RELATÓRIO FINAL - MINI SISTEMA EM C ANSI

Aluno: Gabriel Desterro.
Disciplina: Lógica de Programação.
Docente: Ridis Ribeiro.

1. INTRODUÇÃO.

O objetivo deste projeto é criar um sistema de cadastro escolar contendo no mínimo as seguintes informações: cursos, turmas, unidades curriculares, nomes de alunos, matrículas, data nascimento, endereço, telefone, três notas, média e condição final de aprovação. Além disso, o sistema precisa aceitar uma quantidade indefinida de cursos, turmas e alunos; assim, possibilitando maior escalabilidade.

Para tanto, foram estabelecidas etapas a serem cumpridas em prazos determinados, de forma que o sistema pudesse ser incrementado conforme fosse ampliada as bases de conhecimento dos discentes. As etapas são: pesquisar e fazer um programa dividido em módulos, pesquisar os principais algoritmos de ordenação e implementar um programa para ordenação do cadastro realizado, pesquisar e fazer um programa em C para Arquivar em disco, e por fim, unir todas as etapas anteriores em um sistema.

2. DESENVOLVIMENTO.

2.1 CABEÇALHO.

Para desenvolver o código foram incluídas as bibliotecas: **<stdio.h>**, **<locale.h>**, **<string.h>**, **<stdlib.h>** e **<ctype.h>**.

<stdio.h> - Biblioteca padrão de C ANSI, onde se encontram a maioria das funções usadas no sistema.

<locale.h> - Biblioteca usada para fazer o programa imprimir acentos gráficos dentre outras particularidades da língua portuguesa.

<string.h> - Biblioteca incluída para auxiliar nas “operações” com strings necessárias no programa.

<stdlib.h> - Biblioteca incluída apenas para usar a função de limpar o console, visando melhorar a experiência do usuário.

<ctype.h> - Biblioteca incluída com o objetivo de corrigir eventuais erros de digitação do usuário, evitando transtornos.

2.2 DECLARAÇÃO STRUCT.

Em seguida, foi declarado um registro struct que foi chamado de *'CadastroEscolar'*. O struct tem como base a ideia de agrupar vários dados sobre um mesmo objeto, assim, pode-se reunir todas as informações pertencentes a um único aluno, sejam elas do tipo inteiro (int), real (float), caracter (char); facilitando assim o armazenamento dos dados e a sua manipulação. Nesta parte, foram declaradas as variáveis com tipo e nomes mostrados abaixo:

```
struct CadastroEscolar
        {
                char curso[50];
                char materia[50];
                char nome[50];
                char endereco[50];
                char telefone[50];
                char turma[50];
                char ra[50];
                int dia;
                int mes;
                int ano;
                float  nota1;
                float  nota2;
                float  nota3;
                float media;
        }
```

2.3 FUNÇÃO CADASTRO.

Na sequência foi criada uma função chamada *'cadastro'* que recebe como parâmetro o struct com um ponteiro para um "apelido", que facilita as referências ao struct no decorrer do algoritmo. Nesta função, foram criados dois padrões para cadastro de dados a partir de entradas do usuário que se repetem para cadastrar todos os dados, foram eles:

Para cadastro de strings.

1° - printf("\nDigite o nome do aluno: ") - A função *'printf( )'* para exibir uma mensagem no console formatada com uma quebra de linha *(\n)*;

2° - fgets(A->nome, **sizeof**(A->nome), **stdin**) - A função *‘fgets( )’* para ler a entrada do usuário e salvar os dados na variável especificada (o *‘fgets’* é mais seguro para strings pois evita buffer overflow);

3° - A->nome[**strcspn**(A->nome, "\n")] = 0 - A função *‘strcspn( )’* para percorrer a cadeia de caracteres até encontrar uma quebra de linha (\n) e substituí-la por por um caractere nulo (\0) evitando que o *‘\n’* cause problemas em comparações e exibições;

4° - A->nome[0] **= toupper**(A->nome[0]) - A função *‘toupper’* para a posição zero da cadeia de caracteres, pois essa função converte a primeira letra digitada para maiúscula, facilitando a organização alfabética (no cadastro, foi usada especialmente para a variável ‘nome’).

Para cadastro de números inteiros e reais.

1° - printf("Digite a data de nascimento do aluno (dd/mm/aaaa): ") - A função *‘printf( )’* para exibir uma mensagem no console;

2° - scanf("%d/%d/%d", &A->dia, &A->mes, &A->ano) - A função *‘scanf( )’* para ler a entrada do usuário e salvar os dados na variável especificada;

3° - getchar() **-** A função *‘getchar( )’* para limpar o buffer após cada entrada do usuário.

