(Desafio) Código Fragmentado

Era uma vez, em uma pequena empresa de software, um código que parecia perfeito. Ele havia sido cuidadosamente desenvolvido pelos alunos em uma sala de aula, cada um acrescentando sua parte com dedicação.

O sistema de cadastro de alunos era funcional, com módulos bem definidos, manipulação de arquivos e até um sistema de segurança rudimentar com senha de administrador. Tudo estava indo bem… até que o **estagiário** chegou.

O jovem, entusiasmado para mostrar serviço, decidiu que poderia "otimizar" o código. Ele começou a mexer aqui e ali, separando funções, reorganizando arquivos e criando bibliotecas extras. Mas, sem querer, ele acabou **fragmentando** o código. As funções perderam suas referências, alguns arquivos foram duplicados sem motivo, e, o pior de tudo, a lógica de algumas partes foi completamente quebrada.

O sistema que funcionava perfeitamente na semana anterior agora **não compila mais**, gerando erros de todas as partes. Ao tentar rodar o programa, a tela piscava com mensagens de erro: “Função não encontrada”, “Arquivo não pode ser aberto”, “Entrada inválida” — um verdadeiro pesadelo!

Diante da confusão gerada, os desenvolvedores seniores ficaram desesperados. Sabiam que a equipe de especialistas — os mesmos alunos que haviam criado o código inicialmente — poderiam consertar a bagunça. Eles tinham a experiência, o conhecimento e o trabalho em equipe necessários para restaurar a ordem.

Então, uma chamada de emergência foi feita: **"Chame os especialistas!"**

Agora, **cada um dos alunos** será encarregado de resolver o problema. Eles terão que trabalhar, refatorar o código fragmentado e, incrementar o sistema com novas funcionalidades que não estavam presentes antes.

A cada rodada deve enviar o código no email do supervisor. O primeiro a enviar os dois códigos corretamente, pois eles serão avaliados, receberá um brinde em gratidão por ajudar a empresa: código, a sair de um código fragmentado e voltar a ter um sistema funcionando.

Boa Sorte!

Rodada 1

Criar o Projeto no Dev-C++

O aluno deve iniciar o projeto no Dev-C++, com um arquivo `.c` básico:

```
#include <stdio.h>

int main() {
    // Código inicial
    return 0;
}
```

Criar Mensagem de Boas-Vindas

O aluno vai adicionar uma mensagem de boas-vindas:

```
#include <stdio.h>

int main() {
    printf("Bem-vindo ao Projeto da Sala de Aula!\n");
    return 0;
}
```

Criar Variável para Entrada de Dados

O aluno deve criar uma variável que irá guardar o nome do aluno:

```
#include <stdio.h>

int main() {
    char nome[50];
    printf("Bem-vindo ao Projeto da Sala de Aula!\n");
    printf("Digite seu nome: ");
    scanf("%s", nome);
    return 0;
}
```

Exibir o Nome do Aluno

O aluno vai exibir o nome digitado pelo usuário:

```
#include <stdio.h>

int main() {
    char nome[50];
    printf("Bem-vindo ao Projeto da Sala de Aula!\n");
    printf("Digite seu nome: ");
    scanf("%s", nome);
    printf("Olá, %s! Bem-vindo ao projeto.\n", nome);
    return 0;
}
```

Implementar um Contador de Iterações

O aluno deve criar um contador que mostrará quantos alunos já participaram:

```
#include <stdio.h>

int main() {
    static int contador = 0;
    char nome[50];

    printf("Bem-vindo ao Projeto da Sala de Aula!\n");
    printf("Digite seu nome: ");
    scanf("%s", nome);

    contador++;
    printf("Olá, %s! Você é o aluno número %d.\n", nome,
contador);

    return 0;
}
```

Modularizar Funções

O aluno modulariza o código em funções. Criar uma função para boas-vindas:

```
#include <stdio.h>

void boasVindas() {
    printf("Bem-vindo ao Projeto da Sala de Aula!\n");
}
```

```
int main() {
    char nome[50];
    boasVindas();
    printf("Digite seu nome: ");
    scanf("%s", nome);
    printf("Olá, %s! Bem-vindo ao projeto.\n", nome);
    return 0;
}
```

Adicionar Função para Capturar Nome

Criar uma função separada para capturar o nome do aluno:

```
#include <stdio.h>

void boasVindas() {
    printf("Bem-vindo ao Projeto da Sala de Aula!\n");
}

void capturarNome(char nome[]) {
    printf("Digite seu nome: ");
    scanf("%s", nome);
}

int main() {
    char nome[50];
    boasVindas();
    capturarNome(nome);
    printf("Olá, %s! Bem-vindo ao projeto.\n", nome);
    return 0;
}
```

Modularizar Função de Contador

O aluno modulariza o contador:

```
#include <stdio.h>

void boasVindas() {
    printf("Bem-vindo ao Projeto da Sala de Aula!\n");
}

void capturarNome(char nome[]) {
    printf("Digite seu nome: ");
    scanf("%s", nome);
}

int contadorAlunos() {
    static int contador = 0;
    contador++;
    return contador;
}

int main() {
    char nome[50];
    boasVindas();
    capturarNome(nome);
    int contador = contadorAlunos();
    printf("Olá, %s! Você é o aluno número %d.\n", nome,
contador);
    return 0;
}
```

Adicionar Função para Exibir Nome e Contador

Criar uma função que exibe a mensagem final:

```
#include <stdio.h>

void boasVindas() {
    printf("Bem-vindo ao Projeto da Sala de Aula!\n");
}

void capturarNome(char nome[]) {
    printf("Digite seu nome: ");
    scanf("%s", nome);
}

int contadorAlunos() {
    static int contador = 0;
    contador++;
    return contador;
}

void exibirMensagemFinal(char nome[], int contador) {
    printf("Olá, %s! Você é o aluno número %d.\n", nome,
contador);
}

int main() {
    char nome[50];
    boasVindas();
    capturarNome(nome);
    int contador = contadorAlunos();
    exibirMensagemFinal(nome, contador);
    return 0;
}
```

Criar Arquivos Externos para Funções

O aluno cria o arquivo `funcoes.h` para armazenar as funções criadas:

funcoes.h

```
#ifndef FUNCOES_H
#define FUNCOES_H

void boasVindas();
void capturarNome(char nome[]);
int contadorAlunos();
void exibirMensagemFinal(char nome[], int contador);

#endif
```

main.c

```
#include <stdio.h>
#include "funcoes.h"

int main() {
    char nome[50];
    boasVindas();
    capturarNome(nome);
    int contador = contadorAlunos();
    exibirMensagemFinal(nome, contador);
    return 0;
}
```

Implementar o Código em Arquivo Externo

O aluno implementa as funções no arquivo `funcoes.c`:

funcoes.c

```
#include <stdio.h>
#include "funcoes.h"

void boasVindas() {
    printf("Bem-vindo ao Projeto da Sala de Aula!\n");
}

void capturarNome(char nome[]) {
    printf("Digite seu nome: ");
    scanf("%s", nome);
}

int contadorAlunos() {
    static int contador = 0;
    contador++;
    return contador;
}

void exibirMensagemFinal(char nome[], int contador) {
    printf("Olá, %s! Você é o aluno número %d.\n", nome,
contador);
}
```

Adicionar Controle de Erro para Nome

O aluno adiciona controle para garantir que o nome tenha no mínimo 2 caracteres:

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

void capturarNome(char nome[]) {
    do {
        printf("Digite seu nome (mínimo 2 caracteres): ");
        scanf("%s", nome);
    } while (strlen(nome) < 2);
}
```

Implementar Leitura de Múltiplos Alunos

O aluno adiciona a capacidade de processar múltiplos alunos, permitindo que o programa continue executando até que uma condição de saída seja atingida:

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50];
    int continuar = 1;

    while (continuar) {
        boasVindas();
        capturarNome(nome);
        int contador = contadorAlunos();
```

```
        exibirMensagemFinal(nome, contador);

        printf("Deseja continuar? (1 - Sim / 0 - Não): ");
        scanf("%d", &continuar);
    }

    printf("Encerrando o programa...\n");
    return 0;
}
```

Gravar Informações em Arquivo

O aluno cria uma função para salvar o nome e número do aluno em um arquivo `alunos.txt`:

funcoes.h

```
void salvarAlunoNoArquivo(char nome[], int contador);
```

funcoes.c

```
void salvarAlunoNoArquivo(char nome[], int contador) {
    FILE *arquivo = fopen("alunos.txt", "a");
    if (arquivo != NULL) {
        fprintf(arquivo, "Aluno %d: %s\n", contador, nome);
        fclose(arquivo);
    } else {
        printf("Erro ao abrir o arquivo.\n");
    }
}
```

main.c

```c
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50];
    int continuar = 1;

    while (continuar) {
        boasVindas();
        capturarNome(nome);
        int contador = contadorAlunos();
        exibirMensagemFinal(nome, contador);
        salvarAlunoNoArquivo(nome, contador);

        printf("Deseja continuar? (1 - Sim / 0 - Não): ");
        scanf("%d", &continuar);
    }

    printf("Encerrando o programa...\n");
    return 0;
}
```

Ler Informações de Arquivo

O aluno implementa uma função para ler e exibir as informações gravadas no arquivo `alunos.txt`:

funcoes.h

```
void lerAlunosDoArquivo();
```

funcoes.c

```
void lerAlunosDoArquivo() {
    FILE *arquivo = fopen("alunos.txt", "r");
    char linha[100];

    if (arquivo != NULL) {
        printf("\nAlunos cadastrados:\n");
        while (fgets(linha, sizeof(linha), arquivo) != NULL) {
            printf("%s", linha);
        }
        fclose(arquivo);
    } else {
        printf("Nenhum aluno cadastrado ainda.\n");
    }
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50];
    int continuar = 1;

    while (continuar) {
```

```
        boasVindas();
        capturarNome(nome);
        int contador = contadorAlunos();
        exibirMensagemFinal(nome, contador);
        salvarAlunoNoArquivo(nome, contador);

        printf("Deseja continuar? (1 - Sim / 0 - Não): ");
        scanf("%d", &continuar);
    }

    printf("Deseja ver a lista de alunos cadastrados? (1 - Sim
/ 0 - Não): ");
    scanf("%d", &continuar);
    if (continuar) {
        lerAlunosDoArquivo();
    }

    printf("Encerrando o programa...\n");
    return 0;
}
```

