Documentação Técnica Hotéis Molezinha

Lucas Mateus Fernandes Leandro Soulza Pinheiro

11 de Dezembro de 2017

Sumário

1 #Arquivos de cabeçalho (Include)

Os arquivos de cabeçalho se estruturam em 4 pastas: *Pasta Raiz*, *Importação_Exportacao*, *Modulo_FeedBack*, *Módulo_Registro* e *Módulo_Registro*.

1.1 Pasta_Raiz

São os Arquivos e diretorios Dentro de *Arquivos_Cabecalho*.
Registros.h Possui todas os *enum* e as *structs* com os respectivos *typdef*.
Prototipos.h Possui Todos os prototipos de funções.
Funcoes.h Possui as funções mais gerais que não são específicas de nenhum módulo.

1.2 Modulo_Feedback

Feedback.h Voltado para gerar o .csv

1.3 Importacao_Exportacao

Importacao.h Voltado para Importação e Exportação de arquivos .xml

1.4 Modulo_Reserva

Reserva.h e Pesquisa.h São arquivos voltados para a Pesquisa e o registro de reservas

1.5 Modulo_Registro

Hotel.h, Hospede.h, Codigo_Categoria.h, Acomodacoes.h, Login.h, Funcionarios.h, Fornecedores.h, Produtos.h ,São arquivos que possuêm as funções especificas para registar determinada classe

2 Funções por Arquivo

2.1 Funcoes.h

```
int MenuInicialFeedback();
int MenuListagemFeedback();
void Cria_Pasta(char Url[99]);
int Opcao_Acoes();
int Main_All();
void Apagar_Modificar(char Url[99], int Codigo,int Modificar
    ,MODO Modo,int Registro);
int Confirmacao();
int Intervalo_Vetor(int Vetor[],int Ultimo);
int Modo_Manipulacao();
void Quick_Sort(int vetor[], int inicio, int fim);
int Retorna_Linha_Codigo(char Url[99], int Codigo);
int Valida_Codigo(char Url[99],int Numero_De_Registros,int
    Modo_de_Abertura, int Tipo_DADO);
void Verificacao_Arquivo(char Url[99],int Modo_de_Abertura);
MODO Modo_Bin_ou_Txt(int Modo_de_Abertura);
int Converter_Decimal_Binario(int n0,int n1,int n2,int n3);
int Arquivo_Texto_Vazio(char Url[]);
int Arquivo_Binario_Vazio(char Url[]);
void Verificacao_All();
int Ler_Configuracoes_Retorna_Modo_de_Abertura();
int Valida_Codigo_Produto(int Codigo, int Modo_de_Abertura);
void Recebe_Dados_Produtos(int Codigo[],int Quantidade[],int
     Pagamento[]);
```

2.2 Hotel.h

```
int Retorna_Campo_Struct_Hotel(char Url[99], int Codigo);
void Apagar_Modificar_Hotel_Bin(char Url[99], int Codigo,int
     Modificar,MODO Modo);
void Criar_Modificar_Hotel(int Modo_de_Abertura,int
    Manter_Codigo);
void Gravar_Hotel_Bin(char Url[99],DADOS_HOTEL *Hotel);
void Gravar_Hotel_Txt(char Url[99],DADOS_HOTEL *Hotel);
void Ler_Hotel_Bin();
void Ler_Hotel_Memoria(DADOS_HOTEL Hotel);
void Ler_Hotel_Txt(char Url[99]);
void Main_Hotel(MODO Modo);
int Valida_Acomadacao_Hotel(int Codigo, int Modo_de_Abertura
    );
```

2.3 Hospede.h

```
int Retorna_Campo_Struct_Hospede(char Url[99], int Codigo);
void Apagar_Modificar_Hospede_Bin(char Url[99], int Codigo,
    int Modificar,MODO Modo);
```

```
void Criar_Modificar_Hospede(int Modo_de_Abertura,int
    Manter_Codigo);
void Gravar_Hospede_Bin(char Url[99],DADOS_HOSPEDE *Hospede)
    ;
void Gravar_Hospede_Txt(char Url[99],DADOS_HOSPEDE *Hospede)
    ;
void Ler_Hospede_Bin();
void Ler_Hospede_Memoria(DADOS_HOSPEDE Hospede);
void Ler_Hospede_Txt(char Url[99]);
void Main_Hospede(MODO Modo);
void Nome_Hospede_Codigo(int Codigo, char Nome_Hospede[]);
```

2.4 Codigo_Categoria.h

```
int Retorna_Campo_Struct_Codigo_Categoria(char Url[99], int
    Codigo);
void Apagar_Modificar_Codigo_Categoria_Bin(char Url[99], int
     Codigo,int Modificar,MODO Modo);
void Criar_Modificar_Codigo_Categoria(int Modo_de_Abertura,
    int Manter_Codigo);
void Gravar_Codigo_Categoria_Bin(char Url[99],
    CODIGO_CATEGORIA *Codigo_Categoria);
void Gravar_Codigo_Categoria_Txt(char Url[99],
    CODIGO_CATEGORIA *Codigo_Categoria);
void Ler_Codigo_Categoria_Bin();
void Ler_Codigo_Categoria_Memoria(CODIGO_CATEGORIA
    Codigo_Categoria);
void Ler_Codigo_Categoria_Txt(char Url[99]);
void Main_Codigo_Categoria(MODO Modo);
int Valida_Acomadacao_Codigo_Categoria(int Codigo, int
    Modo_de_Abertura);
```

2.5 Acomodacoes.h

```
int Valida_Codigo_Hotel(int Codigo, int Modo_de_Abertura);
int Retorna_Campo_Struct_Acomodacoes(char Url[99], int
    Codigo);
int Valida_Codigo_Categoria_Acomodacoes(int Codigo, int
    Modo_de_Abertura);
void Apagar_Modificar_Acomodacoes_Bin(char Url[99], int
    Codigo,int Modificar,MODO Modo);
void Criar_Modificar_Acomodacoes(int Modo_de_Abertura,int
    Manter_Codigo);
void Gravar_Acomodacoes_Bin(char Url[99],ACOMODACOES *
    Acomodacoes);
void Gravar_Acomodacoes_Txt(char Url[99],ACOMODACOES *
    Acomodacoes);
void Ler_Acomodacoes_Bin();
void Ler_Acomodacoes_Memoria(ACOMODACOES Acomodacoes);
void Ler_Acomodacoes_Txt(char Url[99]);
```

```
void Recebe_Dados_Acomodacoes(ACOMODACOES *Acomodacoes, int
    Modo_de_Abertura);
void Main_Acomodacoes(MODO Modo);
```

2.6 Login.h

```
int Login(MODO Modo);
void Criptografar(char Senha[]);
void Descriptografar(char Senha[]);
```

2.7 Funcionarios.h

```
void Ler_Funcionarios_Txt(char Url[99]);
void Ler_Funcionarios_Bin();
void Ler_Funcionarios_Memoria(FUNCIONARIOS Funcionarios);
void Gravar_Funcionarios_Txt(char Url[99],FUNCIONARIOS *
    Funcionarios);
void Gravar_Funcionarios_Bin(char Url[99],FUNCIONARIOS *
    Funcionarios);
void Criar_Modificar_Funcionarios(int Modo_de_Abertura,int
    Manter_Codigo);
int Retorna_Campo_Struct_Funcionarios(char Url[99], int
    Codigo);
void Apagar_Modificar_Funcionarios_Bin(char Url[99], int
    Codigo,int Modificar,MODO Modo);
```

