U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO | 1º ANO
EIC0012| PROGRAMAÇÃO | 2018/2019 – 2º SEMESTRE

EXAME - ÉPOCA NORMAL | 2019-06-21

Com consulta. Duração: 2h30m

Nome do estudante: _____ **Código UP:** _____ **NºExm:** _____

1. [4.0]

Indique quais das seguintes afirmações são verdadeiras e quais são falsas, assinalando respetivamente com **V** ou **F.** **NOTA** - a pontuação nesta pergunta será dada pela fórmula: **máximo (0, (nº_respostas_certas – nº_respostas_erradas) x 0.2)**	**V / F**
Num programa escrito em C++ é possível declarar uma variável com o nome **friend**, por exemplo: **string friend;**	
O pedaço de código **int x, n = 0; while (cin >> x) n = n + x;** lê do teclado uma sequência de números inteiros, até que o utilizador tecle CTRL-Z (em Windows) ou CTRL-D (em Linux) e guarda a soma dos números na variável **n**.	
A definição **set<Person> s;** produz um erro de compilação se não tiver sido feito o *overloading* do **operator<** para o tipo **Person**.	
A expressão **(ch == "X")** é válida quer a variável **ch** seja do tipo **char** quer seja do tipo **string**.	
A função **f()** cujo protótipo é **void f(int &a)** deve ser invocada da seguinte forma quando se pretende passar-lhe o parâmetro **5**: **f(int &5);** .	
A instrução **cin>>name;** permite ler um nome com duas palavras (ex: Cristiano Ronaldo) para a variável **name**, do tipo **string**.	
O código **for (int i=1; i>=10; i++) cout << i;** escreve no ecrã o seguinte: **12345789** .	
Sendo **a** um *array* bidimensional de números inteiros, o primeiro elemento do *array* é designado por **a[1,1]** .	
Sendo **s** uma variável do tipo **struct** que tem dois campos, **first** e **second**, do tipo **string**, o valor do campo **first** pode ser mostrado no ecrã usando a instrução **cout << s->first;** .	
A palavra reservada **this** só pode ser usada no interior de métodos (membros-funções) de uma classe e representa um apontador para o objeto sobre o qual o método está a operar.	
Um *function object* é um objeto que se comporta como se fosse uma função.	
Quando uma classe tem um atributo (membro-dado) qualificado como **static**, apenas existe uma instância desse atributo, partilhada por todos os objetos dessa classe.	
Na Standard Template Library de C++, a função **sort()** tanto pode ser usada para ordenar contentores do tipo **vector** como do tipo **list**.	
Considerando a definição **vector<int>::iterator p = v.begin();** a instrução **cout << *p;** mostra no ecrã o primeiro elemento do vetor **v**.	
É possível copiar o conteúdo de um *array* **a1** para outro *array* **a2**, usando a instrução **a2 = a1;** .	
A instrução **int *a = (int *) malloc(10*sizeof(int));** reserva dinamicamente espaço para um *array* **a**, com capacidade para 10 inteiros, sendo possível atribuir o valor **31** ao último elemento do *array* usando a instrução **a[9]=31;**	
Nas classes em que é feita alocação dinâmica de memória para algum dos seus atributos (membros-dados) é conveniente definir o *destructor* da classe.	
Num programa escrito em C++ é possível ter duas variáveis com o mesmo nome se elas estiverem declaradas em *namespaces* diferentes.	
A seguinte declaração significa que a classe **B** é derivada da classe **A** e que todas a funções da classe **B** são públicas: **class A : public B { . . .** *código omitido* **};**	
Um método "puramente virtual" de uma classe é uma função virtual que está declarada mas não definida, isto é, não tem corpo; tem de ser definida nas classes derivadas.	

Nome do estudante: _____ **Código UP:** _____ **NºExm:** _____

Notas: 1) nesta prova apenas é necessário fazer tratamento de erros e indicar os ficheiros de inclusão quando tal for solicitado explicitamente
2) considere que a diretiva using namespace std; está incluída em todos os programas

2. [5.0]

a) [1.0] A função **count()** – ver protótipo ao lado - retorna o número de valores existentes no vetor **v** que são iguais a **value**. Complete o código de **count()**.

```
int count(const vector<int> &v, int value)
```

b) [0.5] Justifique o modo de passagem do parâmetro **v** para a função.