Adicionar Controle de Erro no Arquivo

O aluno implementa um controle de erro para verificar se o arquivo pode ser lido e escrito corretamente:

funcoes.c

```
void salvarAlunoNoArquivo(char nome[], int contador) {
    FILE *arquivo = fopen("alunos.txt", "a");
    if (arquivo == NULL) {
        perror("Erro ao abrir o arquivo para gravação");
        return;
    }

    fprintf(arquivo, "Aluno %d: %s\n", contador, nome);
```

```
    fclose(arquivo);
}

void lerAlunosDoArquivo() {
    FILE *arquivo = fopen("alunos.txt", "r");
    if (arquivo == NULL) {
        perror("Erro ao abrir o arquivo para leitura");
        return;
    }

    char linha[100];
    printf("\nAlunos cadastrados:\n");
    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        printf("%s", linha);
    }
    fclose(arquivo);
}
```

Adicionar Função para Excluir Aluno

O aluno implementa a função para excluir um aluno do arquivo:

funcoes.h

```
void excluirAlunoDoArquivo(char nome[]);
```

funcoes.c

```
void excluirAlunoDoArquivo(char nome[]) {
    FILE *arquivo = fopen("alunos.txt", "r");
    FILE *temp = fopen("temp.txt", "w");
    char linha[100];
    int encontrado = 0;

    if (arquivo == NULL || temp == NULL) {
        perror("Erro ao abrir arquivo");
```

```
        return;
    }

    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        if (strstr(linha, nome) == NULL) {
            fputs(linha, temp);
        } else {
            encontrado = 1;
        }
    }

    fclose(arquivo);
    fclose(temp);

    remove("alunos.txt");
    rename("temp.txt", "alunos.txt");

    if (encontrado) {
        printf("Aluno %s removido com sucesso.\n", nome);
    } else {
        printf("Aluno %s não encontrado.\n", nome);
    }
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50];
    int continuar = 1;

    while (continuar) {
        boasVindas();
        capturarNome(nome);
        int contador = contadorAlunos();
```

```
        exibirMensagemFinal(nome, contador);
        salvarAlunoNoArquivo(nome, contador);

        printf("Deseja continuar? (1 - Sim / 0 - Não): ");
        scanf("%d", &continuar);
    }

    printf("Deseja ver a lista de alunos cadastrados? (1 - Sim
/ 0 - Não): ");
    scanf("%d", &continuar);
    if (continuar) {
        lerAlunosDoArquivo();
    }

    printf("Deseja excluir um aluno? (1 - Sim / 0 - Não): ");
    scanf("%d", &continuar);
    if (continuar) {
        printf("Digite o nome do aluno a ser excluído: ");
        scanf("%s", nome);
        excluirAlunoDoArquivo(nome);
    }

    printf("Encerrando o programa...\n");
    return 0;
}
```

Adicionar Função para Atualizar Dados do Aluno

O aluno cria uma função para atualizar o nome de um aluno no arquivo.

funcoes.h

```
void atualizarAlunoNoArquivo(char nomeAntigo[], char
nomeNovo[]);
```

funcoes.c

```
void atualizarAlunoNoArquivo(char nomeAntigo[], char
nomeNovo[]) {
    FILE *arquivo = fopen("alunos.txt", "r");
    FILE *temp = fopen("temp.txt", "w");
    char linha[100];
    int encontrado = 0;

    if (arquivo == NULL || temp == NULL) {
        perror("Erro ao abrir arquivo");
        return;
    }

    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        if (strstr(linha, nomeAntigo) != NULL) {
            fprintf(temp, "Aluno: %s\n", nomeNovo);
            encontrado = 1;
        } else {
            fputs(linha, temp);
        }
    }

    fclose(arquivo);
    fclose(temp);
```

```
    remove("alunos.txt");
    rename("temp.txt", "alunos.txt");

    if (encontrado) {
        printf("Aluno %s atualizado com sucesso para %s.\n",
nomeAntigo, nomeNovo);
    } else {
        printf("Aluno %s não encontrado.\n", nomeAntigo);
    }
}
```

Adicionar Menu Interativo

O aluno vai adicionar um menu interativo para que o usuário possa escolher diferentes operações: cadastrar, listar, excluir ou atualizar alunos.

funcoes.h

```
void mostrarMenu();
```

funcoes.c

```
void mostrarMenu() {
    printf("\nMenu:\n");
    printf("1 - Cadastrar novo aluno\n");
    printf("2 - Listar alunos\n");
    printf("3 - Excluir aluno\n");
    printf("4 - Atualizar aluno\n");
    printf("0 - Sair\n");
    printf("Escolha uma opção: ");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"
```

```
int main() {
    char nome[50], novoNome[50];
    int opcao, contador;

    do {
        mostrarMenu();
        scanf("%d", &opcao);

        switch (opcao) {
            case 1:
                boasVindas();
                capturarNome(nome);
                contador = contadorAlunos();
                exibirMensagemFinal(nome, contador);
                salvarAlunoNoArquivo(nome, contador);
                break;

            case 2:
                lerAlunosDoArquivo();
                break;

            case 3:
                printf("Digite o nome do aluno a ser excluído:
");
                scanf("%s", nome);
                excluirAlunoDoArquivo(nome);
                break;

            case 4:
                printf("Digite o nome do aluno a ser
atualizado: ");
                scanf("%s", nome);
                printf("Digite o novo nome: ");
                scanf("%s", novoNome);
                atualizarAlunoNoArquivo(nome, novoNome);
                break;

            case 0:
```

```
                printf("Encerrando o programa...\n");
                break;

            default:
                printf("Opção inválida! Tente novamente.\n");
        }
    } while (opcao != 0);

    return 0;
}
```

Adicionar Validações no Menu

O aluno vai implementar validações para garantir que o usuário insira opções válidas no menu e para tratar entradas incorretas.

funcoes.c

```
void mostrarMenu() {
    printf("\nMenu:\n");
    printf("1 - Cadastrar novo aluno\n");
    printf("2 - Listar alunos\n");
    printf("3 - Excluir aluno\n");
    printf("4 - Atualizar aluno\n");
    printf("0 - Sair\n");
    printf("Escolha uma opção: ");
}

int lerOpcao() {
    int opcao;
    if (scanf("%d", &opcao) != 1) {
        while (getchar() != '\n');  // Limpa o buffer
        return -1;  // Retorna valor inválido
```

```
    }
    return opcao;
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], novoNome[50];
    int opcao, contador;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 1:
                boasVindas();
                capturarNome(nome);
                contador = contadorAlunos();
                exibirMensagemFinal(nome, contador);
                salvarAlunoNoArquivo(nome, contador);
                break;

            case 2:
                lerAlunosDoArquivo();
                break;

            case 3:
```

```
                printf("Digite o nome do aluno a ser excluído:
");
                scanf("%s", nome);
                excluirAlunoDoArquivo(nome);
                break;

            case 4:
                printf("Digite o nome do aluno a ser
atualizado: ");
                scanf("%s", nome);
                printf("Digite o novo nome: ");
                scanf("%s", novoNome);
                atualizarAlunoNoArquivo(nome, novoNome);
                break;

            case 0:
                printf("Encerrando o programa...\n");
                break;

            default:
                printf("Opção inválida! Tente novamente.\n");
        }
    } while (opcao != 0);

    return 0;
}
```

Melhorar Tratamento de Arquivos

O aluno vai melhorar o tratamento de arquivos, assegurando que as operações de abertura e fechamento estejam seguras, evitando possíveis problemas de corrupção de dados.

funcoes.c

```
void salvarAlunoNoArquivo(char nome[], int contador) {
    FILE *arquivo = fopen("alunos.txt", "a");
    if (arquivo == NULL) {
        perror("Erro ao abrir o arquivo para gravação");
        return;
    }

    if (fprintf(arquivo, "Aluno %d: %s\n", contador, nome) <
0) {
        perror("Erro ao escrever no arquivo");
    }

    if (fclose(arquivo) != 0) {
        perror("Erro ao fechar o arquivo");
    }
}

void lerAlunosDoArquivo() {
    FILE *arquivo = fopen("alunos.txt", "r");
    if (arquivo == NULL) {
        perror("Erro ao abrir o arquivo para leitura");
        return;
    }

    char linha[100];
    printf("\nAlunos cadastrados:\n");
    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        printf("%s", linha);
    }
```

```
    if (fclose(arquivo) != 0) {
        perror("Erro ao fechar o arquivo");
    }
}
```

Melhorar a Interface de Usuário

O aluno vai melhorar a interface de usuário, tornando as mensagens mais claras e visuais:

funcoes.c

```
void mostrarMenu() {
    printf("\n============================\n");
    printf("        MENU PRINCIPAL       \n");
    printf("============================\n");
    printf("1 - Cadastrar novo aluno\n");
    printf("2 - Listar alunos\n");
    printf("3 - Excluir aluno\n");
    printf("4 - Atualizar aluno\n");
    printf("0 - Sair\n");
    printf("============================\n");
    printf("Escolha uma opção: ");
}

void boasVindas() {
    printf("\n**************************\n");
    printf("** Bem-vindo ao projeto! **\n");
    printf("**************************\n\n");
}

void exibirMensagemFinal(char nome[], int contador) {
    printf("\n**************************\n");
    printf("Olá, %s! Você é o aluno número %d.\n", nome,
contador);
```

```
    printf("****************************\n\n");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], novoNome[50];
    int opcao, contador;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 1:
                boasVindas();
                capturarNome(nome);
                contador = contadorAlunos();
                exibirMensagemFinal(nome, contador);
                salvarAlunoNoArquivo(nome, contador);
                break;

            case 2:
                lerAlunosDoArquivo();
                break;

            case 3:
```

```
                printf("Digite o nome do aluno a ser excluído:
");
                scanf("%s", nome);
                excluirAlunoDoArquivo(nome);
                break;

            case 4:
                printf("Digite o nome do aluno a ser
atualizado: ");
                scanf("%s", nome);
                printf("Digite o novo nome: ");
                scanf("%s", novoNome);
                atualizarAlunoNoArquivo(nome, novoNome);
                break;

            case 0:
                printf("Encerrando o programa...\n");
                break;

            default:
                printf("Opção inválida! Tente novamente.\n");
        }
    } while (opcao != 0);

    return 0;
}
```

Chegou até aqui: envie no email:
matheus.araujo.ecomp@gmail.com
Coloque seu nome no assunto

Parabéns

Se você for o primeiro a enviar o email nessa rodada, você ganhará um brinde, se o código estiver correto e funcionando.

