INSTITUTO SUPERIOR TÉCNICO

Introdução aos Algoritmos e Estruturas de Dados
2012/2013

Enunciado do 1º Projecto

Data de entrega: Sexta-feira, 5 de Abril de 2013 (23h59)

1. Introdução

O objectivo deste projeto é o desenvolvimento de um programa na linguagem C para gestão de um pequeno clube, de nome *Associação XPTO*, que aposta na oferta de várias modalidades desportivas e culturais aos seus associados.

2. Especificação do programa

Apresenta-se em seguida um extrato do regulamento de funcionamento do *XPTO*, obtido a partir do estatuto e dos procedimentos internos da instituição, bem como dados de gestão, de modo a constituir o guia básico do programador.

- O *XPTO* está habilitado a oferecer 26 modalidades codificadas com as referências {`A`, ..., `Z`}.
- O *XPTO* dispõe de uma plataforma integrada através da qual se fazem operações que envolvem a gestão de inscrições e de tesouraria, tais como inscrições nas modalidades, desistências, pagamento de mensalidades, listagem de membros e outras.
- O programa de gestão deverá ter capacidade para registar um máximo de 1000 associados, sendo estes "adeptos" ou "atletas".
- O ato de registo de um novo associado implica o pagamento de 200€. Um associado poderá ter dois estatutos, "adepto" ou "atleta", ficando com o primeiro depois do ato de registo. Um associado possui o estatuto de "atleta" quando está inscrito em pelo menos uma modalidade. Associados sem ligação a qualquer modalidade possuem o estatuto de "adepto", não tendo qualquer encargo mensal com o clube.
- Cada "atleta" possui um encargo mensal de 10€ por mês, por cada modalidade inscrita.
- Cada associado recebe um número de sócio no ato de registo, o qual se inicia em 101 e incrementa automaticamente de uma unidade; este número, uma vez atribuído, nunca mais será recuperado, não estando prevista a eliminação de registos na base de dados do *XPTO*.
- O sistema de gestão deverá registar para cada associado, para além do seu **nome próprio**, **apelido**, **estatuto** e **número de sócio**, o **crédito** atual (ou conta corrente) em euros (€), que regista a diferença entre os valores pagos pelo associado e as despesas do mesmo resultantes da sua ligação ao *XPTO*. Além disso, o sistema deverá registar o número de **meses** que cada associado esteve ligado ao clube.
- No fim de cada mês o programa deverá atualizar todos os valores em dívida para cada atleta, assim como o número de meses que cada associado permaneceu ligado ao clube.

3. Dados de entrada

O programa deverá ler os dados de entrada a partir do standard input, na forma de um conjunto de linhas iniciadas por um carácter, que se passa a designar por comando, seguido de um número variável de informações; o comando e cada uma das informações são separados por um espaço e, dentro de uma informação, não pode haver espaços.

Passamos agora a detalhar os dados de entrada de cada um dos comandos disponíveis — **`r`**, **`i`**, **`k`**, **`l`**, **`+`**, **`p`** e **`x`**. Cada comando indica uma determinada ação que se passa a caracterizar em termos de objectivo, sintaxe e output.

`r` **`r`: registo de novo associado**

```
r <nomeProprio> <apelido>
```

- `nomeProprio`: máximo 50 caracteres (letras, s/ espaços).
- `apelido`: máximo 50 caracteres (letras, s/ espaços).

Observações:
- não implica qualquer output.
- é assumido que o associado pagou o preço de registo (ver secção 1), começando com um crédito (ou saldo) igual a 0. O valor pago no ato de registo deverá ser contabilizado na soma mensal dos pagamentos recebidos pelo *XPTO*.
- Exemplo: `r Carlos Lopes`

`i` **`i`: inscreve um associado numa modalidade**

```
i <numsocio> <modalidade>
```

- `numsocio`: número de sócio, dada por um número inteiro positivo > 100.
- `modalidade`: código da modalidade (`A`, ..., `Z`).

