UNIVERSIDADE FEDERAL DE MINAS GERAIS  
INSTITUTO DE CIÊNCIAS EXATAS  
Disciplina: Algoritmos e Estrutura de Dados II  
Professor: Adriano Veloso  
Nome: Daniel Vasconcelos Chaves de Miranda  
  
Trabalho Prático  
  
**Introdução**  
O problema proposto consiste em simular um dia de atendimento na cantina do ICEX durante o horário de almoço. Deveria se implementar uma fila de pessoas até o caixa para comprar uma ficha, uma pilha de bandejas com um número máximo de 30 unidades, uma fila de pessoas que já compraram a ficha e esperam para pegar a sua bandeja e considerar o fato de que cada um serviria-se de 4 ingredientes diferentes antes de sair da fila. Cada iteração deve ser contada como um espaço de tempo no qual cada pode ser contado como um minuto, a simulação deve ser sobre um período de 4 horas, considerado o tempo do horário de almoço. Ao final deveria ser impresso na tela a média de tempo dos clientes na fila.  
  
**Desenvolvimento**Foram utilizados um TAD Pilha e um TAD Fila, além de funções desenvolvidas no programa principal que manipulavam esses TAD’s para conseguir solucionar o problema corretamente. Ambos os TAD’s forma implementados com apontadores e a struct Item do TAD Fila continha três inteiros, um para contar o tempo, um para ver se o Item já havia passado pela pilha de bandejas e pegado sua bandeja (recebia 0 se não e 1 se sim), e um inteiro chave que significava qual cliente era aquele.  
No programa principal é controlado as duas filas e a pilha proposto no problema tratando um a um os clientes em um loop e somando o tempo final de cada cliente que já havia se servido e após o loop esse tempo total é dividido pela chave do último cliente que saiu da cantina, obtendo então a média de tempo.  
  
**Implementação**Main:  
  
A main é bem simples, ela declara e faz as duas filas e a pilha de bandeija com recursos dos TAD’s Pilha e Fila e chama a função CresceFila.  
Funções:  
void CresceFila (TFila \*Fila, TFila \*Fila2, TPilha \*Pilha) :  
  
A função declara cinco variáveis inteiras para contar o tempo, tempC controla o tempo até o caixa, tempB o tempo até a pegar a bandeja, tempS o tempo até servir, tempT é a soma do tempo total de todos os clientes que já serviram e tempI conta o tempo desde o início.   
Um loop acrescenta n de dois em dois até este chegar do número máximo de clientes estabelecido, enquanto chama as funções Caixa, ReporBandeja, PegaBandeja e Servir para que cada cliente seja tratado um a um, mesmo eles estando crescendo de dois em dois. Isso resolve o problema da fila do caixa aumentar em dois clientes a cada espaço de tempo.  
Outro loop é chamado depois, este continua enquanto a fila de bandejas ainda conter itens, ele trata cliente por cliente chamando as mesmas funções que o primeiro loop, e está ali para quando todos os clientes já foram postos na primeira fila mas ainda não serviram.  
Ambos os loops tem uma condição de que se o tempI for maior ou igual a 480, ou seja, 4 horas desde que o processo começou, o loop é terminado. No final dessa função tempT é dividido pela chave do último item que foi retirado da segunda fila e dessa forma é obtida a média de tempo que os clientes passam na fila, esse valor é impresso na tela.  
  
void Caixa (TFila \*PrFila, TFila \*SgFila, int \*temp):  
  
Essa função desenfileira um cliente da primeira fila, adiciona um no tempo de espera do caixa se o item do cliente for ímpar e soma esse valor ao valor tempo dentro da struct item e enfileira esse mesmo cliente na fila para as bandejas.  
  
void ReporBandeja (TPilha \*Pilha);  
  
Essa função checa se a pilha está cheia e se não estiver ela vê se o tamanho é menor que 10, e nesse caso ela repõe o número máximo de bandejas que não ultrapasse o número máximo estipulado. Se estiverem faltando mais que 10, a função repõe 10 bandejas.  
Essa função é chamada na função cresce fila toda vez que o tempo desde o início for divisível por 12.  
  
void PegarBandeja (TFila \*Fila, TPilha \*Pilha, int \*temp, TipItem \*b, TItem \*a):  
  
Essa função acrescenta o tempo de espera pela bandeja e checa se a pilha de bandejas está vazia, se ela não estiver ela desenfileira um item da segunda fila, soma ao tempo dele no valor da struct item referente ao tempo.  
  
void Servir (TItem \*a, int \*temp):  
  
Essa função acrescenta um ao tempo de servir, se esse tempo é divisível por quatro ele é somado ao tempo armazenado em item.  
  
**Complexidade:**Não existem loops aninhados e o único loop, apesar de crescer de dois em dois, tem complexidade O(n), pois trata cliente por cliente utilizando as funções, que tirando a que a única que contém o loop, têm complexidade O(1). Desta forma, a complexidade do programa como um todo é da ordem de O(n).  
  
**Resultados:**O tempo médio do teste base é de: 46.75 minutos  
O tempo médio do teste com dois caixas é de: 31.5 minutos  
O tempo médio 40 bandejas máximas na pilha é de: 46.75 minutos  
  
**Conclusão:**A mudança efetiva para a otimização do tempo é aumentar o número de caixas, pois o número ótimo de bandejas máximas para a pilha nesse problema já foi atingido.