Fundamentos de Orientação a Objetos
Prof. Dr. Edson Tavares de Camargo

Lista de Exercícios 2

1. Em código C, escreva um programa para trocar os valores duas variáveis inteiras usando ponteiros. Escreva o mesmo código em C++ e verifique se há compatibilidade. Após isso, faça o mesmo usando somente o operador de referência (&) como parâmetro da função. Descreva como comentário no código o que muda em termos de notação de ponteiro ao usar o operador &.

2. Cria uma classe chamada `NumeroImaginario`. Defina um construtor apropriado. Defina as operações de soma, subtração, multiplicação e divisão. Pesquise como realizar a sobrecarrega os operadores de (+), (-), (*) e (/). Exemplo:

```
NumImaginario n1(...);

NumImaginario n2(...);

cout << n1 + n2
```

3. Crie um classe `Fatorial` responsável por calcular o fatorial de um número.

   a) a classe possui um atributo fatorial, que armazena o valor a ser calculado;

   b) a classe deve ter um construtor padrão e um construtor sobrecarregado para receber um número. No construtor sobrecarregado use a lista de inicialização de construtor ao invés de inicializar os atributos no corpo do construtor.

   c) crie dois métodos calcular. O primeiro retorna o valor do fatorial presente no atributo da classe e o segundo recebe um valor que além de atualizar o valor do atributo, também retorna o valor desse fatorial. Por exemplo: `obj.fat()` e `obj.fat(5)`.

   d) crie um método exibir para apresentar o calculo do fatorial da seguinte forma: `obj.fat(5)` irá apresentar 5! x 4! x 3! x 2! x 1! = 120.

Observação: Sua classe deve seguir as definições de fatorial. Para saber mais sobre fatorial, veja o link: https://brasilescola.uol.com.br/matematica/fatorial.htm.

4. Crie uma classe chamada `Arranjo`. Em análise combinatória e fatorial um arranjo, assim como a permutação, é a formação de um reordenamento. A diferença é que, no arranjo, estamos reordenando parte do conjunto, ou seja, queremos saber quantos reordenamentos possíveis podemos formar escolhendo uma quantidade k de um conjunto com n elementos. A fórmula do arranjo é a seguinte:

$$A = \frac{n!}{(n-k)!}$$

   a) sua classe deve fazer uso da classe fatorial criada anteriormente para calcular um arranjo.

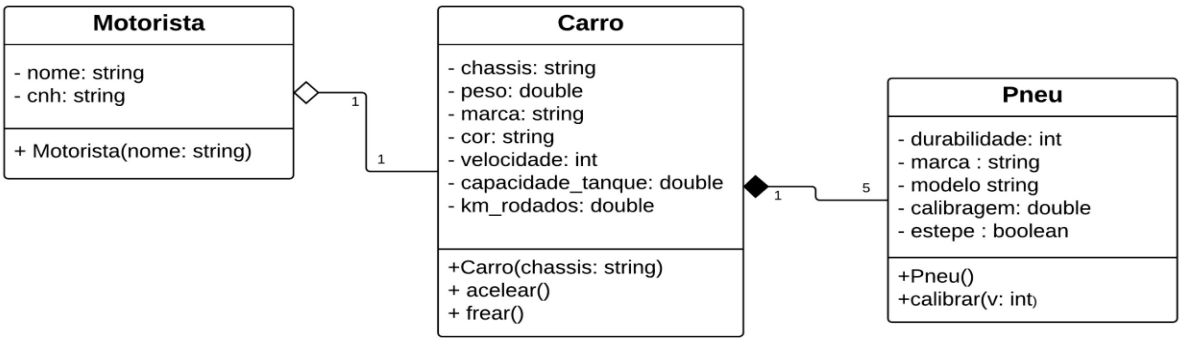
   b) Crie construtores, métodos para exibição e para retornar os cálculos.

   c) Nesse exercícios, a ligação entre a classe `Arranjo` e `Fatorial` é uma composição, agregação ou herança? Justifique.

5. Construa um objeto `Ponto` na linguagem C++. Esse objeto deverá representar um ponto no plano cartesiano (x, y) e terá um único método que permita calcular a distância para outro objeto `Ponto`. Crie ainda métodos `getters e setters` para cada um dos atributos definidos, assim como construtores para receber coordenadas logo ao inicializar um objeto.

