Processamento de Linguagens (3º Ano de Mestrado Integrado em Engenharia Informática)

Trabalho Prático 1

Relatório de Desenvolvimento

André Ricardo Covelo Germano
(A71150)

André Rodrigues Freitas
(A74619)

Sofia Manuela Gomes de Carvalho
(A76658)

15 de Março de 2017

Resumo

Este trabalho prático tem como principais objetivos aprofundar e aplicar conhecimentos obtidos durante as aulas teóricas e práticas de Processamento de Linguagens. Além de pretender aumentar a capacidade de escrever *Expressões Regulares (ER)* para descrição de *padrões de frases* e de, através destas, desenvolver *Processadores de Linguagens Regulares* que filtrem ou transformem textos, pretende também utilizar o sistema de produção para *filtragem de texto* GAWK.

Foi escolhido um de quatro enunciados disponíveis para este primeiro trabalho prático, mais concretamente, o referente ao Processador de tansações da Via Verde. Os resultados obtidos estão de acordo com os objetivos previamente definidos e com o que foi pedido no enunciado.

Conteúdo

1 Introdução **2**
1.1 Processador de transações da Via Verde 2

2 Análise e Especificação **3**
2.1 Descrição informal do problema 3
2.2 Especificação do Requisitos 3

3 Conceção/desenho da Resolução **4**
3.1 Algoritmos 4
3.1.1 Calcular o número de 'entradas' em cada dia do mês 4
3.1.2 Escrever a lista de locais de 'saída' 4
3.1.3 Calcular o total gasto no mês 4
3.1.4 Calcular o total gasto no mês apenas em 'parques' 5
3.1.5 Escrever a lista de locais de 'saída' de 'parques' 6
3.1.6 Determinar o 'operador' mais utilizado e o número de vezes que foi utilizado 6

4 Codificação e Exemplos **7**
4.1 Alternativas, Decisões e Problemas de Implementação 7
4.2 Testes realizados e Resultados 7

5 Conclusão **9**

A Código do Programa **10**

Capítulo 1

Introdução

Supervisor: André Ricardo Covelo Germano, André Rodrigues Freitas, Sofia Manuela Gomes de Carvalho

1.1 Processador de transações da Via Verde

Área: Processamento de Linguagens

Neste primeiro trabalho prático, foram-nos fornecidos quatro diferentes enunciados, ficando ao nosso critério escolher um deles para realizar. A nossa opção recaiu sobre o primeiro enunciado, que consiste em implementar um processador de transações da Via Verde.

A Via Verde envia a cada um dos seus utentes um extrato mensal no formato XML. Como tal, foram-nos fornecidos dados de circulação da Via Verde de um dado utente, referentes aos meses de julho e agosto de 2015. Depois de analisarmos com cuidado o formato desse ficheiro anexo, desenvolvemos um processador de texto com o GAWK para ler um extrato mensal da Via Verde e retirar dele toda a informação pretendida.

Estrutura do Relatório

Este relatório está organizado em cinco capítulos principais que surgem após um pequeno resumo deste trabalho e são seguidos de anexos em que é apresentado todo o código desenvolvido, com todos os resultados obtidos.

O capítulo 1 consiste numa introdução em que é feita a descrição dos objetivos propostos pelo enunciado escolhido e onde se encontra a estrutura do relatório.

No capítulo 2, é inicialmente feita uma descrição informal do problema proposto, traçando as linhas gerais do mesmo e de seguida são especificados os requisitos concretos que devem ser cumpridos.

De seguida, o capítulo 3 engloba todas as opções tomadas ao longo da realização do trabalho prático, assim como as decisões que lideraram o desenho da solução e da sua implementação.

Por fim, o capítulo 5 inclui uma síntese do trabalho realizado com uma análise crítica dos resultados obtidos, possíveis melhorias e ainda trabalho futuro que poderia ser desenvolvido.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

O enunciado do problema 1 (Processador de transações da Via Verde) tem associado a si um ficheiro anexo em XML de um extrato de um utente e pretende que seja desenvolvido um processador de texto com o GAWK, tal como referido anteriormente neste documento, e partindo dos requisitos fundamentais que estão especificados na secção seguinte.

2.2 Especificação do Requisitos

Os requisitos fundamentais deste enunciado são:

a) Calcular o número de 'entradas' em cada dia do mês;

b) Escrever a lista de locais de 'saída';

c) Calcular o total gasto no mês;

d) Calcular o total gasto no mês apenas em 'parques'.