2.8 Fornecedores.h

```
int Retorna_Campo_Struct_Fornecedores(char Url[99], int
    Codigo);
void Apagar_Modificar_Fornecedores_Bin(char Url[99], int
    Codigo,int Modificar,MODO Modo);
void Criar_Modificar_Fornecedores(int Modo_de_Abertura,int
    Manter_Codigo);
void Gravar_Fornecedores_Bin(char Url[99],FORNECEDORES *
    Fornecedores);
void Gravar_Fornecedores_Txt(char Url[99],FORNECEDORES *
    Fornecedores);
void Ler_Fornecedores_Bin();
void Ler_Fornecedores_Memoria(FORNECEDORES Fornecedores);
void Ler_Fornecedores_Txt(char Url[99]);
void Main_Fornecedores(MODO Modo);
```

2.9 Produtos.h

```
void Ler_Produtos_Txt(char Url[99]);
void Ler_Produtos_Bin();
void Ler_Produtos_Memoria(PRODUTOS Produtos);
void Gravar_Produtos_Txt(char Url[99],PRODUTOS *Produtos);
void Gravar_Produtos_Bin(char Url[99],PRODUTOS *Produtos);
```

```
void Criar_Modificar_Produtos(int Modo_de_Abertura,int
    Manter_Codigo);
int Retorna_Campo_Struct_Produtos(char Url[99], int Codigo);
void Apagar_Modificar_Produtos_Bin(char Url[99], int Codigo,
    int Modificar,MODO Modo);
int Valida_Codigo_Hotel_Produtos(int Codigo, int
    Modo_de_Abertura);
```

2.10 Reserva.h

```
int Retorna_Campo_Struct_Reserva(char Url[99], int Codigo);
void Apagar_Modificar_Reserva_Bin(char Url[99], int Codigo,
    int Modificar,MODO Modo);
void Criar_Modificar_Reserva(int Modo_de_Abertura,int
    Manter_Codigo);
void Gravar_Reserva_Bin(char Url[99],RESERVA *Reserva);
void Gravar_Reserva_Txt(char Url[99],RESERVA *Reserva);
void Ler_Reserva_Bin();
void Ler_Reserva_Memoria(RESERVA Reserva);
void Ler_Reserva_Txt(char Url[99]);
void Main_Reserva(MODO Modo);
int Valida_Acomadacao_Reserva(int Codigo, int
    Modo_de_Abertura);
void Mostra_Se_Conta_Paga(int Pago);
void Modo_De_Pagamento(int Modo);
int Valida_Hospede_Reserva(int Codigo, int Modo_de_Abertura)
    ;
void DebugFluxo(char Url[99], FLUXO *Fluxo);
void Apagar_Fluxo(char Url[999], int Codigo);
int Retorna_Campo_Struct_Fluxo(char Url[99], int Codigo);
void Arquivo_Url_Fluxo(char Url[99], int Codigo,char
    Url_Fluxo[]);
int Retorna_Codigos_Reserva(int Codigos[]);
```

2.11 Pesquisa.h

```
void Main_Pesquisa();
int Verifica_Fluxo(char Url[999], DATA Data_Entrada,DATA
    Data_Saida, int Acomodacao_Indisponiveis[]);
PESQUISA Tipo_Pesquisa();
DATA Pesquisa(PESQUISA Pesquisa,int *Indice_Disponiveis,int
    Vetor_Cod_Acomodacao_Disponivel[]);
int Retorna_Acomodacao_Disponiveis_Por_Periodo(int
    Acomodacao_Disponiveis[],PESQUISA Pesquisa_Entrada,
    PESQUISA Pesquisa_Saida);
void Pequisa_Quantidade(int Acomodacao_Invalida[],int
    Inicio_Vetor,int Quantidade);
void Pequisa_Facilidades(int Acomodacao_Invalida[],int
    Inicio_Vetor);
```

```
void Pequisa_Categoria_Acomodacao(int Acomodacao_Invalida[],
    int Inicio_Vetor);
int Valida_Codigo_Acomodacao(int Codigo, int
    Modo_de_Abertura);
void Recebe_Data(DATA *Data, int Auxiliar);
int Todas_Acomodacoes_TXT(char Url[99], int
    Acomodacoes_Disponiveis[],int Acomodacoes_Indisponiveis
    [], int Contador_Acomodacao_Indisponiveis);
void Mostra_Acomodacoes_TXT(int Contador_Acomodacoes, int
    Codigos[], char Url[]);
void Mostra_Acomodacoes_BIN(int Contador_Acomodacoes, int
    Codigos[], char Url[]);
int Todas_Acomodacoes_BIN(char Url[99], int
    Acomodacoes_Disponiveis[],int Acomodacoes_Indisponiveis
    [], int Contador_Acomodacao_Indisponiveis);
int Retorna_Acomodacoes_Indisponiveis_Com_Codigo_Categoria(
    int Codigo_Categoria, int Modo_Abertura, int
    Acomodacao_Indisponiveis[], int
    Contador_Acomodacao_Indisponiveis);
int Retorna_Acomodacoes_Indisponiveis_Com_Quantidade_Pessoas
    (int Adultos, int Criancas,int
    Codigo_Acomodacao_Invalidas[], int Indice_Invalido);
int Retorna_Acomodacoes_Indisponiveis_Com_Facilidades(
    FACILIDADES Facilidade,int Codigo_Acomodacao_Invalidas[],
     int Indice_Invalido);
```

2.12 Importar/Exportar XML.h

```
void Exportar();
IMPORTACAO_EXPORTACAO Set_On_Off();
```

2.13 FeedBack.h

```
void MainFeedback();
int Tipo_Listagem_Hospede();
void Filtro_Hospede_Codigos(int Modo_Feedback);
void Filtro_Hospede_Sexo(int Modo_Feedback);
int Tipo_Listagem_Acomodacao();
void Filtro_Acomodacao_Codigos(int Modo_Feedback);
void Gera_CSV_Acomodacoes_TXT(int Contador_Acomodacoes, int
    Codigos[], char Url[]);
void Gera_CSV_Acomodacoes_BIN(int Contador_Acomodacoes, int
    Codigos[], char Url[]);
void Filtro_Acomodacao_Data_Disponivel(int Modo_Feedback);
void Filtro_Acomodacao_CodCategoria(int Modo_Feedback);
int Tipo_Listagem_Reserva();
void Filtro_Reserva_Data(int Modo_Feedback);
void Filtro_Reserva_Codigo_Acomodacao(int Modo_Feedback);
void Filtro_Reserva_Codigo_Hospede(int Modo_Feedback);
int Tipo_Listagem_Produtos();
```

```
void Filtro_Produtos_Codigos(int Modo_Feedback);
void Filtro_Produtos_Codigos_Em_Estoque_Minimo(int
    Modo_Feedback);
```

3 Informações sobre Funções

3.1 Cria_Pasta

Função recebe por parametro um char de 99 posições, e usa um comando específico do Linux para criar a pasta referente ao nome que foi recebido por parametro.

3.2 Verificacao_All

Função responsável por verificar se existe os arquivos de registro, tanto para arquivos com extensão .bin quanto para extensão .txt, caso esses arquivos não existam são criados através da função Verificacao_ Arquivo.

3.3 OrdenaValoresTxt

Função responsável por ordenar o conteúdo dos arquivos com extensão .txt, pois como toda linha começa com o codigo do dado que está sendo salvo é usado um comando interno do Linux para ordenar todo o arquivo de texto. Porém antes de realizar a ordenação a função chama a função de verificar se o arquivo que está sendo ordenado existe pois caso não exista pode causar erros.

