 30 e, caso o cliente não pegue fila alguma, ele demora 6 minutos para ser completamente servido. A cada minuto dois clientes chegam. Para a resolução do problema, foi criado um programa que simula a movimentação dos clientes e salva o tempo que cada um demorou para ser completamente servido, para que depois seja calculado o tempo médio de atendimento por pessoa.

2. Implementação:

Estrutura de dados:

Estruturas:

Filas: foram utilizadas três estruturas de fila, uma para representar a fila do caixa, outra para representar a fila para pegar a bandeja, uma para ser a fila de se servir os alimentos e a última onde se salva o tempo gasto para ser completamente servido de cada cliente. Em cada célula da lista se salva o tempo gasto por cada cliente no processo de ser servido e de quantos alimentos ele já se serviu (ao chegar em 4 alimentos, a célula é desenfileirada e enfileirada na fila que guarda o tempo gasto de cada cliente)
Pilha: foi utilizada uma pilha para representar a pilha de bandejas.
Funções e procedimentos:

void FPilhaVazia(TPilha *pilha) e void FFilaVazia (TFila *fila) : recebe um apontador para uma variável tipo pilha ou fila e cria a respectiva estrutura com uma célula cabeça, com apontadores para topo e fundo(em pilha) e frente e trás(fila) na mesma célula, além de tamanho zero.

int PVazia(TPilha pilha) e int FVazia(TFila fila): retorna se a fila ou pilha está vazia.

void Empilha(TItem x, TPilha *pilha) e void Enfileira(TItem x, TFila *fila): Recebe um tipo item e um apontador para uma pilha/fila e adiciona esse item no topo(pilha) ou na parte de trás (fila) da estrutura, além de adicionar em 1 o tamanho da estrutura.

void Desempilha(TPilha *pilha, TItem *item) e void Desenfileira(TFila *fila, TItem *item): Recebe um apontador para pilha/fila e o endereço de memória de um item, retira o último item adicionado (no caso de pilha) ou o primeiro (fila) e salva o conteúdo na variável item, além de retirar em 1 o tamanho da estrutura.

int PTamanho(TPilha pilha) e int FTamanho(TFila fila): Retorna o tamanho da pilha ou fila.

void liberaP (TPilha *pilha) e void liberaF (TFila *fila): Desaloca a memória da pilha ou fila.

void PassaTempo (TFila *fband, TFila *fcaixa, TFila *alim, TItem *caixa, TItem *band): Recebe um apontador para todas as estruturas do programa e para as variáveis que representam a pessoa pegando a bandeja e a pessoa sendo atendida no caixa. Percorre todas as estruturas adicionando em 1 o tempo de todas as células, inclusive a das variáveis que representam o caixa e bandeja.

void AndaAlimento (TFila *alim): Recebe um apontador para a estrutura fila que representa a fila para pegar alimentos. Faz com que todos integrantes dessa fila tenham seu atributo de quantos alimentos já se serviu aumentado.

void CriaBandeija (TPilha *pilha): Recebe um apontador para a estrutura pilha, e empilha até que essa tenha 30 bandejas.

void EncheBandeija (TPilha *pilha): Recebe um apontador para uma pilha e testa se essa tem tamanho 30. Caso tenha, retorna. Caso contrário, empilha até que se tenha 30 ou que se empilhe 10 vezes.
int TerminouServir (TFila fila): Recebe a fila de alimentos como parâmetro. Retorna se o primeiro elemento da lista já se serviu dos 4 alimentos.

void ClienteEmbora (TFila *alim,TFila *fim): Recebe um apontador para a fila de alimentos e outro para a fila de clientes que já terminaram de se servir completamente. Caso o cliente tenha terminado de se servir (**int TerminouServir** retorne 1), desenfileira da fila de alimentos e enfileira na fila de clientes já atendidos (fila fim).

Programa principal (main): O programa principal é um loop que enquanto a variável TEMPO não seja 240 (4 horas) irá se repetir. O programa roda de trás pra frente. Para que isso seja possível, o programa começa no tempo 1, com um cliente no caixa e outro na fila. A primeira coisa que se faz é passar o tempo específico de cada cliente (**void PassaTempo**), e então se testa se algum cliente já terminou de servir os alimentos e se já irá embora (**void ClienteEmbora**). Em seguida, faz com que todos os clientes na fila de alimento peguem o próximo alimento (**void AndaAlimento**). O próximo passo do algoritmo é testar se o TEMPO é múltiplo de 12 (para se colocar mais bandejas), caso seja, chama-se a função **void EncheBandeija.** Em seguida, testa-se se há alguém na posição de pegar a bandeja, caso tenha, esse item é enfileirado na fila de alimentos e a variável responsável por representar essa posição é zerada. O próximo passo é testar se há mais bandejas para que as pessoas da fila de bandeja possam pegar. Caso tenha, o caixa é enfileirado na fila de bandejas, e a primeira posição é desenfileirada e salva na variável que representa a pessoa pegando a bandeja. Caso não haja mais bandejas, a pessoa no caixa é enfileirada na fila para pegar bandejas. Para terminar o loop, enfileira-se na fila para o caixa dois novos clientes, e se desenfileira o primeiro, que é salvo na variável que representa o caixa, e se adiciona na variável TEMPO uma unidade. Por fim, acabado o loop, percorre-se a fila responsável por salvar os valores de tempo de cada cliente, salvando esses valores em uma variável **soma**, e divide-se essa variável pelo tamanho dessa fila, que resultará no tempo médio para cada cliente ser atendido.

