**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA**

**UNIDADE EDUCACIONAL CORAÇÃO EUCARISTICO**

**Bacharelado em Engenharia de Software**

João Vitor Santana Lopes(Aluno do Laerte) e Pedro Maia da Costa Silva(Aluno do Cristiano)

Sistema-Hotel-Descanso-Garantido

**Apresentação:**

Este sistema foi feito para o Hotel Descanso Garantido com o intuito de ajudar os seus funcionários a Gerenciar o hotel tanto na área de cadastro de clientes e registrar e dar baixa em estadias além de registrar os funcionários que trabalhão a esse hotel. Deixando todas essas informações guardadas dentro de arquivos txt.

**Backlog do produto:**

**Backlog Geral:** Sprint 1, Sprint 2, Sprint 3 e Sprint 4.

|  |  |
| --- | --- |
| Divisão de Tarefas (Nome de quem vai fazer) | Tarefas para fazer |
| Pedro | Menu (int main) |
| João | Função Cadastra Cliente |
| João | Função Cadastra Funcionário |
| Pedro | Função Cadastra Quarto |
| Pedro | Função Cadastra Estadia |
| João | Função Pesquisa Pessoas |
| Pedro | Função Pesquisa Estadia |
| Pedro e João | Função Finaliza Estadia |
| Pedro | Backlog |
| João | Lista das funções |
| Pedro e João | Caso de teste |
| Pedro e João | Integração do codigo |
| Pedro e João | Video |
| Pedro e João | Testes |

**Backlog Sprint 1:**

|  |  |  |
| --- | --- | --- |
| Fazer | Fazendo | Feito |
| Selecionando casos de teste | Função Cadastra Quarto | Menu (int main) |
| Testes unitários |  | Função Cadastra Cliente |
|  |  |  |

**Backlog Sprint 2:**

|  |  |  |
| --- | --- | --- |
| Fazer | Fazendo | Feito |
| Selecionando casos de teste | Função Cadastra Estadia | Função Finaliza Estadia |
| Testes unitários |  |  |
| Assinatura das funções |  |  |

**Backlog Sprint 3:**

|  |  |  |
| --- | --- | --- |
| Fazer | Fazendo | Feito |
| Selecionando casos de teste | Teste de integração | Menu (int main) |
| Adicionando os casos de teste | Assinatura das funções | Função Cadastra Cliente |
| Documentação | Teste unitários | Função Cadastra Funcionário |

**Backlog Sprint 4:**

|  |  |  |
| --- | --- | --- |
| Fazer | Fazendo | Feito |
|  | Video | Documentação |
|  | Testes | Teste de sistema |
|  | Lista das funções | Adicionando os casos de teste |
|  |  | Registro do backlog |

**Lista de funções e Parâmetros:**

**1. void cadastraCliente ()**

Cadastra um cliente com as seguintes informações, codigo,nome,endereco e número, as informações são guardadas em um arquivo txt chamado "Clientes.txt" no seguinte formato: codigo;nome;endereco;número;, no arquivo cada linha representa um cliente diferente, e as informações são adquiridas da seguinte forma, é perguntado ao usuário seu nome, endereço e número, e são validados os dados da seguinte forma, é conferido se no "nome" existem apenas letras e se no "numero" existem apenas números, e o codigo do cliente é atribuído da seguinte forma, o codigo lê a última linha do arquivo e encontra o codigo do último cliente, e atribui ao próximo cliente o codigo lido + 1.

**2. void cadastraFuncionario ()**

Cadastra um funcionário com as seguintes informações, codigo, nome, cargo, número e salario;, as informações são guardadas em um arquivo txt chamado "Funcionarios.txt" no seguinte formato: codigo;nome;cargo;número;salario;, no arquivo cada linha representa um cliente diferente, e as informações são adquiridas da seguinte forma, é perguntado ao usuário seu nome, cargo, número e salario, e são validados os dados da seguinte forma, é conferido se no "nome" e "cargo" existem apenas letras e se no "numero" e "salario" existem apenas números, e o codigo do cliente é atribuído da seguinte forma, o codigo lê a última linha do arquivo e encontra o codigo do último cliente, e atribui ao próximo cliente o codigo lido + 1.

