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

**BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

GABRIEL KÁICON BATISTA HILÁRIO

**TRABALHO PRÁTICO I**

**SÃO JOÃO EVANGELISTA**

**SETEMBRO - 2022**

GABRIEL KÁICON BATISTA HILÁRIO

SISTEMA DE CADASTRO DE FUNCIONÁRIOS E DE PROJETOS DENTRO DE FUNCIONÁRIOS

**SÃO JOÃO EVANGELISTA**

**SETEMBRO - 2022**

SUMÁRIO

[1. INTRODUÇÃO 5](#_Toc115300956)

[1.1. Objetivo Geral 5](#_Toc115300957)

[1.2. Objetivos Específicos 5](#_Toc115300958)

[1.3. Justificativa 5](#_Toc115300959)

[2. DESENVOLVIMENTO 7](#_Toc115300960)

[2.1. Conceitos Aplicados 7](#_Toc115300961)

[2.1.1. Tipos Abstratos de Dados 7](#_Toc115300962)

[2.1.2. Lista Encadeada 8](#_Toc115300963)

[2.1.3. Lista Sequencial 9](#_Toc115300964)

[2.1.4. Ponteiros 9](#_Toc115300965)

[2.1.5. Arquivos 9](#_Toc115300966)

[2.2. Implementação 10](#_Toc115300967)

[3. CONCLUSÃO 12](#_Toc115300968)

[4. REFERÊNCIAS 14](#_Toc115300972)

[5. APÊNDICES 15](#_Toc115300973)

[5.1. APÊNDICE A – Funções das TADs no arquivo.cpp 15](#_Toc115300974)

[5.2. APÊNDICE B – Sistema.hpp 17](#_Toc115300975)

[5.3. APÊNDICE C – Manipulação de arquivos 18](#_Toc115300977)

[5.4. APÊNDICE D – Menu 19](#_Toc115300978)

[5.5. APÊNDICE E – Função de Inserção de funcionário 20](#_Toc115300980)

[5.6. APÊNDICE F – Função de retorno de funcionário, para inserção 20](#_Toc115300982)

[5.7. APÊNDICE G – Inclusão de Projeto 21](#_Toc115300984)

[5.8. APÊNDICE H – Pesquisa funcionário por ID 21](#_Toc115300986)

[5.9. APÊNDICE I – Exclui Projetos 22](#_Toc115300988)

[5.10. APÊNDICE J – Exclui funcionários sem Projeto 22](#_Toc115300990)

[5.11. APÊNDICE K – Impressão de Funcionário 23](#_Toc115300992)

[5.12. APÊNDICE L – Cálculo de Salário Bruto 23](#_Toc115300993)

# INTRODUÇÃO

Este trabalho prático foi documentado para que seja avaliado em conjunto com os códigos na linguagem C/C++, exigido pelo docente Eduardo Augusto da Costa Trindade, dentro da disciplina de Algoritmos e Estruturas de Dados I, ministrada pelo mesmo. Porém a documentação tem cunho expositivo, onde é descrito as funcionalidades do programa, com testes, e desenvolvimento de novas linhas de raciocínio lógico para realização do trabalho prático.

## Objetivo Geral

Este trabalho tem como objetivo geral apresentar na prática os conhecimentos adquiridos nas aulas de Algoritmos e Estruturas de Dados I, a respeito de Listas com Arranjo, Listas Encadeadas, Tipos Abstratos de Dados (TADs), Manipulação de Arquivos e Ponteiros, utilizando a linguagem de C++ para escrita dos códigos.

## Objetivos Específicos

Esse trabalho tem como objetivos específicos:

* Apresentar conhecimentos em lista encadeada;
* Apresentar conhecimentos em lista com arranjo;
* Apresentar conhecimentos em manipulação de arquivos;
* Aplicar conhecimentos adquiridos em um minissistema, em uma visão empresarial.

## Justificativa

Ao iniciar os estudos de Algoritmos e Estruturas de Dados I, vemos os seguintes conteúdos:

* Ponteiros
* TADs
* Manipulação de Arquivos
* Listas:
  + Lista com Arranjo
  + Lista Encadeada

Vemos em ponteiros, a manipulação de valores da variável por meio do endereço de memória, utilizando ponteiros. Vemos em arquivos, os comandos básicos de leitura e gravação de dados em um arquivo por meio de objetos da biblioteca *fstream*, o *ifstream* para leitura e o *ofstream* para gravação.

Posteriormente vemos um conteúdo mais amplo de listas, que consiste na inserção de itens em uma lista, uma estrutura sendo inserida dentro de outra, semelhante a POO (Programação Orientada a Objetos). Inicialmente são apresentados dois tipos de lista:

* A lista com arranjo, também chamada de lista sequencial que é com alocação estática, ou seja, possui um limite pré-definido, podendo ter um número limitado de itens.
* A lista encadeada ou lista com ponteiro que é a com alocação dinâmica, ou seja, não possui um limite pré-definido podendo ter um número infinito de itens.

A princípio ambas são aplicados de forma individual, porém no trabalho, é exigido que usemos as duas juntas, antes devemos cadastrar funcionários em uma lista encadeada, e devemos criar uma lista sequencial dentro de cada funcionário, para cadastrar projetos para esse funcionário que está na lista encadeada.

Os conteúdos foram aplicados para realização do trabalho, criando um minissistema que nos permite cadastrar e excluir funcionários, adicionar e retirar projetos ao funcionário, e calcular e imprimir o contracheque.

Tendo isso tudo em vista, o trabalho foi exigido para que seja possível desenvolver o raciocínio lógico quanto a aplicação dessas estruturas em conjunto.

# DESENVOLVIMENTO

Nesta seção do documento é apresentado, os conceitos aprendidos e o desenvolvimento do trabalho em si, na linguagem C++.

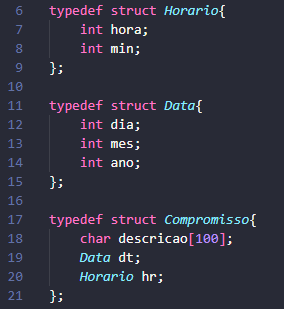
## Conceitos Aplicados

Explicação sucinta dos conceitos de Lista Encadeada, Lista sequencial, arquivos e Tipos Abstratos de Dados(TADs).

## Tipos Abstratos de Dados

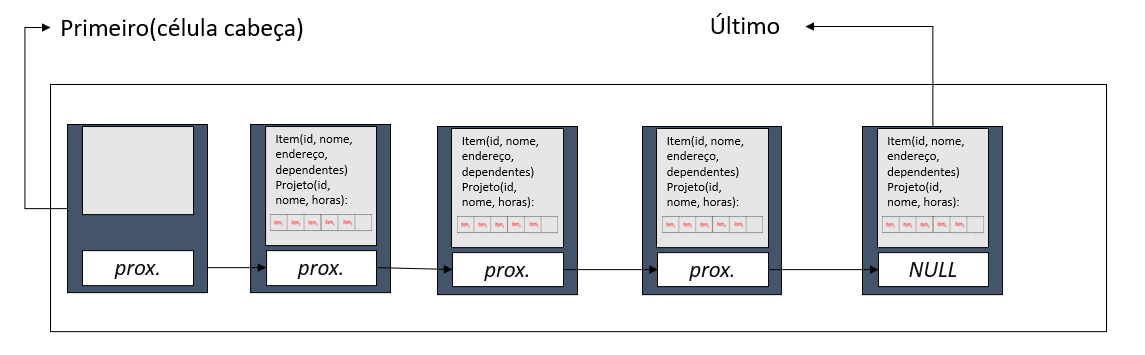
As estruturas utilizadas como registro para criação de objetos, e as funções de manipulação dessas estruturas, ambas compõem uma TAD. Declaramos esses modelos como *structs*, elas são representações de qualquer coisa no mundo real, sendo ela lógica, abstrata ou física, como por exemplo uma pessoa, que é algo físico, ou um filme digital, que é algo lógico/abstrato, veja o exemplo na figura 1. Cada um tem suas características específicas, uma pessoa nome, sexo, idade, CPF, altura, dentre outras, e um filme título, linguagem, elenco, personagens, duração, categoria, ano de lançamento, dentre outros, e tudo isso pode ser definida dentro de uma struct para cada um deles. Resumindo uma Struct é uma espécie de variável modelo para cadastrar diferentes itens, dentro de um software escrito em C/C++. Acompanhado das structs temos as funções para manipulação dos dados dessa lista, e desses itens, que será visto no próximo tópico.

