INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DE MINAS GERAIS – CAMPUS SÃO JOÃO EVANGELISTA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

EDUARDA LUIZA MARTINS DE OLIVEIRA

**TRABALHO PRÁTICO III**

SÃO JOÃO EVANGELISTA

2022

EDUARDA LUIZA MARTINS DE OLIVEIRA

**UNIDADE DE PRONTO ATENDIMENTO DE UM HOSPITAL**

SÃO JOÃO EVANGELISTA

2022

**SUMÁRIO..............................................................................................................**

|  |  |
| --- | --- |
| **1.INTRODUÇÃO** | 4 |
| 1.1. Objetivo Geral | 4 |
| 1.2. Objetivos Específicos | 4 |
| 1.3. Justificativa | 5 |
| **2. DESENVOLVIMENTO** | 7 |
| 2.1. Fila Dinâmica | 7 |
| 2.2. Tipos Abstratos de Dados (TADs) | 8 |
| 2.3. Ponteiro | 9 |
| 2.4. Implementação | 9 |
| **3. CONCLUSÃO** | 13 |
| **4.REFERÊNCIAS** | 14 |
| **5. APÊNDICES** | 15 |
| 5.1. Apêndice A – TADs (\*hpp) | 15 |
| 5.2. Apêndice B – TADs (\*cpp) | 17 |

# **1.INTRODUÇÃO**

O trabalho desenvolvido aqui é um sistema para uma Unidade de Pronto Atendimento (UPA) de um Hospital que irá atender pacientes de acordo com a sua triagem. Neste trabalho simulamos os procedimentos de classificação de risco para definir as prioridades nos atendimentos a pacientes (fila de espera). A prioridade é definida pelas cores vermelho, laranja, amarelo, verde e azul. Importante destacar que a linguagem utilizada será C++ e o objetivo é fixar conceitos sobre estruturas de dados, utilizar a Fila Dinâmica, Tipos Abstratos de Dados (TAD) e usar o raciocínio lógico para resoluções de problemas.

Ao decorrer do trabalho nota-se a funcionalidade de cada função e a importância do mínimo domínio dessas matérias citadas acima para resoluções de problemas que muitas das vezes é difícil de não acontecer, porém com devida persistência é possível o desenvolvimento desse Trabalho Prático finalizado com êxito e com suas funcionalidades em perfeito funcionamento.

O problema que foi proposto da Unidade de Pronto Atendimento de um hospital, é um exemplo perfeito para utilização dessas matérias propostas nesse primeiro momento, é muito importante entender a funcionalidade de cada passo realizado nesse código, de extrema importância também entender conceitos básicos de como funciona um pronto atendimento de um hospital para aplicar isso dentro do desenvolvimento do código.

## **1.1. Objetivo Geral**

Este trabalho tem como objetivo geral a implementação de uma Unidade de Pronto Atendimento de um Hospital, com objetivo de trabalhar com inclusão de uma fila de espera, de acordo com as prioridades de urgências ou emergências de cada pacientes, para um controle do hospital.

## **1.2. Objetivos Específicos**

Esse trabalho tem como objetivos específicos:

* Definir prioridades nos atendimentos a pacientes (fila de espera).
* Incluir a prioridade dos pacientes pelas cores vermelho (emergência), laranja (muito urgente), amarelo (urgente), verde (pouco urgente) e azul (não urgente).
* Cada prioridade pelas cores existe um tempo em horas e minutos para atender os pacientes.
* Utilização de TADs para controle da Fila Dinâmica.
* Apresentar a manipulação da disponibilidade dos médicos, tendo em vista que existe 5 médicos cadastrados, onde irá ter um tempo para atender os pacientes devido cada caso de emergência ou urgência. Assim que terminarem de atender um paciente irá atender os outros que estão na fila de espera.
* Apresentar também a funcionalidade de um hospital.
* Exibir um novo atendimento.
* Exibir a fila de espera.
* Exibir relatórios.
* Atualizar os atendimentos.
* Criar uma estrutura de fila.
* Inserir um elemento no fim.
* Retirar o elemento do início.
* Verificar se a fila está vazia.
* Liberar fila.

