

U.C. 21093

Programação por Objetos

11 de julho de 2013

-- 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 declarada no início de um ciclo for pode ser utilizada fora desse ciclo.
- 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. O mesmo nome da variável pode ser usada mais do que uma vez dentro da mesma função.

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 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.

QUESTÃO 2 (9 valores) (a=1.0; b=2.0; c=1.0; d=2.0; e=3.0)

- a) Escreva o código em C++ para uma função com o nome *FindValue* conforme indicada mais abaixo. *FindValueH* deverá retornar o índice da primeira ou da última posição no vector A que contém um **valor superior ou igual** ao 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. Assuma que a função é chamada apenas com valores que satisfazem a sua pré-condição.

```
int FindValueH(const vector <int> &A, int value, int dir){  
    // pré-condição: 0 <= pos1 <= pos2 <= A.size()  
    (código para ser programado)  
}
```

- b) Escreva código em C++ para a função *SetZero* conforme indicada mais abaixo. *SetValue* deve procurar as primeira e última ocorrências de **valores superiores ou iguais** a um dado *value* no vetor A, alterando todos os elementos entre as duas posições do vetor para o valor 0. 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 *SetZero* abaixo:

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

- c) 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 um dado valor 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 Sell (int n); // subtrai da quantidade em stock  
private:  
    string myName;  
    int myNumInStock;  
};
```

Escreva uma função *VerifyStock*, como abaixo iniciada. *VerifyStock* deverá devolver true se a quantidade pretendida existir em stock ou false caso a quantidade pretendida seja inferior à quantidade em stock. Complete então a função *VerifyStock* assumindo que esta é apenas chamada com valores que satisfizerem a sua pré-condição.

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

- d) 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)
}
```

- e) Escreva o código em C++ da função *MyAllSales*, como iniciada mais abaixo. A função *MyAllSales* tem três parâmetros:
- um vetor de produtos (*Product*) nomeado *inventory*;
 - um vetor de nomes de produtos a vender *names*;
 - o número de itens efetivamente vendidos *NSold*.
- A função deverá procurar os produtos em *names* e tentar vender um produto de cada, 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 MyAllSales (vector <Product> & inventory, const vector
<string> &names, int &NSold)
// pré-condição: não existem produtos com o mesmo nome em
inventory
{
(código para ser programado)
}
```

FIM