09.09.2024

Tarefa 2 - Listas Encadeadas

Júlia Tadeu - 2312392

Theo Canuto - 2311293

Professor Luiz Fernando Seibel

INF1010 - 3WA

2024.2

1. **Objetivo**

O objetivo deste trabalho é implementar um programa que gerencie a prioridade dos atendimentos na emergência de um hospital utilizando uma lista encadeada. Através da definição de um Tipo Abstrato de Dados (TAD) adequado, o sistema deverá ordenar os pacientes de acordo com sua prioridade de atendimento, levando em consideração tanto o estado de saúde (indicado por cores), quanto a ordem de chegada.

Esse exercício visa proporcionar uma compreensão aprofundada sobre a estrutura de dados de listas encadeadas, aplicando conceitos teóricos em um cenário prático e realista. O propósito deste trabalho é fortalecer a capacidade dos alunos em manipular estruturas dinâmicas de dados, utilizando listas encadeadas para gerenciar informações que variam em tempo real, como é o caso da fila de atendimento hospitalar.

Além disso, o trabalho reforça o entendimento sobre a importância de priorizar informações em algoritmos, demonstrando como listas encadeadas podem ser utilizadas para solucionar problemas que demandam organização e processamento eficiente de dados.

2. **Estrutura do programa**

O programa é composto por 2 estruturas de dados e 6 funções principais, além da função *main*. As estruturas de dados são para os pacientes e para a fila. Cada função desempenha um papel essencial para implementar o sistema de gestão de filas de atendimento hospitalar, levando em consideração as prioridades definidas pelas cores (*red* para alta prioridade, *yellow* para média prioridade e *green* para baixa prioridade). A seguir, são descritas essas estruturas de dados e funções.

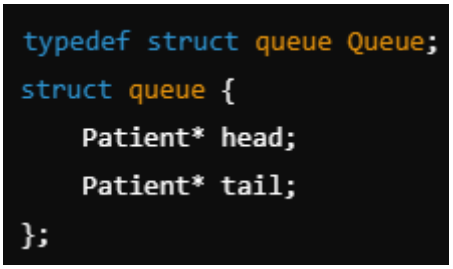
Estrutura de dados *Patient:*

```
typedef struct patient Patient;
struct patient {
    int num;
    char color;
    Patient* next;
};
```

A estrutura *Patient* define os pacientes, contendo:

- *num*: Número de identificação do paciente.
- *color*: Cor da prioridade do paciente ('R' para red, 'Y' para yellow, 'G' para green).
- *next*: Ponteiro para o próximo paciente na fila.

Estrutura da dados *Queue:*



A estrutura *Queue* define a fila de pacientes, contendo:

- *head*: Ponteiro para o primeiro paciente da fila.
- *tail*: Ponteiro para o último paciente da fila.

Função *createPatient(int num, char color):*

- **Descrição:** Esta função cria um novo paciente e inicializa suas propriedades: número de identificação (*num*), cor de prioridade (*color*) e o ponteiro para o próximo paciente na fila (*next*). O paciente é alocado dinamicamente na memória.
- **Parâmetros:** *int num*: Número do paciente, representando a ordem de chegada; *char color*: Cor da pulseira do paciente, indicando sua prioridade: 'R' (*red*), 'Y' (*yellow*), ou 'G' (*green*).
- **Retorno:** Retorna um ponteiro para o novo paciente criado.

Função *enqueue(Queue* q, Patient* p)*

- **Descrição:** Insere um paciente na fila (*queue*) de acordo com sua prioridade. Pacientes com pulseira vermelha têm prioridade máxima e são inseridos no início da fila. Pacientes com pulseira amarela têm prioridade intermediária e são inseridos antes dos pacientes com pulseira verde, mas depois dos pacientes com pulseira vermelha. Pacientes com pulseira verde são inseridos no final da fila.
- **Parâmetros:** *Queue* q:* Ponteiro para a fila onde o paciente será inserido; *Patient* p:* Ponteiro para o paciente a ser inserido.
- **Retorno:** Não possui retorno (*void*).

