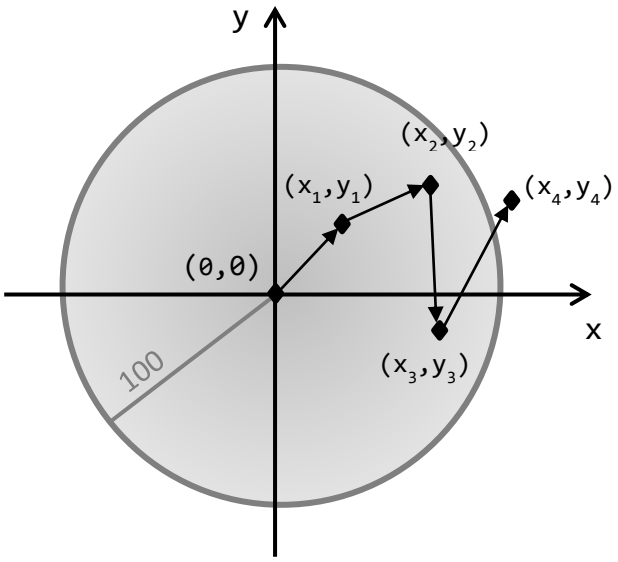
DEPARTAMENTO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMAÇÃO DE COMPUTADORES

TRABALHO PRÁTICO 2

Enquanto estudava Geometria Analítica e Álgebra Linear, Joãozinho se deu conta de que os vetores podem ser usados para guiar o deslocamento de objetos. Em programação de computadores, Joãozinho aprendeu que o computador é uma ferramenta capaz de resolver problemas que exigem muitas repetições de uma determinada tarefa. Ao fazer seu lanche na cantina, Joãozinho ficou intrigado com o movimento de uma mosca, que insistia em voar ao seu redor. Foi aí que ele teve a ideia de juntar tudo que aprendeu para construir uma simulação do voo da mosca. Ajude Joãozinho nesta tarefa.

Considere que as moscas sempre partem da posição (0,0) do eixo cartesiano e que elas se deslocam usando um movimento aleatório, similar ao representado na imagem abaixo. A cada passo da simulação a mosca escolhe uma direção e percorre essa direção por um tempo. Esse deslocamento pode ser representado por um vetor. A direção da mosca é o ângulo do vetor com o eixo x e o seu tempo de deslocamento é dado pela magnitude desse vetor.



Joãozinho quer saber quantos passos de simulação, em média, serão necessários para 10 moscas que tenham saído da posição (0,0) percorrerem uma distância maior ou igual a 100 centímetros.

Representação da Mosca

Cada mosca será identificada por um nome e uma cor RGB. Esses dados devem ser lidos no início do programa e **armazenados em um vetor estático** de tamanho 10, como mostrado ao lado.

```
Turing     230 101 049
Lovelace   220 076 138
Knuth      210 063 019
Naur       151 121 240
Thompson   000 199 252
Neumann    100 200 089
Ritchie    058 125 090
Stroustrup 255 170 000
Kernighan  094 048 235
Lamport    000 097 255
```

Para trabalhar com a mosca, construa uma biblioteca (Mosca.h e Mosca.cpp):

- Represente a **Cor** por um registro que guarde os valores:
  - Red (inteiro)
  - Green (inteiro)
  - Blue (inteiro)

- Construa uma função **operator>>** para ler valores do tipo Cor
- Construa uma função **operator<<** que modifica a cor do texto no terminal

- Represente uma posição no eixo cartesiano por um registro **Ponto**:
  - x (ponto flutuante)
  - y (ponto flutuante)

- Represente um **Vetor** por um registro com dois valores:
  - Magnitude (ponto flutuante)
  - Ângulo (ponto flutuante)

- Represente a **Mosca** por um registro que guarde pelo menos:
  - Nome (vetor de 20 caracteres)
  - Posição atual no eixo x e y (tipo Ponto)
  - Cor (tipo Cor)

- Construa uma função **operator>>** para ler valores do tipo Mosca
- Construa uma função **operator<<** para exibir valores do tipo Mosca
  - O nome da mosca deve ser exibido na sua cor
    (use o operator<< construído para o tipo Cor)
  - Lembre-se de retornar a cor do terminal para a cor padrão

Representação da Cor

O ANSI TrueColor é um padrão para exibir cores RGB no terminal. A maioria dos terminais modernos no Linux, Windows e MacOS possuem suporte. A cor do terminal é modificada enviando para o `cout` um código:

```
"\x1b[38;2;R;G;Bm"  // R, G e B são números na faixa 0 a 255
"\x1b[0m"           // retorna para a cor padrão
```

Por exemplo:

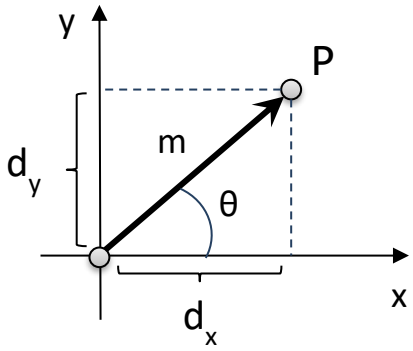
```
cout << "\x1b[38;2;230;101;49m";   // cor R=230, G=101 e B=49
cout << "Turing";                  // exibe texto colorido
cout << "\x1b[0m";                 // retorna para a cor padrão

cout << "\x1b[38;2;220;76;138m";   // cor R=220, G=79 e B=138
cout << "Lovelace";                // exibe texto colorido
cout << "\x1b[0m";                 // retorna para a cor padrão
```

