## 1 Introdução

A proposta do trabalho foi confeccionar um programa em linguagem C que execute as ações de um banco tais como guardar dados do cliente e sua conta e que execute transações como deposito, saque e transferência de saldo entre clientes, como extra a missão era armazenar estes dados, tanto do cliente quanto das transações em arquivos .txt e a partir deles inferir o saldo das contas consultando todas as transações envolvidas com aquela conta. Todas essas tarefas executam perfeitamente no algoritmo construído por mim.

## 2- Solução Proposta

**Main** – A função principal deste código foi propositalmente implementada de forma a deixar a responsabilidade das execuções do que se pede às sub-rotinas, deixando a main responsável apenas por exibir os menus e de acordo com a entrada do usuário alternar entre as funções através de alguns “switch-case”.

**Cadastra –** Recolhe as informações do usuário para inserir um novo cliente e armazena na posição X do vetor de clientes (struct), x é uma variável global que é incrementada sempre que um usuário é cadastrado, tanto via função cadastro quanto via arquivo .txt.

**Busca** - Percorre todas as posiçoes em busca do elemento com auxilio de um loop. Pode buscar por cpf ou por numero de identificação bancário, dentro do loop existe um “if” que confere se o numero que o usuário está buscando é igual ao cliente na posição (do vetor) atual, se sim imprime os dados do cliente na própria função.

**Deposito** – Percorre o vetor de clientes similarmente à função de busca, caso encontre o numero pesquisado, pede ao usuário o valor a ser depositado e a data da transação (data é armazenado em uma variável do tipo char), em seguida a função imprime a nova transação no arquivo de consulta no modo append, isto é, a nova linha é escrita abaixo da ultima linha do arquivo.

**Saque** – Quase idêntica ao deposito, com a diferencia óbvia de debitar o valor no saldo do usuário ao invés de acrescer.

**Transferencia** – Pesquisa pelo usuário a transferir o dinheiro, depois deve buscar pelo usuário a receber, caso encontre, pergunta-se o valor a ser transferido e a data, o primeiro cliente tem o valor debitado em sua conta e o segundo tem o valor creditado.

**Consulta** - Procura pelo cliente informado e ao encontra-lo, percorre o arquivo “contas.txt” em busca de linhas que envolvam o numero do cliente que foi encontrado, todos os depósitos acrescem valor ao saldo, todo saque decrescem no valor d o saldo, transferências que o usuário efetuou também, transferências que o usuário recebeu tem sal valor creditado no saldo. Para dividir os campos do arquivo, usei um separador, o caracter “|”, e para separar os campos no código usei a função “strtok” que identifica o caracter e salva o

que foi lido anteriormente numa variável chamada token, o buffer de token é limpo e ele pode ler outro campo ate que encontre outro “|”, e assim por diante.

**Load** – percorre o arquivo txt “cliente.txt” e a cada linha armazena as informações em uma posição do vetor de clientes, aqui também foi usado a função “strtok” explicada anteriormente.

**Save** - escreve os dados salvos no vetor de clientes, no arquivo. O arquivo é aberto no modo “write”, isto é, apaga os dados anteriormente inseridos e reescreve em cima no arquivo, optei por essa abordagem porque no caso de não existir arquivo de dados, o algoritmo cria um novo arquivo. Existe um if que verifica se está salvando a ultima linha do arquivo, para que não seja escrito um “\n” no fim do arquivo, o que faria o código mão executar como desejado.

**LoadT** – Similar ao Load, mas para o arquivo de contas, para o funcionamento deste código é de grande utilidade saber quantas transações, este é o porque dessa função, percorrer o arquivo e salvar o numero de linhas na variável “transacao”.

### 3 – Analise de Complexidade

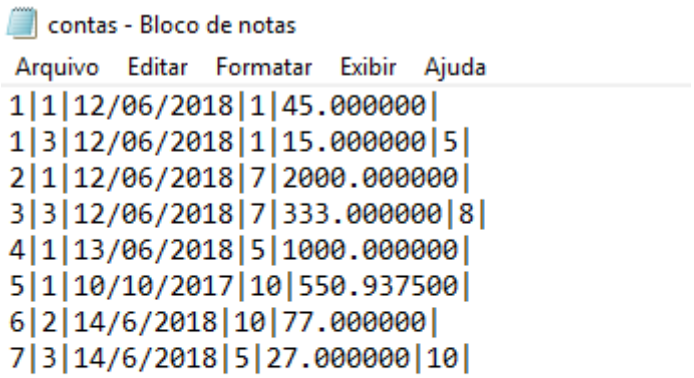
Utilizei a abordagem do pior caso (big O) para analizar a complexidade desse programa .

| Função | Complexidade |
|---|---|
| Main | O(1) |
| Cadastra | O(1) |
| Busca | O(N) |
| Deposito | O(N) . |
| Saque | O(N). |
| Transferencia | $O(N^2)$. |
| Consulta | $O(N^2)$. |
| Load | O(N) |
| Save | O(N) |
| LoadT | O(N) |

Como o algoritmo tem menus que vem e vão, o tempo e complexidade do programa depende muito do usuário, de quantos clientes são cadastrados e quantas operações sao efetuadas.

### 4 – Experimento

Inúmeros testes foram realizados para detectar erros e possível bugs, o presente trabalho, julgo eu, está funcionando exatamente como se pede. Abaixo exemplificarei a disposição dos campos no arquivo.




O primeiro campo é o numero da transação, o segundo é o tipo de transação, seguido da data, numero do cliente, o valor envolvido e, no caso da operação ser uma transferência, existe um sexto campo que representa o cliente que recebeu o dinheiro da transação.

## 5 – Maquina Utilizada

A aplicação foi inteiramente escrita e compilada em um notebook HP 1000, com processador i3 (2,2GHz), 2GB de memoria primária , 500GB de memória secundaria. Sistema operacional windows 10 – 64bits.

## 6 - Conclusão

Minha principal dificuldade foi manipular os arquivos .txt, por muitas vezes o programa não funcionava por conta deles, demorou certo tempo para perceber que uma linha em branco no fim do arquivo era o suficiente pra fazer o programa parar de responder, por muitas vezes me perdi no raciocicio para realizar uma simples tarefa do programa, mas com um mapa do programa feito à mão pude entender melhor, e consegui realizar tudo que foi pedido no trabalho.

## 7 - Referências


**-** **https://www.scriptbrasil.com.br/forum/topic/117251-duvida-como-usar-a-fun%C3%A7%C3%A3o-strcpy/**

**- Slides disponibilizados pelo professor**

**-** **www.cprogressivo.net**