FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Mestrado em Engenharia Informática e Computação

PROGRAMAÇÃO (EIC0012) – 2011/2012 - 2º semestre

Exame da Época Normal - 2012/06/15

duração: 2h15m; com consulta

NOME DA/DO ESTUDANTE: _____ **Nº:** EI_____

1. [5 valores]

a) [1.5] Um programador que pretendia desenvolver uma função para contar o número de valores iguais a zero num vetor de inteiros escreveu o seguinte código:

```
unsigned int countZeros(vector<int> &v) {
   unsigned int numZeros=0;
   for (size_t i=0; i < v.size() ; i++)
      if (v[i] = 0) numZeros++;
   return numZeros;
}
```

Ao usar esta função concluiu que os resultados não eram os esperados (o valor retornado pela função era sempre igual a zero). Identifique e corrija o erro que o programador cometeu.

Se o programador tivesse usado boas práticas na escrita do seu código, o erro que cometeu poderia ter sido detetado pelo compilador. Explique como e porquê.

O parâmetro da função é passado por valor ou por referência? Justifica-se a forma de passagem usada? Porquê?

b) [1.5] Para converter o valor de uma carta de jogar, representado por um inteiro sem sinal, no nome respetivo, representado por uma **string**, um programador escreveu a seguinte função:

```
string cardRankToSymbol (unsigned int rank) {
  switch (rank) {
      case  1: return " A"; // ás
      case  2: return " 2"; // duque
      // … case 3: ...  case 9:  <--  omitidos para poupar espaço
      case 10: return "10";
      case 11: return " J"; // valete
      case 12: return " Q"; // dama
      case 13: return " K"; // rei
      default : return "Invalid card"; // <-- NOTAR
  }
}
```

Outro programador sugeriu que teria sido possível escrever o código de uma forma mais compacta que evita o uso da instrução **switch**, recorrendo a um *array* de **strings** contendo os símbolos de todas as cartas. Reescreva o código da função de forma a implementar a solução proposta por este programador.

```
string cardRankToSymbol (unsigned int rank) {
   string names[] = {" A"," 2"," 3"," 4"," 5"," 6"," 7"," 8"," 9","10"," J"," Q"," K"};









}
```

O mesmo programador sugeriu que também teria sido possível usar um **map** para fazer a conversão. Reescreva o código da função de forma a usar esta solução. NOTA: não precisa de escrever todas as instruções necessárias para a criação do **map**; basta escrever a instrução de inserção dos primeiros dois elementos do **map**.

c) [1.0] Em C/C++ existem diversas funções para calcular o valor absoluto de um número: **abs()**, **labs()** e **fabs()**. Estas funções só diferem no tipo do seu argumento. Uma alternativa ao uso de diferentes funções seria implementar uma *template function* que calcule o valor absoluto do seu argumento qualquer que seja o seu tipo. Escreva o código dessa *template function*, chamada **absolute**.

d) [1.0] Apresenta-se a seguir a definição de uma classe **String** que foi desenvolvida nas aulas teóricas:

```
class String {
   friend std::ostream& operator<<( std::ostream& out, const String& right);
   friend bool operator==(const String& left, const String& right);
   friend String operator+(const String& left, const String& right);
public:
   String();
   String(const char s[]);
   String(const String& right);
   ~String();
   String& operator=(const String& right);
   char& operator[](int index);
   char operator[](int index) const;
   int length() const;
private:
   char* buffer;
   int len;
};
```

Como justifica o uso do qualificativo **friend** nas funções **operator<<**, **operator==** e **operator+**?

Como justifica a necessidade de implementação do destrutor nesta classe?

Que outros membros-função (ou métodos) de uma classe devem, em geral, ser implementados quando é necessário implementar o destrutor?

NOME DA/DO ESTUDANTE: _____ **Nº:** EI_____

2. [5.5 valores]

Considere a interação com um programa que pretende servir de ajuda aos jogadores do jogo de palavras cruzadas.

```
Dictionary size = 12143

Pattern (letters and dots)? .a.b.
Found: balbo
Found: bambo
Found: garbo
Number of matches = 3

Pattern (letters and dots)? ...XA
Found: bruxa
Found: caixa
Found: faixa
Number of matches = 3

Pattern (letters and dots)? Tarta....
Found: tartamudo
Found: tartaruga
Number of matches = 2

Pattern (letters and dots)? ak.
Number of matches = 0
...
```