**3. void cadastraEstadia ()**

Cadastra um Estadia no hotel, conferindo as seguintes coisas: pergunta ao usuário seu codigo de cliente e confere no arquivo "clientes.txt" se tem alguém com esse codigo, depois pergunta a data de saída e de entrada do cliente e confere se as datas colocadas são validas, após isso pergunta a quantidade de hospedes, e confere no arquivo "quartos.txt" se existe algum quarto que comporta a quantidade de hospedes digitada, se existir, o quarto no arquivo é atualizado para "ocupado" e é cadastrado no arquivo "estadia.txt" a estadia do cliente no seguinte formato 'codigoEstadia;codigoCliente;entrada;saida;diarias;numeroQuarto;'.

**4. void cadastraQuarto ()**

Cadastra um novo quarto no hotel, perguntando ao usuário, o número do quarto, quantos hospedes cabem, a diaria do quarto e se ele está ocupado ou não, após isso, confere se todas as entradas digitadas pelo usuário foram validas, se sim, cadastra as informações no arquivo "quartos.txt" no seguinte formato numeroQuarto;hospedes;valorDiario;status;

**5. void finalizarEstadia ()**

Finaliza uma estadia, perguntado ao usuário o codigo da estadia e procurando no arquivo "estadia.txt" a estadia digitada e guarda em um outro arquivo txt chamado "temp." todas as estadias cadastradas menos a estadia que foi digitada pelo usuário, e no arquivo txt "quartos.txt" atualiza o status do quarto equivalente a estadia, atualizando ele para "desocupado", e por fim, todos as estadias cadastradas no temp são escritas novamente no estadia, fazendo assim, a estadia que foi finalizada ser apagada do arquivo estadia, já que temp não possuía ela.

**6. void pesquisaPessoas ()**

Pesquisa as informações de um cliente ou de um funcionário, da seguinte forma: a função pede para o usuário escolher se vai ser pesquisado um cliente ou um funcionário, e de que forma, usando o 'codigo' ou o 'nome', dependendo do que for escolhido, o codigo ira pesquisar no arquivo "Cliente.txt" ou "Funcionarios.txt", o codigo lê linha por linha do arquivo, (já que cada linha representa um cliente/funcionário), procurando um codigo/nome que corresponda com o que foi digitado pelo usuário, (usando a função strtok() para dividir as informações nos ';') caso alguma corresponda, é mostrada as informações do cliente (codigo,nome,endereco e número), ou do funcionário (codigo, nome, cargo, número e salario) na tela e a função acaba, caso as informações digitadas pelo usuário não corresponda a nenhuma cadastrada, o codigo mostra uma mensagem dizendo que não encontrou ninguém com a informação digitada e a função acaba.

**7. void achaEstadia ()**

A função pesquisa uma estadia, perguntado ao cliente o seu codigo para procurar no arquivo "Estadia.txt" todas as estadias relacionadas a ele, se encontrar alguma, mostra na tela todas as informações de todas as estadias, que são código de estadia, datas de entrada e saída, valor a pagar e número do quarto.

**Teste**

**Caso de teste:**

**Pré-requisitos:**

- O sistema deve estar corretamente configurado e em execução.

- Os arquivos "Clientes.txt", "Funcionarios.txt", "Estadia.txt" e "Quartos.txt" devem estar disponíveis para leitura e escrita.

**Passos de Teste e Resultados Esperados**

1.Inicializar o Sistema

-Passo: Executar o programa.

-Resultado Esperado: O sistema deve exibir o menu principal com as opções disponíveis.

2.Cadastrar um Cliente:

|  |  |  |
| --- | --- | --- |
| **Passo** | **Entrada** | **Resultado Esperado** |
| Selecionar a opção 1 - `Cadastrar cliente`. | Nome: "João Silva"  Endereço: "Rua A, 123"  Número: "1234567890" | O sistema deve validar os dados, gerar um código único para o cliente e salvar as informações no arquivo "Clientes.txt". |

3.Cadastrar um Funcionário:

|  |  |  |
| --- | --- | --- |
| **Passo** | **Entrada** | **Resultado Esperado** |
| Selecionar a opção 2 - `Cadastrar funcionário`. | Nome: "Maria Oliveira"  Cargo: "Recepcionista"  Número: "0987654321"  Salário: "2000,00" | O sistema deve validar os dados, gerar um código único para o funcionário e salvar as informações no arquivo "Funcionarios.txt". |