## **1.3. Justificativa**

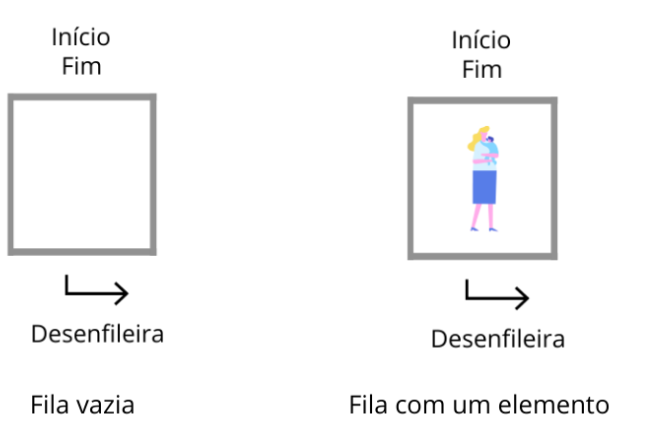
Esse trabalho é justificado devido a necessidade de uma Unidade de Pronto Atendimento de um hospital, onde podemos definir as prioridades nos atendimentos dos pacientes de acordo com as cores (vermelho, laranja, amarelo, verde e azul). Então com a implementação desse código facilitaria então esse controle, deixando tudo então da maneira mais simples para verificar o nível de prioridade de cada paciente, gerando também a organização do hospital, contendo também o controle de cada tempo que o médico irá atender os pacientes.

# **2. DESENVOLVIMENTO**

A seguir será apresentado o que foi utilizado para desenvolvimento desse trabalho, como Fila Dinâmica, TADS e também funções para que o trabalho ficasse funcional e usual na Unidade de Pronto Atendimento de um Hospital.

## **2.1. Fila Dinâmica**

Figura 1



A partir daí, todos os pacientes que chegarem à fila serão posicionados ao final dela.

Figura 2

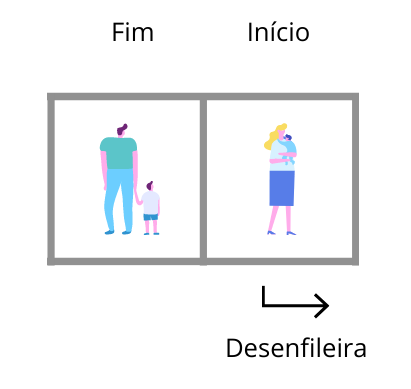
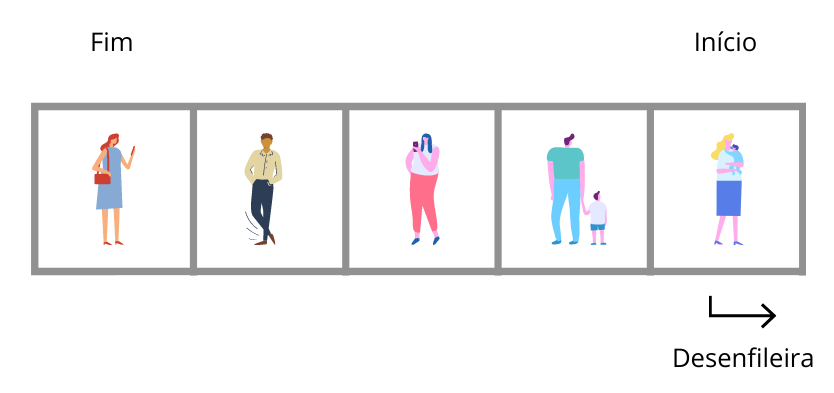
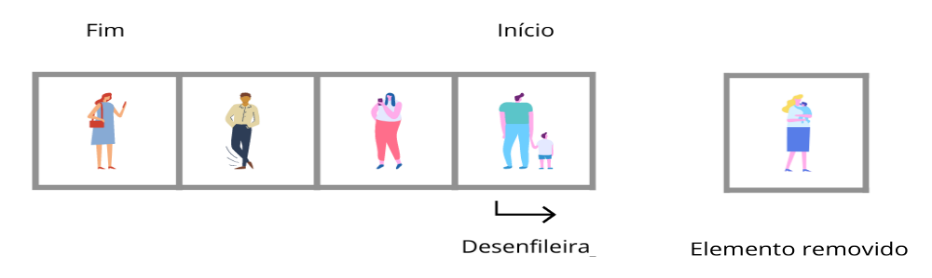


Figura 3



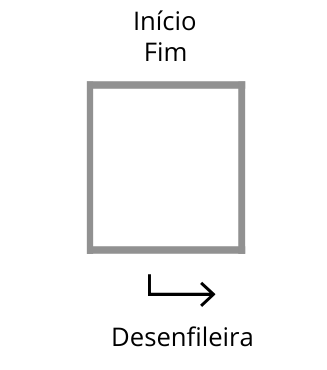
Sempre que o paciente for chamado para ser atendido, este é removido da fila e o paciente que estava na próxima posição passará a ser o início da fila.