NOTAS:

pesquisar palavras com 5 letras
que têm um 'a' (ou 'A') na 2.a letra e
um 'b' (ou 'B') na 4.a letra

pesquisar palavras com 5 letras
que terminam em "XA" (ou "xa")

pesquisar palavras com 9 letras
que começam por "Tarta" (ou "TARTA", ou "tarta", ou …)

A ajuda consiste em procurar, num dicionário, todas as palavras que respeitam um padrão especificado pelo utilizador (ver acima os exemplos e NOTAS) e mostrá-las ao utilizador. Apresenta-se a seguir parte do código do programa.

```
#include ...  // A COMPLETAR, na alínea e)
using namespace std;

const string DICTIONARY_FILENAME = "dictionary.txt";  // ficheiro que contém as palavras do dicionário


_______ match(_____________________________) // função match(); PROTÓTIPO A COMPLETAR
{
  // CÓDIGO A ESCREVER, na alínea d)
}

int main()
{
  string word;
  string pattern;
  vector<string> dictionary;

  //abrir o ficheiro que contém o dicionário
  // CÓDIGO A ESCREVER, na alínea a)

  //ler do ficheiro para o vector
  // CÓDIGO A ESCREVER, na alínea b)

  cout << endl << "Dictionary size = " << dictionary.size() << endl << endl;

  //ler repetidamente um padrão e tentar encontrar palavras que verificam o padrão
  cout << "Pattern (letters and dots)? ";
  while (cin >> pattern) {
    int numMatches = 0;
    for (size_t i=0; i< dictionary.size(); i++)
      if (match(pattern,dictionary[i])) {
        cout << "Found: " << dictionary[i] << endl;
        numMatches++;
      }
      cout << "Number of matches = " << numMatches << endl << endl;
      cout << "Pattern (letters and dots)? ";
  }
  return 0;
}
```

a) [1.5] Escreva o código que tenta abrir o ficheiro que contém todas as palavras do dicionário. Se não for possível abrir o ficheiro, o programa deve apresentar uma mensagem de erro adequada e terminar imediatamente.

b) [1.0] Escreva o código que lê para o vetor **dictionary** o conteúdo do ficheiro de texto. Cada linha do ficheiro contém uma única palavra e não existem linhas vazias.

c) [0.5] Diga quando é que o programa termina.

d) [2.0] Escreva o código da função **match()** que verifica se uma palavra do dicionário corresponde a uma sequência padrão. NOTAS: 1) só faz sentido fazer esta verificação se ambas tiverem o mesmo comprimento; 2) as palavras do dicionário devem estar escritas em letra minúscula, as do padrão podem ser escritas com maiúsculas ou minúsculas.

```
__________ match(_____________________________________________________) {  // protótipo a completar




}
```

e) [0.5] Indique as *include files* necessárias para compilar o programa.

NOME DA/DO ESTUDANTE: _____ **Nº:** EI_____

3. [5.5 valores]

Para desenvolver um jogo de cartas, um programador concebeu, entre outras, as classes **Card** e **Hand** cujas definições parciais se apresentam abaixo.

A classe **Card** representa uma carta:

```
class Card {
  friend ostream& operator<<(ostream& os, const Card& card);
public:
  Card();
  Card(char suit, unsigned int rank, unsigned int points, bool isFaceUp);
  char getSuit() const;             // retorna o naipe
  unsigned int getRank() const;     // retorna o valor
  unsigned int getPoints() const;   // retorna a pontuação da carta
private:
  char suit;            // o naipe da carta: 'C' - copas, 'E' - espadas, 'O' - ouros, 'P' - paus
  unsigned int rank;    // o valor da carta: 1 - ás, 2 - duque, ..., 11 - valete, 12 - dama, 13 - rei
  unsigned int points;  // a pontuação da carta: ás - 11, duque - 2, ..., valete/dama/rei - 10
  bool isFaceUp;        // true = face virada para cima
};
```

A classe **Hand** representa as cartas que um jogador tem na mão:

```
class Hand {
public:
  Hand();
  void addCard(Card c);                 // acrescenta uma carta à mão
  Card getCard();                       // retira uma carta da mão
  void show() const;                    // mostra as cartas da mão, no estado em que cada uma estiver
  unsigned int getPointsTotal() const;  // obtém a pontuação total das cartas da mão
private:
  vector<Card> cards;                   // o conteúdo da mão
};
```

a) [1.0] Implemente o construtor com parâmetros da classe **Card** .

