UNIVERSIDADE ABERTA
www.uab.pt

Ministério da Educação e Ciência

p-Fólio

U.C. 21093

Programação por Objetos

10 de julho de 2012

-- 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.
- No fim da prova, poderá ficar na posse do 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 objectos pessoais deixados em local próprio da sala de exame.
- Utilize unicamente tinta azul ou preta.
- A prova é constituída por 4 páginas (esta página de rosto e três com as questões), contém 2 grupos de questões com um total de 9 alíneas, sem consulta, 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.

Duração: 90 minutos

GRUPO 1 (3 valores) (1a=0.5; 1b=0.5; 1c=1.0; 1d=1.0)
Questões de resposta múltipla, onde apenas uma resposta está correta.

1.

a) Considere a expressão `! (a && !b)`
Esta é equivalente a qual das seguintes expressões?

A. `( a || b)`
B. `(!a) || b`
C. `(!a) && b`
D. `(!a) && (!!b)`
E. `(a || b) && (a || b)`

b) Qual das seguintes afirmações acerca de variáveis em programas C++ está correta?

A. Uma variável deve ser declarada antes de ser utilizada.
B. O mesmo nome da variável não pode ser utilizado em duas funções diferentes.
C. É considerada uma boa prática de programação em C++ declarar todas as variáveis não escalares (ex. vetores) globais e declarar locais as variáveis escalares.
D. É considerada uma boa prática de programação em C++ usar letras individuais para nomes de todas as variáveis.
E. Nenhuma das afirmações está correcta.

c) Seja a seguinte função booleana:

```
bool MyFunction (const vector <int> & A)
// pré-condição: A está ordenado
{
int k;
for (k=1; k<A.size(); k++){
  if (A[k-1] == A[k]) return false;
}
return true;
}
```

Qual das seguintes afirmações explica melhor o que a função *MyFunction* faz?

A. Retorna sempre true.
B. Retorna sempre false.
C. Determina se o vector A está (ou não) realmente ordenado.
D. Determina se o vector A contém (ou não) algum valor duplicado.
E. Determina se todos os valores em A são diferentes (ou não).

d) Relativamente à função da alínea anterior, o que aconteceria se a variável k fosse declarada no ciclo for (`for (int k=1; k<A.size(); k++)`)?

A. A compilação falharia.
B. O código compilava corretamente, mas acontecia um erro durante a execução.
C. O programa executaria normalmente, mas o resultado da função seria diferente.
D. O programa executaria normalmente, sem qualquer alteração no resultado da função.
E. Ocorreria um ciclo infinito, porque a variável seria sempre igual a 1 em cada iteração do ciclo.

GRUPO 2 (9 valores) (1a=1.0; 1b=2.0; 2a=1.0; 2b=2.0; 2c=3.0)

1.

a) Escreva o código em C++ para uma função com o nome *FindValue* conforme indicada mais abaixo. *FindValue* deverá retornar o índice da primeira ou da última posição no vector A que contém o valor pretendido. Se *dir* for 1, deve procurar o primeiro valor, se *dir* for -1, deve procurar o último valor. Caso não encontre o valor, deve retornar -1. Complete então a função *FindValue* abaixo.

```
int FindValue(const vector <int> &A, int value, int dir){
 (código para ser programado)
}
```

b) Escreva código em C++ para a função *SetValue* conforme indicada mais abaixo. *SetValue* deve procurar as primeira e última ocorrências do valor dado *value* no vetor A, alterando todos os elementos entre as duas posições do vetor para o valor dado. Para tal deve usar a função *FindValue* descrita na alínea anterior. Caso esta função retorne -1 (o valor não existe no vetor) ou caso o índice do primeiro valor seja igual ao do último (o valor só ocorre uma vez no vetor), o vetor não é alterado. Complete então a função *SetValue* abaixo:

```
void SetValue (vector <int> & A, int value){
(código para ser programado)
}
```

2. Considere que a classe *Product*, abaixo especificada, foi implementada. Esta classe representa um produto para venda num estabelecimento comercial. Os métodos públicos da classe permitem que uma aplicação cliente obtenha o nome do produto, a quantidade deste produto atualmente em stock, e que subtraia 1 da quantidade de produtos atualmente em stock.

```
class Product {
public:
  Product (string name); // constructor
  string Name () const; // devolve o nome deste produto
  int NumInStock () const; // devolve a quantidade em stock
  void SellOne (); // subtrai da quantidade em stock
private:
  string myName;
  int myNumInStock;
};
```

a) Escreva uma função *VerifyStock*, como abaixo iniciada. *VerifyStock* deverá devolver a quantidade em stock atual de um produto da classe *Product* num vector (*array*) A dado um nome ou devolver -1 se esse produto não existir em A. Complete então a função *VerifyStock* assumindo que esta é apenas chamada com valores que satisfizerem a sua pré-condição.

```
int VerifyStock (const vector <Product> &A, const string
&name){
// pré-condição: não existem produtos com o mesmo nome em A
 (código para ser programado)
}
```

b) Escreva o código em C++ da função *MySale*, como iniciada mais abaixo. A função *MySale* tem dois parâmetros: um vetor (*array*) de produtos (*Product*) nomeado *inventory* e um nome de um produto (que um cliente pretende comprar). A função *MySale* deve tentar realizar a venda do produto indicado pelo nome e devolver *true* ou *false* conforme a venda for possível ou não. A venda acontece se existir pelo menos um produto (de nome indicado) no vetor. Nesse caso, *MySale* deve subtrair 1 do número de produtos em stock e devolver *true*. Se não existir um produto com o nome indicado ou o número em stock desse produto for zero (o produto não existe em stock) então *MySale* deve devolver *false*. Na escrita do código da função *MySale* deverá considerar a função *VerifyStock* especificada na alínea anterior. Complete então a função *MySale* admitindo que esta é apenas chamada com valores que satisfizerem a sua pré-condição.

```
bool MySale (vector <Product> &inventory, const string &name)
{
// pré-condição: não existem produtos com o mesmo nome em
inventory
 (código para ser programado)
}
```

c) Escreva o código em C++ da função *SellN*, como iniciada mais abaixo. A função *SellN* tem quatro parâmetros:

- um vector de produtos (*Product*) nomeado *inventory*;
- o nome do produto a vender *name*;
- o número de itens do produto solicitados *NReq*;
- o número de itens efetivamente vendidos *NSold*.

A função deverá procurar o produto pelo nome e tentar vender o número solicitado em *NReq*, podendo usar apenas a função *MySale*. O número de itens efetivamente vendidos deve ser colocado em *NSold*.
Complete a função abaixo, assumindo que esta é chamada apenas com valores que satisfazem a sua pré-condição.

```
void SellN (vector <Product> & inventory, const string &name,
int NReq, int &NSold){
// pré-condição: não existem produtos com o mesmo nome em
inventory
(código para ser programado)
}
```

FIM