Ao final da função *‘cadastro’* a média é calculada a partir dos dados fornecidos pelo usuário e guardada na variável *‘media’* do struct.

2.4 FUNÇÕES ORDENAÇÃO.

Nesta etapa existem duas funções que se complementam, a função *‘trocarCadastro’* e a função *‘bubbleSort’*, que estão presentes no algoritmo com o objetivo de ordenar o salvamento e a exibição dos dados alfabeticamente pelo nome dos alunos.

A função *‘trocarCadastro’*

L1 | void trocarCadastro(struct CadastroEscolar *a, struct CadastroEscolar *b) {
L2 | struct CadastroEscolar temp = *a;
L3 | *a = *b;
L4 | *b = temp; }

L1 - A função *'trocarCadastro'* é declarada e recebe como parâmetros o struct *'CadastroEscolar'* duas vezes e utiliza um "apelido" (*a, *b*)* com um ponteiro para se referir a cada uma delas;

L2 / L4 - No corpo da função, é declarada uma variável (*temp) com ponteiro, e esta que recebe o valor do struct '*a', em seguida o struct *'*a'* recebe o valor do struct *'*b'*, e por último o struct *'*b'* recebe o valor de *'*temp'*, que inicialmente guardava o valor de *'*a'*.

O objetivo desta função é bem simples, através dos passos indicados acima, o struct *'*a'* passa a ter o valor de *'*b'* e o struct *'*b'* passa a ter o valor de *'*a'*, invertendo a ordem em que foi cadastrada.

A função *'bubbleSort'*

```
L1 | void bubbleSort(struct CadastroEscolar *A, int n) {
L2 |       int i, j;
L3 |       for (i = 0; i < n - 1; i++) {
L4 |           for (j = 0; j < n - i - 1; j++) {
L5 |               if (strcmp(A[j].nome, A[j+1].nome) > 0) {
L6 |                   trocarCadastro(&A[j], &A[j+1]);
L7 |               }
L8 |           }
L9 |       }
L10 |  }
```

L1 - A função *'bubbleSort'* é declarada e recebe como parâmetros o struct *'CadastroEscolar'* com um ponteiro, e um número inteiro *'n'*, que neste código indica o número de alunos cadastrados;

L2 - São declaradas duas variáveis do tipo *'int'* (i, j);

L3 / L4 - Laço *'for'* externo e interno respectivamente, o externo controla o número de repetições necessárias para ordenar todos os elementos e o interno percorre o vetor comparando pares vizinhos dos elementos;

L5 - É usada a função strcmp como condição da estrutura *'if'*. Esta função compara os nomes dos alunos e retorna um valor maior que zero (tornando a condicional verdadeira) se o primeiro parâmetro (A[j].nome) vier depois do segundo parâmetro;

L6 - Chama a função complementar *'trocarCadastro'*.

2.5 FUNÇÃO IMPRIMIR / SALVAR.

Na função *'imprimirSalvarCadasto'*, duas funcionalidades foram implementadas simultaneamente, a de exibir os dados cadastrados e a de salvar os dados cadastrados num arquivo em disco.

A função *'imprimirSalvarCadasto'.*

```
L1 | int imprimirSalvarCadastro (struct CadastroEscolar *A, int size) {
L2 |     int cont;
L3 |     for (cont = 0; cont <= size; cont++) {
L4 |     const char *endereco_do_arquivo = "D:\\Cadastro_Alunos_Senai.txt";
L5 |     FILE *arquivo_cadastro;
L6 |     arquivo_cadastro = fopen(endereco_do_arquivo, "w");
L7 |     if (arquivo_cadastro == NULL) {
L8 |             printf("Erro ao abrir o arquivo\n");
L9 |             return 1;
L10 |  } else { printf("Arquivo salvo com sucesso!\n"); }
L11 | fprintf(arquivo_cadastro, "Cadastro de alunos - Senai\n");
L12 | printf("\nNome do %d° aluno: %s", cont+1, A[cont].nome);
L13 | fprintf(arquivo_cadastro, "\nNome do %d° aluno: %s", cont+1, A[cont].nome);
L14 | …
L15 | fclose(arquivo_cadastro); } }
```

L1 - A função *'imprimirSalvarCadasto'* é declarada e recebe como parâmetros o struct *'CadastroEscolar'* com um ponteiro, e um número inteiro *'size'*, que indica o número de alunos cadastrados;

L2 / L3 - Declara e inicia a variável *'cont'* em zero faz ela repetir até ser menor ou igual a *'size'*, aumentando de um em um*;*

L4 - Define o diretório onde o arquivo será salvo e guarda numa constante do tipo *'char'* com ponteiro;

L5 - Define uma variável com ponteiro do tipo *'FILE'* que será usada para controlar o arquivo;

L6 - É chamada a função *'fopen( )'* para abrir o arquivo no modo *'w'* (este modo abre um arquivo se ele não existir e sobrescreve caso exista) e atribui como valor da variável *'arquivo_cadastro'*;

L7 / L10 - É criada uma condicional *'if( )'* que verifica se houve erro ao abrir o arquivo e retorna uma mensagem indicativa;

L11 / L14 - São usadas as funções *'printf( )'* e *'fprintf( )'* alternadamente para mostrar os dados no console e em seguida salvar os mesmos dados num arquivo .txt, respectivamente;

L15 - É usada a função *'fclose( )'* para fechar o arquivo.