b) [1.0] Implemente o operador **<<** associado à classe **Card**, o qual deve mostrar uma carta da seguinte forma: se a carta estiver com a face virada para cima, mostra o valor da carta e o naipe, num campo com a largura de 4 carateres (ex: `" A.C"` para o ás de copas, `" 5.E"` para o 5 de espadas ou `"10.P"` para o dez de paus); se a carta estiver virada para baixo mostra "XXXX". NOTA: para implementar este operador pode usar a função da alínea 1.b, admitindo que está implementada.

c) [2.5] Escreva o método **getPointsTotal()** da classe **Hand**, tendo em conta que a pontuação de uma mão é calculada da seguinte forma, tal como no jogo *Blackjack*, desenvolvido no âmbito desta unidade curricular: a pontuação é dada pela soma dos pontos das cartas da mão; se a pontuação ultrapassar 21 pontos e a mão contiver pelo menos um "ás", a pontuação desse "ás" passa a ser 1 (um), em vez da sua pontuação normal (11).

```
unsigned int Hand::getPointsTotal() const {




}
```

d) [1] Considere que a classe **Hand** era alterada de forma a que as cartas de uma mão sejam representadas usando um **set** em vez de um **vector**: **set<Card> cards**. Nesta situação, implemente o método **show()** da classe **Hand**. Considere que o operador **<<** associado à classe **Card** está implementado.

NOME DA/DO ESTUDANTE: _____ **Nº:** *EI*_____

4. [4 valores]

Para desenvolver uma aplicação cujo objetivo é ensinar crianças a lidar com dinheiro, é necessário definir uma classe de C++ designada **Purse** (porta-moedas).

Para isso definiu-se, numa *header file*, **coin.h**, uma classe "auxiliar" **Coin** que é usada para representar uma moeda:

```
class Coin
{
public:
  Coin(int v, char u);
  void showDescription() const; //mostra descrição da moeda (ex: 1E para um euro ou 2C para 2 cêntimos)
  double getValue() const; //NOTAR o double: retorna o valor da moeda em euros (ex: 0.01 para 1 cênt.)
private:
  int value;
  char unit; // 'E' para 'euro' ou 'C' para 'cêntimo'
};
```

A classe **Purse** deve representar as moedas contidas no porta-moedas (considere que o porta-moedas apenas contém moedas, não notas) e suportar, entre outras, as seguintes operações:

- Criar um porta-moedas vazio (sem dinheiro).
- Inserir uma moeda – **insertCoin()**.
- Inserir várias moedas, de uma só vez – **insertCoins()**.
- Mostrar as moedas contidas no porta-moedas - **showCoins()**;
- Retornar o montante total contido no porta-moedas – **getTotalAmount()**;
- Retirar uma moeda especificada, retornando a moeda – **removeCoin()**.
- Retirar, de uma só vez, uma ou mais moedas correspondentes a um montante especificado (um valor do tipo *double*), devolvendo as moedas retiradas – **removeAmount()**.
- Trocar uma moeda por duas ou mais moedas que totalizem o mesmo montante, devolvendo as moedas resultantes – **exchangeCoin()**.

NOTAR que algumas destas operações (ex: **removeAmount()**) podem não ser possíveis; deve prever uma forma de os utilizadores da classe obterem essa informação.

SUGESTÃO: antes de iniciar o desenvolvimento, leia todo o enunciado deste problema.

a) [2] Escreva uma *header file* contendo a definição da classe **Purse**. Será valorizado: o uso adequado de "passagem de parâmetros por referência" e do qualificativo "const" nos parâmetros e nos métodos, bem como uma escolha adequada da(s) estrutura(s) de dados usadas para representar a informação. Justifique sucintamente a estrutura de dados escolhida para representar as moedas contidas no porta-moedas.

Conteúdo do ficheiro purse.h (NOTA: não deve incluir o código dos membros-função)

b) [2] Escreva um programa que cria dois objetos, **p1** e **p2**, do tipo **Purse** e executa sobre eles as seguintes operações:

- insere uma moeda de 2 euros em cada um;
- insere as seguintes moedas em **p1**: 5 moedas de 10 cêntimos, 5 moedas de 20 cêntimos, 1 moeda de 50 cêntimos e 1 moeda de 1 euro.
- remove a moeda de 2 euros de **p2**, "solicita" a **p1** que troque essa moeda por moedas mais pequenas e insere estas moedas em **p2**.
- mostra o conteúdo de **p2**.

FIM

JAS/RCS/TPF/ICM