INSTITUTO SUPERIOR TÉCNICO

Introdução aos Algoritmos e Estruturas de Dados

2014/2015 - 2º semestre

Enunciado do 2º Projecto

Data de entrega: 15 de Maio de 2015 (23h59)

1. Introdução

A emissão de cheques, processamento dos mesmos, e reconciliação de contas é um dos processos mais antigos do nosso sistema financeiro.

Neste projecto pretendemos desenvolver um programa em C que registe a emissão de cheques, efectue o seu processamento, e reconcilie as contas dos intervenientes após esse processamento.

2. Especificação do Programa

O nosso programa será constituído por duas entidades chave.

A entidade *cliente* é caracterizada por um identificador (ou referência). A entidade *cheque* é composta pelos atributos identificador (ou referência), valor, e dois identificadores de cliente correspondentes ao emitente e beneficiário do cheque. Os atributos de um cheque são fixados aquando da emissão do mesmo.

Um cheque ao ser emitido é colocado numa "pool" de cheques por processar. Todos os cheques têm identificadores distintos. Ao ser dada a ordem de processamento, este será removido da pool de processamento. Os cheques são normalmente processados por ordem de emissão, mas podem esporadicamente ser processados por outra ordem.

O utilizador pode pedir ao programa listagens dos cheques ainda por processar, bem como informação sobre os valores associados a cada cliente na pool de cheques por processar.

3. Sugestão de Estrutura de Dados

Há várias estruturas de dados passíveis de ser utilizadas para representar o problema. Dado que este sistema terá de processar um grande volume de cheques (e clientes), **a eficiência das mesmas em termos de procura e redimensionamento** é um factor a levar em atenção aquando da escolha da solução de implementação e **será levada em conta na avaliação do projecto**.

4. 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 uma sequência de caracteres, 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 assim, dentro de cada informação, não poderão existir outros espaços.

Os comandos disponíveis são os seguintes:

- **`cheque valor refe refb refc`**

onde **`valor`** é um inteiro positivo correspondente ao valor do cheque, e **`refe`**, **`refb`**, e **`refc`** correspondem respectivamente às referências do cliente emissor, cliente beneficiário, e cheque (também um inteiro longo).

Adiciona um novo cheque à pool de cheques a processar. Pode assumir que nunca serão criados cheques com referências de outros cheques (ainda) por processar. Não imprime nenhum output.

- **`processa`**

Processa o cheque emitido há mais tempo, e remove o cheque da pool de cheques a processar. Não imprime qualquer output ***excepto*** se a pool de cheques for vazia. Neste caso deverá imprimir a informação

```
Nothing to process
```

- **`processaR refc`**

onde **`refc`** é a referência de um cheque.

Processa o cheque com a referência **`refc`**, e remove o cheque da pool de cheques a processar. Não imprime qualquer output ***excepto*** se não existir nenhum cheque com a referida referência. Neste caso deverá imprimir a informação

```
Cheque <refc> does not exist
```

onde `<refc>` é a referência do cheque em causa.

- **`infocheque ref`**

onde **`ref`** é a referência de um cheque.

Imprime a informação referente ao cheque com a referência **`ref`**. Ver Secção 5 sobre o formato do output. Pode assumir que só serão listados cheques da pool de cheques a processar.

- **`infocliente ref`**

onde **`ref`** é a referência de um cliente.

Imprime a informação (sumarizada) referente ao cliente com a referência **`ref`**. Ver Secção 5 sobre o formato do output. Pode assumir que só serão listados clientes activos.[1]

[1] Um cliente do sistema diz-se *activo* se for emitente ou beneficiário de um cheque por processar.

- **info**

Imprime a informação (sumarizada) sobre todos os clientes activos do sistema. Ver Secção 5 sobre o formato do output. Se não existirem clientes activos devera imprimir a frase

```
No active clients
```

- **sair**

Sai do programa e apaga toda a informação sobre o sistema. Imprime o output

```
<ncl> <nch> <vch>
```

onde `<ncl>` é o numero de clientes activos do sistema, e `<nch>` e `<vch>` são respectivamente o número e o valor total de cheques por processar.

5. Dados de Saída

Todos os dados de saída deverão ser escritos no standard output. As respostas são igualmente linhas de texto, com informações separadas por um espaço, não podendo haver espaços em cada uma das informações. Cada frase deverá ser apresentada numa linha distinta.
O comando **cheque** não imprime nenhum output. Os comandos **processa**, e **processaR** e **sair** imprimem output conforme apresentado na Secção 4. Os comandos **infoX** deverão apresentar o seu output conforma se indica a seguir:

infocheque ref: deverá imprimir a frase

```
Cheque-info: <ref> <valor> <refe> --> <refb>
```

onde `<ref>`, `<valor>`, `<refe>`, e `<refb>` são respectivamente a referência do cheque, o valor do cheque, a referência do cliente emissor, e a referência do cliente beneficiário.

infocliente ref: deverá imprimir a frase

```
Cliente-info: <ref> <nche> <vche> <nchb> <vchb>
```

onde `<ref>`, `<nche>`, `<vche>`, `<nchb>` e `<vchb>` são respectivamente a referência do cliente, o número e o valor total dos cheques emitidos pelo cliente que ainda estão por processar, e o número e o valor total dos cheques que ainda estão por processar em que cliente é beneficiário.

info: deverá imprimir a frase

```
*<ref> <nche> <vche> <nchb> <vchb>
```

para todos os clientes activos (com a semântica igual a **infocliente**). Esta listagem deverá estar ordenada por ordem crescente de **ref**.

