

TP1: Simulador Bancário

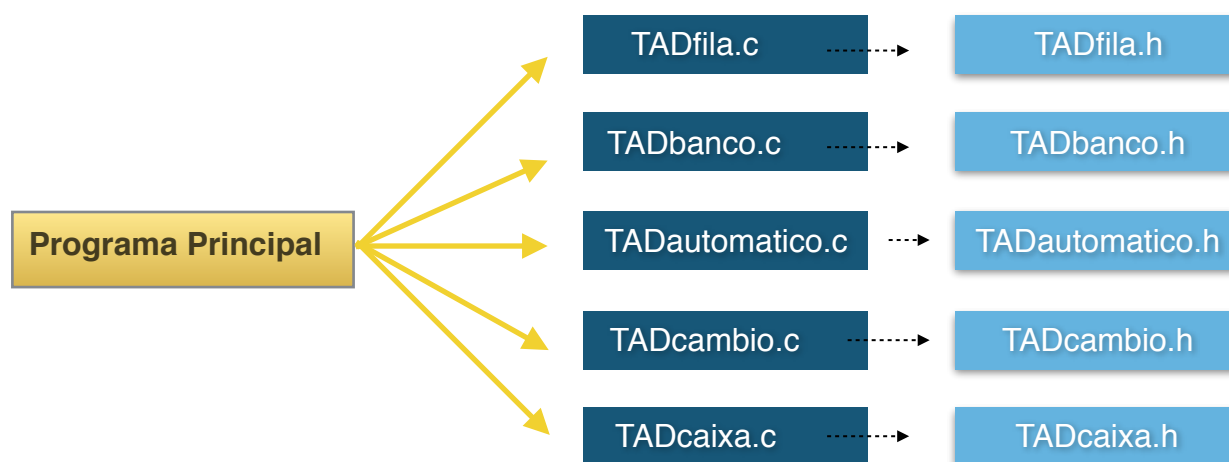
1. Introdução

O trabalho prático propõe a criação de um simulador bancário. Este simulador tem por objetivo gerar um relatório com diversas informações a respeito do fluxo de pessoas e do fluxo de dinheiro no banco. Dentre as informações estão o tempo de espera de cada cliente do banco, as operações realizadas pelos clientes e a variação do dinheiro em caixa. O relatório deve auxiliar o gerente da agência bancária a obter melhorias no atendimento do banco.

Para a realização da tarefa, as estruturas de dados são organizadas em diferentes Tipos Abstratos de Dados (TADs). Esse modelo de organização ajuda na clareza do código e permite ao programador que vê o código um maior entendimento do programa. São objetivos do trabalho ainda o desenvolvimento da prática de programação e a aplicação da análise de complexidade em um contexto real.

2. Implementação

A implementação do trabalho foi feita por meio do programa principal e de cinco Tipos Abstratos de Dados, conforme o esquema.



Cada um dos Tipos Abstratos de Dados é composto por funções específicas que juntas permitem o total funcionamento do programa principal. Os TADs e suas funções serão especificados a seguir.

2.1 Programa Principal

O programa principal contém somente a função “main”. A função principal recebe dois argumentos pelo terminal. Caso a função não receba os dois argumentos, o programa é finalizado. Os argumentos são o arquivo de entrada e o arquivo de saída dos dados, respectivamente. A função utiliza um contador que faz a contagem do número de linhas do arquivo de entrada. Esse contador é importante para que seja alocada dinamicamente a quantidade necessária de memória para dois vetores que são criados posteriormente.

Além disso, a função cria quatro filas, que serão utilizadas pelo programa. Cada uma dessas filas representa um dos tipos de caixa da agência bancária. Outra declaração importante é a de dois vetores: “cliente” e “imprime”. Ambos vetores são do tipo “dados”. O primeiro tem a finalidade de salvar o conteúdo do arquivo de entrada. O segundo vetor, por sua vez, aloca os clientes e seus atributos após o cliente ter deixado o banco. Esse último vetor será utilizado na impressão dos dados no arquivo de saída.

Pode-se dizer que o restante do programa principal é dividido em cinco partes, onde cada uma delas é representada pela chamada de uma função. A primeira função a ser chamada é a “coloca_fila_correta”, que aloca os cliente na fila correta, de acordo com sua operação. A segunda função é a “caixa_automatico”, que utiliza a fila do caixa automático e um vetor que representa os caixas para realizar as operações com os clientes e determinar a ordem de saída e o tempo de espera dos mesmos. A terceira função a ser chamada é a “caixa_cambio”, que realiza operações com os clientes da fila do câmbio e determina o tempo de espera e a hora de saída dos clientes. Assim como a função “caixa_automatico” e a função “caixa_cambio”, a função “caixa_manual”, quarta a ser chamada, executa operações com os clientes caixa manual e determina o tempo de espera e de saída dos clientes.