6. Um pixel além das coordenadas `x` e `y`, possui uma cor e pode estar aceso ou apagado. Escreva uma classe `Pixel` que é uma herança de `Ponto`.

7. Considere o seguinte diagrama de classes:



a) composição e agregação são dois tipos de associações entre classes. No diagrama acima, a associação entre quais classes emprega a composição? Defina o conceito de agregação e composição.

b) no código `Motorista.h` e `Carro.h` do diagrama acima, escreva o codigo em cada arquivo .h para associar as classes de acordo com o diagrama. Em um arquivo de teste (`main.cpp`) instancie dois objetos carros e um objeto motorista. Relacione os objetos corretamente.

c) implemente uma lógica para relacionar a velocidade do carro (atributo velocidade) com a aceleração e a frenagem (métodos `acelerar()` e `frear()`).

d) implemente uma lógica envolvendo os km rodados de um carro com a durabilidade do penu.

e) ainda no arquivo de teste altere a pressão dos pneus e a velocidade do carro. Exiba os seguintes valores: Nome do motorista, nome do carro, cor, km_rodados, velocidade atual, marca, pressão e durabilidade dos pneus.

8. Escreva um programa que defina uma classe `Poligono` com um construtor que recebe valor para Comprimento e Altura. Defina duas subclasses `Triangulo` e `Retangulo`, que calculam a área por uma função `double area()`. No main(), defina dois objetos Triangulo e `Retangulo` e então chame a função `area`() desses objetos.

9. Ainda considerando o exercício 8:

```
void CalcularArea(Poligono &p){

    cout << "\nArea " << p.area();
}

int main()
{
    Retangulo r(9,8);
    Triangulo t(4,5);

    CalcularArea(r);
    CalcularArea(t);

    return 0;
}
```

a) inclua o método `double area()` na classe `Poligono` e a faça retornar o

valor 0.

b) Logo após, considerando o conceito de polimorfismo, implemente o método `CalcularArea(...)` na classe main(), de acordo com o código acima. Reporte o resultado e indique como corrigir o problema.

c) O que acontece se no método `CalcularArea(Poligono &p)` retirarmos o operador &? Por que isso ocorre?

10. Considere o seguinte código:

```
#include <iostream.h>

class Animal {
    public:
       Animal() { cout << "- construindo animal" << endl; }
      ~Animal() { cout << "- destruindo animal" << endl; }
 };

class Mamifero : public Animal {
    public:
       Mamifero() { cout << "-- construindo mamifero" << endl; }
      ~Mamifero() { cout << "-- destruindo mamifero" << endl; }
 };

 class Homem : public Mamifero {
    public:
       Homem() { cout << "--- construindo homem" << endl; }
      ~Homem() { cout << "--- destruindo homem" << endl; }
};

 void main(void) {
    Homem h;
}
```

Quando executado, qual será a saída produzida pela programa?

11. Considere que o código:

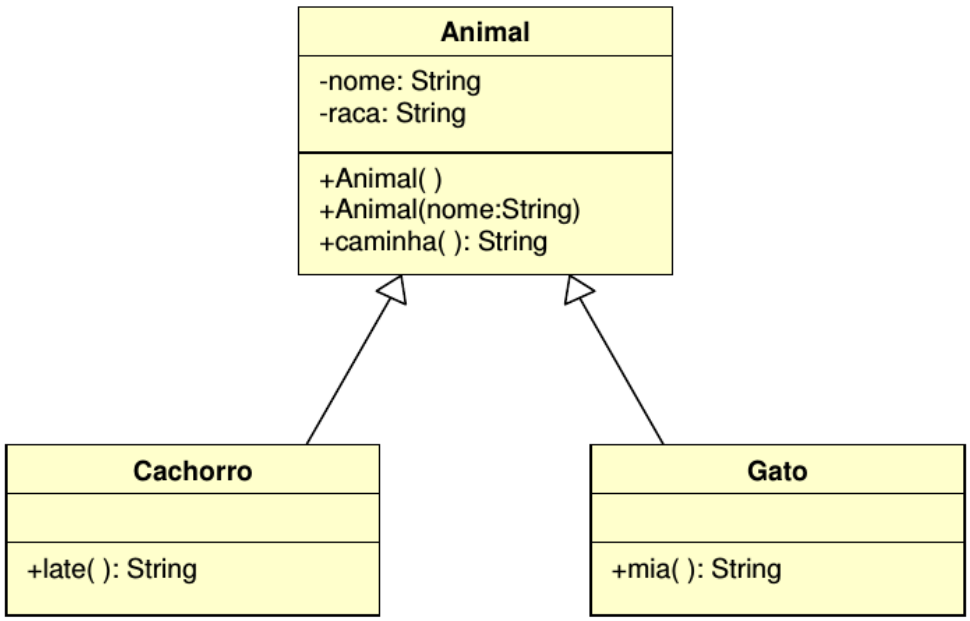
```
class Um : public Dois {
   public:
     Um() {
        Tres tres;
        cout << "alguma coisa" << endl;
     }

     ~Um() {
        cout << "outra alguma coisa" << endl;
     }

   private:
     Quatro _quatro;
   };
```