3.4 Configuracoes

Função responsável por verificar se o arquivo possui a extensão .txt ou .bin, na função é verificado se existe o arquivo Configuracoes.txt, pois ele indica onde os dados estão sendo salvos, ou seja se é arquivo do tipo binário ou texto. Ao final da função é retornado o modo de manipulação (binário ou texto).

3.5 Arquivo_Texto_Vazio

A função recebe por referência um char contendo a url em que que o arquivo está salvo, feito isso o arquivo é aberto em modo de leitura, para que assim possa ser verificado se o arquivo está vazio utilizando uma função interna do próprio C a feof(), se caso estiver no fim do arquivo já retorna sem incrementar no contador, se passar pelo feof() e não estiver no fim do arquivo é incrementado 1 no contador. Ao final da função o arquivo é fechado e verificado se o contador está igual a 0, se estiver zerado indica que o arquivo está vazio e então a função retorna 1, caso contrário é retornado 0 indicando que o arquivo não está vazio.

3.6 Arquivo_Binario_Vazio

A função recebe por referência um char contendo a url em que que o arquivo está salvo, feito isso o arquivo é aberto em modo de leitura, para que assim possa ser verificado se o arquivo está vazio utilizando uma função interna do próprio C a feof(), que caso se caso estiver no fim do arquivo já retorna sem incrementar no contador, se passar pelo feof() e não estiver no fim do arquivo é incrementado 1 no contador. Ao final da função o arquivo é fechado e verificado se o contador está igual a 0, se estiver zerado indica que o arquivo está vazio e então a função retorna 1, caso contrário é retornado 0 indicando que o arquivo não está vazio.

3.7 Opcao_Acoes

Função responsável por mostrar ao usuário um menu com as seguites ações *Criar, Editar, Apagar*, após mostrar o menu o usuário é lido para qual ação ele deseja realizar e é retornado essa ação. Caso seja digitado uma opção inválida é mostrado para o usuário que ele deve inserir uma opção válida e é repetido o menu.

3.8 Opcao_Acoes_Reserva

Função semelhante a função de ações diversas *Opcao_Acoes*, ela é responsável por mostrar ao usuário um menu com as seguites ações *Ler, Criar, Editar, Apagar*, após mostrar o menu para o usuário é lido qual ação ele deseja realizar e é retornado essa ação. Caso seja digitado uma opção inválida é mostrado para o usuário que ele deve inserir uma opção válida e é repetido o menu.

3.9 Modulo

Função semelhante a função de ações diversas *Opcao_Acoes*, ela é responsável por mostrar ao usuário um menu com as seguites ações *Registros, Reservas, Pesquisas, FeedBack, Importar/Exportar*, após mostrar o menu para o usuário é lido para qual das opções ele deseja ir e é retornado essa opção. Caso seja digitado uma opção inválida é mostrado para o usuário que ele deve inserir uma opção válida e é repetido o menu.

3.10 Main_All

Função semelhante a função de ações diversas *Opcao_Acoes*, ela é responsável por mostrar ao usuário um menu com as seguites ações *Hotel, Hospedes, Acomodações, Categoria, Produtos, Fornecedores, Funcionários*, após mostrar o menu para o usuário é lido para qual das opções ele deseja ir e é retornado essa opção. Caso seja digitado uma opção inválida é mostrado para o usuário que ele deve inserir uma opção válida e é repetido o menu.

3.11 Apagar_Modificar

A função recebe um char contendo a URL em que o dado será apagado está e o código que o dado está, a função procura pelo código do dado que sera apagao ou modificado e leva o ponteiro até essa linha, feito isso é mostrado uma menssagem de confirmação de exclusão ou edição, caso o usuário confirme o dado é apagado ou editado. Para que isso ocorra é copiado para outro arquivo todos os dados anteriores e editado/apagado o dado referente ao código passado por parâmetro e copiado o restante dos dados para o arquivo temporário, após todos os dados serem copiados o arquivo temporário é renomeado e o arquivo anterior a edição ou deleção é apagado. Vale ressaltar que a função é modular ou seja, ela funciona para todos os módulos referente a arquivos com extensão .txt.

3.12 Confirmacao

Função responsável por verificar se o usuário deseja realmente realizar a ação que ele pretende, pois assim que a ação for realizada será impossivel reverter o dado.

3.13 Intervalo_Vetor

Função responsável por verificar se o vetor passado por referência existe intervalos entre um código e outro, feito isso ela retorna em que posição esta o intervalo e caso não exista intervalo é retornado -1.

3.14 Modo_Manipulacao

Função responsável por verificar em qual formato os dados serão gravados, as opções que o usuário irá visualizar são: *Arquivo Texto, Arquivo Binário, Banco de Dados, Núvem.* Porém somente o arquivo texto e binário foram implentados, caso seja escolhida outra opção é mostrado para o usuário que aquela opção não foi implementada.

3.15 Quick_Sort

Função responsável por receber por referência um vetor, o inicio e o fim do mesmo para poder ordena-lo em ordem crescente.

3.16 Retorna_Linha_Codigo

Função recebe um char contendo a URL e o código para que possa abrir o arquivo e retornar em qual linha aquele arquivo se encotra.

3.17 Valida_Codigo

Função responsável por receber um código e verificar se ele existe, para seu funcionamento é recibido por parametro um char contendo a URL que o código está cadastrado, o modo de abertura (Texto ou Binário) e o tipo de dado que será validado o código.

3.18 Verificacao_Arquivo

Função responsável por receber por referência uma url contendo o local do arquivo e verificar se ele exista, caso o arquivo não exista ele é criado.

3.19 Modo_Bin_ou_Txt

Função responsável por escrever na struct o modo de abertura de cada tipo de dado, ou seja é escrito na struct MODO de abertura do arquivo, se é para escrita, leitura ou concatenação. Vale ressaltar que a função funciona tanto para arquivos texto quanto para arquivos binários.

3.20 Converter_Decimal_Binario

Função responsável por receber um valor binário de quatro digitos e converte-lo em decimal.

3.21 MenuInicialFeedback

Função responsável por mostrar o menu inicial do módulo de FeedBack para o usuário. Vale ressaltar que caso o usuário digite uma opção invalida é mostrado para ele que não é uma opção válida e repete o menu.

3.22 Ler_Configuracoes_Retorna_Modo_de_Abertura

Função responsável por abrir o arquivo de configurações e retornar se os dados estão sendo salvos em txt ou bin.

3.23 Valida_Codigo_Produto

Função responsável por abrir o arquivo de produtos e verificar se o código recebido por paramêtro esta salvo dentro do arquivo, retornando 1 para verdadeiro e 0 para falso.

3.24 Ler_Hotel_Memoria

Função responsavel para mostrar no terminal o conteudo da struct DADOS_HOTEL passada por refêrencia.

3.25 Gravar_Hotel_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo concatenação de texto e salva a struct DADOS_HOTEL no arquivo, é importante ressaltar que a struct foi passada por referência e não por valor.

3.26 Ler_Hotel_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo leitura de texto lé o seu conteudo de linha em linha (o que equivale a uma struct) e mostra seu conteúdo na tela. É importante ressaltar que esta função percorre todo o arquivo.txt o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a função *Ler_Hotel_Memoria* passando uma struct DADOS HOTEL por vez.

3.27 Recebe_Dados_Hotel

A função recebe uma struct DADOS_HOTEL por referência e prenche com valores digitados pelo usuário.

3.28 Main_Hotel

A função verifica se os arquivos referente ao backup do hotel existe caso não exista ela os cria. Sempre que a função é chamada ela ordena o conteudo do hotel.txt por meio da função *OrdenaValoresTxt* e pede para que o usuário escolha uma ação:(*Ler, Criar, Editar, Apagar*) e a chama caso o o funcionário tenha o nivel de permisão adequado para realizar tal função.

