UNIVERSIDADE FEDERAL DE LAVRAS – GAC109

**PROJETO PRÁTICO:** Sistema de Cadastro em Arquivos com Ordenação

**NOMES:** Dhiogo Vinicius Gonçalves – 202220328 Nycolas Purificação Assis – 202220344

Tiago Gualberto Bento – 202220338

# INTRODUÇÃO

Foi desenvolvido um trabalho que consiste na simulação de um sistema capaz de manipular informações de uma planilha de dados em formato .csv. Esse sistema tem a capacidade de buscar, inserir, editar, excluir, ordenar e exibir os dados contidos na planilha. É importante destacar que o sistema simula o armazenamento de informações sobre clientes de um e-commerce, utilizando como base de dados as informações discutidas posteriormente.

# BASE DE DADOS

A base de dados criada apresenta seis colunas com informações de clientes. Elas são: nome, sexo, CPF, dinheiro, conexão e idade. Vale destacar que, em relação a primeira etapa enviada do trabalho, o campo da idade foi adicionado e a coluna do CPF substituiu a coluna de CEP, visto que buscou-se armazenar um dado que fosse exclusivo para cada cliente. Para a criação dos CPF e das idades utilizou-se a função “ALEATÓRIOENTRE(x;x)” e o sexo, nome, conexão e dinheiro foram inseridos pelos próprios estudantes.

# ESTRUTURA DO CÓDIGO

* **IMPORTAÇÃO**

Inicialmente, foi realizado o processo de leitura de um arquivo no formato .csv. Para garantir a leitura correta de todas as informações presentes nas linhas, foram adicionadas condições específicas. Um procedimento foi criado para lidar com a separação das colunas através da vírgula (",") e a mudança de linha através do caractere de quebra de linha ("\n"). Além disso, foi implementada uma condição para garantir a correta leitura das células no arquivo .csv, mesmo quando estas continham aspas.

Durante a leitura do arquivo, utilizou-se um registro chamado "dados" para armazenar as informações no formato desejado. Por exemplo, o nome foi armazenado como um vetor de caracteres e o valor monetário como um tipo float. As informações contidas no registro foram então gravadas em um arquivo binário chamado "clientes.bin", o qual seria utilizado para as funções subsequentes.

Vale destacar que, no caso das variáveis dinheiro e idade, elas foram inicialmente lidas como strings. Porém, por meio das funções "stof()" e "stoi()", respectivamente, foram convertidas para os tipos de ponto flutuante e inteiro, apropriados para manipulação dos dados.

* **ORDENAÇÃO DE DADOS**

A partir do arquivo binário, foi utilizado o algoritmo Shell Sort para ordenar os dados com base nas colunas contendo o nome e a idade dos clientes. Foi desenvolvido um procedimento que permite ao usuário escolher qual campo deseja utilizar para ordenação dos dados. Para cada opção de ordenação, foi criado um procedimento adicional que abre o arquivo para leitura, verifica os registros marcados como deletados (utilizando um procedimento auxiliar que informa a quantidade de dados no arquivo), copia os registros não apagados para um vetor do tipo "dados" e, em seguida, aplica o algoritmo Shell Sort para ordená-los. A sequência de Knuth foi utilizada para determinar o intervalo de comparação durante o Shell Sort.

Após a conclusão da ordenação, o vetor é copiado novamente para o arquivo binário por meio de um procedimento auxiliar. Esse procedimento apaga todos os dados existentes no arquivo e, em seguida, reescreve os registros, desta vez na ordem correta conforme a ordenação realizada.

* **IMPRESSÃO NA TELA**

Para exibir os dados na tela, foi criado um procedimento que utiliza um procedimento auxiliar para fornecer ao usuário informações sobre a quantidade total de dados no arquivo, bem como a quantidade de registros que foram marcados como apagados. Após essa apresentação inicial, o usuário tem a opção de escolher entre imprimir todos os dados do arquivo na tela ou selecionar um intervalo específico, que vai de um cliente inicial até um cliente final.

É importante destacar que apenas os registros que não foram marcados como apagados são exibidos na tela.