Após essas operações, a função “imprime_saida” é chamada pelo programa e realiza a impressão dos dados no arquivo de saída passado como argumento na execução do programa. Por fim, a memória alocada dinamicamente é liberada.

2.2 TAD Fila

O Tipo Abstrato de dados Fila tem por objetivo implementar funções que permitem a criação e o uso de filas no programa principal. A implementação das filas é feita com uso de apontadores, com o objetivo de não criar restrições quanto ao tamanho da fila, o qual depende da entrada e, portanto, não é constante. Além disso, o TAD fila cria alguns tipos de dados. O TAD fila cria os seguintes tipos de dados:

Tipo dados: É uma estrutura composta de 9 valores. Esse tipo é usado ao longo do programa principal e das listas, já que contém em sua estrutura os atributos de um cliente do banco.

A composição da estrutura do Tipo dados é: hora de chegada do cliente (long int), nome do cliente (vetor de char com 50 posições), operação realizada (vetor de char com 10 posições), valor movimentado em BRL (float), valor movimentado em USD (float), tempo de operação(int), tempo de fila(int), hora de saída do banco (long int) e tempo de espera total (int).

Tipo TCelula: É a estrutura de cada célula da fila. Nela estão contidos um usuário, do tipo dados, e um ponteiro para a próxima posição da fila, que neste caso é a posição de trás.

Tipo TFila: É o tipo de dados que define a fila propriamente dita. Nela são encontrados dois apontadores, que apontam para o início e para o fim da fila. Esses apontadores apontam para uma Célula do tipo TCelula. Foi implementada na estrutura ainda um inteiro, que tem por objetivo registrar o tamanho da fila.

No TAD Fila foram implementadas as seguintes funções:

void FFVazia(TFila *Fila)

A função recebe uma fila por referência e aloca memória para a célula cabeça da fila. Dessa forma, os ponteiros que mostram o início e o fim da fila passam a apontar para essa célula, dando início a uma fila vazia.

int Vazia (TFila Fila)

A função verifica se a fila recebido como argumento está ou não vazia. Caso esteja vazia, a função retorna 1.

void Enfileira (dados cliente_novo, TFila *Fila)

A função Enfileira coloca o cliente passado por argumento no fim da fila. Além disso, a função acrescenta 1 ao tamanho da fila.

dados Desenfileira (TFila *Fila)

A função Desenfileira retira o primeiro cliente da fila e retorna este cliente(tipo dados). Assim, o ponteiro que indica o início da fila passa a indicar o próximo da fila como primeiro. Além disso, a memória ocupada pelo cliente que saiu da fila, alocada dinamicamente, é liberada.

void libera (TFila *Fila)

A função tem por objetivo retirar todos da fila e liberar toda a memória alocada pela fila.

2.3 TAD Banco

O Tipo Abstrato de Dados Banco, por sua vez, implementa apenas três funções, as quais serão utilizadas no programa principal. As funções implementadas neste TAD são auxiliares e foram criadas com o objetivo de deixar o código mais claro e objetivo. As três funções são:

dados proximo_qsaí(dados* vetor_clientes, int n)

A função é utilizada para verificar em um vetor de cliente qual é o cliente que tem menor hora de saída. O objetivo dessa função é verificar qual será o próximo caixa a ser liberado, ou seja, qual será a posição do caixa a ser ocupado pelo próximo cliente da fila. Para isso, a função assume que o menor valor está na posição zero e percorre todo o vetor fazendo comparações para verificar se há alguma posição com valor menor.

void coloca_fila_correta (dados* cliente, FILE* ent, TFila* fila_aut, TFila* fila_cambio, TFila* fila_caixa, int linhas)

A função tem por objetivo inserir o cliente na fila correta, de acordo com a operação realizada por ele. São passados como argumento o vetor onde serão armazenados os clientes, o arquivo que contém a entrada do programa, as três filas utilizadas pelo programa principal (fila do caixa automático, fila do câmbio e fila do caixa manual) e o número de linhas do arquivo de entrada, o qual coincide com o número de clientes que serão lidos. Esta função é basicamente um loop, que é executado a cada linha lida do arquivo de entrada.

