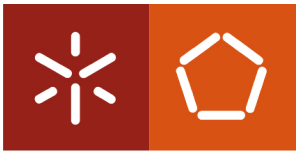

Universidade do Minho

LEI — Licenciatura de Engenharia Informática

Processamento de Linguagens

Compilador de uma LPS

Orlando Costa - a67705, Paulo Araujo - a58925, Rui Oliveira - a67661



Braga, 1 de Abril de 2015

Resumo

Este relatório descreve a resolução de um conjunto de exercícios propostos, que consistem no desenvolvimento de programas na linguagem C com o auxílio de geradores de filtros de texto, como o Flex. Para cada problema é realizada uma breve análise sobre o trabalho efetuado, as decisões que lideraram o seu desenvolvimento e as estruturas implementadas, assim como uma explicação do seu funcionamento.

Os problemas resolvidos consistem no desenvolvimento de filtros de texto que:

- processa um ficheiro XML com descrições de fotografias e gera um álbum HTML.
- processa de um ficheiro XML anotado com tags Enamex e gera páginas HTML apresentando as "pessoas", "países", "cidades"e organizações nele identificadas.
- processa vários ficheiros de texto, compostos por letras de canções, e gera documentos em LATEX para cada uma delas.

Para cada problema é apresentado o código em linguagem C e as expressões regulares desenvolvidas, sendo estes suportados por exemplos e devidos resultados.

Conteúdo

Capítulo 1

Introdução

O presente trabalho enquadra-se na unidade curricular de Processamento de Linguangens da Licenciatura em Engenharia Informática da Universidade do Minho. O trabalho pretende aumentar a experiência em engenharia de linguagens,

1.1 Linguagem de programação imperativa simples

Previamente ao desenvolvimento do compilador existe a necessidade de definir uma linguagem sobre a qual este atua, com base numa qualquer linguagem imperativa. Neste sentido e por simplicidade e familiaridade, a linguagem de programação C é a selecionada. Esta linguagem foi simplificada por forma a adaptar-se aos requisitos propostos, sofrendo as seguinte modificações na sua estrutura:

- Apenas permite manusear variáveis do tipo inteiro (escalar ou array). - Suporta apenas as instruções vulgares de controlo de fluxo de execução (condicional e cíclica), tais como if-else, for, while e do-while. - As instruções que controlam inserção e output de valores (tipicamente printf e scanf) estão adaptadas para suportar apenas inteiros, e então estão renomeadas (printi e scani). - As expressões lógicas devem estar rodeadas por parentises para facilitar a sua distinção e ordem quando em conjunto com expressões aritméticas.

<Programa exemplo (programa exemplo que mostre de forma simples todas as funcionalidades - funçoes, atribuiçoes, ciclos, expressoes logicas e aritmeticas)>

1.2 Arquitetura

O sistema desenvolvido é constituído por 2 principais modelos: parser.l, compiler.y, que são respetivamente o analisador léxico e analisador sintático.

Na Figura 1.1, podemos ver as dependências dos ficheiros entre si.

O analisador sintático, utiliza o ficheiro vmCompiler.h, este modulo é o responsável pelo tratamento das variáveis e funções existentes.

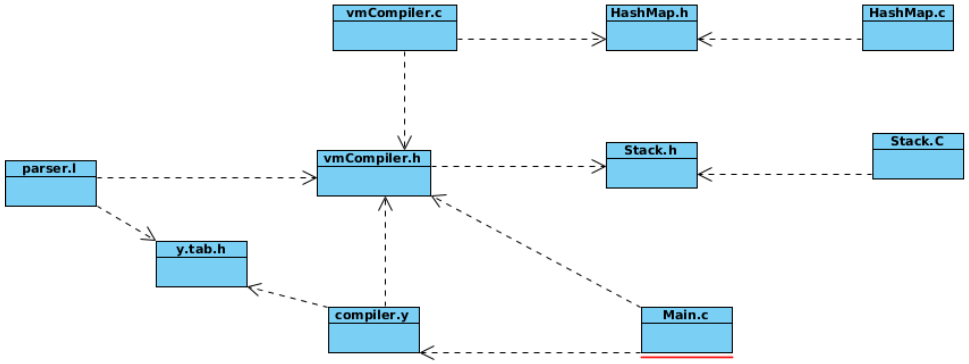

Figura 1.1: Diagrama das pedendencias dos ficheiros

Figura 1.2: Diagrama das estruturas usadas em vmCompiler

Em vmCompiler.h podem-se fazer ações tais como adicionar uma variável, e consultar as variáveis existentes.

O Sistema também usou duas estruturas de dados: um HashMap e ou Stack, hashmap utilizado para guardar as variáveis e as funções. A stack para na compilação controlar as label's dos ciclos.