2.6 FUNÇÃO MAIN.

Iniciando a função *'main( )',* é chamada a função *'setlocale( ),'* usada para permitir que o console interprete corretamente acentos e caracteres especiais em português. Em seguida são feitas as declarações iniciais, tratam-se de variáveis de controle para repetição e navegação pelo menu. Por fim, é declarado o struct *'CadastroEscolar'* como um vetor que permite cadastrar até 100 alunos.

Após a mensagem de boas vindas, inicia-se a lógica principal do programa: um menu com cinco opções acessado com a função *'switch'*, rodando dentro de um laço *'while'*.

Case 1: Cadastro.

Quando o usuário seleciona essa opção, entra-se em um *'while'* interno controlado pela variável *'condicao'*. Dentro desse loop, é chamada a função *'cadastro( )'* que está recebendo como argumento o endereço (ponteiro) do vetor de structs *'A'* na posição *'n'*, para que a função possa preencher os campos diretamente nessa posição da memória. É também uma forma de otimizar o código, já que evita cópias desnecessárias dos dados e permite modificações diretas no vetor principal.

Depois que o cadastro é feito, o programa já verifica a média das notas do aluno cadastrado e classifica se ele está aprovado, em recuperação ou reprovado, mostrando isso ao usuário. Em seguida, pergunta se o usuário deseja cadastrar outro aluno, controlando se o loop continua ou não. O índice *'n'* é incrementado a cada novo cadastro.

Case 2: Alteração.

Primeiro, o programa verifica se existe pelo menos um aluno cadastrado. Se sim, exibe uma lista numerada com os nomes dos alunos cadastrados e pede para o usuário escolher qual deseja alterar. O número escolhido é guardado na variável de controle *'alteracao'* e é decrementado para que fique compatível com a listagem do vetor *'A'* que começa em zero.

Depois disso, é exibido um novo menu, agora com as opções de campos a serem alterados, a lista de opções segue o mesmo padrão relacionado no item 2.2 deste relatório. O valor digitado pelo usuário é guardado na variável de controle *'escolhaAlteracao'* e é tratado em outro *'switch'*, onde cada opção trata a alteração de um campo específico e o faz sobrescrevendo os dados na posição de memória indicada usando a mesma lógica apresentada na função *'cadastro'*.

Case 3: Exclusão.

Usa uma lógica semelhante a do *'case 2'*, primeiro é verificado se há alunos cadastrados, depois há a exibição da lista. Quando o usuário escolhe quais dados deseja deletar, é realizada uma operação semelhante a feita na função 'trocarCadastro':

```
L1 | for (i = exclusao; i<= n-1; i++){
L2 |     A[i] = A[i + 1]; }
L3 | n--; }
```

L1 / L3 - Copia os dados de cada aluno uma posição à frente no vetor *'A'* e sobrescreve o anterior, sobrescrevendo os dados escolhidos para exclusão. No final, decrementa *'n'* para refletir o novo número de alunos.

Case 4: Impressão / Salvamento.

Aqui, o programa chama duas funções: *'bubbleSort( )'* — passando como argumento o struct *'A'* e a variável de controle *'n'*, respectivamente — e *'imprimirSalvarCadastro( )'* — passando como argumento o struct *'A'* e a variável de controle *'n'* com decremento de um no índice.

Ao *'bubbleSort(A, n)'*, é passando o vetor completo de alunos junto com o número de alunos cadastrados. A função vai ordenar os alunos em ordem alfabética com base no nome. Como o vetor é passado diretamente, qualquer mudança feita dentro da função afeta o vetor original.

Logo depois, a função *'imprimirSalvarCadastro(A, n-1)'* é chamada. Aqui, o argumento *'n-1'* indica o índice máximo do vetor, ou seja, o último aluno válido. Essa função exibe as informações no terminal e também as salva em um arquivo *'.txt'*.

3. CONCLUSÃO.

Esse programa foi desenvolvido com o objetivo de funcionar como um sistema de cadastro e gerenciamento de informações de alunos. Ele permite que o usuário

cadastre, altere, exclua, visualize e ordene os dados de até 100 alunos por meio de um menu interativo no terminal.