Programação Orientada a Objetos 2024/2025

Exercícios

Ficha Nº 1

Input/Output simples
Objetos string

1. *Instalação de um IDE*

Instale um IDE. O recomendado é o CLion https://www.jetbrains.com/clion/download.

Verifique que o seu ambiente de desenvolvimento está a funcionar bem testando-o com o programa indicado abaixo.

```
#include <iostream>
using namespace std;
int main() {
    cout<<"Hello World";
    return 0;
}
```

O programa deverá compilar sem erros, e ao executar apresenta “Hello World”.

Caso não tenha ou não conclua a instalação do IDE terá que instalar um fora do tempo de aulas, e como desenrasque para esta aula utilize um compilador *online* (não serve para aulas mais adiante nem para o trabalho prático). Exemplo: https://www.onlinegdb.com/online_c++_compiler

No final do exercício deverá

- Ter um IDE para desenvolvimento em C++, preferencialmente CLion
  Ou
- Um IDE alternativo, mas sabendo que o trabalho prático será entregue como projeto CLion
  Ou
- Não tendo o IDE completamente instalado, tem as informações que lhe permitirão concluir a instalação fora da aula

2. *Input/output introdutório com biblioteca de C++* – Exercício em três partes

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

Parte 1 – Usando C

a) Implemente o programa pretendido utilizando apenas o que conhece da linguagem **C** e as funções biblioteca C standard. Pretende-se apenas uma solução "habitual" parecida com e representativa das que normalmente usaria nos seus programas. O nome pode ser só uma palavra, bastando identificar o que seria necessário fazer se fossem mais do que uma palavra (diga como faria).

- Identifique os pontos fracos no código que podem dar origem a valores errados e *runtime errors*. Explique as fraquezas que identificou.
  - Procure especificamente situações que podem corromper informação ou até fazer o programa terminar.
  - É importante que perceba quais são os pontos fracos (e porquê) de forma a, mais adiante, comparar com a implementação melhor em C++ e perceber como estas situações são evitadas em C++. Verifique junto do professor que identificou todas as situações corretamente.

Parte 2 – Sem *printf* nem *scanf*

b) Escreva o programa usando agora **C++** e as bibliotecas standard desta linguagem**. Não pode utilizar funções *scanf* nem *printf* nem outras da biblioteca C *standard***. Nesta fase do exercício não deverá ainda usar nenhum tipo de dados de biblioteca C++ para armazenar os dados.

- Compare a solução com o código da parte 1 do exercício 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*. Não deverá avançar enquanto não tiver a certeza que compreendeu a razão pela qual esta implementação é melhor que aquela que usou C.

c) Acrescente ao seu programa o necessário para garantir que

i) o valor inteiro introduzido é mesmo um inteiro e com valor positivo (se não for, o programa deve pedir o valor novamente), e que

ii) consegue ler mais do que uma palavra para o nome. Na validação deve usar os mecanismos existentes nos objetos da biblioteca C++ que usou para efetuar a entrada de dados e apenas esses.

Importante: A partir deste ponto, e durante o resto do semestre, deixa de poder usar as funções *printf*, *scanf* e similares. Esta "proibição" aplica-se às aulas, trabalho prático e exames.

- A justificação para esta "proibição" deve ser óbvia com base no que aprendeu na alíneas atrás. Se restarem dúvidas nesse aspeto tire já essa dúvida com o professor antes da aula terminar.

Parte 3 – Sem arrays de caracteres

d) Escreva o programa, mas agora sem usar arrays de caracteres. Sugestão: utilize objetos da classe *string*, usando as informações dadas na aula.

- Compare a solução com o código da versão anterior e identifique as vantagens e desvantagens da utilização dos objetos *string* face às matrizes de caracteres.
  - Por exemplo, quanto ao tamanho previsto do nome que vai ser introduzido.
  - Explique porque é que a versão usando objetos *string* é menos propensa a erros de memória relacionados com a ultrapassagem de limites de arrays.

