Processamento de Linguagens e Compiladores
LCC (3ºano) + MiEFis (4ºano)

Trabalho Prático nº 3 (YACC)

Ano lectivo 19/20

Objectivos e Organização

Este trabalho pratico tem como principais **objectivos**:

- aumentar a experiência de uso do ambiente Linux, da linguagem imperativa C (para codificação das estruturas de dados e respectivos algoritmos de manipulação), e de algumas ferramentas de apoio à programação;
- rever e aumentar a capacidade de escrever *gramáticas independentes de contexto (GIC)*, que satisfaçam a condição LR(), para criar Linguagens de Domínio Específico (DSL);
- desenvolver processadores de linguagens segundo o método da *tradução dirigida pela sintaxe*, suportado numa *gramática tradutora (GT)*;
- utilizar *geradores de compiladores* como o par flex/yacc

Para o efeito, esta folha contém 6 enunciados, dos quais deverá resolver um escolhido em função do número do grupo (NGr) usando a fórmula $exe = (NGr \ \% \ 6) \ + \ 1$.
Neste 3º TP que, tal como os anteriores, pretende-se que seja resolvido com agilidade, os resultados pedidos são simples e curtos. Aprecia-se a imaginação/criatividade dos grupos ao incluir outros processamentos!

Deve entregar a sua solução **até Domingo dia 12 de Janeiro**.
O programa desenvolvido será apresentado aos membros da equipa docente, totalmente pronto e a funcionar (acompanhado do respectivo relatório de desenvolvimento) e será defendido por todos os elementos do grupo, em data a marcar após as semanas dos testes.
O **relatório** a elaborar e a enviar por email, deve ser claro e, além do respectivo enunciado, da descrição do problema, das decisões que lideraram o desenho da solução e sua implementação (incluir a especificação FLex) e Yacc, deverá conter exemplos de utilização (textos fontes diversos e respectivo resultado produzido). Como é de tradição, o relatório será escrito em LaTeX.

Enunciados

Para sistematizar o trabalho que se lhe pede em cada uma das propostas seguintes, considere que deve, em qualquer um dos casos, realizar a seguinte lista de tarefas:

1. Especificar a gramática concreta da linguagem de entrada.
2. Desenvolver um reconhecedor léxico e sintáctico para essa linguagem com recurso ao par de ferramentas geradoras Flex/Yacc.
3. Construir o gerador de código que produza a resposta solicitada. Esse gerador de código é construído associando acções semânticas de tradução às produções da gramática, recorrendo uma vez mais ao gerador Yacc.

Conteúdo

1 Rede Semântica do Museu do Artista

Neste projecto pretende-se descrever parte da *rede semântica* que suporta o **imaginário** Museu virtual do Artista (MvA), cujo ramo artístico (musical, plástico (pintura ou escultura), literário, teatral, etc.) será escolhido pelo grupo de acordo com suas preferencias.

A ideia é imaginar uma linguagem específica (concebida a seu gosto e especificada por uma GIC que terá de criar) que permita descrever três tipos de nodos (ou vértices) dessa *rede semântica* (ou grafo de conceitos) que define o conhecimento que é o espólio do dito MvA:

artista caraterizado pelos seus dados pessoais, de formação e de percurso profissional;

obra (musica, pintura, escultura, livro, peça/filme representado, peça/fiçme encenado, etc.) descrita pelos atributos apropriados para cada área;

evento (concerto, exposição, festival, concurso, sarau literário, sarau musical, etc.) descrito também pelos atributos apropriado para cada caso.

A sua linguagem deve ainda permitir ligar os nodos uns com os outros (definir os arcos de modo a criar o grafo) através de relações binárias tais como produziu (liga artista a obra), participou (liga artista a evento), ensinouCom/aprendeuCom/colaborouCom (liga artista a artista), ou outras que ache apropriadas.

Note que deve depois criar textos-fonte de acordo com os seus conhecimentos (já adquiridos, ou a consultar em enciclopédias) para testar o seu sistema e ver os resultados.