4.Cadastrar um Estadia:

|  |  |  |
| --- | --- | --- |
| **Passo** | **Entrada** | **Resultado Esperado** |
| Selecionar a opção 3 - `Cadastrar Estadia`. | Código do Cliente: "1"  Data de entrada: "01/06/2024"  Data de saída: "05/06/2024"  Número de hóspedes: "2" | O sistema deve validar os dados, calcular o custo total da estadia, atualizar o status do quarto para "ocu" e salvar as informações no arquivo "Estadia.txt". |

5.Cadastrar um Quarto:

|  |  |  |
| --- | --- | --- |
| **Passo** | **Entrada** | **Resultado Esperado** |
| Selecionar a opção 4 - `Cadastrar Quarto`. | Número do quarto: "101"  Hospedes: "2"  Valor diário: "150,00"  Status: "des" ou “ocu” | O sistema deve validar os dados e salvar as informações no arquivo "Quartos.txt". |

6.Pequisa por Cliente:

|  |  |  |
| --- | --- | --- |
| **Passo** | **Entrada** | **Resultado Esperado** |
| Selecionar a opção 5 - `Pesquisar por pessoas`. | Tipo de pessoa: "1" (Cliente)  Tipo de pesquisa: "1" (Código)  Código: "1" | O sistema deve exibir as informações do cliente "João Silva". |

7.Pesquisa por Funcionário:

|  |  |  |
| --- | --- | --- |
| **Passo** | **Entrada** | **Resultado Esperado** |
| Selecionar a opção 5 - `Pesquisar por pessoas`. | Tipo de pessoa: "2" (Funcionário)  Tipo de pesquisa: "2" (Nome)  Nome: "Maria Oliveira" | O sistema deve exibir as informações do funcionário "Maria Oliveira". |

8.Finaliza Estadia:

|  |  |  |
| --- | --- | --- |
| **Passo** | **Entrada** | **Resultado Esperado** |
| Selecionar a opção 6 - `Baixa em Estadia`. | Tipo de pessoa: "2" (Funcionário)  Tipo de pesquisa: "2" (Nome)  Nome: "Maria Oliveira" | O sistema deve exibir as informações do funcionário "Maria Oliveira". |

9.Pesquisar Estadia:

|  |  |  |
| --- | --- | --- |
| **Passo** | **Entrada** | **Resultado Esperado** |
| Selecionar a opção 7 - `Pesquisar Estadia`. | Código do cliente: "1" | O sistema deve exibir as informações das estadias sobre o codigo do cliente se existir. |

10.Sair do Sistema:

|  |  |  |
| --- | --- | --- |
| **Passo** | **Entrada** | **Resultado Esperado** |
| Selecionar a opção 0 - `Sair`. |  | O sistema deve exibir a mensagem "Saindo do Programa..." e encerrar a execução. |

**Codigo do sistema:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#include <locale.h>

struct Cliente

{

int codigo;

char nome[100];

char endereco[100];

char numero[11];

};

struct Funcionarios

{

int codigo;

char nome[100];

char numero[11];

char cargo[50];

char salario[50];

};

struct Estadia

{

int codigoEstadia;

int codigoCliente;

int entrada[3];

int saida[3];

float diarias;

int numeroQuarto;

};

struct Quarto

{

int numeroQuarto;

int hospedes;

float valorDiario;

char status[3];

};

void cadastraCliente ();

void cadastraEstadia ();

void cadastraFuncionario ();

void cadastroQuarto ();

void achaEstadia ();

void pesquisaPessoas ();

void finalizarEstadia ();

int main()

{

int menu;

setlocale(LC\_ALL, "Portuguese");

do

{

printf("Escolha oque voce ira fazer no sistema:\n");

printf(" 1 - Cadastras cliente \n 2 - Cadastrar funcionario \n 3 - Cadastrar Estadia\n 4 - Cadastrar Quarto \n 5 - Pesquisar por pessoas \n 6 - Baixa em Estadia \n 7 - Pesquisar Estadia \n 0 - Sair \n");

scanf("%d", &menu);

while (getchar() != '\n');

if(menu < 0 || menu > 7)

