PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA
UNIDADE EDUCACIONAL CORAÇÃO EUCARÍSTICO
Bacharelado em Engenharia de Software
Maria Clara Gomes Silva de Oliveira
Hotel Descanso Garantido

Apresentação:
O objetivo principal do sistema de gerenciamento de hotel é otimizar e simplificar as operações diárias de um hotel, permitindo o cadastro e a gestão eficiente de clientes, funcionários e estadias, além de monitorar a disponibilidade e o status dos quartos. Através deste sistema, a administração pode registrar dados importantes, verificar a ocupação de quarto em tempo real e garantir um atendimento ágil e preciso aos hóspedes, resultado em uma operação mais organizada e produtiva.

Backlog to produto:
Exemplo:
Apresente o backlog do seu produto a cada semana

ID da tarefa	Descrição da tarefa	Status inicial	Sprint alocado	Pessoa responsável
T1	Implantação do cadastro de cliente	Pronto		Maria Clara
T2	Implantação do cadastro de funcionário	Pronto		Maria Clara
T3	Implementar registro de estadia	Pronto		Maria Clara
T4	Implementar persistência de dados	Pronto		Maria Clara
T5	Implementar verificação de disponibilidade de quartos	Pronto		Maria Clara
T6	Implementar navegação no menu principal	Pronto		Maria Clara

Lista de assinaturas das funções e parâmetros
Explicação da estrutura de dados principal do programa.
Apresentação das assinaturas das funções

As funções e parâmetros utilizados no programa foram:

1. **void gerarCodigo(char *codigo, int tamanho)**

Função gerarCodigo() para efetuar a impressão dos códigos dos clientes, funcionários, e do cadastro de reserva, recebe como parâmetro o ponteiro para o endereço do código, o tamanho do código informado na struct do cliente, funcionário, e cadastro de reserva.

2. **struct cliente cadastroCliente();**

Função cadastroCliente() para cadastro de cliente, sendo necessário o cadastro de nome, endereço, telefone, e imprime o códigos gerado pela função **void gerarCodigo(char *codigo, int tamanho.**

3. **struct funcionario cadastroFuncionario().**

Função cadastroFuncionario() para cadastro de funcionários, sendo necessário inserir o nome, telefone, cargo, salário e imprime o código do funcionário gerado pela função **void gerarCodigo(char *codigo, int tamanho).**

4. **void exibirCliente(struct cliente clie)**

Função exibirCliente() para exibir dados dos clientes, usando os dados inseridos **struct cliente cadastroCliente().**

5. **void exibirFuncionario(struct funcionario func)**

Função exibirFuncionario() para exibir dados dos funcionários, usando os dados inseridos **struct funcionario cadastroFuncionario().**

6. **void inicializarQuartos(struct quarto quartos[])**

Função inicializarQuartos() para inicializar os quartos com capacidade, taxas é status do quarto predefinidas

7. **void cadastrarEstadia(struct estadia estadias[], int *totalEstadias, struct cliente clientes[], int totalCliente, struct quarto quartos[], int totalQuartos)**

A função cadastrarEstadia é responsável por registrar uma nova estadia no sistema, associando um cliente a um quarto disponível por um período específico. Ela valida as informações fornecidas, verifica a disponibilidade do quarto e atualiza o status do quarto para "OCUPADO" caso a estadia seja registrada com sucesso.

8. **void exibirEstadias(struct estadia estadias[], int totalEstadias, struct quarto quartos[], int totalQuartos);**

Função exibirEstadias exibe todas as estadias registradas no sistema. Ela apresenta informações detalhadas sobre cada estadia, incluindo o código da estadia, o código

do cliente, o número do quarto, as datas de entrada e saída, o número de diárias e a quantidade de hóspedes.

9. void verificarDisponibilidade(struct quarto quartos[], int totalQuartos);

Esta função não possui retorno. Ela apenas exibe as informações de disponibilidade dos quartos no console.