Um vez definida a linguagem, deve incluir na sua GIC ações semânticas para criar (através do Yacc) um processador que, após ler um texto-fonte com a descrição dos artistas a incluir no museu (suas obras e eventos), deve gerar um grafo (usando GraphViz / WebDot) que permita uma navegação conceptual sobre esse repositório de conhecimento; o utilizador do visualizador da rede deve poder seleccionar um nodo e saltar para uma página com info sobre esse elemento.

2 Informação Geográfica

O objectivo deste projecto é construir um *site* para as localidades de Portugal face à sua hierarquia geográfica definida à custa da linguagem OrgGeo abaixo descrita.

OrgGeo é uma Linguagem de Domínio Específico (DSL) que se destina a descrever a organização geo-política de um país, ou sua região. Para o efeito, pretende-se identificar as Localidades de cada Concelho de cada Distrito da região em causa.

A gramática independente de contexto G, abaixo apresentada, define o tipo de linguagem pretendida—cada grupo pode criar a sua variante.

O Símbolo Inicial é `OrgGeo`, os Símbolos Terminais são escritos em minúsculas (pseudo-terminais), ou em maiúscula (palavras-reservadas), ou entre apostrofes (sinais de pontuação) e a string nula é denotada por '&'; os restantes serão os Símbolos Não-Terminais.

```
OrgGeo   -->  Distritos
Distritos-->  IdD  DescD  '.'
           |  Distritos  IdD  DescD  '.'
DescD    -->  Concelhos
Concelhos-->  Concelho
           |  Concelhos  ';'  Concelho
Concelho -->  Locais  ':'  IdC
Locais   -->  Local  LstIds
LstIds   -->   &
           |  ','  Locais
IdD      -->  id
IdC      -->  id
Local    -->  IdL .....
IdL      -->  id
```

A informação a incluir em cada `Local` fica ao cuidado do grupo, sendo que se deve contemplar a possibilidade de fornecer um URL para um site do local; caso tal suceda, no site gerado existirá um link para esse site do local.

Uma possível forma de desenvolver o projecto é definida a seguir: Transforme G numa **gramática tradutora**, GT, reconhecível pelo yacc, para gerar um sítio HTML com uma página por cada Distrito, contendo uma lista de items com os Concelhos e para cada um uma outra lista com suas localidades. Complete a GT anterior de modo a gerar também uma página inicial (a *homepage* do seu sítio) com um título (fixo) do projecto e links para as páginas dos Distritos.
Adicionalmente, estenda o seu processador para incluir a construção duma **Tabela de Localidades** que associe a cada identificador de Localidade os respectivos Concelho e Distrito. Note que podem existir Localidades com nomes repetidos, desde que tal não aconteça no mesmo Concelho. Nomes de Concelhos e de Distritos é que não podem ser duplicados.

3 Formato Biblio::Thesaurus

O formato thesaurus ISO 2788 (T2788) é usado para representar ontologias / thesaurus. Dum modo simplificado contem metadados (indicação das línguas, relações entre conceitos, inversas das relações, titulo, etc) e um conjunto de conceitos.

Cada conceito inclui:

- um representante na linguagem de base.
- traduções noutras línguas.
- ligações (através das relações inicialmente identificadas) a outros conceitos.

Pretende-se:

- Escrever uma GIC para o formato T2788 e o respetivo parser/analisador-léxico para reconhecer um Thesaurus ISO 2788.
- Armazená-lo num estrutura de dados em memória (IR).
- Fazer uma travessia da estrutura de dados e gerar uma página HTML para cada conceito, sendo construídas hiperligações de acordo com as relações conceptuais.

O processador que, após o parsing, constrói a IR e faz a travessia geradora deve ser gerado automaticamente através do Yacc a partir da GT que se obtém da GIC inicial associando Ações Semânticas às produções gramaticais.

