Processamento de Linguagens
Licenciatura em Engenharia de Sistemas Informáticos
2024/2025

Trabalho Prático

1 Introdução

Com este trabalho prático pretende-se que os estudantes de *Processamento de Linguagens* adquiram experiência na conceção e implementação de analisadores léxicos e sintáticos, bem como na definição de ações semânticas que traduzem a linguagem de entrada.

O problema proposto é a implementação de uma alternativa à linguagem de interrogação de base de dados relacionais, que seja executada sobre ficheiros de texto organizados num formato separado por vírgulas CSV (*Comma Separated Value*).

O processo para a realização deste trabalho prático deverá passar pelas seguintes fases:

1. especificar a gramática concreta da linguagem de entrada;
2. desenvolver um reconhecedor léxico (recorrendo à biblioteca *l*ex) para processar os símbolos terminais identificados na gramática concreta, e testar esse reconhecedor com um conjunto representativo de *palavras* da linguagem de entrada;
3. desenvolver um reconhecedor sintático (recorrendo à biblioteca *y*acc) para reconhecer a gramática concreta, e testar esse reconhecedor com alguns exemplos frases da linguagem de entrada. e testar esse reconhecedor com um conjunto representativo de *frases* da linguagem de entrada;
4. definir uma *árvore de sintaxe abstrata* para representar a linguagem de entrada, e associar ações semânticas, de tradução, às regras de produção da gramática de forma a determinar a respetiva árvore de sintaxe abstrata;
5. desenvolver o gerador de código que produza a resposta solicitada, através da avaliação da árvore de sintaxe abstrata.

A solução passa pelo desenvolvimento de uma aplicação para o processamento da linguagem `CQL` (*Comma Query Language*) descrita na secção 2.2,

A aplicação implementada deverá ser capaz de ler um ficheiro de texto com uma sequência de instruções, separadas por `;`, e deverá executar os comandos nele contidos.
Na secção seguinte é apresentado o enunciado do tema proposto para este trabalho prático. Na secção 3 são apresentadas as regras para o desenvolvimento deste trabalho prático.

2 Enunciado

Neste trabalho prático pretende-se que os estudantes implementem uma aplicação na linguagem de programação `Python`, usando a biblioteca `ply`, que interprete uma linguagem capaz de especificar algumas instruções que habitualmente encontramos em linguagens de programação.

A aplicação a desenvolver deverá começar por ler um ficheiro de texto (com extensão `.fca`, para funcional do Cávado e do Ave) contendo uma sequência de comandos de especificação da linguagem, aplique esses comandos de forma a calcular o resultado pretendido. O resultado de processar o ficheiro de texto `entrada.fca` é apresentado ao utilizador no terminal (opcionalmente, o programa poderá gerar um ficheiro com a respetiva implementação em código C). Caso não seja apresentado o ficheiro de entrada, os comandos devem ser lidos do terminal à medida que o utlizador os vai inserindo.

2.1 Reconhecedor de ficheiros CSV

O CSV é um formato textual com a forma de tabela, onde as linhas são separadas pela mudança de linha (*new line*) e as colunas por uma vírgula.

Consideramos que a primeira linha corresponde ao cabeçalho da tabela, e as linhas que iniciam pelo símbolo cardinal (`#`), são considerados comentários.

O valor de uma coluna pode estar delimitado por aspas. Neste caso, o seu conteúdo é considerado o valor da célula, sem as aspas. Esta notação é especialmente útil porque permite a inclusão de vírgulas no meio do texto de uma célula.

Considere-se que existem os seguintes ficheiros em determinada pasta:

- `estacoes.csv`

```
Id,Local,Coordenadas
E1,Terras de Bouro/Barral (CIM),"[-8.31808611,41.70225278]"
E2,Graciosa / Serra das Fontes (DROTRH),"[-28.0038,39.0672]"
E3,"Olhão, EPPO","[-7.821,37.033]"
E4,"Setúbal, Areias","[-8.89066111,38.54846667]"
```

- `observacoes.csv`

```
Id,IntensidadeVentoKM,Temperatura,Radiacao,DirecaoVento,IntensidadeVento,Humidade,DataHoraObservacao
E1,2.5,23.2,133.2,NE,0.7,58.0,2025-04-10T19:00
E2,15.1,12.5,679.6,E,4.2,99.0,2025-04-10T19:00
E3,4.0,16.4,0.0,NE,1.1,96.0,2025-04-10T19:00
E4,3.6,16.8,1.6,SW,1.0,88.0,2025-04-10T19:00:00
```

Pretende-se que se desenvolva uma forma de ler um ficheiro no formato CSV, validar se a sua estrutura corresponde às regras definidas acima, e, para cada linha válida, carrega os respetivos dados para uma estrutura de dados em memória. Esta funcionalidade será invocada na primitiva `IMPORT TABLE` (ver 2.2). De forma semelhante, deverá ser disponibilizada a funcionalidade que permita criar um ficheiro CSV a partir de uma tabela de dados em memória (primitiva `EXPORT TABLE`).

