AULA 7 – 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**) é definida a representação interna de cada instância do tipo e são implementadas as funções que lhe estejam associadas.

O tipo abstrato TIME

Pretende-se concluir o desenvolvimento do tipo abstrato de dados TIME, para registar e operar sobre instantes de tempo de um período de 24 horas (dia). Esse tipo abstrato é constituído pelo ficheiro de interface **Time.h** e pelo ficheiro de implementação **Time.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 http://codecheck.it/files/2004212404bknhzfn7c3ppho7j6y6s61ilj e analisando a informação produzida após cada submissão.
- compilando e executando o programa de teste **TimeTest.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 **./TimeTest** para executar todos os testes. Se preferir, pode invocar **./TimeTest 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 **Time.h**, para identificar as funcionalidades disponibilizadas, e o ficheiro **TimeTest.cpp**, para perceber a sequência de testes que será efetuada.

- **Questões:** como é **representado internamente** cada instante de tempo? Que funções definidas em Time.h **operam** com / sobre **instâncias** do tipo TIME? Que funções são **funções auxiliares**?

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

  - funçoes auxiliares privadas: invariant

- 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 às suas **pré-condições** e **pós-condições**.

  - representaçao interna Time: horas, minutos, segundos;- operam com instancias Time: TimeGetHH, Tim

NOME: **Nº MEC:**

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 http://codecheck.it/files/20041815118nz44oe6baza9fcrwip6j88qp e analisando a informação produzida após cada submissão.
- compilando e executando o programa de teste **DateTest.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 **./DateTest** para executar todos os testes. Se preferir, pode invocar **./DateTest 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 **DateTest.cpp**, para perceber a sequência de testes que será efetuada.

- **Questões:** que funções definidas em Date.h **operam** com / sobre **instâncias** do tipo DATE? Que funções são **funções auxiliares**?

  - funçoes que operam com/sobre instancias: DateDestroy, DateCompare, Da

- Analise o ficheiro **Date.cpp**, para verificar o modo como é **representada cada instância** e 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 às suas **pré-condições** e **pós-condições**.

- Depois de superar todos os testes, execute **valgrind ./DateTest** 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: 7 allocs, 7 frees, 1,048 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)
```