Instituto Superior de Engenharia de Coimbra

Engenharia Informática e de Sistemas

Programação Orientada a Objectos (2020/21)

Exercícios

Ficha 1

Input-Output simples / *strings*
Referências / *Overloading* / Parâmetros com valor por omissão

1. Pretende-se um programa que peça o nome e a idade do utilizador e depois imprima essa informação.

a) Implemente o programa pretendido utilizando apenas o que conhece da linguagem C.

b) Identifique os pontos fracos no código que podem dar origem a valores errados e *runtime errors*.

Objectivos do exercício

- Entender os pontos fracos da utilização de funções *printf* e *scanf* decorrentes da ausência de validação por parte do compilador nos tipos de dados dos parâmetros destas funções.

---

2. Escreva um programa em C++ que peça o nome e a idade do utilizador e depois imprima essa informação. A partir deste exercício e durante o resto do semestre deixa de poder usar as funções *printf*, *scanf* e similares.

a) Escreva o programa pretendido em C++. Não pode utilizar funções *scanf* nem *printf* nem nenhuma outra função da biblioteca C *standard*.

b) Compare a solução com o código do exercício 1 e explique porque é que a versão em C++ é mais robusta em termos de compatibilidade de tipos de dados e menos propensa a *runtime errors*.

Objectivos do exercício

- Tomar conhecimento e entender como se usam os objectos *cin* e *cout*. Neste primeiro contacto não se pretende ainda o domínio completo destes objectos mas apenas o necessário para interagir com o utilizador usando os tipos de dados *built-in*.

- Entender o conceito de *namespaces* através do exemplo *using namespace std*. Tomar conhecimento da biblioteca *iostream* e da declaração *#include <iostream>*
- Entender a razão pela qual a utilização de *cin* e *cout* é intrinsecamente mais segura que os seus análogos *scanf* e *printf* da linguagem C.

---

3. Escreva um programa em C++ que peça o nome e a idade do utilizador e depois imprima essa informação. Não pode utilizar funções scanf nem printf. Também não pode usar matrizes de caracteres.

a) Escreva o programa pretendido em C++ com as restrições que foram enunciadas. Sugestão: utilize objectos da classe *string*.

b) Compare a solução com o código do exercício 2 e identifique as vantagens e desvantagens da utilização dos objectos *string* face às matrizes de caracteres. Explique porque é que a versão usando objectos *string* é menos propensa a erros de memória relacionados com a ultrapassagem de limites de matrizes.

Objectivos do exercício

- Tomar conhecimento e entender como se usam os objectos *string*. Neste primeiro contacto pretende-se apenas que se consiga armazenar e posteriormente obter cadeias de caracteres através desses objectos. A manipulação caracter a caracter fica para outro exercício.
- Tomar conhecimento da ajuda *on-line* e usá-la para obter pormenores acerca da classe *string*.
- Entender a diferença entre matrizes de caracteres de tamanho fixo com as restrições e precauções inerentes às matrizes, e os objectos *string* que gerem automaticamente o seu espaço interno para armazenamento de caracteres.
- Entender as vantagens e desvantagens de utilização de objectos *string* em vez de matrizes de caracteres de tamanho fixo para interacção com o utilizador.

---

4. Escreva o código necessário para que a seguinte função main se execute sem erros.

```
int main(){
   imprime("ola");
   imprime("a idade é: ", 25);
   imprime(100, "euros");
}
```

Objectivos do exercício

- Entender e usar a característica de *overloading* da linguagem C++ e as situações onde se usa.
- Entender as restrições sintácticas de *overloading* e as situações de ambiguidade a evitar.

---

5. Escreva a função ou funções **soma( )** de modo que o programa seguinte corra sem erros.

```
int main(){
   cout << "\n" << soma() << soma(1);
   cout << soma(1,2) << soma(1,2,3);
}
```

a) Utilize funções *overloaded*.

b) Utilize funções com parâmetros com valores por omissão.

c) Tente manter no mesmo programa as funções da alínea a) e b) em simultâneo. Explique a razão do compilador se queixar.

