Sistema de gerenciamento médico

Trabalho final - Algoritmo e Estrutura de Dados 1

Alunos: Leonardo Caparica e Milena Bernardi

Este código é um sistema de gerenciamento médico escrito em C++ que utiliza estruturas de dados como arrays, fila (queue), fila de prioridade (priority_queue) e pilha (stack) para gerenciar médicos e pacientes.

Aqui está uma explicação detalhada de cada parte do código:

Estruturas de Dados

- **Medico**: Estrutura que armazena informações de um médico, incluindo nome, CRM, CPF, telefone de emergência e endereço.
- **Paciente**: Estrutura que armazena informações de um paciente, incluindo nome, telefone, CPF, estado de emergência e se recebeu alta.

Constantes e Variáveis Globais

- **MAX_MEDICOS**: Constante que define o número máximo de médicos.
- **RED** e **RESET**: Constantes de string usadas para colorir texto no terminal (vermelho para emergência e reset para texto normal).
- **medicos**: Array de objetos Medico que armazena informações dos médicos cadastrados.
- **pacientesAguardando**: Fila de prioridade que armazena pacientes aguardando consulta, ordenados pelo estado de emergência.
- **pacientesEmConsulta**: Fila que armazena pacientes em consulta.
- **pilhaAtendidos**: Pilha que armazena pacientes que já foram atendidos.

Funções

1. **listarMedicos()**: Exibe a lista de médicos cadastrados.
2. **cadastrarPaciente()**: Permite cadastrar um novo paciente, solicitando informações ao usuário e adicionando o paciente à fila de espera.
3. **listarPacientes()**: Exibe a lista de pacientes aguardando consulta e em consulta.

4. **alocarConsulta()**: Aloca um paciente da fila de espera para consulta, respeitando o limite de médicos disponíveis.
5. **finalizarConsulta()**: Finaliza a consulta de um paciente, dando-lhe alta e movendo-o para a pilha de pacientes atendidos.
6. **listarAtendidos()**: Exibe a lista de pacientes que já foram atendidos.
7. **exibirMenu()**: Exibe o menu de opções para o usuário.

Função Principal

- **main()**: A função principal que controla o fluxo do programa. Exibe o menu, recebe a escolha do usuário e chama a função correspondente usando um loop do-while.

Fluxo do Programa

1. **Exibição do Menu**: O programa exibe o menu principal com opções para listar médicos, cadastrar pacientes, listar pacientes, alocar consultas, finalizar consultas, listar pacientes atendidos e sair.
2. **Escolha do Usuário**: O usuário escolhe uma opção digitando um número.
3. **Chamada das Funções**: Dependendo da escolha, a função correspondente é chamada:
   - **1**: listarMedicos()
   - **2**: cadastrarPaciente()
   - **3**: listarPacientes()
   - **4**: alocarConsulta()
   - **5**: finalizarConsulta()
   - **6**: listarAtendidos()
   - **7**: O programa exibe uma mensagem de saída e encerra o loop.

Detalhes Importantes

- **Uso de Cores**: As cores são usadas para destacar pacientes em estado de emergência.
- **Manipulação de Cin**: O programa utiliza cin.ignore() e cin.get() para limpar o buffer de entrada e esperar que o usuário pressione Enter antes de voltar ao menu.
- **Estruturas de Dados**: A fila de prioridade (priority_queue) garante que pacientes em estado de emergência sejam atendidos primeiro. A pilha (stack) é usada para manter o histórico de pacientes atendidos.