2.2 Descrição da Linguagem

A linguagem `CQL` (*Comma Query Language*) suportará os seguintes comandos:

A. configuração de tabelas de dados

- `IMPORT TABLE estacoes FROM "estacoes.csv";`
  insere numa estrutura de dados em memória o conteúdo do ficheiro `.csv`, ficando acessível através do identificador `estacoes`, correspondente ao nome do ficheiro;

- `EXPORT TABLE estacoes AS "est.csv";`
  guarda os dados da tabela, que estão atualmente em memória, no ficheiro `est.csv`

- `DISCARD TABLE estacoes;`
  elimina os dados da tabela `produtos` que estão em memória.

- `RENAME TABLE estacoes est;`
  altera o identificador da estrutura de dados em memória de `estacoes` para `est`;

- `PRINT TABLE est;`
  imprime os dados de uma tabela no terminal.

B. execução de *queries* sobre tabelas de dados

- `SELECT * FROM observacoes;`
  devolve todas as linhas e colunas da tabela indicada.

- `SELECT DataHoraObservacao,Id FROM observacoes;`
  devolve todas as linhas, mas apenas as colunas indicadas.

- `SELECT * FROM observacoes WHERE Temperatura > 22;`
  devolve todas as linhas que respondam a determinada condição. Sugere-se a possibilidade de permitir mais do que uma condição, separadas por `AND`.

- São aceites diferentes tipos de comparações: igualdade (`=`), desigualdade (`<>`), e outras comparações como: `<`,`>`, `<=` e `>=`.

- Aos exemplos anteriores, poderá ainda ser adicionado o modificador `LIMIT` n para retringir a quantidade de linhas do resultado apresentado (devolvido) ao valor n indicado.

C. criação de novas tabelas de dados

É possível criar novas tabelas em memória a partir de *queries*, ou da junção de outras tabelas:

- `CREATE TABLE mais_quentes SELECT * FROM observacoes WHERE Temperatura > 22 ;`
  Armazena numa nova tabela o resultado da *query*, permitindo que nas instruções seguintes se possa guardar o resultado num ficheiro.

- `CREATE TABLE completo FROM estacoes JOIN observacoes USING(Id);`
  Para juntar tabelas pode-se criar uma nova tabela. A junção de tabelas não irá permitir a seleção de colunas ou linhas, obrigando sempre à união completa de duas tabelas.

D. procedimentos

É possível criar procedimentos. Os procedimentos são declarados usando o comando `PROCEDURE`, e terminam com o comando `END`:

```
PROCEDURE atualizar_observacoes DO
  CREATE TABLE mais_quentes  SELECT * FROM observacoes WHERE Temperatura > 22 ;
  CREATE TABLE completo FROM estacoes JOIN observacoes USING(Id);
END
```

Os procedimentos podem ser executados usando o comando `CALL`. Por exemplo: `CALL atualizar_vendas;`

E. comentários

Os comentários apenas de uma linha são precedidos por `--` (todo o texto seguinte é ignorado). No caso dos comentários em mais do que uma linha, temos os seguintes operadores {`-` ... `-`}.

2.3 Observações

A avaliação deste trabalho terá em conta:

- a qualidade do reconhecedor léxico e da gramática implementados;
- a organização e qualidade do código desenvolvido;
- a estrutura da Árvore Abstrata de Sintaxe;
- a diversidade de operadores da linguagem apresentada no enunciado; A solução deve ser pensada como genérica, com capacidade de adaptação, e não apenas para os exemplos descritos no enunciado.

3 Regras

- O trabalho tem carácter obrigatório para aprovação à unidade curricular, e deve ser realizado em grupo de, no máximo, *três* elementos;
- O relatório deve introduzir o problema a ser resolvido, apresentar a abordagem seguida na sua resolução, quais os objectivos atingidos e quais os problemas encontrados. No relatório devem ser salientados todos os pontos que os alunos achem que poderão valorizar o seu trabalho em relação aos requisitos como, por exemplo, funcionalidades adicionais. Também deverá conter uma secção que descreva de que forma é que a aplicação foi testada. **O relatório poderá ter, no máximo, 15 páginas** (incluindo capa, índices, etc).
- O trabalho contempla uma apresentação e defesa individual em horário a agendar pelo docente. Esta defesa tem aprovação obrigatória, sendo que a falta à defesa corresponderá à não entrega do trabalho pelo estudante (i.e. avaliação de *zero* valores); Será agendado um horário com cada grupo para a apresentação do trabalho.

  Devem ser colocadas todas as referências de *todas as fontes* consultadas durante a elaboração do trabalho (bibliográficas ou mesmo consultas *online*);
- Durante as aulas dedicadas aos trabalhos poderá ser solicitado aos alunos que apresentem o trabalho desenvolvido até esse momento.
- A *data de entrega final* é aquela que foi estabelecida no início do semestre.
- Não serão aceites entregas ou melhorias após a data definida neste enunciado. Não serão aceites entregas ou melhorias nas épocas de exame (este trabalho apenas é válido para a avaliação da época em que é lançado).
- Quando o enunciado não for explícito em relação a um requisito específico, os alunos podem optar pela solução que parecer mais adequada, fazendo a sua apresentação e justificação no relatório. Dúvidas adicionais devem ser colocadas ao docente pessoalmente ou por *email*.
- O esclarecimento de dúvidas acerca deste documento pode originar a publicação de novas versões.