Se quiser pode ir fazer o projeto.

A empresa código agradece seu empenho!

Você quer continuar?
Quer ir para segunda rodada.
Boa sorte!

Segunda Rodada

(Segunda Rodada) - Implementar Função de Backup do Arquivo

O aluno implementa uma função para criar um backup do arquivo de alunos, salvando uma cópia chamada `backup_alunos.txt` antes de qualquer modificação:

funcoes.h

```
void criarBackup();
```

funcoes.c

```
void criarBackup() {
    FILE *arquivo = fopen("alunos.txt", "r");
    FILE *backup = fopen("backup_alunos.txt", "w");
    char linha[100];

    if (arquivo == NULL || backup == NULL) {
        perror("Erro ao criar o backup");
```

```
        return;
    }

    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        fputs(linha, backup);
    }

    fclose(arquivo);
    fclose(backup);
    printf("Backup criado com sucesso.\n");
}
```

(Segunda Rodada) - Chamar Função de Backup ao Excluir Aluno

O aluno altera o fluxo para que, antes de excluir um aluno, o sistema faça automaticamente um backup:

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], novoNome[50];
    int opcao, contador;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }
```

```
        switch (opcao) {
            case 1:
                boasVindas();
                capturarNome(nome);
                contador = contadorAlunos();
                exibirMensagemFinal(nome, contador);
                salvarAlunoNoArquivo(nome, contador);
                break;

            case 2:
                lerAlunosDoArquivo();
                break;

            case 3:
                criarBackup();  // Chama a função de backup
antes de excluir
                printf("Digite o nome do aluno a ser excluído:
");
                scanf("%s", nome);
                excluirAlunoDoArquivo(nome);
                break;

            case 4:
                printf("Digite o nome do aluno a ser
atualizado: ");
                scanf("%s", nome);
                printf("Digite o novo nome: ");
                scanf("%s", novoNome);
                atualizarAlunoNoArquivo(nome, novoNome);
                break;

            case 0:
                printf("Encerrando o programa...\n");
                break;

            default:
                printf("Opção inválida! Tente novamente.\n");
```

```
        }
    } while (opcao != 0);

    return 0;
}
```

(Segunda Rodada) - Implementar Função para Restaurar o Backup

O aluno adiciona uma função para restaurar o arquivo original a partir do backup, permitindo reverter mudanças caso algo dê errado.

funcoes.h

```
void restaurarBackup();
```

funcoes.c

```
void restaurarBackup() {
    FILE *backup = fopen("backup_alunos.txt", "r");
    FILE *arquivo = fopen("alunos.txt", "w");
    char linha[100];

    if (backup == NULL || arquivo == NULL) {
        perror("Erro ao restaurar o backup");
        return;
    }

    while (fgets(linha, sizeof(linha), backup) != NULL) {
        fputs(linha, arquivo);
    }

    fclose(backup);
    fclose(arquivo);
    printf("Backup restaurado com sucesso.\n");
}
```

(Segunda Rodada) - Adicionar Opção para Restaurar Backup no Menu

O aluno adiciona a opção no menu para restaurar o backup quando necessário.

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], novoNome[50];
    int opcao, contador;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 1:
                boasVindas();
                capturarNome(nome);
                contador = contadorAlunos();
                exibirMensagemFinal(nome, contador);
                salvarAlunoNoArquivo(nome, contador);
                break;

            case 2:
```

```
                lerAlunosDoArquivo();
                break;

            case 3:
                criarBackup();
                printf("Digite o nome do aluno a ser excluído:
");
                scanf("%s", nome);
                excluirAlunoDoArquivo(nome);
                break;

            case 4:
                printf("Digite o nome do aluno a ser
atualizado: ");
                scanf("%s", nome);
                printf("Digite o novo nome: ");
                scanf("%s", novoNome);
                atualizarAlunoNoArquivo(nome, novoNome);
                break;

            case 5:  // Nova opção para restaurar backup
                restaurarBackup();
                break;

            case 0:
                printf("Encerrando o programa...\n");
                break;

            default:
                printf("Opção inválida! Tente novamente.\n");
        }
    } while (opcao != 0);

    return 0;
}
```

(Segunda Rodada) - Implementar Função de Pesquisa de Aluno

O aluno cria uma função para pesquisar alunos pelo nome dentro do arquivo, retornando se o aluno está ou não registrado.

funcoes.h

```
int pesquisarAluno(char nome[]);
```

funcoes.c

```
int pesquisarAluno(char nome[]) {
    FILE *arquivo = fopen("alunos.txt", "r");
    char linha[100];

    if (arquivo == NULL) {
        perror("Erro ao abrir o arquivo");
        return 0;
    }

    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        if (strstr(linha, nome) != NULL) {
            fclose(arquivo);
            return 1;  // Aluno encontrado
        }
    }

    fclose(arquivo);
    return 0;  // Aluno não encontrado
}
```

(Segunda Rodada) - Adicionar Pesquisa no Menu

O aluno adiciona a opção de pesquisa no menu, permitindo ao usuário procurar por um aluno.

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], novoNome[50];
    int opcao, contador;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 1:
                boasVindas();
                capturarNome(nome);
                contador = contadorAlunos();
                exibirMensagemFinal(nome, contador);
                salvarAlunoNoArquivo(nome, contador);
                break;

            case 2:
                lerAlunosDoArquivo();
                break;

            case 3:
                criarBackup();
```

```
                printf("Digite o nome do aluno a ser excluído:
");
                scanf("%s", nome);
                excluirAlunoDoArquivo(nome);
                break;

            case 4:
                printf("Digite o nome do aluno a ser
atualizado: ");
                scanf("%s", nome);
                printf("Digite o novo nome: ");
                scanf("%s", novoNome);
                atualizarAlunoNoArquivo(nome, novoNome);
                break;

            case 5:
                restaurarBackup();
                break;

            case 6:  // Nova opção para pesquisar aluno
                printf("Digite o nome do aluno para pesquisar:
");
                scanf("%s", nome);
                if (pesquisarAluno(nome)) {
                    printf("Aluno %s encontrado.\n", nome);
                } else {
                    printf("Aluno %s não encontrado.\n",
nome);
                }
                break;

            case 0:
                printf("Encerrando o programa...\n");
                break;

            default:
                printf("Opção inválida! Tente novamente.\n");
        }
```

```
    } while (opcao != 0);

    return 0;
}
```

(Segunda Rodada) - Adicionar Validação ao Atualizar Aluno

O aluno implementa uma validação ao atualizar o aluno, para verificar se o aluno existe antes de permitir a alteração:

funcoes.c

```
void atualizarAlunoNoArquivo(char nomeAntigo[], char
nomeNovo[]) {
    if (!pesquisarAluno(nomeAntigo)) {
        printf("Aluno %s não encontrado para atualizar.\n",
nomeAntigo);
        return;
    }

    FILE *arquivo = fopen("alunos.txt", "r");
    FILE *temp = fopen("temp.txt", "w");
    char linha[100];
    int encontrado = 0;

    if (arquivo == NULL || temp == NULL) {
        perror("Erro ao abrir arquivo");
        return;
    }

    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        if (strstr(linha, nomeAntigo) != NULL) {
            fprintf(temp, "Aluno %d: %s\n", contadorAlunos(),
nomeNovo);
```

```
            encontrado = 1;
        } else {
            fputs(linha, temp);
        }
    }

    fclose(arquivo);
    fclose(temp);

    remove("alunos.txt");
    rename("temp.txt", "alunos.txt");

    if (encontrado) {
        printf("Aluno %s atualizado com sucesso.\n",
nomeAntigo);
    }
}
```

(Segunda Rodada) - Adicionar Contador de Alunos por Gênero

O aluno adiciona uma função para contar quantos alunos do gênero masculino e feminino estão registrados, com base em uma variável adicional de gênero.

funcoes.h

```
void contarAlunosPorGenero();
```

funcoes.c

```
void contarAlunosPorGenero() {
    FILE *arquivo = fopen("alunos.txt", "r");
    char linha[100];
    int masc = 0, fem = 0;

    if (arquivo == NULL) {
```

```
        perror("Erro ao abrir o arquivo");
        return;
    }

    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        if (strstr(linha, "Masculino") != NULL) {
            masc++;
        } else if (strstr(linha, "Feminino") != NULL) {
            fem++;
        }
    }

    fclose(arquivo);
    printf("Alunos masculinos: %d\n", masc);
    printf("Alunas femininas: %d\n", fem);
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], novoNome[50];
    int opcao, contador;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
```