{

printf("Digite entre as opçoes do menu! \n");

}

switch(menu)

{

case 0:

printf("Saindo do Programa...");

break;

case 1:

cadastraCliente();

break;

case 2:

cadastraFuncionario();

break;

case 3:

cadastraEstadia();

break;

case 4:

cadastroQuarto();

break;

case 5:

pesquisaPessoas();

break;

case 6:

finalizarEstadia();

break;

case 7:

achaEstadia();

break;

}

}

while(menu != 0);

return 0;

}

void cadastraCliente()

{

//inicializa as variaveis

FILE \*fluxo;

fluxo = fopen("Clientes.txt", "a+");

struct Cliente clienteatual;

int erro = 0;

//preenche as variaveis

do

{

erro = 0;

printf("digite o nome do cliente:\n");

gets(clienteatual.nome);

//confere se o usuario digitou apenas letras em seu nome

for (int i = 0; i < strlen(clienteatual.nome); i++)

{

if(isalpha(clienteatual.nome[i]) == 0 && clienteatual.nome[i] != ' ')

{

erro++;

}

}

printf("digite o endereco do cliente:\n");

gets(clienteatual.endereco);

printf("digite o numero do cliente:\n");

gets(clienteatual.numero);

//confere se o usuario digitou apenas numeros em seu numero

for (int i = 0; i < strlen(clienteatual.numero); i++)

{

if(isdigit(clienteatual.numero[i]) == 0)

{

erro++;

}

}

}

while(erro != 0);

//inicializa as variaveis que irão manipular os dados do arquivo

char linha[200];

char ultimalinha[200];

char delimitador[] = ";";

//encontra a ultima linha do arquivo

while(fgets(linha, sizeof(linha), fluxo) != NULL)

{

strcpy(ultimalinha, linha);

}

//le o codigo do ultimo cliente cadastrado e da o codigo + 1 para o proximo cliente cadastrado

clienteatual.codigo = atoi(strtok(ultimalinha, delimitador));

printf("seu codigo para futuras pesquisas e %d \n", clienteatual.codigo + 1);

//coloca as infomações no arquivo txt, neste formato: codigo;nome;endereco;numero;

fflush(fluxo);

fprintf(fluxo, "\n%d;%s;%s;%s;", clienteatual.codigo + 1,clienteatual.nome,clienteatual.endereco,clienteatual.numero);

fclose(fluxo);

}

void cadastraFuncionario()

{

//inicializa as variaveis

FILE \*fluxo;

fluxo = fopen("Funcionarios.txt", "a+");

struct Funcionarios funcionarioatual;

int erro = 0;

//preenche as variaveis

do

{

erro = 0;

printf("digite o nome do funcionario:\n");

gets(funcionarioatual.nome);

//confere se o usuario digitou apenas letras em seu nome

for (int i = 0; i < strlen(funcionarioatual.nome); i++)

{

if(isalpha(funcionarioatual.nome[i]) == 0 && funcionarioatual.nome[i] != ' ')

{

erro++;

}

}

printf("digite o cargo do funcionario:\n");

gets(funcionarioatual.cargo);

//confere se o usuario digitou apenas letras em seu cargo

for (int i = 0; i < strlen(funcionarioatual.cargo); i++)

{

if(isalpha(funcionarioatual.cargo[i]) == 0 && funcionarioatual.cargo[i] != ' ')

{

erro++;

}

}

printf("digite o numero do funcionario:\n");

gets(funcionarioatual.numero);

//confere se o usuario digitou apenas numeros em seu numero

for (int i = 0; i < strlen(funcionarioatual.numero); i++)

{

if(isdigit(funcionarioatual.numero[i]) == 0)

{

erro++;

}

}

printf("digite o salario do funcionario:\n");

gets(funcionarioatual.salario);

//confere se o usuario digitou apenas numeros ou virgulas em seu salario

for (int i = 0; i < strlen(funcionarioatual.salario); i++)

{

if(isdigit(funcionarioatual.salario[i]) == 0 && funcionarioatual.salario[i] != ',')

{

erro++;