Representação do Vetor

O vetor deslocamento da mosca pode ser representado por um ângulo e uma magnitude (coordenadas polares) ou por um ponto no plano (coordenadas cartesianas). Se quisermos deslocar a mosca da origem dos eixos até uma posição P, basta somar a posição atual (x, y) da mosca com os valores d_x e d_y, sendo estes dados por:

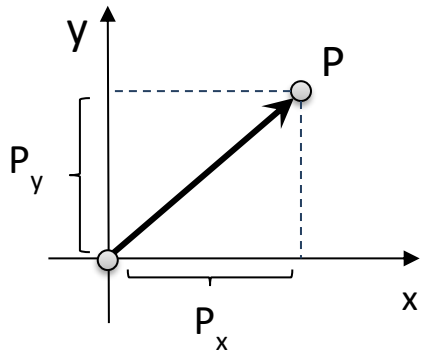
Coordenadas Polares



$$d_x = m * \cos\theta$$

$$d_y = m * \text{sen}\theta$$

Coordenadas Cartesianas



$$d_x = P_x$$

$$d_y = P_y$$

Simulação do Voo

A cada passo da simulação, um número aleatório entre 1 e 10 deve ser sorteado para definir a quantidade de moscas que deverão ser movidas.

- O número deve ser usado para criar um vetor dinâmico de ponteiros para moscas. Depois deve-se sortear que moscas serão movidas. Os endereços das moscas sorteadas devem ser guardados no vetor dinâmico.
- Utilize um laço para percorrer o vetor dinâmico, acessar e movimentar as moscas apontadas. Cada mosca deve ser movida por um vetor aleatório diferente. Implemente uma **função Deslocar** que receba a posição atual da mosca (Ponto) e um Vetor e retorne a nova posição da mosca. Apenas as moscas que ainda não atingiram a distância alvo devem ser deslocadas.

O programa deve exibir o número do passo da simulação, quantas moscas foram sorteadas, quais posições foram sorteadas, e quais moscas atingiram a marca de 100 cm naquele passo. Se nenhuma mosca atingiu a marca, deve-se exibir zero, como mostrado no exemplo abaixo:

```
Simulação
---------
#1: 5 ( 0 2 7 4 8 ) = 0
#2: 2 ( 9 2 ) = 0
#3: 4 ( 0 8 3 7 ) =  Knuth
#4: 8 ( 2 4 1 0 3 5 9 7 ) = Naur Turing
...
```

Use uma linha para cada passo da simulação. Exiba jogo da velha, dois pontos, parênteses, sinal de igual e espaços, exatamente como no exemplo.

Quando uma das moscas atingir a marca de 100 cm, ela deve ser marcada como inativa. Uma mosca inativa não deve ter sua posição atualizada, nem o seu nome exibido nos passos de simulação. Se no sorteio, uma mosca inativa for escolhida deve ser exibido um "x" vermelho no lugar do número sorteado para indicar que uma mosca inativa foi sorteada:

```
...
#50: 4 ( 1 x 6 3 9 ) = 0
#51: 3 ( 1 2 ) = 0
#52: 4 ( x 8 3 7 ) = Tompson
#53: 2 ( x x ) = 0
...
```

A simulação deve parar quando todas as 10 moscas atingirem os 100 cm de deslocamento. Para auxiliar nesta tarefa, crie uma **função Inativas** que receba o vetor de moscas e o tamanho do vetor e retorne verdadeiro apenas quando todas as moscas do vetor estiverem inativas. Utilize o resultado dessa função para decidir quando parar a simulação.

A forma mais simples de saber se a mosca atingiu os 100 cm de distância em relação a origem é converter a sua posição atual em coordenadas polares e olhar a magnitude do vetor resultante. Se esta magnitude for maior ou igual a 100, a mosca deve ser colocada no estado inativo. Para auxiliar nesta tarefa, crie uma **função Magnitude** que receba um Ponto no plano cartesiano e retorne um valor ponto flutuante representando a magnitude do vetor em coordenadas polares.

Ao final da simulação, exiba o resultado como no exemplo abaixo:

```
Resultados
----------------------
Turing   = 182
Lovelace   = 208
Knuth   = 880
Naur   = 405
Thompson   = 439
Neumann   = 286
Ritchie   = 1025
Stroustrup   = 131
Kernighan   = 493
Lamport   = 238
----------------------
Total de passos: 1025
Em média as moscas levaram 428.7 passos
A mosca mais rápida foi Stroustrup com 131 passos
O que equivale a voar 0.763359 cm por passo.
```

Observe que as cores usadas no resumo final são as cores da mosca mais rápida.

EXIGÊNCIAS

- Não escreva mensagens solicitando a entrada de dados
- Não force a entrada em laços de repetição
- Não use variáveis globais
- Não use o tipo string
- Não use vector
- A solução do problema deve utilizar as funções que foram pedidas
- Use comentários, indentação e deixe o código organizado
- Libere toda a memória alocada dinamicamente

ENTREGA DO TRABALHO

Grupos: Trabalho individual

Data da entrega: 25/04/2023 (até a meia-noite)

Valor do Trabalho: 3,0 pontos (na 2a Unidade)

Forma de entrega: enviar apenas os arquivos fonte (.cpp) e os arquivos de inclusão (.h) compactados no formato **zip** através da tarefa correspondente no SIGAA.

O não cumprimento das orientações resultará em **penalidades:**

- Programa não executa no Visual Studio 2022 (3,0 pontos)
- Programa contém partes de outros trabalhos (3,0 pontos)
- Atraso na entrega (1,5 pontos por dia de atraso)
- Arquivo compactado em outro formato que não zip (0,5 ponto)
- Envio de outros arquivos que não sejam os .cpp e .h (0,5 ponto)
- Programa sem comentários e/ou desorganizado (0,5 ponto)