

Trabalho: TP1

Nome: Vitor Cláudio Chaves de Aguiar

Matrícula: 2015053292

Curso: Sistemas de Informação

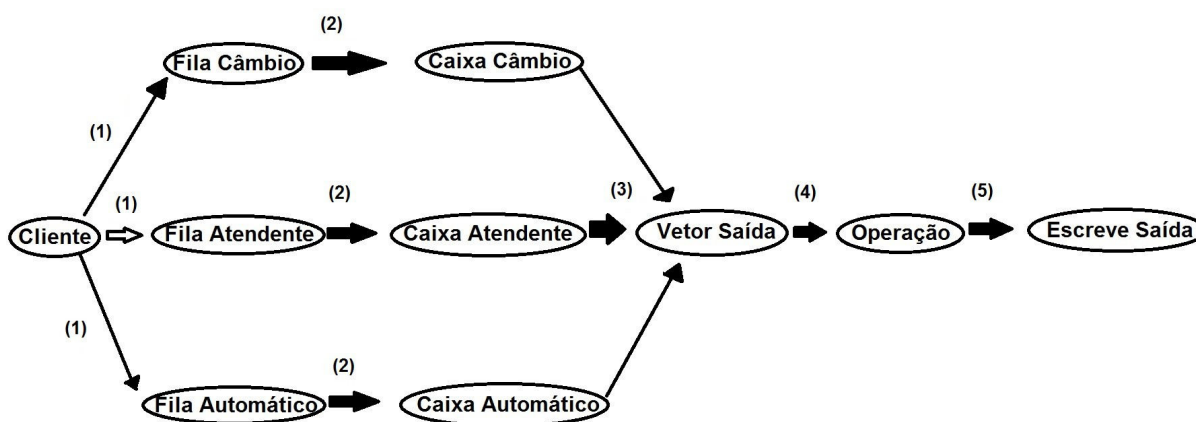
1. Introdução

A necessidade de ir em uma agência bancária ainda existe nos dias de hoje para pagar uma conta, depositar ou sacar dinheiro ou até mesmo trocar reais em dólar. Muitas vezes, para realizar tais operações é preciso primeiro passar por uma fila para ser atendido ou para utilizar de um caixa. Devido a isso, é interessante para um gerente de uma agência observar as operações que estão sendo realizadas e o dinheiro da agência que está sendo movimentado, como também o tempo de fila de cada cliente.

O objetivo desse trabalho é mostrar um relatório que permita o gerente ver essas movimentações e estatísticas para que, com isso, possa tomar decisões se for necessário com a finalidade de melhorar o atendimento e para programar melhor a entrega de dinheiro para a agência.

2. Implementação

Fluxograma do programa



1- Enfileira todos os clientes nas suas respectivas filas.

2- Desenfileira um a um cada cliente enviando para o primeiro caixa a ser liberado e calculando os tempos de saída e de fila.

3- Nesse momento todos os clientes já possuem um tempo de fila e de saída. Deste modo são enviados para um mesmo vetor para ser usado na saída.

4- Na ordem de saída do banco, cada cliente é selecionado, verifica a operação e faz a operação. Deste modo os saldos da agência são alterados e as variações também.

5- Depois de já possuir todas as informações, o cliente é escrito no arquivo.

Estrutura de dados

Para a implementação do trabalho foi utilizado no total 4 filas (sendo uma auxiliar). Estas foram implementadas com ponteiros e com a finalidade de simular as filas das agências bancárias onde o cliente permanece ou não por um determinado tempo.

Funções e Procedimentos

TAD Fila

void inicializaFila(TFila *Fila): recebe uma fila por parâmetro, cria uma célula e faz os apontadores da fila apontarem pra essa célula (mesmo lugar) e o Prox dessa célula pra NULL.

int Vazia(TFila Fila): recebe uma fila e verifica se os apontadores Frente e Trás estão apontando pro mesmo lugar.

void Enfileira(Cliente cliente, TFila *Fila): recebe uma fila e um cliente, cria uma nova célula e coloca a informação do cliente nela (coloca um cliente na fila).

void Desenfileira(TFila *Fila, Cliente *cliente): recebe uma fila e uma referência de um cliente, retira um cliente da fila e passa a informação desse cliente por referência no cliente recebido por parâmetro.