Figura 1 – Struct exemplo, sem ligação com o trabalho



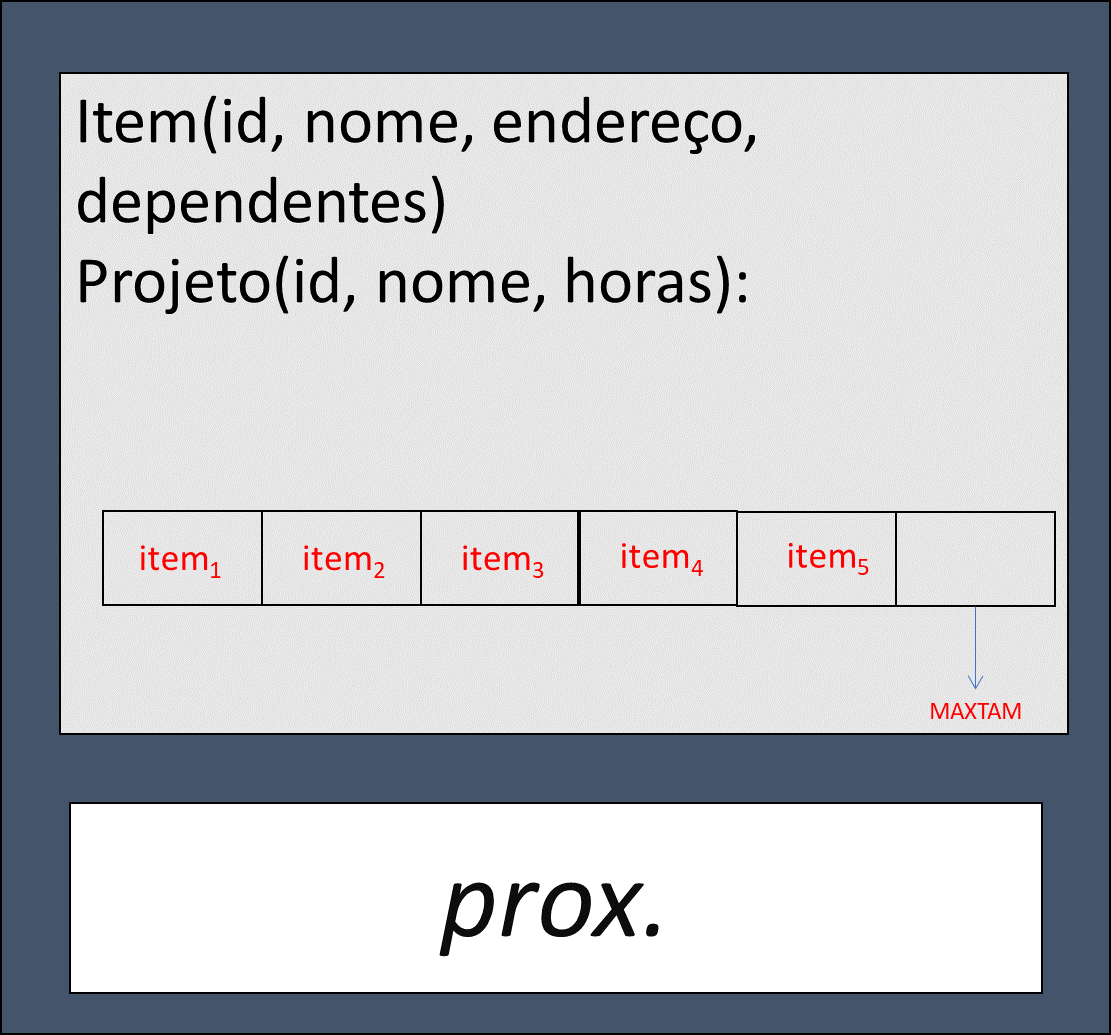
## Lista Encadeada

Figura 2 – Lista Encadeada com Sequencial



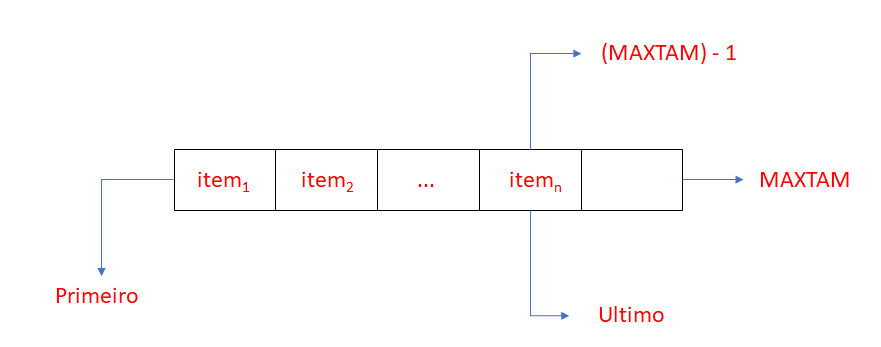
Na figura 2, vemos um desenho esquemático de como seria uma lista encadeada. Cada espaço (em cinza azulado) da lista, tem um apontador, que aponta para a célula seguinte, e uma célula (em cinza claro), essa célula tem o Funcionário (lá está como item) que é uma struct e o projeto, a nossa lista sequencial, que é uma struct, com isso temos, uma struct dentro de um campo de outra struct, comom vemos na figura 3. O primeiro, recebe a célula como NULL, pois ele deve apontar para o primeiro item, e o último, tem o apontador NULL, pois não há uma célula seguinte. Com isso, podemos inserir no inicio ou no fim da lista, e também após um item.

Figura 3 – Célula da Lista



## Lista Sequencial

Figura 4 – Lista Sequencial



Na figura 4, vemos um desenho esquemático de como seria uma lista sequencial, e o limite dela, é o MAXTAM, uma variável didática muito comum de ser usada, e sua função é delimitar o tamanho da lista. Ela é um vetor, porém a tipagem de dados a ser inserida podem ser as structs, podendo armazenar mais de um tipo de variável dentro de uma posição. Os elementos são inseridos dentro do índice do vetor no ultimo, que seria o apontador para ultima posição até o momento, e assim sucessivamente, até que a Lista alcance o tamanho máximo, que seria o valor definido para o MAXTAM.

## Ponteiros

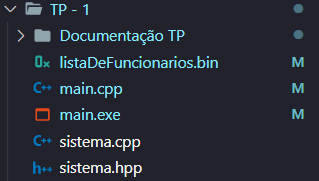
Ponteiros são variáveis especiais, que podem armazenar um endereço de memória, e manipular o conteúdo desses endereços de memória. Tem aplicações como manipulação de vetores, passagem de parâmetro por referência, desenvolvimento de estruturas de dados complexos. Resumindo, um tipo de variável que nos permite exibir uma struct, ou o conteúdo de um vetor ou variável, apenas pelo endereço de memória dele.

## Arquivos

A parte de manipulação de arquivos é mais abstrata, não é convencional para ilustrar ela, mas podemos usar uma analogia para compreender, imagine-se estudando para uma prova, onde se usa um livro e uma folha de papel em branco, o papel é seu arquivo, e o livro é seu programa, quando você quer guardar algo importante do livro, você lê o que está nele e escreve na folha, isso seria semelhante às funções do ofstream, que pega os dados digitados no programa e escreve no arquivo, independente da extensão deste. Imagine novamente, com um caderno e uma folha apenas, o caderno é o programa, e a folha é o arquivo com o que você leu dele, e não estava em branco quando começou tendo algo escrito na folha, você realiza a leitura e exibe isso no caderno, seria semelhante às funções do ifstream, que pega os dados do arquivo e exibe na tela.

