**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 II**

**SÃO JOÃO EVANGELISTA**

**OUTUBRO - 2022**

GABRIEL KÁICON BATISTA HILÁRIO

SISTEMA DE DELIVERY

**SÃO JOÃO EVANGELISTA**

**OUTUBRO - 2022**

SUMÁRIO

[1. INTRODUÇÃO 4](#_Toc116083969)

[1.1. Objetivo Geral 4](#_Toc116083970)

[1.2. Objetivos Específicos 4](#_Toc116083971)

[1.3. Justificativa 4](#_Toc116083972)

[2. DESENVOLVIMENTO 6](#_Toc116083973)

[2.1. Conceitos Aplicados 6](#_Toc116083974)

[2.1.1. Tipos Abstratos de Dados 6](#_Toc116083975)

[2.1.2. Pilha 7](#_Toc116083976)

[2.1.3. Lista Sequencial 8](#_Toc116083977)

[2.1.4. Arquivos 8](#_Toc116083978)

[2.2. Implementação 9](#_Toc116083979)

[3. CONCLUSÃO 14](#_Toc116083980)

[4. REFERÊNCIAS 15](#_Toc116083984)

[5. APÊNDICES 16](#_Toc116083985)

[5.1. APÊNDICE A – 16](#_Toc116083986)

[5.2. APÊNDICE B – **Erro! Indicador não definido.**](#_Toc116083987)

# 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 Pilha, Listas com arranjo (Lista Sequencial), Tipos Abstratos de Dados (TADs) e Manipulação de Arquivos, utilizando a linguagem de C++ para escrita dos códigos.

## Objetivos Específicos

Esse trabalho tem como objetivos específicos:

* Apresentar conhecimentos Pilha;
* Aprimorar conhecimentos em lista com arranjo;
* Desenvolver novas linhas de raciocínio para outros tipos de lógicas de aplicativos.

## Justificativa

Ao iniciar os estudos de Algoritmos e Estruturas de Dados I, vemos os conteúdos de Ponteiros, TADs, Manipulação de Arquivos, Listas com Arranjo e com Ponteiros, e agora vimos Pilha.

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. Seja ela qualquer um dos dois tipos apresentados, no caso 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, e 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. Por fim vemos o conteúdo de Pilha com Arranjo, que funciona como uma pilha na vida real, onde empilhamos as coisas, e para retirar o primeiro item empilhado, devemos remover todos os itens empilhados após ele, para remove-lo, e para remover o segundo, devemos remover todos os itens inseridos após ele, para então remover o segundo, e isso serve para todo e qualquer item empilhado. Resumindo, o primeiro que entra é o último que sai.

No trabalho, é exigido que usemos pilha, antes devemos cadastrar pedidos em uma pilha com arranjo, e utilizei uma lista com arranjo com produtos cadastrados de uma lista. Nem todos os conteúdos foram aplicados para realização do trabalho, apenas os conteúdos de pilha, lista com arranjo e arquivos, criando um minissistema de delivery que nos permite empilhar pedidos contendo produtos.

Tendo isso tudo em vista, o trabalho foi exigido para que seja possível desenvolver o raciocínio lógico quanto a aplicação dessa estrutura, e com o bônus de novamente aplicar elas em conjunto.

# DESENVOLVIMENTO

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

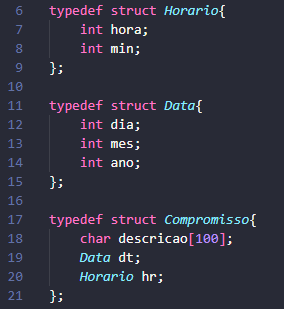
## Conceitos Aplicados

Explicação sucinta dos conceitos de Pilha, 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



## Pilha

Figura 2 – Pilha



Na figura 2, vemos um desenho esquemático de como é a estrutura de dados da pilha. O limite dela é definida na hora da criação, o N seria o topo – 1, seria o último item empilhado. Na figura 3 vemos o processo de empilhamento de itens em uma pilha, e na figura 4 vemos o desempilhamento.

