Processamento de Linguagens
Licenciatura em Engenharia de Sistemas Informáticos
2022/2023

Trabalho Prático 02

1 Introdução

Com este projeto pretende-se que os alunos da UC de Processamento de Linguagens adquiram experiência na definição de analisadores léxicos e sintáticos, bem como na definição de ações semânticas que traduzem as linguagens implementadas.

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

1. especificar a gramática concreta da linguagem de entrada;
2. construir um reconhecedor léxico (recorrendo à biblioteca *l*ex) para reconhecer os símbolos terminais identificados na gramática, e testar esse reconhecedor com alguns exemplos de palavras da linguagem de entrada.
3. construir 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.
4. planear uma árvore de sintaxe abstrata para representar a linguagem de entrada, e associar ações semânticas de tradução às produções da gramática de forma a construir a correspondente árvore de sintaxe abstrata.
5. desenvolver o gerador de código que produza a resposta solicitada, através da avaliação da árvore de sintaxe abstrata.

Nesse sentido, o que se propõe é a implementação de uma linguagem de uma calculadora, especificando expressões aritméticas.

Na secção seguinte são apresentadas as regras deste trabalho prático. Na secção **??** é apresentado o enunciado do problema proposto para este trabalho prático.

2 Regras

- O trabalho tem carácter obrigatório para aprovação à unidade curricular, deve ser realizado em grupo de até 3 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 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 aluno (i.e. avaliação de zero valores); Será agendado um horário com cada grupo para a apresentação do trabalho.

  Devem ser colocadas todas as referências 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).

- O esclarecimento de dúvidas acerca deste documento pode originar a publicação de novas versões.

3 Enunciado

Neste trabalho prático pretende-se que os alunos implementem uma ferramenta em `Python`, usando a bilioteca `PLY`, que interprete uma linguagem capaz de especificar algumas instruções que habitualmente encontramos em qualquer linguagem de programação.

A ferramenta a desenvolver deverá começar por ler um ficheiro de texto (com extensão *.ea*, para expressões aritméticas) contendo uma sequência de comandos de especificação das expressões aritméticas, aplique esses comandos de forma a calcular o resultado pretendido. O resultado de processar o ficheiro de texto `entrada.ea` 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.

3.1 Linguagem para Expressões Aritméticas

A linguagem para especificar expressões aritméticas é definida por uma sequência de instruções, seguidas pelo separador ponto e vírgula.

Uma instrução poderá ser: uma instrução de escrita, uma declaração, uma atribuição, ou uma instrução de saída.

Uma instrução de escrita, comando `ESCREVER`, envia para a consola os valores que estão definidos como argumentos do comando, os quais podem ser expressões aritméticas ou strings.

Uma declaração é iniciada palavra reservada `VARS` apresentando de seguida uma lista de variáveis separadas por uma vírgula, as quais podem ser inicializadas com uma constante inteira.

Uma atribuição passa por atribuir a uma variável: uma expressão aritmética, o resultado de leitura de um valor introduzido pelo utilizador, ou a geração aleatória de um valor.

Nos exemplos temos que os comandos são escritos em maiúsculas, no entanto podem ser utilizadas maiúsculas e minúsculas. Por outro lado, assume-se que para cada comando temos uma versão compacta com as três primeiras letras.

3.1.1 instrução de escrita

Segue um exemplo da utilização de instruções de escrita:

```
ESCREVER "olá mundo!";
ESCREVER "PL ", 2 ,"o ano de", "ESI";
ESCREVER "soma de ", 9, "com ", 3*4 , "=", 9+2*3 ;
```

A instrução `ESCREVER` recebe uma lista de argumentos separados por vírgulas. Os argumentos deste comando podem ser strings (ou seja, sequência de caracteres delimitada por aspas), uma constante numérica inteira, ou outra expressão aritmética. Neste comando podemos ainda ter identificadores de variáveis, ou mesmo invocações de funções definidas pelo utilizador. Esta instrução envia para a consola os valores que estão definidos como argumentos, fazendo a concatenação do texto das strings, com o resultado do cálculo das expressões aritméticas (convertendo-os para a sua representação em string).

3.1.2 declaração de variáveis

```
VAR ano = 2023, mes="maio", dia ;
```

Uma declaração é iniciada pela palavra reservada `VARS` apresentando de seguida uma lista de variáveis separadas por uma vírgula, as quais podem ser inicializadas com uma constante inteira, ou uma string. Qualquer sequência de letras (minúsculas ou maiúculas), o símbolo `'_'`, e números (que inicie por uma letra ou `'_'`) pode ser utilizada como identificador de uma variável.

3.1.3 atribuição a uma variável

Uma atribuição passa por atribuir a uma variável: uma expressão aritmética, o resultado de leitura de um valor introduzido pelo utilizador, ou a geração aletória de um valor.

```
tmp = 7+3 ;
a = 10 (30 + tmp ) ;
a ++; a += 10; b = tmp * (a + 10);
```

Uma expressão aritmética poderá ser atribuída a uma variável, havendo ainda a possibilidade de utilizar os operadores para incrementar a variável. Interessa verificar se as variáveis utilizadas foram previamente declaradas.

```
valor = ENTRADA();
ate10 = ALEATORIO(10);
```

O valor a atribuir a uma variável poderá ser lido da consola, ou então um número aleatório entre zero e o número inteiro indicado (no caso apresentado, 10).

3.1.4 ciclos

A linguagem de expressões aritméticas também suporta ciclos, como qualquer linguagem imperativa, com seguinte sintaxe:

```
PARA i EM [10..20] FAZER < instruções > FIM PARA ;
```

Este comando irá executar 11 vezes as instruções incluídas no ciclo, sendo que em cada iteração o valor da variável i irá sendo alterado, começando em 10, e incrementando até 20. Poderá ainda ser possível definir a valor a incrementar o contador.

Note que se podem usar ciclos dentro de ciclos.

3.1.5 funções

A linguagem de expressões aritméticas poderá também suportar a definição (e invocação) de funções, possibilitando a definição de um conjunto de comandos, os quais poderão ser invocados várias vezes utilizando parâmetros diferentes. Cada grupo deverá sugerir uma implementação e integração para esta funcionalidade da linguagem.

3.1.6 comentários e outras instruções

Deverá ser permitida a utilização de comentários: em uma única linha (`//`), ou mais do que uma linha (`/* ... */`).

Cada grupo poderá ainda sugerir uma implementação e integração de outras funcionalidades, tais como:

- uso do condicional (SE)
- operadores lógicos: `/\`,`\/`, `NEG`

3.1.7 resultado

Desenvolver o gerador de código que produza a resposta solicitada, através da avaliação da árvore de sintaxe abstrata para:

- executar as instruções no ficheiro de entrada, permitindo ver o correspondente resultado;
- criar um ficheiro em C, com um programa equivalente à sequência de instruções.

A avaliação deste trabalho terá em conta:

- a qualidade do reconhecedor léxico e da gramática implementados;
- a estrutura da Árvore Abstrata de Sintaxe;
- a diversidade de operadores da linguagem apresentada no enunciado;
- a organização e qualidade do código Python desenvolvido;
- a possibilidade de geração de código C.

3.2 Dúvidas

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*.