UNIVERSIDADE ABERTA
www.uab.pt

Ministério da Educação e Ciência

p-Fólio

U.C. (21093)

Programação por Objetos

XX de Julho de 2015

-- INSTRUÇÕES --

- O estudante deverá responder à prova na folha de ponto e preencher o cabeçalho e todos os espaços reservados à sua identificação, com letra legível.
- Sempre que não utilize o enunciado da prova para resposta, poderá ficar na posse do mesmo.
- No caso de provas com escolha múltipla, **sem grelha de resposta**, deverá indicar a resposta correcta na folha de ponto, indicando o número da pergunta e a resposta que considera correcta.
- No caso de provas com escolha múltipla, **com grelha de resposta, tabela e/ou espaços para preenchimento**, deverá efectuar as respostas no enunciado, pelo que o mesmo deverá ser entregue ao vigilante, juntamente com a folha de ponto, **não sendo permitido ao estudante levar o enunciado.**
- Verifique no momento da entrega da(s) folha(s) de ponto se todas as páginas estão rubricadas pelo vigilante. Caso necessite de mais do que uma folha de ponto, deverá numerá-las no canto superior direito.
- Em hipótese alguma serão aceites folhas de ponto dobradas ou danificadas.
- Exclui-se, para efeitos de classificação, toda e qualquer resposta apresentada em folhas de rascunho.
- Os telemóveis deverão ser desligados durante toda a prova e os objetos pessoais deixados em local próprio da sala de exame.
- A prova é constituída por **1** página e termina com a palavra **FIM**. Verifique o seu exemplar e, caso encontre alguma anomalia, dirija-se ao professor vigilante nos primeiros 15 minutos da mesma, pois qualquer reclamação sobre defeito(s) de formatação e/ou de impressão que dificultem a leitura não será aceite depois deste período.
- Utilize unicamente tinta azul ou preta.
- Responda às questões de forma clara, sucinta, e apresente todos os cálculos.
- Quando solicitado, apresente ainda uma representação gráfica do resultado final obtido na questão.
- A cotação de cada uma das questões é indicada junto do enunciado da mesma.
- A prova é **SEM CONSULTA**. Todos os elementos necessários à resolução são fornecidos no enunciado.

Duração: 90 minutos

QUESTÃO 1 (12 valores)

Implemente um **Veículo**. O veículo é composto por várias partes: um motor, um tanque de combustível e 4 pneus.
O funcionamento do veículo depende das suas partes, da seguinte forma:

Motor
- Possui uma potência (em hp), uma taxa fixa de consumo (em km/litro) e um tanque de combustível (ver abaixo).
- Possui um comando para **avançar** uma determinada quantidade de km. Se a pressão de mais de um pneu (ver abaixo) estiver abaixo de 20 lb, o consumo do veículo aumenta em 30%. Se houver combustível suficiente, o veículo avança até o tanque esvaziar.

Tanque de Combustível
- Possui uma determinada capacidade e quantidade atual de combustível (ambos em litros).
- Pode ser **abastecido** com uma certa quantidade de combustível, limitado à sua capacidade máxima.

Pneu
- Cada pneu possui uma determina pressão (em lb).
- Pode ser **calibrado** com determinada pressão informada (positiva ou negativa, sendo somada à atual).

Implemente as classes acima, usando composição quando necessário. Lembre-se de implementar *gets* e *sets* necessários, bem como construtores adequados.
No programa principal, faça as seguintes operações:
- Instancie um veículo cujo motor tem 71 hp, consumo de 12 km/litro, tanque com capacidade para 50 litros, pneus dianteiros com 27 lb e traseiros com 23 lb.
- Abasteça o tanque com 30 litros.
- Exiba no ecrã as informações sobre cada componente do veículo.
- Avance 300 km.
- Reduza a pressão do pneu traseiro esquerdo para 17 lb.
- Avance 100 km.
- Reduza a pressão do pneu dianteiro direito para 18 lb.
- Abasteça mais 10 litros.
- Avance 200 km.