Figura 3 – Empilhamento de itens

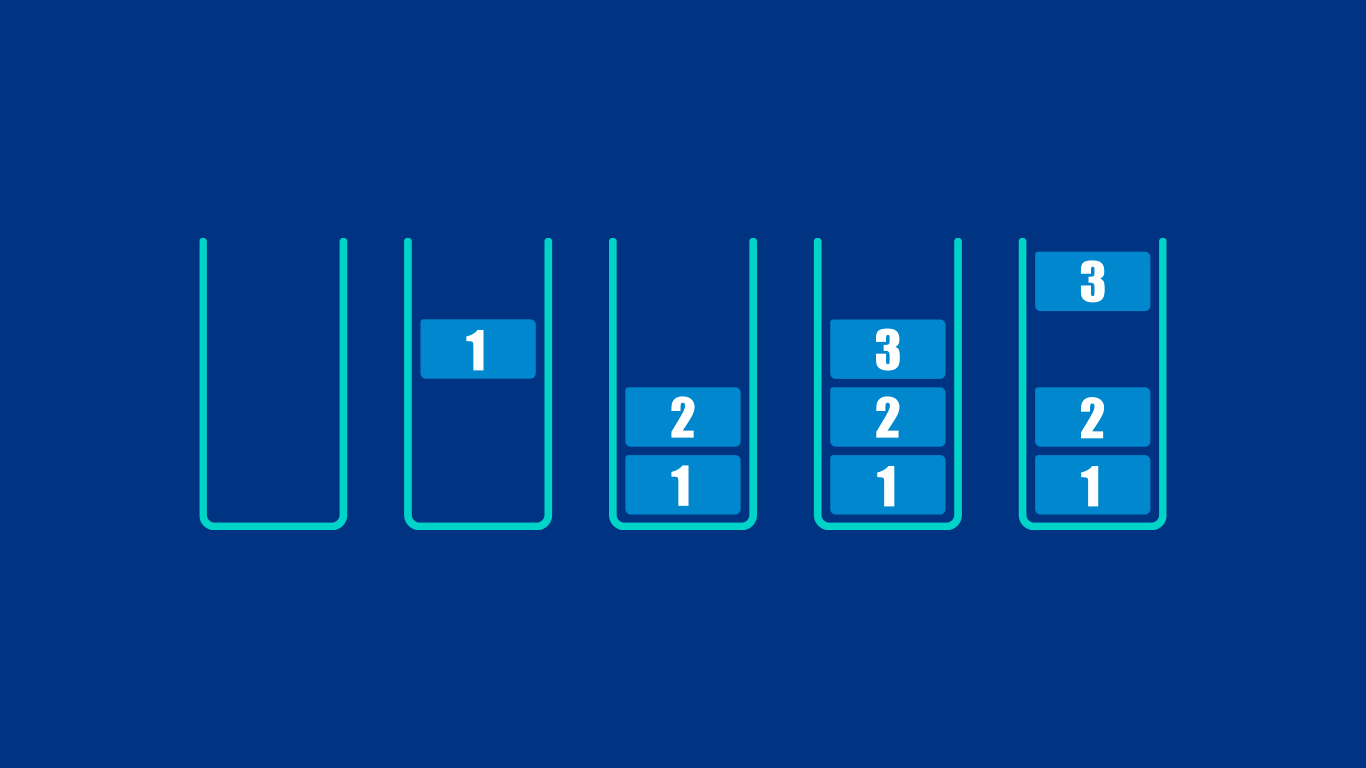
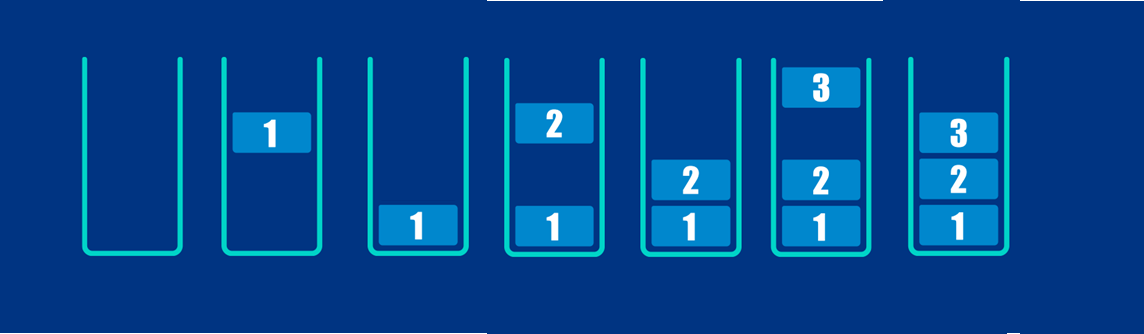
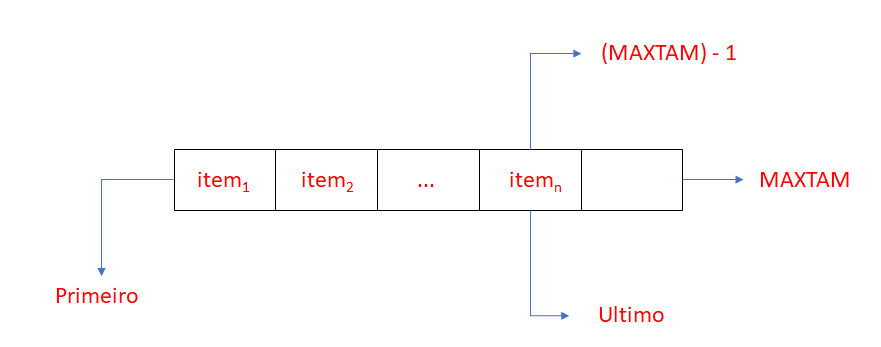