Observações:
- não implica qualquer output.
- É assumido que a primeira mensalidade (10€, ver secção 1) associada à prática de uma modalidade é paga no momento da inscrição, ficando o atleta com um crédito adicional de 10€ correspondente ao primeiro mês. Este valor deverá também entrar nas contas mensais do clube (ver output do comando '+').
- Inscrições envolvendo atletas com crédito negativo deverão ser recusadas.
- Exemplo: `i 123 B`

`k` `k`**: anula a inscrição de um associado numa modalidade**

```
k <numsocio> <modalidade>
```

- `numsocio`: número de sócio, dado por um número inteiro positivo > 100.

- `modalidade`: código da modalidade (`A`, ..., `Z`).

Observações:
- não implica qualquer output.
- Associados que deixam de praticar qualquer modalidade (i.e., atletas com 0 inscrições) passam a ter o estatuto de "adepto".
- Exemplo: `k 123 R`

`l` **`l`: emitir listagem**

```
l <listagem>
```

- `listagem`: número inteiro entre 0 e 3 identificador do tipo de listagem.

Observações:
- Ver Secção 4 para detalhes do output.

`+` **`+`: incrementa 1 mês**

```
+
```

Observações:
- Ver Secção 4 para detalhes do output.
- Incrementa 1 mês, executando os seguintes procedimentos (por esta ordem):
  1. desconta 10€ no crédito de cada "atleta", por cada disciplina inscrita.
  2. Modifica o estatuto de todos os "atletas" com mais de 100€ em divida (i.e., crédito menor que -100€) para o estatuto de "adepto", eliminando todas as inscrições em modalidades que este associado possua.
  3. mostra a soma de todos os pagamentos efectuados no ultimo mês[1] (ver Secção 4).

`p` **`p`: regista pagamento**

```
p <numsocio> <valor>
```

- `numsocio`: número de sócio.
- `valor`: valor (em €) pago pelo associado `numsocio` (número inteiro).

Observações:
- Regista pagamento de `x` € por parte de um associado.
- não implica qualquer output.

`x` **`x`: sair do programa**

```
x
```

Observações:
- Ver secção 4 para detalhes do output.

[1] Não é necessário manter um registo de todos os pagamentos efetuados. Pede-se apenas o montante total pago ao *XPTO* no último mês.

4. Dados de saída

O programa deverá escrever no standard output as respostas aos comandos inseridos no standard input. As respostas são igualmente linhas de texto, com informações separadas por um espaço. Todas as linhas terminam com o carácter `'\n'`.

A informação sobre um associado deverá ser mostrada sempre com o seguinte formato:

`<numsocio> <nome> <apelido> <meses> <crédito> <nModalidades>`

onde

- `numsocio`: número de sócio do associado.
- `nomeProprio`: nome próprio do associado.
- `apelido`: apelido do associado.
- `meses`: número de meses em que o associado esteve ligado ao clube.
- `crédito`: crédito atual do associado.
- `nModalidades`: numero de modalidades atualmente praticadas pelo associado.

`l` **`l`: emitir listagem**

Comandos de listagem:

- `l 0`
  lista todos os associados. A listagem deverá ser ordenada de forma crescente segundo número de sócio.

- `l 1`
  lista os associados com estatuto de "atleta", ou seja, aqueles que estão inscritos em pelo menos uma modalidade. A listagem deverá ser ordenada de forma crescente segundo número de sócio.

- `l 2`
  lista os associados ("atletas" e "adeptos") com crédito (ou "saldo") negativo. A listagem deve ser ordenada de forma decrescente segundo o valor em dívida. Em caso de igualdade no montante em dívida, a listagem deverá seguir o número de sócio de forma crescente.

- `l 3`
  lista de modalidades ativas (i.e., com pelo menos 1 inscrito) por ordem decrescente segundo o número de inscritos. Em caso de igualdade, deverá ordenar as modalidades alfabeticamente, segundo o código da modalidade {`A`, ..., `Z`}.

Nos casos 0, 1 e 2, cada associado deverá ser mostrado seguindo o formato descrito em cima. Cada associado deverá aparecer numa linha distinta.

