GUIÃO 07 – TIPOS ABSTRATOS DE DADOS (TADS)

1 – Stacks & Queues

Pretende-se verificar se um número inteiro positivo (base 10), lido do teclado, é uma **capicua**.

Desenvolva um algoritmo, usando uma **pilha de inteiros (stack)** e uma **fila de inteiros (queue)**, que processa os sucessivos algarismos de um número e conclui se esse número é ou não uma capicua.

Analise as funcionalidades dos tipos abstratos **IntegersQueue** e **IntegersStack** disponibilizados e complete o ficheiro **01_example.c**.

2 – O TAD 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.c** (incompleto).

É possível testar as funções desenvolvidas compilando e executando o programa de teste **Tests.c**, que permite o **teste incremental** de cada uma das funcionalidades do tipo abstrato. É fornecido um ficheiro **Makefile**, para facilitar o processo de compilação em **Linux**. 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.

Tarefas

- Comece por analisar o ficheiro **Date.h**, para identificar as funcionalidades disponibilizadas, e o ficheiro **Tests.c**, 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.c**, 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**.

3 – O TAD 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.c** (incompleto).

É possível testar as funções desenvolvidas compilando e executando o programa de teste **Tests.c**, que permite o **teste incremental** de cada uma das funcionalidades do tipo abstrato. É fornecido um ficheiro **Makefile**, para facilitar o processo de compilação em **Linux**. 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.

Tarefas

- Comece por analisar o ficheiro **Person.h**, para identificar as funcionalidades disponibilizadas, e o ficheiro **Tests.c**, 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.c**, 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, e em **Linux**, 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)
```

4 – Stacks & Queues – Exercício adcional

Analise as funcionalidades dos tipos abstratos **PointersQueue** e **PointersStack** disponibilizados.

Usando os tipos abstratos **DATE** e **PERSON** dos exercícios anteriores, desenvolva um exemplo que use **pilhas de ponteiros (stack)** e **filas de ponteiros (queue)** para representar filas e pilhas de **instâncias** desses tipos.

Crie várias instâncias de DATE e de PERSON e adicione-as às correspondentes estruturas de dados.

Retire e imprima, um a um, cada um dos elementos, até que as estruturas de dados fiquem vazias.

Em **Linux**, execute o **valgrind** para verificar se tem *"memory leaks"* ou outros problemas relacionados com a alocação dinâmica de memória.