3.29 Ler_Hotel_Bin

Muito semelhante a *Ler_Hotel_Txt*, a função abre o arquivo pela url que foi passada por parâmetro em modo leitura de arquivos binários e lé o conteudo referente a uma struct e mostra seu conteudo na tela.é importante ressaltar que esta função percorre todo o arquivo.bin o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a função *Ler_Hotel_Memoria* passando uma struct DADOS HOTEL por vez.

3.30 Gravar_Hotel_Bin

Muito semelhante a *Gravar_Hotel_Txt*, esta função abre o arquivo pela url que foi passada por parâmetro em modo concatenação binária e salva a struct DADOS_HOTEL no arquivo, é importante ressaltar que a struct foi passada por referência e nã ouo por valor.

3.31 Valida_Acomadacao_Hotel

Esta função recebe um codigo por parametro e o pesquisa dentro de um arquivo .txt ou .bin, caso o codigo exista no arquivo é retornado **1** e caso não exista é retornado **0**

3.32 Criar_Modificar_Hotel

Esta função verifica qual o menor numero natural não utilizado como codigo e o usa para o codigo do novo hotel apos isto chama a função *Recebe_Dados_Hotel* para prencher a struct e e por fim ela salva a struct no arquivo atraves da função *Gravar_Hotel_Txt* ou *Gravar_Hotel_Bin*

3.33 Retorna_Campo_Struct_Hotel

Esta função le todas as structs do arquivo binário e retorna qual a posição da struct que possui o codigo passado por parametro,caso retorne **-1** é porque não há hoteis com este código.

3.34 Apagar_Modificar_Hotel_Bin

Está função abre um arquivo binario que teve sua url passada por parametro, alem disso eśtá função recebe um codigo por parametro e uma flag. a Funcionalidade principal destá função é transferir todos os dados de struct para um arquivo temporario exceto a struct com o codigo recebido. caso a flag esteja ligada no momento que o codigo é encontrado é chamada a função *Criar_Modificar_Hotel* para repreencher a struct . apos a transferencia de todos os dados o arquivo original *Hotel.bin* é apagado e o Temp é renomeado para *Hotel.bin*

3.35 Ler_Hospede_Memoria

Função responsavel para mostrar no terminal o conteudo da struct DADOS_HOSPEDE passada por refêrencia.

3.36 Gravar_Hospede_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo concatenação de texto e salva a struct DADOS_HOSPEDE no arquivo, é importante ressaltar que a struct foi passada por referência e não por valor.

3.37 Ler_Hospede_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo leitura de texto lé o seu conteudo de linha em linha (o que equivale a uma struct) e mostra seu conteúdo na tela. É importante ressaltar que esta função percorre todo o arquivo.txt o que faz com que todos os dados salvos no arquivo sejam

exibidos. e para mostrar na tela chama a função *Ler_Hospede_Memoria* passando uma struct DADOS Hospede por vez.

3.38 Recebe_Dados_Hospede

A função recebe uma struct DADOS_HOSPEDE por referência e prenche com valores digitados pelo usuário.

3.39 Main_Hospede

A função verifica se os arquivos referente ao backup do Hospede existe caso não exista ela os cria. Sempre que a função é chamada ela ordena o conteudo do Hospede.txt por meio da função *OrdenaValoresTxt* e pede para que o usuário escolha uma ação:(*Ler, Criar, Editar, Apagar*) e a chama caso o o funcionário tenha o nivel de permisão adequado para realizar tal função.

3.40 Ler_Hospede_Bin

Muito semelhante a *Ler_Hospede_Txt*, a função abre o arquivo pela url que foi passada por parâmetro em modo leitura de arquivos binários e lé o conteudo referente a uma struct e mostra seu conteudo na tela.é importante ressaltar que esta função percorre todo o arquivo.bin o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a função *Ler_Hospede_Memoria* passando uma struct DADOS Hospede por vez.

3.41 Gravar_Hospede_Bin

Muito semelhante a *Gravar_Hospede_Txt*, esta função abre o arquivo pela url que foi passada por parâmetro em modo concatenação binária e salva a struct DADOS_HOSPEDE no arquivo, é importante ressaltar que a struct foi passada por referência e nã ouo por valor.

3.42 Criar_Modificar_Hospede

Esta função verifica qual o menor numero natural não utilizado como codigo e o usa para o codigo do novo Hospede apos isto chama a função *Recebe_Dados_Hospede* para prencher a struct e e por fim ela salva a struct no arquivo atraves da função *Gravar_Hospede_Txt* ou *Gravar_Hospede_Bin*

3.43 Retorna_Campo_Struct_Hospede

Esta função le todas as structs do arquivo binário e retorna qual a posição da struct que possui o codigo passado por parametro,caso retorne **-1** é porque não há hoteis com este código.

3.44 Apagar_Modificar_Hospede_Bin

Está função abre um arquivo binario que teve sua url passada por parametro, alem disso eśtá função recebe um codigo por parametro e uma flag. a Funcionalidade principal destá função é transferir todos os dados de struct para um arquivo temporario exceto a struct com o codigo recebido. caso a flag esteja ligada no momento que o codigo é encontrado é chamada a função *Criar_Modificar_Hospede* para repreencher a struct . apos a transferencia de todos os dados o arquivo original *Hospede.bin* é apagado e o Temp é renomeado para *Hospede.bin*

3.45 Nome_Hospede_Codigo

Está função recebe um codigo por parametro, abre o arquivo Hospede em modo leitura (*.bin ou .txt*) e procura pelo nome do hospede que contem codigo informado retornando uma string (modifca um vetor que foi passado por referencia);

3.46 Ler_Codigo_Categoria_Memoria

Função responsavel para mostrar no terminal o conteudo da struct CODIGO_CATEGORIA passada por refêrencia.

3.47 Gravar_Codigo_Categoria_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo concatenação de texto e salva a struct CODIGO_CATEGORIA no arquivo, é importante ressaltar que a struct foi passada por referência e não por valor.

3.48 Ler_Codigo_Categoria_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo leitura de texto lé o seu conteudo de linha em linha (o que equivale a uma struct) e mostra seu conteúdo na tela. É importante ressaltar que esta função percorre todo o arquivo.txt o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a função *Ler_Codigo_Categoria_Memoria* passando uma struct DADOS Codigo_Categoria por vez.

3.49 Recebe_CODIGO_CATEGORIA

A função recebe uma struct CODIGO_CATEGORIA por referência e prenche com valores digitados pelo usuário.

3.50 Main_Codigo_Categoria

A função verifica se os arquivos referente ao backup do Codigo_Categoria existe caso não exista ela os cria. Sempre que a função é chamada ela ordena o conteudo do Codigo_Categoria.txt por meio da função *OrdenaValoresTxt* e pede para que

o usuário escolha uma ação:(*Ler, Criar, Editar, Apagar*) e a chama caso o o funcionário tenha o nivel de permisão adequado para realizar tal função.

3.51 Ler_Codigo_Categoria_Bin

Muito semelhante a *Ler_Codigo_Categoria_Txt*, a função abre o arquivo pela url que foi passada por parâmetro em modo leitura de arquivos binários e lé o conteudo referente a uma struct e mostra seu conteudo na tela.é importante ressaltar que esta função percorre todo o arquivo.bin o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a função *Ler_Codigo_Categoria_Memoria* passando uma struct DADOS Codigo_Categoria por vez.