Função *dequeue(Queue* q)*

- **Descrição:** Remove o paciente com maior prioridade do início da fila e retorna um ponteiro para o paciente removido. O paciente removido é aquele que será atendido.
- Parâmetros: *Queue* q:* Ponteiro para a fila de onde o paciente será removido.
- **Retorno:** Retorna um ponteiro para o paciente removido. Caso a fila esteja vazia, retorna *NULL*.

Função *removePatient(Queue* q, int num, char color)*

- **Descrição:** Remove um paciente específico da fila, identificado pelo número (*num*) e pela cor de prioridade (*color*). Esta função simula a situação em que um paciente desiste do atendimento, sendo removido da fila sem ser atendido.
- **Parâmetros:** *Queue* q:* Ponteiro para a fila de onde o paciente será removido; *int num:* Número do paciente a ser removido; *char color:* Cor da prioridade do paciente.
- **Retorno:** Não possui retorno (*void*).

Função *printQueue(Queue* q)*

- **Descrição:** Exibe o estado atual da fila, listando todos os pacientes com suas respectivas cores de prioridade e números de identificação. Também exibe a contagem de quantos pacientes estão em cada categoria de prioridade (*red*, *yellow*, *green*).
- **Parâmetros:** *Queue* q:* Ponteiro para a fila cujos pacientes serão exibidos.
- **Retorno:** Não possui retorno (*void*).

Função *manageQueue(FILE* file, Queue* q)*

- **Descrição:** Processa um arquivo de entrada contendo uma lista de operações. As operações podem incluir a inserção de novos pacientes na fila (E), a remoção de pacientes que foram atendidos (S), ou a desistência de um paciente (D). O programa lê o arquivo linha por linha e realiza a operação correspondente. Após cada operação, a fila é impressa para mostrar o estado atualizado.
- **Parâmetros:** *FILE* file:* Ponteiro para o arquivo de entrada que contém as instruções de operação; *Queue* q:* Ponteiro para a fila que será manipulada.
- **Retorno:** Não possui retorno (*void*).

Função *main()*

- **Descrição:** É o ponto de entrada do programa. Nesta função, o arquivo de entrada com as instruções é aberto, a fila de pacientes é inicializada e a função *manageQueue* é chamada para gerenciar a fila de acordo com as operações lidas no arquivo. Ao final da execução, o arquivo é fechado e o programa é encerrado.
- **Parâmetros:** Não possui parâmetros.
- **Retorno:** Retorna um inteiro (*int*) indicando o status de execução do programa, onde 0 geralmente indica que o programa foi executado com sucesso.

3. Soluções

A solução para o gerenciamento da fila de pacientes na emergência foi desenvolvida com base na utilização de listas encadeadas, respeitando as prioridades estabelecidas pelo hospital (*red*, *yellow*, *green*). Abaixo está o passo a passo da solução implementada:

a. Leitura e Processamento do Arquivo de Entrada: O programa começa lendo as instruções de um arquivo de texto que contém operações como a chegada de novos pacientes, atendimento e desistências. Cada linha do arquivo contém uma ação e as informações do paciente. A função *manageQueue* é responsável por processar cada operação e chamar as funções correspondentes para adicionar, remover ou atender pacientes na fila.

b. Inserção de Pacientes: Quando um paciente é adicionado à fila, sua prioridade é determinada pela cor da pulseira(*red*, *yellow*, *green*). Pacientes com pulseiras vermelhas são inseridos no início da fila, pois têm prioridade máxima. Pacientes com pulseiras amarelas são inseridos depois dos vermelhos, mas antes dos pacientes com pulseiras verdes, que têm a menor prioridade. A função *enqueue* lida com essa lógica de ordenação ao inserir novos pacientes na fila.

c. Atendimento de Pacientes: Quando um paciente é atendido, ele é removido da fila usando a função dequeue. O paciente removido sempre é o primeiro da fila, respeitando a ordem de prioridade estabelecida no momento da inserção. Após o atendimento de cada paciente, a função *printQueue* é chamada para mostrar a lista atualizada e a quantidade de pacientes em cada categoria de prioridade.

d. Desistência de Pacientes: Pacientes que desistem do atendimento são removidos da fila utilizando a função *removePatient*. A função busca o paciente na lista encadeada e o remove, ajustando os ponteiros para manter a integridade da fila. Quando um paciente desiste, a fila é atualizada e impressa novamente.

e. Impressão da Fila: A cada inserção, atendimento ou desistência, a função *printQueue* exibe o estado atual da fila, incluindo a contagem de quantos pacientes há em cada cor de prioridade. Isso permite ao usuário acompanhar em tempo real as mudanças na fila.

