

UNIVERSIDADE DO MINHO

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA

GERADOR DE DIAPORAMAS

TRABALHO PRÁTICO N.º 3 - YACC

Processamento de Linguagens



Alexandre Pacheco (A80760)
Diogo Sobral (A82523)
Inês Alves (A81368)

10 de Junho de 2019

Resumo

No âmbito da Unidade Curricular de Processamento de Linguagens, foi proposto ao grupo, como forma de desenvolver os seus conhecimentos à cerca dos conteúdos lecionados nas aulas práticas, a realização de um conjunto de exercícios presentes no ficheiro fornecido. Neste ficheiro estavam presentes vários enunciados de diferentes trabalhos, sendo que o grupo ficou com o enunciado n.º 6: Gerador de Diaporamas.

Deste modo, o objetivo deste trabalho prático passa por aprofundar os conhecimentos relativos à Unidade Curricular em causa.

Conteúdo

1	Introdução	3
2	Descrição do Enunciado	4
3	Descrição da Solução do Problema	5
3.1	Descrição do gerador	5
3.2	Descrição de um slide	5
4	Implementação da Solução	6
4.1	GIC	6
4.2	Criação do HTML	7
4.3	Navbar	8
4.4	Atribuição de IDs às páginas	8
4.5	Créditos	9
5	Resultados Obtidos	10
5.1	Exemplo 1	10
5.1.1	Texto Fonte	10
5.1.2	Resultados	11
5.2	Exemplo 2	12
5.2.1	Texto Fonte	12
5.2.2	Resultados	13
6	Conclusões e Trabalho Futuro	16

1 Introdução

Estando o grupo a frequentar o 3.º ano do Mestrado Integrado em Engenharia Informática, foi-nos proposto, no contexto da Unidade Curricular de Processamento de Linguagens, que aprofundássemos os nossos conhecimentos na área. Posto isto, este trabalho consiste no desenvolvimento de um reconhecedor léxico e sintático para a linguagem de entrada com recurso ao par de ferramentas geradores Flex/Yacc. Para isso, são associadas ações semânticas de tradução às produções da gramática, recorrendo, uma vez mais, ao gerador Yacc.

Assim, este projeto torna-se uma mais-valia no que toca ao aumento de experiência de uso do ambiente Linux e diversas ferramentas auxiliares, bem como ao aumento da capacidade de escrever Gramáticas Independentes de Contexto (GIC), que satisfaçam a condição LR(), para criar Linguagens de Domínio Específico (DSL). Para além disso, também procura aprimorar o desenvolvimento de processadores de linguagens segundo o método da tradução dirigida pela sintaxe, suportado numa gramática tradutora (GT).

Ao longo deste relatório, iremos apresentar todas as etapas e decisões tomadas durante todo o processo de elaboração deste projeto.

2 Descrição do Enunciado

Neste exercício, foi proposto ao grupo que desenvolvesse um Gerador de Diaporamas¹. Para este fim, o processador deve aceitar uma descrição do diaporama que pretendemos construir e gerar uma sequência de páginas HTML temporizadas, isto é, que ao fim de k segundos, carreguem a página seguinte. É ainda importante ter em conta que cada página HTML corresponde a um diapositivo (elemento do diaporama). Posto isto, é fácil perceber que um diaporama é uma sequência de elementos, em que cada elemento pode ser/conter:

- uma página inicial de abertura e créditos
- imagens
- páginas simples
- vídeos
- título
- áudio

Para além destes, a fim de explorarmos ainda mais as nossas capacidades quanto ao tema em questão, decidimos, enquanto grupo, implementar alguns extras, isto é, cada elemento pode ainda ser/conter:

- *background*
- diversos tipos de texto (p.e., itálico, negrito...)
- tabelas
- *navbar*

Como forma de tornar o nosso diaporama mais rico, é também importante referir que cada elemento pode conter diversos *items* dos que foram mencionados anteriormente. Isto significa que por cada elemento é possível termos diversas imagens, diversos vídeos, tabelas, etc... Deste modo, não exigimos qualquer tipo de restrição quanto ao número de *items*, ou seja, não obrigamos a que exista um e só um *item* por elemento, enriquecendo todos os nossos elementos da melhor e mais organizada forma possível.

