SCC0502 Algoritmos e Estruturas de Dados 1

Exercício 1 - Pontos Cartesianos

Objetivo

O objetivo deste primeiro exercício prático é revisar tópicos que já foram vistos em disciplinas anteriores e que serão essenciais para os próximos trabalhos e atividades.

Queremos que os alunos se acostumem com o uso de múltiplos arquivos *.c* e *.h* em um projeto, bem como a construção de um arquivo **Makefile**, responsável por gerenciar a execução do programa.

Descrição

Seu projeto deve ser capaz de receber as coordenadas de pontos do plano cartesiano, na forma *(x, y)*. Esse conjunto de pontos descreve um caminho a ser seguido linearmente, ou seja, um ponto após outro.

Espera-se que seu programa seja capaz de calcular a distância total percorrida do primeiro ponto até o último, na ordem em que foram inseridos.

Sejam os pontos:

$$P_{inicial} = (x_i, y_i), P_{final} = (x_f, y_f)$$

A distância deve ser calculada por:

$$d = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}$$

Entrada

Seu algoritmo deve receber primeiro um valor do tipo inteiro *N*, que representa a quantidade de pontos a serem lidos em sequência. Após isso, seu programa deve ler *N* pares de valores do tipo ponto flutuante na forma : *X Y*.

Exemplo de entrada:

```
3
0 0
3 0
6 4
```

Saída

Como saída, seu programa deve retornar um valor do tipo ponto flutuante, com precisão de duas casas decimais, que representa a distância total percorrida, passando por todos os pontos na ordem em que foram lidos.

Saída da caso acima:

```
8.00
```

Observações da implementação

Como é descrito na sessão objetivos desse documento, não queremos apenas que os alunos resolvam o problema, mas que utilizem métodos que serão comuns no decorrer da disciplina.

Devido a esse objetivo, será exigido que vocês desenvolvam seu projeto representando o objeto **ponto** e o objeto **caminho** de forma abstrata, isto é, com uma *struct* para cada. A memória deve ser alocada dinamicamente e ser devidamente liberada ao fim da execução.

Utilizar múltiplos arquivos *.c* e *.h* para separar a implementação e a responsabilidade dos métodos de cada objeto (ponto.c e caminho.c). Construir funções para realizar operações repetitivas, ou seja, modularizar adequadamente seu código.

Escrever um arquivo **Makefile** que será responsável por gerenciar a execução do projeto dentro da plataforma *run.codes*