Programação Orientada a Objetos 2020/2021

Exame da Época Normal

DEIS – LEI / LEI-PL / LEI-CE

Qual será a saída resultante da execução deste programa?

```
47
48  class Erro {
49  public:
50      virtual string what()const {
51          return "granito; ";
52      }
53  };
54  class ErroDeriv : public Erro {
55  public:
56      string what()const override {
57          return "calcario; ";
58      }
59  };
60  void testar() {
61      throw ErroDeriv();
62      cout << "xisto; ";
63  }
64
65  int main() {
66      try {
67          testar();
68      }
69      catch (Erro & e) {
70          cout << e.what();
71      }
72      catch (string e) {
73          cout << "marmore; ";
74      }
75      cout << "fim; \n";
76  };
```

Qual será a saída resultante da execução deste programa?

```
264  class Item {
265      string msg;
266  public:
267      Item(const string & s) :msg(s) {
268          cout << "Constr " << msg << "; " << endl;
269      }
270      ~Item() {
271          cout << "Destr " << msg << "; " << endl;
272      }
273  };
274
275  void f() {
276      Item * p = new Item("Girassol");
277  }
278  void g() {
279      unique_ptr<Item> a = make_unique<Item>("Magnolia");
280  }
281  int main() {
282      f();
283      g();
284  }
285
```

Considere uma classe que armazena mensagens de texto (conjunto de palavras), que se consideram que pertencem ao objecto em que estão. Pretende-se que os objectos desta classe facilitem o seu uso, podendo ser usados da forma que vem exemplificada abaixo. O armazenamento do texto está sujeito a algumas regras tal como notado nos comentários do exemplo. Implemente a classe cumprindo os requisitos e fazendo com que o código abaixo funcione.

```
Texto a;  → fica sem palavra nenhuma
Texto b({“ola”, “isto”, “tem”, “piada”});  → fica com as 4 palavras indicadas.
                                  → É compatível com qualquer número de palavras.
a << “também” << “quero” << “quero”;  → a fica com “também” e “quero”
                                      → (não repete palavras)
if (a == b) { /* … */ } → a e b seriam iguais se tiverem as mesmas palavras
                        → (pode ser por ordem diferente)
b -= “tem”;  → b fica sem a palavra “tem”
cout << “b agora tem” << (int) b << “palavras”;  // b agora tem 3 palavras
a << b;   → transfere as palavras de b para a que ainda não existissem em a
          → (sendo removidas de b aquelas que foram transferidas)
```

Considere as seguintes classes:

```
class Data{
    int dia, mes, ano;
public:
    Data(int a, int b, int c):dia(a),mes(b),ano(c){}
};

class Fig2D{
    const Data criacao;
    string cor;
public:
    virtual float area() = 0;         // Cálculo da área da figura
    virtual float perimetro() = 0;   // Cálculo do perímetro da figura
    void mudaCor(string a) {cor=a;}
    Data getCriacao() const {return criacao;}
};
```

A classe Fig2D tem como objetivo armazenar informação sobre figuras geométricas com duas dimensões (quadrados, trapézios, triângulos, pentágonos, círculos, retângulos, entre outros). Os seus atributos permitem armazenar a cor da figura e a data de criação.

A classe Fig2D servirá de base para classes derivadas que representem figuras concretas. Ao responder às alíneas, tenha em consideração o seguinte:

i. Todos os objetos são criados indicando obrigatoriamente todos os seus atributos.
ii. Pode adicionar novos métodos às classes já escritas, mas não pode eliminar ou alterar o que já se encontra nas classes Data e Fig2D.
iii. Todas as classes do problema (as já disponibilizadas e as que vai escrever) devem poder ser utilizadas em todas as situações habituais.

a) Implemente 2 classes derivadas da classe Fig2D. Cada uma destas classes deve corresponder a uma figura geométrica concreta.

---

b) Exemplifique com código como poderia criar 2 objetos dinâmicos das classes derivadas implementadas na alínea a).

c) Crie uma classe Desenho que contenha como único atributo um contentor STL de figuras 2D (inicialmente vazio). Adicione os seguintes métodos a esta classe:

i. Adicionar uma nova figura ao Desenho, garantindo que as figuras ficam ordenadas no contentor de acordo com um determinado critério. O critério fica à sua escolha e deve referi-lo explicitamente, indicando igualmente como resolverá eventuais empates.
ii. Implementar uma operação polimórfica, à sua escolha, envolvendo todas as figuras armazenadas no contentor. Refira explicitamente qual o objetivo do método que implementar.
iii. Os objetos da classe Desenho tomam posse das figuras 2D que são armazenadas no contentor. Adicione o que for necessário à sua classe para garantir a correção do código.

Numa aplicação de controlo de eletrodomésticos, um eletrodoméstico de limpeza, além de um nome possui uma função limpa(). As ações específicas que têm de ser efetuadas pelo eletrodoméstico em resposta a esse comando variam consoante a sua função:

- a máquina de lavar a louça fecha-se e dá início à lavagem

- a máquina de a roupa, fecha-se, pesa a roupa que tem no tambor, e inicia o programa de lavagem adequado a esse peso de roupa

Além disso, o eletrodoméstico deve ser capaz de indicar se está ativo ou em descanso. Para criar um eletrodoméstico é sempre preciso indicar o seu nome;

- No caso da máquina de lavar a roupa, ela possui uma caraterítica que corresponde ao peso máximo de roupa que suporta, e deve permitir também o comando de enxaguar;

- e a máquina da louça deve obedecer ao comando de secagem rápida.

Existe ainda um assistente de pessoal (tipo Alexa) que mantém um vector que lhe permite controlar os eletrodomésticos, podendo receber solicitações do seu utilizador para, por exemplo, mandar um determinado eletrodoméstico, identificado pelo nome, limpar. Os eletrodomésticos são adicionados ao longo da existência do assistente. Os eletrodomésticos podem ser partilhados por vários assistentes. Cada assistente pessoal tem um nome.

Não deve ser possível criar uma cópia de um assistente pessoal.

Apresente a declaração das classes que representam este cenário (ficheiros .h). Não deve implementar função nenhuma – apresente apenas os ficheiros .h. Não precisa de especificar using namespaces nem includes excepto se forem includes e outras declarações de coisas suas.

A classe Bilhete representa um bilhete para um concerto. O ficheiro .h correspondente é o seguinte:

```
class Bilhete {
   // ... Não é relevante para a pergunta
public:
   Bilhete(string nomeEspectador, int numero); // é o único construtor
   // ... outras funções não são relevantes
};
```

Construa uma classe que permita a emissão (venda) de bilhetes. O nome que consta do bilhete é fornecido por quem usa a bilheteira; o número é obtido automaticamente a partir de uma sequência gerida pela bilheteira, sendo o número inicial 100. **Haverá vários objectos da bilheteira a fornecer bilhetes (para o mesmo concerto), mas não pode haver bilhetes com o mesmo número, nem pode haver duas pessoas com o mesmo nome no concerto**.

Faça a classe Bilheteira, tendo como base de partida o código abaixo. Pode acrescentar o que for preciso desde que implemente a função emiteBilhete, responsável por obter objectos Bilhete, respeitando os requisitos indicados acima.

Quanto à classe Bilhete, só deverá mexer nela no que considerar necessário para impedir situações que violem as regras descritas acima.

```
class Bilheteira {
   // ...
public:
   // ...
   /* … */ emiteBilhete(string nomeEspectador /* eventual etc */) { /* fazer */ }
};
```