Como referido acima, as páginas têm a si associado um tempo de permanência, pelo que, ao carregar num *browser*, a primeira página da sequência é iniciada e o diaporama decorre ciclicamente.

Para isto, foi criada uma linguagem específica, especificada por uma GIC, que nos permitiu descrever os elementos que compõem o diaporama e a sua sequência. Após a definição desta, foram incluídas na nossa GIC diversas ações semânticas, com o objetivo de criar (através de Yacc) um processador que, depois de ler um texto-fonte com a descrição do diaporama, produzisse o conjunto de páginas HTML que realizasse a apresentação idealizada pelo grupo.

¹apresentações visuais suportadas por um conjunto de diapositivos ou acetatos/slides

3 Descrição da Solução do Problema

Como mencionado na secção anterior, o enunciado deste projeto proponha-nos que desenvolvêssemos uma linguagem simples capaz de realizar diversas funcionalidades. A nossa abordagem ao problema passou por tornar a solução o mais versátil possível. Deste modo, qualquer utilizador seria capaz de adaptar as funcionalidades desenvolvidas para obter o pretendido.

Neste sentido, a nossa solução passou por ser o mais genérica possível. Um diaporama é composto por elementos e cada um destes elementos está dividido em duas partes: *Head* e *Body*. Na *Head* assentam as ligações entre slides sucessivos e no *Body* todo o conteúdo do mesmo.

3.1 Descrição do gerador

Durante o planeamento da solução, achamos, por bem, que a nossa gramática deveria ser capaz de, através de um único ficheiro, construir vários diapositivos. Assim, o gerador é visto como um conjunto de diapositivos que origina uma pasta destinada a cada um. Cada diapositivo é, então, composto por uma página inicial que é obrigatória e um conjunto de elementos. Devido à flexibilidade da solução, a página destinada aos créditos não é obrigatória, no entanto, a mesma pode ser construída tal e qual como os outros elementos do diapositivo são.

3.2 Descrição de um slide

O *Body* contém uma lista com toda a informação que o slide representará, isto é, praticamente todas as propriedades descritas na secção anterior são aqui desenvolvidas. Uma vez que estamos a lidar com listas, não existe um limite definido para o conteúdo de cada slide. Esta decisão permite-nos que um utilizador coloque tudo aquilo que pretenda no diapositivo sendo que, neste modo, é possível ter um slide só com uma imagem, mas também é possível, por exemplo, ter um slide com uma imagem e um vídeo.

A outra componente de cada slide é o *Head*. Esta componente permite-nos não só estabelecer a ligação entre slides, mas também indicar quanto tempo devemos permanecer no mesmo. Quando nos encontramos no último slide, a sintaxe da ligação é diferente do habitual.

Por fim, cada diapositivo tem a possibilidade de ter um título. Para isto, a descrição da gramática para slides contempla dois casos: um em que apenas temos *Head* e *Body* e um segundo com *Head*, Título e *Body*.

4 Implementação da Solução

Para o problema em questão, foi necessário uma implementar um compilador capaz de interpretar a nossa *GIC* previamente definida e traduzi-la para código *HTML*.