Primeiramente, esta função cria uma matriz do tipo char e aloca cada um dos atributos do cliente em uma das suas seis linhas. Para isso, a função utiliza o comando "fgets" para ler toda a linha do arquivo e armazená-la em uma string. Como os atributos estão separados por vírgulas, a função verifica cada "char" da string e o compara com a

vírgula. Quando a vírgula é encontrada, a função começa a completar a próxima linha da matriz, utilizando a verificação de cada char de maneira análoga.

Após completar toda a matriz com os seis atributos, cada linha da matriz é atribuída ao cliente i do vetor de clientes, de maneira que cada atributo é atribuído a um valor da estrutura “dados” relativa ao cliente i . Depois dessa etapa, é verificada então a operação que será feita pelo cliente i , com a finalidade de enfileirar o cliente na fila correta. Verificada a operação, o cliente é enfileirado na fila correspondente à operação e, caso haja mais linhas a serem lidas no arquivo de entrada, todo processo é repetido.

É importante ressaltar que cabe a esta função também fazer algumas validações da entrada. Caso ocorra entradas inválidas, o programa é abortado instantaneamente. São feitas validações para verificar se cada linha contém o número correto de campos, para verificar se a operação realizada pelo cliente é válida e para certificar que as operações de saque e depósito tenham valor de entrada válido.

void imprime_saida (dados* imprime, FILE* saida, int tam1, int tam2, int tam3, int* k)

A função `imprime_saida` tem por objetivo realizar a saída dos dados. A função recebe por argumento o arquivo de saída dos dados, o vetor que contém os clientes a serem impressos e a quantidade de clientes que utilizou cada tipo de caixa. Para realizar a impressão, a função verifica no vetor de clientes quem é aquele que contém menor hora de saída, com a finalidade de imprimir os clientes na ordem em que deixam o banco. Essa ordem de saída é utilizada para aproximar a situação proposta pelo Trabalho Prático de uma situação real, uma vez que os valores em caixa do banco são alterados somente no momento em que o cliente termina sua operação. Para fazer a verificação da menor hora de saída, é utilizada a função “`proximo_qsai`”. Verificado o cliente, a função atualiza os valores do banco, em BRL e em USD.

Após realizar essa etapa, é verificado se o tempo de espera e o valor da operação realizada pelo cliente correspondem ao mínimo ou ao máximo, para que seja gerado um resumo geral das transações bancárias no fim do arquivo de saída. Por fim, é realizada a impressão da saída de dados no arquivo de saída.

2.4 TAD Automático

O Tipo Abstrato de Dados Automático é composto por apenas uma função:

void caixa_automatico(int quant_aut, int quant_bio, TFila* fila_aut, TFila* fila_bio, int tam_fila, dados* aut, dados* bio, dados* imprime, int *k)

O objetivo dessa função é também a finalidade do TAD. Ambos tem por finalidade realizar as operações com os clientes que estão na fila dos caixas automáticos. Os caixas automáticos, em especial, têm a divisão em dois tipos de caixa: com leitor biométrico e sem leitor biométrico. A restrição está nas operações de saque com valores maiores que 300 reais, uma vez que estas podem ser realizadas apenas nos caixas com leitor biométrico. Para que ocorresse essa diferenciação, é verificado a cada cliente que passa a usar um dos caixas se esse cliente pode utilizar o caixa livre ou não.

Os caixas automáticos são representados em dois vetores de clientes, do tipo “dados”. Um vetor representa os caixas normais e o outro representa os caixas com leitor biométrico.

Os primeiros, no máximo 25 clientes, são alocados nos caixas, independente da hora de chegada. Porém, caso haja mais de 10 clientes entre os 25 primeiro que necessitem utilizar o caixa biométrico, os clientes excedentes no uso do caixa com leitor biométrico serão alocados em uma fila auxiliar, com o objetivo de organizar a ordem dos clientes que tiveram que deixar que o cliente de trás passasse à frente.

Dessa forma, é iniciado um novo loop, que só deixa de ser executado quando a fila fica vazia. A primeira verificação da função a partir do vigésimo sexto cliente é se o próximo caixa a ficar livre tem leitor biométrico ou não. Caso tenha leitor biométrico, o cliente a ocupar o caixa livre será o o primeiro da fila auxiliar criada, a não ser que esta fila esteja vazia. Neste caso, quem utilizará o caixa será o primeiro da fila normal.

Caso o caixa não tenha leitor biométrico, será feita a análise do primeiro cliente da fila normal. Se ele puder utilizar o caixa normal, ele ocupa o caixa livre. Porém, caso o cliente não possa utilizar o caixa normal, ele será enfileirado na fila auxiliar e o próximo da fila normal passará pela mesma análise.