Esta questão será avaliada da seguinte forma:
- Declaração de atributos e métodos de cada classe necessária (ficheiros .h) – 4.5 pontos
- Definição dos métodos necessários de cada classe (ficheiros .cpp) – 4.5 pontos
- Definição correta da *main*( ) para as operações solicitadas – 3 pontos

Não esqueça de incluir os *#includes* necessários nos ficheiros .h.

FIM

Possível solução:

```
//------------- main.cpp --------------------------//
#include "Veiculo.h"
using namespace std;
int main() {
 Veiculo fusca;
 Tanque tanque;
 vector <Pneu> pneus;
 Pneu p1, p2, p3, p4;
// Cria a 1ª versão do carro
 tanque.setQtdAtual(30);
 tanque.setCapacidade(50);
 fusca.setTanque(tanque);
 fusca.setPotencia(71);
 fusca.setTxConsumo(12);
 fusca.setTanque(tanque);

 p1.setPressao(27);
 pneus.push_back(p1);
 p2.setPressao(27);
 pneus.push_back(p2);
 p3.setPressao(23);
 pneus.push_back(p3);
 p4.setPressao(23);
 pneus.push_back(p4);
 fusca.setPneus(pneus);

 // Imprime detalhes do carro
 fusca.imprimeInfo();
 // Avança 300 kms
 fusca.avancar(300, fusca.verificaPressao());
 // Reduz pressão no pneu traseiro esquerdo para 17 lb
 pneus.at(2).setPressao(17);
 fusca.setPneus(pneus);
 // Imprime detalhes do carro
 fusca.imprimeInfo();
 // Avança 100 kms
 fusca.avancar(100, fusca.verificaPressao());
 // Reduz pressão no pneu traseiro esquerdo para 18 lb
 pneus.at(3).setPressao(18);
 fusca.setPneus(pneus);
 // Abastece mais 10 litros
 tanque.abastacer(10);
 fusca.setTanque(tanque);
 // Avança 200 kms
 fusca.avancar(200, fusca.verificaPressao());
 return 0;
}

//------------- Ficheiros .h --------------------------//
#include "Pneu.h"
#include "Motor.h"
#include <vector>
namespace std {
class Veiculo : public Motor{
private:
 vector<Pneu> pneus;
```

```
public:
 Veiculo();
 virtual ~Veiculo();
    void imprimeInfo();
    vector<Pneu> getPneus() const;
    void setPneus(vector<Pneu> pneus);
    bool verificaPressao();
};
-----------------------------------------------
#include <iostream.h>
namespace std {
class Tanque {
private:
    int capacidade;
    int qtdAtual;
public:
    Tanque();
    virtual ~Tanque();
    int getCapacidade() const;
    int getQtdAtual() const;
    void setCapacidade(int capacidade);
    void setQtdAtual(int qtdAtual);
    void abastacer(int qtd);
};
-----------------------------------------------
#include <iostream>
namespace std {
class Pneu {
private:
 int pressao;
 int maxPressao;
public:
 Pneu();
 virtual ~Pneu();
    int getMaxPressao() const;
    void setMaxPressao(int maxPressao);
    void calibrar(int);
    int getPressao() const;
    void setPressao(int pressao);
};
-----------------------------------------------
#include "Tanque.h"
namespace std {
class Motor
{
private:
    int potencia;
    int txConsumo;
    Tanque tanque;
public:
    Motor();
    virtual ~Motor();
    int getPotencia() const;
    Tanque getTanque() const;
    float getTxConsumo() const;
    void setPotencia(int potencia);
    void setTanque(Tanque tanque);
    void setTxConsumo(int txConsumo);
    void avancar(int km, bool status);
```

```
};

//------------- Ficheiros .cpp --------------------------//
#include "Pneu.h"
namespace std {
    Pneu::Pneu() {
       this->maxPressao = 0;
       this->pressao = 0;
    }
    int Pneu::getMaxPressao() const {return maxPressao; }
    void Pneu::setMaxPressao(int maxPressao) {this->maxPressao = maxPressao;}
    Pneu::~Pneu() {}
    int Pneu::getPressao() const {return pressao;}