## Implementação

Figura 5 – Organização



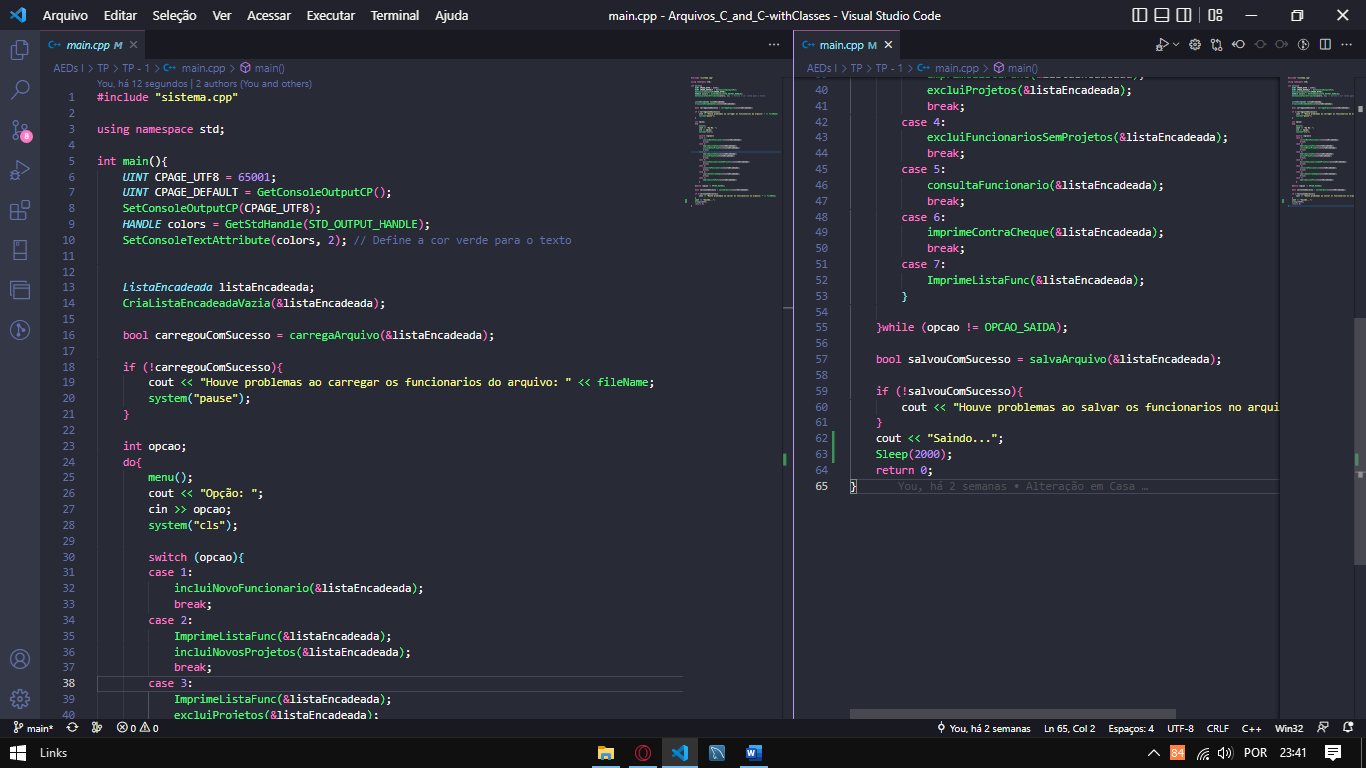
O trabalho foi dividido em 5 arquivos, incluindo a main, e o arquivo bin, como apresentado na figura 4. E o arquivo sistema.cpp com seu arquivo sistema.hpp, e o \*.exe. E a pasta de documentação, com os arquivos que serão enviados.

O sistema.cpp possui uma parte para as funções básicas para manipulação das TADs(Apêndice A) da Lista Encadeada e da Lista Sequencial.

* void menu(); - Menu que é exibido na tela(Apêndice D)
* void incluiNovoFuncionario(ListaEncadeada \*lista); - função para a inclusão de funcionário (Apêndice E)
* Funcionario \*criaFuncionario(); - função com cout e cin que retorna o funcionário, para ser inserido(Apêndice F)
* void incluiNovosProjetos(ListaEncadeada \*lista); - incluir novos projetos, é para perguntar, se quer incluir projetos, e quantos projetos quer incluir, para inserir de fato o projeto com o incluiNovoProjeto(Apêndice G)
* void incluiNovoProjeto(Funcionario \*funcionario); - insere o projeto ao funcionário (Apêndice G)
* void criaProjeto(ListaSequencial \*lista, Projeto \*projeto); - pega o projeto como parâmetro, e faz alterações nele dentro da função (Apêndice G)
* Funcionario \*FuncionarioPorID(ListaEncadeada \*lista); - função para pesquisar um funcionário por id dentro da função, e retornar, um funcionário, para manipulação de outras funções (Apêndice H)
* void excluiProjetos(ListaEncadeada \*lista); - pesquisa o funcionário na lista encadeada, usa das funções da TAD, para remover o projeto do funcionário (Apêndice I)
* void excluiFuncionariosSemProjetos(ListaEncadeada \*lista); - se existir funcionários sem projetos, ou seja com a Lista Sequencial com tamanho = 0, esse funcionário é excluído (Apêndice J)
* void ImprimeListaFunc(ListaEncadeada \*lista); - Imprime a lista de funcionários (Apêndice K)
* void consultaFuncionario(ListaEncadeada \*lista); - pesquisa um funcionário pelo id dele, e exibe na tela as informações dele (Apêndice K)
* int totalHorasSemanais(ListaSequencial lista); - calcula o somatório de horas semanais, de um funcionário (Apêndice L)
* double calculaSalarioBruto(Funcionario \*funcionario); - calcula o salário bruto de acordo com base nas horas semanais (Apêndice L)
* void imprimeContraCheque(ListaEncadeada \*lista); - calcula o salário líquido, com base nos descontos no salário bruto e imprime o contracheque (Apêndice L)
* // FUNÇÕES DE MANIPULAÇÃO DE ARQUIVOS
* bool carregaArquivo(ListaEncadeada \*lista); - ele pega o que foi cadastrado na struct, e insere na última posição da lista, no momento em que o programa é iniciado (Apêndice C)
* bool salvaArquivo(ListaEncadeada \*lista); - ele pega o que foi cadastrado durante a execução e salva no arquivo bin, e deleta ele da memória (Apêndice C)

Após todas as funções, chamo as funções de carregamento e salvamento do arquivo, e as funções exigidas pelo trabalho, como na figura 6.

Figura 6 – Main



# CONCLUSÃO

Ao longo do trabalho encontrei dificuldades para interpretar o que estava sendo pedido, fui a monitoria para sanar minhas dúvidas, conversei com poucos colegas de classe, para me auxiliarem na interpretação e na escrita do código do trabalho, fiz uso dos slides disponibilizados pelo professor para sanar dúvidas a respeito das TADs (Tipos Abstratos de Dados), criação de funções de manipulação de arquivos, e também utilizei as informações da biblioteca, sites e fóruns.

Coloquei o trabalho em modularização separando-o em 5 arquivos, 2 arquivos \*.cpp, o sistema.cpp, com as funções e a main.cpp com a execução das funções, 1 arquivo \*.hpp, o sistema.hpp, onde tinha as structs e o cabeçalho das funções, 1 arquivo \*.bin, listaDeFuncionarios.bin onde os funcionários eram salvos, e 1 arquivo \*.exe, a main.exe, que seria o executável do código, onde as funções compilavam.

Creio eu que meu desenvolvimento apesar das dificuldades de interpretação, foram ótimos, aprendi coisas novas como funções que retornam um item, manipular arquivos, usar duas Estruturas de Dados para compor 1 só. Achei bastante semelhante ao que fiz no Ensino Médio Técnico Integrado, onde fizemos um sistema para atender à uma pequena empresa. Acho que consegui atingir meus objetivos, e explicar bem o que cada função faz, apresentar conhecimentos em lista encadeada, lista sequencial, manipulação de arquivos, e aplicar isso em um software, que atendesse o mínimo que uma empresa precisaria. Estou feliz com o resultado, e ele atendeu às minhas expectativas além do que eu esperava.



# REFERÊNCIAS

COSTA TRINDADE. Eduardo Augusto. Algoritmos e Estrutura de Dados - Arquivos. 2022. Apresentação PDF. Disponível em: [https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/146487/mod\_resource/content/1/Aula%203%20-%20Arquivos.pdf](https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/146487/mod_resource/content/1/Aula%203%20-%20Arquivos.pdfv) . Acesso em: 25 de setembro de 2022.

COSTA TRINDADE. Eduardo Augusto. Algoritmos e Estrutura de Dados - Listas. 2022. Apresentação PDF. Disponível em: <https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/146499/mod_resource/content/1/Aula%206%20-%20Listas.pdf> . Acesso em: 24 de setembro de 2022.