3. Análise de complexidade:

void FPilhaVazia(TPilha *pilha) e void FFilaVazia (TFila *fila) : O(1), faz um número fixo de operações.

int PVazia(TPilha pilha) e int FVazia(TFila fila): O(1), faz somente uma operação, que é testar se o apontador para o topo e fundo (ou frente e trás no caso de fila) apontam para a mesma célula.

void Empilha(TItem x, TPilha *pilha) e void Enfileira(TItem x, TFila *fila): O(1) faz um número fixo de operações, independente da entrada e do tamanho da pilha/fila.

void Desempilha(TPilha *pilha, TItem *item) e void Desenfileira(TFila *fila, TItem *item): O(1), faz um número fixo de operações, independente da entrada e do tamanho da pilha/fila.

int PTamanho(TPilha pilha) e int FTamanho(TFila fila): O(1), só retorna o tamanho da pilha ou fila.

void liberaP (TPilha *pilha) e void liberaF (TFila *fila): O(n), desaloca cada elemento da fila/pilha (um por um). Depende do tamanho da estrutura e fará um número fixo de operações O(1) para cada elemento.

void PassaTempo (TFila *fband, TFila *fcaixa, TFila *alim, TItem *caixa, TItem *band): O(n), depende do tamanho da entrada e fará um número fixo de operações O(1) para cada elemento da entrada.

void AndaAlimento (TFila *alim): O(1), já que o tamanho máximo dessa fila é de 4.

void CriaBandeija (TPilha *pilha): O(1), pois irá empilhar até 30 bandejas. Porém, caso se considere que se possa mudar o número de bandejas, será O(n).

void EncheBandeija (TPilha *pilha): O(1), pois irá empilhar no máximo 10 bandejas.

int TerminouServir (TFila fila): O(1), faz um número fixo de operações para o primeiro elemento da fila.

void ClienteEmbora (TFila *alim,TFila *fim): O(1), faz um número fixo de operações e chama uma função também O(1).

Programa principal (main):$O(n^2)$ caso se considere o tempo máximo (4 horas) como variável. Caso contrário, O(n), referente a repetição da função **void PassaTempo** dentro do loop principal.

3.Resultados:

1 fila de caixa, 1 caixa, 1 fila de bandeja, 1 pilha de 30 bandejas, 10 bandejas a cada 12 minutos:

Tempo médio: 63 minutos.

1 fila de caixa, 2 caixas, 1 fila de bandeja, 1 pilha de 30 bandejas, 10 bandejas a cada 12 minutos:
Tempo médio: 64 minutos.

1 fila de caixa, 1 fila de bandeja, 1 pilha de 30 bandejas, 10 bandejas a cada 10 minutos:
Tempo médio: 64 minutos.

1 fila de caixa, 1 caixa, 1 fila de bandeja, 1 pilha de 60 bandejas, 10 bandejas a cada 12 minutos:
Tempo médio: 64 minutos

3.Conclusão:

A partir desses resultados é possível chegar à conclusão que é necessário aumentar (dobrar) todas as partes do processo de atendimento para se ter um atendimento melhor. Caso somente uma das partes do processo seja melhorada, não trará melhora visível, já que se formará um gargalo em outra parte, pois a cada minuto aparecem 2 novos clientes, e todas as partes do processo somente atendem uma pessoa a cada tempo. Exemplo: 2 caixas e 2 pilhas de bandeja, porém somente uma fila de alimento teria basicamente o mesmo resultado, pois somente uma pessoa consegue servir cada alimento por vez, sendo então necessária a criação de outra fila para que os alimentos sejam servidos. O atraso causado pela reposição das bandejas é basicamente desprezível, pois o maior atraso se dá pela falta de caixas e de pilhas de bandejas. A bandeja acaba pela primeira vez com cerca de 180 minutos (aproximadamente), atrasando poucas pessoas e somente de 10 em 10 minutos, bem menos que a deficiência de caixas e pilhas, que atrasam uma pessoa por minuto. Porém, em um cenário ideal, com 2 caixas, duas pilhas de bandejas e duas filas de alimento, a falta de reposição se tornaria o gargalo, sendo necessário ao menos dobrar a taxa de reposição. Os efeitos de mudança de somente a quantidade de bandejas e/ou taxa de reposição só seriam notados em intervalos de tempos maiores.