Sistema de Gerenciamento de Funcionários

Projeto de lp1

Professor: Francisco Sant’anna

Alunos: Marcelo Paulino e Matheus Mota

Universidade do Estado do Rio de Janeiro

Descrição do Projeto e Objetivos

RH Empresa é um programa em C para gerenciar funcionários via terminal, armazenando registros em arquivo texto (funcionarios.txt) e permitindo cadastro, busca, edição, exclusão lógica (marcar inativo) e exportação de listagens, permitindo o gerenciamento de funcionários em uma empresa.

Objetivos:

- Registrar e manter informações básicas de funcionários (nome, papel, dados específicos, tempo na empresa e data de nascimento).
- Fornecer operações do ciclo de vida do colaborador: cadastro, consulta, edição, inativação, restauração e exportação.
- Garantir persistência simples e legível usando arquivo texto para facilitar inspeção manual.

Estruturas de Dados

O sistema representa cada funcionário com uma struct Funcionario que reúne dados comuns (id, nome, status, tempo de empresa e aniversário) e um campo variável papel_info. O tipo do papel é indicado por um enum (FuncaoPapel) – ele funciona como uma tag que diz qual conjunto de dados específicos deve ser usado. Esses dados específicos estão em diferentes structs e são armazenados dentro de uma union (CargoInfo), o que significa que apenas um dos blocos ocupa memória por funcionário. Na persistência, o programa converte os campos ativos da union em dois inteiros ao gravar no arquivo texto e, ao carregar, usa o enum do papel para reconstruir corretamente o membro da union.

```
    PAPEL_OUTRO = 0,
    PAPEL_DESENVOLVEDOR = 1,
    PAPEL_FAXINEIRO = 2,
    PAPEL_GERENTE = 3,
    PAPEL_SEGURANCA = 4
};

enum DevNivel { NIVEL_ESTAGIARIO = 0, NIVEL_JUNIOR, NIVEL_PLENO, NIVEL_SENIOR };
enum DevLinguagem { LING_C = 0, LING_PYTHON, LING_JAVA, LING_OUTRA };

typedef struct {…
} InfoDesenvolvedor;

enum FaxAndar { ANDAR_1 = 1, ANDAR_2 = 2, ANDAR_3 = 3 };
enum FaxCargo { FAX_COMUM = 0, FAX_CHEFE = 1 };

typedef struct {…
} InfoFaxineiro;

enum GerenciaFuncao { GER_BANCARIO = 0, GER_JUDICIARIO, GER_ADMINISTRATIVO };
enum GerCargo { GERENTE_CHEFE = 0, GER_SUBGERENTE, GER_COMUM, GER_JOVEM_APRENDIZ };

typedef struct {…
} InfoGerenteDet;

enum SegLocal { LOC_ANDAR1 = 1, LOC_ANDAR2 = 2, LOC_ANDAR3 = 3, LOC_ENTRADA = 4 };
enum SegCargo { SEG_SUPERVISOR = 0, SEG_COMUM = 1 };

typedef struct {…
} InfoSeguranca;

typedef union {…
} CargoInfo;

typedef struct {
    int id;
    char nome[50];
    enum Status ativo;
    enum FuncaoPapel papel;
```

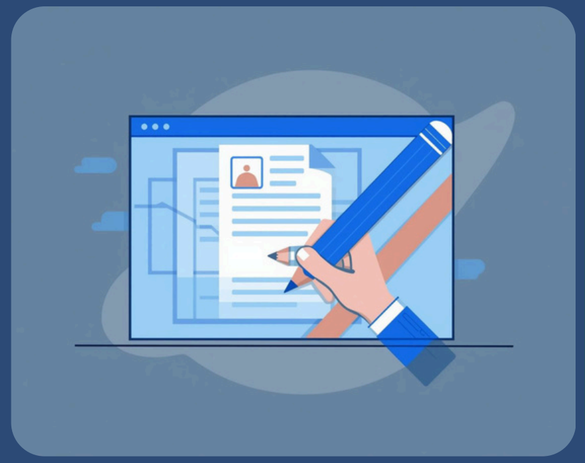
Principais Funcionalidades do Sistema

Cadastrar

Adiciona novos funcionários ao sistema

Buscar

Localiza funcionários por diferentes critérios



Editar

Atualiza informações dos funcionários existentes

Excluir

Remove registros de funcionários do sistema

Além das quatro funções principais, também há a função “restaura”, que permite mudar o status de inativo para ativo após uma exclusão.

```c
void restaurar_funcionario_txt(void) {
    int id;
    printf("Digite o ID do funcionario para restaurar: ");
    if (scanf("%d", &id) != 1) { limpar_buffer(); printf("ID invalido.\n"); return; }

    int n = 0;
    Funcionario *vet = carregar_todos_txt(NOME_ARQUIVO, &n);
    if (!vet) { printf("Arquivo vazio ou inexistente.\n"); return; }

    for (int i = 0; i < n; i++) {
        if (vet[i].id == id) {
            if (vet[i].ativo == INATIVO) {
                vet[i].ativo = ATIVO;
                if (!salvar_todos_txt(NOME_ARQUIVO, vet, n))
                    printf("Erro ao salvar alteracoes no arquivo.\n");
                else
                    printf("Funcionario (ID: %d) restaurado com sucesso para ATIVO.\n", id);
            } else {
                printf("Este funcionario ja esta ativo.\n");
            }
            free(vet);
            return;
        }
    }

    free(vet);
    printf("Funcionario nao encontrado.\n");
}
```

Organização do Programa

O programa inicia chamando o menu, que funciona como o centro de controle. Nele, o usuário escolhe ações como cadastrar, buscar, editar, excluir, restaurar, listar ou exportar funcionários, digitando números para acessar cada uma das funcionalidades do programa

As operações seguem um padrão: quando é necessário acessar os dados, o programa lê todo o arquivo funcionarios.txt, reconstrói os registros em memória e, após qualquer mudança, salva novamente o arquivo completo.

O cadastro cria um novo funcionário preenchendo dados gerais e informações específicas conforme o papel escolhido, armazenadas através de uma union controlada por um enum. A exclusão é lógica: o funcionário apenas muda para INATIVO, podendo ser restaurado depois.

As funções de listagem e busca exibem apenas funcionários ativos, e a exportação gera um arquivo separado com uma listagem formatada. Funções auxiliares cuidam da limpeza de entrada, da data de aniversário, da conversão dos dados para texto e da reconstrução da union ao carregar o arquivo.

Conclusões e Aprendizados

O projeto resultou em um sistema funcional e completo para gerenciar funcionários, demonstrando na prática o uso de estruturas de dados em C. Durante o desenvolvimento foram consolidados conceitos importantes sobre a linguagem C e a programação como um todo. Aprendemossobre a manipulação de arquivos para persistência, organização de dados com structs, enums e union discriminada, construção de menus interativos e tratamento básico de erros, todos aprendizados que tornam o código robusto e extensível para aprimoramentos futuros.

Fim! Obrigado pela atenção.

Email

mota.matheus_1@graduacao.uerj.br
Neves.marcelo@graduacao.uerj.br