Compiladores
Roteiro de Laboratório 03 – Tabelas de Símbolos e de *Strings*

1 Introdução

A tarefa deste laboratório é incorporar uma tabela de símbolos (variáveis) e de literais (*strings*) no *front-end* do compilador para EZLang.

Uma tabela de *strings* é simplesmente um repositório das *strings* constantes que aparecem no código fonte de entrada. O papel desse repositório é armazenar as *strings* durante o processo de compilação para evitar que muitos módulos do compilador precisem manipulá-las. Isso é desejável porque operar sobre *strings* é algo custoso e propenso a erros. Idealmente, somente duas partes do compilador devem "encostar" nas *strings*: o *scanner*, que lê as *strings* e as armazena na tabela, e o *back-end*, que exibe as *strings* (ou faz um *dump*) no momento adequado da execução do programa de saída.

Uma tabela de símbolos é uma base de dados das variáveis (identificadores) que surgem no código fonte. Essa tabela é utilizada por vários módulos do compilador, que consultam e adicionam informações na tabela ao longo do processo de compilação. Exemplos de informações que são armazenadas na tabela de símbolos: lexema (nome da variável), linha (e arquivo) de declaração, tipo da variável, etc. Em geral, sempre que surgir a pergunta sobre aonde armazenar alguma nova informação obtida pelo compilador provavelmente a resposta será: na tabela de símbolos!

Mais detalhes sobre as tabelas de símbolos e de *strings* podem ser vistos nos *slides* da Aula 03.

2 Semântica de EZLang para declaração e uso de variáveis

As regras semânticas de EZLang para declaração e uso de variáveis são simples: todas as variáveis do programa de entrada devem ser declaradas antes de serem utilizadas no corpo do programa. Além disso, não é correto redeclarar variáveis, isto é, um identificador (lexema) não pode aparecer mais de uma vez na seção de declaração de variáveis. Sabendo dessas regras, podemos começar a construção do nosso analisador semântico.

2.1 Utilizando as Tabelas de Símbolos e de *Strings*

Realize as atividades abaixo para concluir a tarefa deste laboratório.

ATIVIDADE 0: Baixe o arquivo de código disponibilizados pelo professor no Classroom (arquivo `CC_Lab03_src_c.zip`). Entenda a interface das estruturas descritas no arquivo `tables.h`. O arquivo `tables.c` correspondente provê uma implementação miserável (sequencial) dessas funções. A ideia aqui é focar na *utilização* das tabelas e não na sua implementação.

ATIVIDADE 1: Utilizando o código disponibilizado, modifique o seu *scanner* do laboratório anterior para incluir as *strings* da entrada na tabela de *strings*. Utilize uma variável global para armazenar um ponteiro para essa tabela. A criação da tabela deve ser feita na função principal do *parser*, antes da chamada da função `yyparse`. Lembre-se de destruir a tabela ao final da compilação para evitar vazamentos de memória.

ATIVIDADE 2: Utilizando o código disponibilizado, modifique o seu *parser* do laboratório anterior para incluir as variáveis declaradas na tabela de símbolos. Siga as mesmas instruções gerais da atividade anterior para criação dessa nova tabela.

Inclua ações semânticas (isto é, código C) nas regras sintáticas da gramática que lida com a declaração de variáveis do programa de entrada. (Para relembrar, veja o Exemplo 03 e o Exercício 04 do Laboratório 02, que mostram como utilizar regras semânticas no `bison`.) Ao reconhecer uma nova variável, o seu analisador deve verificar se ela já foi declarada consultando a tabela de símbolos. Passando nesse teste, inclua a variável (identificador) na tabela, juntamente com as seguintes informações: linha de declaração e tipo da variável (inteiro, real, etc).

Após preencher a tabela de símbolos com as declarações de variáveis, inclua novas ações semânticas, agora para verificar todas as ocorrências de variáveis no corpo do programa. Em todas as regras sintáticas aonde aparece um *token* `ID`, inclua uma ação semântica para testar se a variável foi previamente declarada (basta consultar a tabela de símbolos). Caso o teste falhe, seu compilador deve exibir uma mensagem de erro, como descrito abaixo.

2.2 Mensagens do analisador semântico

O seu analisador semântico deve exibir as seguintes mensagens.

Utilização de variáveis. Se uma variável for utilizada sem ser declarada, imprima no terminal:

```
SEMANTIC ERROR (XX): variable 'VV' was not declared.
```

Onde `XX` é o número da linha do programa onde o erro foi detectado e `VV` é o nome da variável.

Redeclarações de variáveis. Se uma variável for redeclarada no programa de entrada, imprima no terminal:

```
SEMANTIC ERROR (XX): variable 'VV' already declared at line YY.
```

Onde `XX` é o número da linha do programa onde o erro foi detectado, `VV` é o nome da variável e `YY` é o número da linha aonde a variável foi originalmente declarada.

Impressão das tabelas. Ao final do processo de análise, imprima as tabelas de símbolos e de *strings* no terminal. Veja os arquivos de saída disponibilizados para testes.

Demais mensagens. As mensagens de erros léxicos e sintáticos são as mesmas do Laboratório 02 e devem continuar sendo exibidas como antes.

3 Implementado as Tabelas de Símbolos e de Literais

ATIVIDADE 3 (opcional): Implemente a sua versão da tabela de símbolos e de *strings*, substituindo a versão disponibilizada pelo professor. Idealmente, você deve implementar uma tabela *hash*.

Algumas observações importantes:

- O seu compilador pode terminar a execução ao encontrar o primeiro erro no programa de entrada.
- Os programas de entrada para teste são os mesmos de sempre (`in.zip`). As saídas esperadas desta tarefa estão no arquivo `out03_c.zip`.