Produza a saída:

“um dois tres quatro cinco seis sete”.

Não é permitido alterar o código da Classe `Um` com exceção de suas saídas (os dois `cout`). É permitido criar outras classes. Dica: não é necessário criar mais do que três outras classes.

12. Implemente os diagramas de classe abaixo:



a) Crie uma classe teste e faça o gato miar e o cachorro latir. Faça os dois animais caminharem.

13. Crie uma classe `Data` para representar o dia, mês e ano. Os atributos são privados, mas a classe deve possuir métodos públicos para apresentar o dia, mês e ano.
    a) O dia deve estar em 1 e 30;
    b) O mês entre 1 e 12;
    c) O ano deve estar entre 1 e 2100.
    d) Nenhuma validação de dia/mês é necessário além das descritas anteriormente.
    e) Crie ainda dois construtores diferentes. O construtor padrão deve inicializar dia, mês e ano com o dia de hoje;
    f) Crie métodos get para cada um dos atributos;
    g) No arquivo main.cpp instancie objetos e faça testes.

14. Crie uma classe chamada Ingresso que possui as seguintes características:
    a) Atributos: nome do comprador, data do evento e valor. Todos os atributos são privados (ou protegidos);
    b) O atributo data do evento é uma composição com a classe `Data` descrita no exercício 13.

c) Crie um método `exibir()` que apresenta o nome do comprador, o valor pago e data do evento.
d) Crie um método `imprimeValor()` para apresentar o valor pago.
e) Crie diferentes construtores;
f) No arquivo `main.cpp` instancie objetos faça testes;
   Observação: para usar a classe `Data` do exercício anterior, lembre-se que os arquivos .cpp e .h da classe `Data` devem estar no mesmo projeto do codeBlocks deste exercício. Se preferir copie os arquivos do exercício 13 para este novo projeto e adicione-os ao projeto ou recrie as classes copiando o conteúdo.

15. A partir da classe Ingresso, construída no exercício 14:
    a) Crie uma classe `IngressoVip`, que representa um ingresso VIP. A classe herda da classe Ingresso e possui um valor adicional como atributo;
    b) O valor adicional é fornecido ao construir o objeto;
    c) Sobrescreva o método `imprimeValor`() para apresentar o novo valor do ingresso;
    d) No arquivo `main.cpp` instancie objetos faça testes.
       Observação: para usar a classe Ingresso do exercício anterior, lembre-se que os arquivos .cpp e .h da classe devem estar no mesmo projeto do codeBlocks deste exercício. Se preferir copie os arquivos .h e .cpp este novo projeto e adicione-os ao projeto ou recrie as classes copiando o conteúdo.

16. Qual a diferença entre overload (sobrecarga) e override (sobrescrita) em C++?

17. Identifique os polimorfismos possíveis em C++, utilizando exemplos.

18. Exemplifique em código uma situação de polimorfismo estático e uma de polimorfismo dinâmico em C++.

19. O código abaixo compila? Justifique.

```
#include <iostream>

using std::cout;
using std::endl;

template <typename T>

T max(T x, T y){
   return (x > y)? x : y;
}

int main() {
   cout << max(3, 7) << endl;
   cout << max(3.0, 7.0) << endl;
   cout << max(3, 7.0) << endl;
   return 0;
}
```

20. Deduza a saída do código abaixo:

```
#include <iostream>
using namespace std;

template <typename T>

void fun(const T & x) {
   static int contador = 0;
   cout << "x = " << x << " contador = " << contador << endl;
   ++contador;
   return;
}

int main() {
   fun<int> (1);
   cout << endl;
   fun<int>(1);
   cout << endl;
   fun<double>(1.1);
   cout << endl;
   return 0;
}
```