3.52 Gravar_Codigo_Categoria_Bin

Muito semelhante a *Gravar_Codigo_Categoria_Txt*, esta função abre o arquivo pela url que foi passada por parâmetro em modo concatenação binária e salva a struct CODIGO_CATEGORIA no arquivo, é importante ressaltar que a struct foi passada por referência e nã ouo por valor.

3.53 Criar_Modificar_Codigo_Categoria

Esta função verifica qual o menor numero natural não utilizado como codigo e o usa para o codigo do novo Codigo_Categoria apos isto chama a função *Recebe_CODIGO_CATEGORIA* para prencher a struct e e por fim ela salva a struct no arquivo atraves da função *Gravar_Codigo_Categoria_Txt* ou *Gravar_Codigo_Categoria_Bin*

3.54 Retorna_Campo_Struct_Codigo_Categoria

Esta função le todas as structs do arquivo binário e retorna qual a posição da struct que possui o codigo passado por parametro,caso retorne **-1** é porque não há Codigo de categorias com este numero.

3.55 Apagar_Modificar_Codigo_Categoria_Bin

Está função abre um arquivo binario que teve sua url passada por parametro, alem disso eśtá função recebe um codigo por parametro e uma flag. a Funcionalidade principal destá função é transferir todos os dados de struct para um arquivo temporario exceto a struct com o codigo recebido. caso a flag esteja ligada no momento que o codigo é encontrado é chamada a função *Criar_Modificar_Codigo_Categoria* para repreencher a struct . apos a transferencia de todos os dados o arquivo original *Codigo_Categoria.bin* é apagado e o Temp é renomeado para *Codigo_Categoria.bin*

3.56 Valida_Acomadacao_Codigo_Categoria

Esta função recebe um codigo por parametro e o pesquisa dentro do arquivo Codigo_Categoria *.txt ou .bin*, caso o codigo exista no arquivo é retornado **1** e caso não exista é retornado **0**.

3.57 Ler_Acomodacoes_Memoria

Função responsavel para mostrar no terminal o conteudo da struct Acomodacoes passada por refêrencia.

3.58 Gravar_Acomodacoes_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo concatenação de texto e salva a struct Acomodacoes no arquivo, é importante ressaltar que a struct foi passada por referência e não por valor.

3.59 Ler_Acomodacoes_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo leitura de texto lé o seu conteudo de linha em linha (o que equivale a uma struct) e mostra seu conteúdo na tela. É importante ressaltar que esta função percorre todo o arquivo.txt o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a função *Ler_Acomodacoes_Memoria* passando uma struct DADOS Acomodacoes por vez.

3.60 Recebe_Acomodacoes

A função recebe uma struct Acomodacoes por referência e prenche com valores digitados pelo usuário.

3.61 Main_Acomodacoes

A função verifica se os arquivos referente ao backup do Acomodacoes existe caso não exista ela os cria. Sempre que a função é chamada ela ordena o conteudo do Acomodacoes.txt por meio da função *OrdenaValoresTxt* e pede para que o usuário escolha uma ação:(*Ler, Criar, Editar, Apagar*) e a chama caso o o funcionário tenha o nivel de permisão adequado para realizar tal função.

3.62 Ler_Acomodacoes_Bin

Muito semelhante a *Ler_Acomodacoes_Txt*, a função abre o arquivo pela url que foi passada por parâmetro em modo leitura de arquivos binários e lé o conteudo referente a uma struct e mostra seu conteudo na tela.é importante ressaltar que esta função percorre todo o arquivo.bin o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a

função *Ler_Acomodacoes_Memoria* passando uma struct DADOS Acomodacoes por vez.

3.63 Gravar_Acomodacoes_Bin

Muito semelhante a *Gravar_Acomodacoes_Txt*, esta função abre o arquivo pela url que foi passada por parâmetro em modo concatenação binária e salva a struct Acomodacoes no arquivo, é importante ressaltar que a struct foi passada por referência e não por valor.

3.64 Criar_Modificar_Acomodacoes

Esta função verifica qual o menor numero natural não utilizado como codigo e o usa para o codigo do novo Acomodacoes apos isto chama a função *Recebe_Acomodacoes* para prencher a struct e e por fim ela salva a struct no arquivo atraves da função *Gravar_Acomodacoes_Txt* ou *Gravar_Acomodacoes_Bin*

3.65 Retorna_Campo_Struct_Acomodacoes

Esta função le todas as structs do arquivo binário e retorna qual a posição da struct que possui o codigo passado por parametro,caso retorne **-1** é porque não há Acomodações com este numero.

3.66 Apagar_Modificar_Acomodacoes_Bin

Está função abre um arquivo binario que teve sua url passada por parametro, alem disso eśtá função recebe um codigo por parametro e uma flag. a Funcionalidade principal destá função é transferir todos os dados de struct para um arquivo temporario exceto a struct com o codigo recebido. caso a flag esteja ligada no momento que o codigo é encontrado é chamada a função *Criar_Modificar_Acomodacoes* para repreencher a struct . apos a transferencia de todos os dados o arquivo original *Acomodacoes.bin* é apagado e o Temp é renomeado para *Acomodacoes.bin*

3.67 Login

Função responsável por receber o código do usuário, nome de usuário e senha, para que seja efetuado o login com todas as informações e permissões do usuário. Vale ressaltar que como a senha está criptografada é chamado a função para descriptografar a senha.

3.68 Criptografar

Função responsável por receber por paramêtro um char contendo a senha e realizar uma criptografia nessa senha. Para isso é pegado cada posição do vetor e em posições pares é somado 1 no caractere e em impares é subtraído 1.

3.69 Descriptografar

Função responsável por receber por paramêtro um char contendo a senha e realizar uma descriptografia nessa senha. Para isso é pegado cada posição do vetor e em posições pares é subtraído 1 no caractere e em impares é somado 1.

3.70 Ler_Funcionarios_Memoria

Função responsavel para mostrar no terminal o conteudo da struct Funcionarios passada por refêrencia.

3.71 Gravar_Funcionarios_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo concatenação de texto e salva a struct Funcionarios no arquivo, é importante ressaltar que a struct foi passada por referência e não por valor.

3.72 Ler_Funcionarios_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo leitura de texto lé o seu conteudo de linha em linha (o que equivale a uma struct) e mostra seu conteúdo na tela. É importante ressaltar que esta função percorre todo o arquivo.txt o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a função *Ler_Funcionarios_Memoria* passando uma struct FUNCIONARIOS por vez.

3.73 Recebe_Funcionarios

A função recebe uma struct Funcionarios por referência e prenche com valores digitados pelo usuário.

3.74 Main_Funcionarios

A função verifica se os arquivos referente ao backup do Funcionarios existe caso não exista ela os cria. Sempre que a função é chamada ela ordena o conteudo do Funcionarios.txt por meio da função *OrdenaValoresTxt* e pede para que o usuário escolha uma ação:(*Ler, Criar, Editar, Apagar*) e a chama caso o o funcionário tenha o nivel de permisão adequado para realizar tal função.

3.75 Ler_Funcionarios_Bin

Muito semelhante a *Ler_Funcionarios_Txt*, a função abre o arquivo pela url que foi passada por parâmetro em modo leitura de arquivos binários e lé o conteudo referente a uma struct e mostra seu conteudo na tela.é importante ressaltar que esta função percorre todo o arquivo.bin o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a

função *Ler_Funcionarios_Memoria* passando uma struct FUNCIONARIOS por vez.