TAD Caixa

void inicializaCaixas(Caixa *caixa): recebe um vetor de caixas e coloca os valores necessários para definir os caixas do programa (colocando 0 ou 1 para as suas flags).

int caixaAtendenteVazio(Caixa *caixa): recebe um vetor de caixas e verifica se existe um caixa de atendente disponível.

int caixaAutomaticoVazio(Caixa *caixa): recebe um vetor de caixas e verifica se existe um caixa automático disponível.

int caixaCambioVazio(Caixa *caixa): recebe um vetor de caixas e verifica se existe um caixa de câmbio disponível.

int caixaCambioDisponivel(Caixa *caixa): recebe um vetor de caixas e verifica o primeiro caixa de câmbio que vai ser liberado e retorna o seu índice do vetor.

int caixaAtendenteDisponivel(Caixa *caixa): recebe um vetor de caixas e verifica o primeiro caixa de atendente que vai ser liberado e retorna o seu índice do vetor.

int caixaAutomaticoDisponivel(Caixa *caixa): recebe um vetor de caixas e verifica o primeiro caixa automático que vai ser liberado e retorna o seu índice do vetor.

int caixaAutomaticoDisponivelProximo(Caixa *caixa, int i): recebe um vetor de caixas e um índice que diz a partir de qual posição do vetor será analisado se possui algum caixa automático disponível. Se existir, retorna o índice desse caixa.

void liberaCaixa(Caixa *caixa, int i): recebe um vetor de caixa e um índice. Coloca o valor 0 na flag 'ocupado' do caixa com o índice recebido.

void enviarParaOCAixa(Caixa *caixa, int i): recebe um vetor de caixa e um índice. Coloca o valor 1 na flag 'ocupado' do caixa com o índice recebido.

unsigned long int menorTempoSaidaAtendente(Caixa *caixa): recebe um vetor de caixa. Verifica qual é o caixa que será liberado primeiro e retorna o valor de saída dele.

unsigned long int menorTempoSaidaAutomatico(Caixa *caixa): recebe um vetor de caixa. Verifica qual é o caixa que será liberado primeiro e retorna o valor de saída dele.

unsigned long int menorTempoSaidaAutomaticoProximo(Caixa *caixa, int i): recebe um vetor de caixa e um índice. Verifica qual caixa será liberado primeiro a partir desse índice e retorna o valor de saída dele.

void inicializaTempoComCaixa(Cliente *cliente, Caixa *caixa, int i, Cliente *saida, int *posicao): recebe um cliente, o vetor de caixa, um índice, o vetor de cliente usado na saída, posição de inserção desse cliente no vetor de saída. Calcula os valores de tempo de saída e tempo de fila. Neste caso o tempo de fila será 0 devido ao estado inicial que terá caixa disponível.

void inicializaTempoSemCaixa(Cliente *cliente, Caixa *caixa, int i, Cliente *saida, int *posicao): recebe um cliente, o vetor de caixa, um índice, o vetor de cliente usado na saída, posição de inserção desse cliente no vetor de saída. Calcula os valores de tempo de saída e tempo de fila.

TAD Arquivo

char *strtok_single (char * str, char const * delims): recebe a string e o separador. Diferente do strtok convencional, a função separa a string quando encontra o separador porém não ocorre problemas ao se deparar com dois separadores seguidos.

int leQuantidadeDeLinhas(char *caminhoDoArquivo, Cliente **cliente): recebe o caminho do arquivo e um vetor de cliente. Abre o arquivo para ler e contar quantas linhas ele possui com a finalidade de alocar memória para o vetor de clientes.

void preecheDadosDoArquivo(char *caminhoDoArquivo, Cliente *cliente): recebe o caminho do arquivo e um cliente. Abre o arquivo para ler cada linha e separar cada informação para armazenar nas estruturas corretas.

void imprimeSaida(Cliente *cliente, char *caminhoDoArquivo, int numeroDeLinhas, Agencia *agencia): recebe o vetor de clientes para saída, caminho do arquivo, numero de linhas do arquivo e a agência. Faz a operação dos clientes na ordem de saída deles. Calcula as variações de BRL, USD e tempo. Abre o arquivo para escrita e escreve todas as informações dos clientes e variações.

