Trabalho Prático 1:
Servidor de emails

Vinicius Cardoso Antunes
Matrícula: 2020027210

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brasil

viniciusc@ufmg.br

1. Introdução:

Este trabalho consiste em desenvolver um servidor de emails, ou seja, um programa capaz de gerenciar usuários e realizar o armazenamento de mensagens, assim como leitura e escrita com diferentes ordens de prioridade.

Ao final deste trabalho, espera-se que o aluno tenha adquirido novas experiências com o desenvolvimento de software focado em uso racional da memória e processamento de um computador utilizando algoritmos e estruturas de dados.

2. Método:

2.1. Estruturas de Dados:

Para a implementação do projeto, foram criadas as classes Email, ListaDeEmails, CaixaDeEntrada, ListaDeCaixas e ServidorDeEmails, que representam o Email, Caixa de Entrada e o Servidor de Emails.

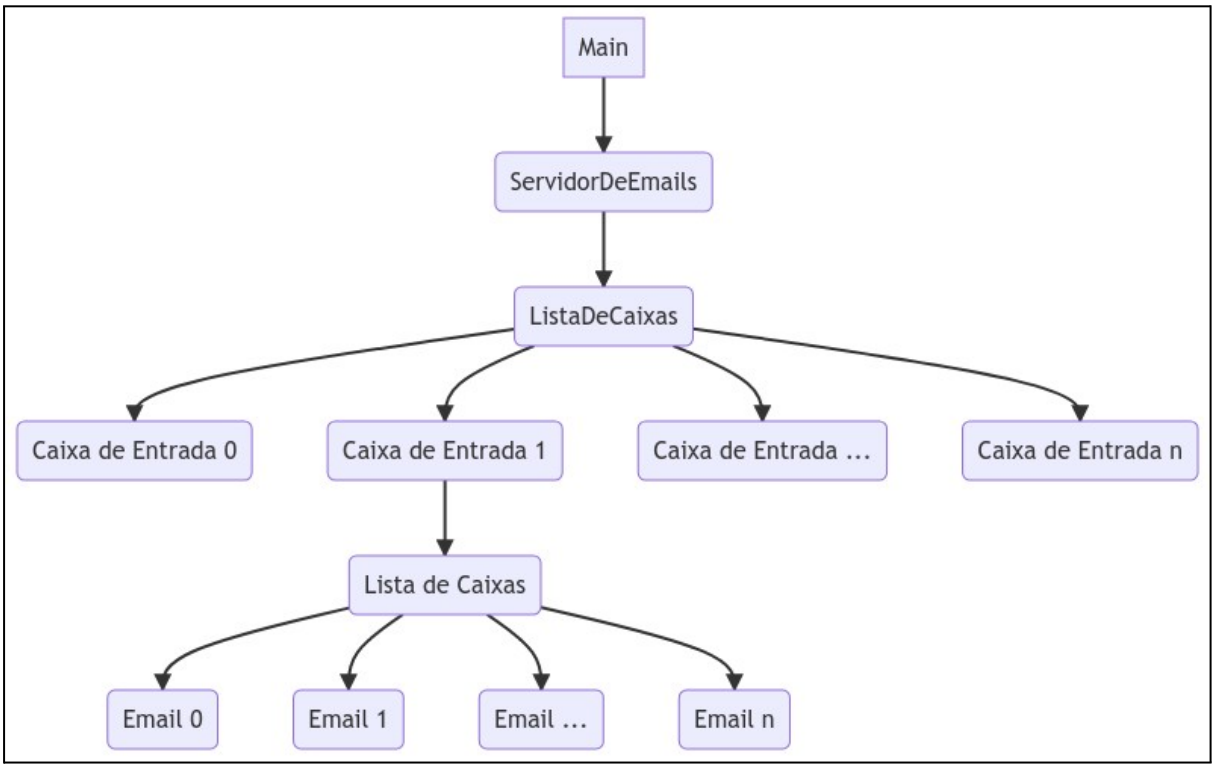
A função Main cria um objeto do tipo Servidor de Emails e processa o argumento passado pelo usuário contendo o nome do arquivo de entrada. Enquanto não chega ao fim do arquivo, o sistema recebe a operação que o servidor deve realizar, sendo elas:

- CADASTRA
- REMOVE
- ENTREGA
- CONSULTA

Para realizar as operações, foi desenvolvido neste trabalho as classes Email, que possui prioridade e a mensagem, a Caixa de Entrada, que possui um id e armazena uma lista de emails e o Servidor de Emails, o qual armazena uma lista de caixas de entrada.