10. void gravarEstadiasClientesFuncionariosEmArquivo (struct estadia estadias[], int totalEstadias, struct cliente clientes[], int totalClientes, struct funcionario funcionarios[], int totalFuncionarios)

A função gravarEstadiasClientesFuncionariosEmArquivo grava os dados das estadias, clientes e funcionários em um arquivo de texto. Esse arquivo pode ser utilizado para persistir os dados, permitindo a leitura posterior ou para fins de relatório.

TESTES

Casos de teste do software:

Entradas	Classes Válidas	Resultado esperado	Classes inválidas	Resultado esperado
Cadastro de cliente, com nome , endereço, telefone.	Cliente cadastrado com sucesso.	Cliente cadastrado com sucesso, com a saída do código do cliente.	Cadastro de cliente sem os dados necessários	Mensagem de cadastro inválido
Cadastro de funcionário.	Cadastro de funcionários com nome, telefone, cargo e salário.	Funcionário cadastrado com sucesso, com a saída do código do funcionário.	Cadastro do cargo sem ser m das opções disponíveis	Mensagem de cargo invalido
Cadastro de estadia, entrar com o código do cliente, quantidade de hóspede, número do quarto, data de entrada e data de saida e a quantidade de diárias	Entrar com data de entrada e saída, número do quarto, código do cliente e diárias.	Estadia cadastrada com sucesso, com a saída do código da estadia	Colocar o código do cliente errado, colocar quarto já ocupado ou com um número que não seja de 1 a 8	A mensagem aparece que não existe cliente com aquele código, número de quarto invalido, se o quarto estiver ocupado mensagem quarto não encontrado ou ocupado, impossibilitand

				o a reserva do quarto
Verificar a disponibilidade do quarto		Verifica se está livre ou ocupado, capacidade de hóspedes, e o valor da diária		
Gravar os dados em arquivo txt		Dados da estadia, cliente e funcionários grvados em “dados_hotel.txt”		

Relatório de Execução de Testes

Teste 2: Menu				
Entradas	**Classes Válidas**	**Resultado Esperado**	**Classes Inválidas**	**Resultado Esperado**
Opção: Número inteiro.	Opção escolhida entre 1 a 9	Entrada na função correspondente à opção digitda.	Opção escolhida menor que 1 ou maior que 9, e vaor não numérico	Solicitar a entrada de uma opção válida

Relatório de execução de testes		
Entradas	**Resultado**	**Aprovado?**
Valor: 1	Cadastro de Cliente	Sim
Valor: 2	Exibir Cliente	Sim
Valor: 3	Cadastro funcionário	Sim
Valor: 4	Exibir funcionários	Sim
Valor: 5	Cadastrar estadias	Sim
Valor: 6	Exibir estadias	Sim
Valor: 7	Verificar a disponibilidade de quartos	Sm
Valor: 8	Gravar os dados em arquivo	Sim
Valor: 9	Sair	Sim

