**1. Introdução**

A manipulação de tipos abstratos de dados (TAD) é um dos aspectos fundamentais em Ciência da Computação. Exemplos especiais são as Listas, Pilhas e Filas que pode, fazer um trabalho, a principio com vetores, ser mais eficiente com esses tipos abstratos de dados.

O objetivo desse trabalho é implementar vários Tipo Abstrato de Dados (TAD), seja lista, fila ou pilha,para a manipulação de dados. Deverão ser implementadas funções para a utilização básica desses tipos como, por exemplo, inserção e remoção. Esse TAD será testado em um programa que controla o tráfego de pessoas em um ambiente. Espera-se com isso praticar os conceitos básicos de programação.

* 1. **O Caso a ser tratado**

*“Todos os dias, muitas pessoas almoçam no restaurante da cantina do ICEX. Um dos problemas enfrentados pelos usuários, no entanto, são as grandes filas enfrentadas para pagar pela comida e se servir. Além disso, a quantidade de bandejas e pratos disponíveis é limitada, podendo ocorrer falta desses recursos dependendo da situação. A pessoa que gerencia a cantina quer contratar um sistema que melhore a distribuição de tarefas dentro da cantina. Para isso, ele contará com os alunos de AEDS II da UFMG do semestre 2017/1 para criar uma solução.*

*Problema:*

*Os alunos deverão desenvolver um programa que simule um dia na hora do almoço da cantina do ICEX. Para tal, eles deverão utilizar as estruturas de dados fila e pilha. O cenário é o seguinte:*

*• 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;*

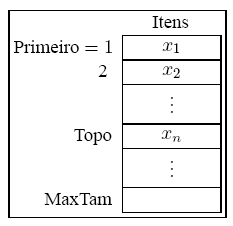
*• O ato de ser servido de um alimento (arroz, feijão, guarnição e salada) consome um espaço de tempo (iteração), dessa forma, para que o usuário complete seu prato, ele consumirá 4 espaços de tempo.”*

**2. Implementação**

**2.1 Estruturas de Dados Utilizadas**

2.1.1 Pilha Implementada com Arranjos: Para a implementação de uma pilha de bandejas foi usada uma pilha com arranjos. A quantidade máxima da pilha pode ser alterada em *main.c,* onde está definido MAX para 30, por padrão, o tamanho máximo da pilha de bandejas.

A pilha funciona da seguinte forma:

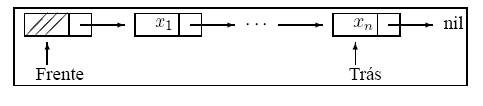
  
 *Diagrama da Pilha Implementada com Arranjos.*

2.1.1.1 Funções e Procedimentos

O TAD criado possui as seguintes funções:

* **void FPVaziaV(TPilha \*):** Faz a Pilha ficar vazia.
* **void EmpilhaV(TBANDEJA, TPilha \*):** Insere o item x no topo da pilha.
* **int DesempilhaV(TPilha\*, TBANDEJA\*):** Retorna o item x no topo da pilha, retirando-o da pilha.
* **int VaziaV(TPilha):** Esta função retorna true se a Pilha está vazia; senão retorna .false.
* **int TamanhoV(TPilha):** Retorna o tamanho da Pilha.

2.1.2 Fila Implementada com Apontadores: Para a implementação de todas as filas necessárias nesse caso, o uso deste TAD foi essencial. Na fila da ficha, na fila da bandeja e na fila antes de servir o alimento, esse TAD foi implementado para o controle do tráfego de pessoas.



*Diagrama da Fila Implementada com Apontadores.*

2.1.2.1 Funções e Procedimentos

O TAD criado possui as seguintes funções:

* **void FFVazia(TFila \*):** Faz a fila ficar vazia.
* **void Enfileira(TItem , TFila \*):** Insere o item x no final da fila.
* **void Desenfileira(TFila \*, TItem \*):** Retorna o item x no início da fila, retirando-o da fila.
* **void Enfileira(TItem , TFila \*):** Esta função retorna true se a fila está vazia; senão retorna false.

**2.2 Programa Principal**

O programa principal cria uma fila para pegar uma ficha *(Fila\_ficha)*, uma fila para pegar a bandeja *(Fila\_bandeja)* e uma fila para se servir com a comida *(Fila\_comida).* Além disso, cria também a pilha onde as bandejas são armazenadas *(Pilha\_bandeja).* Nesse momento, variáveis como *CAIXA\_1* e *CAIXA\_2* são criadas. Essas variáveis servem para armazenar os dados dos alunos, que fica sendo transportado pelas variáveis *Alunos* e *Alunos2*. Foi necessária a criação de uma variável do tipo *TItem* chamada *Matricula,* usada para a geração da matrícula dos alunos.

No programa principal, também utilizamos um arquivo .TXT chamado LOG.txt, onde será armazenada as principais informações sobre o processo como, por exemplo, as matrículas geradas e os caixas em que os alunos foram atendidos, além dos dados de análise final (*Tempo médio de atendimento*, *Tempo corrido, Quantidade de Alunos servidos).*

**2.3 Organização do Código, Decisões de Implementação e Detalhes Técnicos**

O código está dividido em três arquivos principais: *PILHA\_VETOR.h* e *Fila.h* implementam o Tipo Abstrato de Dados enquanto o arquivo *main.c* implementa o programa principal. Existe ainda mais dois arquivos para um futuro melhoramento e testes do código: *Pilha.h* e *FILA\_VETOR.h.*

O valor especificado para a constante *MAX* foi *30*, para a constante *FUNCIONAMENTO* foi *240* e para a constante *QCAIXAS* foi *1*, de forma a permitir diferentes testes com o programa. No caso da constante *QCAIXAS* só é permitido alterar para *‘1’* ou ‘*2’* o seu valor.

