Instituto de Computação da UNICAMP

Disciplina MC102: Algoritmos e Programação de Computadores - Turmas EF

Primeiro Semestre de 2015

Laboratório Nº 15. Peso 2.

Prazo de entrega: **05/06/2015 às 23:59:59**

PROFESSOR: Alexandre Xavier Falcão
MONITORES: João do Monte Gomes Duarte
MONITORES: Jadisha Yarif Ramírez Cornejo
MONITORES: Takeo Akabane
MONITORES: Eduardo Spagnol Rossi
MONITORES: Guilherme Augusto Sakai Yoshike

Navegação Submarina

Uma *startup* da área marítima iniciou a construção de protótipos de submarinos que serão utilizados na exploração do fundo do mar. Com o objetivo de utilizar os recursos – que são limitados – da maneira mais eficiente possível, os projetistas decidiram avaliar o desempenho do primeiro protótipo em uma região marítima de cartografia conhecida. Esta região marítima é mapeada em `L` pontos que definem segmentos de reta representando a profundidade do fundo desse trecho de região marítima.

Na fase de testes, os projetistas desejam avaliar a capacidade do protótipo em cumprir uma determinada rota, que pode assumir quatro direções: para **direita**, para **esquerda**, para **cima** e para **baixo**. O protótipo tem avaliação positiva se percorrer a rota adequadamente.

Para evitar qualquer tipo de colisão com o fundo do mar e, assim, poupar recursos, os projetistas concluíram que seria conveniente o auxílio de um programa capaz de calcular a **posição** do submarino e verificar possíveis **colisões** do submarino com o fundo do mar durante o percusso de uma rota.

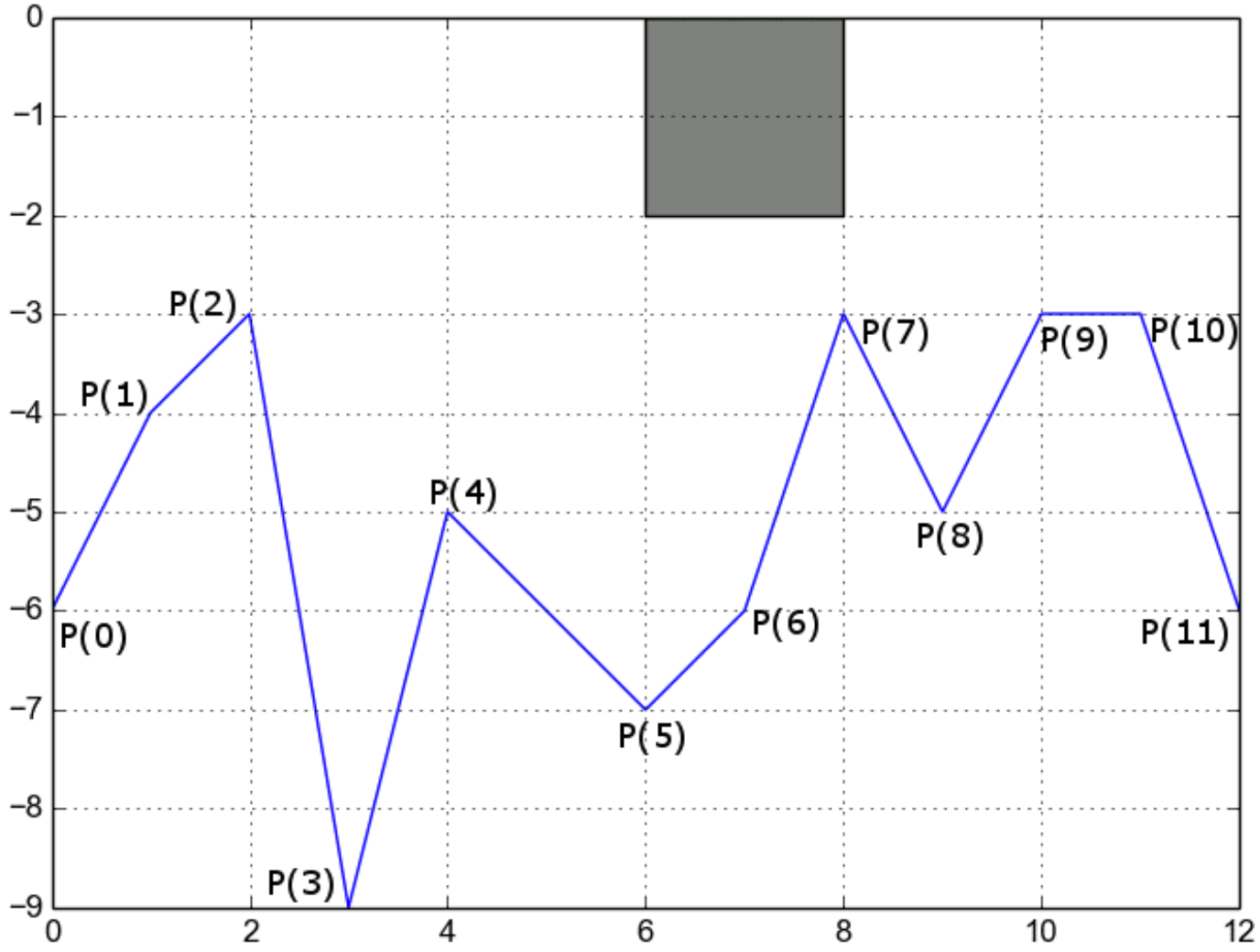
Neste exercício, você ajudará os projetistas no desenvolvimento de um programa que simule a navegação do protótipo do submarino em uma região marítima conhecida. Para isso, você deve considerar as seguintes especificações:

