UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Universidade Federal de Santa Catarina - Campus Araranguá
Disciplina: DEC7565 - Construção de Compiladores
Professor: Alison Roberto Panisson
Alunos: Leandro Seiti Kakimoto Silva (17103448), Tobias Rossi Müller (17102342)

Relatório do Trabalho Final - Implementação Parcial de um Compilador

Araranguá, 17 de março de 2022

1. Análise Léxica

Nesta etapa definimos os tokens que são reconhecidos pelo programa fonte. Os tokens são detectados a partir de expressões regulares. Definimos as palavras reservadas como ‘if’, ‘int’, ‘float’, e ‘main’, além dos demais tokens, como ‘PLUS’ definindo o símbolo da operação aritmética de adição, e ‘RPAREN’ definindo o símbolo de fechamento de parênteses. Também definimos que o analisador ignore caracteres como espaço e quebra de linha. E por fim, registramos os símbolos para as operações aritméticas, parênteses, chaves, colchetes e a estrutura de variáveis do tipo inteiro, ponto flutuante e string.

2. Análise Sintática

A biblioteca PLY utiliza o analisador LR que realiza a redução do corpo de uma regra de produção a sua cabeça na ordem pós-ordem. Visto que ações S-atribuídas podem ser implementadas durante o processo de análise ascendente, e que o analisador sintático ascendente corresponde a um caminhamento pós-ordem, esta definição então pode ser utilizada pela PLY.

Aqui definimos a procedência das cadeias de entrada, primeiro definindo o início padrão de um código em C, validando uma expressão do tipo “int main(void){“, que ao final da leitura de todas as declarações, espera “}” como caracter final da pilha. Entre estas duas definições, realizamos a declaração do bloco de código principal, este deve aceitar a declaração de variáveis, atribuição de valores a estas, além da comparação entre valores de variáveis e realização de operações aritméticas com as mesmas.

A gramática utilizada tem as seguintes regras mais importantes para o funcionamento. Um dos pontos chave é o símbolo não terminal chamado de “declaracoes”. Sabemos que dentro de escopos praticamente qualquer declaração pode ser realizada, por exemplo o início de um laço de repetição, uma estrutura condicional, declarações ou atribuições de variáveis entre outras operações. Sendo assim, as produções do símbolo não terminal “declaracoes” podem ser tanto uma “declaracao_variavel”, quanto uma “atribuicao”, quanto um “condicional” ou um “estrutura_de_repeticao”. Cada um desses novos símbolos não terminais são posteriormente definidos adequadamente. Além disso, outro símbolo não terminal chave para a compreensão da gramática utilizada foi a “expressao”. A ideia geral é de que uma expressão pudesse gerar qualquer expressão matemática possível de se escrever dentro do código. Ela pode gerar a si mesma cercada de parênteses, ou com um sinal negativo à frente, duas expressões com um operador binário ao centro e duas expressões com um comparador ao centro. O símbolo não terminal produzido por “expressao” que terminará em um símbolo terminal é a abstração “valor”, que pode receber qualquer valor explícito como número inteiro, flutuante ou string. Outro produção terminal é o símbolo “id”.

Para facilitar a análise semântica, símbolos não terminais que abstraem a abertura de chaves e o fechamento foram criados. Isso permitiu que de dentro dessas abstrações o valor da variável global de contexto fosse alterada, adicionando-se ou retirando-se 1 do valor atual da variável.

3. Análise Semântica

Utilizamos a tabela de símbolos principalmente para realizar verificações semânticas em variáveis, isto é, para verificar se uma variável já existe no caso de tentativa de uma segunda declaração, ou no caso de atribuição sem declaração anterior. Ao encontrar um erro, a ação tomada neste projeto foi a impressão de uma mensagem de erro. A abordagem utilizada para o gerenciamento do contexto foi a subida de um nível de contexto a cada *bracket* do código, gerando assim, diferentes níveis de contexto a serem tratados dependendo de onde uma variável, atribuição ou comparação é feita no código.

Além disso, sempre que uma declaração de variável foi realizada, uma verificação foi executada para verificar sua existência prévia na tabela de símbolos. Caso a variável existisse, uma mensagem de erro indicando que uma variável já havia sido declarada com tal nome é mostrada. Outra verificação realizada para a análise semântica é quanto a atribuição de valor a uma variável. Para que a atribuição seja realizada de forma adequada, deve ser checado pelo menos que a variável sendo atribuída exista, ou seja, tenha sido anteriormente declarada. Caso não tenha sido, uma mensagem de erro é mostrada explicando que a variável ainda não foi declarada.