Figura 4 – Desempilhamento de itens



## 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 última posição até o momento, e assim sucessivamente, até que a Lista alcance o tamanho máximo, que seria o valor definido para o MAXTAM.

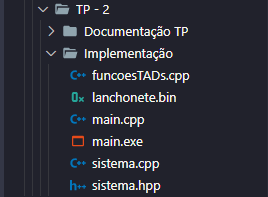
## Arquivos

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

Utilizei uma modularização que está na figura 5, dividindo em 6 arquivos, 3 arquivos \*.cpp, 1 arquivo \*.bin, 1 arquivo \*.hpp, e 1 arquivo \*.exe.

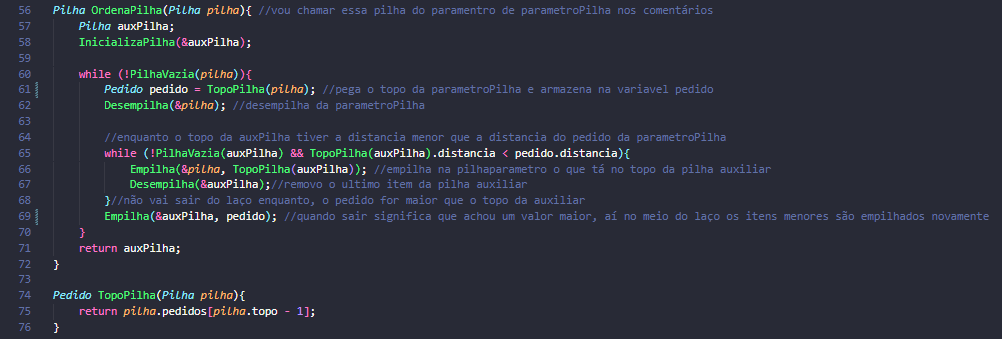
Figura 5



No arquivo funcoesTADs.cpp, tenho todas as funções das TADs de pilha (Apêndice A) e de lista (Apêndice B). No sistema.hpp (Apêndice C), contém as TADs de pilha e lista, além do cabeçalho das funções das TADs, do arquivo.cpp e da manipulação de arquivos. A manipulação de arquivos (Apêndice D) utilizadas foi mesma de anteriormente, porém adaptado para esse trabalho prático.

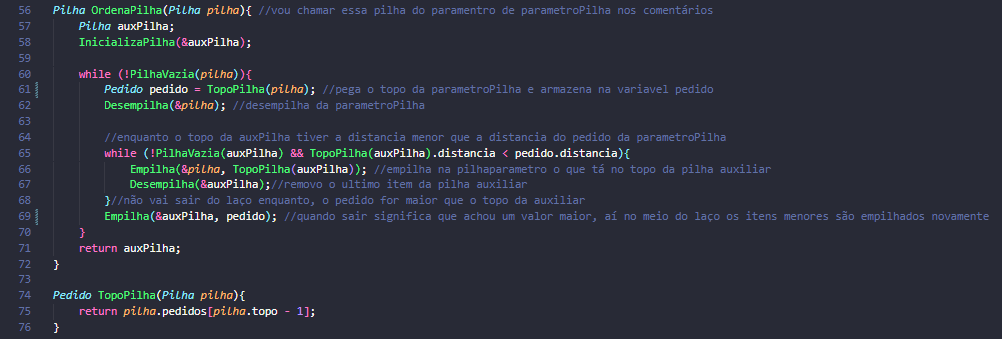
Das funções apresentadas nas TADS, temos 2 diferentes do padrão, a de ordenar pilha, na figura 6, e uma que recolhe o pedido que está topo da pilha, na figura 7.

A função OrdenaPilha, serve para ordenar uma pilha com base na distancia dos pedidos contidos nessa pilha.