```
            case 1:
                boasVindas();
                capturarNome(nome);
                contador = contadorAlunos();
                exibirMensagemFinal(nome, contador);
                salvarAlunoNoArquivo(nome, contador);
                break;

            case 2:
                lerAlunosDoArquivo();
                break;

            case 3:
                criarBackup();
                printf("Digite o nome do aluno a ser excluído:
");
                scanf("%s", nome);
                excluirAlunoDoArquivo(nome);
                break;

            case 4:
                printf("Digite o nome do aluno a ser
atualizado: ");
                scanf("%s", nome);
                printf("Digite o novo nome: ");
                scanf("%s", novoNome);
                atualizarAlunoNoArquivo(nome, novoNome);
                break;

            case 5:
                restaurarBackup();
                break;

            case 6:
                printf("Digite o nome do aluno para pesquisar:
");
                scanf("%s", nome);
                if (pesquisarAluno(nome)) {
```

```
                printf("Aluno %s encontrado.\n", nome);
            } else {
                printf("Aluno %s não encontrado.\n",
nome);
            }
            break;

        case 7:  // Nova opção para contar alunos por
gênero
            contarAlunosPorGenero();
            break;

        case 0:
            printf("Encerrando o programa...\n");
            break;

        default:
            printf("Opção inválida! Tente novamente.\n");
        }
    } while (opcao != 0);

    return 0;
}
```

(Segunda Rodada) - Capturar Gênero do Aluno ao Cadastrar

O aluno ajusta a função de cadastro para capturar o gênero do aluno (masculino ou feminino) e salvar essa informação no arquivo.

funcoes.h

```
void capturarGenero(char genero[]);
```

funcoes.c

```
void capturarGenero(char genero[]) {
    printf("Digite o gênero do aluno (Masculino/Feminino): ");
    scanf("%s", genero);
}

void salvarAlunoNoArquivo(char nome[], char genero[], int
contador) {
    FILE *arquivo = fopen("alunos.txt", "a");
    if (arquivo == NULL) {
        perror("Erro ao abrir o arquivo para gravação");
        return;
    }

    if (fprintf(arquivo, "Aluno %d: %s (%s)\n", contador,
nome, genero) < 0) {
        perror("Erro ao escrever no arquivo");
    }

    fclose(arquivo);
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
```

```
    char nome[50], genero[20], novoNome[50];
    int opcao, contador;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 1:
                boasVindas();
                capturarNome(nome);
                capturarGenero(genero);  // Captura o gênero
do aluno
                contador = contadorAlunos();
                exibirMensagemFinal(nome, contador);
                salvarAlunoNoArquivo(nome, genero, contador);
                break;

            case 2:
                lerAlunosDoArquivo();
                break;

            case 3:
                criarBackup();
                printf("Digite o nome do aluno a ser excluído:
");
                scanf("%s", nome);
                excluirAlunoDoArquivo(nome);
                break;

            case 4:
```

```
                printf("Digite o nome do aluno a ser
atualizado: ");
                scanf("%s", nome);
                printf("Digite o novo nome: ");
                scanf("%s", novoNome);
                atualizarAlunoNoArquivo(nome, novoNome);
                break;

            case 5:
                restaurarBackup();
                break;

            case 6:
                printf("Digite o nome do aluno para pesquisar:
");
                scanf("%s", nome);
                if (pesquisarAluno(nome)) {
                    printf("Aluno %s encontrado.\n", nome);
                } else {
                    printf("Aluno %s não encontrado.\n");
                }
                break;

            case 7:
                contarAlunosPorGenero();
                break;

            case 0:
                printf("Encerrando o programa...\n");
                break;

            default:
                printf("Opção inválida! Tente novamente.\n");
        }
    } while (opcao != 0);

    return 0;
}
```

(Segunda Rodada) - Implementar Função para Ordenar Alunos

O aluno adiciona uma função para ordenar os alunos alfabeticamente e mostrar a lista ordenada.

funcoes.h

```
void ordenarAlunos();
```

funcoes.c

```
void ordenarAlunos() {
    FILE *arquivo = fopen("alunos.txt", "r");
    char alunos[100][100];
    char linha[100];
    int i = 0, j;

    if (arquivo == NULL) {
        perror("Erro ao abrir o arquivo");
        return;
    }

    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        strcpy(alunos[i], linha);
        i++;
    }
    fclose(arquivo);

    for (int k = 0; k < i - 1; k++) {
        for (int l = k + 1; l < i; l++) {
            if (strcmp(alunos[k], alunos[l]) > 0) {
                char temp[100];
                strcpy(temp, alunos[k]);
```

```
                strcpy(alunos[k], alunos[l]);
                strcpy(alunos[l], temp);
            }
        }
    }

    printf("\nAlunos ordenados:\n");
    for (j = 0; j < i; j++) {
        printf("%s", alunos[j]);
    }
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], genero[20], novoNome[50];
    int opcao, contador;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 1:
                boasVindas();
                capturarNome(nome);
                capturarGenero(genero);
                contador = contadorAlunos();
```

```
                exibirMensagemFinal(nome, contador);
                salvarAlunoNoArquivo(nome, genero, contador);
                break;

            case 2:
                lerAlunosDoArquivo();
                break;

            case 3:
                criarBackup();
                printf("Digite o nome do aluno a ser excluído:
");
                scanf("%s", nome);
                excluirAlunoDoArquivo(nome);
                break;

            case 4:
                printf("Digite o nome do aluno a ser
atualizado: ");
                scanf("%s", nome);
                printf("Digite o novo nome: ");
                scanf("%s", novoNome);
                atualizarAlunoNoArquivo(nome, novoNome);
                break;

            case 5:
                restaurarBackup();
                break;

            case 6:
                printf("Digite o nome do aluno para pesquisar:
");
                scanf("%s", nome);
                if (pesquisarAluno(nome)) {
                    printf("Aluno %s encontrado.\n", nome);
                } else {
                    printf("Aluno %s não encontrado.\n");
                }
```

```
                break;

            case 7:
                contarAlunosPorGenero();
                break;

            case 8:  // Nova opção para ordenar alunos
                ordenarAlunos();
                break;

            case 0:
                printf("Encerrando o programa...\n");
                break;

            default:
                printf("Opção inválida! Tente novamente.\n");
        }
    } while (opcao != 0);

    return 0;
}
```

(Segunda Rodada) - Implementar Validação para Nome Vazio

O aluno adiciona uma validação para impedir que um nome vazio seja registrado, evitando entradas inválidas no arquivo.

funcoes.h

```
int validarNome(char nome[]);
```

funcoes.c

```
int validarNome(char nome[]) {
    if (strlen(nome) == 0) {
        printf("Nome não pode estar vazio. Tente
novamente.\n");
        return 0;
    }
    return 1;
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], genero[20], novoNome[50];
    int opcao, contador;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
```

```
        }

        switch (opcao) {
            case 1:
                boasVindas();
                capturarNome(nome);
                if (validarNome(nome)) {
                    capturarGenero(genero);
                    contador = contadorAlunos();
                    exibirMensagemFinal(nome, contador);
                    salvarAlunoNoArquivo(nome, genero,
contador);
                }
                break;

            // Demais opções inalteradas
        }
    } while (opcao != 0);

    return 0;
}
```

(Segunda Rodada) - Implementar Função para Excluir Alunos por Gênero

O aluno implementa uma função para excluir todos os alunos de um gênero específico.

funcoes.h

```
void excluirAlunosPorGenero(char genero[]);
```

funcoes.c

```
void excluirAlunosPorGenero(char genero[]) {
    FILE *arquivo = fopen("alunos.txt", "r");
    FILE *temp = fopen("temp.txt", "w");
    char linha[100];

    if (arquivo == NULL || temp == NULL) {
        perror("Erro ao abrir arquivo");
        return;
    }

    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        if (strstr(linha, genero) == NULL) {
            fputs(linha, temp);
        }
    }

    fclose(arquivo);
    fclose(temp);

    remove("alunos.txt");
    rename("temp.txt", "alunos.txt");

    printf("Alunos do gênero %s excluídos com sucesso.\n",
genero);
}
```

main.c

```
#include <stdio.h>
```

```
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], genero[20], novoNome[50];
    int opcao, contador;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 9:  // Nova opção para excluir alunos por
gênero
                printf("Digite o gênero dos alunos a serem
excluídos: ");
                scanf("%s", genero);
                excluirAlunosPorGenero(genero);
                break;

            // Demais opções inalteradas
        }
    } while (opcao != 0);

    return 0;
}
```

(Segunda Rodada) - Implementar Limpeza Automática de Arquivo Vazio

O aluno implementa uma função que exclui automaticamente o arquivo de alunos caso todos tenham sido removidos.

funcoes.h

```
void limparArquivoSeVazio();
```

funcoes.c

```
void limparArquivoSeVazio() {
    FILE *arquivo = fopen("alunos.txt", "r");

    if (arquivo == NULL) {
        perror("Erro ao abrir o arquivo");
        return;
    }

    fseek(arquivo, 0, SEEK_END);
    if (ftell(arquivo) == 0) {  // Verifica se o arquivo está
vazio
        fclose(arquivo);
        remove("alunos.txt");
        printf("Arquivo 'alunos.txt' removido, pois está
vazio.\n");
    } else {
        fclose(arquivo);
    }
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"
```

```
int main() {
    char nome[50], genero[20], novoNome[50];
    int opcao, contador;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 9:
                printf("Digite o gênero dos alunos a serem
excluídos: ");
                scanf("%s", genero);
                excluirAlunosPorGenero(genero);
                limparArquivoSeVazio();  // Nova
funcionalidade
                break;