Para verificar qual será o próximo caixa a ficar livre, será usada a função “proximo_qsai”. Feita a verificação, o próximo cliente ocupa o caixa recebe seu tempo de fila, sua hora de saída e seu tempo de espera total. O cliente que deixa o caixa, por sua vez, é armazenado no vetor de clientes “imprime”, que é utilizado para imprimir os resultados no arquivo de saída. Feito isso, o loop se repete até que todos clientes sejam alocados no vetor “imprime”.

2.5 TAD Câmbio

O Tipo Abstrato de Dados Câmbio é simples. Também composto por apenas uma função, ele tem objetivo similar ao TAD Automático. O objetivo deste TAD é realizar operações com os clientes que utilizam o Câmbio. A função contida no TAD Câmbio é:

void caixa_cambio(int quant_cambio, Tfila* fila_cambio, int tam_fila, dados* cambio, dados* imprime, int *k)

Essa função representa o caixa de câmbio como um vetor do tipo “dados”, de apenas uma posição. Como não há restrições para o uso do caixa Câmbio, os clientes são alocados no caixa na ordem em que estão na fila. Para que seja computado o tempo de fila dos clientes e, assim, a hora de saída dos mesmos, é feita a verificação do tempo de saída do cliente que deixa o caixa e da hora de chegada do primeiro da fila.

Como só existe um caixa de câmbio, não é necessário verificar qual será o próximo caixa a ser liberado, o que facilita a implementação desta função. Ao sair do caixa, o cliente é incluído no vetor “imprime”, que será utilizado ao fim do programa, para impressão dos dados no arquivo de saída.

O processo realizado na função é repetido até que a fila do câmbio fique vazia.

2.6 TAD Caixa

O Tipo Abstrato de Dados Caixa, assim como o TAD Câmbio, tem implementação mais simples que o o TAD Automático, por não conter restrições. A única função que compõe o TAD é:

```
void caixa_manual(int quant_caixa, Tfila* fila_caixa, int tam_fila, dados*
caixa, dados* imprime, int *k)
```

A função tem implementação parecida com a “caixa_cambio”. A principal diferença entre as funções é a necessidade de um loop extra para verificar qual caixa será o próximo a ficar livre, uma vez que existem três caixas manuais.

Os caixas manuais também são representados como um vetor do tipo "dados", mas neste caso o vetor contém três posições. Os primeiros 3 clientes da fila são alocados no vetor e, a partir daí, a cada vez que algum cliente deixa o banco é feito o calculo do tempo de fila do cliente que ocupa o caixa. A verificação de qual será o próximo caixa a ser liberado utiliza a função “proximo_qsai”.

A função repete suas operações até que todos clientes sejam alocados no vetor “imprime”. Essa alocação é feita na medida em que os clientes deixam o banco.

3. Casos de teste

Foram executados quatro testes para verificação do correto funcionamento do programa. Os testes foram divididos em teste 0, teste 1, teste 2 e teste 3.

3.1 Teste 0

O teste 0 foi realizado com o exemplo presente na especificação do Trabalho Prático. Exclusivamente neste teste, o número de caixas foi reduzido. Foram utilizados apenas 2 caixas automáticos, um com leitor biométrico e o outro sem. O número de caixas manuais foi reduzido para 1 e o número de caixas de câmbio se manteve.



```
teste0.txt
1 1444771699,André Alves,SAQUE,-300.00,,200
2 1444771730,Junia Jansen,DEPOSITO,300.37,,550
3 1444771735,Ricardo Romário,CÂMBIO,500.00,-125.44,150
4 1444771801,Sandreí Sousa,SAQUE,-700.00,,150
5 1444771805,Giovana Galvão,SAQUE,-100.00,,60
6 1444771811,Fabio Feliciano,CAIXA,-1500.00,,480
7 1444771831,Larissa Lousano,CAIXA,500.00,,480
```



```
saida0.txt
1 |1444771865,Giovana Galvão,SAQUE,-100.00,0.00,60
2 1444771885,Ricardo Romário,CÂMBIO,400.00,-125.44,150
3 1444771899,André Alves,SAQUE,100.00,-125.44,200
4 1444771951,Sandreí Sousa,SAQUE,-600.00,-125.44,150
5 1444772280,Junia Jansen,DEPOSITO,-299.63,-125.44,550
6 1444772291,Fabio Feliciano,CAIXA,-1799.63,-125.44,480
7 1444772311,Larissa Lousano,CAIXA,-1299.63,-125.44,480
8 VARIACAO DE BRL: -1799.63 a 400.00
9 VARIACAO DE USD: -125.44 a 0.00
10 VARIACAO DE TEMPO: 60 a 550
11
```