Figura 6  


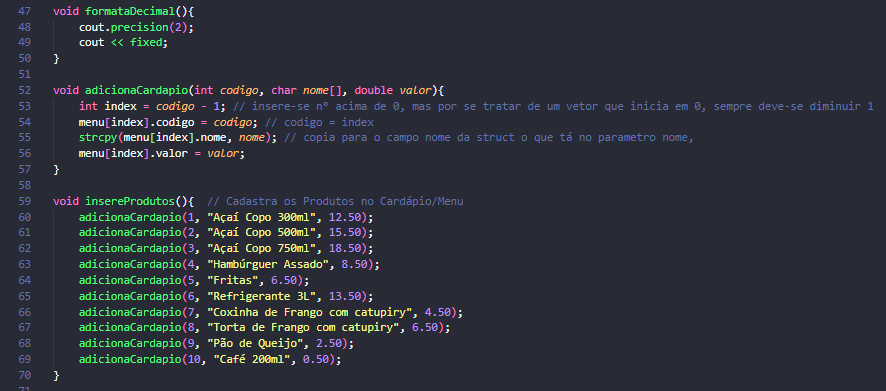
A função TopoPilha, serve para guardar o último pedido inserido na pilha.

Figura 7



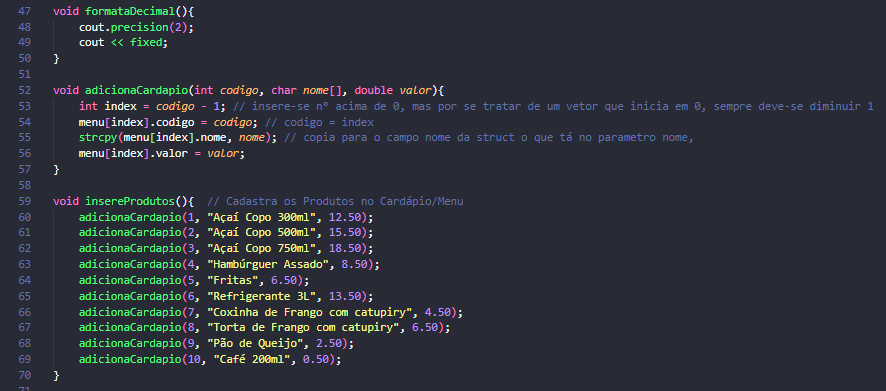
A função formataDecimal na figura 8, serve para deixar os valores de tipo double, em 2 casas decimais.

Figura 8

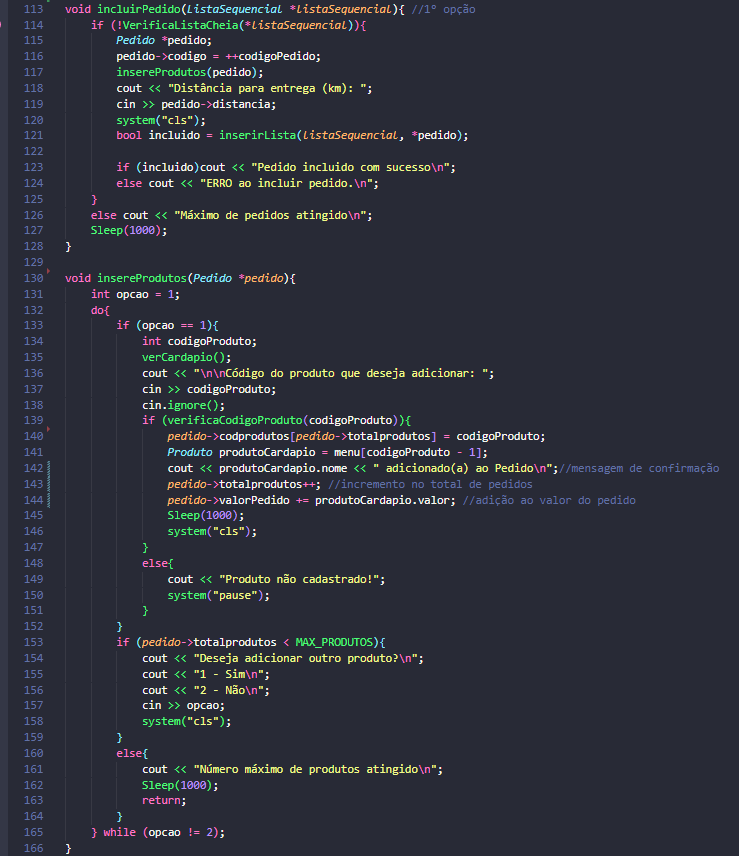


Na figura 9, vemos duas funções agindo em conjunto. A função adicionaCardapio, serve para que seja possível adicionar produtos no vetor de Produtos chamado menu, que está no arquivo sistema.hpp (Apêndice C), e a função insereProdutos, faz uso da função anterior para inserir os produtos de fato, no cardápio no momento da inicialização.

