SSC0603 – Estrutura de Dados – Trabalho 2: Detran Maps Simulator v1.01

Usando dados reais obtidos de um GPS de baixo custo, desenvolveremos um programa que permita ler tais dados, realizar estimativas de distância percorrida e velocidade, e realizar análises dos dados adquiridos, seguindo as especificações abaixo:

1) O software deve ser capaz de ler informações em graus de latitude e longitude, extraindo informações de velocidade e posição.
2) As informações devem ser guardadas em uma lista de encadeamento duplo, fazendo uso de ponteiros.
3) Não é permitida a existência de nós com dados inválidos ou nós avulsos que não participarão do processo de alguma forma.
4) O padrão de entrada é:
    a) Quantidade de entradas de GPS.
    b) Entradas de GPS textuais na forma “latitude longitude” em graus com casas decimais.
        i) Os números “latitude” e “longitude” são separados por um espaço.
        ii) Cada entrada (“latitude longitude”) é separada por um caractere de quebra de linha (‘\n’).
        iii) “latitude” e “longitude” são números (sem sinal) com ponto flutuante e devem ser armazenados em variáveis de tipo *double* (é muito importante usar a precisão *double*, devido ao número de casas após a vírgula requerido nos cálculos).
        iv) Pelo tipo de variável ser *double*, utilize o *scanf* com o especificador de conversão “%lf”.
        v) Considere o intervalo de tempo de um segundo entre entradas e tempo inicial igual a zero.
        vi) Utilize *double* para outras variáveis que utilizem ponto flutuante.
    c) Modo de execução.
        i) “0” - Exibição de todas as informações já processadas.
        ii) “1” - Verificação de excesso de velocidade, sendo passadas três entradas separadas por quebra de linha:

(1) Posição do radar por meio de índice na lista encadeada.
(2) Distância de cobertura (alcance do radar) nos sentidos do trajeto.
(3) Limite máximo de velocidade em m/s.

iii) “2” - Simplificação do caminho por meio de varredura nos dois sentidos (ver exemplos abaixo para melhor compreensão) e posterior impressão. Para este modo, será dado o valor, em metros, que será o limite de distância no processo de simplificação, devendo ser armazenado em uma variável de tipo *double*.

d) A velocidade será medida entre o ponto atual e o ponto anterior. A velocidade é, portanto, a distância percorrida em 1 segundo (entre o instante de tempo t e t - 1), indicada em m/s.

e) A velocidade do ponto inicial é nula.

f) A distância percorrida é cumulativa, ou seja, acumular-se-ão as distâncias percorridas a cada novo dado de coordenada GPS lido.

g) A distância percorrida inicial é nula.

5) O padrão de saída para cada modo de execução é:

a) “0” - Exibição das informações latitude, longitude, tempo, velocidade e distância percorrida até o dado nó.

b) “1”.

i) “autuado”, se o motorista ultrapassou o limite máximo.

(1) Caso o motorista seja autuado, a impressão da maior velocidade aferida pelo radar deverá também ser impressa, após uma quebra de linha.

ii) “liberado”, do contrário.

iii) O limite máximo considera uma velocidade máxima que não pode ser ultrapassada (em relação a aquela coordenada, ou, "Entrada X") considerando um determinado alcance do radar (metros para frente e para trás da atual posição desta entrada da posição do radar). Portanto, são analisadas as entradas vizinhas anteriores e posteriores limitadas pelo alcance do radar.

c) “2” - Após a remoção dos pontos de GPS não relevantes, imprimir os pontos restantes na mesma forma que o modo de execução “0”.

d) Utilize o especificador adequado para cada tipo de variável nas funções *printf* (https://en.cppreference.com/w/c/io/fprintf).

6) Todas as regiões de memória alocadas dinamicamente DEVEM ser liberadas antes do encerramento da execução.

7) Não usar *Variable-Length Arrays* (declaração de um vetor por meio de uma expressão não constante para números de elementos).

8) COMENTAR O CÓDIGO!!!

As coordenadas usadas são do hemisfério SUL, no Brasil, portanto Latitude S (Sul) e Longitude W (Oeste). Neste trabalho foi omitido o "sinal" (+/-) e o N/S e E/W das coordenadas. Para visualizar as coordenadas dos exemplos fornecidos no GoogleMaps, adicione um sinal negativo na Lat. e Long. (S e W). Por exemplo: digite no "search" do GoogleMaps “-22.00517, -47.891024” e você estará em São Carlos!