Testes e Resultados:

Foram realizados testes utilizando diferentes cenários de entrada para garantir que o programa se comportasse corretamente em diversas situações. Aqui estão alguns cenários de teste:

- **Cenário 1:** Inserção de Pacientes com Diferentes Prioridades

Arquivo de entrada:

Descrição: Neste cenário, cinco pacientes são inseridos na fila, sendo dois com prioridade *red*, um com prioridade *yellow*, e dois com prioridade *green*.

Esperado: A fila deve priorizar os pacientes *red*, seguidos pelo paciente *yellow* e, por fim, os pacientes *green*.

Saída:

```
Patient 1 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 0, Green: 1

Patient 2 with RED (high) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 1 - Priority GREEN (low)
Red: 1, Yellow: 0, Green: 1

Patient 3 with YELLOW (medium) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 1

Patient 4 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 2

Patient 5 with RED (high) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 5 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 2, Yellow: 1, Green: 2
```

- **Cenário 2:** Atendimento de Paciente com Prioridade Máxima

Arquivo de entrada:

```
E 1 G
E 2 R
E 3 Y
E 4 G
S 2 R
```

Descrição: O paciente 2 com prioridade *red* é atendido e removido da fila.

Esperado: O paciente 2 deve ser removido da fila e a fila restante deve ser exibida.
Saída:

```
Patient 1 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 0, Green: 1

Patient 2 with RED (high) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 1 - Priority GREEN (low)
Red: 1, Yellow: 0, Green: 1

Patient 3 with YELLOW (medium) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 1

Patient 4 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 2
```

- **Cenário 3:** Desistência de Paciente

  Arquivo de entrada:

```
E 1 G
E 2 R
E 3 Y
E 4 G
D 3 Y
```

Descrição: O paciente 3 com prioridade *yellow* desiste do atendimento.

Esperado: O paciente 3 deve ser removido da fila e a fila restante deve ser exibida.

Saída:

```
Patient 1 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 0, Green: 1

Patient 2 with RED (high) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 1 - Priority GREEN (low)
Red: 1, Yellow: 0, Green: 1

Patient 3 with YELLOW (medium) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 1

Patient 4 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 2

Patient 3 with YELLOW (medium) priority left the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 1, Yellow: 0, Green: 2
```

- **Cenário 4:** Inserção de Novos Pacientes Após Atendimento

  Arquivo de entrada:

```
E 1 G
E 2 R
S 2 R
E 3 Y
E 4 G
E 5 R
```

  Descrição: Após o atendimento do paciente 2, novos pacientes são inseridos na fila, incluindo um novo paciente com prioridade vermelha.

  Esperado: A fila deve priorizar o paciente 5, com prioridade *red* após a remoção do paciente 2.

Saída:

```
Patient 1 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 0, Green: 1

Patient 2 with RED (high) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 1 - Priority GREEN (low)
Red: 1, Yellow: 0, Green: 1

Patient 2 with RED (high) priority was treated (removed from the queue).


Patient list:
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 0, Green: 1

Patient 3 with YELLOW (medium) priority joined the hospital queue.


Patient list:
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 1, Green: 1

Patient 4 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 0, Yellow: 1, Green: 2

Patient 5 with RED (high) priority joined the hospital queue.


Patient list:
Patient 5 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 2
```

- **Cenário 5:** Desistência Dupla do Mesmo Paciente

Arquivo de entrada:

```
E 1 G
E 2 R
E 3 Y
E 4 G
D 3 Y
D 3 Y
```

Descrição: O paciente com prioridade amarela (3) desiste do atendimento. Em seguida, uma segunda tentativa de remover o mesmo paciente (que já não está na fila) é feita.

Esperado: Na primeira operação de desistência, o paciente 3 é removido corretamente da fila. Na segunda tentativa, o programa deve indicar que o paciente não está mais na fila.