3.76 Gravar_Funcionarios_Bin

Muito semelhante a *Gravar_Funcionarios_Txt*, esta função abre o arquivo pela url que foi passada por parâmetro em modo concatenação binária e salva a struct Funcionarios no arquivo, é importante ressaltar que a struct foi passada por referência e nã ouo por valor.

3.77 Criar_Modificar_Funcionarios

Esta função verifica qual o menor numero natural não utilizado como codigo e o usa para o codigo do novo Funcionarios apos isto chama a função *Recebe_Funcionarios* para prencher a struct e e por fim ela salva a struct no arquivo atraves da função *Gravar_Funcionarios_Txt* ou *Gravar_Funcionarios_Bin*

3.78 Retorna_Campo_Struct_Funcionarios

Esta função le todas as structs do arquivo binário e retorna qual a posição da struct que possui o codigo passado por parametro,caso retorne **-1** é porque não há Codigo de categorias com este numero.

3.79 Apagar_Modificar_Funcionarios_Bin

Está função abre um arquivo binario que teve sua url passada por parametro, alem disso eśtá função recebe um codigo por parametro e uma flag. a Funcionalidade principal destá função é transferir todos os dados de struct para um arquivo temporario exceto a struct com o codigo recebido. caso a flag esteja ligada no momento que o codigo é encontrado é chamada a função *Criar_Modificar_Funcionarios* para repreencher a struct . apos a transferencia de todos os dados o arquivo original *Funcionarios.bin* é apagado e o Temp é renomeado para *Funcionarios.bin*

3.80 Recebe_Dados_Funcionarios

Está função Recebe do usuário informações para preencher a struct porem antes de fazer sua principal função é verificado se existe algum Codigo categoria já cadastrado

3.81 Ler_Fornecedores_Memoria

Função responsavel para mostrar no terminal o conteudo da struct Fornecedores passada por refêrencia.

3.82 Gravar_Fornecedores_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo concatenação de texto e salva a struct Fornecedores no arquivo, é importante ressaltar que a struct foi passada por referência e não por valor.

3.83 Ler_Fornecedores_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo leitura de texto lé o seu conteudo de linha em linha (o que equivale a uma struct) e mostra seu conteúdo na tela. É importante ressaltar que esta função percorre todo o arquivo.txt o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a função *Ler_Fornecedores_Memoria* passando uma struct FORNECEDORES por vez.

3.84 Recebe_Fornecedores

A função recebe uma struct Fornecedores por referência e prenche com valores digitados pelo usuário.

3.85 Main_Fornecedores

A função verifica se os arquivos referente ao backup do Fornecedores existe caso não exista ela os cria. Sempre que a função é chamada ela ordena o conteudo do Fornecedores.txt por meio da função *OrdenaValoresTxt* e pede para que o usuário escolha uma ação:(*Ler, Criar, Editar, Apagar*) e a chama caso o o funcionário tenha o nivel de permisão adequado para realizar tal função.

3.86 Ler_Fornecedores_Bin

Muito semelhante a *Ler_Fornecedores_Txt*, a função abre o arquivo pela url que foi passada por parâmetro em modo leitura de arquivos binários e lé o conteudo referente a uma struct e mostra seu conteudo na tela.é importante ressaltar que esta função percorre todo o arquivo.bin o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a função *Ler_Fornecedores_Memoria* passando uma struct FORNECEDORES por vez.

3.87 Gravar_Fornecedores_Bin

Muito semelhante a *Gravar_Fornecedores_Txt*, esta função abre o arquivo pela url que foi passada por parâmetro em modo concatenação binária e salva a struct Fornecedores no arquivo, é importante ressaltar que a struct foi passada por referência e nã ouo por valor.

3.88 Criar_Modificar_Fornecedores

Esta função verifica qual o menor numero natural não utilizado como codigo e o usa para o codigo do novo Fornecedores apos isto chama a função *Recebe_Fornecedores* para prencher a struct e e por fim ela salva a struct no arquivo atraves da função *Gravar_Fornecedores_Txt* ou *Gravar_Fornecedores_Bin*

3.89 Retorna_Campo_Struct_Fornecedores

Esta função le todas as structs do arquivo binário e retorna qual a posição da struct que possui o codigo passado por parametro,caso retorne **-1** é porque não há Codigo de categorias com este numero.

3.90 Apagar_Modificar_Fornecedores_Bin

Está função abre um arquivo binario que teve sua url passada por parametro, alem disso eśtá função recebe um codigo por parametro e uma flag. a Funcionalidade principal destá função é transferir todos os dados de struct para um arquivo temporario exceto a struct com o codigo recebido. caso a flag esteja ligada no momento que o codigo é encontrado é chamada a função *Criar_Modificar_Fornecedores* para repreencher a struct . apos a transferencia de todos os dados o arquivo original *Fornecedores.bin* é apagado e o Temp é renomeado para *Fornecedores.bin*

3.91 Recebe_Dados_Fornecedores

Está função Recebe do usuário informações para preencher a struct porem antes de fazer sua principal função é verificado se existe algum Codigo categoria já cadastrado

3.92 Valida_Codigo_Categoria_Acomodacoes

Está função abre o arquivo Codigo_Categoria (*.txt ou .bin*) e pesquisa pelo codigo passado por parâmetro caso encontre o codigo é retornado **1** caso não encontre é retornado (**-1 ou 0**)

3.93 Valida_Codigo_Hotel

Está função abre o arquivo Hotel (*.txt ou .bin*) e pesquisa pelo codigo passado por parâmetro caso encontre o codigo é retornado **1** caso não encontre é retornado (**-1 ou 0**)

3.94 Recebe_Dados_Acomodacoes

Está função Recebe do usuário informações para preencher a struct porem antes de fazer sua principal função é verificado se existe algum Codigo categoria já cadastrado

3.95 Ler_Produtos_Memoria

Função responsavel para mostrar no terminal o conteudo da struct PRODUTOS passada por refêrencia.

3.96 Gravar_Produtos_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo concatenação de texto e salva a struct PRODUTOS no arquivo, é importante ressaltar que a struct foi passada por referência e não por valor.

3.97 Ler_Produtos_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo leitura de texto lé o seu conteudo de linha em linha (o que equivale a uma struct) e mostra seu conteúdo na tela. É importante ressaltar que esta função percorre todo o arquivo.txt o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a função *Ler_Produtos_Memoria* passando uma struct DADOS HOTEL por vez.

3.98 Recebe_Dados_Produtos

A função recebe uma struct PRODUTOS por referência e prenche com valores digitados pelo usuário.

3.99 Main_Produtos

A função verifica se os arquivos referente ao backup do hotel existe caso não exista ela os cria. Sempre que a função é chamada ela ordena o conteudo do hotel.txt por meio da função *OrdenaValoresTxt* e pede para que o usuário escolha uma ação:(*Ler, Criar, Editar, Apagar*) e a chama caso o o funcionário tenha o nivel de permisão adequado para realizar tal função.

3.100 Ler_Produtos_Bin

Muito semelhante a *Ler_Produtos_Txt*, a função abre o arquivo pela url que foi passada por parâmetro em modo leitura de arquivos binários e lé o conteudo referente a uma struct e mostra seu conteudo na tela.é importante ressaltar que esta função percorre todo o arquivo.bin o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a função *Ler_Produtos_Memoria* passando uma struct DADOS HOTEL por vez.

3.101 Gravar_Produtos_Bin

Muito semelhante a *Gravar_Produtos_Txt*, esta função abre o arquivo pela url que foi passada por parâmetro em modo concatenação binária e salva a struct

PRODUTOS no arquivo, é importante ressaltar que a struct foi passada por referência e nã ouo por valor.