    void Pneu::setPressao(int pressao) {this->pressao = pressao;}
    void Pneu::calibrar(int p)
    {
     if ((pressao + p) < maxPressao) pressao += p;
     else cout << "A pressão é demais...";
    }
}

-------------------------------------------------------------
#include "Motor.h"
namespace std
{
    Motor::Motor()
    {
     this->potencia = 0;
     this->txConsumo = 0;
    }
    int Motor::getPotencia() const { return potencia; }
    Tanque Motor::getTanque() const { return tanque; }
    float Motor::getTxConsumo() const { return txConsumo; }
    void Motor::setPotencia(int potencia) { this->potencia = potencia; }
    void Motor::setTanque(Tanque tanque) { this->tanque = tanque; }
    void Motor::setTxConsumo(int txConsumo) { this->txConsumo = txConsumo; }
    Motor::~Motor()  {}
    void Motor::avancar(int km, bool status)
    {
     int distCapazPercorrer, i=1;
     // Se mais de um pneu estiver abaixo de 20 lb, aumenta em 30% o consumo
     if (status) distCapazPercorrer = tanque.getQtdAtual() * txConsumo;
     else distCapazPercorrer = tanque.getQtdAtual() * (txConsumo - (30 *
txConsumo)/100);

     if (distCapazPercorrer > km)
     {
      for (i = 1; i < km; ++i) cout << "Percorreu: " << i << "kms" << endl;
     }
     else cout << "Combustível insuficiente..." << endl;
// atualiza total de combustível
     tanque.setQtdAtual((tanque.getQtdAtual() - i));
     }
}

-------------------------------------------------------------
#include "Tanque.h"
namespace std {
    Tanque::Tanque() {
```

```
        this->capacidade = 0;
        this->qtdAtual = 0;
    }
    int Tanque::getCapacidade() const { return capacidade; }
    int Tanque::getQtdAtual() const { return qtdAtual;}
    void Tanque::setCapacidade(int capacidade) {this->capacidade = capacidade;}
    void Tanque::setQtdAtual(int qtdAtual) { this->qtdAtual = qtdAtual;}
    Tanque::~Tanque() {}
    void Tanque::abastacer(int qtd)
    {
     int total = 1;
     while (((qtdAtual + 1) < capacidade) && (total <= qtd))
     {
      ++qtdAtual;
      ++total;
     }
     cout << "Encheu " << total << "litros" << endl;
    }
}
-----------------------------------------------------------------
#include "Veiculo.h"
namespace std {
    Veiculo::Veiculo() {
     this->pneus.reserve(4);
    }
    Veiculo::~Veiculo() {}
    vector<Pneu> Veiculo::getPneus() const { return pneus; }
    void Veiculo::setPneus(vector<Pneu> pneus) {this->pneus = pneus;}
    void Veiculo::imprimeInfo()
    {
     cout << "Potência do motor " << this->getPotencia() << "hp" << endl;
     cout << "Taxa de consumo " << this->getTxConsumo() << " km/lt" << endl;
     cout << "Capacidade do tanque " << this->getTanque().getCapacidade() <<
endl;
     cout << "Total de combustível: " << this->getTanque().getQtdAtual() << endl;
     cout << "Pressao do 1º pneu: " << this->pneus.at(0).getPressao() << endl;
     cout << "Pressao do 2º pneu: " << this->pneus.at(1).getPressao() << endl;
     cout << "Pressao do 3º pneu: " << this->pneus.at(2).getPressao() << endl;
     cout << "Pressao do 4º pneu: " << this->pneus.at(3).getPressao() << endl;
    }
    bool Veiculo::verificaPressao()
    {
     int conta = 0;
     if (this->pneus.at(0).getPressao() < 20) ++conta;
     if (this->pneus.at(1).getPressao() < 20) ++conta;
     if (this->pneus.at(2).getPressao() < 20) ++conta;
     if (this->pneus.at(3).getPressao() < 20) ++conta;
     if (conta >= 2) return false;
     else return true;
    }
}
```