Fonte: Elaborada por Maria Clara

Código

```
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

enum cargo { RECEPCIONISTA, AUXILIAR_DE_LIMPEZA, GARCOM, GERENTE };

enum statusQuarto { LIVRE, OCUPADO };

struct funcionario {
    char codigo[7];
    char nome1[100];
    char telefone[20];
    enum cargo cargo;
    float salario;
};

struct cliente {
    char codigoC[6];
    char nomeC[100];
    char endereco[100];
    char telefoneC[20];
};

struct quarto {
    int numeroQuarto;
    int capacidadeDeHospede;
    float valorDiaria;
```

```
    enum statusQuarto statusQuarto;
};

struct estadia {
    char codigoE[7];
    char codigoC[6];
    int numeroQuarto;
    int diaEntrada, mesEntrada, anoEntrada;
    int diaSaida, mesSaida, anoSaida;
    int diarias;
    int quantidadeDeHospede;
};

// Function prototypes
void gerarCodigo(char *codigo, int tamanho);
struct cliente cadastroCliente();
struct funcionario cadastroFuncionario();
void exibirCliente(struct cliente clie);
void exibirFuncionario(struct funcionario func);
void inicializarQuartos(struct quarto quartos[]);
void cadastrarEstadia(struct estadia estadias[], int *totalEstadias,
                      struct cliente clientes[], int totalClientes,
                      struct quarto quartos[], int totalQuartos);
void exibirEstadias(struct estadia estadias[], int totalEstadias,
                    struct quarto quartos[], int totalQuartos);
void verificarDisponibilidade(struct quarto quartos[], int totalQuartos);
void gravarEstadiasClientesFuncionariosEmArquivo(
    struct estadia estadias[], int totalEstadias, struct cliente clientes[],
    int totalClientes, struct funcionario funcionarios[],
```

```
    int totalFuncionarios);

void gerarCodigo(char *codigo, int tamanho) {
    const char caracteres[] = "0123456789";
    int numCaracteres = strlen(caracteres);
    for (int i = 0; i < tamanho - 1; i++) {
        codigo[i] = caracteres[rand() % numCaracteres];
    }
    codigo[tamanho - 1] = '\0';
}

// Cadastro de cliente
struct cliente cadastroCliente() {
    struct cliente clie;
    printf("Insira o nome do cliente: ");
    fgets(clie.nomeC, sizeof(clie.nomeC), stdin);
    clie.nomeC[strcspn(clie.nomeC, "\n")] = '\0';

    printf("Insira o endereço: ");
    fgets(clie.endereco, sizeof(clie.endereco), stdin);
    clie.endereco[strcspn(clie.endereco, "\n")] = '\0';

    printf("Insira o número de telefone: ");
    fgets(clie.telefoneC, sizeof(clie.telefoneC), stdin);
    clie.telefoneC[strcspn(clie.telefoneC, "\n")] = '\0';

    gerarCodigo(clie.codigoC, sizeof(clie.codigoC));
    printf("O código do cliente é: %s\n", clie.codigoC);
```

```
    return clie;
}

// Cadastrar funcionário
struct funcionario cadastroFuncionario() {
    struct funcionario func;
    printf("Insira o nome do funcionário: ");
    fgets(func.nome1, sizeof(func.nome1), stdin);
    func.nome1[strcspn(func.nome1, "\n")] = '\0';

    printf("Insira o telefone: ");
    fgets(func.telefone, sizeof(func.telefone), stdin);
    func.telefone[strcspn(func.telefone, "\n")] = '\0';

    char cargoStr[50];
    printf("Insira o cargo do funcionário (RECEPCIONISTA, AUXILIAR DE LIMPEZA,
GARCOM, GERENTE): ");
    fgets(cargoStr, sizeof(cargoStr), stdin);
    for (int i = 0; i < strlen(cargoStr); i++) {
        cargoStr[i] = toupper(cargoStr[i]);
    }
    cargoStr[strcspn(cargoStr, "\n")] = '\0';

    if (strcmp(cargoStr, "RECEPCIONISTA") == 0) {
        func.cargo = RECEPCIONISTA;