3.2 Teste 1

O teste 1 foi realizado com um número grande de clientes, com o objetivo de verificar o funcionamento da função “caixa_automatico” quando o número de clientes para utilizar os caixas automáticos é maior que 25.


```
teste1.txt
saida0.txt
teste1.txt
1 1444771699,André Alves,SAQUE,-300.00,,200
2 1444771730,Junia Jansen,DEPOSITO,300.37,,550
3 1444771735,Ricardo Romário,CÂMBIO,500.00,-125.44,150
4 1444771801,Sandrei Sousa,SAQUE,-700.00,,150
5 1444771805,Giovana Galvão,SAQUE,-100.00,,60
6 1444771811,Fabio Feliciano,CAIXA,-1500.00,,480
7 1444771831,Larissa Lousano,CAIXA,500.00,,480
8 1444771840,Joao,SAQUE,-600.00,,210
9 1444771950,Renato,DEPOSITO,437.00,,300
10 1444772690,Camila,SAQUE,-250.00,,150
11 1444772692,Roberto,SAQUE,-700.00,,220
12 1444772700,Karolina,SAQUE,-100.00,,60
13 1444772801,Luciana,CAIXA,-1500.00,,400
14 1444772804,José,CAIXA,800.00,,500
15 1444772235,Ronaldo Gáúcho,SAQUE,-300.00,,150
16 1444772279,Vitor Mendes,DEPOSITO,720,,550
17 1444772236,Felipe Souza,SAQUE,-500.00,,320
18 1444772390,Amilton Junior,SAQUE,-700.00,,125
19 1444772406,Jose Nunes,SAQUE,-100.00,,75
20 1444772500,Ricardo Guimaraes,DEPOSITO,800.00,,390
21 1444772540,Josefina Carvalho,DEPOSITO,500.00,,480
22 1444772547,Breno Vieira,SAQUE,-300.00,,100
23 1444772600,Douglas Sena,DEPOSITO,240.50,,550
24 1444772630,Julia Costa,CÂMBIO,1000.00,-240.00,130
25 1444772690,Alexandre kalil,SAQUE,-700.00,,350
26 1444772700,Leonardo Silva,SAQUE,-370.00,,70
27 1444772733,Levir Culpí,DEPOSITO,900.00,,380
28 1444772769,Giovana Amaral,CAIXA,500.00,,610
29 1444772801,Tales de Castro,SAQUE,-400.00,,170
30 1444772885,Wenndel César,DEPOSITO,850.40,,295
31 1444772900,Jonh Diggle,CÂMBIO,2000.00,-490.50,150
32 1444772984,Tamirez Prato,SAQUE,-770.00,,180
33 1444773000,Galvão Bueno,SAQUE,-400.00,,90
34 1444773023,Arthur Soares,DEPOSITO,500.00,,300
35 1444773100,Barbara Viana,SAQUE,-600.00,,280
```

```
teste1.txt
saida1.txt
1 1444771865,Giovana Galvão,SAQUE,-100.00,0.00,60
2 1444771885,Ricardo Romário,CÂMBIO,400.00,-125.44,150
3 1444771899,André Alves,SAQUE,100.00,-125.44,200
4 1444771951,Sandrei Sousa,SAQUE,-600.00,-125.44,150
5 1444772050,Joao,SAQUE,-1200.00,-125.44,210
6 1444772250,Renato,DEPOSITO,-763.00,-125.44,300
7 1444772280,Junia Jansen,DEPOSITO,-462.63,-125.44,550
8 1444772291,Fabio Feliciano,CAIXA,-1962.63,-125.44,480
9 1444772311,Larissa Lousano,CAIXA,-1462.63,-125.44,480
10 1444772385,Ronaldo Gáúcho,SAQUE,-1762.63,-125.44,150
11 1444772481,Jose Nunes,SAQUE,-1862.63,-125.44,75
12 1444772515,Amilton Junior,SAQUE,-2562.63,-125.44,125
13 1444772556,Felipe Souza,SAQUE,-3862.63,-125.44,320
14 1444772647,Breno Vieira,SAQUE,-3362.63,-125.44,100
15 1444772760,Karolina,SAQUE,-3462.63,-125.44,60
16 1444772760,Julia Costa,CÂMBIO,-2462.63,-365.44,130
17 1444772770,Leonardo Silva,SAQUE,-2832.63,-365.44,70
18 1444772791,Jose,CAIXA,-2032.63,-365.44,587
19 1444772829,Vitor Mendes,DEPOSITO,-1312.63,-365.44,550
20 1444772840,Camila,SAQUE,-1562.63,-365.44,150
21 1444772890,Ricardo Guimaraes,DEPOSITO,-762.63,-365.44,390
22 1444772912,Roberto,SAQUE,-1462.63,-365.44,220
23 1444772971,Tales de Castro,SAQUE,-1862.63,-365.44,170
24 1444773020,Josefina Carvalho,DEPOSITO,-1362.63,-365.44,480
25 1444773040,Alexandre kalil,SAQUE,-2062.63,-365.44,350
26 1444773050,Jonh Diggle,CÂMBIO,-62.63,-855.94,150
27 1444773090,Galvão Bueno,SAQUE,-462.63,-855.94,90
28 1444773113,Levir Culpí,DEPOSITO,437.37,-855.94,380
29 1444773150,Douglas Sena,DEPOSITO,677.87,-855.94,550
30 1444773164,Tamirez Prato,SAQUE,-92.13,-855.94,180
31 1444773180,Wenndel César,DEPOSITO,758.27,-855.94,295
32 1444773201,Luciana,CAIXA,-741.73,-855.94,400
33 1444773323,Arthur Soares,DEPOSITO,-241.73,-855.94,300
34 1444773379,Giovana Amaral,CAIXA,258.27,-855.94,610
35 1444773380,Barbara Viana,SAQUE,-341.73,-855.94,280
VARIACAO DE BRL: -3462.63 a 758.27
VARIACAO DE USD: -855.94 a 0.00
VARIACAO DE TEMPO: 60 a 610
```

