UNIVERSIDADE FEDERAL DE MINAS GERAIS

TRABALHO PRATICO 1:
GESTÃO DO RESTAURANTE DA CANTINA DO ICEX

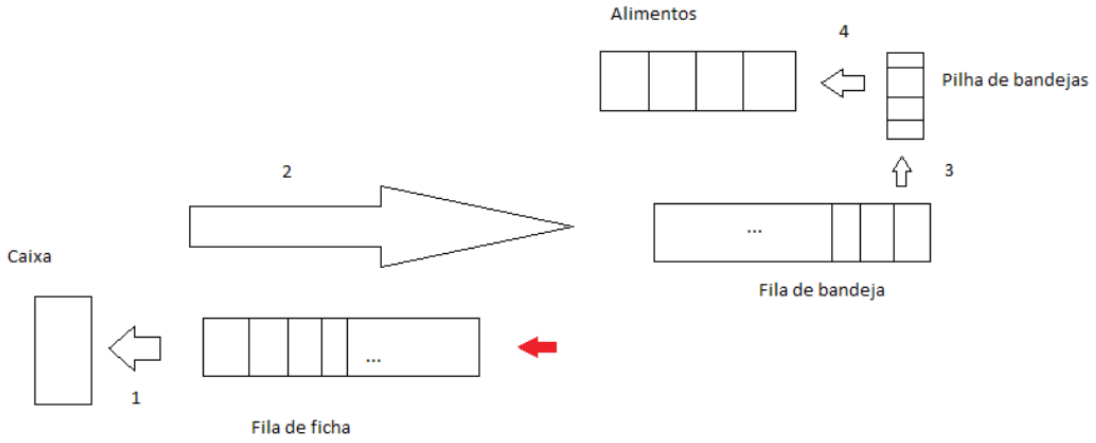
Trabalho apresentado para a disciplina AEDS II, professor Adriano Veloso, compondo a media da disciplina, por Fernando Elias 2016109089, aluno do curso de Matematica, como Formação Livre.

Belo Horizonte

Maio de 2017

INTRODUÇÃO

O restaurante da cantina do icex, funciona durante 4 horas para almoço servindo refeições, possuem caixas e os estudantes formam filas para comprar fichas, quando chegam ao caixa demoram 1 minuto para comprar a ficha, em posse da ficha segue para fila de bandejas, quando chega a pilha de bandejas demora 1 minuto para pega-lá juntamente com todos talheres e podem se servir (arroz, feijão, guarnição e salada) levando 4 minutos. Sendo que o caixa fecha para compra de fichas, após 4 horas e todos clientes que já compraram a ficha devem ser atendidos. O gerente da cantina deseja saber o tempo médio para atendimento de um cliente e quais modificações fazer para a cantina funcionar melhor. Observe a figura, ela ilustra o problema.



IMPLEMENTAÇÃO

Vamos usar a estrutura fila para controlar acesso ao caixa e as bandejas e para organizar as bandejas, usamos a estrutura pilha.

Observação: Usamos como Chave para a fila o tempo em minutos que o cliente chegou. Logo o tempo gasto por ele para se servir é igual a (tempo atual – tempo em que chegou + 5). Estes 5 minutos são o tempo que gastou para pegar a bandeja (1minuto) e para se servir (4minutos).

Abaixo o algoritmo usado e que foi implementado em C utilizando o conceito de TAD, pilha e fila. O codigo completo consta em 5 Anexos.

PSEUDOCODIGO DO MAIN

INICIO

Declaração de variaveis, poteiros, inclusão dos TAD pilha e fila.

Empilhar todas as bandejas antes de abrir o restaurante;

Faça{ Se ainda não deu 4 horas então coloque dois cientes na fila;

Se a fila do caixa não estiver vazia, retirar cliente e passar para fila de bandeja;

Se a Pilha de bandejas não estiver vazia então retirar um cliente da fila e uma bandeja da pilha e adicionar o tempo gasto pelo cliente ao tempo total;

Se o resto na divisão de "tempo atual" por "intervalo para haver reposição" for nulo então reponha k bandejas;

} enquanto (não der 4 horas ou fila de caixa vazia ou fila de bandejas vazia);

media= tempo total dividido por número de clientes;

Imprimir a media;

Liberar a memoria;

FIM

A implementação da TAD pilha e fila possuem complexidade O(1), pois fazem somente atribuições e rodam uma única vez.

Enquanto no main temos um (do while) no melhor caso rodará n vezes, e sua complexidade é O(n).

Os demais itens, fazem somente atribuições, alocações, comparações ou teste lógico, logo são constantes, ou seja, O(1)

TESTES E A INTUIÇÃO

Para o caso inicial: Duas pessoas chegando na fila por minuto, tendo 30 bandejas e a cada 12 minutos repondo 10. A intuição nos leva que o primeiro cliente gasta 6 min, o segundo 7 min e o terceiro (chegou no instante 2 e saiu no instante 9).

Tempo total= 6 + 7 + 7 + 8 + 8 + - - - + 244 + 244 + 245 + 245 + 246

= (7 + 245)*(239) + (246 + 6) = 240*252

Tempo Medio = 240 * 252 / 480 = 126 minutos.