Saída:

```
Patient 1 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 0, Green: 1

Patient 2 with RED (high) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 1 - Priority GREEN (low)
Red: 1, Yellow: 0, Green: 1

Patient 3 with YELLOW (medium) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 1

Patient 4 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 2

Patient 3 with YELLOW (medium) priority left the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 1, Yellow: 0, Green: 2

Couldn't find Patient 3.


Patient list:
Patient 2 - Priority RED (high)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 1, Yellow: 0, Green: 2
```

- **Cenário 6:** Desistência de Paciente Já Atendido

  Arquivo de entrada:

Descrição: O paciente 2 com prioridade *red* é atendido e removido da fila. Em seguida, uma tentativa de desistência para o mesmo paciente é realizada.

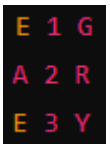
Esperado: O programa deve atender o paciente 2 corretamente e removê-lo da fila. Quando for feita a tentativa de desistência de um paciente já atendido, o programa deve informar que o paciente não está mais na fila.

Saída:

```
Patient 3 with YELLOW (medium) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 1

Patient 2 with RED (high) priority was treated (removed from the queue).


Patient list:
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 1, Green: 1

Couldn't find Patient 2.


Patient list:
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 1, Green: 1
```

- **Cenário 7:** Ação Inválida

  Arquivo de entrada:



Descrição: Neste cenário, é realizada uma operação com uma ação inválida ('A') que não corresponde às operações esperadas ('E', 'S', ou 'D').

Esperado: O programa deve reportar um erro ao encontrar a operação inválida e continuar processando as operações restantes.

Saída:

```
Patient 1 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 0, Green: 1

Invalid action: A is neither an Entrance, an Exit or a Desistance.


Patient list:
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 0, Green: 1

Patient 3 with YELLOW (medium) priority joined the hospital queue.


Patient list:
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 1, Green: 1
```

- **Cenário 7:** Cor Inválida

  Arquivo de entrada:

```
E 1 G
E 2 X
E 3 Y
```

Descrição: Neste cenário, um paciente é inserido com uma cor de prioridade inválida ('X'), que não está entre as cores aceitas ('R', 'Y', ou 'G').

Esperado: O programa deve rejeitar a operação com a cor inválida e continuar processando as operações válidas.

Saída:

```
Patient 1 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 0, Green: 1

Invalid priority. X is neither RED (high), YELLOW (medium) or GREEN (low) priority
Patient list:
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 0, Green: 1

Patient 3 with YELLOW (medium) priority joined the hospital queue.


Patient list:
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 1, Green: 1
```

4. **Observações e conclusões**

Durante o desenvolvimento do programa, algumas dificuldades e facilidades se destacaram. Abaixo estão as principais observações:

Facilidades:

- A utilização de listas encadeadas foi apropriada para este tipo de problema, pois permitiu a inserção e remoção de pacientes em diferentes posições da fila com eficiência.
- A implementação das funções de gerenciamento da fila, como *enqueue*, dequeue e *removePatient*, seguiu uma lógica simples e direta, facilitando a manipulação das prioridades dos pacientes.
- O uso de arquivos para ler as operações permitiu simular um cenário realista e forneceu flexibilidade para testar diferentes casos de entrada.

Dificuldades:

- O principal desafio foi implementar corretamente a inserção ordenada de pacientes na fila, especialmente considerando a necessidade de priorizar pacientes com pulseira vermelha sobre os amarelos e verdes. A lógica de inserção exigiu cuidado para garantir que o programa mantivesse a fila ordenada de acordo com a prioridade e a ordem de chegada.
- Outro desafio foi garantir que a remoção de pacientes (seja por atendimento ou desistência) ajustasse corretamente os ponteiros da lista encadeada, evitando erros como a perda de nós ou referências incorretas.

Compilação e Execução:

O programa foi desenvolvido e testado no *Visual Studio*. Para compilar e executar corretamente, o arquivo de entrada arq.txt deve ser colocado na pasta *Resource Files* do projeto.

No Visual Studio:

- Adicione o arquivo .c ao projeto.
- Coloque o arquivo arq.txt em *Resource Files*.
- Compile e execute o programa diretamente, sem a necessidade de especificar o caminho completo para o arquivo arq.txt, pois ele será acessado automaticamente no diretório de recursos.