Os requisitos opcionais são, por exemplo:

e) Escrever a lista de locais de 'saída' de 'parques';

f) Determinar o 'operador' mais utilizado e o número de vezes que foi utilizado.

Como se pode ver pela segunda lisra de requisitos, decidimos acrescentar mais algumas alíneas ao enunciado que nos pareceram ter conteúdo potencialmente relevante.

Capítulo 3

Conceção/desenho da Resolução

3.1 Algoritmos

Para todas as alíneas deste enunciado, declaramos o separador de campo (FS) como sendo ">"ou "</", sendo isso sempre feito dentro de um bloco BEGIN. No apêndice A encontra-se o código correspondente a cada alínea.

3.1.1 Calcular o número de 'entradas' em cada dia do mês

Para atingir este requisito, como já foi especificado acima, começamos por alterar o separador de campo.

De seguida, selecionamos as linhas em que o primeiro campo ($1) da linha correspondia a "DATA_ENTRADA"e à medida que alguma linha fazia *match*, guardávamos num vetor o número de ocorrências de dada "DATA_ENTRADA", em que os índices do vetor correspondem à data em questão.

Por fim, num bloco END, imprimimos cada indíce do vetor, que corresponde à data de entrada, seguido dos valores guardados em cada posição do vetor, que correspondem ao número de ocorrências de cada uma das datas imprimidas.

3.1.2 Escrever a lista de locais de 'saída'

O primeiro passo dado nesta alínea envolveu, também, a alteração do FS.

Para além disto, procuramos as linhas que faziam *match* com "<SAIDA", visto querermos os locais de 'saída'. O "<"antes de "SAIDA"deve-se à existência de mais campos no ficheiro de formato XML que possuem a expressão "SAIDA"como, por exemplo, o caso de "DATA_SAIDA", e que também seriam alvo do *match* caso não tivéssemos colocado o "<"antes. Assim, garantimos que apenas estamos a encontrar a informação referente aos locais de 'saída'. Neste requisito, recorremos a uma forma semelhante à usada no requisito anterior, ou seja, sempre que alguma linha fazia *match*, essa saída era armazenada como sendo o índice de um vetor, no entanto, não guardamos informação sobre o número de vezes que cada saída aparecia, visto que essa informação não era pedida.

No bloco END, imprimimos todos os índices do vetor, pois correspondem aos locais propriamente ditos. Através deste método foi também possível evitar que se imprimissem locais repetidos, obtendo assim uma lista sem repetições.

3.1.3 Calcular o total gasto no mês

Nesta alínea, tivemos em conta o facto de as despesas da Via Verde, embora pertençam todas ao extrato do mês de agosto, terem sido feitas em meses distintos e, por isso, decidimos definir um método que determinasse o total gasto em cada um dos meses. Para calcular o total gasto num mês, decidimos procurar pelas "DATA_SAIDA", pois não tinha valores *null*, ao contrário da "DATA_ENTRADA", e permitindo-nos assim identificar inequivocamente o mês da despesa em causa.

Alteramos o FS e sempre que o primeiro campo ($1) de alguma linha correspondia a "DATA_SAIDA", recorremos a uma função de manipulação de *strings*: o *split*. Com o auxílio desta, separamos as linhas pelo -", armazenando

no vetor "mes"os valores do segundo campo ($2) das linhas com as datas de saída, ou seja, guardando na primeira posição o dia, na segunda o mês e na terceira o ano em que foi feita a saída.

Para o total gasto, começamos por substituir as ocorrências da vírgula por um ponto quando o valor do primeiro campo ($1) da linha corresponde a "IMPORTANCIA". Esta substituição foi necessária, pois o GAWK não reconhece um valor como *float* se este estiver separado por vírgula, não somando assim as casas decimais do valor do segundo campo ($2) das linhas em que o primeiro campo corresponde a "IMPORTANCIA"($1∼/IMPORTANCIA/). Após esta substituição, foi acumulado num vetor em que os índices correspondem ao segundo campo do vetor "mes", ou seja, os índices correspondem ao mês da data de saída, o valor da importância (campo 2 das linhas em que o primeiro campo faz *match* com "IMPORTANCIA").