* **EXCLUSÃO DE DADOS**

Ao lidar com a exclusão de registros, o arquivo binário é aberto para leitura e escrita, utilizando um procedimento auxiliar para verificar a quantidade de dados presentes no arquivo. Em seguida, uma mensagem é exibida para que o usuário possa digitar o CPF do cliente cujas informações desejam ser buscadas. A busca é realizada de forma sequencial, percorrendo os registros até encontrar o cliente correspondente. Utilizando um marcador booleano incorporado ao registro "dados", o cliente a ser excluído é marcado como um registro apagado (o valor booleano é definido como 1). No entanto, é importante ressaltar que o registro não é completamente removido do arquivo binário. Essa marcação permite que o registro seja tratado como excluído sem ser fisicamente deletado do arquivo. Durante a comparação entre o CPF digitado e os CPFs presentes no arquivo binário, é utilizada a função "strcmp()" da biblioteca "string.h", que verifica a igualdade entre os vetores de caracteres.

* **BUSCA DE DADOS**

Para realizar a busca de um cliente específico, o programa permite ao usuário procurar tanto pelo nome do cliente quanto pelo CPF. Para cada um desses casos de busca, foi criado um procedimento que abre o arquivo binário para leitura. Utilizando o procedimento auxiliar criado para obter o tamanho do arquivo, é possível percorrer todo o arquivo até encontrar o cliente com o nome ou CPF desejado. Durante essa busca, também é utilizado o "strcmp()" para comparar os vetores de caracteres e verificar correspondências.

No entanto, é importante observar que a busca só será feita nos registros que não estejam marcados como apagados, utilizando o marcador booleano mencionado anteriormente. Caso a busca seja bem-sucedida, o ponteiro de leitura do arquivo é posicionado no registro correspondente ao cliente encontrado, e todas as informações relacionadas a esse cliente são exibidas. Por outro lado, se a busca não encontrar correspondências, uma mensagem informando que o cliente não foi encontrado é exibida na tela.

* **INSERÇÃO DE DADOS**

Foi desenvolvido um procedimento responsável por adicionar novos clientes à base de dados, garantindo que o CPF não esteja duplicado. Para isso, o arquivo foi aberto em modo de leitura. Uma mensagem solicita ao usuário que digite o CPF do novo cliente, e em seguida é verificado se o CPF já existe na base de dados. Caso seja um CPF novo, uma série de mensagens solicita as informações do cliente a serem adicionadas.

Após a coleta das informações, é realizada uma verificação para determinar se o arquivo está sendo ordenado por nome ou idade. Em caso afirmativo, o novo cliente é inserido no local correto do arquivo, mantendo a ordenação. É importante ressaltar que os novos clientes são adicionados ao final do arquivo através de um procedimento auxiliar criado para essa finalidade.

* **EXPORTAÇÃO**

Também é possível exportar o arquivo binário criado e modificado. Para isso, foi criado um procedimento que abre o arquivo binário para leitura e abre um arquivo .csv, cujo nome é inserido pelo usuário, limpando todos seus dados e copiando todos os dados do arquivo binário. Para isso, utilizou-se do fato da vírgula separar as colunas e a quebra de linha separar as linhas.

* **EDIÇÃO**

Outra possibilidade do sistema é editar os dados de um cliente. Para isso, o usuário deve digitar o CPF do cliente, e caso o mesmo esteja no sistema, são exibidas mensagens para o armazenamento das novas informações do cliente. Após isso, o registro do cliente é atualizado e é sobrescrito no arquivo binário.

* **PROCEDIMENTO AUXILIARES**

Além dos procedimentos auxiliares mencionados, como a obtenção da quantidade de dados no arquivo, a escrita de novos dados no final do arquivo e a reescrita completa do arquivo com os dados, foi desenvolvido um procedimento para limpar a tela do terminal. Essa função visa manter o ambiente mais organizado e dinâmico durante a interação com o sistema.

Também foi criado um procedimento que contém o menu principal do sistema, apresentando todas as opções mencionadas anteriormente, como a ordenação e inserção de dados. Isso permite que o usuário escolha qual função deseja executar, tendo acesso a todas as opções disponíveis no sistema.

Além disso, foram criadas funções que verificam se a conexão, o CPF, o dinheiro, o sexo e a idade atendem ao padrão de entrada esperado. Dessa forma, evitamos que o sistema quebre caso o usuário digite algum valor que não corresponde a realidade (letras no campo idade, por exemplo).

# CONCLUSÃO

O sistema atual permite ler e modificar a base de dados utilizada, oferecendo simplicidade e acessibilidade ao usuário. Para melhorias, pode-se considerar a capacidade de importar diferentes bases de dados e aprimorar o layout para maior dinamicidade. Além disso, o sistema não possui limite na quantidade de dados a serem armazenados, o que é essencial em sistemas de cadastro. Essas melhorias contribuiriam para a flexibilidade, experiência do usuário e escalabilidade do sistema.