1. O prótotipo do submarino é **retangular**, com comprimento `W` e altura `H`, a altura e comprimento serão sempre maiores ou iguais a 2.
2. A região marítima de testes tem profundidade conhecida em cada ponto de sua extensão. A profundidade no ínicio da região é igual a `-3` vezes a altura do submarino (`P(0) = (0,-3H)`), de modo que o protótipo possa navegar perto da região costeira totalmente submerso e de maneira segura (sem o risco de colisão com o fundo do mar).
3. Todos os valores de **profundidade** passados ao programa são expressados como valores **negativos**.
4. A fundura entre duas profundidades conhecidas e consecutivas definidas pelos pontos `P(i)` e `P(i+1)` varia **linearmente**, ou seja, os pontos `P(i)` e `P(i+1)` definem um segmento de

reta.

5. O protótipo sempre parte da região **esquerda** do mar. A partir daí, o submarino cumpre uma rota que indica se ele deve ir à direita, descer, subir ou ir à esquerda. Cada movimento do submarino é caracterizado por um passo igual à `1`, em qualquer direção.
6. A posição de referência do submarino é seu **ponto inferior direito**.

A figura abaixo ilustra o cenário descrito no **exemplo de execução #3**. Pode-se notar que o submarino tem extensão igual a `2` e altura igual a `2`, e seu ponto referencial é `(2,-2)`; a região marítima tem `L=11` pontos definindo a sua profundidade, além do ponto inicial `P(0)=(0,-6)`.



Para os projetistas, o programa será de grande utilidade se ele puder gerar uma resposta que indique qual posição o submarino parou depois de percorrer inteiramente uma rota, ou se puder determinar se o submarino colidiu com o fundo do mar, durante o percurso, indicando em que ponto exato ocorreu a colisão e a quantidade de passos dada até o momento dela.

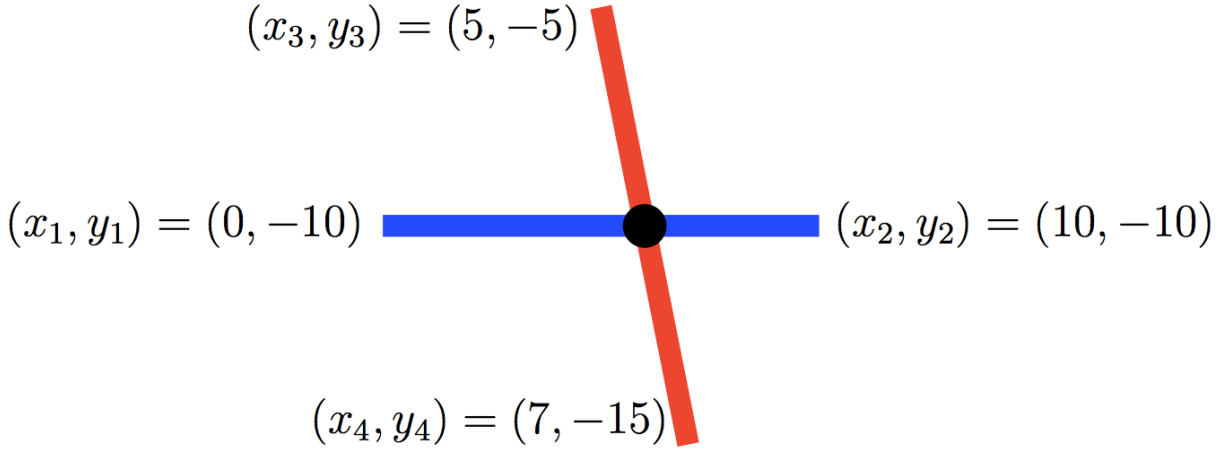
Para atingir esse objetivo, torna-se clara a necessidade de se realizar testes de interseção entre segmentos de reta. Podemos determinar se dois segmentos de reta tem interseção pelo seguinte cálculo (mais detalhes aqui): dado dois segmentos de retas P delimitado pelos pontos (x_1, y_1), (x_2, y_2) e Q delimitado pelos pontos (x_3, y_3), (x_4, y_4), as retas definidas por P e Q **não** terão interseção caso o determinante $D = (x_4 - x_3)(y_2 - y_1) - (x_2 - x_1)(y_4 - y_3)$ seja igual a zero. Caso contrário, existirá interseção entre os segmentos de reta se:

