**Estruturas de Dados**

1. **Struct Paciente**
   * Representa um paciente.
   * Campos: nome, idade, motivoConsulta, prioridade, inicioAtendimento, fimAtendimento, numeroConsultorio, proximo.
   * proximo é um ponteiro para o próximo paciente na fila.
2. **Struct Fila**
   * Representa uma fila de pacientes.
   * Campos: inicio, fim.
   * inicio é um ponteiro para o primeiro paciente na fila.
   * fim é um ponteiro para o último paciente na fila.
3. **Struct HistoricoAtendimentos**
   * Representa o histórico de atendimentos.
   * Campos: inicio, fim.
   * Similar à struct Fila, mas armazena pacientes atendidos.

**Funções**

1. **Fila \*criarFila()**
   * Cria e inicializa uma nova fila.
   * Retorna um ponteiro para a fila criada.
2. **HistoricoAtendimentos \*criarHistorico()**
   * Cria e inicializa um novo histórico de atendimentos.
   * Retorna um ponteiro para o histórico criado.
3. **Paciente \*criarPaciente(char \*nome, int idade, char \*motivoConsulta, int prioridade)**
   * Cria e inicializa um novo paciente com os dados fornecidos.
   * Retorna um ponteiro para o paciente criado.
4. **void adicionarPaciente(Fila \*f, Paciente \*p)**
   * Adiciona um paciente ao final da fila especificada.
   * Parâmetros: ponteiro para a fila (f), ponteiro para o paciente (p).
5. **Paciente \*atenderPaciente(Fila \*filaUrgente, Fila \*filaPrioritaria, Fila \*filaNormal, int numeroConsultorio)**
   * Atende o próximo paciente da fila com a maior prioridade disponível.
   * Atualiza o tempo de início do atendimento e o número do consultório.
   * Retorna um ponteiro para o paciente atendido.
6. **void finalizarAtendimento(Paciente \*p)**
   * Define o tempo de fim de atendimento para o paciente especificado.
   * Parâmetro: ponteiro para o paciente (p).
7. **void removerPaciente(Paciente \*p)**
   * Libera a memória alocada para o paciente especificado.
   * Parâmetro: ponteiro para o paciente (p).
8. **void visualizarFila(Fila \*f)**
   * Exibe os pacientes na fila especificada.
   * Parâmetro: ponteiro para a fila (f).
9. **void adicionarHistorico(HistoricoAtendimentos \*historico, Paciente \*p)**
   * Adiciona um paciente ao final do histórico de atendimentos.
   * Parâmetros: ponteiro para o histórico (historico), ponteiro para o paciente (p).
10. **void visualizarHistorico(HistoricoAtendimentos \*historico)**
    * Exibe os pacientes no histórico de atendimentos.
    * Parâmetro: ponteiro para o histórico (historico).

**Função main()**

1. **Inicialização**
   * Cria as filas (filaNormal, filaPrioritaria, filaUrgente) e o histórico de atendimentos (historico).
2. **Loop do Menu**
   * Exibe um menu de opções ao usuário.
   * Lê a escolha do usuário e executa a ação correspondente:
     1. Adicionar paciente.
     2. Visualizar filas.
     3. Atender paciente.
     4. Visualizar histórico de atendimentos.
     5. Sair do sistema.

**Uso de Funções de Biblioteca**

1. **#include <stdio.h>**
   * Inclui funções de entrada e saída, como printf e scanf.
2. **#include <stdlib.h>**
   * Inclui funções de gerenciamento de memória, como malloc e free.
3. **#include <string.h>**
   * Inclui funções de manipulação de strings, como strcpy.
4. **#include <time.h>**
   * Inclui funções de manipulação de tempo, como time.

**Como Utilizá-los**

* **Structs e Ponteiros**
  + As structs são usadas para representar dados complexos.
  + Ponteiros são usados para manipular e acessar esses dados dinamicamente.
* **Funções**
  + As funções são utilizadas para modularizar o código, facilitando a manutenção e a reutilização.
  + A alocação dinâmica de memória (malloc) permite a criação de estruturas de dados flexíveis.
* **Manipulação de Fila**
  + Os pacientes são adicionados ao final da fila e atendidos a partir do início, seguindo uma lógica FIFO (First In, First Out) modificada para priorização.
* **Gerenciamento de Tempo**
  + A função time é usada para registrar os tempos de início e fim dos atendimentos, permitindo a análise do tempo de atendimento.