A partir deste ponto deverá ficar muito claro que usar *arrays* de caracteres para guardar cadeias de caracteres como se fazia em linguagem C é bastante desvantajoso e fortemente desaconselhado. Este "desaconselhamento" aplica-se às aulas, TP e exames. O exercício serve para perceber o porquê.

e) Acrescente ao programa a seguinte funcionalidade:

i) imprimir o número de caracteres do nome.

ii) imprimir uma letra do nome em cada linha. Utilize um for tradicional e depois o "for-each" de C++, mediante a explicação dada acerca deste tipo de ciclo.

Tente obter o máximo de informação acerca de strings antes da explicação do professor.

No final do exercício deverá perceber, de forma clara e concreta, os seguintes tópicos

- Exemplo de alguns dos vários pontos fracos da linguagem C no que diz respeito à possibilidade de cometer erros que o compilador não detete, sendo o uso de scanf/printf um exemplo entre outros possíveis, e por que não deve usar bibliotecas C a que estava habituado.
- O que são *cin*, *cout*, e *cerr*, como se usam, e o que são e significam << e >> nesse contexto.
- Como *cin* e *cout* são um exemplo de como a linguagem C++ evita a necessidade de código propenso a erros.
- Formas simples de uso de *cin* e de validação de dados de entrada.
- O que é a classe *string*, para que serve, quais as suas vantages e como se usa.
- Quais as funções membro principais de classe *string*.
- Como aceder a caracteres individuais de uma *string*, obter o número de caracteres, como percorrer os caracteres de uma *string*.
- Como obter informação acerca das funções membro das classes biblioteca de C++

3. Leitura de uma frase inteira

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 "Fernando" o programa deve avisar que conhece alguém com esse nome.

Requisito de implementação: o nome é lido todo de uma só vez (uma linha de texto). Só depois de lido é que é processado.

a) Verifique rapidamente se consegue cumprir este requisito usando apenas cin >> *objecto-de-string*

b) Utilize um objeto da classe **istringstream** após a introdução dada nesta aula.

No final do exercício deverá

- Ter uma noção acerca de como funciona o buffer de caracteres associado à entrada de dados e como são consumidos por cin >>
- Ter tido uma primeira introdução à classe istringstream e como esta pode ajudar a controlar a forma como os caracteres são consumidos (analogia: sscanf)
- Perceber como o uso de istringstream pode ajudar no controlo da entrada de dados, por exemplo, linhas de texto de uma só vez
- Ter uma noção mais alargada das funções existentes na classe string (ex., para comparação)

4. Análise de caracteres em *strings*

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".

No final do exercício deverá

- Ter experimentado mais cenários de entrada de dados (exemplo, aguardar por *enter*)
- Ter uma noção mais alargada das funções existentes na classe string para acesso a caracteres individuais e mais um exemplo como os operadores habituais de C podem tomar outros significados em C++ (neste caso, [ ])

5. Entrada de dados flexível, sem saber exatamente qual o próximo tipo de dados

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.

No final do exercício deverá

- Ter experimentado estratégias para lidar com entrada de dados de tipo imprevisto
- Consolidado os conhecimentos de exercícios anteriores

6. Leitura de um ficheiro de texto

Modifique o exercício anterior da seguinte maneira

- Inicialmente é pedido o nome de um ficheiro que já deverá existir (pode assumir que o nome do ficheiro não tem espaços).
- Os números que estavam a ser lidos do teclado passam a ser lidos do ficheiro cujo nome indicou.
- O output continua a ser enviado para o ecrã.

Neste exercício deverá usar objetos ***ifstream***, que serão brevemente apresentados na aula.

No final do exercício deverá

- Ter experimentado a forma habitual de ler ficheiros de texto em C++

7. Entrada de dados flexível (consolidação)

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.

No final do exercício deverá

- Ter consolidado estratégias para lidar com entrada de dados de tipo imprevisto