3.102 Criar_Modificar_Produtos

Esta função verifica qual o menor numero natural não utilizado como codigo e o usa para o codigo do novo hotel apos isto chama a função *Recebe_Dados_Produtos* para prencher a struct e e por fim ela salva a struct no arquivo atraves da função *Gravar_Produtos_Txt* ou *Gravar_Produtos_Bin*

3.103 Retorna_Campo_Struct_Produtos

Esta função le todas as structs do arquivo binário e retorna qual a posição da struct que possui o codigo passado por parametro,caso retorne **-1** é porque não há hoteis com este código.

3.104 Apagar_Modificar_Produtos_Bin

Está função abre um arquivo binario que teve sua url passada por parametro, alem disso eśtá função recebe um codigo por parametro e uma flag. a Funcionalidade principal destá função é transferir todos os dados de struct para um arquivo temporario exceto a struct com o codigo recebido. caso a flag esteja ligada no momento que o codigo é encontrado é chamada a função *Criar_Modificar_Produtos* para repreencher a struct . apos a transferencia de todos os dados o arquivo original *Produtos.bin* é apagado e o Temp é renomeado para *Produtos.bin*

3.105 Valida_Codigo_Hotel_Produtos

Está função abre o arquivo Hotel (*.txt ou .bin*) e pesquisa pelo codigo passado por parâmetro caso encontre o codigo é retornado **1** caso não encontre é retornado (**-1 ou 0**)

3.106 Ler_Reserva_Memoria

Função responsavel para mostrar no terminal todo o conteudo da struct Reserva passada por refêrencia.

3.107 Mostra_Se_Conta_Paga

Função responsável por receber por paramêtro uma variavel do tipo inteiro referente se a conta foi paga ou não, caso a variavel chegue na função com o valor 0 mostra na tela que a conta está pendente, se não mostra que a conta esta paga.

3.108 Modo_De_Pagamento

Função responsável por receber por paramêtro uma variavel do tipo inteiro referente ao modo de pagamento que cliente ira utilizar, dentro da função é feito um switch para cada modo de pagamento que são: *Dinheiro, Debito, Crédito, Cheque.*

3.109 Gravar_Reserva_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo concatenação de texto e salva a struct Reservas no arquivo, é importante ressaltar que a struct foi passada por referência e não por valor.

3.110 Ler_Reserva_Txt

A função abre o arquivo pela url que foi passada por parâmetro em modo leitura de texto lé o seu conteudo de linha em linha (o que equivale a uma struct) e mostra seu conteúdo na tela. É importante ressaltar que esta função percorre todo o arquivo.txt o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a função *Ler_Reserva_Memoria* passando uma struct Reserva por vez.

3.111 Recebe_Dados_Reserva

A função recebe uma struct Reserva por referência e prenche com valores digitados pelo usuário.

3.112 Arquivo_Url_Fluxo

Função responsável por acessar o arquivo fluxo, onde todos os dados da reserva são salvos para um controle interno da aplicação, a função monta a URL em que será salvo os dados da reserva dentro desse arquivo Fluxo, que é separado pela pasta referente ao ano da reserva e em arquivos que contém o número referente ao més da reserva.

3.113 Main_Reserva

A função verifica se os arquivos referente ao backup do Reserva existe caso não exista ela os cria. Sempre que a função é chamada ela ordena o conteudo do Reserva.txt por meio da função *OrdenaValoresTxt* e pede para que o usuário escolha uma ação:(*Ler, Criar, Editar, Apagar*) e a chama caso o o funcionário tenha o nivel de permisão adequado para realizar tal função.

3.114 Retorna_Codigos_Reserva

Função responsável por abrir o arquivo Reserva.txt e salvar em um vetor todos os códigos armazenados nesse arquivo.

3.115 Ler_Reserva_Bin

Muito semelhante a *Ler_Reserva_Txt*, a função abre o arquivo pela url que foi passada por parâmetro em modo leitura de arquivos binários e lé o conteudo referente a uma struct e mostra seu conteudo na tela.é importante ressaltar que esta função percorre todo o arquivo.bin o que faz com que todos os dados salvos no arquivo sejam exibidos. e para mostrar na tela chama a função *Ler_Reserva_Memoria* passando uma struct de Reserva por vez.

3.116 Gravar_Reserva_Bin

Muito semelhante a *Gravar_Reserva_Txt*, esta função abre o arquivo pela url que foi passada por parâmetro em modo concatenação binária e salva a struct Reserva no arquivo, é importante ressaltar que a struct foi passada por referência e não por valor.

3.117 Valida_Acomadacao_Reserva

Função responsável por abrir o arquivo de Acomodacoes (Binário ou Texto) e verificar se o código recebido por paramêtro esta salvo dentro desse arquivo, em caso de verdadeiro retorna 1 e em caso de falso retorna 0.

3.118 Valida_Hospede_Reserva

Função responsável por abrir o arquivo de Hospedes (Binário ou Texto) e verificar se o código recebido por paramêtro esta salvo dentro desse arquivo, em caso de verdadeiro retorna 1 e em caso de falso retorna 0.

3.119 Criar_Modificar_Reserva

Função responsável por salvar a struct de Reservas dentro do arquvio .bin ou .txt, vale ressaltar que essa função pode ser utilizada tanto para salvar um valor novo no arquivo, quanto para editar um valor já existente dentro do arquivo.

3.120 Retorna_Campo_Struct_Reserva

Esta função le todas as structs do arquivo binário e retorna qual a posição da struct que possui o codigo passado por parametro,caso retorne **-1** é porque não há Reservas com este numero.

3.121 Apagar_Modificar_Reserva_Bin

Está função abre um arquivo binario que teve sua url passada por parametro, alem disso eśtá função recebe um codigo por parametro e uma flag. A funcionalidade principal destá função é transferir todos os dados de struct para um arquivo temporario exceto a struct com o codigo recebido. caso a flag esteja ligada no momento que o codigo é encontrado é chamada a função

Criar_Modificar_Reserva para repreencher a struct após a transferência de todos os dados o arquivo original *Reserva.bin* é apagado e o Temp é renomeado para *Reserva.bin*

3.122 Retorna_Campo_Struct_Fluxo

Esta função le todas as structs do arquivo binário de FLUXO e retorna qual a posição da struct que possui o codigo passado por parametro.

3.123 Apagar_Fluxo

Função responsável por abrir o arquivo de Fluxo recebido por parâmetro e apagar o dado referente ao código que também é recebido por parâmetro.

3.124 Todas_Acomodacoes_TXT

Função responsável por abrir o arquivo Acomodacoes.txt e retornar um vetor contendo todas as acomodações disponiveis para aquela Pesquisa/Reserva.

3.125 Todas_Acomodacoes_BIN

Função responsável por abrir o arquivo Acomodacoes.bin e retornar um vetor contendo todas as acomodações disponiveis para aquela Pesquisa/Reserva.

3.126 Main_Pesquisa

Função responsável por gerenciar o módulo de pesquisa e mostar para o usuário as possiveis opções que ele possui para que possa realizar a sua pesquisa.

3.127 Mostra_Acomodacoes_TXT

Função responsável por abrir o arquivo de Acomodacoes.txt e mostrar todas as acomodações referente ao código que também é recebido por parâmetro.

3.128 Mostra_Acomodacoes_BIN

Função responsável por abrir o arquivo de Acomodacoes.bin e mostrar todas as acomodações referente ao código que também é recebido por parâmetro.

3.129 Verifica_Fluxo

Função responsável por abrir o arquivo de Fluxo e verificar quais acomodações estão indisponiveis naquela determinada data.