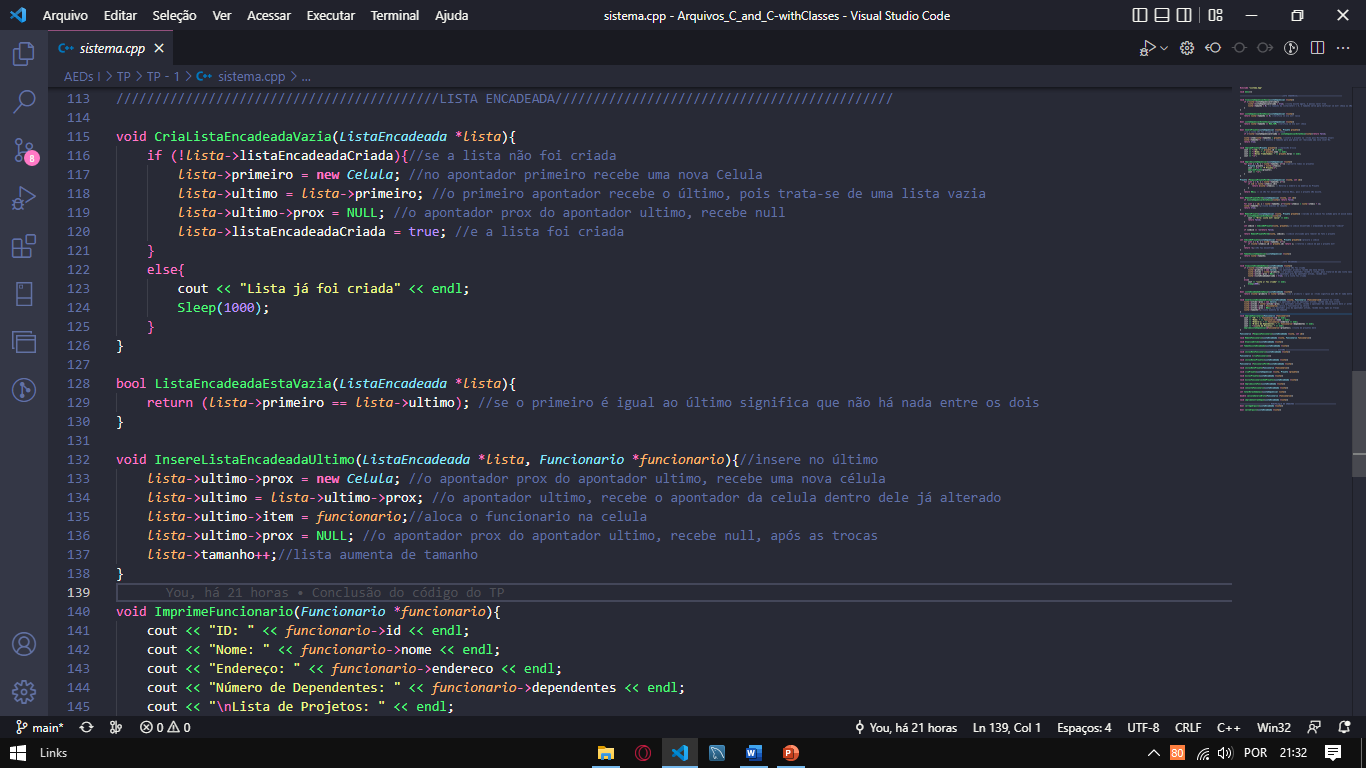
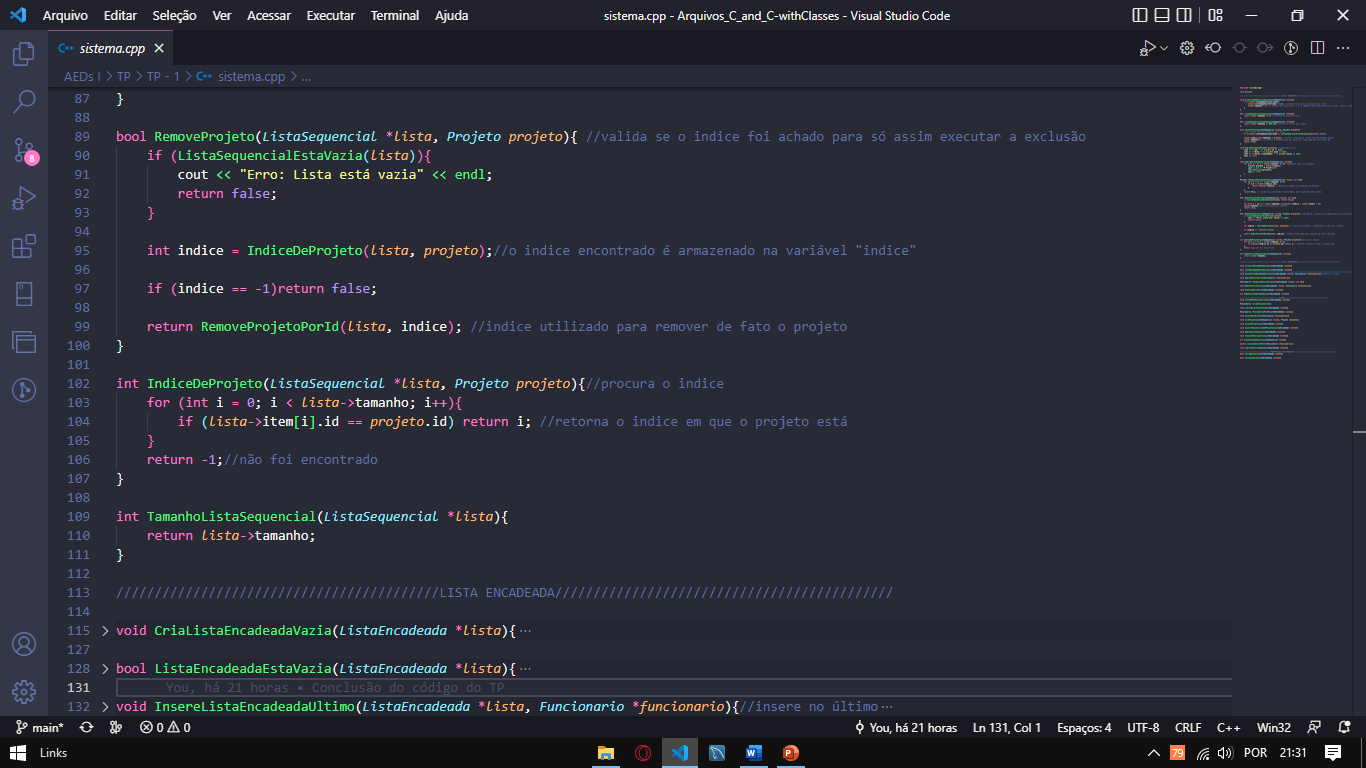
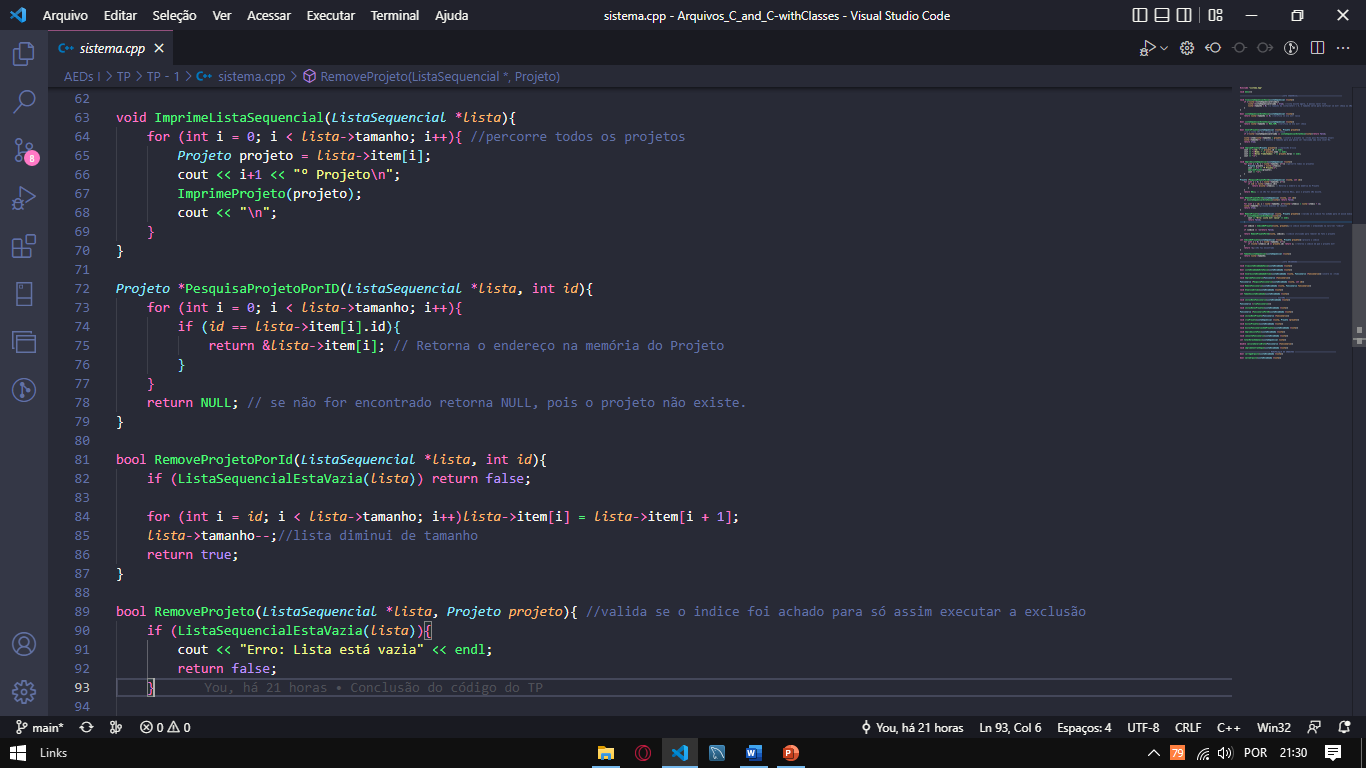
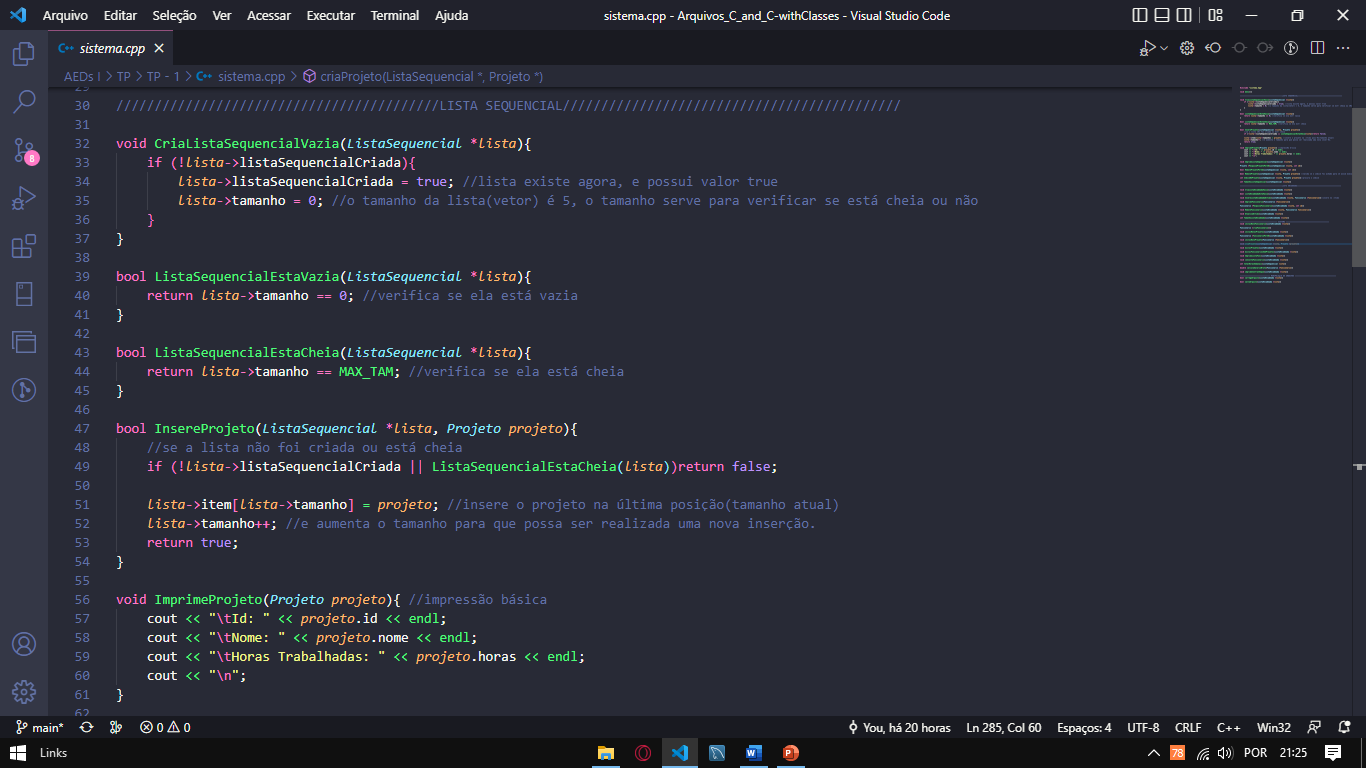
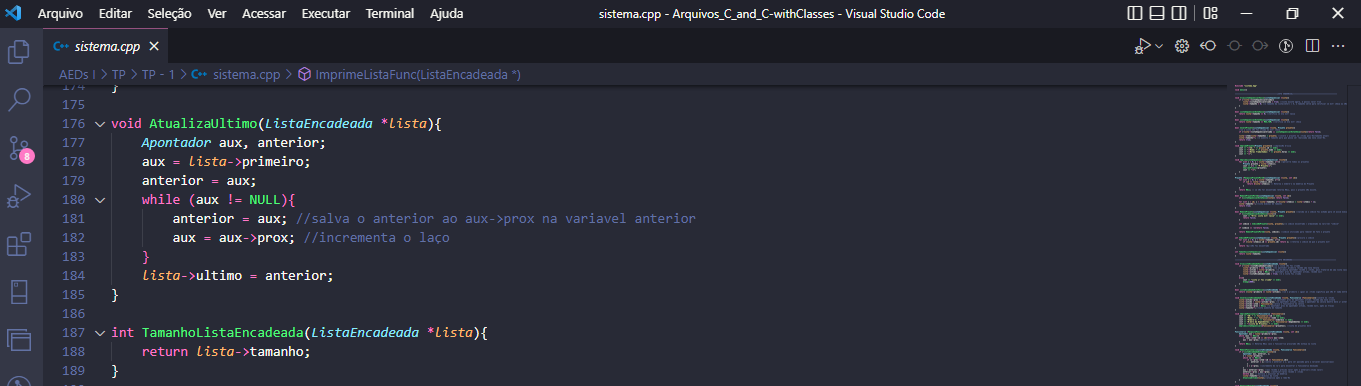
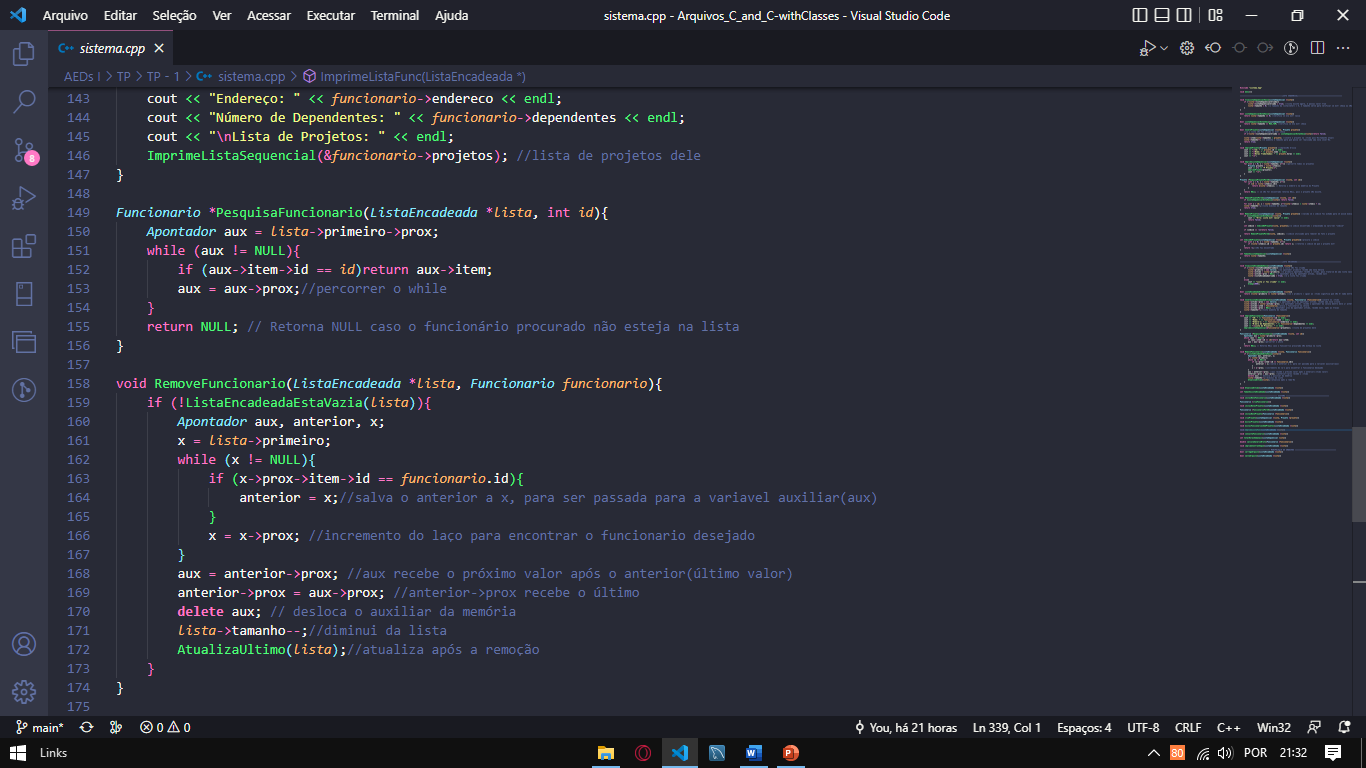
COSTA TRINDADE. Eduardo Augusto. Algoritmos e Estrutura de Dados - Listas utilizando Ponteiro. 2022. Apresentação PDF. Disponível em: <https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/146500/mod_resource/content/1/Aula%207%20-%20Listas%20Encadeadas.pdf> . Acesso em: 26 de setembro de 2022.