Extração de distância de dados de GPS:

```
// Definam esta constante antes de inclusões de bibliotecas. Apesar de comum na maioria
das bibliotecas, a definição de M_PI não é um padrão ISO-C.
// Código baseado em https://bit.ly/2m2ycQc
#ifndef M_PI
#define M_PI 3.14159265358979323846264338327795
#endif

double grauParaRadiano(double angulo)
{
	return (angulo * M_PI) / 180.0;
}

double distancia(
	double latitude1,
	double longitude1,
	double latitude2,
	double longitude2)
{
	double diferencaLatitude = grauParaRadiano(latitude1 - latitude2);
	double diferencaLongitude = grauParaRadiano(longitude1 - longitude2);
	double a =
		pow(sin(diferencaLatitude / 2.0), 2.0) +
		cos(grauParaRadiano(latitude2)) *
		cos(grauParaRadiano(latitude1)) *
		pow(sin(diferencaLongitude / 2.0), 2.0);

	return 6378137.0 * (2.0 * atan2(sqrt(a), sqrt(1.0 - a)));
}
```

Exemplo:

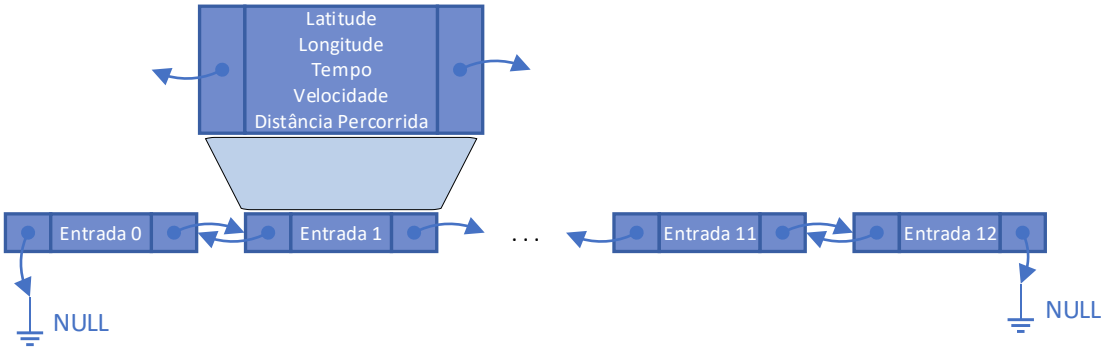
Entrada:

```
13 // Quantidade de entradas de GPS.
22.00517 47.891024
22.005123 47.89104467
22.00508983 47.89108183
22.0050745 47.89113433
22.00507367 47.89119983
22.00508117 47.8912795
22.00509033 47.89136533
22.00509717 47.89145033
22.00510117 47.8915275
22.00510367 47.891594
22.00510417 47.89165233
22.00510483 47.891706
22.00510617 47.8917525
0 // Impressão das coordenadas de GPS e informações processadas.
```

Saída:

```
// Latitude, Longitude, Tempo, Velocidade, Distância Percorrida
22.005170 47.891024 0 0.000000 0.000000
22.005123 47.891045 1 5.650236 5.650236
22.005090 47.891082 2 5.323880 10.974116
22.005074 47.891134 3 5.680899 16.655015
22.005074 47.891200 4 6.760883 23.415897
22.005081 47.891280 5 8.265013 31.680910
22.005090 47.891365 6 8.917003 40.597912
22.005097 47.891450 7 8.805825 49.403738
22.005101 47.891528 8 7.977147 57.380884
22.005104 47.891594 9 6.869100 64.249984
22.005104 47.891652 10 6.020492 70.270476
22.005105 47.891706 11 5.539764 75.810240
22.005106 47.891753 12 4.801579 80.611818
```

Após a leitura das entradas, os dados se organizarão de forma semelhante aos diagramas abaixo:



Com a estrutura corretamente inicializada, basta apenas imprimir cada uma das entradas, começando pela “Entrada 0”.

Exemplo:

Entrada:

```
13 // Quantidade de entradas de GPS.
22.00517 47.891024
22.005123 47.89104467
22.00508983 47.89108183
22.0050745 47.89113433
22.00507367 47.89119983
22.00508117 47.8912795
22.00509033 47.89136533
22.00509717 47.89145033
22.00510117 47.8915275
22.00510367 47.891594
22.00510417 47.89165233
22.00510483 47.891706
22.00510617 47.8917525
1 // Modo de operação de radar.
4 // O radar está em “Entrada 4”.
10.0 // Sua distância de cobertura é de 10 metros.
8.0 // A velocidade máxima permitida é 8 m/s.
```

Saída:

```
autuado
8.265013
```

Com a estrutura de exemplo anterior, tem-se:

Neste caso, o radar está localizado em “Entrada 4”, possuindo cobertura de 10 metros (cobrindo os nós “Entrada 3”, “Entrada 4” e “Entrada 5”) e velocidade máxima de 8 m/s.