- $T_1 = \dfrac{(x_4 - x_3)(y_3 - y_1) - (x_3 - x_1)(y_4 - y_3)}{D}$ esteja entre 0 e 1, **e**
- $T_2 = \dfrac{(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)}{D}$ também esteja entre 0 e 1.

Para achar o ponto de interseção, basta substituir T_1 na equação paramétrica do segmento P, resultando no ponto

$$(x_1 + T_1(x_2 - x_1), y_1 + T_1(y_2 - y_1)) \ .$$

Note porém que estamos lidando com um caso especial, em que um dos segmentos é paralelo ao eixo x. Com essa informação é possível simplificar consideravelmente tais expressões. Abaixo é dado um exemplo de cálculo de interseção.



Para o exemplo acima, D será $(7-5)(-10+10) - (10-0)(-15+5) = 100$, T_1 será $((7-5)(-5+10) - (5-0)(-15+5))/100 = 60/100 = 0.6$ e T_2 será $((10-0)(-5+10) - (5-0)(-10+10))/100 = 50/100 = 0.5$. O ponto de interseção será então $(0 + 0.6(10-0), -10 + 0.6(-10+10)) = (6, -10)$.

Considere, então, as especificações de entrada e saída abaixo:

Entrada: duas linhas com valores inteiros e uma linha de caracteres, sendo que os valores e os caracteres devem estar separados por um espaço em branco entre si.

- Na primeira linha são dados `L`, no intervalo (fechado) [2, 50], `W` e `H`, no intervalo (fechado) [2,25].
- Na segunda linha, são dados `2 x L` valores correspondentes aos pontos P(1), P(2), ..., P(L), no formato $P(1)_x$ $P(1)_y$ $P(2)_x$ $P(2)_y$ ... $P(L)_x$ $P(L)_y$, onde $P(i)_x$ é a distância até a margem esquerda em metros no intervalo (fechado) [0,100] e $P(i)_y$ a profundidade da região marítima para o ponto P(i) no intervalo (fechado) [-100,0] (lembre-se que o ponto P(0) tem profundidade igual à `3` vezes a altura do submarino).
- Na terceira linha, são dados `15` caracteres correspondentes à rota que o submarino deverá percorrer. Leve em consideração a seguinte convenção: para direita (`d`), para esquerda (`e`), para baixo (`b`) e para cima (`c`). Cada um dos `15` caracteres indica que o submarino deve se deslocar `1` passo na direção determinada pelo caractere correspondente.

Saída: imprimir uma das seguintes mensagens:

- Caso ele percorra toda rota, imprimir "`Alcancou a posicao (x,y).`", sendo **x** e **y** a posição de parada do submarino, com precisão de 2 casas decimais no seu ponto referência (ponto inferior direito do submarino).
- Se o submarino colidir com o fundo do mar, imprimir "`Colidiu em (x,y) no passo z.`" sendo **x** e **y** o ponto mais próximo a margem esquerda em que houve colisão com a parte **inferior** do submarino, com precisão de 2 casas decimais, e **z** denota a quantidade de passos efetuada pelo submarino até o momento da colisão.

Implementação

Para esse laboratório, você obrigatoriamente deve definir uma **estrutura** para o conceito de Ponto. Além dessa estrutura, você **deve** criar uma função **`intersecao`** que verifica se existe colisão entre o submarino e um segmento de reta definido por dois pontos. Essa função deve retornar 1 caso exista colisão e 0 caso contrário. Além disso, ela deve receber como parametro um ponto `v` **passado por referência** que deve ser preenchido com o ponto de interseção, caso exista.