COSTA TRINDADE. Eduardo Augusto. Algoritmos e Estrutura de Dados – Ponteiros. 2022. Apresentação PDF. Disponível em: . Acesso em : 28 de setembro de 2022.

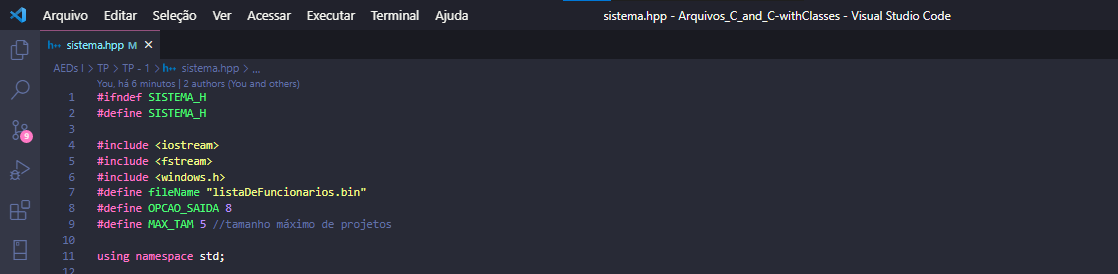
Programar em C++/Entrada e saída de dados 2. **WIKI LIVROS**, 16/04/2020.Disponível em: <https://pt.wikibooks.org/wiki/Programar_em_C%2B%2B/Entrada_e_saída_de_dados_2#:~:text=biblioteca%20padrão%20fstreamEditar&text=Esta%20biblioteca%20define%203%20novos,"in%20from%20a%20file"%20>. Acesso em: 25 de setembro de 2022.

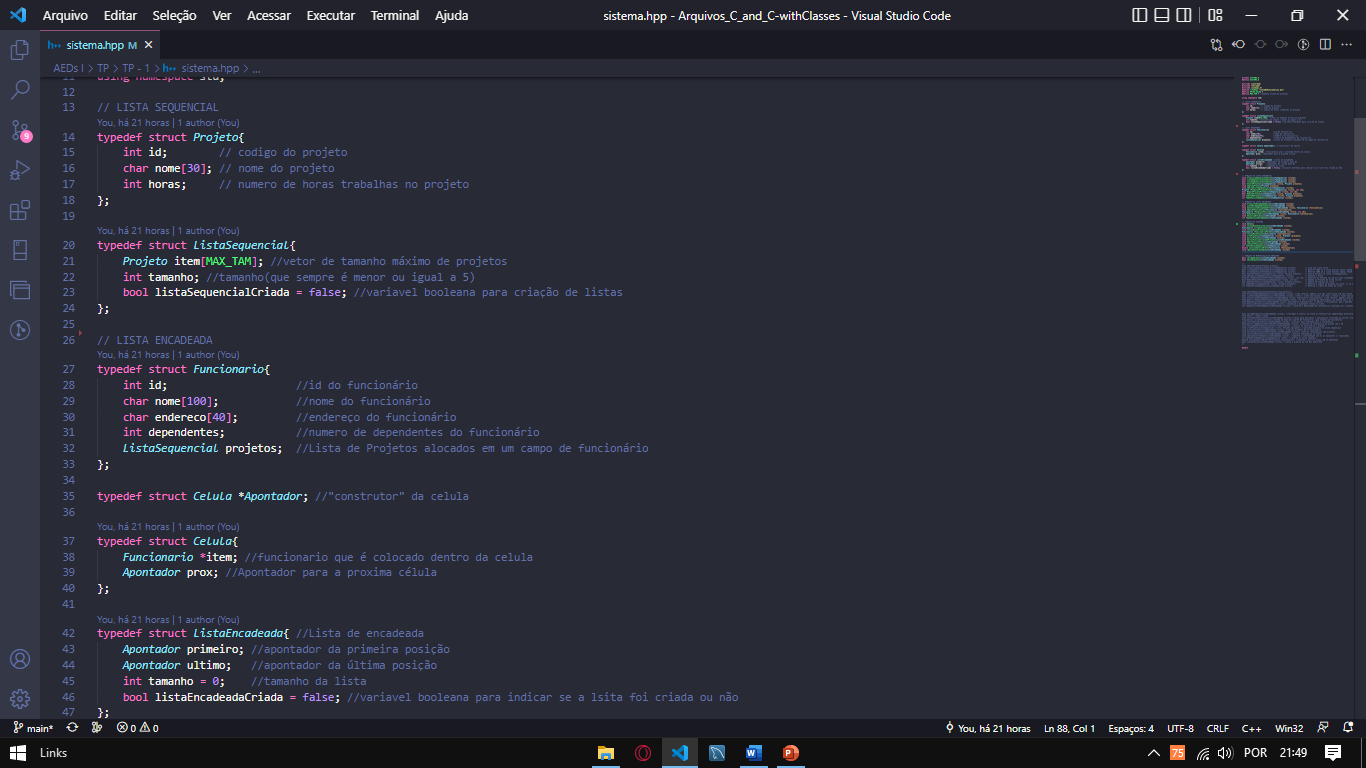
# APÊNDICES

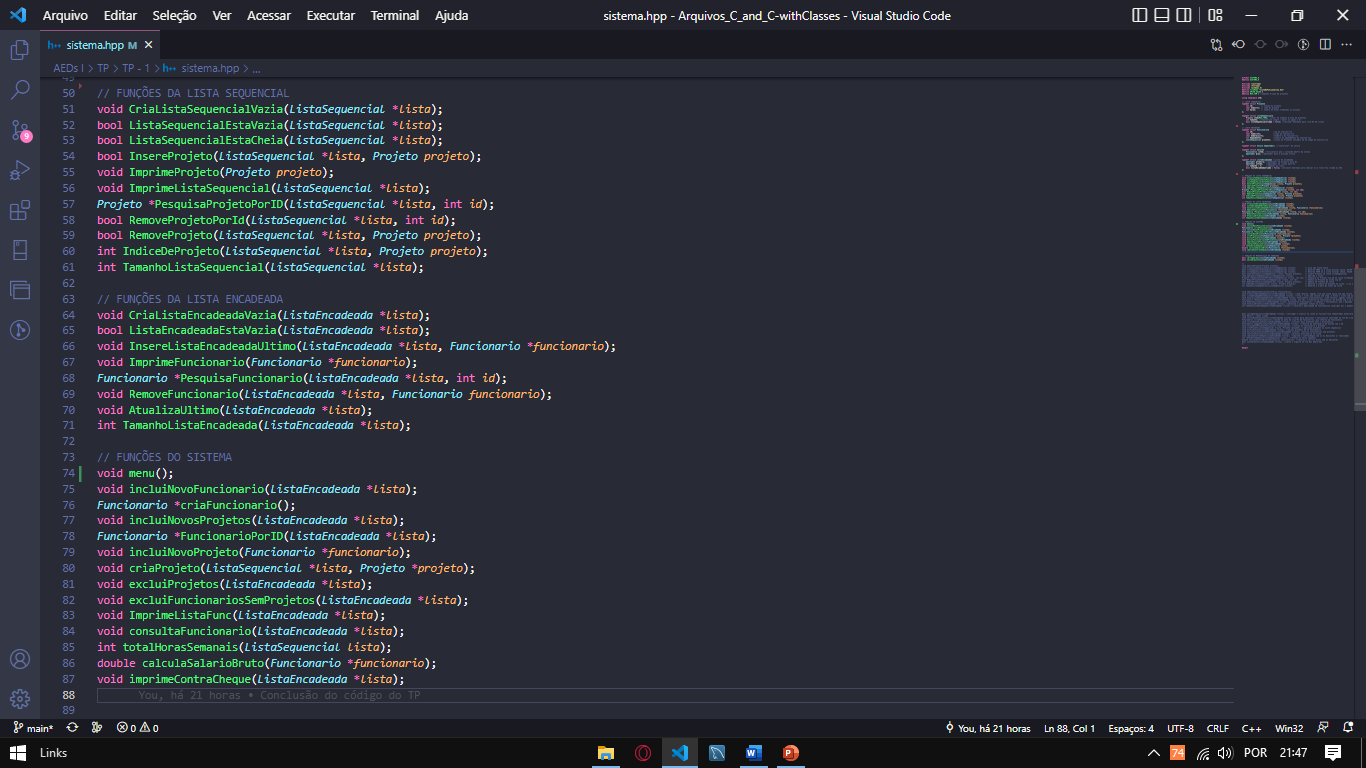
* 1. APÊNDICE A – Funções das TADs no arquivo.cpp

