Introdução:

O problema proposto foi desenvolver um programa para simular o funcionamento de um dia na hora do almoço na cantina do ICEX, seguindo as instruções a seguir:

• Um usuário da cantina chega na fila de compra de ficha;
• Quando ele chegar no caixa para comprar a ficha significa que ele saiu da fila (o usuário da primeira posição da fila é o que será atendido em seguida);
• Com a ficha em mãos, o usuário irá para outra fila: a de bandejas e talheres;
• Um usuário que estiver no momento de pegar uma bandeja (inclusos talheres e pratos) já está fora da fila (como no caso do caixa);
• Um usuário só poderá pegar uma badeja caso exista pelo menos uma na pilha de bandejas, caso contrário, ele deverá esperar para reposição;
• O ato de ser servido de um alimento (arroz, feijão, guarnição e salada) consome um espaço de tempo (iteração), dessa forma, para que o usuário complete seu
prato, ele consumirá 4 espaços de tempo.

Como solução, o programa criado utilizou os tipos de estruturas de dados (TADs) que foram conhecidos durante as aulas para organizar o funcionamento da cantina.

Desenvolvimento:

Foram utilizadas pilhas e filas para fazer a distribuição dos processos, as estruturas criadas para a solução foram:

TFila FilaCompra – Fila onde os usuários que entram no processo são inseridos, e aguardam para serem atendidos no caixa e então realizar a compra de fichas.

TFila FilaBandeja – Após a compra de fichas o usuário é inserido nesta fila para aguardar e retirar uma bandeja, assim como os talheres para a refeição, levando em conta a disponibilidade de bandejas;

TFila FilaComida – Esta estrutura não estava prevista na descrição do trabalho, mas foi adicionada para controlar a distribuição dos alimentos e finalização do atendimento.

TPilha PilhaBandeja – Armazena os conjuntos de bandejas e talheres para a retirada dos usuários antes de se servirem com os alimentos.

Foi utilizada também uma estrutura interativa (for) para processar o andamento do tempo, como o problema visa verificar o funcionamento do estabelecimento durante 4 horas, a interação vai de 0 a 239, sendo cada interação equivalente a um minuto.

Implementação:

(1) Funções:

void FPVazia(TPilha *Pilha) – Cria uma pilha vazia a partir de um ponteiro.

int PilhaVazia(TPilha Pilha) - Verifica se a pilha se encontra vazia.

void Empilha(TItem x, TPilha *Pilha) – Adiciona um item à pilha de dados.

void Desempilha(TPilha *Pilha, TItem *Item) – Retira o item no topo da pilha e o coloca no ponteiro passado como argumento.

int Tamanho(TPilha Pilha) – Retorna o tamanho atual da pilha.

void FFVazia(TFila *Fila) – Cria uma Fila vazia a partir de um ponteiro.

int FilaVazia(TFila Fila) – Verifica se a fila se encontra vazia.

void Enfileira(TItem x, TFila *Fila) - Adiciona um item na ultima posição da fila.

void Desenfileira(TFila *Fila, TItem *Item) – Retira o primeiro item da fila e o coloca no ponteiro passado como argumento.

void AbreRestaurante(TPilha *Bandeja,TFila *Compra,TFila *Serve,TFila *Comida) – Inicializa as estruturas de dados que serão utilizadas no funcionamento do programa.

(2) Variáveis Pontuais:

Diferenca – Auxilia na reposição de bandejas executada a cada 12 tempos, para evitar que a quantidade reposta ultrapasse o limite da pilha(30 bandejas).

TempoAtual – Faz a contagem do tempo corrido a cada interação.

TempoTotal – Contabiliza o tempo total de atendimentos realizados cada vez que um usuário finaliza o processo.

TempoDeFila – Ajuda no controle da fila criada para os usuários se servirem, garantindo que eles cumpram os 4 tempos gastos para se servir de cada alimento.

NumeroAtendimentos – Contabiliza o número de usuários atendidos durante o funcionamento do programa.

ContadorChave – Auxilia na geração das chaves de cada usuário para verificação de cada atendimento no log de funcionamento do programa.

(3) Estruturas interativas e condicionais:

for(TempoAtual=0;TempoAtual<240;TempoAtual++) – Loop que faz a contagem dos tempos e onde se encontra todo o desenvolvimento do processo de atendimento dos usuários, principal interação do programa.

if((!(FilaVazia(FilaComida)))&&(TempoDeFila>=4)) – Verificar se existem usuários na fila onde são servidos os alimentos, e caso o tempo de fila seja maior que 4, ou seja, o usuário já se serviu de todos os alimentos, realiza a finalização do atendimento.

if((!(FilaVazia(FilaBandeja)))&&(!(PilhaVazia(PilhaBandeja)))) – Verifica se existem usuários na fila para retirar bandeja e talheres, e se a pilha de bandejas está disponível para que o usuário prossiga com o atendimento e vá para a fila onde são servidos os alimentos.

if((TempoAtual % 12)==0) – Verifica o momento de reposição de bandejas na pilha, que deve ser a cada 12 tempos.

Análise:

Como o algoritmo possui um tempo de funcionamento previamente definido (4h ou 240 minutos), a principal estrutura de interações tem numero de operações definida.

Assim temos a função de complexidade em função das comparações e atribuições para iniciação da pilha de bandejas:

$F(n) = 3n + 30$

$F(240) = 3 \times 240 + 30 = 750$.

Ordem de Complexidade = O(1).

Conclusão:

Diante dos testes realizados foi observado que o programa funciona melhor com duas filas para o caixa, chegando a reduzir o tempo médio de atendimento pela metade, pois como a fila de novos usuários cresce mais rápido do que a velocidade de atendimento do caixa, o tempo de atendimento dos usuários cresce a cada interação, o que torna o processo ineficiente, com tempo médio de atendimento de 1h, o que impossibilita o atendimento de todos os usuários.

Observação: Na descrição do problema não ficou claro como proceder ao fim das 4h de tempo de funcionamento da cantina. Não ficou claro se o programa deveria encerrar a entrada de novos usuários e finalizar o atendimento de todos, ou apenas encerrar as atividades, e como o tópico criado com essa dúvida não foi respondido a tempo, optei por apenas encerrar e contabilizar os usuários que finalizaram o atendimento dentro das 4h de funcionamento.