

Trabalho de desenvolvimento do compilador da linguagem C-

Victor Grudtner, Matheus Nogueira, Wilson Camilo

¹Departamento de Ciência da Computação – Universidade Federal de Lavras (UFLA)
Lavras - MG, Brasil, 37200-000

Resumo. *O desenvolvimento das linguagens de programação de alto nível permitiu um avanço na computação, facilitando a implementação de programas e algoritmos de maneira mais rápida e simples. O compilador possui um papel fundamental nesse desenvolvimento das linguagens de programação, porque ele traduz linguagens de alto nível em linguagem de máquina. Este trabalho descreve as etapas de desenvolvimento de um compilador para linguagem c-, o qual foi dividido em 3 etapas: análise léxica, sintática e semântica. Ao fim desta primeira etapa obteve-se um analisador léxico que transforma um código fonte que recebe como entrada em tokens.*

1. Introdução

Um compilador é dividido em diversas fases, sendo elas: análise léxica, análise sintática, análise semântica, geração de código intermediário, otimização de código intermediário, geração de código objeto e otimização de código objeto. Onde a análise léxica gera um conjunto de tokens; a análise sintática os agrupa e verifica por erros sintáticos; a análise semântica verifica os erros semânticos e captura as informações de tipo; a geração do código intermediário utiliza o resultado das etapas anteriores para gerar normalmente, um código de três endereços; a otimização de código intermediário tenta melhorar o código intermediário; a geração de código objeto gera um código em linguagem de máquina; e por fim, a otimização do código objeto, que tenta melhorar o código objeto [Alfred V. Aho].

O restante do artigo está organizado na seguinte forma: Seção 2 contém a descrição do trabalho e os detalhes e instruções que devem ser seguidas ao longo do seu desenvolvimento. A seção 3, 4 e 5, são referentes a análise léxica, sintática e semântica respectivamente, onde é descrito os detalhes de cada uma dessas etapas no desenvolvimento deste trabalho. A seção 6, se refere aos materiais e métodos utilizados neste trabalho e a seção 7 é a conclusão. As etapas de análise léxica e semântica só serão acrescentadas neste artigo após o desenvolvimento de cada uma, assim, este artigo não é a versão final, podendo sofrer alterações ao longo da implementação do compilador.

2. Descrição do trabalho

Para o desenvolvimento da linguagem c- será necessário seguir as seguintes definições para criar o analisador léxico:

- Tipos de dados: inteiro, real, caractere, arranjo e registro.
- Funções: recursão, parâmetros passados por valor.
- Comandos: atribuição, if/else, while, E/S simples (tratados como funções).
- Comentários: texto entre /* e */ (sem comentários aninhados).
- Palavras reservadas: int, float, struct, if, else, while, void, return (caixa baixa).

- Símbolo inicial: « programa »

O objetivo desta primeira etapa é implementar um analisador léxico para a linguagem especificada. O analisador léxico deverá ser implementado usando a ferramenta Lex. Ele deverá retornar, a cada chamada, o token reconhecido. além de reconhecer os tokens da linguagem, o analisador léxico deverá detectar possíveis erros e reportá-los ao usuário. O programa deverá informar o erro e seu local (linha e coluna).

Os espaços em branco (espaços, tabulações, quebras de linha, etc.) e comentários não são tokens. Portanto, devem ser descartados. Será feito também um programa para testar o analisador léxico. Este programa imprime a linha, a coluna, o lexema e o tipo do token que foi identificado.

3. Análise léxica

Análise léxica é a primeira etapa, dentre as etapas de um compilador, porém, alguns compiladores tem uma etapa inicial antes da análise léxica. Essa etapa é conhecida com pre-processamento, nela é feita a expansão de macros de programação em código que será usado na compilação [REF]. Neste trabalho não será implementado um pré-processador de macros.

Nessa primeira etapa o arquivo de código fonte é utilizado como entrada do analisador léxico. O parser léxico retira os caracteres inválidos e os comentários do código e transforma os lexemas válidos em tokens que são adicionados na tabela de símbolos. A tabela de símbolos é uma estrutura de dados que armazena os tokens, é através desta tabela que as outras fases do compilador recebem a entrada de dados.

A descrição do funcionamento do analisador léxico pode ser visualizada através do AFD (Autômato Finito Determinístico) apresentado na Figura 1. O AFD foi criado a partir da definição da linguagem descrita no trabalho que foi implementado usando o Lex.

4. Análise sintática

Esta seção será abordada na próxima etapa deste trabalho.

5. Análise semântica

Esta seção será abordada na próxima etapa deste trabalho.

6. Materiais e métodos

Para o desenvolvimento do analisador léxico foi utilizado lex (Lexical Analyser Generator). O lex permite que a linguagem de programação seja definida por meio de expressões regulares [Lesk and <http://dinosaur.compilertools.net/lex/>]. A plataforma de desenvolvimento foi o SO (Sistema Operacional) Linux, com kernel na versão 3.18 para arquitetura x86_x64. Usando a linguagem de programação C no desenvolvimento do analisador léxico.

Para a realização dos testes foi utilizado arquivos que contém o código fonte usando a definição da linguagem deste trabalho. O analisador léxico recebe o código fonte como entrada, limpa os comentários e cria tokens do código sem comentários, além do comentário foi definido alguns erros léxicos os quais são:

- Numero definido que não está entre os tipos da linguagem como hexadecimal e binário.
- Variáveis não podem conter letra maiúscula.
- Variáveis devem começar com uma letra.
- Caracter inválido (qualquer caracter que não está definido na linguagem).
- Número real inválido.
- Token inválido (qualquer outra entrada que não está definida na linguagem).

Note que para a simplificação das transições entre estados ao invés de inserir todo um alfabeto, foi criado intervalos de caracteres/dígitos, no qual se exclui do intervalo por meio do símbolo - (menos) o(s) caracteres que não fazem parte do intervalo, exemplo:

alfabeto -c

No exemplo acima o intervalo corresponde a todas as letras do alfabeto excluindo o "c".

7. Conclusão

O objetivo de criar um analisador léxico foi atingido, no qual o processamento de um código fonte e transformar em tokens seguindo a definição da linguagem de programação também foi atingido.

Referências

The c preprocessor. <https://gcc.gnu.org/onlinedocs/cpp/index.html>. Acessado: 28/05/2016.

Alfred V. Aho, Monica S. Lam, R. S. J. D. U. *Compilers: Principles, Techniques, and Tools*. 2 edition.

Lesk, M. E. and <http://dinosaur.compilertools.net/lex/>, E. S. Lex - a lexical analyzer generator. <http://dinosaur.compilertools.net/lex/> Acessado: 28/05/2016.