Abaixo mostra-se um exemplo de Thesaurus em formato ISO 2788, sendo que o carater '`#`' inicia um **comentário** até ao fim da linha:

```
                          # directivas / metadados
%language PT EN           # línguas: PT EN
```

```
%baselang EN                # língua de base: EN
%inv NT BT                  # NT, BT são relações inversas  a NT B => b BT a

                            # conceitos:
animal                      # termo na baselang
PT animal                   # LINGUA termo
NT cat, dog, cow,           # NT = narrow term = termo específico
  fish, ant
NT camel
BT Life being               # BT = broader term = termo genérico
                            # linha em branco : separador de conceitos
cat
PT gato
BT animal
SN animal que tem sete      # scope note = nota explicativa
  vidas e m(e)ia

#comentário                 # desde o símbolo '#' até ao fim da linha
```

4 Linguagem para definição de dados genealógicos

Ao falar de histórias de vida, histórias de família, etc., é muitas vezes necessário incluir dados referentes às relações de parentesco e outros dados de cariz genealógico.

Pretende-se criar uma notação genealógica (ngen) compacta para definir este tipo de relacionamentos familiares. Deve para isso construir um GIC que defina formalmente esta linguagem e que, após ser transformada numa GT por adição de Ações Semânticas às suas produções, sirva de suporte à geração automática do processador pretendido através do recurso ao Yacc.

Esta notação ngen deve cobrir elementos como:

Nomes . Exemplos:

- `João Manuel Rodrigues da Silva`
- `João Manuel/Rodrigues da Silva` (separação nome apelido)
- `João Manuel Rodrigues da Silva%2` (distingue entre 2 elementos com o mesmo nome)

Eventos e Datas . Exemplo:

- `*1996` (evento=Nasceu ; data 1996)
- `+1996` (evento=Faleceu; data 1996)
- `+c1996` (evento=Faleceu; data cerca de 1996)
- `cc(1996) P` (evento=casamento; data 1996; com a pessoa P)
- `ev(1996:x)` (evento=x; data 1996)

Parentescos . Exemplo, dada uma pessoa P1:

- `P P2` (relação P1 tem como pai P2)
- `PM P2` (relação P1 tem como Avó paterna P2)
- `-P P2` (relação P2 tem como pai P1)
- `-PP P2` (relação P2 tem como Avô paterno P1)
- `F P2` (relação de P2 com casamento atrás descrito)

Note que estas *relações têm inversa*.

Ficheiros anexos Fotografias e Documentos complementares. Exemplo, dada uma pessoa P1:

- `FOTO file.jpg`
- `HIST file.tex` (em file.tex há uma história em que P1 participa)

Abaixo mostra-se um exemplo de texto-fonte na linguagem sugerida:

```
Manuel da Silva *1977 +2011 [3]  // nome, nascimento, morte, II=3
M  Maria da Silva +2009          // mãe nome, morte da mãe
P  Joaquim Oliveira da Silva     // Pai
MM Joaquina *1930                // mãe da mãe
MP [45]                          // pai da mãe é o #I45, descrito anteriormente
FOTO f.jpg
HIST h1.tex
CC 2000 [2]                      // #I3 casou-se em 2000 com #I8, IF=2
Maria Felisbina *1980 [8]        // Conjugue, nome, nascimento, II=8
F  Serafim da Silva *2004        // Filho (ref. ao CC anterior [3][8])
F  Ana da Silva *2006 [7]{       //
     FOTO f1.jpg                 // dados extra referentes à Ana #I7
     HIST h1.tex
   }
```

O processador que terá de desenvolver, deve, após reconhecer um texto-fonte (semelhante ao que se mostra acima), gerar factos elementares do género que se exemplo abaixo (pode alterar os detalhes, por exemplo gerando cláusulas Prolog que possam ser testadas num interpretador dessa linguagem de programação):

```
#I3 nome Manuel da Silva
#I3 data-nacimento  1977
#I3 data-falecimento 2011
#I3 tem-como-mae #aut1
#aut1 nome Maria da Silva
#aut1 data-nascimento 2009
#I3 tem-como-mae #aut2
#aut2 nome Joaquim Oliveira da Silva
#I3 temo-como-MM #aut3
#aut3 nome Joaquina
#aut3 data-nascimento 1930
#I3 temo-como-MP #I45
#I3 FOTO f.jpg
#I3 HIST h1.tex
#F2 = #I3 #I8
#F2 data-casamento 2000
#I8 nome Maria Felisbina
#I8 data-nascimento 1980
#aut4 nome Serafim da Silva
#aut4 data-nascimento 2004
#F2 tem-como-filho #aut4
#I7 nome Ana da Silva
#I7 data-nascimento 2006
#F2 tem-como-filho #I7
#I7 FOTO f1.jpg
#I7 HIST h1.tex
```

Para além dos factos pedidos, podem gerar outras coisas que achem interessante (como por exemplo DOT para desenhar as árvores genealógicas).