            // Demais opções inalteradas
        }
    } while (opcao != 0);

    return 0;
}
```

(Segunda Rodada) - Adicionar Função para Contagem Total de Alunos

O aluno adiciona uma função que exibe a contagem total de alunos registrados.

funcoes.h

```
int contarTotalAlunos();
```

funcoes.c

```
int contarTotalAlunos() {
    FILE *arquivo = fopen("alunos.txt", "r");
    char linha[100];
    int contador = 0;

    if (arquivo == NULL) {
        perror("Erro ao abrir o arquivo");
        return 0;
    }

    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        contador++;
    }

    fclose(arquivo);
    return contador;
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], genero[20], novoNome[50];
    int opcao, contador;
```

```
    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 10:  // Nova opção para contar o total de
alunos
                contador = contarTotalAlunos();
                printf("Total de alunos registrados: %d\n",
contador);
                break;

            // Demais opções inalteradas
        }
    } while (opcao != 0);

    return 0;
}
```

(Segunda Rodada) - Implementar Função para Verificar Backup Antes de Excluir

O aluno adiciona uma verificação para garantir que o backup seja restaurado antes de excluir o arquivo principal.

funcoes.h

```
int verificarBackup();
```

funcoes.c

```
int verificarBackup() {
    FILE *backup = fopen("backup_alunos.txt", "r");

    if (backup != NULL) {
        fclose(backup);
        return 1;  // Backup existente
    }

    return 0;  // Backup inexistente
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], genero[20], novoNome[50];
    int opcao, contador;

    do {
        mostrarMenu();
        opcao = lerOpcao();
```

```
        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 3:
                if (verificarBackup()) {
                    printf("Restaurando backup antes de
excluir aluno...\n");
                    restaurarBackup();
                }
                printf("Digite o nome do aluno a ser excluído:
");
                scanf("%s", nome);
                excluirAlunoDoArquivo(nome);
                break;

            // Demais opções inalteradas
        }
    } while (opcao != 0);

    return 0;
}
```

(Segunda Rodada) - Implementar Log de Ações no Sistema

O aluno adiciona um log para registrar as ações realizadas no sistema, como adicionar, excluir e atualizar alunos.

funcoes.h

```
void registrarLog(char acao[], char detalhes[]);
```

funcoes.c

```
void registrarLog(char acao[], char detalhes[]) {
    FILE *log = fopen("log_sistema.txt", "a");

    if (log == NULL) {
        perror("Erro ao abrir o log");
        return;
    }

    fprintf(log, "Ação: %s - Detalhes: %s\n", acao, detalhes);
    fclose(log);
}
```

main.c

```
#include <stdio.h>
```

```
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], genero[20], novoNome[50];
    int opcao, contador;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 1:
                boasVindas();
                capturarNome(nome);
                if (validarNome(nome)) {
                    capturarGenero(genero);
```

```
                    contador = contadorAlunos();
                    exibirMensagemFinal(nome, contador);
                    salvarAlunoNoArquivo(nome, genero,
contador);
                    registrarLog("Adição", nome);  // Registro
de log
                }
                break;

            case 3:
                printf("Digite o nome do aluno a ser excluído:
");
                scanf("%s", nome);
                excluirAlunoDoArquivo(nome);
                registrarLog("Exclusão", nome);  // Registro
de log
                break;

            case 4:
                printf("Digite o nome do aluno a ser
atualizado: ");
                scanf("%s", nome);
                printf("Digite o novo nome: ");
                scanf("%s", novoNome);
                atualizarAlunoNoArquivo(nome, novoNome);
```

```
                registrarLog("Atualização", nome);  //
Registro de log
                break;


            // Demais opções inalteradas
        }
    } while (opcao != 0);


    return 0;
}
```

(Segunda Rodada) - Gerar Relatório Completo em Arquivo

O aluno implementa uma função para gerar um relatório completo dos alunos, salvando os dados em um arquivo separado.

funcoes.h

```
void gerarRelatorioCompleto();
```

funcoes.c

```
void gerarRelatorioCompleto() {
    FILE *arquivo = fopen("alunos.txt", "r");
    FILE *relatorio = fopen("relatorio_completo.txt", "w");
    char linha[100];
```

```
    if (arquivo == NULL || relatorio == NULL) {
        perror("Erro ao abrir arquivo");
        return;
    }

    fprintf(relatorio, "Relatório Completo dos Alunos\n\n");

    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        fputs(linha, relatorio);
    }

    fclose(arquivo);
    fclose(relatorio);

    printf("Relatório completo gerado com sucesso!\n");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"
```

```
int main() {
    char nome[50], genero[20];
    int opcao;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 11:  // Nova opção para gerar relatório
completo
                gerarRelatorioCompleto();
                break;

            // Demais opções inalteradas
        }
    } while (opcao != 0);
```

```
    return 0;
}
```

(Segunda Rodada) - Implementar Busca por Parte do Nome

O aluno adiciona uma funcionalidade para buscar alunos a partir de uma parte do nome, exibindo os resultados.

funcoes.h

```
void buscarAlunoPorParteNome(char parte[]);
```

funcoes.c

```
void buscarAlunoPorParteNome(char parte[]) {
    FILE *arquivo = fopen("alunos.txt", "r");
    char linha[100];
    int encontrou = 0;

    if (arquivo == NULL) {
        perror("Erro ao abrir o arquivo");
        return;
    }

    printf("Resultados da busca:\n");
```

```
    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        if (strstr(linha, parte) != NULL) {
            printf("%s", linha);
            encontrou = 1;
        }
    }

    if (!encontrou) {
        printf("Nenhum aluno encontrado com a parte '%s' no
nome.\n", parte);
    }

    fclose(arquivo);
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], genero[20], parteNome[50];
    int opcao;
```

```
    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 12:  // Nova opção para buscar aluno por
parte do nome
                printf("Digite parte do nome para buscar: ");
                scanf("%s", parteNome);
                buscarAlunoPorParteNome(parteNome);
                break;

            // Demais opções inalteradas
        }
    } while (opcao != 0);

    return 0;
```

```
}
```

(Segunda Rodada) - Implementar Exportação dos Dados em Formato CSV

O aluno adiciona uma função para exportar os dados dos alunos em formato CSV.

funcoes.h

```
void exportarParaCSV();
```

funcoes.c

```
void exportarParaCSV() {
    FILE *arquivo = fopen("alunos.txt", "r");
    FILE *csv = fopen("alunos_exportados.csv", "w");
    char linha[100];

    if (arquivo == NULL || csv == NULL) {
        perror("Erro ao abrir arquivo");
        return;
    }

    fprintf(csv, "Nome, Gênero\n");

    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
```

```
        char nome[50], genero[20];
        sscanf(linha, "%[^,], %s", nome, genero);
        fprintf(csv, "%s, %s\n", nome, genero);
    }

    fclose(arquivo);
    fclose(csv);

    printf("Dados exportados para alunos_exportados.csv com
sucesso!\n");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], genero[20];
    int opcao;

    do {
        mostrarMenu();
        opcao = lerOpcao();
```

```
        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 13:  // Nova opção para exportar para CSV
                exportarParaCSV();
                break;

            // Demais opções inalteradas
        }
    } while (opcao != 0);

    return 0;
}
```

(Segunda Rodada) - Adicionar Função para Alterar Gênero de Aluno

O aluno cria uma função que permite modificar o gênero de um aluno já existente no sistema.

funcoes.h

```
void alterarGeneroAluno(char nome[], char novoGenero[]);
```

funcoes.c

```
void alterarGeneroAluno(char nome[], char novoGenero[]) {
    FILE *arquivo = fopen("alunos.txt", "r");
    FILE *temp = fopen("temp.txt", "w");
    char linha[100];
    int alterado = 0;

    if (arquivo == NULL || temp == NULL) {
        perror("Erro ao abrir arquivo");
        return;
    }

    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        char nomeAluno[50], genero[20];
        sscanf(linha, "%[^,], %s", nomeAluno, genero);

        if (strcmp(nomeAluno, nome) == 0) {
```

```
            fprintf(temp, "%s, %s\n", nomeAluno, novoGenero);
            alterado = 1;
        } else {
            fputs(linha, temp);
        }
    }
    fclose(arquivo);
    fclose(temp);
    remove("alunos.txt");
    rename("temp.txt", "alunos.txt");

    if (alterado) {
        printf("Gênero do aluno '%s' alterado para '%s'.\n",
nome, novoGenero);
    } else {
        printf("Aluno '%s' não encontrado.\n", nome);
    }
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"
```

```
int main() {
    char nome[50], genero[20], novoGenero[20];
    int opcao;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            case 14:  // Nova opção para alterar gênero
                printf("Digite o nome do aluno: ");
                scanf("%s", nome);
                printf("Digite o novo gênero: ");
                scanf("%s", novoGenero);
                alterarGeneroAluno(nome, novoGenero);
                break;
```

```
            // Demais opções inalteradas
        }
    } while (opcao != 0);

    return 0;
}
```

(Segunda Rodada) - Melhorar a Interface de Menu com Opções Numeradas

O aluno reformula a interface do menu, deixando as opções numeradas e mais claras.

funcoes.h

```
void mostrarMenu();
```

funcoes.c

```
void mostrarMenu() {
    printf("\n--- Menu do Sistema ---\n");
    printf("1 - Adicionar aluno\n");
    printf("2 - Listar todos os alunos\n");
    printf("3 - Excluir aluno\n");
    printf("4 - Atualizar nome de aluno\n");
    printf("5 - Excluir alunos por gênero\n");
```