4.1 GIC

Como tal começamos a nossa implementação por definir a nossa *GIC* que especificava a linguagem pretendida. No processo de desenvolvimento desta gramática, tivemos principalmente focados em formar um estilo apresentável, legível e simples, para que um utilizador comum, que não domina uma linguagem como *HTML*, possa facilmente converter os seus pensamentos em código *HTML*, acabando assim por surgir a seguinte *GIC*:

```

T = {START, END, NEXT, LAST, LINK, VIDEO, HEAD, BODY, IMAGEM, TD, TH, TAB,
      ENDTAB, TITULO, BACKGROUND, AUDIO, LIST, ENDLIST, PAGINI, TEXT, ENDTEXT,
      BOLD, IT, MARK, EM, STRONG, ENDBOLD, ENDIT, ENDMARK, ENDEM, ENDSTRONG,
      SMALL, ENDSMALL, ';;', ':', '|', '+', '{', '}' }

N = {generator, Designacao, Elementos, Elemento, Head, Body, Url, Nome,
      Conteudo, Tabela, Cabecalho, Linhas, Linha, Coluna, Titulo, Diaporama
      Multimidia, ListaHtml, Lista, PagIni, Texto, TextoHtml, Formatacao, Frase }

S = generator

P = {
p0: Diaporama -> START Designacao PagIni Elementos END

p1: Elementos -> Elemento
p2:           | Elementos NEXT Elemento

p3: Elemento -> HEAD Head BODY Body
p4:           | HEAD Head Titulo BODY Body

p5: Body -> Conteudo
p6:           | Body ';' Conteudo

p7: Conteudo -> Multimidia
p8:           | Tabela
p9:           | TextoHtml
p10:          | ListaHtml

p11: Multimidia -> IMAGEM Nome Url
p12:           | VIDEO Nome Url
p13:           | AUDIO Nome Url

p14: URL -> SENTENCE
p15: NOME -> WORD

p16: Head -> LAST
p17:           | LINK NUM

p18: PagIni -> PAGINI Head Titulo

```

```
p19: Designacao -> '{' SENTENCE NUM '}'  
p20:           | '{' SENTENCE NUM BACKGROUND SENTENCE '}'  
  
p21: Tabela -> TAB Cabecalho Linhas ENDTAB  
  
p22: Cabecalho -> Linha  
  
p23: Linhas -> Linha  
p24:           | Linhas ':' Linha  
  
p25: Linha -> Coluna  
p26:           | Linha '|' Coluna  
  
p27: Coluna -> TD SENTENCE  
p28:           | TH SENTENCE  
  
p29: Titulo -> TITULO SENTENCE NUM  
  
p30: ListaHTML -> LIST Lista ENDLIST  
  
p31: TextoHtml -> TEXT Texto ENDTEXT  
  
p32: Texto -> Frase  
p33:           | Texto '+' Frase  
  
p34: Frase -> SENTENCE  
p35:           | Formatacao  
p36:           | SENTENCE Formatacao  
p37:           | Formatacao SENTENCE  
  
p38: Formatacao -> BOLD Frase ENDBOLD  
p39:           | IT Frase ENDIT  
p40:           | MARK Frase ENDMARK  
p41:           | EM Frase ENDEM  
p42:           | STRONG Frase ENDSTRONG  
p43:           | SMALL Frase ENDsmall  
  
p44: Lista -> SENTENCE  
p45:           | Lista '+' SENTENCE  
  
p46: generator -> Diaporama  
p47:           | generator Diaporama  
}
```

4.2 Criação do HTML

Uma vez que, por cada elemento contido no nosso Gerador de Diaporamas, é necessária a criação de um ficheiro HTML, decidimos que, quando o nosso interpretador de texto se deparasse com um tipo *Head*, ele iria criar um novo ficheiro HTML, guardando o endereço deste ficheiro globalmente para ser utilizado pelas restantes funcionalidades do programa.

```
Head : LAST
| LINK NUM
```

```
{openFile(1,0); elemento++;}
{openFile(0,$2); elemento++;}
```

Visto que numa sequência de diaporamas também podemos pretender que, passado alguns segundos, o diapositivo seja alterado, tivemos em atenção este mesmo caso, existindo assim 2 representações do elemento *Head*. Desta forma, é-nos permitido obter a opção de, aquando da criação do ficheiro, referir se a página deverá ser redirecionada para a seguinte ao fim de alguns segundos ou se deverá permanecer inalterada até obter um *input* do utilizador. Esta funcionalidade extra só foi implementada devido à existência de uma *navbar* criada automaticamente que permite iterar pela sequência de elementos.

4.3 Navbar

Outra das funcionalidades extras implementadas foi a navbar.