Analisando os nós cobertos pelo radar, vê-se que houve uma infração em “Entrada 5”.

Exemplo:

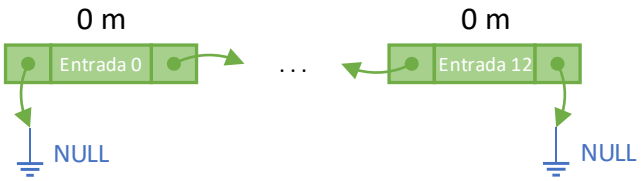
Entrada:

```
13 // Quantidade de entradas de GPS.
22.00517 47.891024
22.005123 47.89104467
22.00508983 47.89108183
22.0050745 47.89113433
22.00507367 47.89119983
22.00508117 47.8912795
22.00509033 47.89136533
22.00509717 47.89145033
22.00510117 47.8915275
22.00510367 47.891594
22.00510417 47.89165233
22.00510483 47.891706
22.00510617 47.8917525
2 // Modo de operação de simplificação e impressão do trajeto.
20.0 // Distância máxima entre nós.
```

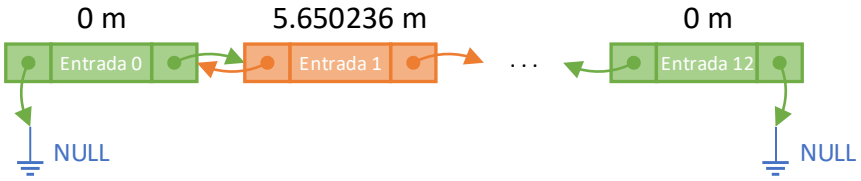
Saída:

```
22.005170 47.891024 0 0.000000 0.000000
22.005074 47.891134 3 5.680899 16.655015
22.005081 47.891280 5 8.265013 31.680910
22.005097 47.891450 7 8.805825 49.403738
22.005104 47.891594 9 6.869100 64.249984
22.005106 47.891753 12 4.801579 80.611818
```

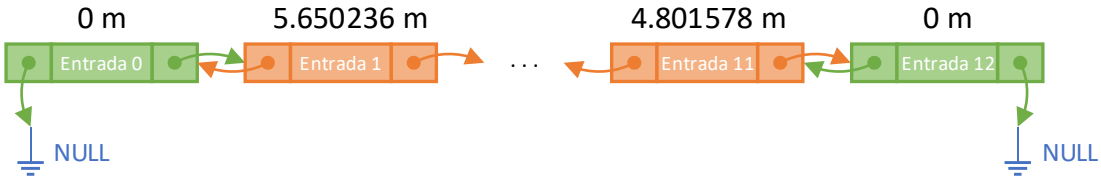
Tendo a estrutura do primeiro exemplo, é iniciado um passo do processo de simplificação. O passo é feito simultaneamente pelos dois lados, sendo sincronizados pela distância percorrida até então.



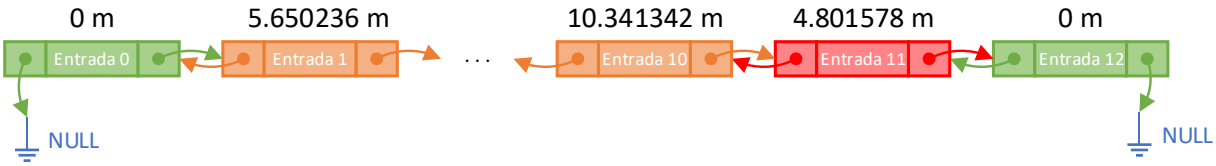
Como os dois lados possuem um empate de distâncias percorridas, inicia-se o processo pelo lado esquerdo.



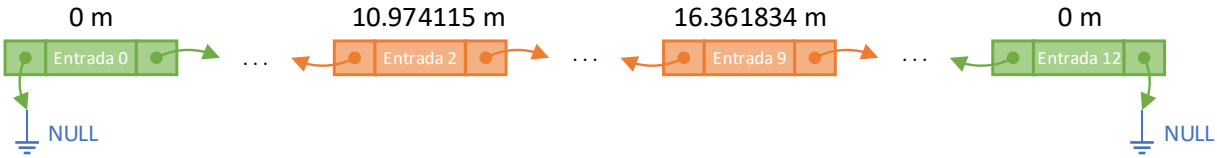
Agora, o lado direito possui menor distância (0 m < 5.650236 m).

