Lista de Exercícios (Programação em C++)

1. Corrija o programa a seguir:

```cpp
#include <iostream>
class Math{
public:
  static double media_aritmetica(int a, int b){
    return a + b / 2;
  }
};
int main(){
  cout << Math::media_aritmetica(2, 1);
}
```

2. Desenvolva uma classe em C++ para simular operações de depósito, saque, atualização de rendimentos e apresentação de saldos em uma poupança. Sua classe deverá funcionar com o programa-exemplo mostrado a seguir:

```c
int main(void){
  ContaCorrente c; // inicializa a conta corrente
  c.deposita(300); // deposita 300 dinheiros
  c.rendimento();  // reajusta o valor do deposito em mais 5% do valor atual
  c.saca(200);     // realiza um saque de 200 dinheiros
  c.saldo();       // imprime o saldo da poupança na tela
}
```

3. Qual a saída do programa a seguir? Explique o porquê.

```cpp
#include <iostream>

int x = 5;

int &f() {
  return x;
}
main() {
  f() = 10;
  std::cout<<x;
}
```

4. Qual a saída do programa a seguir? Explique o porquê.

```cpp
#include <iostream>
main() {
   const int x = 4;
   x++;
   std::cout << x;
}
```

5. Considere o programa a seguir:

```cpp
```

```
#include <iostream>
using namespace std;
class Banco {
public:
  int valor;
  Banco( int quantidade ) {
    valor = quantidade;
    cout << "Valor " << valor << endl;
  }
  ~Banco() {
    cout << "Voce matou: " << valor << endl;
  }
};
class Cliente {
public :
  Banco dado;
  Cliente(int A) : dado(A) {
     cout << "Novo Objeto\n";
  }
  Cliente( const Cliente& X ) : dado(X.dado.valor + 10) {
     cout << "Caiu aqui\n";
  }
};

void Desafio(Cliente porque){
  Cliente Jaco(porque);
  cout << "Depois de Jaco\n";
}

int main(){
  Cliente Resposta(1);
  cout << "Chama Desafio\n";
  Desafio(Resposta);
  cout << "terminou..." << endl;
}
```

Este programa apresenta, quando executado, as seguintes informações na saída padrão:

```
Valor 1
Novo Objeto
Chama Desafio
Valor 11
Especialidade da casa
Valor 21
Especialidade da casa
Depois de Jaco
Voce matou: 21
Voce matou: 11
terminou...
Voce matou: 1
```

Justifique, para cada linha que aparece na saída do programa, nos moldes previstos pela linguagem C++, o motivo de sua existência naquela exata posição. Siga o seguinte modelo:

> Valor 1: Aparece porque...
>
> Novo Objeto: Aparece porque ...
>
> Chama Desafio: Aparece porque...

6. Um cartunista surrealista resolveu criar um pequeno programa de computador para fazer combinações de cores, assumindo cada cor como um indivíduo capaz de reproduzir e criar novas cores. Cada cor é uma combinação de

quantidades de Vermelho(red), Verde(green) e Azul(blue). Seu programa permitia juntar, separar e comparar cores e entendia cada cor como um objeto de uma classe. Veja o protótipo da classe que ele criou:

CPP

```cpp
class Color{
  float r,g,b; // cores no intervalo 0-255
public:
  // construtor da classe
  // guarda o estado inicial do objeto
  Color(float _r=0, float _g=0, float _b=0);

  // retorna um novo individuo de cor igual aa media
  // da sua cor com a cor passada como parametro
  Color operator+(Color c);

  // se for fornecido um dos pais do individuo
  // a funcao separa a cor do outro parental
  Color operator-(Color c);

  // retorna ``1'' se o individuo c possui cor igual
  // ao proprio objeto, e ``0'', caso contrario
  int operator==(Color c);

  // Mostra os valores de vermelho, verde e azul da cor.
  void print();
};
```

Usando sua classe, o cartunista preparou o trecho de código abaixo transcrito. Neste código, dois indivíduos `a` e `b` são criados e combinados para produzir um novo indivíduo `c`. O cartunista resolve fazer um *teste de DNA* para saber se `b` é pai de `c`, criando um novo indivíduo `d` para usá-lo no teste!

CPP

```cpp
int main(void){
  Color a(255,0,0), b(0,120, 121), c, d;
  c = a+b;
  d = c-a;
  d.print();
  if(d == b){
    cout << "Fail! DNA positivo: b eh pai de c!\n";
  }
  else{
    cout << "Ufa! que alivio!";
  }
}
```

Sua tarefa é implementar a classe `Color` usando o protótipo acima e testá-la com a função `main()` fornecida!