TAD Agencia

void inicializaAgencia(Agencia *agencia): recebe agência. Coloca os valores iniciais (= 0) para as variáveis da agência.

void fazerOperacaoSaque(Cliente *cliente, Agencia *agencia): recebe um cliente e a agência. Soma no saldo BRL da agência o valor da operação do cliente.

void fazerOperacaoDeposito(Cliente *cliente, Agencia *agencia): recebe um cliente e a agência. Soma no saldo BRL da agência o valor da operação do cliente.

void fazerOperacaoCambio(Cliente *cliente, Agencia *agencia): recebe um cliente e a agência. Soma no saldo BRL da agência e no de USD o valor da operação do cliente.

void verificarOperacao(Cliente *cliente, TFila *filaAutomatico, TFila *filaAtendente, TFila *filaCambio): recebe um cliente e as filas dos caixas. Verifica qual operação o cliente fará e enfileira na fila correta.

void desenfileiraCambio(Caixa *caixa, TFila *filaCambio, int *posicaoVetorSaida, Cliente *clientesSaida): recebe um vetor de caixa, a fila de câmbio, a posição do vetor de saída e o vetor de saída. Pega todos os clientes que estão enfileirados e inicializam os tempos de fila.

void desenfileiraAtendente(Caixa *caixa, TFila *filaAtendente, int *posicaoVetorSaida, Cliente *clientesSaida): recebe um vetor de caixa, a fila de atendente, a posição do vetor de saída e o vetor de saída. Pega todos os clientes que estão enfileirados e inicializam os tempos de fila.

void desenfileiraAutomatico(Caixa *caixa, TFila *filaAutomatico, int *posicaoVetorSaida, Cliente *clientesSaida): recebe um vetor de caixa, a fila automático, a posição do vetor de saída e o vetor de saída. Pega todos os clientes que estão enfileirados, envia para um caixa (se não tiver disponível, libera um) e inicializam os tempos de fila. Se tiver os dois tipos de caixa automático, da prioridade ao sem leitor biométrico se a operação permitir. Se liberar sem leitor biométrico e o primeiro da fila não puder realizar a operação, acha a primeira pessoa na fila que pode e passa na frente.

Programa Principal

O programa principal valida a entrada se está passando o valor correto de argumentos. Cria a quantidade necessária de Caixa, Agência, Cliente e Fila, inicializando os seus valores. Enfileira todos os clientes nas suas filas a partir de suas operações. Cria um vetor de clientes que será usado na saída. Desenfileira os clientes calculando os valores de tempo de saída e tempo de fila. Calcula os saldos de BRL, USD, as variações e escreve no arquivo a saída desejada. A saída é escrita no arquivo na ordem que os clientes saíram da agência e, contudo, o saldo da agência é calculado após a saída de cada um.

Organização do Código, Decisões de Implementação e Detalhes Técnicos

O código está dividido em 4 arquivos .h e 5 arquivos .c. Sendo que o Main.c é o programa principal. Foi utilizado do compilador gcc no Ubuntu.

Para passar uma pessoa na frente na fila de caixa automático foi utilizado uma fila auxiliar. Se busca o cliente que pode utilizar do caixa e envia ele pro caixa. Enquanto isso, todos os outros vão para a fila auxiliar.

A saída é impressa na ordem que os clientes saem do banco, ou seja, a operação de cada um é feita na hora da saída e não na entrada do caixa. O programa valida o arquivo de entrada de diversas maneiras apresentadas nos casos de teste.

3. Análise de complexidade

TAD Fila

void inicializaFila(TFila *Fila): executa apenas comandos $O(1)$ sem nenhum loop. Portanto temos a complexidade **$O(1)$** .

int Vazia(TFila Fila): executa apenas comandos $O(1)$ sem nenhum loop. Portanto temos a complexidade **$O(1)$** .

void Enfileira(Cliente cliente, TFila *Fila): executa apenas comandos $O(1)$ sem nenhum loop. Portanto temos a complexidade **$O(1)$** .

void Desenfileira(TFila *Fila, Cliente *cliente): executa apenas comandos $O(1)$ sem nenhum loop. Portanto temos a complexidade **$O(1)$** .