Como existe um campo "VALOR_DESCONTO", assumimos que esse valor poderia ser descontado à importância e, através desssa subtração, seria obtido o valor a pagar realmente pelo utente. Por isso, decidimos guardar também esse valor. Tal como para a "IMPORTANCIA", foi necessário substituir as ocorrências da vírgula por um ponto quando o valor do primeiro campo ($1) da linha corresponde a "VALOR_DESCONTO", exatamente pelas razões já referidas no parágrafo anterior. Após esta substituição, foi acumulado num vetor em que os índices correspondem ao segundo campo do vetor "mes", ou seja, os índices correspondem ao mês da data de saída, o valor do desconto (campo 2 das linhas em que o primeiro campo faz *match* com "VALOR_DESCONTO").

No bloco END, guardamos na variável "total"a diferença entre a soma dos valores da importância e a soma dos valores do desconto para depois ser imprimida como o total real pago pelo utente. Imprimimos também o índice do vetor e o valor da posição, ou seja, o mês ("07"ou "08", consoante seja julho ou agosto) e o valor total gasto nesse mês, respetivamente. Ao lado de cada importância imprimimos o desconto a aplicar a esse valor.

A soma dos totais dos dois meses em causa ("total") corresponde ao total que está no fim do ficheiro XML, comprovando a correção desta resolução.

3.1.4 Calcular o total gasto no mês apenas em 'parques'

Tal como na alínea anterior, tivemos em conta o facto de as despesas terem ocorrido em meses diferentes, mesmo pertencendo ao extrato de um único mês e, por isso, usamos um método idêntico escolhendo as "DATA_SAIDA"para identificar os meses, uma vez que estas não possuem valores *null*.

O separador de campo (FS) foi, mais uma vez, alterado num bloco BEGIN e de seguida fez-se uma manipulação de *strings* semelhante à da alínea anterior. Através da utilização da função *split*, separamos as linhas pelo -", armazenando no vetor "b"os valores do segundo campo ($2) das linhas com as datas de saída, ou seja, guardando na primeira posição o dia, na segunda o mês e na terceira o ano em que foi feita a saída. Logo a seguir, foram guardados no "mes"apenas os meses, ou seja, os valores da segunda posição do vetor "b".

Para determinar o que foi gasto, foi necessário ter em conta a especificação da alínea que desta vez pedia o total gasto apenas em 'parques'. Começamos por usar a função *sub* para substituir as ocorrências da vírgula por um ponto quando o valor do primeiro campo ($1) da linha corresponde a "IMPORTANCIA". Esta substituição foi necessária, pois o GAWK não reconhece um valor do tipo *float* se este estiver separado por vírgula, não tendo em conta assim as casas decimais do valores das importâncias, o que levaria a que apenas fosse selecionada a parte inteira e, em cálculos como adições, isso iria conduzir a resultados incorretos. Após esta substituição, foi guardada numa variável "valor"a quantia da importância, ou seja, do segundo campo das linhas em que o primeiro campo faz *match* com "IMPORTANCIA".

Como existe um campo "VALOR_DESCONTO", assumimos que esse valor poderia ser descontado à importância tal como na alínea anterior. Tal como para a "IMPORTANCIA", foi necessário substituir as ocorrências da vírgula por um ponto quando o valor do primeiro campo ($1) da linha corresponde a "VALOR_DESCONTO", exatamente pelas razões já referidas no parágrafo anterior. Após esta substituição, foi guardada numa variável "desc"a quantia do desconto, ou seja, do segundo campo das linhas em que o primeiro campo faz *match* com "VALOR_DESCONTO".

Como apenas queremos as despesas correspondentes a 'parques', foram ainda procuradas as linhas cujo primeiro campo corresponde a "TIPO"e o valor do segundo campo ($2) destas linhas foi guardado no "tipo". Através de uma condição usando um *if*, foi possível apenas considerar as despesas cujo "tipo"era "Parques de estacionamento"e assim foram acumulados no vetor "a"os valores das despesas correspondentes apenas a 'parques', sendo que em cada posição "mes"está o total gasto em 'parques' nesse mês. O mesmo foi feito para um vetor "desconto"que guarda informação de forma semelhante ao vetor "a", mas para os descontos.

No bloco END, guardamos na variável "total"a diferença entre a soma dos valores da importância e a soma dos valores do desconto para depois ser imprimida como o total real pago pelo utente. Imprimimos também o índice do vetor

e o valor da posição, ou seja, o mês ("07"ou "08", consoante seja julho ou agosto) e o valor total gasto nesse mês, respetivamente. Ao lado de cada importância imprimimos o desconto a aplicar a esse valor.