No caso 3, cada modalidade deverá aparecer numa linha distinta com o seguinte formato:

`modalidade inscritos`

onde `modalidade` corresponde ao código da modalidade (`A`, ..., `Z`) e `inscritos` ao número de associados inscritos nessa modalidade.

`+` **+: incrementa 1 mês**

Output: escreve um número inteiro com o valor total em € pago no ultimo mês.

`x` **x: sair do programa**

Output:
`numAssociados balanco`

escreve um número inteiro com o número total de associados (`numAssociados`) do clube e a soma dos créditos de todos os associados (`balanco`).

5. Exemplos

- **Dados de Entrada 1**

```
r Rosa Mota
r Carlos Lopes
r Fernanda Ribeiro
l 0
+
l 0
+
x
```

- **Dados de Saída 1**

```
101 Rosa Mota 0 0 0
102 Carlos Lopes 0 0 0
103 Fernanda Ribeiro 0 0 0
600
101 Rosa Mota 1 0 0
102 Carlos Lopes 1 0 0
103 Fernanda Ribeiro 1 0 0
0
3 0
```

- **Dados de Entrada 2**

```
r Rosa Mota
r Carlos Lopes
```

```
r Fernanda Ribeiro
r Sebastiao Fonseca
r Jose Soares
r Fernando Mamede
r Eusebio Ferreira
l 0
+
i 101 Z
i 102 A
i 102 B
i 102 Z
i 103 Z
l 1
+
l 1
x
```

- **Dados de Saída 2**

```
101 Rosa Mota 0 0 0
102 Carlos Lopes 0 0 0
103 Fernanda Ribeiro 0 0 0
104 Sebastiao Fonseca 0 0 0
105 Jose Soares 0 0 0
106 Fernando Mamede 0 0 0
107 Eusebio Ferreira 0 0 0
1400
101 Rosa Mota 1 10 1
102 Carlos Lopes 1 30 3
103 Fernanda Ribeiro 1 10 1
50
101 Rosa Mota 2 0 1
102 Carlos Lopes 2 0 3
103 Fernanda Ribeiro 2 0 1
7 0
```

- **Dados de Entrada 3**

```
r Rosa Mota
r Carlos Lopes
r Maria Ribeiro
r Sebastiao Fonseca
r Jose Soares
r Fernando Mamede
r Eusebio Ferreira
i 101 Z
i 102 A
i 102 B
i 102 Z
i 103 Z
+
p 103 10
k 103 Z
+
+
+
```

```
l 2
x
```

- **Dados de Saída 3**

```
1450
10
0
0
102 Carlos Lopes 4 -90 3
101 Rosa Mota 4 -30 1
7 -110
```

- **Dados de Entrada 4**

```
r Rosa Mota
r Carlos Lopes
r Maria Ribeiro
r Sebastiao Fonseca
r Jose Soares
r Fernando Mamede
r Eusebio Ferreira
i 101 Z
i 102 A
i 102 B
i 102 Z
i 103 Z
+
p 103 10
+
+
+
l 3
x
```

- **Dados de Saída 4**

```
1450
10
0
0
Z 3
A 1
B 1
7 -140
```

6. Compilação do Programa

O compilador a utilizar é o `gcc` com as seguintes opções de compilação: `-ansi -Wall -pedantic`. Para compilar o programa deve executar o seguinte comando:

```
$ gcc -ansi -Wall -pedantic -o proj1 *.c
```

o qual deve ter como resultado a geração do ficheiro executável `proj1`, caso não haja erros de compilação. A execução deste comando não deverá escrever qualquer resultado no terminal. Caso a execução deste comando escreva algum resultado no terminal, considera-se que o programa não compilou com sucesso. Por exemplo, durante a compilação do programa, o compilador não deve escrever mensagens de aviso (warnings).