TAD Caixa

void inicializaCaixas(Caixa *caixa): possui atribuições $O(1)$ e 3 loops, porém estes loops possuem valores definidos como condição de parada e o conteúdo deles são comandos $O(1)$. Portanto temos a complexidade **$O(1)$** .

int caixaAtendenteVazio(Caixa *caixa): possui um loop, porém esse loop possui valor definido como condição de parada e o conteúdo deles são comandos $O(1)$. Portanto temos a complexidade **$O(1)$** .

int caixaAutomaticoVazio(Caixa *caixa): possui um loop, porém esse loop possui valor definido como condição de parada e o conteúdo deles são comandos $O(1)$. Portanto temos a complexidade **$O(1)$** .

int caixaCambioVazio(Caixa *caixa): possui um loop, porém esse loop possui valor definido como condição de parada e o conteúdo deles são comandos $O(1)$. Portanto temos a complexidade **$O(1)$** .

int caixaCambioDisponivel(Caixa *caixa): utiliza-se de dois loops, porém ambos possuem valores definidos como condição de parada. Portanto temos a complexidade **$O(1)$** .

int caixaAtendenteDisponivel(Caixa *caixa): utiliza-se de dois loops, porém ambos

possuem valores definidos como condição de parada. Portanto temos a complexidade **O(1)**.

int caixaAutomaticoDisponivel(Caixa *caixa): utiliza-se de dois loops, porém ambos possuem valores definidos como condição de parada. Portanto temos a complexidade **O(1)**.

int caixaAutomaticoDisponivelProximo(Caixa *caixa, int i): possui um loop que depende do valor de i para determinar a sua condição de parada. Portanto temos a complexidade **O(n)**.

void liberaCaixa(Caixa *caixa, int i): possui apenas uma operação **O(1)**. Portanto temos a complexidade **O(1)**.

void enviarParaOCAixa(Caixa *caixa, int i): possui apenas uma operação **O(1)**. Portanto temos a complexidade **O(1)**.

unsigned long int menorTempoSaidaAtendente(Caixa *caixa): possui um loop, porém esse loop possui valor definido como condição de parada e o conteúdo deles são comandos **O(1)**. Portanto temos a complexidade **O(1)**.

unsigned long int menorTempoSaidaAutomatico(Caixa *caixa): possui um loop, porém esse loop possui valor definido como condição de parada e o conteúdo deles são comandos **O(1)**. Portanto temos a complexidade **O(1)**.

unsigned long int menorTempoSaidaAutomaticoProximo(Caixa *caixa, int i): possui um loop que depende do valor de i para determinar a sua condição de parada. Portanto temos a complexidade **O(n)**.

void inicializaTempoComCaixa(Cliente *cliente, Caixa *caixa, int i, Cliente *saida, int *posicao): executa apenas comandos **O(1)**. Portanto temos a complexidade **O(1)**.

void inicializaTempoSemCaixa(Cliente *cliente, Caixa *caixa, int i, Cliente *saida, int *posicao): executa apenas comandos **O(1)**. Portanto temos a complexidade **O(1)**.

TAD Arquivo

char *strtok_single (char * str, char const * delims): executa apenas comandos **O(1)**. Portanto temos a complexidade **O(1)**.

int leQuantidadeDeLinhas(char *caminhoDoArquivo, Cliente **cliente): possui um loop que depende de quando o arquivo vai terminar, ou seja, vai executar n vezes o loop. Portanto temos a complexidade **O(n)**.

void preecheDadosDoArquivo(char *caminhoDoArquivo, Cliente *cliente): possui um loop que depende de quando o arquivo vai terminar, ou seja, vai executar n vezes o loop. Portanto temos a complexidade **O(n)**.

void imprimeSaida(Cliente *cliente, char *caminhoDoArquivo, int numeroDeLinhas, Agencia *agencia): possui dois loops, porém não estão dentro do outro. A condição de parada depende de quantas linhas o arquivo possui. Portanto temos a complexidade **O(n)**.

TAD Agencia

void inicializaAgencia(Agencia *agencia): executa apenas comandos $O(1)$. Portanto temos a complexidade $O(1)$.

void fazerOperacaoSaque(Cliente *cliente, Agencia *agencia): executa apenas comandos $O(1)$. Portanto temos a complexidade $O(1)$.

void fazerOperacaoDeposito(Cliente *cliente, Agencia *agencia): executa apenas comandos $O(1)$. Portanto temos a complexidade $O(1)$.

void fazerOperacaoCambio(Cliente *cliente, Agencia *agencia): executa apenas comandos $O(1)$. Portanto temos a complexidade $O(1)$.

void verificarOperacao(Cliente *cliente, TFila *filaAutomatico, TFila *filaAtendente, TFila *filaCambio): executa apenas comandos $O(1)$. Portanto temos a complexidade $O(1)$.

void desenfileiraCambio(Caixa *caixa, TFila *filaCambio, int *posicaoVetorSaida, Cliente *clientesSaida): possui um loop que depende de quantos clientes estão enfileirados. Portanto temos a complexidade $O(n)$.