Com base no número de elementos descrito na ‘Designação’ de um diaporama, é inicializado numa variável global o número de slides que vão existir. Para além disso a ‘Designação’ ainda contem o nome do diaporama e irá ser guardado esse nome para eventuais referências e para a criação da pasta onde se irão encontrar os ficheiros.

```
Designacao : '{' SENTENCE NUM '}'  

           {diagrama_nome = strdup($2);  

            mkdir(diagrama_nome,0700); n_slides = $3;  

            UrlBackground = "";  

            asprintf(&$$, "%s-%d", $2,$3);}  

| '{' SENTENCE NUM BACKGROUND SENTENCE '}'  

| {diagrama_nome = strdup($2);  

|   mkdir(diagrama_nome,0700); n_slides = $3;  

|   UrlBackground = strdup($5);  

|   asprintf(&$$, "%s-%d", $2,$3);}
```

Com base nos dados guardados é criada uma navbar por cada elemento percorrido através da função *char * navbar()* que, conforme os parâmetros acima descritos e a posição do elemento atual, retorna o código HTML necessário para criar uma barra de navegação com hiperligações para as restantes páginas.



Figura 1: Navbar

4.4 Atribuição de IDs às páginas

À medida que o interpretador de texto corre é necessário atribuir IDs ou nomes às diferentes páginas criadas. Visto que estas não recebem nenhum destes como um dos parâmetros da linguagem tem que ser o compilador a tratar desses casos. Sendo assim, o compilador para diferentes diaporamas terá diferentes nomes e, para cada diaporama, o ID será dado pela posição da página com o sufixo p.

Sendo assim a segunda página de um diaporama será p2.html, a terceira p3.html e assim em diante.

4.5 Créditos

Devido à versatilidade da solução implementada, também foi possível tornar a página de créditos muito mais genérica e com mais ferramentas ao dispor do utilizador. Sendo assim esta página é vista como qualquer outro elemento, possibilitando mais opções para além de texto puro. Segue-se um exemplo de uma página de créditos implementando a nossa GIC:

```
HEAD  
LAST  
  
TITULO 'Creditos' 45  
  
BODY  
LIST  
    'Alexandre Pacheco' +  
    'Diogo Sobral' +  
    'Inês Alves'  
ENDLIST
```

5 Resultados Obtidos

Nas subseções abaixo serão demonstradas algumas das capacidades na gramática criada.

5.1 Exemplo 1

5.1.1 Texto Fonte

```
START
{ 'teste' 2 }
PAGINI
    LINK 10
    TITULO 'Apresentação Teste' 144
HEAD
    LINK 10
    TITULO 'Slide 1' 40
BODY
    TAB
        TH 'Número de Ocorrências' |
        TH 'Designacao'
        TD '25' |
        TD 'Braga' :
        TD '54' |
        TD 'Lisboa'
    ENDTAB
NEXT
HEAD
    LAST
    TITULO 'Créditos' 45
BODY
    LIST
        'Diogo Sobral' +
        'Alexandre Pacheco' +
        'Inês Alves'
    ENDLIST
END
```

5.1.2 Resultados

0 1 2

Apresentação Teste

Figura 2: Página Inicial

0 1 2

Slide 1

Número de Ocorrências	Designacao
25	Braga
54	Lisboa

Figura 3: Slide intermédio

0 1 2

Créditos

- Diogo Sobral
- Alexandre Pacheco
- Inês Alves

Figura 4: Créditos

5.2 Exemplo 2

5.2.1 Texto Fonte

```

START
{ 'teste' 4 BACKGROUND 'background.jpg' }
PAGINI
    LINK 10
    TITULO 'Apresentação Teste' 144
HEAD
    LINK 8
TITULO 'Slide 1' 40
BODY
    VIDEO VideoT 'teste.mp4' ;
    AUDIO Sonoro 'sound.mp3'
NEXT
HEAD
    LINK 10
TITULO 'Slide 2' 40
BODY
    TAB
        TH 'Número de Ocorrências' |
        TH 'Designacao'
        TD '25' |
        TD 'Braga' :
        TD '54' |
        TD 'Lisboa'
    ENDTAB
NEXT
HEAD
    LINK 10
BODY
    IMAGEM Olivia 'dashboard.png' ;
    LIST
        'Olas' +
        'Estou' +
        'afinal'
    ENDLIST
NEXT
HEAD
    LAST
TITULO 'Créditos' 45
BODY
    LIST
        'Diogo Sobral' +
        'Alexandre Pacheco' +
        'Inês Alves'
    ENDLIST
END