```
    printf("6 - Contar total de alunos\n");
    printf("7 - Gerar relatório completo\n");
    printf("8 - Buscar aluno por parte do nome\n");
    printf("9 - Exportar dados para CSV\n");
    printf("10 - Alterar gênero de aluno\n");
    printf("0 - Sair\n");
    printf("Escolha uma opção: ");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], genero[20];
    int opcao;

    do {
        mostrarMenu();
        opcao = lerOpcao();

        if (opcao == -1) {
```

```
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
        }

        switch (opcao) {
            // Opções do menu já implementadas
        }
    } while (opcao != 0);

    return 0;
}
```

(Segunda Rodada) - Implementar Sistema de Senhas para Proteger Exclusão de Dados

O aluno implementa um sistema de senhas para proteger a exclusão de dados.

funcoes.h

```
int verificarSenha();
```

funcoes.c

```
int verificarSenha() {
    char senha[20];
    printf("Digite a senha de administrador: ");
    scanf("%s", senha);
```

```
    if (strcmp(senha, "admin123") == 0) {
        return 1;  // Senha correta
    } else {
        printf("Senha incorreta! Acesso negado.\n");
        return 0;  // Senha incorreta
    }
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char nome[50], genero[20];
    int opcao;
    do {
        mostrarMenu();
        opcao = lerOpcao();
        if (opcao == -1) {
            printf("Entrada inválida! Por favor, insira um
número.\n");
            continue;
```

```
        }
        switch (opcao) {
            case 3:  // Excluir aluno com senha
                if (verificarSenha()) {
                    printf("Digite o nome do aluno a ser
excluído: ");
                    scanf("%s", nome);
                    excluirAlunoDoArquivo(nome);
                }
                break;
            // Demais opções inalteradas
        }
    } while (opcao != 0);
    return 0;
}
```

Chegou até aqui: envie no email: matheus.araujo.ecomp@gmail.com

Coloque seu nome no assunto Parabéns

Se você for o primeiro a enviar nesta rodada e você ganha além do brinde.

Você quer continuar?
Quer ir para segunda Rodada Bônus.
Boa sorte!

Rodada Bônus

Refatorar a Estrutura de Funções

O código foi fragmentado e duplicado.

O primeiro especialista começa refatorando e organizando as funções básicas em arquivos de cabeçalho e de implementação.

funcoes.h

```
#ifndef FUNCOES_H

#define FUNCOES_H


void boasVindas();

void capturarNome(char nome[]);

void capturarGenero(char genero[]);

int validarNome(char nome[]);
```

```
void exibirMensagemFinal(char nome[], int contador);

void salvarAlunoNoArquivo(char nome[], char genero[], int
contador);

void excluirAlunoDoArquivo(char nome[]);

void atualizarAlunoNoArquivo(char nome[], char novoNome[]);

void listarAlunos();

int contadorAlunos();

void buscarAlunoPorParteNome(char parte[]);

void exportarParaCSV();

void alterarGeneroAluno(char nome[], char novoGenero[]);

void gerarRelatorioCompleto();

int verificarSenha();

void mostrarMenu();

int lerOpcao();

void registrarLog(char operacao[], char nome[]);


#endif
```

funcoes.c (Refatorado)

```
#include <stdio.h>

#include <string.h>

#include "funcoes.h"
```

```
// Definições das funções (mantidas conforme a implementação anterior)
// Funções foram movidas e organizadas em seus devidos lugares

void boasVindas() {
    printf("Bem-vindo ao Sistema de Cadastro de Alunos!\n");
}

void capturarNome(char nome[]) {
    printf("Digite o nome do aluno: ");
    scanf("%s", nome);
}

void capturarGenero(char genero[]) {
    printf("Digite o gênero do aluno: ");
    scanf("%s", genero);
}

int validarNome(char nome[]) {
    if (strlen(nome) < 2) {
        printf("Nome inválido!\n");
        return 0;
    }
```

```
    return 1;
}

// E assim sucessivamente para as outras funções...
```

Criar Login com Múltiplos Usuários

Para melhorar a segurança, o especialista adiciona um sistema de login com múltiplos usuários.

funcoes.h

```
int login();
```

funcoes.c

```
#include <stdio.h>
#include <string.h>

int login() {
    char username[50], password[50];

    printf("Digite o nome de usuário: ");
    scanf("%s", username);
    printf("Digite a senha: ");
```

```
    scanf("%s", password);

    if (strcmp(username, "admin") == 0 && strcmp(password,
"admin123") == 0) {
        printf("Login bem-sucedido!\n");
        return 1;
    } else {
        printf("Credenciais inválidas!\n");
        return 0;
    }
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    if (!login()) {
        printf("Acesso negado.\n");
        return 0;
    }

    // Restante do programa aqui
```

```
    // menu e funcionalidades conforme antes
    int opcao;
    char nome[50], genero[20];

    do {
        mostrarMenu();
        opcao = lerOpcao();
        switch (opcao) {
            case 1:
                boasVindas();
                capturarNome(nome);
                if (validarNome(nome)) {
                    capturarGenero(genero);
                    int contador = contadorAlunos();
                    exibirMensagemFinal(nome, contador);
                    salvarAlunoNoArquivo(nome, genero,
contador);
                    registrarLog("Adição", nome); // Registro
de log
                }
                break;

            case 2:
                listarAlunos();
```

```
                break;
            // Outras opções já implementadas
        }
    } while (opcao != 0);

    return 0;
}
```

Adicionar Relatórios Estatísticos

Este especialista é encarregado de implementar uma função que gera estatísticas sobre o número de alunos e o balanceamento de gêneros.

funcoes.h

```
void gerarRelatorioEstatistico();
```

funcoes.c

```
void gerarRelatorioEstatistico() {
    FILE *arquivo = fopen("alunos.txt", "r");
    char linha[100], genero[20];
    int totalAlunos = 0, masculino = 0, feminino = 0, outros = 0;

    if (arquivo == NULL) {
```

```
        perror("Erro ao abrir arquivo");
        return;
    }

    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        sscanf(linha, "%*[^,], %s", genero);
        totalAlunos++;
        if (strcmp(genero, "Masculino") == 0) masculino++;
        else if (strcmp(genero, "Feminino") == 0) feminino++;
        else outros++;
    }

    fclose(arquivo);

    printf("\n--- Relatório Estatístico ---\n");
    printf("Total de alunos: %d\n", totalAlunos);
    printf("Masculino: %d\n", masculino);
    printf("Feminino: %d\n", feminino);
    printf("Outros: %d\n", outros);
}
```

main.c

```c
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    if (!login()) {
        printf("Acesso negado.\n");
        return 0;
    }

    int opcao;
    char nome[50], genero[20];

    do {
        mostrarMenu();
        opcao = lerOpcao();
        switch (opcao) {
            case 1:
                boasVindas();
                capturarNome(nome);
                if (validarNome(nome)) {
                    capturarGenero(genero);
                    int contador = contadorAlunos();
```

```
                    exibirMensagemFinal(nome, contador);
                    salvarAlunoNoArquivo(nome, genero,
contador);
                    registrarLog("Adição", nome);
                }
                break;

            case 2:
                listarAlunos();
                break;

            case 15: // Novo relatório estatístico
                gerarRelatorioEstatistico();
                break;
        }
    } while (opcao != 0);

    return 0;
}
```

Implementar Backup Automático dos Dados

Agora, o sistema terá uma função de backup automático toda vez que o programa é encerrado.

funcoes.h

```
void realizarBackup();
```

funcoes.c

```
#include <stdio.h>

void realizarBackup() {
    FILE *arquivoOrigem = fopen("alunos.txt", "r");
    FILE *arquivoBackup = fopen("backup_alunos.txt", "w");
    char linha[100];

    if (arquivoOrigem == NULL || arquivoBackup == NULL) {
        perror("Erro ao abrir arquivo");
        return;
    }

    while (fgets(linha, sizeof(linha), arquivoOrigem) != NULL)
{
        fputs(linha, arquivoBackup);
    }
```

```
    fclose(arquivoOrigem);
    fclose(arquivoBackup);

    printf("Backup realizado com sucesso!\n");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    if (!login()) {
        printf("Acesso negado.\n");
        return 0;
    }

    int opcao;
    char nome[50], genero[20];

    do {
```

```
        mostrarMenu();
        opcao = lerOpcao();
        switch (opcao) {
            case 1:
                boasVindas();
                capturarNome(nome);
                if (validarNome(nome)) {
                    capturarGenero(genero);
                    int contador = contadorAlunos();
                    exibirMensagemFinal(nome, contador);
                    salvarAlunoNoArquivo(nome, genero,
contador);
                    registrarLog("Adição", nome);
                }
                break;

            case 2:
                listarAlunos();
                break;

            case 15:
                gerarRelatorioEstatistico();
                break;
```

```
        }
    } while (opcao != 0);

    realizarBackup();
    return 0;
}
```

Implementar Restauração de Backup

O aluno implementa a função que permite restaurar os dados a partir de um arquivo de backup.

funcoes.h

```
void restaurarBackup();
```

funcoes.c

```
#include <stdio.h>

void restaurarBackup() {
    FILE *arquivoOrigem = fopen("backup_alunos.txt", "r");
    FILE *arquivoDestino = fopen("alunos.txt", "w");
    char linha[100];

    if (arquivoOrigem == NULL || arquivoDestino == NULL) {
        perror("Erro ao abrir arquivo");
```

```
        return;
    }

    while (fgets(linha, sizeof(linha), arquivoOrigem) != NULL)
{
        fputs(linha, arquivoDestino);
    }

    fclose(arquivoOrigem);
    fclose(arquivoDestino);

    printf("Backup restaurado com sucesso!\n");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    if (!login()) {
        printf("Acesso negado.\n");
        return 0;
    }
```

```
    int opcao;
    do {
        mostrarMenu();
        opcao = lerOpcao();
        switch (opcao) {
            case 1:
                // Outras funcionalidades...
                break;
            case 16:  // Restauração de backup
                restaurarBackup();
                break;
        }
    } while (opcao != 0);

    realizarBackup();
    return 0;
}
```

Adicionar Logs Detalhados de Ações

Este aluno cria um sistema de logs que detalha as ações realizadas no sistema.

funcoes.h