void desenfileiraAtendente(Caixa *caixa, TFila *filaAtendente, int *posicaoVetorSaida, Cliente *clientesSaida): possui um loop que depende de quantos clientes estão enfileirados. Portanto temos a complexidade $O(n)$.

void desenfileiraAutomatico(Caixa *caixa, TFila *filaAutomatico, int *posicaoVetorSaida, Cliente *clientesSaida): possui 4 loops, todos eles possuem uma condição de parada dependente. Porém em uma condição 2 desses loops estão dentro do outro. Portanto podemos temos a complexidade de $O(n^2)$.

PROGRAMA PRINCIPAL: devido ao programa chamar apenas uma vez algumas funções, a sua complexidade será calculada a partir da maior complexidade dentre as outras funções. Deste modo, temos a complexidade de $O(n^2)$.

4. Testes

4.1 Passagem de argumento incorreta na execução do programa:

 vitor-aguiar@vitoraguiar-VirtualBox: /mnt/aeds

```
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ ./TP11 arquivo.csv
```

```
O numero de argumentos esta errado. O programa deve receber 2 argumentos.
```

```
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ █
```


4.2 Exemplo da especificação com os 29 caixas:

entrada - Bloco de notas		saida - Bloco de notas	
Arquivo	Editar	Arquivo	Editar
1444771699,André Alves,SAQUE,-500.00,,200		1444771865,Giovana Galvão,SAQUE,-100.00,0.00,60	
1444771730,Junia Jansen,DEPOSITO,300.37,,550		1444771885,Ricardo Romário,CÂMBIO,400.00,-125.44,150	
1444771735,Ricardo Romário,CÂMBIO,500.00,-125.44,150		1444771899,André Alves,SAQUE,-100.00,-125.44,200	
1444771801,Sandrei Sousa,SAQUE,-700.00,,150		1444771951,Sandrei Sousa,SAQUE,-800.00,-125.44,150	
1444771805,Giovana Galvão,SAQUE,-100.00,,60		1444772280,Junia Jansen,DEPOSITO,-499.63,-125.44,550	
1444771811,Fabio Feliciano,CAIXA,-1500.00,,480		1444772291,Fabio Feliciano,CAIXA,-1999.63,-125.44,480	
1444771831,Larissa Lousano,CAIXA,500.00,,480		1444772311,Larissa Lousano,CAIXA,-1499.63,-125.44,480	
		VARIACAO DE BRL: -1999.63 a 400.00	
		VARIACAO DE USD: -125.44 a 0.00	
		VARIACAO DE TEMPO: 60 a 550	

4.3 Colocando mais de um cliente com operação de câmbio para simular a situação com fila:

entrada - Bloco de notas		saida - Bloco de notas	
Arquivo	Editar	Arquivo	Editar
1444771699,André Alves,SAQUE,-500.00,,200		1444771865,Giovana Galvão,SAQUE,-100.00,0.00,60	
1444771730,Junia Jansen,DEPOSITO,300.37,,550		1444771885,Ricardo Romário,CÂMBIO,400.00,-125.44,150	
1444771735,Ricardo Romário,CÂMBIO,500.00,-125.44,150		1444771899,André Alves,SAQUE,-100.00,-125.44,200	
1444771740,Adriano,CÂMBIO,350.00,-150.44,200		1444771951,Sandrei Sousa,SAQUE,-800.00,-125.44,150	
1444771801,Sandrei Sousa,SAQUE,-700.00,,150		1444772085,Adriano,CÂMBIO,-450.00,-275.88,345	
1444771805,Giovana Galvão,SAQUE,-100.00,,60		1444772280,Junia Jansen,DEPOSITO,-149.63,-275.88,550	
1444771811,Fabio Feliciano,CAIXA,-1500.00,,480		1444772291,Fabio Feliciano,CAIXA,-1649.63,-275.88,480	
1444771831,Larissa Lousano,CAIXA,500.00,,480		1444772311,Larissa Lousano,CAIXA,-1149.63,-275.88,480	
		VARIACAO DE BRL: -1649.63 a 400.00	
		VARIACAO DE USD: -275.88 a 0.00	
		VARIACAO DE TEMPO: 60 a 550	

