IFMG – CAMPUS FORMIGA
CIÊNCIA DA COMPUTAÇÃO

FELIPE DE ASSIS COSTA
R.A.: 0049539

PROJETO ALGORITMOS II

Documentação Sistema de Hotel

FORMIGA – MG
2021

1. Funções por arquivo
   a) acomodacao.c

```c
int verificarCodigoAcomodacao(structAcomodacao *acomodacao, structAcomodacao *acomodacaoDados, int *quantidadeAcomodacoes);

int retornaPosicaoAcomodacao(structAcomodacao *acomodacao, structAcomodacao *acomodacaoDados, int *quantidadeAcomodacoes);

void cadastrarAcomodacao(structAcomodacao *acomodacao, structAcomodacao *acomodacaoDados, int *quantidadeAcomodacoes);

void gerarArqAcomodacao(structAcomodacao *acomodacaoDados, int quantidadeAcomodacoes, FILE *arqQtdAcomodacoes, FILE *arqDadosAcomodacoes);

void exibeAcomodacao(structAcomodacao *acomodacao, structAcomodacao *acomodacaoDados, int *quantidadeAcomodacoes);

void listarAcomodacoes(structAcomodacao *acomodacao, int *quantidadeAcomodacoes);

void atualizarAcomodacao(structAcomodacao *acomodacao, structAcomodacao *acomodacaoDados, int *quantidadeAcomodacoes, FILE *arqDadosAcomodacoes);

void deletarAcomodacao(structAcomodacao *acomodacao, structAcomodacao *acomodacaoDados, int *quantidadeAcomodacoes,
                       FILE *arqQtdAcomodacoes, FILE *arqDadosAcomodacoes);
```

   b) categoriaAcomodacao.c

```c
int verificaCodigoCategoria(structCategoriaAcomodacao *categoriaAcomodacao, structCategoriaAcomodacao *categoriaAcomodacaoDados,
                            int *quantidadeCategorias);

int retornaPosicaoCategoria(structCategoriaAcomodacao *categoriaAcomodacao, structCategoriaAcomodacao *categoriaAcomodacaoDados,
                            int *quantidadeCategorias);

void cadastrarCategoria(structCategoriaAcomodacao *categoriaAcomodacao, structCategoriaAcomodacao *categoriaAcomodacaoDados,
                        int *quantidadeCategorias);

void gerarArqCategoria(structCategoriaAcomodacao *categoriaAcomodacaoDados, int quantidadeCategorias, FILE *arqQtdCategorias,
                       FILE *arqDadosCategorias);

void exibeCategoria(structCategoriaAcomodacao *categoriaAcomodacao, structCategoriaAcomodacao *categoriaAcomodacaoDados,
                    int *quantidadeCategorias);

void listarCategorias(structCategoriaAcomodacao *categoriaAcomodacao, int *quantidadeCategorias);

void atualizarCategorias(structCategoriaAcomodacao *categoriaAcomodacao, structCategoriaAcomodacao *categoriaAcomodacaoDados,
                         int *quantidadeCategorias, FILE *arqDadosCategorias);

void deletarCategoria(structCategoriaAcomodacao *categoriaAcomodacao, structCategoriaAcomodacao *categoriaAcomodacaoDados,
                      int *quantidadeCategorias, FILE *arqQtdCategorias, FILE *arqDadosCategorias);
```

   c) fornecedor.c

```c
int verificarCodigoFornecedor(structFornecedores *fornecedores, structFornecedores *fornecedoresDados, int *quantidadeFornecedores);

void cadastrarFornecedor(structFornecedores *fornecedores, structFornecedores *fornecedoresDados, int *quantidadeFornecedores);

void gerarArqFornecedor(structFornecedores *fornecedorDados, int quantidadeFornecedores, FILE *arqQtdFornecedores,
                        FILE *arqDadosFornecedores);

void exibeFornecedor(structFornecedores *fornecedores, structFornecedores *fornecedoresDados, int *quantidadeFornecedores);

void listarFornecedores(structFornecedores *fornecedores, int *quantidadeFornecedores);

void atualizaFornecedor(structFornecedores *fornecedores, structFornecedores *fornecedoresDados, int *quantidadeFornecedores,
                        FILE *arqDadosFornecedores);

void deletarFornecedor(structFornecedores *fornecedores, structFornecedores *fornecedoresDados, int *quantidadeFornecedores,
                       FILE *arqQtdFornecedores, FILE *arqDadosFornecedores);
```

   d) horaData.c

```c
void data();
```

   e) hospede.c

```c
int verificarCodigoHospede(structHospede *hospede, structHospede *hospedeDados, int *quantidadeHospedes);

void cadastrarHospede(structHospede *hospede, structHospede *hospedeDados, int *quantidadeHospedes);

void gerarArqHospedes(structHospede *hospedeDados, int quantidadeHospedes, FILE *arqQtdHospedes, FILE *arqDadosHospedes);

void exibeHospede(structHospede *hospede, structHospede *hospedeDados, int *quantidadeHospedes);

void listarHospedes(structHospede *hospede, int *quantidadeHospedes);

void atualizarHospede(structHospede *hospede, structHospede *hospedeDados, int *quantidadeHospedes, FILE *arqDadosHospedes);

void deletarHospede(structHospede *hospede, structHospede *hospedeDados, int *quantidadeHospedes, FILE *arqQtdHospedes,
                    FILE *arqDadosHospedes);
```