}

}

}

while(erro != 0);

//inicializa as variaveis que irão manipular os dados do arquivo

char linha[200];

char ultimalinha[200];

char delimitador[] = ";";

//encontra a ultima linha do arquivo

while(fgets(linha, sizeof(linha), fluxo) != NULL)

{

strcpy(ultimalinha, linha);

}

//le o codigo do ultimo cliente cadastrado e da o codigo + 1 para o proximo cliente cadastrado

funcionarioatual.codigo = atoi(strtok(ultimalinha, delimitador));

printf("seu codigo para futuras pesquisas e %d \n", funcionarioatual.codigo + 1);

//coloca as infomações no arquivo txt, neste formato: codigo;nome;cargo;numero;salario;

fflush(fluxo);

fprintf(fluxo, "\n%d;%s;%s;%s;%s;", funcionarioatual.codigo + 1,funcionarioatual.nome,funcionarioatual.cargo,funcionarioatual.numero,funcionarioatual.salario);

fclose(fluxo);

}

void cadastraEstadia ()

{

char estadia2;

FILE \*estadia, \*quarto, \*cliente;

estadia = fopen("Estadia.txt", "a+");

quarto = fopen("Quartos.txt", "r+");

cliente = fopen("Clientes.txt", "r");

int hospedes, certo = 0;

int comphospedes;

long pos;

struct Estadia registro;

char linha[200];

char ultimalinha[200];

char delimitador[] = ";";

if(!estadia)

{

printf("Ocorreu um error ao abrir um arquivo");

fclose(estadia);

}

if(!quarto)

{

printf("Ocorreu um error ao abrir um arquivo");

fclose(quarto);

}

do

{

while(fgets(linha, sizeof(linha), estadia) != NULL)

{

strcpy(ultimalinha, linha);

}

registro.codigoEstadia = atoi(strtok(ultimalinha, delimitador));

do

{

printf("Digite o codigo do Cliente que deseja se hospedar:\n");

scanf("%d", &registro.codigoCliente);

}while(registro.codigoCliente < 1);

while(fgets(linha, sizeof(linha), cliente) != NULL)

{

char\*campo = strtok(linha, delimitador);

int compcodigo = atoi(campo);

if(compcodigo == registro.codigoCliente)

{

certo++;

}

}

if(certo == 0)

{

printf("\nNenhum cliente tem esse codigo");

break;

}

do

{

printf("Digite o dia de entrada do cliente:\n");

scanf("%d", &registro.entrada[0]);

}

while(registro.entrada[0] < 1 || registro.entrada[0] > 31);

do

{

printf("Digite o mes de entrada do cliente:\n");

scanf("%d", &registro.entrada[1]);

}

while(registro.entrada[1] < 1 && registro.entrada[1] > 12);

do

{

printf("Digite o ano de entrada do cliente:\n");

scanf("%d", &registro.entrada[2]);

}

while(registro.entrada[2] < 2024);

do

{

printf("Digite o dia de saida do cliente:\n");

scanf("%d", &registro.saida[0]);

}

while(registro.saida[0] < 1 || registro.saida[0] > 31);

do

{

printf("Digite o mws de saida do cliente:\n");

scanf("%d", &registro.saida[1]);

}

while(registro.saida[1] < 1 || registro.saida[1] > 12);

do

{

printf("Digite o ano de saida do cliente:\n");

scanf("%d", &registro.saida[2]);

}

while(registro.saida[2] < registro.entrada[2]);

do

{

printf("Digite quantos hospedes iram ser hospedados:");

scanf("%d", &hospedes);

} while (hospedes < 1);

certo = 0;

int certo2 = 0;

while (getchar() != '\n');

while(fgets(linha, sizeof(linha), quarto) != NULL)

{

pos = ftell(quarto) - strlen(linha);

char \*campo = strtok(linha, delimitador);

registro.numeroQuarto = atoi(campo);

campo = strtok(NULL, delimitador);

comphospedes = atoi(campo);

if(hospedes == comphospedes)

{

campo = strtok(NULL, delimitador);

registro.diarias = atoi(campo);

campo = strtok(NULL, delimitador);

certo++;

if(strcmp(campo, "des") == 0)