3.3 Teste 2

O teste 2 tem por objetivo verificar o funcionamento do programa quando o número de pessoas que executam a operação Câmbio é maior que 1.

```
teste2.txt
saida2.txt
1 1444772540,Josefina Carvalho,DEPOSITO,500.00,,480
2 1444772547,Breno Vieira,SAQUE,-300.00,,100
3 1444772600,Douglas Sena,DEPOSITO,240.50,,550
4 1444772630,Julia Costa,CÂMBIO,1000.00,-240.00,130
5 1444772690,Alexandre kalil,SAQUE,-700.00,,350
6 1444772700,Leonardo Silva,CÂMBIO,-370.00,100.00,70
7 1444772733,Levir Culpí,DEPOSITO,900.00,,380
8 1444772769,Giovana Amaral,CAIXA,500.00,,610
9 1444772801,Tales de Castro,SAQUE,-400.00,,170
10 1444772885,Wenndel César,DEPOSITO,850.40,,295
11 1444772900,Jonh Diggle,CÂMBIO,2000.00,-490.50,150
```

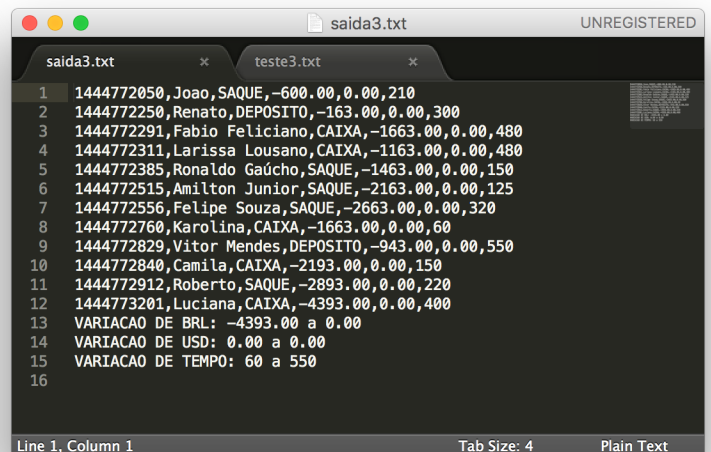
```
teste2.txt
saida2.txt
1 1444772647,Breno Vieira,SAQUE,-300.00,0.00,100
2 1444772760,Julia Costa,CÂMBIO,700.00,-240.00,130
3 1444772830,Leonardo Silva,CÂMBIO,330.00,-140.00,130
4 1444772971,Tales de Castro,SAQUE,-70.00,-140.00,170
5 1444773020,Josefina Carvalho,DEPOSITO,430.00,-140.00,480
6 1444773040,Alexandre kalil,SAQUE,-270.00,-140.00,350
7 1444773050,Jonh Diggle,CÂMBIO,1730.00,-630.50,150
8 1444773113,Levir Culpí,DEPOSITO,2630.00,-630.50,380
9 1444773150,Douglas Sena,DEPOSITO,2870.50,-630.50,550
10 1444773180,Wenndel César,DEPOSITO,3720.90,-630.50,295
11 1444773379,Giovana Amaral,CAIXA,4220.90,-630.50,610
12 VARIACAO DE BRL: -300.00 a 4220.90
13 VARIACAO DE USD: -630.50 a 0.00
14 VARIACAO DE TEMPO: 100 a 610
```

