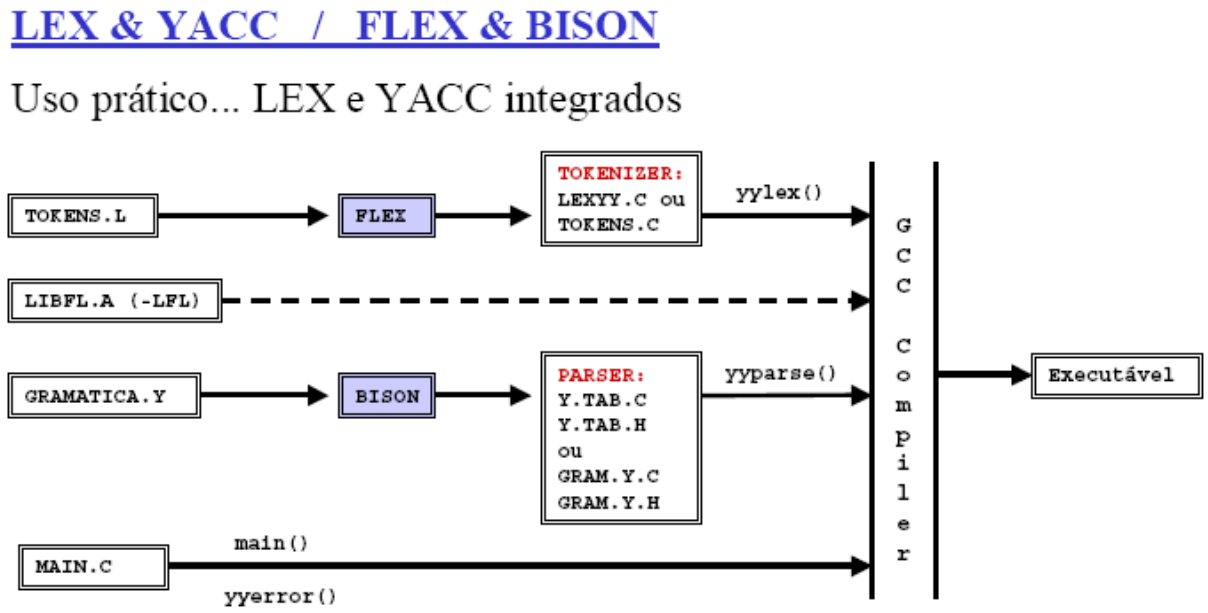
UNISINOS - UNIVERSIDADE DO VALE DO RIO DOS SINOS
CIÊNCIAS EXATAS E TECNOLÓGICAS – Curso de Informática

COMPILADORES I

Disciplina: **Compiladores I**
Professor responsável: ***Fernando Santos Osório***
Semestre: 2006/2
E-mail: ***fosorio@ unisinos.br***
Web: ***http://inf.unisinos.br/~osorio/compil.html***
Horário: ***Tutoria***

GERADORES DE ANALISADORES SINTÁTICOS – YACC / BISON

O **YACC/BISON** serve para gerar automaticamente programas para análise sintática (usualmente em "C") de códigos fonte de uma linguagem específica qualquer. Estas ferramentas possuem como entrada uma descrição de uma gramática, que especifica uma determinada linguagem, e gera como saída um programa em C ou C++ que será o *parser* desta linguagem. Uma vez compilado este *parser*, gerado automaticamente pelo YACC/BISON, ele terá como entrada arquivos com códigos fonte da linguagem especificada pela gramática, executando assim a "validação" códigos fonte através da execução deste *parser*. O *parser* gerado pelo YACC/BISON permite realizar apenas a validação do código fonte, indicando se está correto sintaticamente ou não, mas também permite implementar de modo bastante simples e direto ferramentas como interpretadores (executando diretamente ações a medida que vai reconhecendo comandos corretos na linguagem especificada) ou tradutores (realizando uma tradução dirigida pela sintaxe) e pode também servir de base para a criação de compiladores mais completos e sofisticados. Note que os *tokens* gerados pelos programas criados pelo LEX/FLEX são usualmente fornecidos como entrada para o processo seguinte de análise sintática realizado pelo programa criado pelo YACC/BISON, ou seja, ambos atuam de forma integrada.



EXEMPLO DE ARQUIVO BATCH (DOS/GCC), INTEGRANDO O USO DO BISON E FLEX:

```
bison -y -d %1.y
flex -i %1.l
gcc y.tab.c lex.yy.c -o%1.exe -lfl
```

TRABALHO PRÁTICO

LISTA DE EXERCÍCIOS 2

Implementar Analisadores SINTÁTICOS para de acordo com o descritos abaixo, enviando os arquivos do Lex/Flex (*.l) e do Yacc/Bison (*.y) para o professor por e-mail (fosorio@gmail.com) até a 11ª. semana do semestre.