Caso seja feita a soma dos totais dos dois meses em causa ("total"), verifica-se que corresponde efetivamente ao total gasto em 'parques' que se iria obter pela inspeção detalhada do ficheiro.

3.1.5 Escrever a lista de locais de 'saída' de 'parques'

A alteração do separador campo (FS) voltou a ser efetuada nesta nova alínea, tendo depois sido guardada numa nova variável, chamada "saida", os valores do segundo campo ($2) em que o primeiro campo fazia *matching* com "<SAIDA", ou seja, a variável "saida"contém todas as saídas presentes neste ficheiro anexado.

De seguida, quando o primeiro campo de uma linha corresponde a "TIPO", se o segundo campo corresponde a "Parques de estacionamento", guardam-se apenas os locais de saída de parques nos índices do vetor "a".

O bloco END permite imprimir os locais de saída de parques guardados anteriormente, ou seja, os índices do vetor "a".

3.1.6 Determinar o 'operador' mais utilizado e o número de vezes que foi utilizado

Na última alínea, assim como em todas as outras, o primeiro passo passou por alterar e redefinir o separador de campo.

Seguidamente, é armazenado num vetor o número de vezes que um determinado operador aparece quando o primeiro campo de alguma linha corresponde a "OPERADOR", sendo o índice desse vetor o operador propriamente dito que consta no segundo campo da linha que faz *match*.

Por fim, no bloco END, é feita uma condição if dentro de um ciclo *for* que permite determinar qual o operador com maior número de ocorrências no vetor criado. Após isto, é impresso o operador mais utilizado ("op") e o número de vezes que foi utilizado ("m").

Capítulo 4

Codificação e Exemplos

4.1 Alternativas, Decisões e Problemas de Implementação

Relativamente às alíneas a) e b), a implementação foi relativamente simples e a decisão do que fazer foi imediata, tendo ficado apenas a dúvida se seria possível realizar a alínea b) sem estar a acumular o número de vezes que cada data aparecia, visto isso ser informação desnecessária que não foi usada e não era pedida.

Já as alíneas c) e d) suscitaram-nos mais dúvidas e passsaram por fases diferentes. Inicialmente tínhamos optado por apenas somar o total das importâncias sem ter em conta o mês, mas achamos que seria pouco coerente com a informação que tínhamos disponível no ficheiro e face ao enunciado ter deixado uma oportunidade de expansão, optamos por realizar estas alíneas com o método já referido no capítulo anterior, dividindo nos meses efetivos das despesas e não nos focando apenas no facto de o extrato ser do mês de agosto. Deparamo-nos também com o facto de as somas de valores não estarem a dar resultados esperados, o que se devia à vírgula que não corresponde ao marcador da casas decimais do GAWK e que levou a que alterássemos quando necessário para um ponto, pois já é reconhecido. Posteriormente, reparamos ainda na existência de um campo com um valor de desconto que poderia ser subtraído à importância, reduzindo o montante a pagar pelo utente e implementamos também essa funcionalidade.

As alíneas e) e f) foram realizadas segundo a sugestão do enunciado de acrescentar mais requisitos e não causaram grandes adversidades.

4.2 Testes realizados e Resultados

Mostram-se a seguir alguns testes feitos e os respetivos resultados da execução no terminal das soluções desenvolvidas, usando o ficheiro *viaverde.xml* que serviu de base ao enunciado.

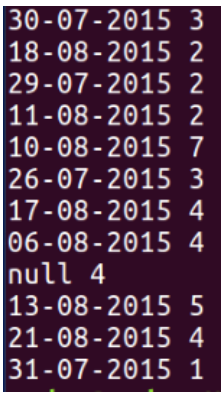

Figura 4.1: Exemplo de calcular o número de 'entradas' em cada dia do mês

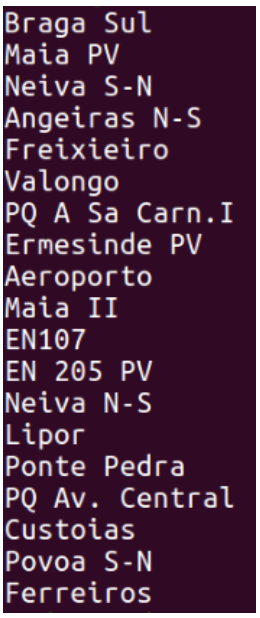

Figura 4.2: Exemplo de escrever a lista de locais de 'saída'