f) hotel.c

```c
int confirmaSeExisteHotelCadastrado(int *confirmacaoHotel);

void cadastrarHotel(structHotel *hotelDados, structHotel *hotel, int *confirmacaoHotel, FILE *arqConfirmacaoHotel, FILE *arqDadosHotel);

void exibeHotel(structHotel *hotel);
```

g) operadorSistema.c

```c
int verificarCodigoOperador(structOperadoresSistema *operadoresSistema, structOperadoresSistema *operadoresSistemaDados,
                            int *quantidadeOperadores);

int verificarUsuarioOperador(structOperadoresSistema *operadoresSistema, structOperadoresSistema *operadoresSistemaDados,
                             int *quantidadeOperadores);

void cadastrarOperador(structOperadoresSistema *operadoresSistema, structOperadoresSistema *operadoresSistemaDados,
                       int *quantidadeOperadores);

void gerarArqOperador(structOperadoresSistema *operatoresSistemaDados, int quantidadeOperadores, FILE *arqQtdOperadores,
                      FILE *arqDadosOperadores);

void exibeOperador(structOperadoresSistema *operadoresSistema, structOperadoresSistema *operadoresSistemaDados,
                   int *quantidadeOperadores);

void listaOperadores(structOperadoresSistema *operadoresSistema, int *quantidadeOperadores);

void atualizaOperador(structOperadoresSistema *operadoresSistema, structOperadoresSistema *operadoresSistemaDados,
                      int *quantidadeOperadores, FILE *arqDadosOperadores);

void deletarOperador(structOperadoresSistema *operadoresSistema, structOperadoresSistema *operadoresSistemaDados,
                     int *quantidadeOperadores, FILE *arqQtdOperadores, FILE *arqDadosOperadores);
```

h) produtosDisponiveis.c

```c
int verificarCodigoProduto(structProdutosConsumo *produtosConsumo, structProdutosConsumo *produtosConsumoDados, int *quantidadeProdutos);

void cadastrarProduto(structProdutosConsumo *produtosConsumo, structProdutosConsumo *produtosConsumoDados, int *quantidadeProdutos);

void gerarArqProdutos(structProdutosConsumo *produtosConsumoDados, int quantidadesProdutos, FILE *arqQtdProdutos, FILE *arqDadosProdutos);

void exibirProduto(structProdutosConsumo *produtosConsumo, structProdutosConsumo *produtosConsumoDados, int *quantidadeProdutos);

void listarProdutos(structProdutosConsumo *produtosConsumo, structProdutosConsumo *produtosConsumoDados, int *quantidadeProdutos);

void atualizarProduto(structProdutosConsumo *produtosConsumo, structProdutosConsumo *produtosConsumoDados, int *quantidadeProdutos,
                      FILE *arqDadosProdutos);

void deletarProduto(structProdutosConsumo *produtosConsumo, structProdutosConsumo *produtosConsumoDados, int *quantidadeProdutos,
                    FILE *arqQtdProdutos, FILE *arqDadosProdutos);
```

i) reservas.c

```c
int verifDatas(structReserva *reservaDados);

int qtdDiasReserva(structReserva *reserva, structReserva *reservaDados, int *quantidadeReservas, int i);

void cadastrarReserva(structReserva *reserva, structReserva *reservaDados, int *quantidadeReservas);

void gerarArqReservas(structReserva *reservaDados, int quantidadeReservas, FILE *arqQtdReservas, FILE *arqDadosReservas);

int verificarCodigoReserva(structReserva *reserva, structReserva *reservaDados, int *quantidadeReservas);

int retornaPosicaoCodigoReserva(structReserva *reserva, structReserva *reservaDados, int *quantidadeReservas);

int verificaSePodeCadastrar(structReserva *reserva, structReserva *reservaDados, int quantidadeReservas);

void exibeReserva(structReserva *reserva, structReserva *reservaDados, int *quantidadeReservas);

void deletaReserva(structReserva *reserva, structReserva *reservaDados, int *quantidadeReservas, FILE *arqQtdReservas,
                   FILE *arqDadosReservas);

```

j) transacoes.c

```c
int verifCodigoCheckin(structCheckin *checkin, structCheckin *checkinDados, int *quantidadeCheckin);

void cadastrarCheckin(structCheckin *checkin, structCheckin *checkinDados, int *quantidadeCheckin);

void gerarArqChecki(structCheckin *checkinDados, int quantidadeCheckin, FILE *arqQtdCheckin, FILE *arqDadosCheckin);
```

k) vendas.c

feedback.h

a)

```c
void csvHospede(structHospede *hospede, int qtd);

void csvAcomodacoes(structAcomodacao *a, int qtd);

void csvReservas(structReserva *r, int qtd);

void csvProdutosConsumo(structProdutosConsumo *p, int qtd);

void csvCategorias(structCategoriaAcomodacao *c, int qtd);