```
void registrarLog(char operacao[], char nome[]);
```

funcoes.c

```
#include <stdio.h>
#include <time.h>

void registrarLog(char operacao[], char nome[]) {
    FILE *arquivoLog = fopen("log.txt", "a");
    time_t agora;
    time(&agora);

    if (arquivoLog == NULL) {
        perror("Erro ao abrir arquivo de log");
        return;
    }

    fprintf(arquivoLog, "Operação: %s | Nome: %s | Data e
Hora: %s", operacao, nome, ctime(&agora));
    fclose(arquivoLog);
```

```
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    if (!login()) {
        printf("Acesso negado.\n");
        return 0;
    }

    int opcao;
    char nome[50], genero[20];
    do {
        mostrarMenu();
        opcao = lerOpcao();
        switch (opcao) {
            case 1:
                boasVindas();
                capturarNome(nome);
```

```
                capturarGenero(genero);
                salvarAlunoNoArquivo(nome, genero,
contadorAlunos());
                registrarLog("Adição", nome);  // Registrar a
operação
                break;
            case 2:
                listarAlunos();
                registrarLog("Listagem de alunos", "N/A");
                break;
            case 16:
                restaurarBackup();
                registrarLog("Restauração", "N/A");
                break;
        }
    } while (opcao != 0);

    realizarBackup();
    return 0;
}
```

Implementar Sistema de Permissões por Nível de Usuário

Este aluno implementa um sistema de permissões que diferencia usuários comuns de administradores.

funcoes.h

```
int verificarNivelPermissao(char usuario[]);
```

funcoes.c

```
int verificarNivelPermissao(char usuario[]) {
    if (strcmp(usuario, "admin") == 0) {
        return 1;  // Administrador
    } else {
        return 0;  // Usuário comum
    }
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char usuario[50];
```

```
    printf("Digite o nome de usuário: ");
    scanf("%s", usuario);

    if (!login()) {
        printf("Acesso negado.\n");
        return 0;
    }

    int opcao;
    do {
        mostrarMenu();
        opcao = lerOpcao();
        if (opcao == 3 && verificarNivelPermissao(usuario) ==
0) {
            printf("Permissão negada. Apenas administradores
podem excluir dados.\n");
            continue;
        }

        switch (opcao) {
            case 1:
                // Funcionalidades como antes...
                break;
```

```
            case 16:
                restaurarBackup();
                break;
        }
    } while (opcao != 0);

    realizarBackup();
    return 0;
}
```

Implementar Validação de Senha Forte

O aluno adiciona uma validação para exigir senhas fortes na criação de novas contas de usuários.

funcoes.h

```
int validarSenhaForte(char senha[]);
```

funcoes.c

```
#include <ctype.h>
#include <string.h>
```

```
int validarSenhaForte(char senha[]) {
    int temLetra = 0, temNumero = 0, temEspecial = 0;

    for (int i = 0; i < strlen(senha); i++) {
        if (isalpha(senha[i])) temLetra = 1;
        if (isdigit(senha[i])) temNumero = 1;
        if (!isalnum(senha[i])) temEspecial = 1;
    }

    if (temLetra && temNumero && temEspecial && strlen(senha)
>= 8) {
        return 1;  // Senha válida
    }

    printf("A senha deve ter pelo menos 8 caracteres,
incluindo letras, números e caracteres especiais.\n");
    return 0;
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"
```

```
int main() {
    char usuario[50], senha[50];

    printf("Digite o nome de usuário: ");
    scanf("%s", usuario);

    do {
        printf("Digite a senha: ");
        scanf("%s", senha);
    } while (!validarSenhaForte(senha));

    if (!login()) {
        printf("Acesso negado.\n");
        return 0;
    }

    // Demais funcionalidades...
}
```

Relatório de Alunos Cadastrados por Data

O aluno adiciona uma função para gerar um relatório com os alunos cadastrados em uma data específica.

funcoes.h

```
void gerarRelatorioPorData(char data[]);
```

funcoes.c

```
#include <stdio.h>
#include <string.h>

void gerarRelatorioPorData(char data[]) {
    FILE *arquivo = fopen("alunos.txt", "r");
    char linha[100], nome[50], genero[20], dataCadastro[20];

    if (arquivo == NULL) {
        perror("Erro ao abrir arquivo");
        return;
    }

    printf("Alunos cadastrados em %s:\n", data);
    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
```

```
        sscanf(linha, "%[^,], %[^,], %s", nome, genero,
dataCadastro);
        if (strcmp(dataCadastro, data) == 0) {
            printf("Nome: %s, Gênero: %s\n", nome, genero);
        }
    }

    fclose(arquivo);
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    char data[20];

    if (!login()) {
        printf("Acesso negado.\n");
        return 0;
    }
```

```
    printf("Digite a data (DD/MM/AAAA): ");
    scanf("%s", data);

    gerarRelatorioPorData(data);

    // Restante do código...
}
```

Criação de Logs de Erro

Este aluno implementa um sistema de logs separado para erros.

funcoes.h

```
void registrarLogErro(char erro[]);
```

funcoes.c

```
#include <stdio.h>
#include <time.h>

void registrarLogErro(char erro[]) {
    FILE *arquivoLogErro = fopen("log_erro.txt", "a");
```

```
    time_t agora;
    time(&agora);

    if (arquivoLogErro == NULL) {
        perror("Erro ao abrir arquivo de log de erros");
        return;
    }

    fprintf(arquivoLogErro, "Erro: %s | Data e Hora: %s",
erro, ctime(&agora));
    fclose(arquivoLogErro);
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    if (!login()) {
        registrarLogErro("Tentativa de login falhou");
        printf("Acesso negado.\n");
        return 0;
```

```
    }

    // Código do sistema...
}
```

Sistema de Backup Incremental

Este aluno implementa a criação de backups incrementais, que só salvam mudanças feitas nos dados.

funcoes.h

```
void realizarBackupIncremental();
```

funcoes.c

```
#include <stdio.h>

void realizarBackupIncremental() {
    FILE *arquivoOrigem = fopen("alunos.txt", "r");
    FILE *arquivoBackup = fopen("backup_incremental.txt",
"a");
    char linha[100];
```

```
    if (arquivoOrigem == NULL || arquivoBackup == NULL) {
        registrarLogErro("Erro ao abrir arquivos para backup
incremental");
        return;
    }

    while (fgets(linha, sizeof(linha), arquivoOrigem) != NULL)
{
        fputs(linha, arquivoBackup);
    }

    fclose(arquivoOrigem);
    fclose(arquivoBackup);

    printf("Backup incremental realizado com sucesso!\n");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
```

```
    if (!login()) {
        registrarLogErro("Tentativa de login falhou");
        printf("Acesso negado.\n");
        return 0;
    }

    realizarBackupIncremental();  // Realiza backup
incremental no final
    return 0;
}
```

Envio de Relatórios por E-mail

Este aluno adiciona uma funcionalidade que simula o envio de relatórios por e-mail.

funcoes.h

```
void enviarRelatorioPorEmail(char email[], char conteudo[]);
```

funcoes.c

```
#include <stdio.h>

void enviarRelatorioPorEmail(char email[], char conteudo[]) {
```

```
    printf("Enviando o seguinte relatório para %s:\n", email);
    printf("Conteúdo: %s\n", conteudo);
    // Aqui seria onde a integração com um servidor de e-mail
ocorreria
    printf("Relatório enviado com sucesso!\n");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    if (!login()) {
        registrarLogErro("Tentativa de login falhou");
        printf("Acesso negado.\n");
        return 0;
    }

    char email[50] = "admin@exemplo.com";
    char conteudo[200] = "Relatório de alunos cadastrados.";

    enviarRelatorioPorEmail(email, conteudo);
```

```
    return 0;
}
```

Exportação para JSON e XML

O aluno cria funções para exportar os dados dos alunos em formato JSON e XML.

funcoes.h

```
void exportarParaJSON();
void exportarParaXML();
```

funcoes.c

```
#include <stdio.h>

void exportarParaJSON() {
    FILE *arquivo = fopen("alunos.txt", "r");
    FILE *json = fopen("alunos.json", "w");
    char linha[100], nome[50], genero[20];

    if (arquivo == NULL || json == NULL) {
        registrarLogErro("Erro ao abrir arquivos para
exportação JSON");
```

```
        return;
    }

    fprintf(json, "[\n");
    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        sscanf(linha, "%[^,], %s", nome, genero);
        fprintf(json, "  { \"nome\": \"%s\", \"genero\":
\"%s\" },\n", nome, genero);
    }
    fprintf(json, "]\n");

    fclose(arquivo);
    fclose(json);

    printf("Exportação para JSON concluída!\n");
}

void exportarParaXML() {
    FILE *arquivo = fopen("alunos.txt", "r");
    FILE *xml = fopen("alunos.xml", "w");
    char linha[100], nome[50], genero[20];

    if (arquivo == NULL || xml == NULL) {
```

```
        registrarLogErro("Erro ao abrir arquivos para
exportação XML");
        return;
    }

    fprintf(xml, "<alunos>\n");
    while (fgets(linha, sizeof(linha), arquivo) != NULL) {
        sscanf(linha, "%[^,], %s", nome, genero);
        fprintf(xml, "  <aluno>\n    <nome>%s</nome>\n
<genero>%s</genero>\n  </aluno>\n", nome, genero);
    }
    fprintf(xml, "</alunos>\n");

    fclose(arquivo);
    fclose(xml);

    printf("Exportação para XML concluída!\n");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"
```