6. Exemplos (Input/Output)

Exemplo 1

Dados de Entrada:

```
cheque 50 1 2 1
cheque 100 2 3 2
cheque 200 1 3 3
cheque 300 1 4 4
processa
processa
info
sair
```

Dados de Saída:

```
*1 2 500 0 0
*3 0 0 1 200
*4 0 0 1 300
3 2 500
```

Exemplo 2

Dados de Entrada:

```
cheque 50 1 2 1
processa
processa
cheque 100 2 3 2
cheque 200 1 3 3
cheque 300 4 5 4
cheque 400 6 7 5
cheque 501 8 9 6
cheque 602 9 1 7
cheque 703 1 4 8
processa
processa
processaR 1
infocheque 7
processaR 8
infocheque 7
cheque 808 1 4 9
infocheque 9
info
sair
```

Dados de Saída:

```
Nothing to process
Cheque 1 does not exist
Cheque-info: 7 602 9 --> 1
Cheque-info: 7 602 9 --> 1
Cheque-info: 9 808 1 --> 4
*1 1 808 1 602
*4 1 300 1 808
*5 0 0 1 300
*6 1 400 0 0
*7 0 0 1 400
*8 1 501 0 0
*9 1 602 1 501
7 5 2611
```

Exemplo 3

Dados de Entrada:

```
cheque 50 1 2 1
processa
cheque 100 2 3 2
cheque 200 1 3 3
processa
infocliente 3
infocliente 3
```

Dados de Saída:

```
Cliente-info: 3 0 0 1 200
Cliente-info: 3 0 0 1 200
*4 1 300 0 0
*5 0 0 1 300
*6 1 400 0 0
*7 0 0 1 400
*8 1 501 0 0
```

```
cheque 300 4 5 4
cheque 400 6 7 5
cheque 501 8 9 6
processa
info
cheque 602 9 1 7
cheque 703 1 4 8
infocliente 9
processa
processa
processa
processa
infocliente 4
processa
info
sair
```

```
*9 0 0 1 501
Cliente-info: 9 1 602 1 501
Cliente-info: 4 0 0 1 703
No active clients
0 0 0
```

Exemplo 4

Dados de Entrada:

```
info
cheque 50 1000 2000 4
infocliente 2000
processa
info
cheque 100 2001 3002 9
cheque 200 3003 3002 150
processa
infocliente 3003
infocliente 3003
cheque 300 4004 3002 4003
cheque 400 5005 3002 5005
cheque 501 1 2 10001
processa
info
cheque 602 90000000 3001 6006
cheque 703 1 3 5004
infocliente 90000000
processaR 10001
processaR 150
processaR 4003
processa
infocliente 3001
info
cheque 50 1000 2000 4
cheque 100 2001 3002 9
cheque 200 3003 3002 150
cheque 300 4004 3002 4003
infocheque 4
infocliente 3002
info
infocliente 4004
sair
```

Dados de Saída:

```
No active clients
Cliente-info: 2000 0 0 1 50
No active clients
Cliente-info: 3003 1 200 0 0
Cliente-info: 3003 1 200 0 0
*1 1 501 0 0
*2 0 0 1 501
*3002 0 0 2 700
*4004 1 300 0 0
*5005 1 400 0 0
Cliente-info: 90000000 1 602 0 0
Cheque 150 does not exist
Cliente-info: 3001 0 0 1 602
*1 1 703 0 0
*3 0 0 1 703
*3001 0 0 1 602
*90000000 1 602 0 0
Cheque-info: 4 50 1000 --> 2000
Cliente-info: 3002 0 0 3 600
*1 1 703 0 0
*3 0 0 1 703
*1000 1 50 0 0
*2000 0 0 1 50
*2001 1 100 0 0
*3001 0 0 1 602
*3002 0 0 3 600
*3003 1 200 0 0
*4004 1 300 0 0
*90000000 1 602 0 0
Cliente-info: 4004 1 300 0 0
10 6 1955
```

7. 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 proj2 *.c
```

o qual deve ter como resultado a geração do ficheiro executável **`proj2`**, 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).

8. Execução do Programa

O programa deve ser executado da forma seguinte:

```
$ ./proj2 < test01.in > test01.myout
```

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

```
$ diff test01.out test01.myout
```

9. 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 todos os ficheiros fonte que constituem o programa. Se utilizar ficheiros cabeçalho (**`.h`**) não se esqueça de os juntar também ao pacote.

- 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`** (se for o caso), criados durante o desenvolvimento do projecto:
  ```
  $ zip proj2.zip *.c *.h
  ```

  Se só tiver um único ficheiro (e.g., **`proj2.c`**), bastará escrever:
  ```
  $ zip proj2.zip proj2.c
  ```

- 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: **15 de Maio de 2015 (23h59m)**. Até à data limite poderá efectuar o número de submissões que desejar, sendo utilizada para efeitos de avaliação a última submissão efectuada. Deverá portanto verificar cuidadosamente que a última submissão corresponde à versão do projecto que pretende que seja avaliada. Não existirão excepções a esta regra.

10. Avaliação do Projecto

a. 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, **adequação das estruturas de dados escolhidas**, 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`** por 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. A respectiva classificação será 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.

b. Atribuição Automática da Classificação

- A classificação da primeira componente da avaliação do projecto é obtida através da execução *automática* de um conjunto de testes 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 poderão 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 disponibilizado 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.