INE5416 - Paradigmas de Programação (2015/2)
Gustavo Zambonin
Relatório 2 - Estrutura das linguagens

Questão 1

- Yacc é um programa que constrói a parte de um compilador responsável por analisar a parte sintática de um código-fonte qualquer, baseado em uma gramática de livre contexto (um conjunto de regras de produção para palavras numa linguagem formal). Desenvolvido na AT&T em meados da década de 1970, existem diversas versões do Yacc, sejam elas abertas ou não.

- Lex é um programa que gera analisadores léxicos. Geralmente é utilizado em conjunto com o Yacc, entregando ao usuário o código-fonte do analisador na linguagem de programação C. Recomenda-se produzir a entrada do analisador sintático com este programa a partir de expressões regulares. Internamente, ele funciona através de autômatos finitos determinísticos (máquinas que aceitam linguagens regulares).

- Flex é a versão aberta do Lex. Funciona de maneira similar, de modo a ler *tokens* e reconhecê-los como certas funções da linguagem de programação. Estes *tokens* podem ser operadores, palavras-chave, identificadores, números etc.. Esta é a primeira fase do funcionamento de um compilador atualmente.

- Bison é a versão aberta do Yacc. Novamente, funciona de maneira similar. Estes programas executam a segunda fase do funcionamento de um compilador, analisando se a sequência de *tokens* faz sentido para a sintaxe especificada pela gramática. Estas ferramentas são essenciais para a formalização da descrição de uma linguagem de programação e intrínsecas ao funcionamento de computadores atualmente.

Questão 2

- `scan.l` é uma tabela de expressões regulares e fragmentos de código, que é transformada então para um programa que lê uma entrada qualquer, dividindo-a em partes que assemelham-se às expressões dadas. Este reconhecimento é feito utilizando um autômato gerado pelo analisador léxico. Nota-se, por exemplo, que o programa reconhecerá diversos *tokens* diferentes como sendo constantes decimais ou hexadecimais, mas para uma palavra reservada, a função correspondente será sempre executada.

```
0[xX]{H}+{IS}?      { count(); return(CONSTANT); }
0{D}+{IS}?          { count(); return(CONSTANT); }
```

  Ex.: neste caso, uma constante pode ser reconhecida como hexadecimal ou decimal.

- `y.tab.h` nada mais é do que um cabeçalho que especifica diversas funções específicas à linguagem de programação C e nomeia-as de modo amigável.

- `gram.y` decide exatamente como o compilador deve entender o que é escrito a partir desta ordem sintática, incluindo a ordem dos *tokens* e diversas variações do que pode ser expressado, através de uma gramática.

```
iteration_statement
    : WHILE '(' expr ')' statement
    | DO statement WHILE '(' expr ')' ';'
    | FOR '(' ';' ';' ')' statement
    | FOR '(' ';' ';' expr ')' statement
    | FOR '(' ';' expr ';' ')' statement
    | FOR '(' ';' expr ';' expr ')' statement
    | FOR '(' expr ';' ';' ')' statement
    | FOR '(' expr ';' ';' expr ')' statement
    | FOR '(' expr ';' expr ';' ')' statement
    | FOR '(' expr ';' expr ';' expr ')' statement
    ;
```

  Ex.: em `iteration_statement`, observa-se que é possível abrigar diversas expressões booleanas dentro de uma simples operação FOR, enquanto existe apenas uma possibilidade para a operação WHILE.