Na figura abaixo são representadas todas as estruturas utilizadas, assim como a ligação entre elas:

Figura 1: Estruturas de dados do Servidor de Emails



2.2. Classes:

- **Email:**

  Atributos:
  - prioridade (int)
  - mensagem (string)
  - proximo (Email)

  Métodos:
  - Email(Email *email)
  - Email(int prioridade, string mensagem)
  - ~Email()
  - GetPrioridade() - int
  - GetMensagem() - string

- **ListaDeEmails:**

  Atributos:
  - tamanho (inteiro)

- primeiro (Email)
- ultimo (Email)

Métodos:
- ListaDeEmails()
- ~ListaDeEmails()
- IsVazia() - bool
- InserirEmailOrdenado(Email *email) - void
- RemoverEmailInicio() - string
- Limpar() - void

- **CaixaDeEntrada:**

Atributos:
- id (int)
- lista_de_emails (ListaDeEmails)
- proximo (CaixaDeEntrada)

Métodos:
- CaixaDeEntrada()
- CaixaDeEntrada(int id)
- ~CaixaDeEntrada()
- IsVazia() - bool
- GetId() - int
- AdicionarEmail(Email *email) - void
- LerEmail() - string

- **ListaDeCaixas:**

Atributos:
- tamanho (int)
- primeiro (CaixaDeEntrada)
- ultimo (CaixaDeEntrada)

Métodos:
- ListaDeCaixas()
- ~ListaDeCaixas()
- IsVazia() - bool
- GetCaixa(int id) - CaixaDeEntrada
- InserirCaixa(CaixaDeEntrada *caixa_de_entrada) - void
- RemoverCaixa(int id) - void
- PesquisarCaixa() - int
- Limpar() - void

- **ServidordeEmails:**

  Atributos:
  - caixas_de_entrada (ListaDeCaixas)

  Métodos:
  - ServidorDeEmails()
  - ~ServidorDeEmails()
  - CadastrarId(int id) - void
  - RemoverId(int id) - void
  - EntregarMensagem(int id, int prioridade, string mensagem) - void
  - ConsultarId(int id) - void

3. Análise de Complexidade:

Para analisar a complexidade do projeto, é necessário realizar a análise dos métodos das estruturas de dados e algoritmos de pesquisa utilizados:

ListaDeEmails::InserirEmailOrdenado(): A inserção ordenada em uma lista encadeada, no pior caso, irá percorrer toda a lista procurando a posição correta para o objeto, menos o último elemento, que assim como o primeiro elemento, é verificado antes da lista ser percorrida, com isso o pior caso é **O(n)** e o melhor caso é **O(1)**, como foi descrito anteriormente.

ListaDeEmails::RemoverEmailInicio(): A remoção de um objeto do início da lista possui melhor e pior caso iguais, **O(1)**.

ListaDeCaixas::InserirCaixa(): A inserção no final da lista possui pior caso e melhor caso iguais, **O(1)**.

ListaDeCaixas::RemoverCaixa(): Assim como no caso de inserção ordenada, o pior caso para remover um objeto da lista ocorre quando é necessário percorrer toda a lista procurando a posição certa, sendo **O(n)**. o melhor caso é quando o objeto procurado é o primeiro da lista, **O(1)**.

ListaDeCaixas::PesquisarCaixa(): A pesquisa em uma lista encadeada possui melhor e pior caso **O(n)**.

4. Estratégias de Robustez:

Caso o arquivo de entrada não exista ou seja inválido, é gerada uma exceção.

Ao passar os parâmetros na linha de comando, é retornada uma exceção se o usuário não informar o nome de algum arquivo.

Caso o arquivo contenha uma operação que não seja CADASTRA, REMOVE, ENTREGA ou CONSULTA, o programa lança uma exceção informando que há uma operação inválida no arquivo de entrada.

5. Conclusões:

Neste trabalho foi desenvolvido um servidor de emails utilizando a linguagem C++ e estruturas de dados (Lista Encadeada). Além disso, foi realizada a análise de complexidade do programa.

Dentre os aprendizados, podem ser citados a manipulação de arquivos de texto tanto para escrita quanto para leitura, a passagem de parâmetros pela linha de comando utilizando o argv e argc, os conceitos de programação orientada a objetos em geral, como a abstração das classes que devem ser utilizadas, os atributos e métodos, os métodos construtores, Setters e Getters, e o uso de estruturas muito comuns em projetos de todas as linguagens.

6. Bibliografia:

TUTORIALS POINT. **C++ Files and Streams**. Disponível em: www.tutorialspoint.com/cplusplus/cpp_files_streams.htm . Acesso em 02 out. 2022.

AGOSTINHO BRITO. **Programação Orientada a Objetos em C++**. Disponível em: agostinhobritojr.github.io/tutorial/cpp/ . Acesso em 02 out. 2022

MIND BENDING. **Argumentos e Parâmetros em C**. Disponível em: mindbending.org/pt/argumentos-e-parametros-em-c. Acesso em 03 out. 2022

WIKIPEDIA. **Netpbm**. Disponível em: https://en.wikipedia.org/wiki/Netpbm. Acesso em 04 out. 2022

7. Instruções para compilação e execução:

- Acesse o diretório TP
- A partir de um terminal Linux, compile os arquivos do TP utilizando o comando *make*.
- Será gerado na pasta ‘obj’ os object files
- Será gerado na pasta ‘bin’ o arquivo run.out, que é o executável do projeto.
- Execute o arquivo run.out com o comando:

./bin/run.out <arquivo_entrada.txt>

- Será gerado uma saída na linha de comando