c) [1.0] Escreva a função **main()** de um programa que permite testar a função **count()**. O programa deve ler do teclado um conjunto de valores, guardando-os num vector, até que o utilizador tecle **CTRL-Z** (em Windows) ou **CTRL-D** (em Linux). No final, deve invocar **count()** e escrever no ecrã o número de valores iguais a zero.

```
int main()
```

d) [0.5] Pretende-se transformar a função **count()**, da alínea **a**, numa *template function* que permita contar o número de ocorrências de um valor, num vetor cujos elementos são de um tipo genérico **T**. Indique as alterações que seria necessário fazer no código de **a)**.

e) [0.5] Para que a *template function* possa ser usada para contar objetos de uma **class Date**, usada para representar datas, o que seria necessário fazer?

f) [1.0] Pretende-se fazer o *overload* da função **count()**, da alínea **a**, para poder ser usada com um *array* de inteiros. Indique as alterações (apenas as alterações) a introduzir no código de **count()** e no código de **main()**.

Considere que o número máximo de elementos a ler do teclado é 100.

g) [0.5] Na STL, existe uma função, cujo protótipo é **size_t count(InputIterator first, InputIterator last, const T& value)**, que permite fazer a referida contagem. Indique as alterações que é necessário introduzir no código da função **main()** da alínea **c**, por forma a usar a função **count()** da STL.

Nome do estudante: _____ **Código UP:** _____ **NºExm:** _____

3. [5.0]

Pretende-se implementar uma classe **Time** para representar tempos, em horas, minutos e segundos.
Para além dos **atributos** (membros-dados), **h**, **m** e **s**, representados por números inteiros, a classe deve ter os seguintes **métodos** (membros-funções):

- **2 construtores**,
  - o um com 3 parâmetros, representando horas, minutos e segundos; o valor por omissão dos parâmetros é zero;
  - o outro com 1 parâmetro do tipo *string*, representando um tempo no formato HH:MM:SS;
- **setH()** , **setM()** e **setS()** para cada um dos atributos;;
- **getH()** , **getM()** e **getS()** para cada um dos atributos;
- um método **getStr()** que retorna o tempo sob a forma de uma *string*, no formato HH:MM:SS;
- **operadores +** e **–** para somar e subtrair objetos da classe **Time**.

a) [1.0] Escreva (ao lado) a definição da classe **Time**. Por agora, não implemente os métodos.

NOTAS: **1)** por simplificação, inclua apenas o protótipo de um dos métodos **get** e de um dos métodos **set**; **2)** use o qualificativo **const** sempre que tal se justificar.

b) [1.0] Escreva o código do construtor que tem como parâmetro uma *string*. Considere que a *string* representa um tempo válido, no formato HH:MM:SS.

c) [0.5] Escreva o código do método **getStr()** acima referido.

d) [1.0] Escreva o código que implementa o **operador +**. O resultado deve ser sempre representado em horas, minutos e segundos, ainda que o número de horas resultantes da soma possa ultrapassar 24h.

e) [1.0] Escreva a função **main()** de um programa que lê do teclado uma *string* representando um tempo válido e mostra no ecrã o resultado da sua soma com o tempo oito horas, zero minutos e zero segundos.

```
int main()
```

f) [0.5] Indique 2 formas alternativas que poderiam ser usadas para o construtor da alínea **b** "anunciar" que o parâmetro que recebeu representa um tempo inválido. Não reescreva o código do construtor, mas indique eventuais alterações ao protótipo, se tal se justificar.

Nome do estudante: _____ **Código UP:** _____ **NºExm:** _____

4. [3.0]

O ficheiro de texto **temp.txt**, gravado no diretório "**C:\Documents**", contém o registo das temperaturas medidas ao longo de uma experiência realizada durante um certo período de tempo. Cada linha do ficheiro contém uma hora, no formato HH:MM, e a temperatura medida nessa hora, separadas pelo caráter '**=**' (ver exemplo ao lado). Os dados estão ordenados cronologicamente, são todos válidos e não têm erros de formatação.

Pretende-se desenvolver um programa para determinar a duração total dos períodos em que a temperatura subiu. O programa deve fazer o seguinte:

- abrir o referido ficheiro; caso não seja possível abrir o ficheiro, o programa deve apresentar uma mensagem de erro e terminar com o código de terminação 1;
- caso o ficheiro tenha sido aberto com sucesso, ler e mostrar o seu conteúdo no ecrã, acrescentando, em cada linha, informação sobre a evolução da temperatura, como ilustrado ao abaixo (isto é, deve ser mostrado o conteúdo de cada uma das linhas do ficheiro e, a partir da 2ª linha, deve ser escrita uma frase que indique como evoluiu a temperatura no período anterior e a duração desse período) e, no final, mostrar o tempo total de subida da temperatura; neste caso, o programa termina retornando o código de terminaçao 0 (zero).