3.4 Teste 3

O último teste foi realizado para verificar o funcionamento do programa quando o número de pessoas na fila do caixa manual é maior que 3.



```
1 1444771811,Fabio Feliciano,CAIXA,-1500.00,,480
2 1444771831,Larissa Lousano,CAIXA,500.00,,480
3 1444771840,Joao,SAQUE,-600.00,,210
4 1444771950,Renato,DEPOSITO,437.00,,300
5 1444772690,Camila,CAIXA,-1250.00,,150
6 1444772692,Roberto,SAQUE,-700.00,,220
7 1444772700,Karolina,CAIXA,1000.00,,60
8 1444772801,Luciana,CAIXA,-1500.00,,400
9 1444772235,Ronaldo Gaúcho,SAQUE,-300.00,,150
10 1444772279,Vitor Mendes,DEPOSITO,720,,550
11 1444772236,Felipe Souza,SAQUE,-500.00,,320
12 1444772390,Amilton Junior,SAQUE,-700.00,,125
```



```
1 1444772050,Joao,SAQUE,-600.00,0.00,210
2 1444772250,Renato,DEPOSITO,-163.00,0.00,300
3 1444772291,Fabio Feliciano,CAIXA,-1663.00,0.00,480
4 1444772311,Larissa Lousano,CAIXA,-1163.00,0.00,480
5 1444772385,Ronaldo Gaúcho,SAQUE,-1463.00,0.00,150
6 1444772515,Amilton Junior,SAQUE,-2163.00,0.00,125
7 1444772556,Felipe Souza,SAQUE,-2663.00,0.00,320
8 1444772760,Karolina,CAIXA,-1663.00,0.00,60
9 1444772829,Vitor Mendes,DEPOSITO,-943.00,0.00,550
10 1444772840,Camila,CAIXA,-2193.00,0.00,150
11 1444772912,Roberto,SAQUE,-2893.00,0.00,220
12 1444773201,Luciana,CAIXA,-4393.00,0.00,400
13 VARIACAO DE BRL: -4393.00 a 0.00
14 VARIACAO DE USD: 0.00 a 0.00
15 VARIACAO DE TEMPO: 60 a 550
16
```

4. Estudo de complexidade

A análise de complexidade será feita a cada função executada pelo programa e, por fim, será analisado o programa principal. Toda o estudo será feito utilizando a notação O .

Função FFVazia: A função faz uma alocação dinâmica e 3 atribuições. Portanto, a função é $O(1)$.

Função Vazia: A função executa apenas uma comparação. Logo, a função é $O(1)$.

Função Enfileira: A função Enfileira faz uma alocação dinâmica e 4 atribuições. Portanto, ela é $O(1)$.

Função Desenfileira: A função Desenfileira realiza 4 atribuições apenas e libera uma memória alocada. Portanto, a função é da ordem $O(1)$.

Função libera: A função libera realiza um loop, que é executado n vezes, de acordo com o número de elementos na fila. Logo, a função é da ordem $O(n)$.

Função proximo_qsai: A função realiza um loop, que é executado n vezes. A quantidade de execuções do loop depende do numero de elementos no vetor. No pior caso, a função realiza uma comparação e uma atribuição a cada loop. Portanto, a ordem de complexidade da função é $O(n)$.

Função coloca_fila_correta: A função é composta por um loop do tipo "while", que abrange todas as outras operações. Esse loop é executado n vezes, de acordo com o número de linhas do arquivo de entrada. As operações dentro do loop são, em maioria, $O(1)$. Há dois "for" aninhados dentro do loop, porém são eles são executados um numero fixo de vezes e, portanto, são $O(1)$. Há ainda dentro do loop outro "while", que é executado até que a virgula seja encontrada, ou seja, n vezes. Logo, esse ultimo "while" é $O(n)$, ja que dentro dele existem apenas uma comparação e 4 atribuições. Existem também mais dois "for", mas ambos com numero fixo de execucoes. As demais operações são comparações. Portanto, a função é da ordem $O(n^2)$.

