Table of contents

Navegação

Parte 1 - magnitude de vector - 20%

- Implementa uma função que recebe 2 pontos, (x_1, y_1) e (x_2, y_2), e calcula a magnitude do vector formado por esses dois pontos.
- A função tem o seguinte cabeçalho:

```
double vec_mag(double x1, double y1, double x2, double y2)
```

- A magnitude de um vector é dada por $\sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}$
- Pode usar as seguintes funções da biblioteca math.h
  - `sqrt` calcula a raiz quadrada do argumento

```
double sqrt(double x);
```

  - `pow`, devolve a potência da base elevada ao expoente

```
double pow(double x, double y);
```

Parte 2 - receber coordenadas - 40%

- Escreve um programa que recebe coordenadas em tempo real.
  - Estas coordenadas são fornecidas pelas seguintes funções, já implementadas na biblioteca `lab1.h`:

```
double get_x();
double get_y();
double get_timestamp();
int gps_update();
```

  - A função `get_x` devolve a coordenada no eixo x, em metros;
  - A função `get_y` devolve a coordenada no eixo y, em metros;
  - A função `get_timestamp` devolve uma marca temporal em segundos.
  - A função `gps_update` atualiza o módulo de localização para a leitura seguinte - esta função deve sempre ser chamada entre 2 leituras consecutivas, e devolve -1 se não houverem novas leituras, e.g.

```
x = get_x();
y = get_y();
t = get_timestamp();
gps_update();
novo_x = get_x();
novo_y = get_y();
novo_t = get_timestamp();
```

- Implementa um programa que recebe continuamente novas localizações enquanto estas existirem e escreve na consola a marca temporal, x e y de cada coordenada, e.g.

```
timestamp, x, y
24743, -0.68, 0.72
24760, -52.44, -19.25
24776, -124.91, 284.46
24792, -183.57, 530.38
24809, -235.90, 758.32
24825, -283.43, 1053.49
24842, -341.43, 1274.74
24859, -430.48, 1479.85
```

Parte 3 - Calcula velocidade actual - 20%

- No ciclo anterior, calcula a velocidade actual em m/s
- A velocidade atual é a magnitude do vetor formado pelos pontos da última e penúltimas coordenadas, a dividir pelo tempo que passou entre essas 2 leituras (isto é, a diferença entre as duas últimas marcas temporais em segundos)

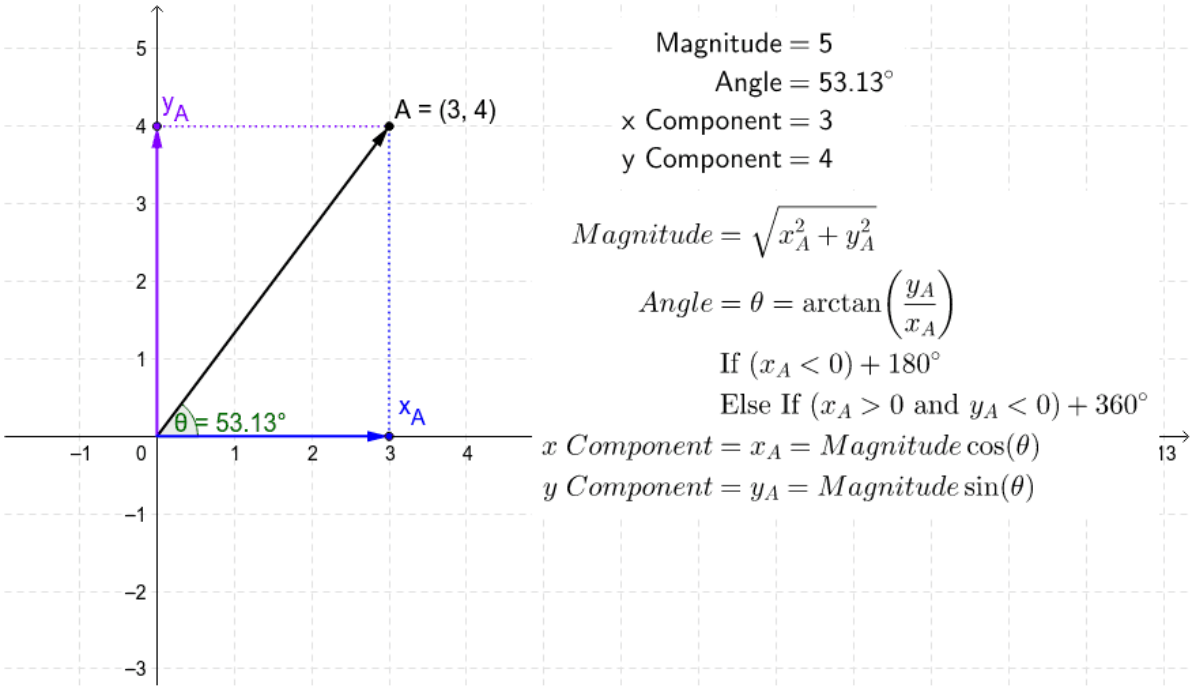
- Escreve na consola, para cada coordenada recebida, a velocidade actual.
- Na primeira coordenada não é possível calcular a velocidade.
- Recebe primeira uma coordenada e começa a calcular a velocidade a partir da segunda.
- Implementa esta funcionalidade na main.

Parte 4 - distância ao destino - 10%

- No ciclo anterior, escreve na consola a distância em kilometros entre a posição actual e o destino.
- A coordenada destino é dada pelas variáveis globais `target_x` e `target_y`
- A distância é a magnitude do vector formado entre a coordenada actual e coordenada destino.
- Lembra-te que as coordenadas estão em metros, é necessária a conversão ou das coordenadas ou da distância para kilometros.

Parte 5 - erro angular - 10%

- Escreve na consola o erro angular entre o rumo actual e o rumo directo ao destino.
- O rumo atual é o ângulo do vetor formado pelos pontos da última e penúltimas coordnadas
- Para calcular o erro angular, deve-se primeiro calcular o ângulo do vetor formado pelos pontos da última coordenada e a coordenada destino. Depois é necessário verificar a diferença entre esse ângulo e o rumo atual.
- Usa a função vec_ang para calcular o ângulo do vector
  - a função vec_ang recebe 1 vetor (na forma de 2 pontos, tal como `vec_mag`) e devolve o seu ângulo



Magnitude = 5

Angle = 53.13°

x Component = 3

y Component = 4

$$Magnitude = \sqrt{x_A^2 + y_A^2}$$

$$Angle = \theta = \arctan\left(\frac{y_A}{x_A}\right)$$

If $(x_A < 0) + 180°$

Else If $(x_A > 0 \text{ and } y_A < 0) + 360°$

$$x\ Component = x_A = Magnitude\cos(\theta)$$

$$y\ Component = y_A = Magnitude\sin(\theta)$$