3.130 Tipo_Pesquisa

Função responsável por mostrar quais são os tipos de pesquisa que são possiveis de serem realizadas e perguntar quais filtros ele deseja utilizar.

3.131 Retorna_Acomodacoes_Indisponiveis_Com_Codigo_Categoria

Função responsável por gerar um vetor contendo todas as acomodações que estão indisponiveis com base no código de categoria que foi recebido.

3.132 Retorna_Acomodacoes_Indisponiveis_Com_Quantidade_Pessoas

Função responsável por gerar um vetor contendo todas as acomodações que estão indisponiveis com base na quantidade de pessoas que aquela acomodação possui.

3.133 Retorna_Acomodacoes_Indisponiveis_Com_Facilidades

Função responsável por gerar um vetor contendo todas as acomodações que estão indisponiveis com base nas facilidades que o usuário deseja para aquela reserva.

3.134 Pesquisa

Função responsável por realizar as pesquisas com base nos filtros que o usuário passou. Vale ressaltar que que a função pega todas as acomodações que são indisponíveis para aquela consulta, feito isso é gerado outro vetor que contém o complemento das indisponíveis, gerando assim as acomodações disponivéis.

3.135 Valida_Codigo_Acomodacao

Esta função recebe um codigo por parametro e o pesquisa dentro do arquivo Acomodações *.txt ou .bin*, caso o codigo exista no arquivo é retornado **1** e caso não exista é retornado **0**.

3.136 Recebe_Data

Função responsável por receber uma data e a validar.

3.137 Main_IE

Está função é responsavel por capitar o codigoo de qual ação será realizada e chama-la

3.138 Set_On_Off

Está função é responsavel para prencher e retornar uma struct com o switch dos modulos, que devem ser exportados, caso a pessoa tente ligar a exportação de um arquivo vazio é exibida uma mensagem de erro e setado para desligado (0), pois na hora de importar é apagado o arquivo e ao tentar exportar um arquivo vazio iria apenas a pagar os dados na hora da importação.Ao fim da função

3.139 Exportar

Está função chama a *Set_On_Off* e exporta os modulos que estiverem com 1. Para exportar os arquivos primeiro é passado todos os dados do arquivo de backup para um vetor dinamico (há um vetor dinamico para cada modulo setado como 1).Apos todos os dados estarem no buffer é criado um arquivo.xml com o nome que foi informado pelo usuario e transferido todos os dados respeitando a sintaxi.
Obs:Há um vetor Indice[] onde cada celula corresponde ao numero de celulas do vetor dinamico ja alocado.

3.140 Importacao

Na hora de importar é pesquisado pela tag ¡tabela=????¿ , '????' corresponde ao modulo . apos achar o modulo correto é pesquisado a tag ¡registro¿ que correponde a uma struct.
Na importação é apagado todos os dados pois antes de importar para nao dar erro no codigo o arquivo de backup é aberto em modo w ou wb.

3.141 MainFeedback

Função responsável por controlar todo o módulo de FeedBack, essa função pergunta para o usuário qual dos campos deseja realizar o FeedBack, em que as opções são: *Hospedes, Acomodações, Reservas, Produtos.*

3.142 Tipo_Listagem_Reserva

Função responsável perguntar ao usuário qual filtro ele deseja utilizar para realizar o FeedBack de reservas os filtros possiveis sao: *Código Acomodação, Código Hospedes, Data Reservada.*

3.143 Tipo_Listagem_Produtos

Função responsável perguntar ao usuário qual filtro ele deseja utilizar para realizar o FeedBack de produtos os filtros possiveis sao: *Produtos de Consumo, Produtos em estoque mínimo.*

3.144 Tipo_Listagem_Acomodacao

Função responsável perguntar ao usuário qual filtro ele deseja utilizar para realizar o FeedBack de acomodações os filtros possiveis sao: *Código Acomodação, Categoria, Data Disponível.*

3.145 Tipo_Listagem_Hospede

Função responsável perguntar ao usuário qual filtro ele deseja utilizar para realizar o FeedBack de hospedes os filtros possiveis sao: *Código Hospedes, Sexo.*

3.146 Filtro_Produtos_Codigos_Em_Estoque_Minimo

Função responsável por gerar o FeedBack que o usuário deseja, tela ou CSV, para isso é pedido um Range entre códigos para que possa ser aberto o arquivo de produtos e mostrar ao usuário todos os produtos que estão em estoque mínimo.

3.147 Filtro_Produtos_Codigos

Função responsável por gerar o FeedBack que o usuário deseja, tela ou CSV, para isso é pedido um Range entre códigos para que possa ser aberto o arquivo de produtos e mostrar ao usuário todos os campos referente a esse determinado produto.

3.148 Filtro_Reserva_Codigo_Acomodacao

Função responsável por gerar o FeedBack que o usuário deseja, tela ou CSV, para isso é pedido o código da acomodação para que possa ser aberto o arquivo de acomodações e mostrar ao usuário todos os campos referente a essa determinada acomodação.

3.149 Filtro_Reserva_Codigo_Hospede

Função responsável por gerar o FeedBack que o usuário deseja, tela ou CSV, para isso é pedido o código do hospede para que possa ser aberto o arquivo de hospedes e mostrar ao usuário todos os campos referente a esse determinado hospede.

3.150 Filtro_Reserva_Data

Função responsável por gerar o FeedBack que o usuário deseja, tela ou CSV, para isso é pedido uma data referente a entrada e uma data referente a saída para que possa ser aberto o arquivo de reservas e mostrar ao usuário todos os campos referente a essa determinada reserva.

3.151 Filtro_Acomodacao_Codigos

Função responsável por gerar o FeedBack que o usuário deseja, tela ou CSV, para isso é pedido um Range entre códigos para que possa ser aberto o arquivo de acomodações e mostrar ao usuário todos os campos referente a essa determinada acomodação.

3.152 Gera_CSV_Acomodacoes_TXT

Função responsável por gerar o CSV de acomodações que estão salvas em formato de texto, para isso é verificado se o código de acomodação que foi recebido por parametro está salvo no arquivo, se estiver a acomodação é salva em um arquivo com o formato .csv.

3.153 Gera_CSV_Acomodacoes_BIN

Função responsável por gerar o CSV de acomodações que estão salvas em formato de binário, para isso é verificado se o código de acomodação que foi recebido por parametro está salvo no arquivo, se estiver a acomodação é salva em um arquivo com o formato .csv.

3.154 Filtro_Acomodacao_Data_Disponivel

Função responsável por gerar o FeedBack que o usuário deseja, tela ou CSV, para isso é pedido uma data referente a entrada e uma data referente a saída para que possa ser aberto o arquivo de reservas e mostrar ao usuário todas as acomodações que estão disponiveis nessa data.

3.155 Filtro_Acomodacao_CodCategoria

Função responsável por gerar o FeedBack que o usuário deseja, tela ou CSV, para isso é pedido o código da categoria para que possa ser aberto o arquivo de Acomodaçõeses e mostrar ao usuário todas as acomodações que possui aquele código de categoria.

3.156 Filtro_Hospede_Codigos

Função responsável por gerar o FeedBack que o usuário deseja, tela ou CSV, para isso é pedido o código do hospede para que possa ser aberto o arquivo de Hospedes e mostrar ao usuário todos os hospedes que possuem aquele código.

3.157 Filtro_Hospede_Sexo

Função responsável por gerar o FeedBack que o usuário deseja, tela ou CSV, para isso é pedido o sexo do hospede para que possa ser aberto o arquivo de Hospedes e mostrar ao usuário todos os hospedes que possuem aquele sexo.