Objectivos do exercício

- Consolidar conhecimento acerca da característica de *overloading* da linguagem C++.
- Entender e usar a característica de parâmetros com valor por omissão da linguagem C++.
- Determinar quando usar *overloading*, parâmetros por omissão, ou ambos.
- Entender as restrições inerentes ao uso simultâneo de *overloading* e parâmetros por omissão.

---

6. Pretende-se uma função que troque os valores de duas variáveis inteiras pertencentes ao contexto de chamada dessa função.
Exemplo

```
int main(){
   int a = 5, b = 10;
   troca(a,b);
   cout << "\na = " << a << "\nb = " << b;
} // aparece a = 10 e b = 5
```

a) Escreva a função pretendida utilizando parâmetros do tipo ponteiro.

b) Explique por que razão são necessários os ponteiros.

c) Escreva a função pretendida utilizando parâmetros do tipo referência.

d) Compare ponteiros com referências a nível de funcionamento e a nível de sintaxe. Visualize a localização dos dados e variáveis na memória do computador. Faça diagramas para o ajudar na visualização.

Objectivos do exercício

- Entender e usar a característica de referências da linguagem C++ e o modo como estas funcionam.
- Compreender as semelhanças e as diferenças entre referências e ponteiros.
- Entender as restrições sintácticas associadas ao uso de referências.
- Entender as situações onde usar e as situações onde não usar referências

---

7. Escreva um programa em C++ que peça o nome completo do utilizador e depois imprima os vários nomes desse utilizador, cada um numa linha diferente. Se um dos nomes do utilizador for "Silva" o programa deve avisar que conhece alguém com esse nome.

Objectivos do exercício

- Dominar a leitura de linhas completas contendo espaços.
- Treinar o processamento de *strings* contendo espaços e extrair palavras individuais.
- Experimentar a comparação de objectos *string*.

---

8. Escreva um programa em C++ que peça palavras ao utilizador. Após cada palavra lida, o programa escreve essa palavra ao contrário no ecrã. Se a palavra for um palíndromo (fica igual ao original quando escrita ao contrário), o programa escreverá à frente "palíndromo". Antes de proceder a nova leitura de palavra o programa deve apresentar a mensagem "carregue em *enter* para prosseguir" e aguardar que seja premida essa tecla. O programa termina quando é escrita a palavra "fim".

Objectivos do exercício

- Treinar o processamento de *strings* caracter a caracter.
- Treinar métodos de efectuar pausas através de objectos *cin*.
- Consolidar competências de consulta da ajuda online do IDE.

---

9. Escreva um programa que leia texto a partir do teclado. Se o utilizador escrever um número por extenso ("um", "dois"…), o programa responderá imprimindo esse número em decimal (1,2…). Se o utilizador tiver escrito um número em formato decimal, então o programa responderá com esse mesmo número por extenso. Lide apenas com números entre 1 e 10 e ignore tudo o que não for número. Antes de passar ao próximo número o programa aguarda que se carregue em *enter*. O programa termina quando for escrita a palavra fim.

Objectivos do exercício

- Treinar a leitura de informação genérica a partir do teclado.
- Conseguir detectar o tipo de informação introduzido (exemplo: texto ou inteiro) e agir em conformidade.
- Tomar conhecimento e usar a classe *istringstream*.
- Treinar a conversão de cadeias de caracter para inteiros usando *instringstream*.

---

10. Escreva um programa que leia um número por extenso ("um", "dois"…) e depois um número inteiro. O programa deverá verificar se o número por extenso corresponde ao número inteiro e indicar a palavra "certo" ou "errado" consoante o caso. De seguida procede a nova leitura de número por extenso / inteiro, sem qualquer pausa. O programa termina quando é escrita a palavra "fim" em vez de um número por extenso.

Objectivos do exercício

- Treinar a leitura de informação genérica a partir do teclado.
- Treinar a conversão de cadeias de caracter para inteiros.
- Lidar com situações em que é necessário controlar a entrada de dados, por exemplo, limpar o *buffer*.