Na figura 1.2, pode-se ver as estruturas utilizadas em vmCompiler:

- Scope - têm uma map com a informação das variáveis, onde a chave é nome da variável e o valor é um EntryVar;
- EntryFun - guarda a informação sobre o tipo de uma função (argumentos de entrada e tipo de retorno);
- EntryVar - guarda o tipo, nome o endereço relativo de uma dada variável;

Com as estruturas anteriores em mente, as variáveis criadas em vmCompiler são as representadas na figura 1.3, onde pode-se ver duas variáveis do tipo Scope uma para o contexto global outra para o contexto interior a uma função.

Existe uma map de EntryFun (mFuncMap) onde a chave é o nome da função.

A variável DecFunAux, é um apontador temporário para um função que se esteja a declarar.

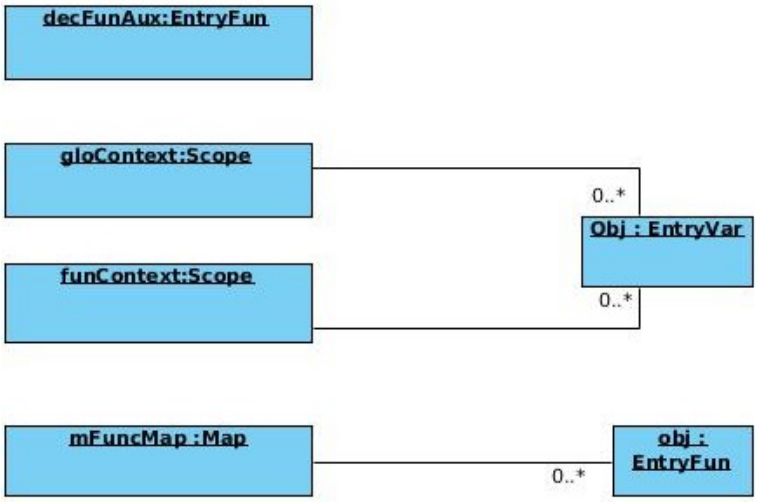

Figura 1.3: Diagrama dos objetos existente em vmCompiler

1.3 Estruturas de dados

1.3.1 Stack

De forma a evitar confusão na atribuição de *labels* relativas a *ifs* e *loops* é utilizado um contador de condições. À medida que é encontrada uma instrução que implique o uso de uma condição, este contador é incrementado e o seu valor é colocado numa stack. Deste modo, o valor que se encontra no topo da stack é relativo ao último *ciclo/if* encontrado. Sempre que é encontrado o final de uma condição, o valor no topo da stack é removido. Através do uso de um contador e de uma stack, é muito mais simples de gerir as *labels* e as operações de controlo, como *JUMPs* e *JZs*.

1.3.2 HashMap

Capítulo 2

Compilador

2.1 Analisador léxico

O analisador léxico construido deteta todos os símbolos terminais da linguagem (palavras reservadas, sinais e variáveis). É de se destacar a deteção de comentários, que são ignorados. De forma a que fosse mais fácil detetar os erros de sintaxe, o parser conta as linhas que já interpretou. Desta forma o analisador sintático quando dá erro diz a linha onde o erro aconteceu. Para passar valores como o nome de variaveis ou números utiliza-se o yylval.

2.2 Analalisador sintatico/semantico

2.3 Geração de codigo máquina

A cada regra da gramática, associamos acções a serem executadas à medida que estas são reconhecidas. Assim sendo, é feita uma tradução da linguagem desenvolvida para a linguagem *assembly* da VM, à medida que cada instrução ou expressão é identificada. A maioria destas acções implica uma instrução de escrita no ficheiro de output.

Capítulo 3

Testes

Capítulo 4

Conclusão

Terminado o desenvolvimento do trabalho, é importante referir que o mesmo nos permitiu aprofundar o conhecimento acerca do Gerador Léxico Flex assim como da análise léxica no geral, obrigando-nos também a utilizar ferramentas tais como HTML e Latex.
Relativamente ao problema do "Museu da Pessoa", a dificuldade recaiu na definição da estrutura de suporte de dados, dado que devido à falta de claridade do enunciado foi necessário re-implementar a estrutura de forma a que esta admitisse a funcionalidade de ter um índice geral em HTML.
No problema de "Processamento de Entidades Nomeadas"foi necessário chegar a um consenso acerca das tags que deveriam ser validadas e o relacionamento possível que estas teriam entre si. Após essa decisão, o desenvolvimento da estrutura que suporta esta informação tornou-se relativamente simples.
O problema "Processamento de Ficheiros com Canções"foi resolvida através da implementação de uma estrutura capaz de evitar que a ordem dos dados no ficheiro não seja significativa (e então armazena em memória a informação). A presença de certos elementos em datasets mais diversos (tais como header's não esperados e anotações em músicas) deram origem a problemas, sendo que a solução consistiu em ignorar essa informação.
Cada elemento do grupo realizou um exercício do enunciado proposto, apoiando-se mutuamente na existência de dificuldades. Apesar das dificuldades iniciais, encontramos-nos satisfeitos com o resultado final e estamos confiantes para o próximo trabalho.