Função imprime_saida: A função é composta por um while que abrange praticamente todas as operações. Esse loop é executado n vezes, de acordo com o tamanho da entrada. Dentro desse loop existe ainda um "for", o qual contém uma comparação e 3 atribuições. Esse segundo loop aninhado é executado n vezes, também de acordo com o tamanho da entrada. Logo, o "for" é $O(n)$. As demais operações dentro do "while" são $O(1)$: atribuições, comparações e uma impressão. Portanto, essa função tem ordem de complexidade $O(n^2)$.

Função caixa_automatico: A função contém dois "while". O primeiro deles é executado até n vezes e contém 4 "if's". Dentro de cada if, existem apenas operações atribuição, com exceção do ultimo. Este contém a função Enfileira, que é $O(1)$. Logo, o primeiro while é $O(n)$. O segundo while, por sua vez, contém um "if" e um "else". As operações o "if" dependem de uma comparação, que é da ordem $O(n)$, já que ela utiliza a função "proximo_qsaí". Existe um "for" aninhado ao "if". O "for" é executado n vezes, de acordo com o tamanho da fila do caixa automático. Dentro deste "for" existe apenas um "if", que contém operações de atribuição, comparações e duas chamadas para a função "proximo_qsaí", que é $O(n)$. Portanto, o "for" é $O(n^2)$ e a ordem de complexidade do "if" é $O(n^3)$. Por fim, a complexidade da função é $O(n^4)$.

Função caixa_cambio: A função contém dois "while", sendo o primeiro executado n vezes, de acordo com a chamada da função e o segundo executado n vezes, de acordo com o tamanho da fila do câmbio. O primeiro loop contém apenas operações $O(1)$. Já o segundo loop tem um "for" aninhado. Este "for" é executado n vezes, de acordo com o tamanho da fila. Dentro deste "for" existe um "if", o qual contém apenas atribuições e uma comparação que chama a função "proximo_qsaí", da ordem $O(n)$. Portanto, o segundo loop é $O(n^3)$ e a função tem ordem de complexidade $O(n^3)$.

Função caixa_manual: Assim como a função "caixa_cambio", a função contém dois "while", sendo o primeiro executado n vezes, de acordo com a chamada da função e o segundo executado n vezes, de acordo com o tamanho da fila do câmbio. O primeiro loop contém apenas operações $O(1)$. Já o segundo loop tem um "for" aninhado. Este "for" é executado n vezes, de acordo com o tamanho da fila. Dentro deste "for" existe um "if", o qual contém apenas atribuições e uma comparação que chama a função "proximo_qsaí", da ordem $O(n)$. Portanto, o segundo loop é $O(n^3)$ e a função tem ordem de complexidade $O(n^3)$.

Função Principal: A função principal executa algumas atribuições. Além disso, a função contém um "while", que é executado n vezes, de acordo com o número de linhas da entrada. Logo, esse loop é $O(n)$. Além disso, a função chama as funções "coloca_fila_correta"- $O(n^2)$, "caixa_automatico"- $O(n^4)$, "caixa_cambio"- $O(n^3)$, "caixa_manual"- $O(n^3)$ e "imprime_saida"- $O(n^2)$. Portanto, o programa principal tem ordem de complexidade $O(n^4)$.

6. Conclusão

A realização do Simulador Bancário é uma atividade complexa e exige bastante atenção aos detalhes. Por ser um trabalho longo, a importância da organização do código foi muito evidente ao longo da execução da tarefa. Outra inferência da realização do Trabalho Prático 1 foi a a verificação das vantagens trazidas pelo uso de Tipos Abstratos de Dados. Além da melhor organização do código, a implementação de TADs facilita na solução de eventuais problemas durante a execução do código e também permite que alterações no código sejam feitas facilmente.

Além da importância do uso de TADs, a tarefa serviu como uma excelente pratica de programação, uma vez que exigiu um amplo conhecimento das funções e das formas implementação de diversos comandos da linguagem C. Por fim, a execução do Trabalho Prático 1 foi de suma importância para o entendimento do conteúdo abordado no curso de AEDS II.

7. Referências

- [1] Ziviani, N., Projeto de Algoritmos com Implementações em Pascal e C, 2a Edição, Editora Thomson, 2004.
- [2] Stack Overflow - <http://stackoverflow.com>
- [3] C Plus Plus - <http://www.cplusplus.com>

8. Anexos

Listagem dos Programas:

- Main.c
- TADfila.c e TADfila.h
- TADbanco.c e TADbanco.h
- TADautomatico.c e TADautomatico.h
- TADcambio.c e TADcambio.h
- TADcaixa.c e TADcaixa.h