Mas quando compilamos o codigo a resposta é 152 min. Parece que foi um erro, mas na verdade esquecemos de considerar que havera instantes em que a pilha de bandejas está vazia. Adaptando para repor, por exemplo, 10 bandejas a cada 8 minutos, nunca a pilha de bandejas estara vazia, agora a intuição coincide com a realidade.

Para variarmos o número de fila de bandejas, pegamos o if que desenfileira do caixa e o replicamos o qunto quisermos, ou colocamos um for que varia de 1 até o número de filas que desejarmos. Analogamente para o if que desempilha e para o que empilha, já para variar o número de bandejas iniciais, repostas e tempo de reposição, simplesmente modificamos as constantes predefinidas(#define).

Assim o codigo roda todas as configurações que desejar, inclusive as da tabela.

Nº Fila Caixa	Numero Caixas	Nº Fila Bandeja	Numero Bandejas	Numero Reposição	Tempo de Reposição	Tempo Médio
1	2	1	30	10	12	152 min.
1	2	1	30	10	8	126 min.
1	2	1	40	18	15	66 min.
1	2	2	30	10	12	147 min.
1	2	2	40	18	15	6 min.

CONCLUSÃO

Alcançamos a proposta de simular o funcionamento da cantina e otimizar o tempo médio. Como temos dois clientes chegando a cada minuto, o ideal seria ter dois caixas, pois todo cliente que chega passará somente 1 minuto na fila de caixa, para fila de bandejas o ideal é termos 2 filas de bandejas, com duas pilhas e com uma taxa de reposição maior do que ou igual ao tempo para a pilha esvaziar, logo bandejas não serão um problema. E os dois clientes que pegaram bandejas no mesmo instante de tempo, ter dois lugares/espaços para servir e cada um tendo os 4 itens, assim eles iriam direto e gastariam 4 min, logo o tempo medio é 6 minutos por cliente, este é o tempo otimo, caso uma das condiçoes ideal no mínimo (2 caixas, 2 filas de bandejas, 2 pilhas de bandejas com as pilhas de bandejas nunca vazias e 2 lugares/espaços para servir) não seja satisfeita, o tempo medio será maior que 6 min.

ANEXO 1 : TAD Fila.h

```
#ifndef _FILA_H_
#define _FILA_H_

// DEFINIÇÃO DO TIPO
typedef int TipoChave;
typedef struct TItem {
   TipoChave ChaveFila;
   // outros componentes
} TItem;
typedef struct CelulaFila   *ApontadorFila;
typedef struct CelulaFila {
   TItem Item;
   ApontadorFila Prox;
} TCelula;
typedef struct {
   ApontadorFila Frente;
   ApontadorFila Tras;
   int TamanhoFila;
} TipoFila;

// CABEÇALHO DAS FUNÇOES
void FFVazia(TipoFila *Fila);
int Vazia(TipoFila *Fila);
int TamanhoFila(TipoFila *Fila);
void Enfileira(TItem Item, TipoFila *Fila);
void Desenfileira(TipoFila *Fila, TItem *Item);
#endif
```

ANEXO 2 : TAD Fila.c

```
#include "fila.h"

void FFVazia(TipoFila *Fila)
{
   Fila->Frente = (ApontadorFila) malloc(sizeof(TCelula));
   Fila->Tras = Fila->Frente;
   Fila->Frente->Prox = 'NULL';
   Fila->TamanhoFila=0;
}

// Esta função retorna 1 (true) se a ‘Fila’ está vazia; senão retorna 0 (false)
int Vazia(TipoFila *Fila)
{
   return(Fila->TamanhoFila == 0);
}

int TamanhoFila(TipoFila *Fila)
{
   return (Fila->TamanhoFila);
}

void Enfileira(TItem x, TipoFila *Fila)
{
   Fila->Tras->Prox = (ApontadorFila) malloc(sizeof(TCelula));
   Fila->Tras = Fila->Tras->Prox;
   Fila->Tras->Item = x;
   Fila->Tras->Prox = 'NULL';
   Fila->TamanhoFila ++;
```

```
}

void Desenfileira(TipoFila *Fila, TItem *x)
{
   if (Vazia(Fila))
   {
      printf("Erro: A fila está vazia!!!\n");
      return 0;
   }
   ApontadorFila q = Fila->Frente;
   Fila->Frente = Fila->Frente->Prox;
   *x = Fila->Frente->Item;
   free(q);
   Fila->TamanhoFila --;
}
```

ANEXO 3 : TAD Pilha.h

```
#ifndef _PILHA_H_
#define _PILHA_H_

 // DEFINIÇÃO DO TIPO
typedef int TChave;
typedef struct {
   TChave ChavePilha;
   // outros componentes
} TipoItem;
typedef struct CelulaPilha      *ApontadorPilha;
typedef struct CelulaPilha {
   TipoItem Item;
   ApontadorPilha Prox;
} TipoCelula;
typedef struct {
   ApontadorPilha Fundo;
   ApontadorPilha Topo;
   int Tamanho;
} TPilha;

// CABEÇALHO DAS FUNÇOES
void FPVazia(TPilha *Pilha);
int Vvazia(TPilha *Pilha);
void Empilha(TipoItem Item, TPilha *Pilha);
void Desempilha(TPilha *Pilha, TipoItem *Item);
int Tamanho(TPilha Pilha);
#endif
```