Figura 4



Este processo se repete até que a fila esteja vazia e nenhum usuário tenha que ser atendido.

Figura 5



As figuras acima apresentam como funciona uma Fila Dinâmica, estrutura foi muito utilizada no desenvolvimento desse trabalho. Como apresentado acima podemos ter uma noção de como funciona a estrutura de dados da fila dinâmica, para iniciar essa explicação temos que ter em mente que toda fila é uma estrutura do tipo FIFO (First In First Out), ou seja, o primeiro elemento inserido será o primeiro a ser removido. Cada elemento da estrutura pode armazenar um ou vários dados e um ponteiro para o próximo elemento, o que permite o encadeamento e mantêm uma estrutura linear. A estrutura do tipo fila possui um ponteiro denominado INICIO, onde todas operações de remoção acontecem e outro denominado FIM, onde acontecem as inserções.

## **2.2. Tipos Abstratos de Dados (TADs)**

As TADs são modelos matemáticos de estruturas de dados que definem o tipo de dados a ser armazenado, as operações possíveis sobre estes dados e o tipo de dados das operações. A TAD tem o papel de definir o que se pode fazer com uma estrutura de dados, que é o tipo de informação ou conjunto das mesmas que pode ser armazenada em uma variável, tem também as funções de acesso dos dados e quais os parâmetros para estas funções e quais os retornos estas funções geram quando se acessam os dados.

## **2.3. Ponteiro**

Ponteiro também conhecido como apontador é uma variável que armazena o endereço de memória, e o interessante que ele pode apontar para um endereço inválido, que contém um valor NULL, nesse Trabalho Prático foi muito utilizado esse conceito para implementação da fila dinâmica, a próxima posição do último sempre apontava para NULL, onde não tinha nem item ali, era realmente nada que ele apontava. Então sabemos que ponteiro é de extrema importância para realização de trabalhos utilizando a fila, pois é utilizado para percorrer a mesma, por isso ele é indispensável.

## **2.4. Implementação**

Para o desenvolvimento desse trabalho foi utilizado diversas funções, que estão referenciadas nos Apêndices A e B no fim do documento, a seguir irei listar e explicar de maneira detalhada como elas funcionam:

* void inicializaFila(TFila \*fila) – função padrão de uma TAD para inicializar a estrutura de fila.
* bool verificaFilaVazia(TFila \*fila) – função padrão de uma TAD para verificar se a fila só foi inicializada e não foi inserido nada nela.
* void enfileira(TFila \*fila, TAtendimento item) – função padrão de uma TAD alterada para o trabalho, que insere de acordo com a prioridade. Nela temos dois apontadores criados localmente, para guardar valores, o atual para percorrer a fila, enquanto o anterior guarda o valor anterior ao que será inserido.
* void desenfileira(TFila \*fila, TAtendimento \*item) – função padrão de uma TAD, que remove do início da fila.
* void esvaziaFila(TFila \*fila) – função usada para executar, a limpeza total da fila, onde o desenfileira é executado até que a fila esteja vazia
* void imprimeFila(TFila \*fila) – função padrão de uma TAD, onde é imprimido os itens da fila
* void mostraUrgencia(TPaciente paciente) – função do sistema usada para mostrar a urgência do caso de cada paciente por meio de uma string, de acordo com o valor da prioridade do paciente;
* void cadastraMedicos(Hospital \*hospital) – função executada no inicio do código para cadastrar os médicos de forma automática, por meio da próxima função.
* void cadastraMedico(Hospital \*hospital, string nome, string especialidade, string crm) – função que insere os médicos no hospital de fato, usando os valores dos parâmetros para atribui-los aos valores da struct médico.
* void novoAtendimento(Hospital \*hospital) – nesta função, criamos uma atendimento atribuindo colocando um paciente nele, e mais abaixo adiciona o médico por meio da atualiza. Para isso foi criada uma lista por meio da biblioteca vector, temos a variavel novo\_paciente, recebendo o retorno da próxima função que será apresentada, logo em seguida esse mesmo retorno é mandado para a triagem, e ele é colocado na lista vector, depois é mostrado o nível de urgencia visto em uma das funções anteriores, e depois a previsão de atendimento com base em uma das próximas funções. Depois é perguntado se vai querer fazer um novo atendimento, caso sim, ele vai repetir o processo, e quando a resposta for não, ele vai entrar no for, percorrendo a lista de pacientes, e inserindo cada um deles na fila, e depois ele atualiza o que já foi feito antes, por meio de uma função apresentada logo a frente, isso se foi feito algo.
* TPaciente cadastraPaciente() – função de cadastro de paciente, que cria o paciente de forma local, insere os dados dele, e retorna ele, para ser usado como paramêtro em outras funções.
* void realizaTriagem(TPaciente \*paciente) – função para realizar a triagem, um conjunto de 18 perguntas, que inicialmente possui todas as respostas inseridas como N, depois temos a variavel urgencia\_encontrada, que serve para definir a urgencia com que o paciente deve ser atendido. O primeiro if, serve para indicar se a urgencia foi encontrada(lembrando que há os níveis de urgência), no segundo if, serve para caso a urgencia tenha sido encontrada, ele define o nível de urgência para cada caso, nas primeiras 5 perguntas, se houver pelo menos um S a urgência é vermelha, e ele já pula para o atendimento, se nas primeiras 10 se houver pelo menos um S a urgencia é laranja, ele já pula direto pro atendimento, se nas primeiras 15, se houver pelo menos um S a urgencia é Amarela, se dentro das 18, se houver pelo menos um S a urgencia é verde, e se a urgencia não foi encontrada, então a urgência é Azul.
* void mostraPrevisaoAtendimento(TPaciente paciente) – mostra a previsão de atendimento de acordo com a urgência encontrada na última função apresentada, essa função só vai retorna a mensagem falando qual o nível de urgência.
* void atualiza(Hospital \*hospital) – função que chama a finalizaAtendimentos(Hospital \*hospital, chrono::system\_clock::time\_point data\_atual) e bool iniciaAtendimentos(Hospital \*hospital), para que sejam finalizados os atendimentos, e iniciados os atendimentos que estão na fila de espera.
* bool finalizaAtendimentos(Hospital \*hospital, chrono::system\_clock::time\_point data\_atual) – a função cria uma variável booleana, para ser manipulada dentro do for, e dentro dele é criado uma variável para pegar o horário do sistema, por meio da biblioteca chrono, depois ele vai para um for começando em 0, e indo até a quantidade de atendimentos em execução no hospital. Uma variável atendimento será usada para recolher os atendimentos da lista de atendimentos. Na linha abaixo, temos uma variável para pegar o horário de término previsto no momento em que o paciente foi inserido no atendimento, e convertido para o mesmo tipo de variável da variável usada para pegar o horário do sistema. Se esse horário já tiver sido excedido, significa que o paciente já foi atendido, logo o médico já está disponível para um novo atendimento, e logo em seguida, é retornado a mensagem de finalização do atendimento.
* bool iniciaAtendimentos(Hospital \*hospital) – na função temos uma variável boolena, para ser manipulada no while, e dentro dele temos um for, para percorrer os médicos e achar 1 que esteja disponível, para atender os pacientes que estão na lista de espera, estes que serão removidos de lá, e inseridos em um atendimento. Depois é pego o horário do sistema, que será usado como data de início do atendimento, e para calcular o tempo de atendimento junto com a variação, de acordo com a prioridade do paciente, e assim calcular a data de término do atendimento. E assim insere o atendimento de forma definitiva na lista de atendimentos, e retorna a mensagem de confirmação.
* void exibeAtendimentos(Hospital hospital) – a função exibe atendimento, imprime toda a lista de atendimentos que ainda estão sendo realizados, novamente usando o for para percorrer a lista de atendimentos.
* void exibeRelatorio(Hospital hospital) – a função para exibir o relatório, que mostra o que já foi feito durante o tempo de execução do programa, um relatório mesmo.
* void exibeFilaEspera(Hospital hospital) – a função para exibir a Fila de espera, que são aqueles pacientes que ainda estão esperando para serem atendidos, pois todos os médicos estão ocupados atendendo alguém.

# **3. CONCLUSÃO**

A unidade de pronto atendimento é muito importante para todo hospital, e a implementação desse código também foi de extrema importância para desenvolvimento do controle de um hospital, um controle bem amplo sobre a quantidade médicos para atender os pacientes que estão na fila de espera de acordo com cada prioridade. Com e a utilização de códigos e tecnologia foi algo que realmente se deu muito bem com as soluções desse problema proposto, ou seja, serviu para cronometrar o tempo que cada médico gasta para atender os pacientes de acordo com cada prioridade através das cores.