```
int main() {
    if (!login()) {
        registrarLogErro("Tentativa de login falhou");
        printf("Acesso negado.\n");
        return 0;
    }

    exportarParaJSON();
    exportarParaXML();
    return 0;
}
```

Melhorias na Interface do Usuário

Este aluno melhora a interface, tornando-a mais intuitiva com menus mais claros.

funcoes.h

```
void mostrarMenuMelhorado();
```

funcoes.c

```
void mostrarMenuMelhorado() {
    printf("\n--- Sistema de Cadastro de Alunos ---\n");
    printf("1. Adicionar aluno\n");
    printf("2. Listar alunos\n");
    printf("3. Exportar dados (JSON/XML)\n");
    printf("4. Restaurar backup\n");
    printf("0. Sair\n");
    printf("-------------------------------------\n");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    if (!login()) {
        registrarLogErro("Tentativa de login falhou");
        printf("Acesso negado.\n");
        return 0;
    }
```

```
    int opcao;
    do {
        mostrarMenuMelhorado();
        opcao = lerOpcao();
        switch (opcao) {
            case 1:
                // Adicionar aluno...
                break;
            case 3:
                exportarParaJSON();
                exportarParaXML();
                break;
            case 4:
                restaurarBackup();
                break;
        }
    } while (opcao != 0);

    return 0;
}
```

Relatório de Alunos Inativos

Este aluno implementa um relatório que exibe alunos que não foram modificados em 6 meses.

funcoes.h

```
void gerarRelatorioAlunosInativos();
```

funcoes.c

```
#include <stdio.h>
#include <time.h>

void gerarRelatorioAlunosInativos() {
    // Simulando verificação de inatividade
    printf("Relatório de alunos inativos (mais de 6 meses sem modificações):\n");
    printf("Aluno: João Silva\n");
    printf("Aluno: Maria Souza\n");
    // Seria necessário armazenar as datas de modificações para um sistema real
}
```

main.c

```
#include <stdio.h>
#include <string.h>
```

```c
#include "funcoes.h"

int main() {
    if (!login()) {
        registrarLogErro("Tentativa de login falhou");
        printf("Acesso negado.\n");
        return 0;
    }

    gerarRelatorioAlunosInativos();
    return 0;
}
```

Notificações Automáticas

O sistema agora envia notificações automáticas para os alunos.

funcoes.h

```c
void enviarNotificacoesAutomaticas();
```

funcoes.c

```
void enviarNotificacoesAutomaticas() {
    printf("Enviando notificações automáticas para os alunos...\n");
    // Aqui seria implementada uma integração com um serviço de envio de e-mails
    printf("Notificações enviadas com sucesso!\n");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    if (!login()) {
        registrarLogErro("Tentativa de login falhou");
        printf("Acesso negado.\n");
        return 0;
    }

    enviarNotificacoesAutomaticas();
    return 0;
}
```

Suporte a Múltiplos Idiomas

O aluno adiciona suporte a múltiplos idiomas para o sistema.

funcoes.h

```
void definirIdioma(int opcao);

void exibirMensagemBoasVindas();
```

funcoes.c

```
#include <stdio.h>

char *idiomaAtual;

void definirIdioma(int opcao) {
    switch (opcao) {
        case 1:
            idiomaAtual = "Português";
            break;
        case 2:
            idiomaAtual = "Inglês";
```

```
            break;
        default:
            idiomaAtual = "Português";
    }
}

void exibirMensagemBoasVindas() {
    if (idiomaAtual == "Português") {
        printf("Bem-vindo ao sistema!\n");
    } else if (idiomaAtual == "Inglês") {
        printf("Welcome to the system!\n");
    }
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    int idioma;
```

```
    printf("Selecione o idioma:\n1. Português\n2. Inglês\n");
    scanf("%d", &idioma);
    definirIdioma(idioma);
    exibirMensagemBoasVindas();

    // Continuação do código...
    return 0;
}
```

Estatísticas Avançadas com Gráficos ASCII

O aluno implementa um sistema de estatísticas com gráficos simulados em ASCII.

funcoes.h

```
void exibirEstatisticas();
```

funcoes.c

```
void exibirEstatisticas() {
    printf("Estatísticas de alunos cadastrados:\n");
    printf("Gênero Masculino: ##### 50%%\n");
    printf("Gênero Feminino: #### 40%%\n");
```

```
    printf("Outro: ## 10%%\n");

    // Dados de exemplo, em um sistema real, as estatísticas
seriam dinâmicas

}
```

main.c

```
#include <stdio.h>

#include <string.h>

#include "funcoes.h"


int main() {

    int idioma;


    printf("Selecione o idioma:\n1. Português\n2. Inglês\n");

    scanf("%d", &idioma);

    definirIdioma(idioma);

    exibirMensagemBoasVindas();


    exibirEstatisticas();  // Exibe estatísticas com gráficos
ASCII

    return 0;

}
```

Implementação de Sistema de Cache

O aluno implementa um sistema de cache para otimizar a leitura de dados.

funcoes.h

```
char* buscarAlunoCache(int id);
```

funcoes.c

```
#include <stdio.h>
#include <string.h>

char cacheAluno[100][100];  // Exemplo de cache em memória
int cacheId[100];
int cacheTamanho = 0;

char* buscarAlunoCache(int id) {
    for (int i = 0; i < cacheTamanho; i++) {
        if (cacheId[i] == id) {
            return cacheAluno[i];
        }
    }
```

```
    // Se o aluno não estiver no cache, buscá-lo do arquivo
(simulado)
    sprintf(cacheAluno[cacheTamanho], "Aluno com ID %d", id);
    cacheId[cacheTamanho] = id;
    cacheTamanho++;

    return cacheAluno[cacheTamanho - 1];
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    int idioma;

    printf("Selecione o idioma:\n1. Português\n2. Inglês\n");
    scanf("%d", &idioma);
    definirIdioma(idioma);
    exibirMensagemBoasVindas();
```

```
    exibirEstatisticas();

    // Busca aluno usando cache
    char *aluno = buscarAlunoCache(101);
    printf("Aluno encontrado: %s\n", aluno);

    return 0;
}
```

Auditoria de Usuários

O aluno implementa um sistema de auditoria para registrar ações de usuários.

funcoes.h

```
void registrarAuditoria(char usuario[], char acao[]);
```

funcoes.c

```
#include <stdio.h>
#include <time.h>

void registrarAuditoria(char usuario[], char acao[]) {
```

```
    FILE *arquivoAuditoria = fopen("auditoria.txt", "a");
    time_t agora;
    time(&agora);

    if (arquivoAuditoria == NULL) {
        registrarLogErro("Erro ao abrir arquivo de
auditoria");
        return;
    }

    fprintf(arquivoAuditoria, "Usuário: %s | Ação: %s | Data e
Hora: %s", usuario, acao, ctime(&agora));
    fclose(arquivoAuditoria);

    printf("Ação registrada na auditoria!\n");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
```

```
    int idioma;

    printf("Selecione o idioma:\n1. Português\n2. Inglês\n");
    scanf("%d", &idioma);
    definirIdioma(idioma);
    exibirMensagemBoasVindas();

    exibirEstatisticas();

    // Busca aluno usando cache
    char *aluno = buscarAlunoCache(101);
    printf("Aluno encontrado: %s\n", aluno);

    // Registro de auditoria
    registrarAuditoria("admin", "Buscou aluno");
    return 0;
}
```

Slide 21 (Aluno 21): Função de Logout Seguro

O aluno 21 adiciona uma função de logout seguro, finalizando a sessão do usuário de forma controlada.

funcoes.h

```
void logout();
```

funcoes.c

```
void logout() {
    printf("Logout realizado com sucesso. Até logo!\n");
    registrarAuditoria("admin", "Logout realizado");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    int idioma;

    printf("Selecione o idioma:\n1. Português\n2. Inglês\n");
    scanf("%d", &idioma);
    definirIdioma(idioma);
    exibirMensagemBoasVindas();
```

```
    exibirEstatisticas();

    // Busca aluno usando cache
    char *aluno = buscarAlunoCache(101);
    printf("Aluno encontrado: %s\n", aluno);

    // Registro de auditoria
    registrarAuditoria("admin", "Buscou aluno");

    // Logout seguro
    logout();
    return 0;
}
```

Relatório Final do Projeto

O aluno cria um relatório final do projeto, que exibe todas as funções implementadas e um resumo do progresso.

funcoes.h

```
void gerarRelatorioFinal();
```

funcoes.c

```
void gerarRelatorioFinal() {
    printf("\n--- Relatório Final do Projeto ---\n");
    printf("1. Logs de erro implementados\n");
    printf("2. Backup incremental adicionado\n");
    printf("3. Suporte a múltiplos idiomas\n");
    printf("4. Sistema de cache otimizado\n");
    printf("5. Auditoria de usuários\n");
    printf("6. Estatísticas avançadas com gráficos ASCII\n");
    printf("7. Relatório gerado com sucesso!\n");
}
```

main.c

```
#include <stdio.h>
#include <string.h>
#include "funcoes.h"

int main() {
    int idioma;
    printf("Selecione o idioma:\n1. Português\n2. Inglês\n");
    scanf("%d", &idioma);
```

```
    definirIdioma(idioma);

    exibirMensagemBoasVindas();

    exibirEstatisticas();

    // Busca aluno usando cache

    char *aluno = buscarAlunoCache(101);

    printf("Aluno encontrado: %s\n", aluno);

    // Registro de auditoria

    registrarAuditoria("admin", "Buscou aluno");

    // Logout seguro

    logout();

    // Gerar relatório final

    gerarRelatorioFinal();

    return 0;

}
```

Chegou até aqui: envie no email: matheus.araujo.ecomp@gmail.com Coloque seu nome no assunto Parabéns

Se você for o primeiro a enviar o email nessa rodada, você ganhará um brinde, se o código estiver correto e funcionando.

Pode ir fazer o projeto.

A empresa código agradece seu empenho!