    } else if (strcmp(cargoStr, "AUXILIAR DE LIMPEZA") == 0) {
        func.cargo = AUXILIAR_DE_LIMPEZA;
    } else if (strcmp(cargoStr, "GARCOM") == 0) {
        func.cargo = GARCOM;
```

```
    } else if (strcmp(cargoStr, "GERENTE") == 0) {
        func.cargo = GERENTE;
    } else {
        printf("Cargo inválido!\n");
        exit(1);
    }

    printf("Insira o salário do funcionário: ");
    scanf("%f", &func.salario);
    while (getchar() != '\n');

    gerarCodigo(func.codigo, sizeof(func.codigo));
    printf("O código do funcionário é: %s\n", func.codigo);

    return func;
}

// Exibir os dados do cliente
void exibirCliente(struct cliente clie) {
    printf("Cliente cadastrado:\n");
    printf("Nome: %s\n", clie.nomeC);
    printf("Endereço: %s\n", clie.endereco);
    printf("Telefone: %s\n", clie.telefoneC);
    printf("Código: %s\n", clie.codigoC);
}

// Exibir os dados do funcionário
void exibirFuncionario(struct funcionario func) {
    printf("\nFuncionário cadastrado:\n");
```

```
    printf("Nome: %s\n", func.nome1);
    printf("Telefone: %s\n", func.telefone);
    printf("Cargo: ");
    switch (func.cargo) {
        case RECEPCIONISTA:
            printf("RECEPCIONISTA\n");
            break;
        case AUXILIAR_DE_LIMPEZA:
            printf("AUXILIAR DE LIMPEZA\n");
            break;
        case GARCOM:
            printf("GARÇOM\n");
            break;
        case GERENTE:
            printf("GERENTE\n");
            break;
        default:
            printf("Indefinido\n");
            break;
    }
    printf("Salário: %.2f\n", func.salario);
    printf("Código: %s\n", func.codigo);
}

// Inicializar quartos
void inicializarQuartos(struct quarto quartos[]) {
    struct quarto netuno = { 1, 2, 200.90, LIVRE };
    struct quarto urano = { 2, 3, 230.00, LIVRE };
    struct quarto saturno = { 3, 5, 330.00, LIVRE };
```

```
    struct quarto jupiter = { 4, 4, 400.00, LIVRE };
    struct quarto marte = { 5, 4, 500.00, LIVRE };
    struct quarto terra = { 6, 4, 550.00, LIVRE };
    struct quarto venus = { 7, 2, 600.00, LIVRE };
    struct quarto mercurio = { 8, 7, 750.00, LIVRE };

    quartos[0] = netuno;
    quartos[1] = urano;
    quartos[2] = saturno;
    quartos[3] = jupiter;
    quartos[4] = marte;
    quartos[5] = terra;
    quartos[6] = venus;
    quartos[7] = mercurio;
}

// Cadastrar estadia
void cadastrarEstadia(struct estadia estadias[], int *totalEstadias,
                      struct cliente clientes[], int totalClientes,
                      struct quarto quartos[], int totalQuartos) {
    if (*totalEstadias >= 100) {
        printf("Limite de estadias cadastradas atingido!\n");
        return;
    }

    struct estadia estadia;
    printf("Insira o código do cliente: ");
    fgets(estadia.codigoC, sizeof(estadia.codigoC), stdin);
    estadia.codigoC[strcspn(estadia.codigoC, "\n")] = '\0';
```

```
int encontrado = 0;
for (int i = 0; i < totalClientes; i++) {
    if (strcmp(estadia.codigoC, clientes[i].codigoC) == 0) {
        encontrado = 1;
        break;
    }
}
if (!encontrado) {
    printf("Cliente não encontrado!\n");
    return;
}

printf("Insira a quantidade de hóspedes: ");
scanf("%d", &estadia.quantidadeDeHospede);

printf("Insira o número do quarto (1 a 8): ");
scanf("%d", &estadia.numeroQuarto);
while (getchar() != '\n');

if (estadia.numeroQuarto < 1 || estadia.numeroQuarto > 8) {
    printf("Número de quarto inválido!\n");
    return;
}