Código Completo:

```
#include <iostream>
```

```cpp
#include <string>
#include <stack>
#include <queue>
#include <vector>
#include <functional>
using namespace std;

const int MAX_MEDICOS = 5;
const string RED = "\033[31m";
const string RESET = "\033[0m";

struct Medico {
    string nome;
    string crm;
    string cpf;
    string telefoneEmergencia;
    string endereco;
};

struct Paciente {
    string nome;
    bool alta;
    string telefone;
    string cpf;
    int emergencia;

    bool operator<(const Paciente &other) const
    {
```

```
        return emergencia < other.emergencia;
    }
};

// Arrays para armazenar médicos e pacientes
Medico medicos[MAX_MEDICOS] = {
    {"Dr. Joao Silva", "123456", "111.111.111-11", "21999999999", "Rua A, 123"},
    {"Dra. Maria Souza", "654321", "222.222.222-22", "21988888888", "Rua B, 456"},
    {"Dr. Pedro Santos", "987654", "333.333.333-33", "21977777777", "Rua C, 789"},
    {"Dra. Ana Lima", "456789", "444.444.444-44", "21966666666", "Rua D, 101"},
    {"Dr. Lucas Oliveira", "321654", "555.555.555-55", "21955555555", "Rua E, 202"}};

priority_queue<Paciente> pacientesAguardando; // Fila de pacientes aguardando consulta
queue<Paciente> pacientesEmConsulta;         // Fila de pacientes em consulta
stack<Paciente> pilhaAtendidos;              // Pilha de pacientes atendidos

void listarMedicos(){
    cout << "Lista de Medicos:" << endl;
    for (int i = 0; i < MAX_MEDICOS; i++){
        cout << "Nome: " << medicos[i].nome << ", CRM: " << medicos[i].crm << ", CPF: " << medicos[i].cpf
            << ", Telefone de Emergencia: " << medicos[i].telefoneEmergencia << ", Endereco: " <<
medicos[i].endereco << endl;
    }
    cout << "Pressione Enter para voltar ao menu." << endl;
    cin.ignore();
    cin.get();
}

void cadastrarPaciente(){
```

```
    Paciente p;
    cout << "Digite o nome do paciente: ";
    cin.ignore();
    getline(cin, p.nome);
    cout << "Digite o telefone do paciente: ";
    getline(cin, p.telefone);
    cout << "Digite o CPF do paciente: ";
    getline(cin, p.cpf);
    cout << "O paciente está em estado de emergência? (1 para sim, 0 para não): ";
    cin >> p.emergencia;
    p.alta = false;

    pacientesAguardando.push(p);

    cout << "Paciente cadastrado com sucesso!" << endl;
}

void listarPacientes(){
    if (pacientesAguardando.empty() && pacientesEmConsulta.empty()){
        cout << "Nenhum paciente cadastrado." << endl;
        return;
    }

    cout << "Lista de Pacientes Aguardando Consulta:" << endl;
    priority_queue<Paciente> tempQueue = pacientesAguardando; // Fazer uma cópia da fila para exibir os pacientes
    while (!tempQueue.empty()){
        Paciente p = tempQueue.top();
        tempQueue.pop();
```

```
        cout << (p.emergencia ? RED : "") << "Nome: " << p.nome << ", Telefone: " << p.telefone << ", CPF: " << p.cpf
            << ", Emergência: " << (p.emergencia ? "Sim" : "Não") << ", Alta: " << (p.alta ? "Sim" : "Não") << RESET << endl;
    }

    cout << "Lista de Pacientes em Consulta:" << endl;
    queue<Paciente> tempQueueConsulta = pacientesEmConsulta; // Fazer uma cópia da fila para exibir os pacientes em consulta
    while (!tempQueueConsulta.empty()){
        Paciente p = tempQueueConsulta.front();
        tempQueueConsulta.pop();
        cout << (p.emergencia ? RED : "") << "Nome: " << p.nome << ", Telefone: " << p.telefone << ", CPF: " << p.cpf
            << ", Emergência: " << (p.emergencia ? "Sim" : "Não") << ", Alta: " << (p.alta ? "Sim" : "Não") << RESET << endl;
    }

    cout << "Pressione Enter para voltar ao menu." << endl;
    cin.ignore();
    cin.get();
}

void alocarConsulta(){
    if (pacientesAguardando.empty()){
        cout << "Nenhum paciente para alocar consulta." << endl;
