Documentação Técnica

Lucas de Miranda Martins

1. Introdução

Este documento descreve o sistema desenvolvido para o gerenciamento de uma clínica médica, contemplando controle de agendamentos, pacientes, profissionais, procedimentos, estoque e movimentações financeiras. O sistema é implementado em linguagem C, com persistência dos dados em arquivos binários para garantir eficiência e segurança.

2.Estrutura de dados

Abaixo nas imagens estão todas as estruturas utilizadas para a realização do trabalho

```c
typedef struct
{
    long int codigo;
    long int qnt; // quantidade u
} Codmedicamentosmateriais;

typedef struct
{
    long int codigo;
    char descricao[1000];
    float precocustounid, total;
    long int qnt;        // quant
} Codmateriaisrecebidos; // vai s

typedef struct
{
    char rua[200];
    char bairro[200];
    char cidade[200];
    char estado[10]; // Ex: "SP"
    long int numero;
} Endereco;
```

```c
typedef struct
{
    long int codigo;
    char nomeCompleto[200];
    char cpf[50];
    Endereco endereco;
    char telefone[20];
    char dataNascimento[40];
    char historicoMedico[2000];
} Paciente;

typedef struct
{
    long int codigo;
    char nomeCompleto[200];
    char crm[50];
    char especialidade[400];
    char cpf[50];
    char telefone[20];
    char email[300];
} Profissional;
```

```c
typedef struct
{
    long int codigo;
    char nomeFantasia[100];
    char razaoSocial[200];
    char inscricaoEstadual[200];
    char cnpj[40];
    Endereco endereco;
    char telefone[20];
    char email[300];
} Fornecedor;

typedef struct
{
    long int codigo;
    char descricaoProcedimento[1000];
} AmbienteMedico;
```

```c
typedef struct
{
    long int codigo;
    char descricao[1000];
    char fabricante[300];
    long int codfornecedor; // codigo do fornecedor
    float precoCusto;
    float precoVenda;
    long int quantidadeEstoque;
    long int estoqueMinimo;
} MedicamentoMaterial;

typedef struct
{
    long int codigo;
    char descricao[2000];
    float custo;
    long int tempoEstimado;
    Codmedicamentosmateriais *codmedicamentosemateriais;
    long int codambientemedico; // codigo do ambiente medico
    long int tamcodmedicamentosmateriais;
} Procedimento;
```

```c
typedef struct
{
    long int codigo, codpaciente, codprofissional, codprocedimento;
    char data[20], horario[20], horariofim[20];  ;
} Agendamento;

typedef struct
{
    long int codigo;
    char cnpj[1000], fornecedor[1000];
    char data[1000];
    float frete, imposto, totaldetudo;
    Codmateriaisrecebidos *codmedicamentosmateriaisrecebidos;
    long int codfornecedor;
    long int tammedicamentosmateriaisrecebidos;
} Entradaestoque;

typedef struct
{
    char data[100];
    int tipo; // 0 para lançamento a vista 1 para retirada 2 para a re
    float valor;
    long int codigoagendamento;    // para os a vista
    long int codigoentradaestoque; // registtra  o codigo da entrada d
} LacamentosRetiradas;
```

```c
typedef struct
{
    long int codigoa
    long int codigo;
    float valor;
    char data[100];
} Receber;
```

Todas as funções têm o objetivo de registrar as informações de um determinda parte do sistema.

3. Funcionamento do Sistema

O sistema permite o armazenamento de dados em três formas distintas: **memória**, **arquivo `.txt`**, e **arquivo binário (`.bin`)**. No início da execução da aplicação, o usuário deve escolher o tipo de armazenamento desejado.

Após essa escolha, o usuário é redirecionado ao **menu principal**, localizado na função `main`. Esse menu direciona o usuário para os **submenus** da camada `view`, que por sua vez fazem chamadas às funções responsáveis por cada operação. O sistema é dividido em módulos, conforme descrito a seguir:

3.1 Cadastros

As opções de 1 a 6 no menu principal correspondem ao **CRUD (Create, Read, Update, Delete)** das entidades principais do sistema. Cada uma dessas opções leva o usuário a um submenu específico, onde é possível realizar as operações de cadastro, edição, exclusão e visualização dos dados.

3.2 Agendamento e Realização de Procedimentos

A opção 7 leva ao submenu responsável pelo **agendamento de procedimentos**. Nele, a função `cadastrarAgendamento` realiza todas as validações necessárias para garantir que as informações fornecidas correspondam aos dados previamente cadastrados. Caso todas as condições sejam satisfeitas, o agendamento é efetuado com sucesso e o valor do procedimento é automaticamente debitado.

3.3 Importação e Exportação de Dados

As opções 8 e 9 permitem ao usuário **importar e exportar dados**. O submenu correspondente oferece a escolha de importar ou exportar apenas uma tabela ou múltiplas tabelas, conforme a necessidade do usuário. Os dados podem ser salvos ou lidos de arquivos `.txt` ou `.bin`, de acordo com o tipo de armazenamento escolhido.

3.4 Controle de Estoque

A opção 10 do menu principal acessa o módulo de **estoque**, que apresenta um submenu com as seguintes funcionalidades:

- Registro de entrada de estoque.
- Verificação do estoque mínimo de materiais.

3.5 Módulo de Caixa

O módulo de caixa permite:

- Cadastrar uma conta a receber, armazenada na `struct Receber`, vinculando uma data e o código do agendamento.
- Registrar pagamentos à vista, que são armazenados na `struct LancamentosRetiradas`, correspondente ao histórico do caixa.
- Realizar o pagamento de uma entrada de estoque.
- Visualizar o histórico de movimentações no caixa.

3.6 Geração de Relatórios

O módulo de relatórios permite a geração de relatórios exibidos diretamente na tela ou salvos em um arquivo `.csv` chamado `relatorio.csv`. O sistema possibilita a aplicação de **filtros específicos** em cada tipo de relatório, facilitando a análise de dados conforme o interesse do usuário.

---

4. Informações Importantes

- O sistema suporta três formas de armazenamento: **em memória**, **em arquivos `.txt`**, ou **em arquivos binários (`.bin`)**.
- Quando o armazenamento escolhido for em arquivo, os dados são **persistidos** mesmo após o encerramento do programa.
- Ao optar pelo armazenamento em memória, os dados existem apenas durante a execução do programa e são perdidos ao encerrá-lo.
- O sistema utiliza uma estrutura modularizada, dividida entre as camadas de `main`, `view`, e funções auxiliares, garantindo a organização e manutenção facilitada do código.