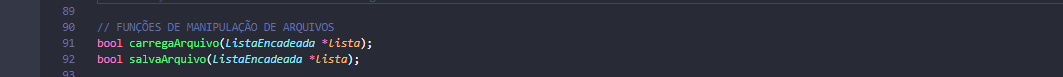
  


* 1. APÊNDICE B – Sistema.hpp

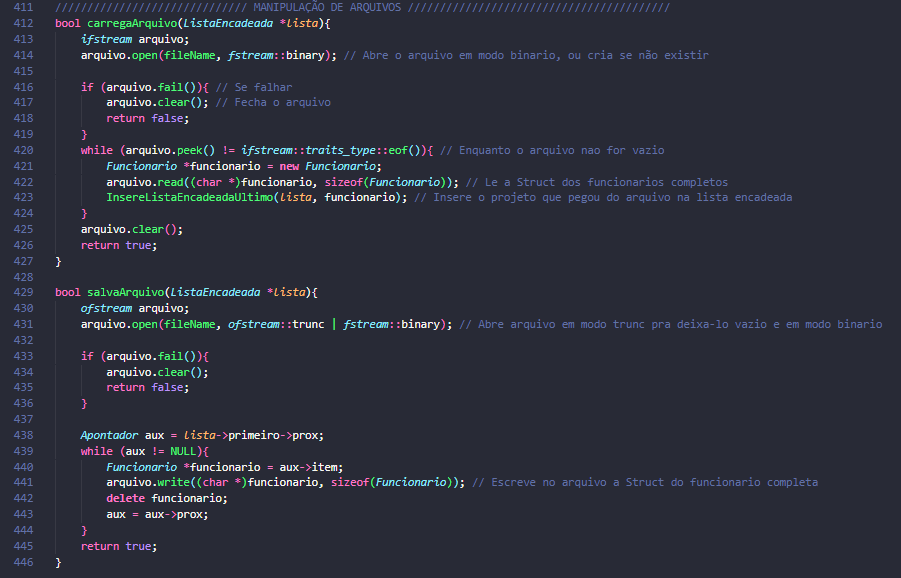




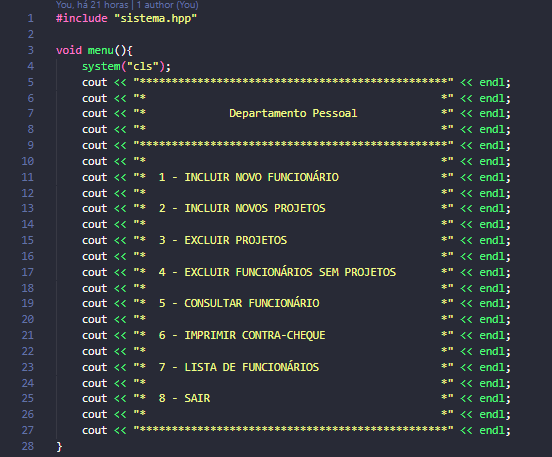


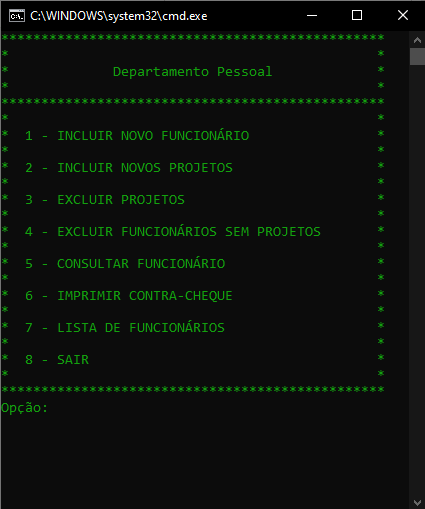


* 1. APÊNDICE C – Manipulação de arquivos

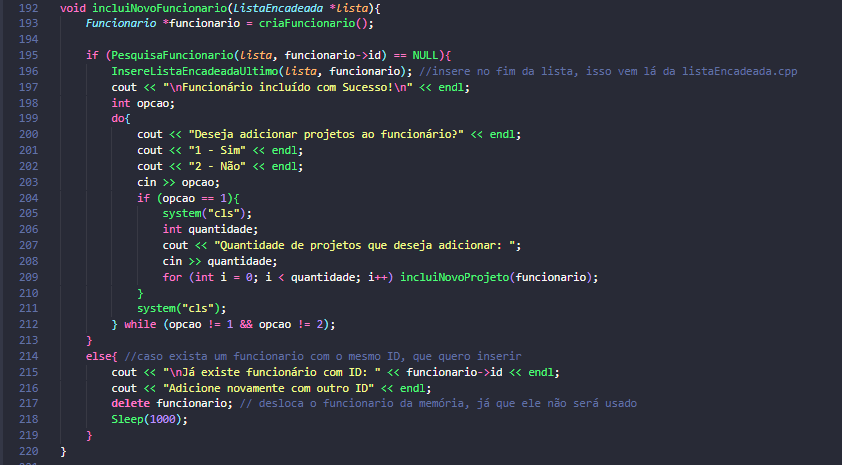


* 1. APÊNDICE D – Menu

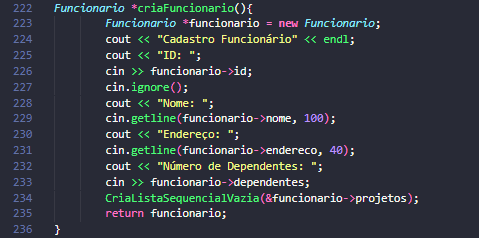




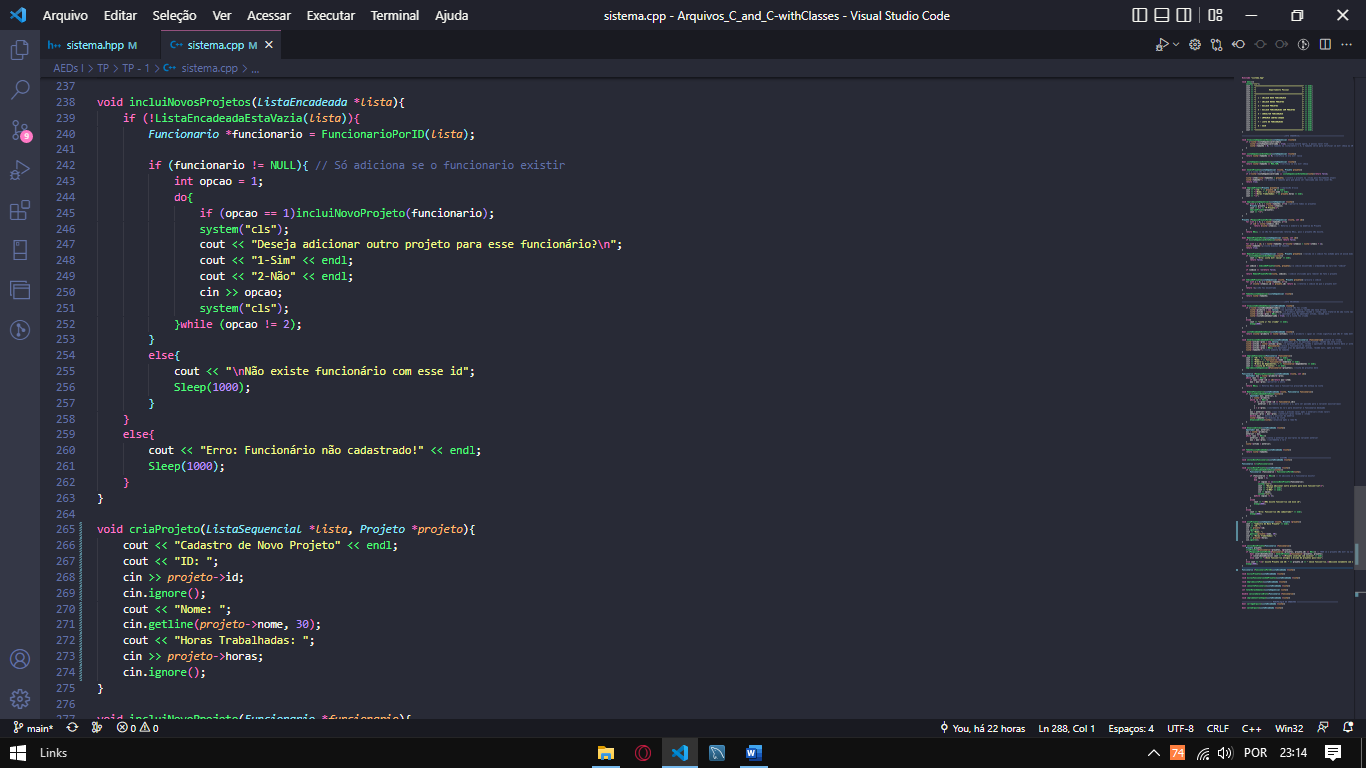
* 1. APÊNDICE E – Função de Inserção de funcionário

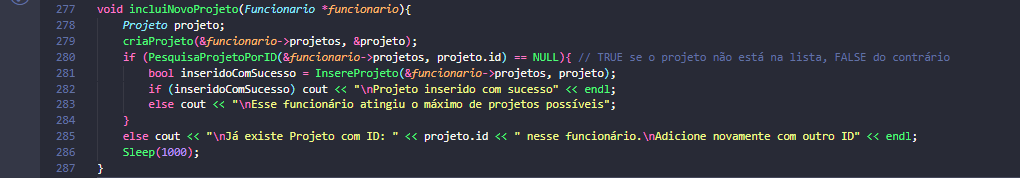