Figura 9

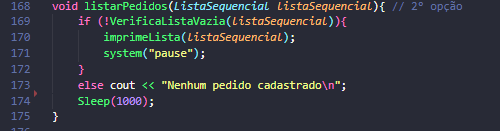


A Menu (Apêndice F) contém as opções exigidas no trabalho. A começar pela primeira opção, a Inclusão de pedido mostrada na figura 10. O código começa verificando se a lista está cheia, se tiver, a inserção não será feita, é criada uma variável pedido, que recebe um código de acordo com a variável que evita a repetição de códigos, para que o usuário não cometa o erro de digitar um código já existente, logo em seguida vai para função insereProdutos, que tem o pedido como parâmetro, para que sejam inseridos produtos apenas naquele pedido, lá dentro da função, é inserido um código já existente, e é feita uma verificação de existência desse código, feita pela função verificaCodigoProduto (Apêndice G), se o código existe, o produto é incluído no pedido, e depois disso, vem a mensagem de confirmação, e se quer que insere um novo produto, é claro, respeitando o limite de produtos por entrega.

Figura 10

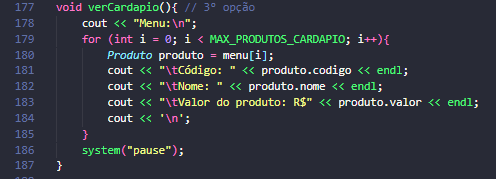
A segunda opção é a de Listar pedidos, onde a função listapedidos, na figura 11 é chamada, que simplesmente verifica se lista está vazia, e se não tiver, ele exibe a lista existente.

Figura 11



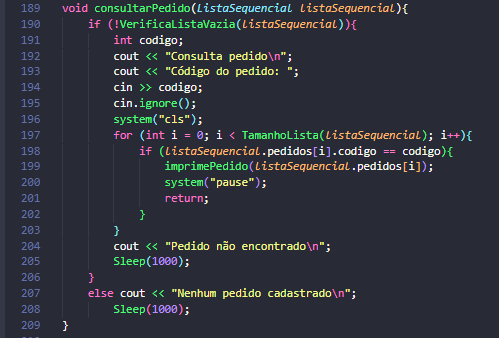
A terceira opção é a de ver o cardápio, onde a função verCardapio, na figura 12 é chamada, que imprime os produtos cadastrados no inicio da execução do programa.

Figura 12



A quarta opção é a de consultar pedido, onde a função consultaPedido, na figura 13, procura por um pedido dentro da lista, e imprime ele, caso ele seja encontrado.

Figura 13



Temos a quinta opção, que é a de imprimir a lista de entrega, para isso temos função que converte uma lista para uma pilha na figura 14, para que seja possível imprimir a lista de entregas (que na verdade é uma pilha de entrega) na figura 15. A conversão de uma estrutura de dados em outra é possível nesse caso, pois ambas as estruturas possuem pedidos. Para mostramos a pilha de entregas, devemos ordenar ela, da distancia mais curta primeiro, e a mais longa por último.

Figura 14

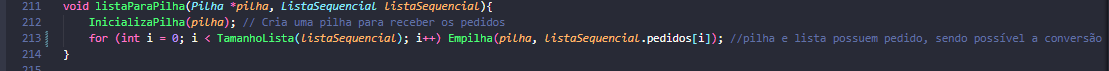
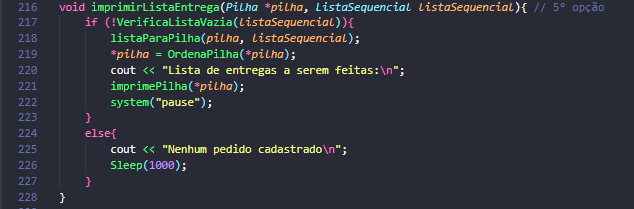
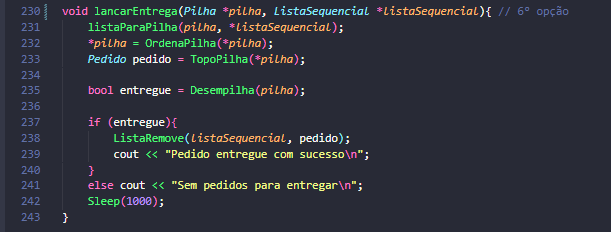


Figura 15



Temos a sexta opção, que é a função de lançar entrega vista na figura 16, que de forma resumida, ela simplesmente remove o topo da pilha de entregas.