Exemplo de conteúdo de **temp.txt**

```
08:00 = 19.5
10:30 = 19.4   desceu durante  02:30
11:30 = 19.3   desceu durante  01:00
12:00 = 19.3   estável durante 00:30
12:30 = 19.7   subiu durante   00:30
13:00 = 20.0   subiu durante   00:30
14:30 = 20.2   subiu durante   01:30
15:15 = 20.2   estável durante 00:45
16:30 = 20.9   subiu durante   01:15
17:00 = 21.0   subiu durante   00:30
17:30 = 20.5   desceu durante  00:30
18:00 = 21.5   subiu durante   00:30
```

Saída do programa no caso do exemplo apresentado:

```
08:00 = 19.5
10:30 = 19.4 - desceu durante  02:30
11:30 = 19.3 - desceu durante  01:00
...
18:00 = 21.5 - subiu durante   00:30
A TEMPERATURA SUBIU DURANTE 04:45
```

Escreva o código do referido programa. Use objetos da classe **Time**, descrita na pergunta **3**, para representar os tempos e considere que os **operadores +** e **–**, para somar e subtrair objetos da classe **Time**, foram implementados. Indique os ficheiros de inclusão e o local onde incluiria a definição e implementaçao da classe **Time**, considerando que o código vai ser escrito num único ficheiro "**.cpp**".

NOTA: os tempos registados no ficheiro e a duração dos períodos apenas indicam horas e minutos; considere que os segundos são zero, mas não os apresente nas durações apresentadas no ecrã.

Nome do estudante: _____ **Código UP:** _____ **NºExm:** _____

5. [3.0]

Apresenta-se ao lado a definição parcial de 2 classes usadas para representar livros que contêm apenas texto. A classe **Page** é usada para representar uma página do livro. A classe **Book** tem como atributos (membros-dados) duas estruturas de dados: uma onde são guardadas as páginas do livro e outra onde é guardado o índice remissivo. O índice remissivo de um livro surge habitualmente na parte final do livro e indica para um conjunto de palavras selecionadas, todas as páginas do livro em que essas palavras ocorrem (ex: a palavra "algoritmo" surge nas páginas 10, 75, 213, 331; a palavra "computador" surge nas páginas 25, 26, 50; etc.)

a) [1.0] Declare duas estruturas de dados alternativas que poderiam ter sido usadas para representar as páginas e o índice e diga se considera que alguma delas seria preferível às estruturas usadas. Justifique sucintamente a resposta.

```
class Page {
public:
  Page(int pageNumber);
  bool hasWord(const string &word) const; // a página contém esta palavra?
  int  getPageNumber() const; // retorna o número da página
  void show(ostream &out) const; // mostra o conteúdo da página
  // ... outros métodos
private:
  int pageNumber;  // o número da página
  vector <string> lines; // o conteúdo da página
};
//-----------------------------------------------------------
class Book {
public:
  Book(const string &filename); // cria livro a partir de ficheiro
  void buildIndex(const set<string> &words); // cria índice remissivo
  void showIndex(ostream &out) const; // mostra índice remissivo
  // ... outros métodos
private:
  // ... outros atributos
  vector<Page> pages; // as páginas do livro
  map<string, vector<int>> index; // o índice remissivo
};
```

b) [1.0] Escreva o código do método **buildIndex()** da classe **Book**, o qual deve construir o índice remissivo do livro, guardando-o em **index**. O parâmetro **words** deste método deve conter o conjunto de palavras que farão parte do índice remissivo. Use as estruturas de dados usadas na declaração inicial da classe **Book** (ver introdução do problema).

c) [1.0] Complete o código abaixo da função **main()** de um programa de teste da classe **Book**. O programa deve criar uma variável do tipo **Book** a partir do texto guardado no ficheiro "**C++ Book.txt**", construir o índice remissivo do livro e mostrar o índice no ecrã. Por simplificação, considere que se pretende que o índice remissivo contenha apenas 3 palavras: "computer", "algorithm" e "program". Considere que todos os métodos das classes **Page** e **Book** foram implementados.

```
int main()
```