```

5.2.2 Resultados



Figura 5: Página Inicial

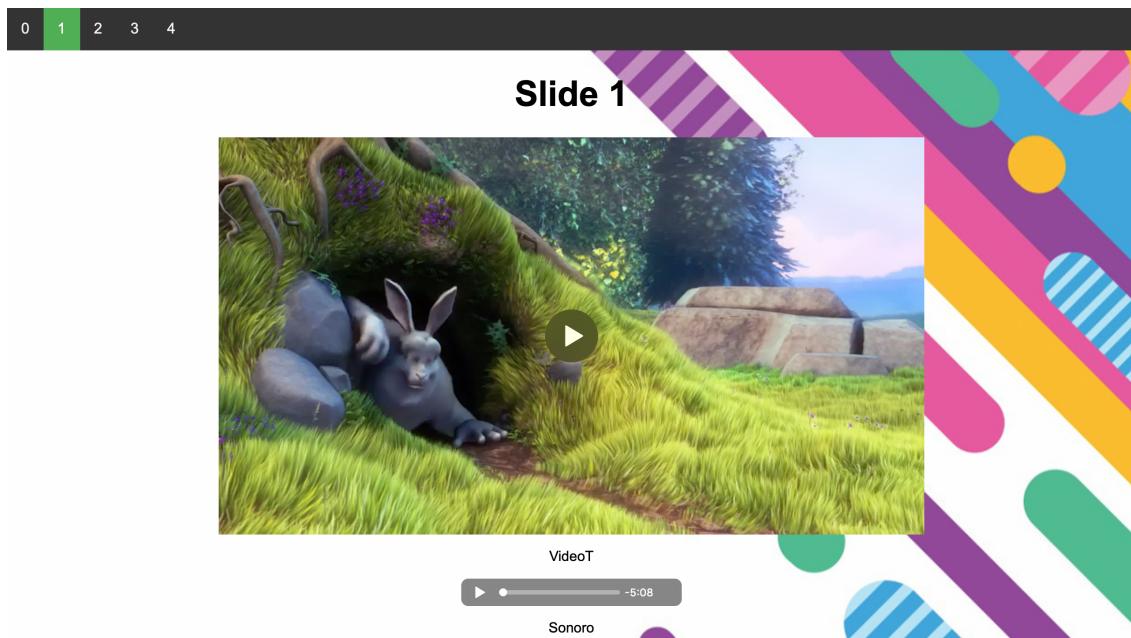


Figura 6: Slide intermédio



Figura 7: Slide intermédio

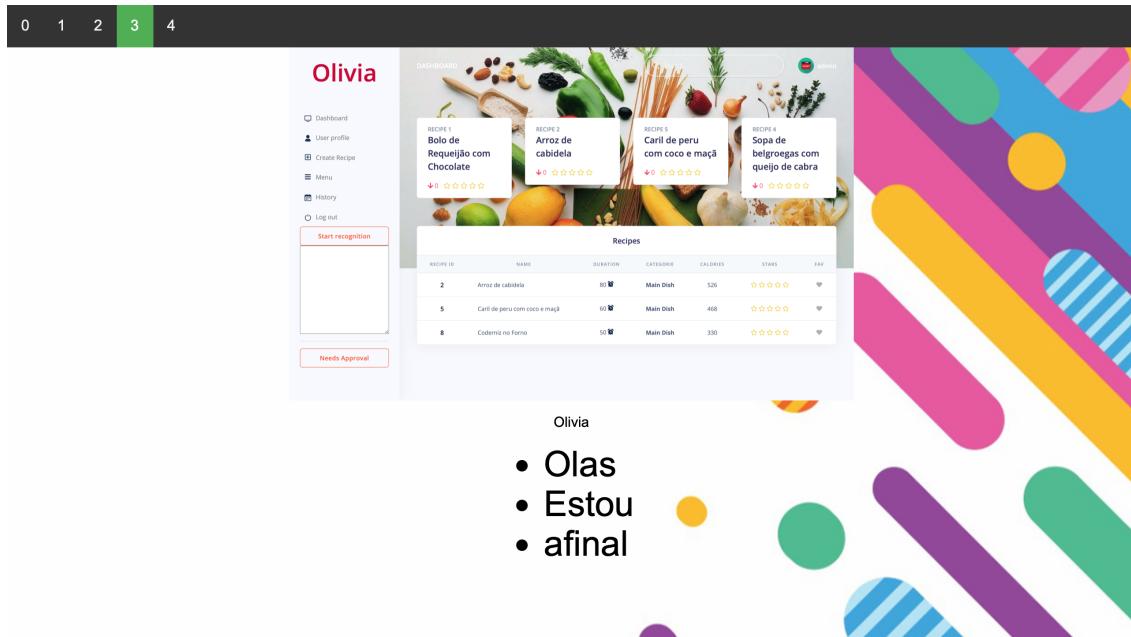


Figura 8: Slide intermédio

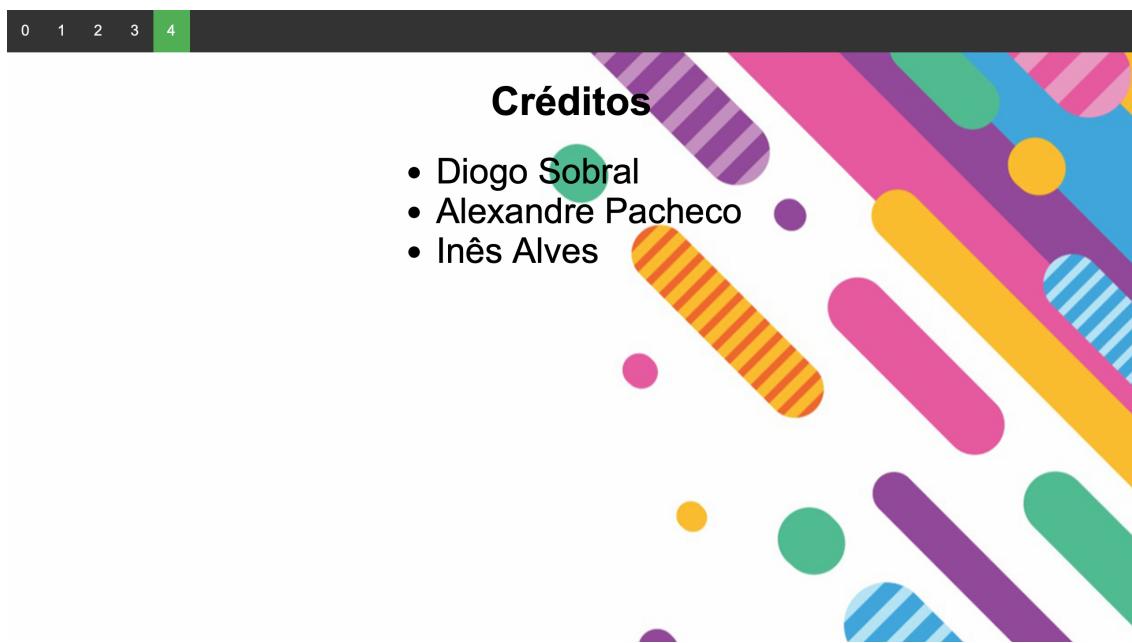


Figura 9: Créditos

6 Conclusões e Trabalho Futuro

A elaboração deste trabalho prático permitiu-nos consolidar e explorar todos os conhecimentos adquiridos nas aulas práticas ao longo do ano. Se, por um lado, foi necessário familiarizarmos-nos com o *Yacc*, por outro, este trabalho também exigiu que todos os conhecimentos adquiridos até ao momento fossem postos em prática.

O bom planeamento inicial permitiu que o processo de implementação fosse agilizado e que o produto final fosse algo de que todos os elementos do grupo se orgulhem. Todavia, tendo em conta a fase atribulada do semestre em que nos encontramos, existem aspetos que poderiam ser melhorados caso tivéssemos mais tempo para os mesmos.

Uma vez que qualquer tecnologia usada para apresentações tem sempre uma forte componente estética, para trabalho futuro, poderíamos implementar na gramática uma forma de especificar o estilo da página criada através do uso de *CSS*. Esta adição traria uma grande versatilidade ao uso da ferramenta e uma melhoria do resultado final.