Figura 16



# CONCLUSÃO

Ao longo do trabalho tive várias ideias de como resolver o problema, essa parte de pilha é mais interessante, reutilizei os códigos padrões das TADs e fiz as devidas alterações, reutilizei um pouco do código do último trabalho que fiz, para a manipulação de arquivos, e reutilizei textos do meu 1° trabalho prático. Fiz uso dos slides disponibilizados pelo professor para sanar dúvidas a respeito de pilha.

Coloquei o trabalho em modularização separando-o em 6 arquivos:

* Dois arquivos \*.cpp:
* sistema.cpp, com as funções exigidas
* funcoesTADs, com as funções das TADs
* main.cpp com a execução das funções,
* Um arquivo \*.hpp, o sistema.hpp, onde tinha as structs e o cabeçalho das funções,
* Um arquivo \*.bin, lanchonete.bin onde os funcionários eram salvos,
* Um arquivo \*.exe, a main.exe, que seria o executável do código, onde as funções compilavam.

Creio eu que meu desenvolvimento foi bom nesse trabalho, desenvolvi novas formas de raciocínio e tive que abstrair e diminuir muitas coisas para que o trabalho ficasse leve, sem ser muito extenso. Consegui atingir meus objetivos, e explicar bem o que cada função faz, apresentar conhecimentos em Pilha, e com confiança aprimorar meus conhecimentos em lista sequencial e fazer uso da mesma dentro do meu programa novamente, e aplicar isso em um software de *delivery*, que aumenta minha percepção para conseguir criar um software de entregas. Estou feliz e satisfeito com o resultado, e ele atendeu às minhas expectativas além do que eu esperava.



# REFERÊNCIAS

TRINDADE. Eduardo. 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: 7 de outubro de 2022.

TRINDADE. Eduardo. 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: 7 de outubro de 2022.

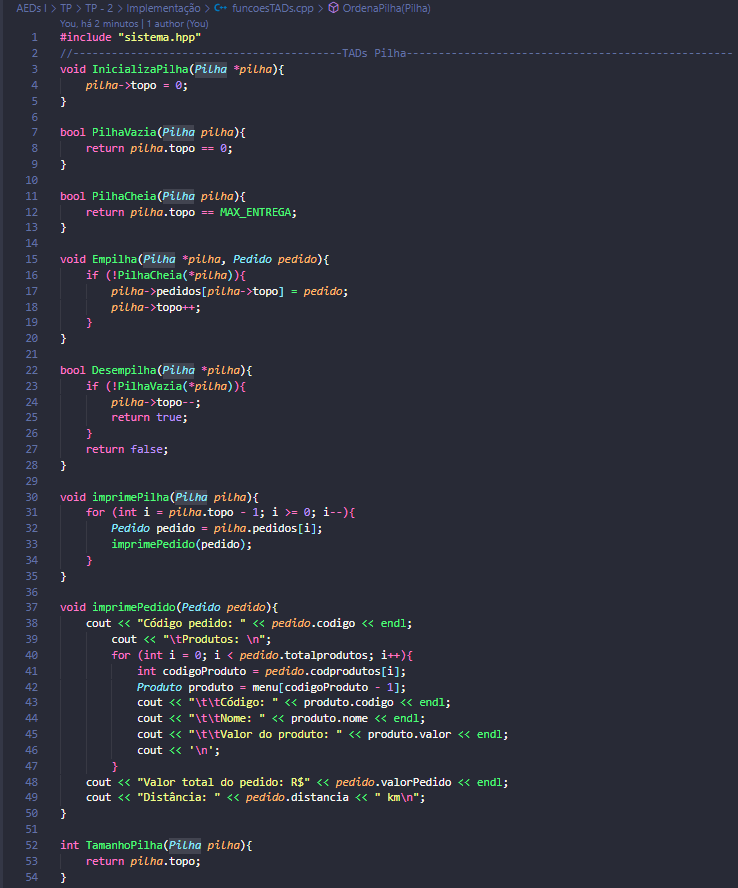
TRINDADE. Eduardo. Algoritmos e Estrutura de Dados – Pilha. 2022. Apresentação PDF. Disponível em: <https://ead.ifmg.edu.br/saojoaoevangelista/pluginfile.php/146500/mod_resource/content/1/Aula%207%20-%20Listas%20Encadeadas.pdf> . Acesso em: 7 de outubro de 2022.

Ana Paula Andrade. O que é e como funciona a Estrutura de Dados Pilha. **TREINAWEB.** Outubro de 2020. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-e-como-funciona-a-estrutura-de-dados-pilha>. Acesso em: 7 de outubro de 2022.