        return;
    }
    else if (pacientesEmConsulta.size() == MAX_MEDICOS){
        cout << RED << "TODOS OS MEDICOS ESTAO OCUPADOS!!!" << endl;
        return;
```

```
    }
    else {
        Paciente p = pacientesAguardando.top();
        pacientesAguardando.pop();
        p.alta = false; // Paciente ainda não foi atendido
        pacientesEmConsulta.push(p);
        cout << "Consulta alocada para o paciente: " << p.nome << endl;
    }
}

void finalizarConsulta(){
    if (pacientesEmConsulta.empty()){
        cout << "Nenhum paciente em consulta para finalizar." << endl;
        return;
    }

    cout << "Escolha o paciente para dar alta:" << endl;
    vector<Paciente> tempQueue;
    int index = 0;

    while (!pacientesEmConsulta.empty()){
        Paciente p = pacientesEmConsulta.front();
        pacientesEmConsulta.pop();
        tempQueue.push_back(p);
        cout << (p.emergencia ? RED : "") << index++ << ". Nome: " << p.nome << ", Telefone: " << p.telefone << ",
CPF: " << p.cpf
            << ", Emergência: " << (p.emergencia ? "Sim" : "Não") << RESET << endl;
    }
```

```
    int escolha;
    cout << "Digite o número do paciente: ";
    cin >> escolha;

    if (escolha >= 0 && escolha < tempQueue.size()){
        Paciente p = tempQueue[escolha];
        p.alta = true;
        cout << "Paciente " << p.nome << " recebeu alta." << endl;
        pilhaAtendidos.push(p);
    }
    else {
        cout << "Escolha inválida." << endl;
    }

    // Colocar de volta os pacientes restantes na fila de consulta
    for (int i = 0; i < tempQueue.size(); ++i){
        if (i != escolha){
            pacientesEmConsulta.push(tempQueue[i]);
        }
    }
}

void listarAtendidos(){
    if (pilhaAtendidos.empty()){
        cout << "Nenhum paciente atendido." << endl;
        return;
    }
```

```
    cout << "Lista de Pacientes Atendidos:" << endl;
    stack<Paciente> tempStack = pilhaAtendidos; // Fazer uma cópia da pilha para exibir os pacientes
    while (!tempStack.empty()){
        Paciente p = tempStack.top();
        tempStack.pop();
        cout << (p.emergencia ? RED : "") << "Nome: " << p.nome << ", Telefone: " << p.telefone << ", CPF: " << p.cpf
            << ", Emergência: " << (p.emergencia ? "Sim" : "Não") << ", Alta: " << (p.alta ? "Sim" : "Não") << RESET <<
endl;
    }
    cout << "Pressione Enter para voltar ao menu." << endl;
    cin.ignore();
    cin.get();
}

void exibirMenu(){
    system("cls");
    cout << "Sistema de Gerenciamento Médico" << endl
        << endl;
    cout << "1. Listar Médicos" << endl;
    cout << "2. Cadastrar Paciente" << endl;
    cout << "3. Listar Pacientes" << endl;
    cout << "4. Alocar Consulta" << endl;
    cout << "5. Finalizar Consulta e Dar Alta" << endl;
    cout << "6. Listar Pacientes Atendidos" << endl;
    cout << "7. Sair" << endl;
    cout << "Escolha uma opção: ";
}
```

```
int main(){
    int opcao;

    do {
        exibirMenu();
        cin >> opcao;

        switch (opcao)
        {
        case 1:
            listarMedicos();
            break;
        case 2:
            cadastrarPaciente();
            break;
        case 3:
            listarPacientes();
            break;
        case 4:
            alocarConsulta();
            cin.ignore();
            cin.get();
            break;
        case 5:
            finalizarConsulta();
            cin.ignore();
            cin.get();
```

```
            break;
        case 6:
            listarAtendidos();
            cin.ignore();
            cin.get();
            break;
        case 7:
            cout << "Saindo..." << endl;
            break;
        default:
            cout << "Opção inválida. Tente novamente." << endl;
        }
    } while (opcao != 7);

    return 0;
}
```