Resultados dos Testes:

O programa foi testado com diferentes arquivos de entrada simulando 8 cenários, incluindo inserção, atendimento e desistência de pacientes, como demonstrado nos exemplos acima.

O programa funcionou corretamente na inserção, remoção e exibição dos pacientes. Não foram identificados erros durante os testes, e o comportamento do programa foi consistente com o enunciado da tarefa.

Por fim, ao longo dos testes, foi notado que o programa não incluía tratamento para possíveis entradas inválidas no arquivo (como números fora do intervalo esperado ou cores inválidas). Para incluir esse tratamento de erro, a função *manageQueue* foi alterada em três partes, conforme mostrado a seguir.

- Antes:

```
if (action == 'E') {
    Patient* p = createPatient(num, color);
    enqueue(q, p);
    if (color == 'R') {
        printf("Patient %d with RED (high) priority joined the hospital queue.\n\n", num);
    }
    else if (color == 'Y') {
        printf("Patient %d with YELLOW (medium) priority joined the hospital queue.\n\n", num);
    }
    else printf("Patient %d with GREEN (low) priority joined the hospital queue.\n\n", num);
}
```

- Depois:

```
if (action == 'E') {
    Patient* p = createPatient(num, color);
    enqueue(q, p);
    if (color == 'R') {
        printf("Patient %d with RED (high) priority joined the hospital queue.\n\n", num);
    }
    else if (color == 'Y') {
        printf("Patient %d with YELLOW (medium) priority joined the hospital queue.\n\n", num);
    }
    else if (color == 'G') printf("Patient %d with GREEN (low) priority joined the hospital queue.\n\n", num);
    else printf("Invalid priority. %c is neither RED (high), YELLOW (medium) or GREEN (low) priority", color);
}
```

- Antes:

```
else if (action == 'S') {
    Patient* p = dequeue(q);
    if (p != NULL) {
        if (color == 'R') {
            printf("Patient %d with RED (high) priority was treated (removed from the queue).\n\n", num);
        }
        else if (color == 'Y') {
            printf("Patient %d with YELLOW (medium) priority was treated (removed from the queue).\n\n", num);
        }
        else printf("Patient %d with GREEN (low) priority was treated (removed from the queue).\n\n", num);
        free(p);
    }
}
```

- Depois:

```
else if (action == 'S') {
    Patient* p = dequeue(q);
    if (p != NULL) {
        if (color == 'R') {
            printf("Patient %d with RED (high) priority was treated (removed from the queue).\n\n", num);
        }
        else if (color == 'Y') {
            printf("Patient %d with YELLOW (medium) priority was treated (removed from the queue).\n\n", num);
        }
        else if (color == 'G') printf("Patient %d with GREEN (low) priority was treated (removed from the queue).\n\n", num);
        else printf("Invalid priority. %c is neither RED (high), YELLOW (medium) or GREEN (low) priority", color);
        free(p);
    }
}
```

- Antes:

```
else {
    removePatient(q, num, color);
}
```

- Depois:

```
else if (action == 'D') {
    removePatient(q, num, color);
}
else printf("Invalid action: %c is neither an Entrance, an Exit or a Desistance.\n\n", action);
```

Conclusão:

O uso de listas encadeadas e structs foi uma escolha acertada para o gerenciamento dinâmico de filas hospitalares com diferentes níveis de prioridade. As listas encadeadas permitiram uma inserção e remoção ágil de pacientes, mantendo a ordem de prioridade sem a necessidade de deslocar elementos na memória, como seria necessário em outras estruturas, como vetores.

Além disso, as structs facilitaram a organização das informações dos pacientes, permitindo que cada um deles fosse representado de forma clara e eficiente. Com isso, o programa alcançou o objetivo de manipular os pacientes conforme as prioridades estabelecidas, mantendo bom desempenho tanto no tempo de execução quanto no uso de memória.