Link do código no *Github*: <https://github.com/gKaicon/Arquivos_C_and_C-withClasses/tree/main/AEDs%20I/TP/TP%20-%202>

# APÊNDICES

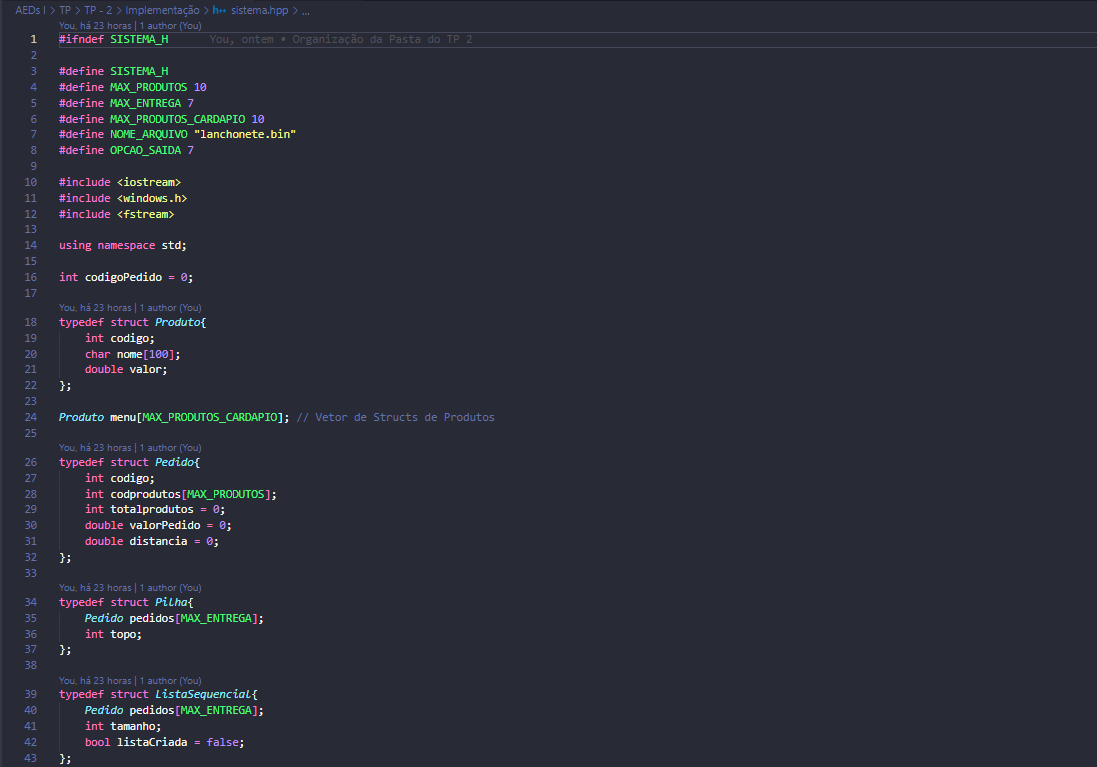
* 1. APÊNDICE A – TADs de Pilha



* 1. APÊNDICE B – TADs da Lista

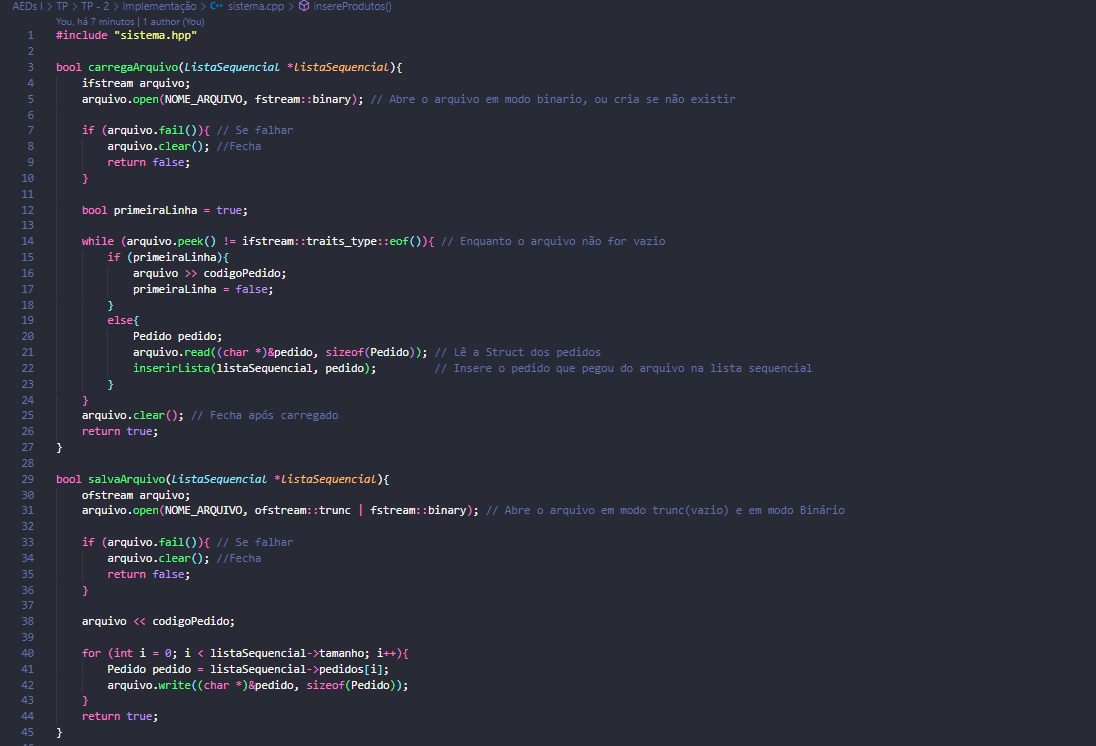


