UNIVERSIDADE ABERTA
www.uab.pt

Ministério da Educação e Ciência

p-Fólio

U.C. (21093)

Programação por Objetos

XX de Julho de 2016

-- 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)

Pedro deseja codificar uma aplicação de **controlo de tarefas** para colocar em seu PDA. As especificações da aplicação são as seguintes:
O registo de cada **tarefa** contém o número da prioridade, representado por um valor numérico, do tipo *float*. Isso permite entrar com intervalos intermediários. Além da prioridade, o registo deve conter:
o nome da tarefa, a data limite de execução (se houver), o percentual já concluído (que resulta da média do somatório dos percentuais da lista de itens a serem executados) e o detalhamento da tarefa.
Para cada tarefa há uma **lista de itens** que descrevem a sua execução. Para cada item de execução, registam-se:

- o percentual correspondente (que é actualizado pelo Pedro de vez em quando)
- a descrição da execução
- a data da execução (quando for concluída)

Quando uma tarefa receber 100% de execução, esta deve ser movida automaticamente para a lista de tarefas concluídas, podendo ser apagada, se for o caso.
Veja o exemplo desse controlo em papel:

TAREFA 1.1. - ANIVERSÁRIO DO FÁBIO
Data limite = 06/08/2016
Percentual já concluído = 20%
Detalhamento = planeamento dos preparativos para a festa de aniversário do Fábio, no sábado, dia 6 de agosto.
Lista de Itens para serem executados:
[20%] Aluguel do salão e da animação - 01/03/2016
[20%] Encomenda do bolo, salgados e doces - 15/07/2016
[5%] Compra das bebidas
[25%] Compra dos itens para a decoração - 01/07/2016
[30%] Arrumação do salão

Esta questão será avaliada da seguinte forma:

- Declaração de atributos e métodos de todas as classes necessárias (ficheiros .h) – 4 pontos
- Definição dos **métodos na classe Tarefa** necessários para (ficheiro .cpp):
  - Criar uma tarefa; (1 ponto)
  - Imprimi-la na console; (1 ponto)
  - Editar um item da tarefa; (3 pontos)
  - Verificar o percentual de conclusão da tarefa (3 pontos)

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

FIM

SOLUÇÃO:

```
#include "Tarefa.h"

namespace std {
class ListaTarefas {
 list<Tarefa> tarefas;
 list<Tarefa>::iterator it;
public:
 ListaTarefas();
 virtual ~ListaTarefas();
 void adicionaTarefa();
 void eliminaTarefa();
 void consultaTarefa();
 void atualizaTarefa();
};
}
```

```
#include <list>
#include "ItemExecucao.h"

namespace std {

class Tarefa {
private:
 float numPrioridade;
 string nome;
 struct datas
 {
  int dia;
  int mes;
  int ano;
 } data;
 string detalhamento;
 list<ItemExecucao> items;
 list<ItemExecucao>::iterator it;
 enum statusTarefa
  {
   concluida,
   ativa
  };
 statusTarefa status;
public:
 Tarefa();
 virtual ~Tarefa();
    string getDetalhamento() const;
    list<ItemExecucao> getItems() const;
    string getNome() const;
    float getNumPrioridade() const;
     void setDetalhamento(string detalhamento);
    void setItems(list<ItemExecucao> items);
    void setNome(string nome);
    void setNumPrioridade(float numPrioridade);
    void imprimeTarefa();
    float percentualConcluido();
    void editaItem();
};
}
```

```
#include <iostream>
```

```
#include <string>
#include <ctime>

namespace std {

class ItemExecucao {
private:
 float percentual;
 string descricao;
 struct datas
 {
  int dia;
  int mes;
  int ano;
 } data;
public:
 ItemExecucao();
 virtual ~ItemExecucao();
 void imprimeItem();
    string getDescricao() const;
    float getPercentual() const;
    void setDescricao(string descricao);
    void setPercentual(float percentual);
};

}

#include "Tarefa.h"

namespace std {

 Tarefa::Tarefa() {
  int nItens;
  string entrada;
  cout << "Entre o número de prioridade da tarefa:" << endl;
  cin >> numPrioridade;
  cout << "Entre o título da tarefa:" << endl;
  cin >> nome;
  // captura a data atual do sistema
  cout << "Dê a data limite (dd/mm/aaaa):" << endl;
  cin >> entrada;
  data.dia = std::atoi(entrada.substr(0,2).c_str());
  data.mes = std::atoi(entrada.substr(3,2).c_str());
  data.ano = std::atoi(entrada.substr(6,4).c_str());
  cout << "Dê detalhamento: " << endl;
  cin >> detalhamento;
  cout << "Quantos itens terá a tarefa? " << endl;
  cin >> nItens;
  for (int i=0; i<nItens; ++i) items.push_back(ItemExecucao());
  status = ativa;
 }

    void Tarefa::imprimeTarefa()
    {
     float totalPercentual = 0;
     cout << numPrioridade << endl;
     cout << nome << endl;
     cout << "Ano: "<< data.ano << endl;
     cout << "Mês: "<< data.mes<< endl;
     cout << "Dia: "<<  data.dia << endl;
     cout << detalhamento << endl;
```

```
     for (it = items.begin(); it != items.end(); it++)
     {
      (*it).imprimeItem();
      totalPercentual = totalPercentual + (*it).getPercentual();
     }
     cout << "Tarefa completa em: " << totalPercentual/items.size() << "%"
<< endl;
    }

    float Tarefa::percentualConcluido()
    {
     float totalPercentual = 0;
     for (it = items.begin(); it != items.end(); it++)
     {
      (*it).imprimeItem();
      totalPercentual = totalPercentual + (*it).getPercentual();
      }
     if (totalPercentual >= 100) status = concluida;

     return totalPercentual;
    }

    void Tarefa::editaItem()
    {
     string resp;
     float percentual;
        cout << "Entre 's' quando for o item que deseja atualizar
percentual:" << endl;
        it = items.begin();
     while (it != items.end())
     {
      (*it).imprimeItem();
      cout << endl;
      cout << "É esse?" << endl;
      cin >> resp;
      if (resp.compare("s")==0)
      {
       cout << "Qual é o percentual de conclusão novo?" << endl;
       cin >> percentual;
       (*it).setPercentual(percentual);
       break;
      }
      it++;
     }
    }

}
```