Segue o funcionamento geral do programa com um conjunto maior de operações, disponível em “sugestao_arq.txt”. Ele foi projetado para lidar com entradas mais variadas e complexas, garantindo o gerenciamento correto da fila, além de realizar o tratamento de erros citados.

```
Patient 1 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 1 - Priority GREEN (low)
Red: 0, Yellow: 0, Green: 1

Patient 2 with RED (high) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 1 - Priority GREEN (low)
Red: 1, Yellow: 0, Green: 1

Patient 3 with YELLOW (medium) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 1

Patient 4 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 2 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 2

Invalid priority. X is neither RED (high), YELLOW (medium) or GREEN (low) priority
Patient list:
Patient 2 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 2

Patient 6 with RED (high) priority joined the hospital queue.
```

```
Patient list:
Patient 2 - Priority RED (high)
Patient 6 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 2, Yellow: 1, Green: 2

Patient 2 with RED (high) priority was treated (removed from the queue).


Patient list:
Patient 6 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 2

Patient 7 with RED (high) priority joined the hospital queue.


Patient list:
Patient 6 - Priority RED (high)
Patient 7 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Red: 2, Yellow: 1, Green: 2

Patient 8 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 6 - Priority RED (high)
Patient 7 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Patient 8 - Priority GREEN (low)
Red: 2, Yellow: 1, Green: 3

Patient 9 with YELLOW (medium) priority joined the hospital queue.


Patient list:
Patient 6 - Priority RED (high)
Patient 7 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 9 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Patient 8 - Priority GREEN (low)
Red: 2, Yellow: 2, Green: 3

Patient 1 with GREEN (low) priority was treated (removed from the queue).
```

```
Patient list:
Patient 7 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 9 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Patient 8 - Priority GREEN (low)
Red: 1, Yellow: 2, Green: 3

Patient 10 with RED (high) priority joined the hospital queue.


Patient list:
Patient 7 - Priority RED (high)
Patient 10 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 9 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Patient 8 - Priority GREEN (low)
Red: 2, Yellow: 2, Green: 3

Patient 11 with GREEN (low) priority joined the hospital queue.


Patient list:
Patient 7 - Priority RED (high)
Patient 10 - Priority RED (high)
Patient 3 - Priority YELLOW (medium)
Patient 9 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Patient 8 - Priority GREEN (low)
Patient 11 - Priority GREEN (low)
Red: 2, Yellow: 2, Green: 4

Patient 3 with YELLOW (medium) priority left the hospital queue.


Patient list:
Patient 7 - Priority RED (high)
Patient 10 - Priority RED (high)
Patient 9 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Patient 8 - Priority GREEN (low)
Patient 11 - Priority GREEN (low)
Red: 2, Yellow: 1, Green: 4

Patient 4 with GREEN (low) priority was treated (removed from the queue).
```

```
Patient list:
Patient 10 - Priority RED (high)
Patient 9 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Patient 8 - Priority GREEN (low)
Patient 11 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 4

Patient 8 with GREEN (low) priority left the hospital queue.


Patient list:
Patient 10 - Priority RED (high)
Patient 9 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Patient 11 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 3

Invalid action: A is neither an Entrance, an Exit or a Desistance.


Patient list:
Patient 10 - Priority RED (high)
Patient 9 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Patient 11 - Priority GREEN (low)
Red: 1, Yellow: 1, Green: 3

Patient 8 with RED (high) priority was treated (removed from the queue).


Patient list:
Patient 9 - Priority YELLOW (medium)
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Patient 11 - Priority GREEN (low)
Red: 0, Yellow: 1, Green: 3

Patient 4 with GREEN (low) priority was treated (removed from the queue).


Patient list:
Patient 1 - Priority GREEN (low)
Patient 4 - Priority GREEN (low)
Patient 11 - Priority GREEN (low)
Red: 0, Yellow: 0, Green: 3

Patient 6 with RED (high) priority was treated (removed from the queue).
```

```
Patient list:
Patient 4 - Priority GREEN (low)
Patient 11 - Priority GREEN (low)
Red: 0, Yellow: 0, Green: 2

Patient 17 with RED (high) priority joined the hospital queue.


Patient list:
Patient 17 - Priority RED (high)
Patient 4 - Priority GREEN (low)
Patient 11 - Priority GREEN (low)
Red: 1, Yellow: 0, Green: 2
```