5 Tradutor YAML – JSON

Construa um tradutor YAML-to-JSON (Y2J) entre estes 2 formatos de representação de informação tão em voga atualmente no seio das comunidades informáticas.

Para implementar este tradutor terá de definir, através de uma GIC, o formato de entrada (linguagem Yaml) e depois de ter um parser/analisador-léxico para reconhecer textos-fonte escritos nessa notação deverá juntar Ações Semânticas às produções da gramática para obter uma GT que, com recurso ao Yacc, permita gerar automaticamente o tradutor. Observe para isso os exemplos a seguir mas procure as definições dos formatos em causa.

Exemplo de um texto em Yaml:

```
---
# <- yaml supports comments, json does not
# did you know you can embed json in yaml?
# try uncommenting the next line
# { foo: 'bar' }

json:
  - rigid
  - better for data interchange
yaml:
  - slim and flexible
  - better for configuration
object:
        key: value
  array:
    - null_value:
    - boolean: true
    - integer: 1
paragraph: >
   Blank lines denote

   paragraph breaks
content: |-
   Or we
   can auto
   convert line breaks
   to save space
```

Exemplo de um texto contendo a mesma informação, mas agora em formato Json:

```
{
  "json": [
    "rigid",
    "better for data interchange"
  ],
  "yaml": [
    "slim and flexible",
    "better for configuration"
  ],
  "object": {
    "key": "value",
    "array": [
      {
        "null_value": null
      },
```

```
      {
        "boolean": true
      },
      {
        "integer": 1
      }
    ]
  },
  "paragraph": "Blank lines denote\nparagraph breaks\n",
  "content": "Or we\ncan auto\nconvert line breaks\nto save space"
}
```

Sugestão: veja a página `https://www.json2yaml.com/convert-yaml-to-json`[1].

6 Construtor de diaporama

Pretende-se, neste problema, construir um Gerador de Diaporamas (vulgo, apresentações visuais suportadas por um conjunto de diapositivos ou acetatos/slides). Para isso o seu processador deve aceitar uma descrição do *diaporama* que se pretende construir e gerar uma sequência de páginas HTML temporizadas (que ao fim de k segundos, carreguem a pagina seguinte), em que cada página HTML corresponderá a um diapositivo (elemento do diaporama).
O **diaporama** é uma sequência de elementos. Cada elemento pode ser/conter:

- Uma página inicial de abertura e créditos
- imagens
- páginas simples (exemplo: uma lista de items)
- video[2]
- opcionalmente, título
- opcionalmente, audio

As páginas têm associado um tempo de permanência. Ao carregar num browser a primeira página da sequência, o diaporama deverá decorrer ciclicamente.
Exemplo de uma página HTML temporizada 'p1.html':

```
<html>
<head>
  <meta charset="UTF-8"/>
  <meta http-equiv="REFRESH" content="34 ;URL=p2.html">
</head>
<body>
 <h1>Pagina p1</h1> ...ao fim de 34s carrega p2.html
 <hr/>
 <img width="100%" src="img1.jpg"/>
</body>
</html>
```

Para desenvolver este trabalho, deve imaginar uma linguagem específica (concebida a seu gosto e especificada por uma GIC que terá de criar) que permita descrever os elementos que compõem o diaporama e a sua sequência (temporizada). Um vez definida a linguagem, deve incluir na sua GIC ações semânticas para criar (através do Yacc) um processador que, após ler um texto-fonte com a descrição do diaporama produza o conjunto de páginas HTML que realizarão a apresentação pretendida.
Note que deve depois criar textos-fonte especificando diversos diaporamas para testar o seu sistema e ver os resultados.

[1] Se necessário, simplifique alguns detalhes que lhe pareçam demasiado complexos.
[2] Ver os elementos HTML5 para audio e video