Universidade do Minho

Report 2007

Vamos escrever relatórios

Pedro Faria
60998
a60998@alunos.uminho.pt
Pinto
61049
a61049@alunos.uminho.pt
Medeiros
61041
a61041@alunos.uminho.pt

Braga,5 de Maio de 2014

Conteúdo

Lista de Figuras

Lista de Tabelas

Resumo

De forma a colocar em prática os conhecimentos adquiridos até ao momento nas aulas teóricas e teórico-práticas da já referida Unidade Curricular, fizemos uso, com maior destaque, das ferramentas Flex - The Fast Lexical Analyzer já abordada no primeiro trabalho e também Yacc - Yet Another Compiler Compiler. O enunciado que escolhemos foi "Report 2007: vamos escrever relatórios", no qual era pedido que se criasse um compilador capaz de aceitar e processar relatórios escritos numa determinada linguagem. No final, teria de gerar, de forma bem estruturada, a sua respetiva versão em formato HTML e LaTeX.

Palavras-Chave: palabras

1 Introdução

O presente relatório descreve todo o processo de desenvolvimento, as tomadas de decisão, os principais obstáculos encontrados e, finalmente, os resultados obtidos na resolução do enunciado do segundo trabalho prático da Unidade Curricular de Processamento de Linguagens.

Vemos este projeto com grande entusiasmo porque consideramos que as ferramentas que precisamos de aplicar têm muita potencialidade e nos permitem fazer coisas muito interessantes. Para além disso, consideramos que é a melhor aproximação possível aos problemas da reais, em termos de complexidade, que podemos conseguir, mantendo, ainda assim, o nível de um exemplo académico.

2 Descrição das ferramentas utililizadas

Para o desenvolvimento deste projeto prático necessitamos das seguintes ferramentas: Flex (The Fast Lexical Analyzer), Yacc (Yet Another Compiler Compiler), a linguagem de programação C, HTML e LaTeX.

Os dois primeiros componentes tratam-se, respetivamente, de um analisador léxico e de um analisador sintáctico. São utilizados, a maior parte das vezes, em conjunto, isto é, o Yacc usa uma gramática formal de modo a analisar sintaticamente cada entrada enquanto que o Flex é responsável por fazer a distinção das expressões regulares. Estas expressões regulares são reconhecidos como tokens pelo Yacc.

O GCC, popular compilador da linguagem de programação C, é utilizado de modo a combinar o analisador léxico com o analisador sintáctico.

Por fim, foi também necessária a utilização de LaTeX e HTML, que foram gerados pelo compilador, com o objetivo de obter versões do relatório escrito em cada uma dessas linguagens.

3 Descrição da Linguagem desenvolvida

A nossa linguagem é maioritariamente baseada no esboço dado pelo enunciado mas conta, no entanto, com algumas variantes adicionais que decidimos acrescentar.

Tal como se pode ver na figura a seguir, a estrutura de um relatório pode ser dividida em três partes: uma parte inicial, um corpo e uma parte final correspondendo, respetivamente a: FrontMatter, Body e BackMatter. BREPORT e EREPORT são símbolos terminais constantes, isto é, palavras reservadas que servem, essencialmente, para representar o início e o fim da estrutura do relatório.

uma vez mais, temos os símbolos terminais constantes BFM e EFM a sinalizar o início e o fim desta primeira parte do relatório.

A lista de todos os símbolos não-terminais considerados apresenta-se de seguida. O FrontMatter é constituído

Por exemplo, cada vez que quisermos adicionar um autor ao nosso relatório, basta escrever BAUTHOR.

E de seguida adicionar os campos definidos na estrutura. Na nossa linguagem apenas é obrigatório sempre que se inicia o BAUHTOR, adicionar o nome(BNAME nome ENAME). Isto porque nas regras do Yacc assim defenimos:

A seguir, a parte principal do relatório que conterá o nosso Body:

O Body será constituído por uma lista de Chapters, e estes terão um Title e uma ElemList. Estes elementos podem ser várias coisas: uma imagem, uma tabela, uma section, etc.

Um parágrafo é uma lista de vários elementos, desde o texto até palavras em itálico ou negrito por exemplo. Para começar um parágrafo basta escrever "BPARA"e lá dentro inserir qualquer um dos elementos apresentados acima, iniciados respectivamente pela etiqueta. Outro tipo de elementos dentro do body podem ser as figuras e tabelas que serão iniciadas como BFIG, e BTABLE.