4.4 Colocando mais de três clientes com operação de caixa para simular a situação com fila:

entrada - Bloco de notas		saida - Bloco de notas	
Arquivo	Editar	Arquivo	Editar
1444771699,André Alves,SAQUE,-500.00,,200		1444771865,Giovana Galvão,SAQUE,-100.00,0.00,60	
1444771730,Junia Jansen,DEPOSITO,300.37,,550		1444771885,Ricardo Romário,CÂMBIO,400.00,-125.44,150	
1444771735,Ricardo Romário,CÂMBIO,500.00,-125.44,150		1444771899,André Alves,SAQUE,-100.00,-125.44,200	
1444771820,João,CAIXA,520.00,,230		1444771935,Renato,CAIXA,-200.00,-125.44,85	
1444771801,Sandrei Sousa,SAQUE,-700.00,,150		1444771951,Sandrei Sousa,SAQUE,-900.00,-125.44,150	
1444771850,Renato,CAIXA,-100.00,,85		1444772050,João,CAIXA,-380.00,-125.44,230	
1444771805,Giovana Galvão,SAQUE,-100.00,,60		1444772280,Junia Jansen,DEPOSITO,-79.63,-125.44,550	
1444771811,Fabio Feliciano,CAIXA,-1500.00,,480		1444772291,Fabio Feliciano,CAIXA,-1579.63,-125.44,480	
1444771831,Larissa Lousano,CAIXA,500.00,,480		1444772415,Larissa Lousano,CAIXA,-1079.63,-125.44,584	
		VARIACAO DE BRL: -1579.63 a 400.00	
		VARIACAO DE USD: -125.44 a 0.00	
		VARIACAO DE TEMPO: 60 a 584	


4.5 Quando alguma linha do arquivo de entrada possui tamanho maior que 300 caracteres:

```
vitor-aguiar@vitoraguiar-VirtualBox: /mnt/aeds$ gcc -g -Wall -std=c99 *.c -o tp1
vitor-aguiar@vitoraguiar-VirtualBox: /mnt/aeds$ ./tp1 arquivo.csv arquivo2.csv

Numero de caracteres na linha invalido. Maximo eh 300.

vitor-aguiar@vitoraguiar-VirtualBox: /mnt/aeds$
```


4.6 Quando algum nome do arquivo de entrada possui mais que 100 caracteres ou se esta vazio:


 vitor-aguiar@vitoraguiar-VirtualBox: /mnt/aeds

```
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ gcc -g -Wall -std=c99 *.c -o tp1
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ ./tp1 arquivo.csv arquivo2.csv
```

Numero de caracteres no campo nome invalido. Maximo eh 100.

vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds\$ █

4.7 Quando alguma operação do arquivo de entrada é diferente de SAQUE, DEPOSITO, CAIXA ou CÂMBIO:


 vitor-aguiar@vitoraguiar-VirtualBox: /mnt/aeds

```
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ gcc -g -Wall -std=c99 *.c -o tp1
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ ./tp1 arquivo.csv arquivo2.csv
```

Alguma operacao de um cliente no arquivo esta invalida.

vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds\$ █

4.8 Quando em alguma linha do arquivo de entrada falta alguma informação (possui algo diferente do que 5 vírgulas):

 vitor-aguiar@vitoraguiar-VirtualBox: /mnt/aeds

```
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ gcc -g -Wall -std=c99 *.c -o tp1
vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds$ ./tp1 arquivo.csv arquivo2.csv
```

Numero de informacoes em uma linha esta incorreto.

vitor-aguiar@vitoraguiar-VirtualBox:/mnt/aeds\$ █

4.9 Colocando quinze clientes com operação no caixa automático sem leitor e onze clientes com operação no caixa com leitor:

entrada - Bloco de notas		saida - Bloco de notas	
Arquivo	Editar	Arquivo	Editar
Formatar	Exibir	Formatar	Exibir
Ajuda		Ajuda	
1444771699,André Alves,SAQUE,-300.00,,200		1444771770,Claudia,DEPOSITO,350.37,0.00,20	
1444771730,Junia Jansen,DEPOSITO,300.37,,550		1444771790,Maluf,DEPOSITO,950.37,0.00,55	
1444771735,Maluf,DEPOSITO,600,,55		1444771795,Gustavo,DEPOSITO,1030.37,0.00,20	
1444771740,Spencer,DEPOSITO,200,,115		1444771840,Walmir,DEPOSITO,1080.37,0.00,80	
1444771735,Ricardo Romário,CÂMBIO,500.00,-125.44,150		1444771855,Spencer,DEPOSITO,1280.37,0.00,115	
1444771801,Sandrei Sousa,SAQUE,-700.00,,150		1444771865,Giovana Galvão,SAQUE,1180.37,0.00,60	
1444771745,Bruna,DEPOSITO,225,,350		1444771870,Vitor,DEPOSITO,1330.37,0.00,100	
1444771750,Claudia,DEPOSITO,350.37,,20		1444771885,Ricardo Romário,CÂMBIO,1830.37,-125.44,150	
1444771755,Douglas,DEPOSITO,700,,280		1444771899,André Alves,SAQUE,1530.37,-125.44,200	
1444771760,Walmir,DEPOSITO,50,,80		1444771945,Franklin,DEPOSITO,1755.37,-125.44,160	
1444771810,Bruno,SAQUE,-850.00,,300		1444771951,Sandrei Sousa,SAQUE,1055.37,-125.44,150	
1444771820,Samir,SAQUE,-750.00,,150		1444771970,Samir,SAQUE,305.37,-125.44,150	
1444771830,Maria,SAQUE,-650.00,,200		1444771970,Ralph,DEPOSITO,605.37,-125.44,180	
1444771765,Fernando,DEPOSITO,1000,,400		1444772030,Maria,SAQUE,-44.63,-125.44,200	
1444771770,Vitor,DEPOSITO,150,,100		1444772035,Douglas,DEPOSITO,655.37,-125.44,280	
1444771775,Gustavo,DEPOSITO,80,,20		1444772095,Bruna,DEPOSITO,880.37,-125.44,350	
1444771780,Rafael,DEPOSITO,600,,315		1444772095,Rafael,DEPOSITO,1480.37,-125.44,315	
1444771840,Pedro,SAQUE,-550.00,,300		1444772100,Marcio,SAQUE,1030.37,-125.44,250	
1444771850,Marcio,SAQUE,-450.00,,250		1444772101,Alexandre,SAQUE,680.37,-125.44,241	
1444771860,Alexandre,SAQUE,-350.00,,150		1444772110,Bruno,SAQUE,-169.63,-125.44,300	
1444771785,Franklin,DEPOSITO,225,,160		1444772135,Silvio,SAQUE,-569.63,-125.44,245	
1444771805,Giovana Galvão,SAQUE,-100.00,,60		1444772140,Pedro,SAQUE,-1119.63,-125.44,300	
1444771811,Fabio Feliciano,CAIXA,-1500.00,,480		1444772145,Gabriel,SAQUE,-1919.63,-125.44,245	
1444771831,Larissa Lousano,CAIXA,500.00,,480		1444772165,Fernando,DEPOSITO,-919.63,-125.44,400	
1444771790,Ralph,DEPOSITO,300,,180		1444772170,Lourdes,SAQUE,-1519.63,-125.44,300	
1444771870,Lourdes,SAQUE,-600.00,,200		1444772180,Sara,SAQUE,-2019.63,-125.44,300	
1444771880,Sara,SAQUE,-500.00,,150		1444772280,Junia Jansen,DEPOSITO,-1719.26,-125.44,550	
1444771890,Silvio,SAQUE,-400.00,,100		1444772291,Fabio Feliciano,CAIXA,-3219.26,-125.44,480	
1444771900,Gabriel,SAQUE,-800.00,,50		1444772311,Larissa Lousano,CAIXA,-2719.26,-125.44,480	
		VARIACAO DE BRL: -3219.26 a 1830.37	
		VARIACAO DE USD: -125.44 a 0.00	
		VARIACAO DE TEMPO: 20 a 550	

5. Conclusão

O trabalho foi bem interessante, é possível ver uma implementação semelhante para uma situação real, ou seja, isso poderia ser um produto no futuro.

O tempo demandado para a realização do trabalho foi longo, encontrando dificuldades sendo algumas podendo ser evidenciadas como: lógica para fazer o cliente passar na frente na fila de caixa automático, demora para entender como os cálculos de tempo de saída e tempo de fila iriam ser realizados.

6. Referências

- <http://stackoverflow.com/>
- <http://www.cplusplus.com/>
- <http://www.cprogressivo.net/>
- <http://www.tutorialspoint.com/>

7. Anexos

Listagem dos programas:

- Fila.h
- Fila.c
- Caixa.h
- Caixa.c
- Arquivo.h
- Arquivo.c
- Agencia.h
- Agencia.c
- Main.c