```

2. Struct por arquivo

a) acomodacao.h

```c
typedef struct structAcomodacao
{
    int codigo;
    char descricao[CHAR_G];
    char facilidades[CHAR_G];
    int categoria; // verificar atraves do codigo na struct categoriaAcomodacao
} structAcomodacao;
```

b) categoriaAcomodacao.h

```c
typedef struct structCategoriaAcomodacao
{
    int codigo;
    char descricao[CHAR_G];
    float valorDiaria;
    int quantMaxDePessoas;
} structCategoriaAcomodacao;
```

c) feedback.h

Não possui structs

d) fornecedor.h

```c
typedef struct structFornecedores
{
    int codigo;
    char nomeFantasia[CHAR_G];
    char razaoSocial[CHAR_G];
    char inscricaoEstadual[CHAR_P];
    char cnpj[CHAR_P];
    structEndereco endereco;
    char telefone[CHAR_P];
    char email[CHAR_G];
} structFornecedores;
```

e) horaData.h

Não possui structs

f) hospede.h

```c
typedef struct structHospede
{
    int codigo;
    char nome[CHAR_G];
    structEndereco endereco;
    char cpf[CHAR_P];
    char telefone[CHAR_P];
    char email[CHAR_G];
    char sexo[CHAR_P];
    char estadoCivil[CHAR_P];
    int diaNasc, mesNasc, anoNasc;
} structHospede;
```

g) hotel.h

```c
typedef struct structEndereco
{
    char numero[8];
    char rua[CHAR_M];
    char bairro[CHAR_P];
    char cidade[CHAR_M];
} structEndereco;

typedef struct structHotel
{
    char nomeFantasia[CHAR_G];
    char razaoSocial[CHAR_G];
    char inscricaoEstadual[CHAR_P];
    char cnpj[CHAR_P];
    structEndereco endereco;
    char telefone[CHAR_P];
    char email[CHAR_G];
    char nomeResponsavel[CHAR_P];
    char telefoneResponsavel[CHAR_P];
    int horaCheckin, minutoCheckin;
    int horaCheckout, minutoCheckout;
    float margemLucro;
} structHotel;
```

h) operadorSistema.h

```c
typedef struct structOperadoresSistema
{
    int codigo;
    char nome[CHAR_G];
    char usuario[CHAR_P];
    char senha[CHAR_P];
    int permissoes;
} structOperadoresSistema;
```

i) produtosDisponiveis.h

```c
typedef struct structProdutosConsumo
{
    int codigo;
    char descricao[CHAR_G];
    int estoque;
    int estoqueMinimo;
    float precoCusto;
    float precoVenda;
} structProdutosConsumo;
```

j) reserva.h

```
15  typedef struct structReserva{
16      int codigo;
17      int diaEntrada;
18      int mesEntrada;
19      int anoEntrada;
20      int diaSaida;
21      int mesSaida;
22      int anoSaida;
23      int codigoAcomodacao;
24      int codigoHospede;
25  } structReserva;
```

k) transacoes.h

```
15  typedef struct structCheckin
16   {
17      int codigo;
18      int codigoReserva;
19      int formaPagamento; // 1 check in ou 2 check out
20      float valorDiaria;
21      float totalConta;
22  } structCheckin;
```

l) vendas.h

```
15  typedef struct structVendas{
16      int quantidadeProdutos;
17      float valorProduto;
18      float totalTemp;
19      float valorTotalProtudos;
20      int codigoCheckin;
21  } structVendas;
```

3. Informações sobre as funções

a) Funções que iniciam com **Cadastro**:
Recebem os dados de entrada e as aloca em um ponteiro.

b) Funções que iniciam com **Verif ou Verificar**:
Recebem o codigo ou nome como entrada e verifica se o cadastro já foi realizado. Retorna 1 caso já esteja no sistema ou 0 caso não tenha sido cadastrado.

c) Funções que iniciam com **GerarArq**:
Usa-se os dados de entrada para gerar um arquivo binário e um arquivo CSV para futuras implementações de leitura. O programa em si faz apenas a leitura de arquivos binários.

d) Funções que iniciam com **Exibe**:
Recebe um codigo de entrada e com ele retorna todos os dados correspondente a ele.

e) Funções que iniciam com **Listar**:
Lista na tela do usuario todos os itens ja cadastrados.

f) Funções que iniciam com **Atualizar**:
Com base no codigo de entrada (é inalterado e unico) abre-se a opção de editar os campos já cadastrados.

g) Funções que iniciam com **Deletar**:
Recebe um codigo de entrada e com isso abre a opção de deletar o item correspondente. Obs. Deleta do arquivo binario e CSV também.