* 1. APÊNDICE F – Função de retorno de funcionário, para inserção

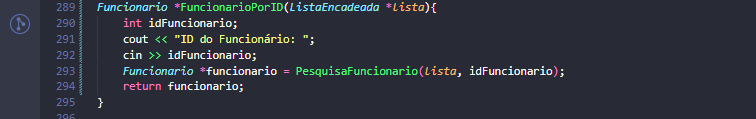


* 1. APÊNDICE G – Inclusão de Projeto

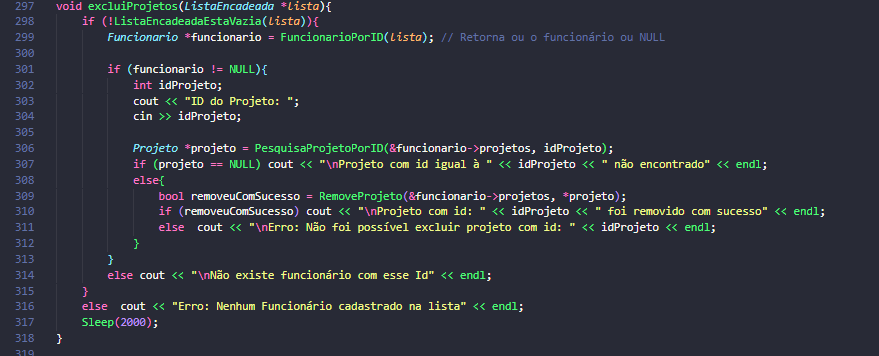




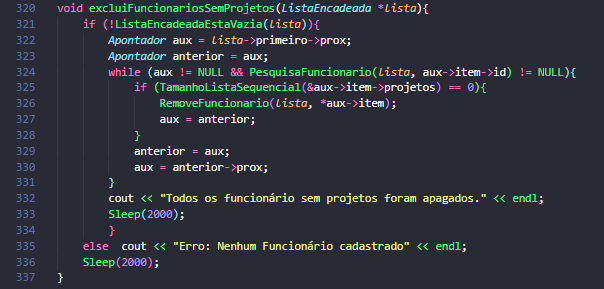
* 1. APÊNDICE H – Pesquisa funcionário por ID



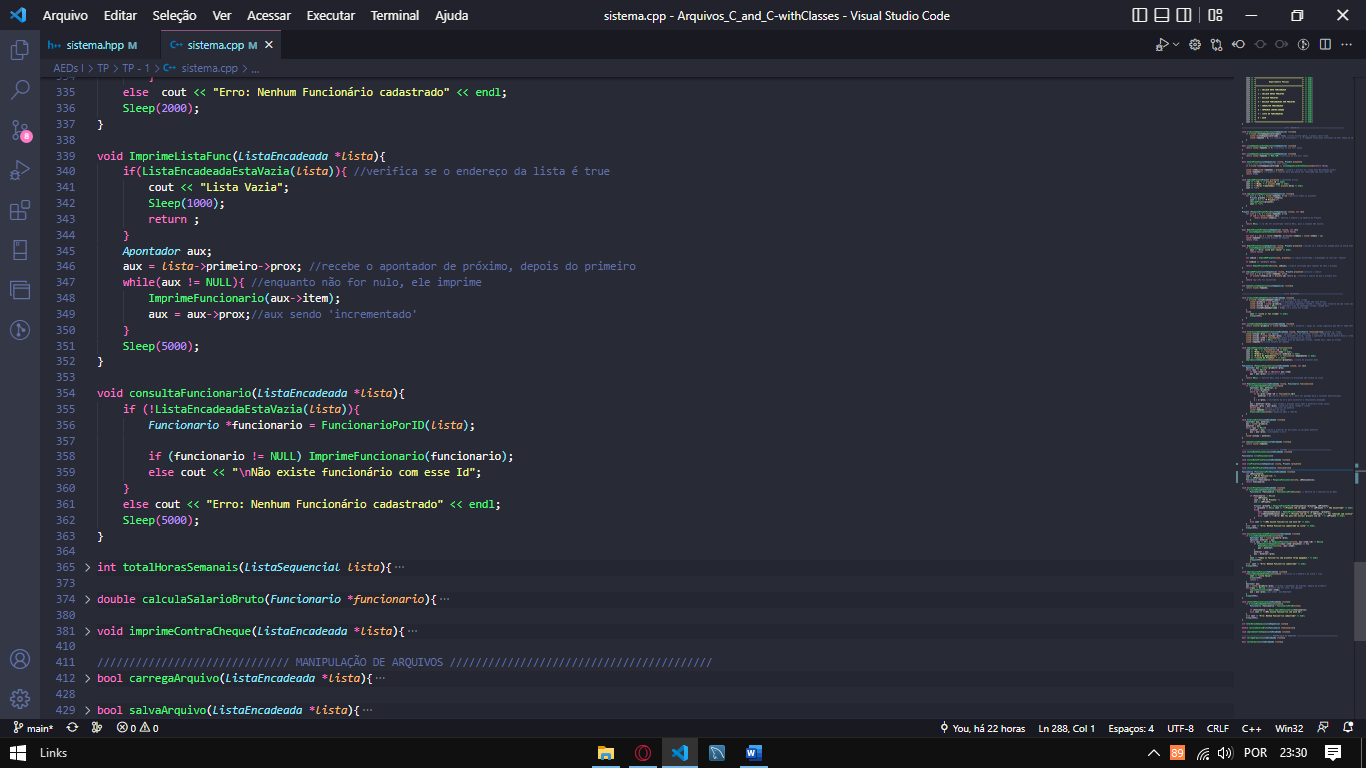
* 1. APÊNDICE I – Exclui Projetos



* 1. APÊNDICE J – Exclui funcionários sem Projeto



* 1. APÊNDICE K – Impressão de Funcionário



* 1. APÊNDICE L – Cálculo de Salário Bruto