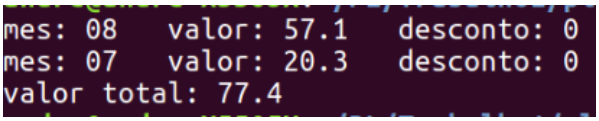

Figura 4.3: Exemplo de calcular o total gasto no mês

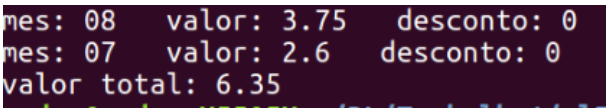

Figura 4.4: Exemplo de calcular o total gasto no mês apenas em 'parques'

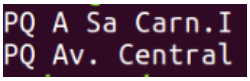

Figura 4.5: Exemplo de escrever a lista de locais de 'saída' de 'parques'

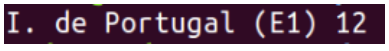

Figura 4.6: Exemplo de determinar o 'operador' mais utilizado e o número de vezes que foi utilizado

Capítulo 5

Conclusão

Em suma, este projeto permitiu-nos aprofundar os nossos conhecimentos na área do Processamento de Linguagens obtidos durante as aulas, especialmente os da ferramenta de processamento de texto GAWK.

Julgamos que conseguimos cumprir os objetivos definidos no enunciado e até com um pouco mais de profundidade apesar de termos atravessado algumas dificuldades naturais da aprendizagem e aplicação de conceitos recentes. Era possível expandir ainda mais os objetivos e requisitos, pois a quantidade de informação sobre a Via Verde presente no documento é extensa e permite várias combinações de informação úteis quer para o utente em questão, quer para a empresa.

Apêndice A

Código do Programa

Mostra-se a seguir o código desenvolvido para as várias alíneas e que está explicado na secção de Algoritmos do capítulo 3.

3.1.1 Calcular o número de 'entradas' em cada dia do mês

```
BEGIN{
   FS = ">|</"
}

$1~/DATA_ENTRADA/ {a[$2]++}

END{
   for (i in a) print i, a[i]
}
```

3.1.2 Escrever a lista de locais de 'saída'

```
BEGIN{
   FS = ">|</"
}

$1~/<SAIDA/ {a[$2]}

END{
   for (i in a) print i
}
```

3.1.3 Calcular o total gasto no mês

```
BEGIN{
   FS = ">|</"
}

$1~/DATA_SAIDA/ {split($2, mes, "-")}
$1~/IMPORTANCIA/ {sub(",","."); a[mes[2]]+=$2}
$1~/VALOR_DESCONTO/ {sub(",","."); desconto[mes[2]]+=$2}

END{
   for(i in a){total+=a[i]-desconto[i]; print "mes:", i,"  valor:", a[i],"  desconto:", desconto[i]};
   print "valor total:", total;
}
```

3.1.4 Calcular o total gasto no mês apenas em 'parques'

```
BEGIN{
   FS = ">|</"
}

$1~/DATA_SAIDA/ {split($2,b,"-"); mes=b[2]}
$1~/IMPORTANCIA/ {sub(",","."); valor=$2}
$1~/VALOR_DESCONTO/ {sub(",","."); desc=$2}
$1~/TIPO/ {tipo=$2; if(tipo=="Parques de estacionamento"){a[mes]+=valor; desconto[mes]+=desc}}

END{
   for(i in a){total+=a[i]-desconto[i]; print "mes:", i,"  valor:", a[i],"  desconto:", desconto[i]};
   print "valor total:", total;
}
```

3.1.5 Escrever a lista de locais de 'saída' de 'parques'

```
BEGIN{
   FS = ">|</"
}

$1~/<SAIDA/ {saida=$2}
$1~/TIPO/ {if($2=="Parques de estacionamento") a[saida]}

END{
   for (i in a) print i
}
```

3.1.6 Determinar o 'operador' mais utilizado e o número de vezes que foi utilizado

```
BEGIN{
   FS = ">|</"
}

$1~/OPERADOR/ {a[$2]++}

END{
   for (i in a) {if(a[i]>m){m=a[i]; op=i;}}
   print op, m;
}
```