Existem tomadas de decisões simples e de fácil compreensão no código, simplesmente para direcionar qual o caminho que o programa deve tomar com base nos valores das constantes e variáveis, mas nada de grande complexidade.

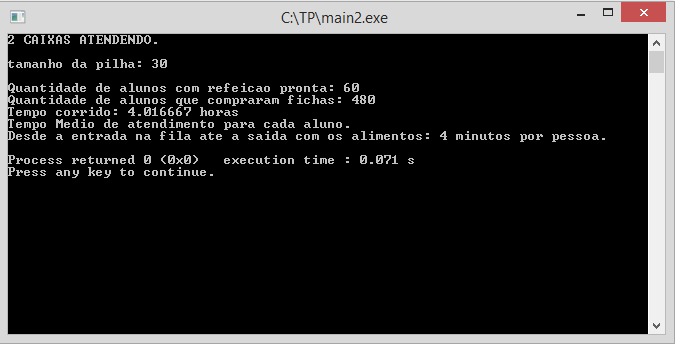
O compilador utilizado foi o GCC aplicado ao CODE::BLOCKS 16.01. Para executá-lo, use o padrão de compilação do GCC, seja no sistema operacional Windows ou Linux.

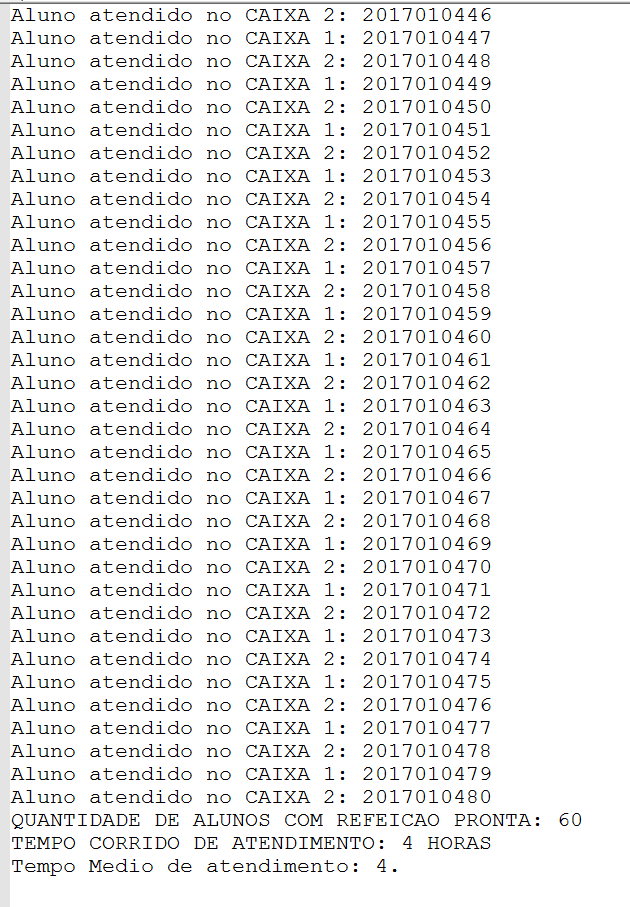
**3. Análise de Complexidade**

* **Função FFVazia():** A função faz apenas atribuições O(1).
* **Função Vazia\_fila():** A função faz, também, apenas atribuições. O(1).
* **Função Enfileira():** A função faz atribuições a fim de acrescentar mais um item a fila. O(1);
* **Função Desenfileira():** A função trabalha com O(1), para retirar o item, apenas atribuições são feitas.
* **Função FPVaziaV():** Da mesma forma que FFVazia(), essa função também apenas atribui. O(1).
* **Função VaziaV():**A função faz, também, apenas atribuições. O(1).
* **Função EmpilhaV():**A função faz atribuições a fim de acrescentar mais um item a fila. O(1);
* **Função DesempilhaV():**A função trabalha com O(1), para retirar o item, apenas atribuições são feitas.
* **Função TamanhoV():** A função só retorna o valor do tamanho da estrutura utilizada. O(1)
* **Função Principal():** A função principal é onde todas as informações e ações são trabalhadas e executadas. Existe um loop de O(n). Mas também existe a divisão do processo, mais parecida com O(n log n).

*Opinião: Acredito que o meu programa, com base nas funções apresentadas acima, se classifica em O(n log n). Mas estou na duvida se não é apenas O(n).*

**4. Testes**

 Vários testes foram realizados com o programa de forma a verificar o seu funcionamento. Os testes foram realizados em um Intel Core i7, com 8 Gb de memória. A figura abaixo mostra a saída em tela e de um .TXT, respectivamente, de uma execução típica.

****

**4. Conclusão**

A implementação do trabalho transcorreu sem maiores problemas e os resultados ficaram dentro do imaginado, apesar da incerteza sobre a validação da resposta.

A principal dificuldade encontrada foi saber exatamente o que era esperado na saída do programa. Imaginar possíveis resultados e não perder o foco do proposto, além da dúvida sobre as informações passadas, dificultou um pouco a solução do problema. Espero que esteja, pelo menos, no caminho certo e imagino que seja possível melhorias e aperfeiçoamentos significativos nesse programa.

**Referências:**

[1] Ziviani, N., Projeto de Algoritmos com Implementações em Pascal e C, 3 Edição, Editora Cengage Learning, 2011.

[2] Celes, W e Cerqueira R. e Rangel J. L., Introdução a Estruturas de Dados, 6 Edição, Editora Campos, 2004.

**Anexos:**

* *PILHA\_VETOR.h*
* *Fila.h*
* *main.c*
* *Pilha.h*
* *FILA\_VETOR.h.*
* *LOG.TXT*