* 1. APÊNDICE C – Sistema.hpp

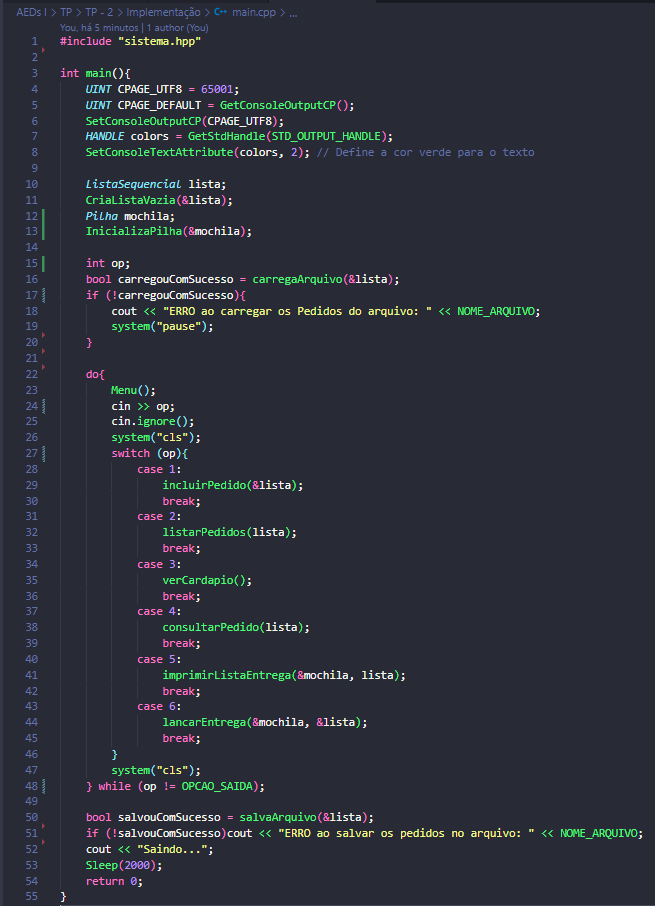




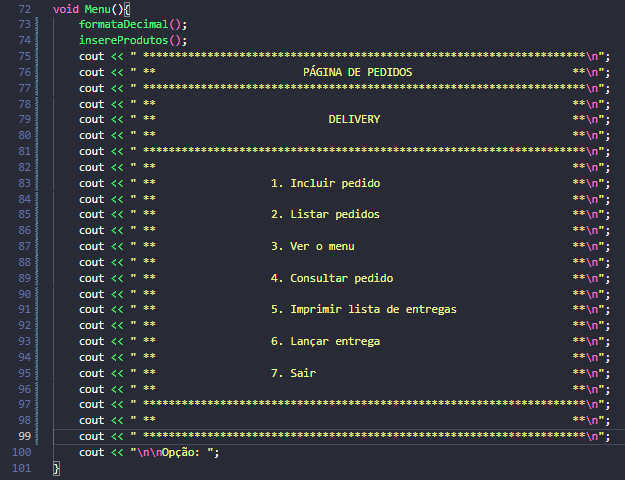
* 1. APÊNDICE D – Manipulação de Arquivos



* 1. APÊNDICE E – Main.cpp



* 1. APÊNDICE F – Menu





* 1. APÊNDICE G – Validação do código do produto

