AULA 8 – TIPOS ABSTRATOS DE DADOS

A linguagem C não suporta o paradigma da programação orientada a objetos. No entanto, é possível usar alguns princípios desse paradigma no desenvolvimento de código em C.

O programador pode definir um (novo) tipo de dados – designado como **tipo abstrato**, por ser independente da implementação usada – especificando, inicialmente, as suas operações e, depois, escolhendo uma representação interna apropriada e implementando quer um conjunto de funções (públicas) de interface quer (eventuais) funções auxiliares (privadas).

A interface do tipo abstrato é habitualmente definida num ficheiro cabeçalho (**.h**); no ficheiro de implementação (**.c**) são implementadas as funções que lhe estejam associadas.

O tipo abstrato DATE

Pretende-se concluir o desenvolvimento do tipo abstrato de dados DATE, para registar e operar sobre datas. Esse tipo abstrato é constituído pelo ficheiro de interface **Date.h** e pelo ficheiro de implementação **Date.cpp** (incompleto).

É possível testar as funções desenvolvidas de dois modos:

- usando o sítio **CodeCheck** (http://horstmann.com/codecheck/), completando as funções do tipo abstrato em https://codecheck.io/files/2105081837rccydf7xgvvcgcbxbcceguow e analisando a informação produzida após cada submissão.
- compilando e executando o programa de teste **Tests.cpp**, que permite o **teste incremental** de cada uma das funcionalidades do tipo abstrato. É fornecido um ficheiro **Makefile**, para facilitar o processo de compilação. Após a compilação pode invocar **./Tests** para executar todos os testes. Se preferir, pode invocar **./Tests N**, com N = 1,2,… para executar apenas até ao teste N.

Nota: os ficheiros surgem com a extensão .cpp (e não .c) por compatibilidade com o CodeCheck; mas é usada a linguagem C.

- Comece por analisar o ficheiro **Date.h**, para identificar as funcionalidades disponibilizadas, e o ficheiro **Tests.cpp**, para perceber a sequência de testes que será efetuada.

- **Questões:** como é **representada internamente** cada data? Que funções definidas em Date.h **operam** com / sobre **instâncias** do tipo DATE? Que funções são **funções auxiliares**?

- Analise o ficheiro **Date.cpp**, para verificar o modo como são implementadas as diferentes funções. Há alguma função auxiliar ***"privada"***?

- Use a Makefile para **compilar** o módulo e o programa de teste. Tente perceber o significado dos erros / avisos indicados.

- De modo faseado, **complete e teste** cada uma das **funções incompletas**. Tenha em atenção a especificação de cada função e as suas **pré-condições** e **pós-condições**.

NOME: **Nº MEC:**

O tipo abstrato PERSON

Pretende-se concluir o desenvolvimento do tipo abstrato de dados PERSON, para registar e operar sobre instâncias que registam dados de uma pessoa. Esse tipo abstrato é constituído pelo ficheiro de interface **Person.h** e pelo ficheiro de implementação **Person.cpp** (incompleto).

É possível testar as funções desenvolvidas de dois modos:

- usando o sítio **CodeCheck** (http://horstmann.com/codecheck/), completando as funções do tipo abstrato em https://codecheck.io/files/2105081837rccydf7xgvvcgcbxbcceguow e analisando a informação produzida após cada submissão.
- compilando e executando o programa de teste **Tests.cpp**, que permite o **teste incremental** de cada uma das funcionalidades do tipo abstrato. É fornecido um ficheiro **Makefile**, para facilitar o processo de compilação. Após a compilação pode invocar **./Tests** para executar todos os testes. Se preferir, pode invocar **./Tests N**, com N = 1,2,… para executar apenas até ao teste N.

Nota: os ficheiros surgem com a extensão .cpp (e não .c) por compatibilidade com o CodeCheck; mas é usada a linguagem C.

- Comece por analisar o ficheiro **Person.h**, para identificar as funcionalidades disponibilizadas, e o ficheiro **Tests.cpp**, para perceber a sequência de testes que será efetuada.

- **Questões:** como é **representada internamente** cada instância? que funções definidas em Person.h **operam** com / sobre **instâncias** do tipo PERSON?

- Analise o ficheiro **Person.cpp**, para verificar como são implementadas as diferentes funções.

- Use a Makefile para **compilar** o módulo e o programa de teste. Tente perceber o significado dos erros / avisos indicados.

- De modo faseado, **complete e teste** cada uma das **funções incompletas**. Tenha em atenção a especificação de cada função e as suas **pré-condições** e **pós-condições**.

- Depois de superar todos os testes, execute **valgrind ./Tests** para verificar se tem *"memory leaks"* ou outros problemas relacionados com a alocação dinâmica de memória. Se não tiver problemas deverá obter um relatório semelhante ao abaixo.

```
==4485==                     HEAP                     SUMMARY:
==4485==         in use at exit: 0 bytes in 0 blocks
==4485==   total heap usage: 13 allocs, 13 frees, 4,233 bytes allocated
==4485==
==4485==  All heap blocks were freed -- no leaks are possible
==4485==
==4485== For counts of detected and suppressed errors, rerun with: -v
==4485== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```