{

fseek(quarto, pos, SEEK\_SET);

fflush(quarto);

fprintf(quarto, "%d;%d;%.2f;ocu;", registro.numeroQuarto, hospedes, registro.diarias);

certo2++;

break;

}

}

}

if(certo == 0)

{

printf("Nao existe um quarto com esse numero de hospedes.\n");

break;

}

if(certo2 == 0)

{

printf("Nao existe um quarto disponivel com esse numero de hospedes.\n");

break;

}

int dias = 0;

dias = registro.saida[0] - registro.entrada[0];

dias = dias + (30 \* (registro.saida[1] - registro.entrada[1]));

dias = dias + (365 \* (registro.saida[2] - registro.entrada[2]));

registro.diarias = registro.diarias \* dias;

fflush(estadia);

fprintf(estadia, "\n%d;%d;", registro.codigoEstadia + 1, registro.codigoCliente);

fflush(estadia);

fprintf(estadia, "%d;%d;%d;",registro.entrada[0], registro.entrada[1], registro.entrada[2]);

fflush(estadia);

fprintf(estadia, "%d;%d;%d;", registro.saida[0], registro.saida[1], registro.saida[2]);

fprintf(estadia, "%.2f;%d;", registro.diarias, registro.numeroQuarto);

fflush(estadia);

printf("Deseja cadastrar mais alguma estadia('s' para sim e 'n' para não):\n");

scanf(" %c", &estadia2);

}

while(estadia2 != 'n');

fclose(estadia);

fclose(quarto);

}

void cadastroQuarto ()

{

char quarto2;

FILE \*quarto;

quarto = fopen("Quartos.txt", "a+");

struct Quarto registro;

if(!quarto)

{

printf("Ocorreu um error ao abrir um arquivo");

exit(1);

}

do

{

do

{

printf("Qual o numero do quarto que voce deseja registrar:\n");

scanf("%d", &registro.numeroQuarto);

}

while(registro.numeroQuarto < 100);

do

{

printf("Quantos hospestes cabem dentro deste quarto:\n");

scanf("%d", &registro.hospedes);

}

while(registro.hospedes < 0);

do

{

printf("Quanto e a diaria deste quarto:\n");

scanf("%f", &registro.valorDiario);

}

while(registro.valorDiario < 0.01);

do

{

while (getchar() != '\n');

printf("Qual o status do quarto ele esta (des = desocupado ou ocu = ocupado):\n");

gets(registro.status);

}

while((strcmp(registro.status, "des") != 0 && strcmp(registro.status, "ocu") != 0));

fprintf(quarto, "\n%d;%d;%.2f;%s;", registro.numeroQuarto, registro.hospedes, registro.valorDiario, registro.status);

printf("Voce deseja cadastrar mais um quarto('s' para sim e 'n' para nao):\n");

scanf(" %c", &quarto2);

}

while(quarto2 != 'n');

fclose (quarto);

}

void finalizarEstadia ()

{

int codigo, numQuarto, comparar;

int compararQuarto, hospedes;

float custos, diaria;

char linha[200];

char delimitador[] = ";";

long pos;

FILE \*estadia, \*quarto, \*temp;

temp = fopen("Temp.txt", "w");

estadia = fopen("Estadia.txt", "r");

quarto = fopen("Quartos.txt", "r+");

if(!estadia)

{

printf("Ocorreu um error ao abrir um arquivo");

fclose(estadia);

return;

}

if(!quarto)

{

printf("Ocorreu um error ao abrir um arquivo");

fclose(quarto);

return;

}

if(!temp)

{

printf("Ocorreu um error ao abrir um arquivo");

fclose(temp);

return;

}

do{

printf("Digite o codigo da estadia que o cliente deseja dar baixa:\n");

scanf("%d", &codigo);

}while(codigo < 1);

while(fgets(linha, sizeof(linha), estadia) != NULL)//Preencher o txt temp

{

char \*campo = strtok(linha, delimitador);

comparar = atoi(campo);

if(comparar == codigo)

{

campo = strtok(NULL, delimitador);

campo = strtok(NULL, delimitador);

campo = strtok(NULL, delimitador);

campo = strtok(NULL, delimitador);

campo = strtok(NULL, delimitador);

campo = strtok(NULL, delimitador);

campo = strtok(NULL, delimitador);

campo = strtok(NULL, delimitador);

custos = atof(campo);

campo = strtok(NULL, delimitador);

numQuarto = atoi(campo);