7. Prepare um projeto em C++ capaz de armazenar e realizar operações com polígonos formados por conjuntos de pontos em duas dimensões.

   Deve ser previsto no projeto a criação de três classes:

   - `Ponto`
   - `Poligono`
   - `Retangulo`

   a. Crie uma classe denominada `Point` para representar pontos no espaço bidimensional. Na sua implementação, você deverá encapsular duas variáveis x e y do tipo float para guardar a posição do ponto. Apenas as funções da classe poderão ter acesso direto a essas variávies, de modo que os clientes da classe somente poderão modificá-las usando métodos específicos que você definir. Implemente, na sua classe, métodos que realizem as seguintes operações:

Função	Descrição
`setX(float)`	Define o valor da coordenada $x$do ponto.
`setY(float)`	Define o valor da coordenada y do ponto.
`setXY(float,float)`	Define, em uma mesma função, os valores da coordenadas x e y do ponto.
`float getX()`	Recupera o valor da coordenada x do ponto.
`float getY()`	Recupera o valor da coordenada y do ponto.
`add(Point p1)`	Adiciona as coordenadas (x, y) do ponto com as coordenadas de um ponto $P1(x_1, y_1)$ fornecido, armazenando o resultado $(x + x_1, y + y_1)$ nas coordenadas de um novo ponto, que deverá ser retornado para o cliente da classe.
`sub(Point p1)`	Subtrai as coordenadas (x, y) do ponto com as coordenadas de um ponto $P1(x_1, y_1)$ fornecido, armazenando o resultado $(x - x_1, y - y_1)$ nas coordenadas de um novo ponto, que deverá ser retornado para o cliente da classe.
`norma()`	Calcula a distância do ponto para a origem do sistema de coordenadas.
`translada(float a, float b)`	Translada o ponto (x, y) de $(+a, +b)$, de modo que, após a execução do método, as coordenadas do ponto serão $(x + a, y + b)$.
`imprime()`	Imprime o ponto na forma `(xpos, ypos)`.

b. Defina uma classe chamada `Poligono` para representar polígonos convexos. Assuma que o tamanho dos polígonos será limitado a 100 vértices. Utilize a classe `Point` que você definiu na etapa anterior para guardar informações com as posições dos N vértices do polígono. Sua classe deverá prever as seguintes funcionalidades:

- Inserir vértice no polígono. Assuma que os vértices deverão ser inseridos conforme a sequência (anti-horária) em que figuram ao redor do polígono. As arestas do polígono serão então compostas pelos segmentos $(x_0, y_0) \rightarrow (x_1, y_1), (x_1, y_1) \rightarrow (x_2, y_2)$ etc., com exceção da última aresta, que será formada pelo segmento $(x_{N-1}, y_{N-1}) \rightarrow (x_0, y_0)$.
- Recuperar a quantidade de vértices que foram inseridos no polígono
- Calcular a área do polígono. Procure na Internet a fundamentação matemática necessária para implementar essa funcionalidade.
- Transladar o polígono para $(+a, +b)$ usando uma função `translada(float a, float b)`.
- Rotacionar o polígono de θ graus no sentido anti-horário em torno de um ponto (x_0, y_0) fornecido pelo usuário.
- Imprimir o polígono armazenado da forma `(x0,y0)→(x1,y1)→(x2,y2)→…`

c. Utilizando a implementação da classe `Poligono` desenvolvida na etapa anterior, crie uma subclasse `Retangulo`

derivada da superclasse `Poligono`. O construtor da nova classe deverá ser da forma `Retangulo(float x, float y, float largura, float altura)`, denotando a posição (x, y) do retângulo no espaço (coordenadas do canto superior esquerdo) e suas dimensões - altura e largura. Realize com esta classe as seguintes tarefas:

- Assegure-se de que o construtor da classe utiliza os métodos da superclasse para armazenar a estrutura interna do retângulo.

d. Crie um pequeno exemplo para testar sua implementação da classe `Retangulo`.

- No exemplo, defina um novo retângulo na posição $(x, y) = (0, 0)$, com altura e largura iguais a `3` e `4`, respectivamente.
- Imprima a estrutura poligonal gerada para o retângulo.
- Calcule a sua área usando a função já implementada na classe `Poligono` e mostre o resultado.
- Mude a posição do retângulo usando a função `translada(float,float)` para $(x, y) = (-3, 4)$ e recalcule a área do retângulo. Compare-a com a área calculada antes da transformação geométrica.
- Rotacione o Retângulo de $+30^o$ em relação ao seu centro de massa e recalcule a sua área. Compare-a com a área calculada antes da transformação geométrica.

Last updated 2023-09-05 09:40:01 -0300