struct quarto *quartoEscolhido = &quartos[estadia.numeroQuarto - 1];
if (quartoEscolhido->statusQuarto == OCUPADO) {
    printf("Quarto ocupado!\n");
    return;
```

```
    }
    if (estadia.quantidadeDeHospede > quartoEscolhido->capacidadeDeHospede) {
        printf("Capacidade de hóspedes excedida para este quarto!\n");
        return;
    }

    printf("Insira a data de entrada (dd mm aaaa): ");
    scanf("%d %d %d", &estadia.diaEntrada, &estadia.mesEntrada,
&estadia.anoEntrada);

    printf("Insira a data de saída (dd mm aaaa): ");
    scanf("%d %d %d", &estadia.diaSaida, &estadia.mesSaida, &estadia.anoSaida);

    gerarCodigo(estadia.codigoE, sizeof(estadia.codigoE));
    printf("O código da estadia é: %s\n", estadia.codigoE);

    printf("Insira a quantidade de diárias: ");
    scanf("%d", &estadia.diarias);
    while (getchar() != '\n');

    estadias[*totalEstadias] = estadia;
    (*totalEstadias)++;

    quartoEscolhido->statusQuarto = OCUPADO;

    printf("Estadia cadastrada com sucesso!\n");
}

// Exibir estadias
```

```
void exibirEstadias(struct estadia estadias[], int totalEstadias,
                    struct quarto quartos[], int totalQuartos) {
    printf("\nEstadias cadastradas:\n");
    for (int i = 0; i < totalEstadias; i++) {
        printf("Código da estadia: %s\n", estadias[i].codigoE);
        printf("Código do cliente: %s\n", estadias[i].codigoC);
        printf("Número do quarto: %d\n", estadias[i].numeroQuarto);
        printf("Data de entrada: %d/%d/%d\n", estadias[i].diaEntrada,
               estadias[i].mesEntrada, estadias[i].anoEntrada);
        printf("Data de saída: %d/%d/%d\n", estadias[i].diaSaida,
               estadias[i].mesSaida, estadias[i].anoSaida);
        printf("Diárias: %d\n", estadias[i].diarias);
        printf("Quantidade de hóspedes: %d\n", estadias[i].quantidadeDeHospede);
        printf("\n");
    }
}

// Verificar disponibilidade de quartos
void verificarDisponibilidade(struct quarto quartos[], int totalQuartos) {
    printf("\nDisponibilidade de quartos:\n");
    for (int i = 0; i < totalQuartos; i++) {
        printf("Quarto %d:\n", quartos[i].numeroQuarto);
        printf("  Capacidade de hóspedes: %d\n", quartos[i].capacidadeDeHospede);
        printf("  Valor da diária: %.2f\n", quartos[i].valorDiaria);
        printf("  Status: %s\n", quartos[i].statusQuarto == LIVRE ? "LIVRE" :
"OCUPADO");
    }
}
```

```
// Gravar dados em arquivo
void gravarEstadiasClientesFuncionariosEmArquivo(
    struct estadia estadias[], int totalEstadias, struct cliente clientes[],
    int totalClientes, struct funcionario funcionarios[],
    int totalFuncionarios) {
    FILE *arquivo = fopen("dados_hotel.txt", "w");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo para escrita!\n");
        return;
    }

    fprintf(arquivo, "Estadias:\n");
    for (int i = 0; i < totalEstadias; i++) {
        fprintf(arquivo, "Código da estadia: %s\n", estadias[i].codigoE);
        fprintf(arquivo, "Código do cliente: %s\n", estadias[i].codigoC);
        fprintf(arquivo, "Número do quarto: %d\n", estadias[i].numeroQuarto);
        fprintf(arquivo, "Data de entrada: %d/%d/%d\n", estadias[i].diaEntrada,
                estadias[i].mesEntrada, estadias[i].anoEntrada);
        fprintf(arquivo, "Data de saída: %d/%d/%d\n", estadias[i].diaSaida,
                estadias[i].mesSaida, estadias[i].anoSaida);
        fprintf(arquivo, "Diárias: %d\n", estadias[i].diarias);