ANEXO 4 : TAD Pilha.c

```
#include "pilha.h"
void FPVazia(TPilha *Pilha) {
    Pilha->Topo = (ApontadorPilha*) malloc(sizeof(TipoCelula));
    Pilha->Fundo = Pilha->Topo;
    Pilha->Topo->Prox = 'NULL';
    Pilha->Tamanho = 0;
}
int Vvazia(TPilha *Pilha) {
    return (Pilha->Topo == Pilha->Fundo);
}
void Empilha(TipoItem x, TPilha *Pilha) {
    ApontadorPilha Aux = (ApontadorPilha*) malloc(sizeof(TipoCelula));
    Pilha->Topo->Item = x;
    Aux->Prox = Pilha->Topo;
    Pilha->Topo = Aux;
    Pilha->Tamanho++;
}
void Desempilha(TPilha *Pilha, TipoItem *Item) {
    if (Vazia(*Pilha)) {
        printf("Erro lista vazia\n");
        return;
    }
    ApontadorPilha q = Pilha->Topo;
    Pilha->Topo = q->Prox;
    *Item = q->Prox->Item;
    free(q);
    Pilha->Tamanho--;
}
int Tamanho(TPilha Pilha) {  return Pilha.Tamanho; }
```

ANEXO 5 : main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "fila.h"
#include "pilha.h"
#define ban 30 //NUMERO DE BANDEJAS
#define rep 10 //NUMERO DE BANDEJAS REPOSTAS
#define temp 12 //TEMPO PARA REPOR "rep" BANDEJAS
int main ()
{
   int i=1, k, m, t, p, soma=0, MediaPorCliente;
   //FILAS DE CAIXA E BANDEJA
   TipoFila FilaCaixa, FilaBandeja;
   TItem Cliente;
   FFVazia(&FilaCaixa);
   FFVazia(&FilaBandeja);
//PILHA DE BANDEJA
   TPilha PilhaBandeja;
   TipoItem Bandeja;
   Bandeja.ChavePilha=1;
   FPVazia(&PilhaBandeja);
   //EMPILHANDO TODAS AS BANDEJAS/TALHERES ANTES DE ABRIR O
RESTAURANTE
   for(k=1; k<=ban; k++)
      {
         Bandeja.ChavePilha=k;
         Empilha(Bandeja, &PilhaBandeja);
         //  printf("\nBandeja: %d", Bandeja.ChavePilha);
      }
  //  printf("\nTamanho pilha Bandeja: %d", Tamanho(PilhaBandeja));
```

```
// CHEGADA DOS DOIS PRIMEIROS CLIENTES
            Cliente.ChaveFila=0;
            Enfileira(Cliente, &FilaCaixa);
            Enfileira(Cliente, &FilaCaixa);

do
   {
      //DEPOIS DE 4 HORAS NÃO ENTRA CLIENTE NA FILA
      if(i<240)
         {
            Cliente.ChaveFila=i;
            Enfileira(Cliente, &FilaCaixa);
            Enfileira(Cliente, &FilaCaixa);

         }

      if(Vazia(&FilaCaixa)==0)
         {
            Desenfileira(&FilaCaixa, &Cliente);
            Enfileira(Cliente, &FilaBandeja);
         }


      if(Vvazia(&PilhaBandeja)==0)
         {
            Desenfileira(&FilaBandeja, &Cliente);
            Desempilha(&PilhaBandeja, &Bandeja);
          //  printf("\nCliente numero: %d", Cliente.ChaveFila);
           // printf("\nBandeja: %d", Bandeja.ChavePilha);
            MediaPorCliente = (i - Cliente.ChaveFila) +5;
```

```
                soma = soma + MediaPorCliente;
            }

        if(i%temp==0) // REPÕE 10 BANDEJAS A CADA 12 MINUTOS.
            {
                for(m=1; m<=rep; m++)
                    {
                        Bandeja.ChavePilha=m;
                        Empilha(Bandeja, &PilhaBandeja);
                    }
            }
        //  printf("\nTamanho pilha: %d", Tamanho(PilhaBandeja));
        //  printf("\nSoma: %d", soma);
          i++;
          t = Vazia(&FilaBandeja);
          p = Vazia(&FilaCaixa);
       }
   while((i<=240)||(t==0)||(p==0));

//   printf("\n Tamanho caixa: %d", TamanhoFila(&FilaCaixa));
//   printf("\n Tamanho bandeja: %d", TamanhoFila(&FilaBandeja));
   printf("\n Media: %d", soma/480);
 //  printf("\nBanana: %d ", i-1);
   free(&FilaCaixa);
   free(&FilaBandeja);
   free(&PilhaBandeja);
   return 0;
}
```