7. Execução do Programa

O programa deve ser executado da forma seguinte:

```
$ ./proj1 < test1.in > test1.myout
```

Posteriormente poderá comparar o seu output com o output previsto usando o comando `diff`

```
$ diff test1.out test1.myout
```

8. Entrega do Projecto

A entrega do projecto deverá respeitar o procedimento seguinte:

- Na página da disciplina aceda ao sistema para entrega de projectos. O sistema será activado uma semana antes da data limite de entrega. Instruções acerca da forma de acesso ao sistema serão oportunamente fornecidas.
- Efectue o upload de um ficheiro arquivo com extensão `.zip` que inclua os ficheiros fonte (`.c`) e cabeçalho (`.h`) que constituem o programa.
- Para criar um ficheiro arquivo com a extensão .zip deve executar o seguinte comando na directoria onde se encontram os ficheiros com extensão .c e .h, criados durante o desenvolvimento do projecto:
  ```
  $ zip proj1.zip *.c *.h
  ```
- Como resultado do processo de upload será informado se a resolução entregue apresenta a resposta esperada num conjunto de casos de teste.
- O sistema não permite submissões com menos de 10 minutos de intervalo para o mesmo grupo. Exemplos de casos de teste serão oportunamente fornecidos.
- Data limite de entrega do projecto: 5 de Abril de 2013 (23h59m). Até à data limite poderá efectuar o número de entregas que desejar, sendo utilizada para efeitos de avaliação a última entrega efectuada. Deverá portanto verificar cuidadosamente que a última entrega realizada corresponde à versão do projecto que pretende que seja avaliada. Não serão abertas excepções.

9. Avaliação do Projecto

- **Componentes da Avaliação**

Na avaliação do projecto serão consideradas as seguintes componentes:

1. A primeira componente avalia o desempenho da funcionalidade do programa realizado. Esta componente é avaliada entre 0 e 16 valores.

2. A segunda componente avalia a qualidade do código entregue, nomeadamente os seguintes aspectos: comentários, indentação, estruturação, modularidade, abstracção, entre outros. Esta componente poderá variar entre -4 valores e +4 valores relativamente à classificação calculada no item anterior e será atribuída na discussão final do projecto. Nesta componente será também utilizado o sistema valgrind for forma a detectar fugas de memória ("memory leaks") ou outras incorrecções no código, que serão penalizadas. Aconselha-se por isso que os alunos utilizem este sistema para fazer debugging do código e corrigir eventuais incorrecções antes da submissão do projecto.

3. Durante a discussão final do projecto será averiguada a participação de cada elemento do grupo na realização do projecto, bem como a sua compreensão do trabalho realizado, sendo a respectiva classificação ponderada em conformidade, isto é, elementos do mesmo grupo podem ter classificações diferentes. Elementos do grupo que se verifique não terem participado na realização do respectivo projecto terão a classificação de 0 (zero) valores.

- **Atribuição Automática da Classificação**

- A classificação da primeira componente da avaliação do projecto é obtida automaticamente através da execução de um conjunto de testes executados num computador com o sistema operativo GNU/Linux. Torna-se portanto essencial que o código compile correctamente e que respeite o formato de entrada e saída dos dados descrito anteriormente. Projectos que não obedeçam ao formato indicado no enunciado serão penalizados na avaliação automática, podendo, no limite, ter 0 (zero) valores se falharem todos os testes. Os testes considerados para efeitos de avaliação podem incluir (ou não) os disponibilizados na página da disciplina, além de um conjunto de testes adicionais. A execução de cada programa em cada teste é limitada na quantidade de memória que pode utilizar, até um máximo de 64 Mb, e no tempo total disponível para execução, sendo o tempo limite distinto para cada teste.

- Note-se que o facto de um projecto passar com sucesso o conjunto de testes disponibilizado na página da disciplina não implica que esse projecto esteja totalmente correcto. Apenas indica que passou alguns testes com sucesso, mas este conjunto de testes não é exaustivo. É da responsabilidade dos alunos garantir que o código produzido está correcto.

- Em caso algum será disponibilizada qualquer tipo de informação sobre os casos de teste utilizados pelo sistema de avaliação automática. A totalidade de ficheiros de teste usados na avaliação do projecto serão disponibilizados na página da disciplina após a data de entrega.