Atividade B2-4 - Implementar fila de atendimento – Hospital

Enunciado:

Instruções

Vimos que em ciência da computação, as filas são uma das estruturas de dados fundamentais amplamente utilizadas para organizar e manipular elementos em uma sequência específica. No mundo, por exemplo, as pessoas aguardam sua vez em uma ordem linear, enquanto que uma fila de dados é uma coleção ordenada de elementos.

Imagine o seguinte cenários:

Uma clínica médica precisa gerenciar a chegada e o atendimento de pacientes. A clínica possui uma equipe de médicos e enfermeiros que estão disponíveis para realizar consultas e procedimentos. A clínica lida com dois tipos de pacientes: aqueles que requerem atendimento urgente e aqueles que estão na fila normal.

Seu Objetivo: Implementar um sistema de filas de atendimento que leve em consideração a prioridade desses pacientes: Normal, prioritário (gestantes; +60 anos), URGENTE (De acordo com estado clínico).

No momento da entrada, atendente sinaliza na ficha de entrada a prioridade, conforme descrito acima.

Requisitos do sistema:

O sistema de filas de atendimento deve ser capaz de:

1. Adicionar pacientes à fila:
   - O sistema deve permitir a adição de pacientes à fila normal.
   - O sistema deve permitir a adição de pacientes à fila de atendimento urgente.
2. Atender pacientes:
   - O sistema deve atender aos pacientes de acordo com a prioridade de atendimento. Esta prioridade deve ser atendida a cada "n" pacientes da fila normal .
   - Os pacientes da fila de atendimento urgente devem ser atendidos IMEDIATAMENTE, devem ser atendidos antes dos pacientes da fila normal e da fila dos prioritários.
   - Caso haja mais de um paciente na fila de atendimento urgente, o atendimento deve ser realizado de acordo com a ordem de chegada. Esta mesma regra deve ser adotada aos paciente de fila prioritária
   - O sistema deve registrar o tempo de início e término do atendimento de cada paciente.
3. Remover pacientes atendidos:
   - Após o atendimento, o sistema deve remover o paciente da fila.
4. Visualizar informações da fila:
   - O sistema deve permitir a visualização da fila de pacientes da fila normal.
   - O sistema deve permitir a visualização da fila de pacientes da fila de atendimento urgente.
   - Para cada paciente na fila, as informações relevantes devem ser exibidas, como nome, idade e motivo da

4. Visualizar informações da fila:
   - O sistema deve permitir a visualização da fila de pacientes da fila normal.
   - O sistema deve permitir a visualização da fila de pacientes da fila de atendimento urgente.
   - Para cada paciente na fila, as informações relevantes devem ser exibidas, como nome, idade e motivo da consulta.

   -O sistema deve mostrar no painel o histórico de atendimentos (Fila de chamadas atendidas)
   -O sistema deve informar no painel o paciente que está sendo chamado, neste caso, informa o número do consultório

Requisitos técnicos:

1-Dar a solução na linguagem C
2-Implementar utilizando o paradigma de fila
3-A estrutura de dados "fila" deve conter métodos necessários para atender aos Requisitos do sistema:

Código Fonte:

Link para o repositório: https://github.com/CamilleGFAlmeida/Fatec-AMS-ED2024-1-1681432412033-Camille/tree/main/Atividade-B2-4-Hospital

```
/*------------------------------------------------------------------------*/
/*   FATEC - São Caetano do Sul              Estrutura de Dados     */
/*                                                                        */
/*                              Camille Guillen                           */
/*            Objetivo: Implementação de Filas - Hospital                 */
/*                                                                        */
/*                                                  Data:02/05/2024  */
/*------------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_NAME_LENGTH 50

// Definição da estrutura de paciente
typedef struct {
    char name[MAX_NAME_LENGTH];
    int age;
    char priority; // 'N' para normal, 'P' para prioritário, 'U' para urgente
    char reason[100];
} Patient;

// Definição do nó da fila
typedef struct Node {
    Patient patient;
```

```
    struct Node* next;
} Node;

// Definição da fila
typedef struct {
    Node* front;
    Node* rear;
} Queue;

// Função para inicializar a fila
void initializeQueue(Queue* queue) {
    queue->front = queue->rear = NULL;
}

// Função para verificar se a fila está vazia
int isEmpty(Queue* queue) {
    return queue->front == NULL;
}

// Função para adicionar paciente à fila
void enqueue(Queue* queue, Patient patient) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->patient = patient;
    newNode->next = NULL;

    if (isEmpty(queue)) {
        queue->front = queue->rear = newNode;
    } else {
        queue->rear->next = newNode;
        queue->rear = newNode;
```

```
    }
}

// Função para remover e obter o próximo paciente da fila
Patient dequeue(Queue* queue) {
    if (isEmpty(queue)) {
        printf("A fila está vazia.\n");
        exit(EXIT_FAILURE);
    }

    Node* temp = queue->front;
    Patient patient = temp->patient;

    queue->front = queue->front->next;
    free(temp);

    if (queue->front == NULL) {
        queue->rear = NULL;
    }

    return patient;
}

// Função para visualizar a fila
void displayQueue(Queue* queue, char priority) {
    Node* current = queue->front;

    printf("Fila %s:\n", (priority == 'N') ? "Normal" : ((priority == 'P') ?
"Prioritária" : "Urgente"));
```

```
    while (current != NULL) {
        if (current->patient.priority == priority) {
            printf("Nome: %s, Idade: %d, Motivo: %s\n", current->patient.name,
current->patient.age, current->patient.reason);
        }
        current = current->next;
    }
    printf("\n");
}

// Função para adicionar um paciente à fila com base nas informações fornecidas
pelo usuário
void addPatient(Queue* normalQueue, Queue* prioritaryQueue, Queue*
urgentQueue) {
    Patient newPatient;
    printf("\nPara casos prioritários, digite: \n > - Gestantes \n > - Idosos (60 anos
ou mais) \n > - Problemas Graves\n");
    printf("\nDigite o nome do paciente: ");
    scanf("%s", newPatient.name);
    printf("\nDigite a idade do paciente: ");
    scanf("%d", &newPatient.age);
    printf("\nDigite o motivo da consulta: ");
    scanf(" %[^\n]s", newPatient.reason);

    printf("\nDigite a prioridade do paciente ('N' para normal, 'P' para prioritário,
'U' para urgente): ");

    scanf(" %c", &newPatient.priority);

    if (newPatient.priority == 'P') {
        // Verificar se o paciente atende aos critérios para ser incluído na fila
prioritária
```

```
        if (strcmp(newPatient.reason, "Problemas Graves") == 0 || newPatient.age
>= 60 || strcmp(newPatient.reason, "Gestante") == 0) {
            enqueue(prioritaryQueue, newPatient);
            printf("Paciente adicionado à fila prioritária com sucesso.\n");
        } else {
            printf("Este paciente não atende aos critérios para ser incluído na fila
prioritária.\n");
            enqueue(normalQueue, newPatient);
            printf("Paciente adicionado à fila normal.\n");
        }
    } else {
        switch (newPatient.priority) {
            case 'N':
                enqueue(normalQueue, newPatient);
                break;
            case 'U':
                enqueue(urgentQueue, newPatient);
                break;
            default:
                printf("Prioridade inválida. O paciente não será adicionado à fila.\n");
        }
    }
}


// Função para atender pacientes com base na prioridade
void servePatients(Queue* normalQueue, Queue* prioritaryQueue, Queue*
urgentQueue) {
    printf("\nAtendendo pacientes...\n");


    if (!isEmpty(urgentQueue)) {
        Patient urgentPatient = dequeue(urgentQueue);
```

```
        printf("\nAtendendo paciente urgente: %s, idade %d, motivo: %s\n",
urgentPatient.name, urgentPatient.age, urgentPatient.reason);
    } else if (!isEmpty(prioritaryQueue)) {
        Patient prioritaryPatient = dequeue(prioritaryQueue);
        printf("\nAtendendo paciente prioritário: %s, idade %d, motivo: %s\n",
prioritaryPatient.name, prioritaryPatient.age, prioritaryPatient.reason);
    } else if (!isEmpty(normalQueue)) {
        Patient normalPatient = dequeue(normalQueue);
        printf("\nAtendendo paciente normal: %s, idade %d, motivo: %s\n",
normalPatient.name, normalPatient.age, normalPatient.reason);
    } else {
        printf("\nNão há pacientes para atender.\n");
    }
}

// Função para remover pacientes após atendimento
void removePatient(Queue* queue) {
    if (!isEmpty(queue)) {
        dequeue(queue);
        printf("Paciente removido da fila após o atendimento.\n");
    } else {
        printf("Não há pacientes na fila para remover.\n");
    }
}

// Função para exibir o menu de opções
void displayMenu() {
    printf("\n\nMenu:\n");
    printf("1. Adicionar paciente à fila\n");
    printf("2. Atender pacientes\n");
    printf("3. Remover paciente da fila após o atendimento\n");
```

```
    printf("4. Visualizar fila de pacientes\n");
    printf("5. Sair\n");
    printf("\nEscolha uma opção: ");
}

int main() {
    Queue normalQueue, prioritaryQueue, urgentQueue;
    initializeQueue(&normalQueue);
    initializeQueue(&prioritaryQueue);
    initializeQueue(&urgentQueue);

    int choice;
    do {
        displayMenu();
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addPatient(&normalQueue, &prioritaryQueue, &urgentQueue);
                break;
            case 2:
                servePatients(&normalQueue, &prioritaryQueue, &urgentQueue);
                break;
            case 3:
                removePatient(&normalQueue);
                removePatient(&prioritaryQueue);
                removePatient(&urgentQueue);
                break;
            case 4:
                displayQueue(&normalQueue, 'N');
```

```
            displayQueue(&prioritaryQueue, 'P');
            displayQueue(&urgentQueue, 'U');
            break;
        case 5:
            printf("\nEncerrando o programa...\n");
            break;
        default:
            printf("\nOpção inválida. Por favor, escolha outra opção.\n");
    }
  } while (choice != 5);

  return 0;
}
```