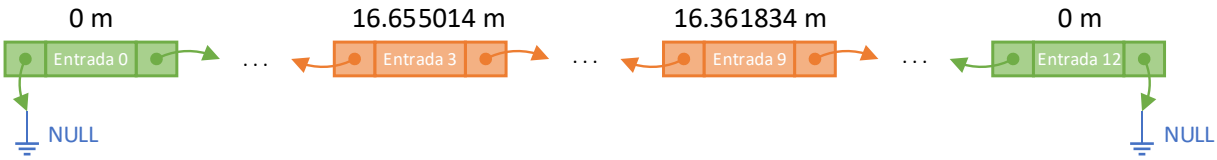


O lado direito ainda possui menor distância (5.650236 m > 4.801578).

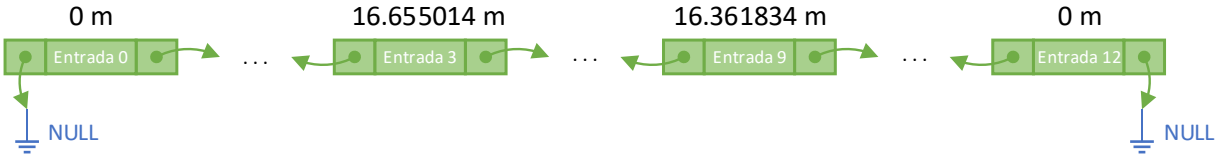


Continuando...

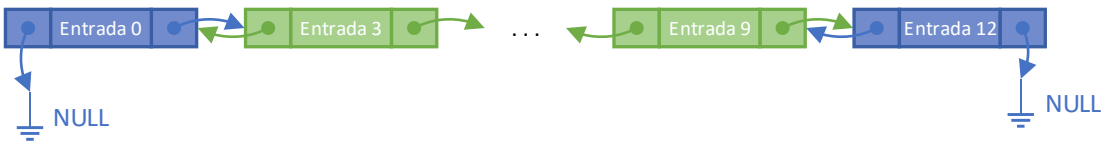




O passo do algoritmo é interrompido aqui, já que a distância limite é de 20 metros.

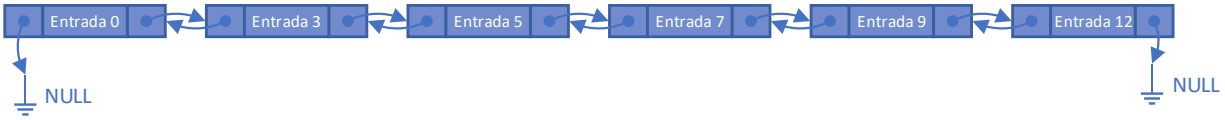


Aqui, são removidos os nós “Entrada 1”, “Entrada 2”, “Entrada 10” e “Entrada 11”, ...



... dando início a um novo passo do algoritmo, mas como entradas os nós “Entrada 3” e “Entrada 9”.

Como a distância entre os dois nós atuais é menor que 20 metros, são excluídos todos os nós entre eles, finalizando o algoritmo.



Exemplo:

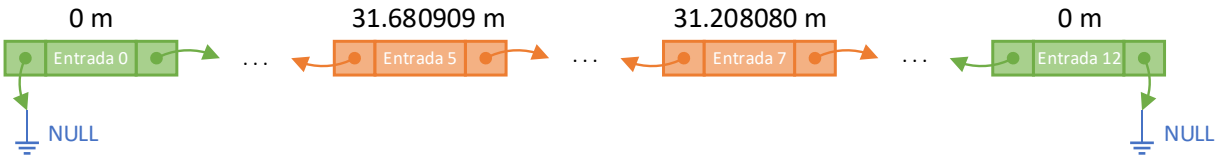
Entrada:

```
13 // Quantidade de entradas de GPS.
22.00517 47.891024
22.005123 47.89104467
22.00508983 47.89108183
22.0050745 47.89113433
22.00507367 47.89119983
22.00508117 47.8912795
22.00509033 47.89136533
22.00509717 47.89145033
22.00510117 47.8915275
22.00510367 47.891594
22.00510417 47.89165233
22.00510483 47.891706
22.00510617 47.8917525
2 // Modo de operação de simplificação e impressão do trajeto.
60.0 // Distância máxima entre nós.
```

Saída:

```
22.005170 47.891024 0 0.000000 0.000000
22.005090 47.891365 6 8.917003 40.597912
22.005106 47.891753 12 4.801579 80.611818
```

Neste caso, acontecerá um "encontrão" dos dois lados no nó "Entrada 6", durante a execução do algoritmo.



Os "encontrões" deverão ser tratados da seguinte forma: os nós extremos e o nó "encontrão" permanecerão na lista e os intermediários serão removidos.