}

if(comparar != codigo)

{

campo = strtok(NULL, delimitador);

int cliente = atoi(campo);

campo = strtok(NULL, delimitador);

int diaen = atoi(campo);

campo = strtok(NULL, delimitador);

int mesen = atoi(campo);

campo = strtok(NULL, delimitador);

int anoen = atoi(campo);

campo = strtok(NULL, delimitador);

int diasa = atoi(campo);

campo = strtok(NULL, delimitador);

int mesai = atoi(campo);

campo = strtok(NULL, delimitador);

int anosa = atoi(campo);

campo = strtok(NULL, delimitador);

float preco = atof(campo);

campo = strtok(NULL, delimitador);

numQuarto = atoi(campo);

fprintf(temp, "%d;%d;", comparar, cliente);

fflush(temp);

fprintf(temp, "%d;%d;%d;",diaen, mesen, anoen);

fflush(temp);

fprintf(temp, "%d;%d;%d;", diasa, mesai, anosa);

fprintf(temp, "%.2f;%d;\n", preco, numQuarto);

fflush(temp);

}

}

fclose(temp);

fclose(estadia);

while(fgets(linha, sizeof(linha), quarto) != NULL)//Mudar de ocupado para desocupado.

{

pos = ftell(quarto) - strlen(linha);

char \*campo = strtok(linha, delimitador);

compararQuarto = atoi(campo);

if(numQuarto == compararQuarto)

{

campo = strtok(NULL, delimitador);

hospedes = atoi(campo);

campo = strtok(NULL, delimitador);

diaria = atof(campo);

fseek(quarto, pos, SEEK\_SET);

fflush(quarto);

fprintf(quarto, "%d;%d;%.2f;des;", numQuarto, hospedes, diaria);

break;

}

}

fclose(quarto);

temp = fopen("Temp.txt", "r");

estadia = fopen("Estadia.txt", "w");

if (!estadia || !temp)

{

printf("Ocorreu um erro ao reabrir os arquivos");

if (estadia) fclose(estadia);

if (temp) fclose(temp);

return;

}

while (fgets(linha, sizeof(linha), temp) != NULL)//reescrever o arquivo estadia com os dados de temp

{

fprintf(estadia,"%s", linha);

}

fclose(estadia);

fclose(temp);

printf("Pronto! A estadia acabou de ser finalizada o cliente agora deve pagar: %.2f\n", custos);

}

void pesquisaPessoas()

{

//inicializa as variaveis

FILE \*fluxo;

int tipopessoa;

int tipopesquisa;

char linha[250];

char codigo[11];

char codigo2[11];

char nome[100];

char delimitador[] = ";";

int certo = 0;

//pergunta ao usuario em que arquivo o codigo ira pesquisar e como

do

{

printf("Voce deseja saber as informacoes de um Cliente ou de um Funcionario? Digite '1' para cliente e '2' para funcionario: ");

scanf("%d", &tipopessoa);

while (getchar() != '\n');

}

while(tipopessoa != 1 && tipopessoa != 2);

do

{

printf("Você deseja pesquisar usando o seu codigo ou seu nome? (se lembre que o nome deve ser exatamente igual ao cadastrado) Digite '1' para codigo e '2' para nome: ");

scanf("%d", &tipopesquisa);

while (getchar() != '\n');

}

while(tipopesquisa != 1 && tipopesquisa != 2);

//abre o arquivo de acordo com o que o usuario escolher

if(tipopessoa == 1)

{

fluxo = fopen("Clientes.txt", "r");

}

else

{

fluxo = fopen("Funcionarios.txt", "r");

}

if(tipopesquisa == 1)

{

printf("Digite o seu codigo:\n");

gets(codigo);

}

else

{

printf("Digite o seu nome:\n");

gets(nome);

}

//while que confere todas as linhas

while(fgets(linha, sizeof(linha), fluxo) != NULL)

{

char \*campo = strtok(linha, delimitador);

// Pesquisa cliente

if(tipopessoa == 1)

