Relatório - Trabalho 2 : Análise Sintática

Disciplina: SCC0217 - Compiladores

Data de entrega: 09/06/2019

Integrantes:

Antonio Moreira - 9778943
Gabriel Scalici - 9292970
Henrique Freitas - 8937225
Luca Porto - 9778943
Romeu Bertho - 7151905

Introdução

O trabalho tem como objetivo a implementação de um analisador léxico, isto é, o reconhecimento de tokens do código-fonte para que tais informações possam ser enviadas à análise sintática e dar continuidade ao processo de compilação.

Foi utilizado a linguagem *C* para o projeto com a biblioteca *Flex* do primeiro trabalho e a biblioteca *YACC para* a construção da gramática e possibilidade de construir o analisador sintático em conjunto com o léxico.

O ambiente utilizado para o projeto foi o sistema operacional *Linux* e os editores *Atom, VisualCode* e *SublimeText*.

Como compilar:

O trabalho já foi previamente compilado (no sistema operacional macOS), porém pode ser compilado novamente no computador do usuário. Foi testado no Linux, de forma que consta que funciona da mesma maneira.

Para a compilação basta utilizar o comando *"make",* com as bibliotecas flex e bison instaladas.

Como testar:

Foram disponibilizados alguns programas de teste na pasta *"test"*, lembrando que há programas teste de cada uma das partes do projeto, inclusive da primeira parte (análise léxica). Separados em pasta *"lex"* e *"sin"* para programas da análise léxica e sintática respectivamente.

Para executar basta digitar *"make run",* para os programas colocados na pasta *"sem",* onde também serão salvos os arquivos de saída com todos os passos do programa, respectivos erros e demais.

Estrutura básica:

Como exemplo de algumas estruturas principai do código, temos anexado trechos dos códigos com nossas definições.

```
/* Tipos de dados */
enum type_e {
    TYPE_PROGRAM,
    TYPE_FUNCTION,
    TYPE_STRING,
    TYPE_CONST,
    TYPE_INTEGER,
    TYPE_FLOAT,
    TYPE_CHAR,
    TYPE_VAR,
    TYPE_PROC,
    TYPE_NONE
};
```

(Tipos de dados)

```
/* Valores de dados */
union val_u {
    int    intval;
    double floatval;
    char   charval;
    char   *strval;
};
```

(Valores possíveis)

```
/* Estrutura de dado que tem tipo e valor */
struct data_s {
    enum type_e type;
    union val_u value;
};
```

(Estrutura dos dados utilizados com struct para armazenar duas informações)

```
/* Estrutura para guardar um simbolo */
struct sym_s {
    int           scope;
    enum type_e   type; /* TYPE_VAR ou TYPE_PROC */
    char          id[MAX_LENGTH_IDENT + 1];
    struct data_s data;
    struct sym_s  **parlist;
    int           n_parlist;
};
```

(Para guardar o símbolo, também com uma struct contendo diversas informações)

Organização:

- Makefile : Arquivo para facilitar a compilação do projeto e execução.
- Relatorio: Relatório contendo as informações do projeto.
- main: programa compilado.
- lalg.l : programa Lex para a linguagem LALG
- lalg.y : programa em Bison para a linguagem LALG.
- test: pasta contendo os arquivos de teste.

Decisões de projeto:

- **lalg.y**

Para armazenar os símbolos, foram utilizadas tabelas hash da biblioteca <search.h>. Uma tabela hash é utilizada por escopo, ou seja, ao necessitar de um novo escopo, e uma nova tabela para a armazenar

Foi utilizado *union* para os dados, como por exemplo, struct *data_s* guarda o tipo e o valor de um dado.

A detecção dos erros semânticos manipula as tabelas hash e estruturas de dados auxiliares como listas temporárias de variáveis, parâmetros e argumentos.

Conclusão:

O trabalho apresentou uma dificuldade maior do que a esperada pelo grupo, tendo em vista o trabalho de análise semântica que se mostrou mais simples. Foi percebido a tamanha ligação entre as diferentes análise e a complexidade de cada uma delas, tendo seu papel definido dentro de um compilador.

Compilador não será muito utilizado na prática (salvo problemas com parser), porém observamos ser de muita importância o conhecimento dos seus mecanismos e tecnologias, de forma a compreender uma linguagem de comutação mais a fundo.

Referências bibliográficas:

- Analisador sintático:
  http://osorio.wait4.org/oldsite/compil/tutoria/exercicios-yacc-bison.pdf

- Introdução à compiladores:
  http://producao.virtual.ufpb.br/books/tautologico/compiladores-livro/livro/livro.pdf

- Documentação bison:
  http://www.gnu.org/software/bison/manual/html_node/index.html