1. Considere a seguinte linguagem, SIMPLE cuja gramática aparece descrita em: http://memphis.compilertools.net/interpreter.html (considere apenas o léxico e a gramática!) (também pode ser obtida em: http://inf.unisinos.br/~osorio/compil/43/exemplos-compil/simple.zip)

   Usando esta linguagem, faça:

   I) Altere o analisador léxico de modo que os número reconhecidos possam ser de 2 tipos: números inteiros (positivos ou negativos) e números decimais (com ponto decimal, positivos ou negativos, onde entretanto a *notação científica **NÃO** deve ser considerada válida*);

   II) Altere o analisador léxico de modo que os nomes de variáveis possam aceitar nomes tradicionais de variáveis, ou seja, iniciados por uma letra e seguidos de letras, dígitos ou caracteres de sublinhado (underscore = "_"), tendo um comprimento máximo de até 10 caracteres;

   III) Inclua na linguagem uma parte separada no código usada para a declaração explícita de variáveis (seção DECL), e uma outra parte separada para a implementação do programa (seção CODE), onde ficarão os comando já definidos atualmente na linguagem. Considere a seguinte descrição abaixo da nova sintaxe desta linguagem SIMPLE:

   DECL
   <nome_var> : <tipo_var>; ...

   CODE

   <statement>; ...

   END

   Onde: <tipo_var> pode ser de 2 tipos – "INT" ou "REAL"

   IV) Inclua na gramática da linguagem o comando REPEAT <comando> UNTIL <condição>

   Crie pelo menos 3 programas fontes usando esta linguagem (*.smp) e explorando os recursos por ela oferecidos (programas sintaticamente corretos) e envie para o professor como resposta deste exercício um arquivo (zip) contendo os seguintes arquivos: simple.l, simple.y, prog1.smp, prog2.smp e prog3.smp.

2. Considere a linguagem SIMPLE melhorada e ampliada e que foi implementada no exercício anterior. Implemente um programa do tipo XREF (*Cross-reference*) para esta linguagem, onde o objetivo do XREF é apresentar uma listagem de referências as variáveis declaradas e usadas no programa. A saída do programa deve conter uma tabela da forma descrita mais abaixo, sempre listando o nome das variáveis e as linhas (contador da linha de código) onde foi encontrada uma referência a esta variável. Veja a seguir um exemplo de programa e a respectiva saída que deve ser obtida pelo programa XREF:

Programa SIMPLE (Exemplo.smp)

```
Linha   Código
1:      DECL
2:              A1: INT;
3:              VALOR: INT;
4:              X: REAL;
5:              Y: REAL;
6:              Z: REAL;
7:      CODE
8:              X := 8;
9:              Y := 12;
10:             WHILE X != Y
11:             DO
12:                     IF X > Y
13:                     THEN X := X-Y
14:                     ELSE  Y := Y-X
15:                     FI;
16:             OD;
17:             VALOR := 0.1;
18:             PRINT VALOR;
19:             PRINT X;
20:     END
```

Tabela de *Cross-References* - XREF

Variável	**Declaração**	**Atribuição de Valor (escrita)**	**Uso da Variável (leitura)**
A1	2 (Int)		
VALOR	3 (Int)	17	18
X	4 (Real)	8 13	10 12 13 14 19
Y	5 (Real)	9 14	10 12 13 14
Z	6 (Real)		

Envie para o professor como resposta deste exercício um arquivo (zip) contendo os seguintes arquivos: smp-xref.l, smp-xref.y.

OBS: *Note que nas linhas 3 e 17 ocorre um erro de semântica (incompatibilidade de tipos), mas por enquanto este programa está focado apenas em uma análise léxica e sintática. Erros como este não serão detectados nesta análise léxico-sintática.*

3. Considere um subset da linguagem pascal (SIMPAS) e implemente a mesma funcionalidade do exercício anterior, um programa de XREF (*cross-reference*) para esta linguagem. Dica: a descrição da linguagem SIMPAS pode ser obtida em:
   http://www.inf.unisinos.br/~osorio/compil/43/exemplos-compil/simpas.zip
   ou se preferir um pascal mais completo (PASFULL):
   http://www.inf.unisinos.br/~osorio/compil/43/exemplos-compil/

BOM TRABALHO!