{

// Pesquisa por código

if(tipopesquisa == 1)

{

if(strcmp(campo, codigo) == 0)

{

campo = strtok(NULL, delimitador);

printf("O seu nome é %s\n", campo);

campo = strtok(NULL, delimitador);

printf("O seu endereço é %s\n", campo);

campo = strtok(NULL, delimitador);

printf("O seu número é %s\n", campo);

certo++;

break;

}

}

// Pesquisa por nome

if(tipopesquisa == 2)

{

strcpy(codigo2, campo);

campo = strtok(NULL, delimitador);

if(strcmp(campo, nome) == 0)

{

printf("o seu codigo e %s\n", codigo2);

campo = strtok(NULL, delimitador);

printf("O seu endereco e %s\n", campo);

campo = strtok(NULL, delimitador);

printf("O seu numero e %s\n", campo);

certo++;

break;

}

}

}

// Pesquisa funcionário

if(tipopessoa == 2)

{

// Pesquisa por código

if(tipopesquisa == 1)

{

if(strcmp(campo, codigo) == 0)

{

campo = strtok(NULL, delimitador);

printf("O seu nome e %s\n", campo);

campo = strtok(NULL, delimitador);

printf("O seu cargo e %s\n", campo);

campo = strtok(NULL, delimitador);

printf("O seu numero e %s\n", campo);

campo = strtok(NULL, delimitador);

printf("O seu salario e %s\n", campo);

certo++;

break;

}

}

// Pesquisa por nome

if(tipopesquisa == 2)

{

strcpy(codigo2, campo);

campo = strtok(NULL, delimitador);

if(strcmp(campo, nome) == 0)

{

printf("o seu codigo e %s\n", codigo2);

campo = strtok(NULL, delimitador);

printf("O seu cargo e %s\n", campo);

campo = strtok(NULL, delimitador);

printf("O seu numero e %s\n", campo);

campo = strtok(NULL, delimitador);

printf("O seu salario e %s\n", campo);

certo++;

break;

}

}

}

}

if(certo == 0)

{

printf("Não foi possivel encontrar nenhum Cliente/Funcionario com o Codigo/Nome digitado.\n");

}

fflush(fluxo);

fclose(fluxo);

}

void achaEstadia ()

{

FILE \*estadia;

int cliente;

int compCliente;

char delimitador[] = ";";

char linha[200];

int codigo, numeroQuarto;

int diaEntrada, mesEntrada, anoEntrada;

int diaSaida, mesSaida, anoSaida;

float valor;

estadia = fopen("Estadia.txt", "r");

if (estadia == NULL) {

printf("Erro ao abrir o arquivo.\n");

}

do{

printf("digite o codigo do cliente que deseja realizar as buscas das estadias:\n");

scanf("%d", &cliente);

}while(cliente < 1);

while(fgets(linha, sizeof(linha), estadia) != NULL)

{

char \*campo = strtok(linha, delimitador);

codigo = atoi(campo);

campo = strtok(NULL, delimitador);

compCliente = atoi(campo);

if(cliente == compCliente)

{

printf("Estas sao as informacoes da Estadia:\n");

printf("Codigo Estadia: %d\n", codigo);

campo = strtok(NULL, delimitador);

diaEntrada = atoi(campo);

campo = strtok(NULL, delimitador);

mesEntrada = atoi(campo);

campo = strtok(NULL, delimitador);

anoEntrada = atoi(campo);

printf("Este foi o dia de entrada: %d/%d/%d\n", diaEntrada, mesEntrada, anoEntrada);

campo = strtok(NULL, delimitador);

diaSaida = atoi(campo);

campo = strtok(NULL, delimitador);

mesSaida = atoi(campo);

campo = strtok(NULL, delimitador);

anoSaida = atoi(campo);

printf("Este foi o dia de saida: %d/%d/%d\n", diaSaida, mesSaida, anoSaida);

campo = strtok(NULL, delimitador);

valor = atoi(campo);

printf("Este é o valor a ser pago: %.2f\n", valor);

campo = strtok(NULL, delimitador);

numeroQuarto = atoi(campo);

printf("Este é o numero do quarto: %d\n\n", numeroQuarto);

}

}

fflush(estadia);

fclose(estadia);

}