Documentação

1. Introdução:

No trabalho prático foi proposto que fosse realizado um programa que simulasse o dia a dia de um restaurante universitário (por exemplo o bandejão da UFMG). O objetivo principal do trabalho era saber qual a média de tempo os usuários gastavam para completar o atendimento. Para ser atendido, o usuário deveria passar por uma fila para comprar a ficha, uma fila para pegar a bandeja, e servir-se do alimento. O tempo gasto para comprar a ficha e pegar uma bandeja, gastasse 1 minuto. Já para servir-se do alimento, ele gastaria 4 minutos, desconsiderando-se que não haja fila alguma o primeiro usuário gasta 6 minutos para encerrar todo o processo.

2. Desenvolvimento

Para resolver o problema proposto foi implementado uma estrutura abstrata de dados com as implementações da estrutura de uma lista e uma pilha. Foi utilizado a lista para enfileiras as pessoas. Já que a estrutura da fila funciona como *FIFO* (*First in, first out*), ou seja, o primeiro elemento a entrar na estrutura é o primeiro a sair, assim a primeira pessoa a entrar na fila será a primeira a ser atendida. Já a estrutura de pilhas foi utilizado para empilhar bandejas, pois essa estrutura funciona como *LIFO - Last In, First Out*, ou seja, o último a entrar é o primeiro a sair da estrutura.

No primeiro caso foi implementado uma fila de ficha, uma fila de bandeja, uma pilha de bandeja contendo 30 bandejas.

No primeiro caso foi implementado duas filas de fichas, uma fila de bandeja, duas pilhas de bandejas contendo 30 bandejas.

No primeiro caso foi implementado uma fila de ficha, uma fila de bandeja, uma pilha de bandeja contendo 30 bandejas.

No primeiro caso foi implementado uma fila de ficha, duas filas de bandejas, uma pilha de bandeja contendo 40 bandejas.

3. Implementação

Inicialmente foi declarada as variáveis: filasfichas, filasbandejas, pilhasbandejas, bandejas. Para serem executadas no while de acordo com cada um dos casos testes. Posteriormente foram iniciadas todas as filas e pilhas criadas. Além disso foi feito um while para preencher a pilha de bandejas com a quantidade de bandejas definida em cada um dos casos:

```
//colocar numeros de bandejas
while(j<bandeijas){
```

```
        Empilha(qtdPBandeja,&PilhaBandeja);
        j++;
    }
```

Para todos os casos testes foi implementado um WHILE para realizar todos os procedimentos durante o tempo proposto de 4 horas e utilizado uma variável tempo :

```
while(tempo<240){
...


tempo++;
}
```

Dentro desse while principal foi desenvolvido um while para simular a quantidade de fila para comprar a ficha existente. Por exemplo no caso 2 caso teste deve ser implementado duas filas para comprar a ficha assim a variavel filasficha=2 e o while abaixo ira ser realizado duas vezes simulando a existencia de duas filas para a compra de ficha :

```
while(i<filasfichas){
            Enfileira(qtdFila,&FilaFicha);//Entrou 1 pessoas
            filaatendidos++;
            Enfileira(qtdFila,&FilaFicha);//Entrou 1 pessoas
            filaatendidos++;
            Desenfileira (&FilaFicha,&qtdFila);
            cronometro++;
            repor++;
            Enfileira(qtdFila,&FilaBandeja);


            i++;
        }
```

Dentro dessa estrutura, que é universal para todos os casos é realizado duas vezes a função "Enfileira(qtdFila,&FilaFicha)" pois a cada minuto passado irá entrar duas pessoas nesta fila. E é feito um contador chamado "cronometro" para estipular o tempo gasto pela pessoa para completar seu atendimento.

A posteriori, é implementado outro while para simular a existência de 'k' filas de bandejas. Por exemplo no caso 3, existem duas filas de bandejas, assim todo o conteúdo do while será executado duas vezes.

```
while(k<filasbandejas){
                    ....
                    k++;
    }
```

Dentro dessa estrutura é implementado um if que é universal para todos os casos. Sua função é verificar se existem bandejas na pilha. Se existir o usuário poderá entrar na fila de bandejas e prosseguir com seu atendimento. Dentro dessa estrutura é realizado um Desenfileira na fila de bandejas e um Desempilha na pilha de bandejas assumindo que a pessoa agora está servindo os alimentos.

```
if (!VerificaPVazia(PilhaBandeja)){
            Desenfileira (&FilaBandeja,&qtdFila);
            Desempilha(&PilhaBandeja,&qtdPBandeja);
            cronometro++;
            repor++;
              ...
            }
```

Depois é realizado uma verificação para verificar se o contador 'repor' atingiu 12 minutos , que foi o tempo estipulado no trabalho para realizar a reposição das bandejas. Se essa variável for 12 e a quantidade de bandejas for menos que 20 o programa irá entrar em uma função para repor mais 10 bandejas na pilha. Assim a pessoa espera mais um minuto para completar seu atendimento

```
            if (repor==12){
               repor=0;
               if (TamanhoPilha(PilhaBandeja)<20){
                  j=0;
                  while(j<10){
                     l=0;
```

```
            while(l<pilhasbandejas){
                Empilha(qtdPBandeja,&PilhaBandeja);
                l++;
            }
            j++;
        }
        cronometro++;// tempo a mais para esperar a repoicao
    }
```

Após esse passo é incrementado a variável cronometro de mais 4 minutos para simular que ela esta servindo seu almoço.

Para compilar o programa é necessário compilar cada um dos casos:

gcc –c caso1.c

gcc –c caso2.c

gcc –c caso3.c

gcc –c casoAleatorio.c

E o tad.c que contem a implementação das listas e filas

gcc –c tad.c

E posteriormente compilar o main:

gcc main.c caso1.o caso1.o caso2.o caso3.o casoAleatorio.o tad.o

4. Analise

Analise de complexidade das principais funções:

```
while(j<bandeijas){      O(1)
    Empilha(qtdPBandeja,&PilhaBandeja);
    j++;
}
```

```
while(tempo<240){
```

```
    i=0;
    while(i<filasfichas){     O(1)


       ...
    }
     k=0;
    while(k<filasbandejas){ O(1)
          ...
                while(j<10){    O(n)
                   l=0;
                   while(l<pilhasbandejas){ O(1)
                      ...
                   }
       }
    tempo++;
}
```

5. Resultados

Foi observado que para cada uma das implementações o tempo aumenta ou diminui de acordo com a quantidade de filas e pilhas existentes.

6. Conclusão

A partir de comparações feito com os resultados foi possível verificar que o segundo caso foi de pior desempenho pois entram mais pessoas na fila sobrecarregando cada vez mais as filas