Note que apesar de a única estrutura obrigatória no seu código ser a estrutura Ponto, você pode e é encorajado a definir outras estruturas que possam auxiliá-lo nessa tarefa.

Observações

- IMPORTANTE: em caso de colisão, esta se dará sempre com o casco inferior do submarino. Desta forma basta testar interseções com o segmento de reta definido pelo canto inferior esquerdo e canto inferior direito do navio. O ponto a ser impresso no caso de colisão deverá ser sempre o ponto de interseção do segmento de reta inferior do submarino com o segmento de reta do fundo do mar mais próximo da margem esquerda.
- IMPORTANTE: a coordenada x de um ponto `P(i)` é sempre menor que a coordenada do ponto seguinte, ou seja, sempre vale que $P(i)_x < P(i+1)_x$.
- IMPORTANTE: os vértices que delineiam o retângulo do submarino nunca pertencerão a nenhum dos segmentos de retas de `P(i)` até `P(i+1)` (incluindo os pontos `P(i)` e `P(i+1)`), para todo `i` em `[0,L-1]`.
- Você **deve** quebrar a linha após imprimir o resultado.
- Ao usar a função `scanf` para ler um caractere (com a máscara `%c`), deve-se acrescentar um espaço antes de `%c` para garantir que somente um caracter válido seja lido (ou seja, para ignorar espaços e quebras de linhas).
  - Por exemplo, para um `char x`, o comando `scanf("%c",&x);` assinalaria para `x` o caractere de espaço `' '` para a entrada `" A "` (note o espaço na entrada!) enquanto que `scanf(" %c",&x);` (note o espaço antes de `%c`) assinalaria `'A'` para a variável `x`.
- Dica: você pode utilizar vetores de pontos para o armazenamento das profundidades.
- Dica: para cada entrada aberta deste laboratório, existe uma animação correspondente na pasta **aux**.

Exemplos de execução

Exemplo 1 (veja figura ilustrativa aqui):

```
5 2 2
1 -12 2 -10 3 -12 4 -6 6 -6
d b d b e c d d d c e b e b e
```

```
Alcancou a posicao (3.00,-4.00).
```

Exemplo 2 (veja figura ilustrativa aqui):

```
11 2 2
1 -8 2 -9 3 -11 4 -2 5 -8 6 -9 7 -17 8 -4 9 -7 10 -4 12 -6
d b b b b d d d d c d d e e e
Colidiu em (3.56,-6.00) no passo 6.
```

Exemplo 3 (veja figura ilustrativa aqui):

```
11 2 2
1 -4 2 -3 3 -9 4 -5 6 -7 7 -6 8 -3 9 -5 10 -3 11 -3 12 -6
d d d b d d c d d d e e e e b
Alcancou a posicao (6.00,-3.00).
```

Exemplo 4 (veja figura ilustrativa aqui):

```
10 3 2
2 -15 3 -7 4 -13 5 -14 8 -14 10 -12 11 -4 12 -5 14 -5 15 -6
d d d d b b b b b b b b b b e
Colidiu em (3.83,-12.00) no passo 15.
```

Exemplo 5 (veja figura ilustrativa aqui):

```
10 3 2
2 -15 3 -6 4 -13 5 -14 8 -14 10 -12 11 -4 12 -5 14 -5 15 -6
d d d d b b b b b b b b b b d
Alcancou a posicao (8.00,-12.00).
```

Notas: Textos em `azul` denotam dados de entrada do programa.
Textos em `vermelho` denotam dados de saída do programa.

Observações gerais:

- O número máximo de submissões é 15;
- O seu programa deve estar completamente contido em um único arquivo denominado `submarino.c`;
- Para a realização dos testes automáticos, a compilação se dará da seguinte forma: `gcc submarino.c -o submarino -Wall -std=c99 -pedantic`;
- Não se esqueça de incluir no início do programa uma breve descrição dos objetivos, da entrada, da saída, seu nome e RA;
- Após cada submissão, você deve aguardar um minuto até poder submeter seu trabalho novamente.

Criterios importantes:

Independentemente dos resultados dos testes do SuSy, o não cumprimento dos critérios abaixo

implicará nota zero nesta tarefa de laboratório.

- O único header aceito para inclusão é stdio.h.
- Você deve necessariamente especificar e implementar as funções e estruturas requisitadas na seção **Implementação**.