Resumindo, o nossa linguagem é o esboço feito pelo professor, com mais algumas modificações que a tornam mais completa. De certo modo achamos que o que estava no enunciado era suficiente para tornar um relatório completo e usável. A nosso ver, não havia nada a faltar para acrescentar melhorias na linguagem.

4 Especificação do FLEX

Quanto ao prodecimento a desenvolver para tratar da análise lexical, achámos necessário definir quatro estados como fizemos anteriormente no primeiro trabalho prático:

Ao introduzir estes quatro estados no Flex, conseguimos facilitar um pouco o nosso trabalho. No caso do codeblock, interessa-nos ter um estado porque isso nos permite guardar todo o que está dentro das etiquetas respectivas, inclusive outras etiquetas. Dentro do codeblock podemos fazer um BEGIN de outro tipo, sem compremeter o principal objectivo, que é colocar tudo o que está lá em formato de código. Para criar as tabelas, poupa-nos algum trabalho. Depois em relação aos estados list e keywords, que são muito parecidos, usamos estados porque assim basta inserir cada elemento da lista por linha sem precisar de usar algum tipo de separador, o que torna a nossa linguagem mai prática. Além de estados, o restante *Flex* é bastante simples e intuitivo pois são apenas as etiquetas que nós definimos para a nossa linguagem. Como por exemplo, este exerto inicial:

Como podemos ver por esse pedaço de código, é facilmente visível e perceptível o significado de cada umas das etiquetas que iremos usar na nossa linguagem.

5 Utilização do YACC

Aqui encontrar-se-ão referenciados todos os tokens deinidos na especificação FLEX e estes serão palavras reservadas da nossa própria linguagem. Cada token terá uma tarefa específicaca quanto ao processamento da linguagem em uso uma vez que cada palavra indica um tipo de instrucão diferente.

Aqui é efetuada uma tradução direta entre os símbolos definidos na gramática apresentada e o código C. É desta forma que guardamos na estrutura de dados todo o conteúdo relevante e que nos permite gerar relatórios tanto em LaTeX como em HTML.

Ao utilizar o YACC podemos definir código C a ser exucutado no momento em que cada regra da gramática definida é verificada.

6 Estruturas na linguagem C

No que diz respeito as estruturas de dados de forma a armazenar o conteúdo do relatório para posteriormente ser possível imprimir o conteúdo após fazer o processamento da linguagem definida, são descritas de seguida as decisóes e a forma de implementação destas no âmbito deste projecto. Assim sendo tornou-se necessária a criação de estruturas de dados para os dados relativamente a autores, nomeadamente o nome, o numero, o email, url e o affiliation.

Relativamente à parte inicial do report, frontmatter, que corresponde à capa do relatório e definida pelo titulo, subtitulo, data,instituição, tabela de conteúdo (toc), lista de figuras (lof) e lista de tabelas (lot) da seguinte forma:

Relativamente ao capitulo do relatorio este é implementado em uma lista ligada.

No que se refere ao paragrafo temos um id e o item, implementado da seguinte forma:

Como se faz notar recorre-se a tipos primitivos do C ou a estruturas básicas fácil implementaço. No primeiro trabalho pratico, usamos módulos de listas ligadas feitos por nos e agora optamos por utilizar o mesmo método.

Fica de notar que facilitou o processo, após já ter-mos trabalhado anteriormente com as mesmas, e será concerteza usada por nos futuro.

7 Geração de Relatórios - Latex e Html

Depois de escrita toda a gramatica e definidas todas as nossas tags a usar. Fica a faltar gerar, com o recurso a linguagem C, um relatorio em HTML. Todas a nossas funções que imprimem o HTML foram feitas no ficheiro ???????. Nos optamos por dividir e minimizar ao máximo a complexidade das funções para posteriormente ser mais fácil resolver algum problema. Deste modo, evitamos a implementação de funções demasiado grandes.

Fazendo a ligação das tags ao HTML, através da nossa linguagem decidimos o que gerar conforme o que decidimos.

8 Makefile

Para facilitar o processo de gestão quer de desenvolvimento quer de distribuição do software desenvolvido, seguimos os conselhos dos docentes e criamos um makefile com as funções mais comuns, tentado seguir os standards quer de nomenclatura quer das funcionalidades.

9 Conclusão

Após terminarmos este trabalho pratico podemos retirar algumas conclusões, vendo o resultado final, visivelmente e esteticamente agradável.

HTML.

De forma a concluir podemos dizer que este enunciado pós em causa todos os conhecimentos/conceitos abordados nas aulas o que de certa ajudou a combater algumas dificuldades que foram aparecendo na respectiva resolução e que levou a que enriquecêssemos mais a esse mesmo nível.