        fprintf(arquivo, "Quantidade de hóspedes: %d\n",
                estadias[i].quantidadeDeHospede);
    }

    fprintf(arquivo, "\nClientes:\n");
    for (int i = 0; i < totalClientes; i++) {
        fprintf(arquivo, "Nome: %s\n", clientes[i].nomeC);
```

```
    fprintf(arquivo, "Endereço: %s\n", clientes[i].endereco);
    fprintf(arquivo, "Telefone: %s\n", clientes[i].telefoneC);
    fprintf(arquivo, "Código: %s\n", clientes[i].codigoC);
}

fprintf(arquivo, "\nFuncionários:\n");
for (int i = 0; i < totalFuncionarios; i++) {
    fprintf(arquivo, "Nome: %s\n", funcionarios[i].nome1);
    fprintf(arquivo, "Telefone: %s\n", funcionarios[i].telefone);
    fprintf(arquivo, "Cargo: ");
    switch (funcionarios[i].cargo) {
        case RECEPCIONISTA:
            fprintf(arquivo, "RECEPCIONISTA\n");
            break;
        case AUXILIAR_DE_LIMPEZA:
            fprintf(arquivo, "AUXILIAR DE LIMPEZA\n");
            break;
        case GARCOM:
            fprintf(arquivo, "GARÇOM\n");
            break;
        case GERENTE:
            fprintf(arquivo, "GERENTE\n");
            break;
        default:
            fprintf(arquivo, "Indefinido\n");
            break;
    }
    fprintf(arquivo, "Salário: %.2f\n", funcionarios[i].salario);
    fprintf(arquivo, "Código: %s\n", funcionarios[i].codigo);
```

```
    }

    fclose(arquivo);
    printf("Dados gravados em arquivo com sucesso!\n");
}

int main() {
    srand(time(NULL));

    int opcao;
    int totalClientes = 0;
    int totalFuncionarios = 0;
    int totalEstadias = 0;
    struct cliente clientes[100];
    struct funcionario funcionarios[100];
    struct estadia estadias[100];
    struct quarto quartos[8];
    inicializarQuartos(quartos);

    while (1) {
        printf("^^^^^^^^^^^^^^^^^^^^^^^^");
        printf("\n  Seja bem-vindo\n");
        printf("\nAo Hotel Bom Descanso\n");
        printf("^^^^^^^^^^^^^^^^^^^^^^^^");
        printf("\n  .....Menu.....\n");
        printf("1. Cadastrar cliente\n");
        printf("2. Exibir cliente\n");
        printf("3. Cadastrar funcionário\n");
        printf("4. Exibir funcionário\n");
```

```
printf("5. Cadastrar estadia\n");
printf("6. Exibir estadias\n");
printf("7. Verificar disponibilidade de quartos\n");
printf("8. Gravar dados em arquivo\n");
printf("9. Sair\n");
printf("Escolha uma opção: ");
scanf("%d", &opcao);
while (getchar() != '\n');

switch (opcao) {
case 1:
    clientes[totalClientes++] = cadastroCliente();
    break;
case 2:
    for (int i = 0; i < totalClientes; i++) {
        exibirCliente(clientes[i]);
    }
    break;
case 3:
    funcionarios[totalFuncionarios++] = cadastroFuncionario();
    break;
case 4:
    for (int i = 0; i < totalFuncionarios; i++) {
        exibirFuncionario(funcionarios[i]);
    }
    break;
case 5:
    cadastrarEstadia(estadias, &totalEstadias, clientes, totalClientes, quartos, 8);
    break;
```

```
        case 6:
            exibirEstadias(estadias, totalEstadias, quartos, 8);
            break;
        case 7:
            verificarDisponibilidade(quartos, 8);
            break;
        case 8:
            gravarEstadiasClientesFuncionariosEmArquivo(
                estadias, totalEstadias, clientes, totalClientes, funcionarios,
totalFuncionarios);
            break;
        case 9:
            printf("Saindo do programa...\n");
            return 0;
        default:
            printf("Opção inválida!\n");
            break;
        }
    }
}
```