Ao decorrer do desenvolvimento desse projeto confirmamos que o domínio dessas funcionalidades é realmente muito importante, tendo em vista que ouve muitos erros durante o desenvolvimento do projeto, e como estávamos utilizando diversos conceitos dentro da programação, alguém sem o devido conhecimento muito provavelmente não conseguiria solucionar estes problemas.

Ao fim desse trabalho, podemos concluir que todos os objetivos que foi proposto no trabalho foi alcançado de forma que o código está rodando sem erros e de maneira funcional de acordo com o que foi proposto no início do Trabalho Prático. Objetivos como apresentar a funcionalidade de um hospital, exibir um novo atendimento, atualizar os atendimentos, inserir um elemento no fim, retirar elemento do início, verificar se a fila está vazia, cadastro de médico, triagem, cadastro de paciente, saída de dados e também o registro de hospital (UPA).

Ao longo do trabalho encontrei dificuldades para interpretar o que estava sendo pedido, pedi ajuda a colegas de classe para me auxiliar na interpretação e na escrita do código do trabalho, utilizei os slides disponibilizados pelo professor para sanar dúvidas a respeito das TADs (Tipos Abstratos de Dados). Foi utilizado as informações da biblioteca, sites, e ajuda de colegas de sala para ajudar na conclusão do Trabalho Prático. Acho que consegui de forma sucinta apresentar conhecimentos em uma fila dinâmica. Estou feliz com o resultado, e ele atendeu às minhas expectativas.

# **4.REFERÊNCIAS**

TRINDADE. Eduardo. Algoritmos e Estrutura de Dados – Fila com Ponteiro. 2022. Apresentação PDF. Disponível em: <https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/149127/mod_resource/content/1/Aula%2011%20-%20Fila%20com%20Ponteiro.pdf>

TRINDADE. Eduardo. Algoritmos e Estrutura de Dados – Fila com Arranjo. 2022. Apresentação PDF. Disponível em:

<https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/147825/mod_resource/content/1/Aula%209%20-%20Fila.pdf>

TRINDADE. Eduardo. Algoritmos e Estrutura de Dados – Data, hora e tempo em C++. 2022. Apresentação PDF. Disponível em:

<https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/149625/mod_resource/content/1/Aula%2013%20-%20Data%2C%20hora%20e%20tempo%20em%20C%2B%2B.pdf>

TRINDADE. Eduardo. Algoritmos e Estrutura de Dados – Classe Vector. 2022. Apresentação PDF. Disponível em:

<https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/149748/mod_resource/content/2/Aula%2014%20-%20Classe%20Vector.pdf>

TRINDADE. Eduardo. Algoritmos e Estrutura de Dados – TAD Fila com Ponteiro. 2022. Apresentação PDF. Disponível em:

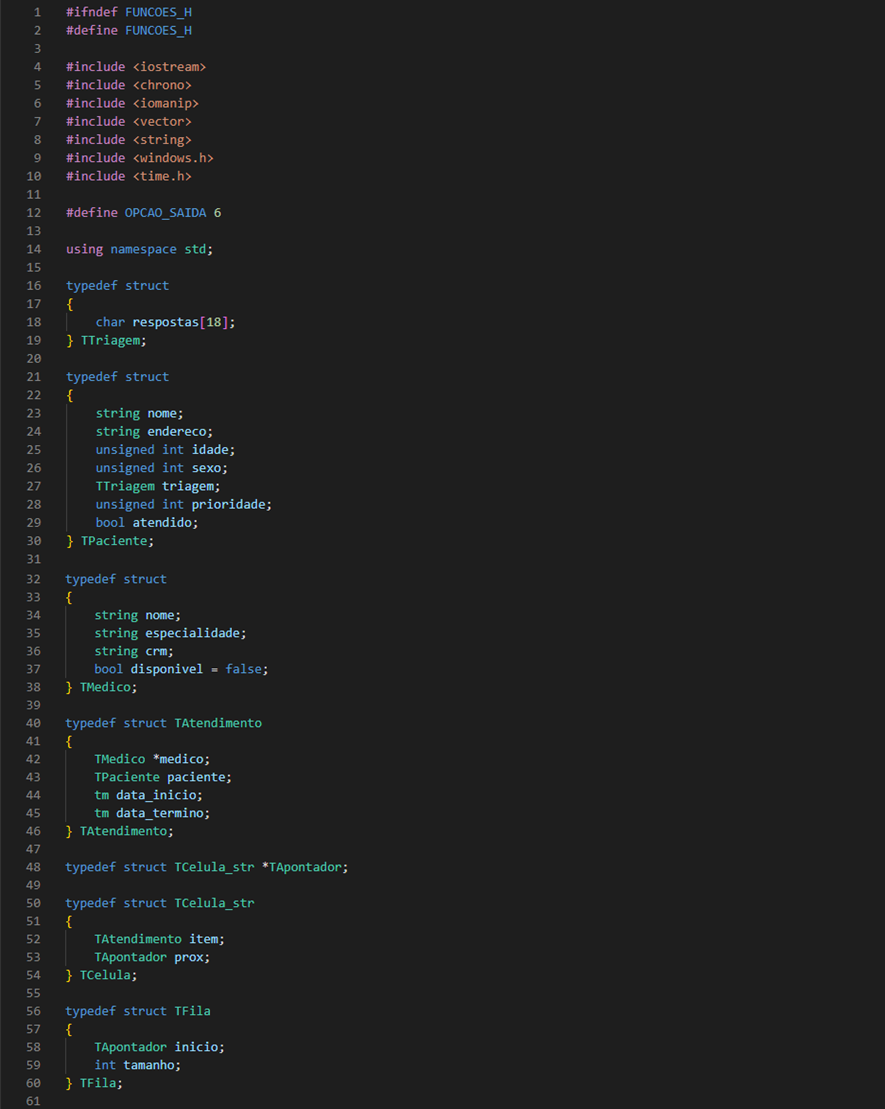
<https://github.com/edutrindade/Algoritmos-e-Estruturas-de-Dados-I/tree/master/10.%20Fila%20com%20Ponteiro/10.2%20TAD%20Fila%20com%20Prioridade>

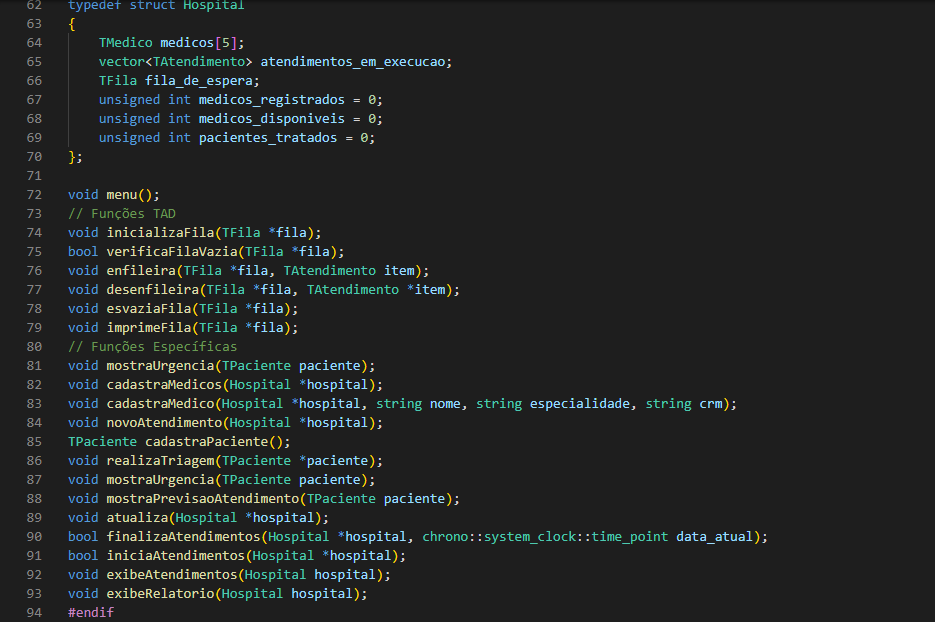
TRINDADE. Eduardo. Algoritmos e Estrutura de Dados – Bibliotecas de Tempo. 2022. Apresentação PDF. Disponível em:

<https://github.com/edutrindade/Algoritmos-e-Estruturas-de-Dados-I/tree/master/12.%20Bibliotecas%20de%20tempo>

# **5. APÊNDICES**

## **5.1. Apêndice A – TADs (\*hpp)**



****

## **5.2. Apêndice B – TADs (\*cpp)**

