TEORIA DA COMPUTAÇÃO E COMPILADORES

DESENVOLVIMENTO DE COMPILADOR

PROF. FERNANDO KAKUGAWA

Instruções para Entrega da Atividade:

- Este trabalho deverá ser desenvolvido em grupos de, **no máximo**, 5 alunos;
- Forma de entrega: o código fonte, a aplicação executável e códigos de teste;
- Durante a apresentação deve haver a descrição sobre a equivalência da Linguagem de cada grupo com a Linguagem Java;
- Data de Entrega: **30/11/2023 às 23:59;**
- Durante a aula do dia 01/12 e 08/12 cada grupo deverá apresentar a Linguagem de Programação implementada.

Enunciado:

Cada grupo deve definir a sua própria gramática e todos os tokens necessários. Os requisitos **mínimos** são:

- Deve ter 3 tipos de variáveis, sendo que as variáveis;
- Deve ter a estrutura de controle **`if ... else`**;
- Deve ter duas estruturas de repetição (**`while , do ... while , for`**);
- A parte de expressões envolvendo os operadores matemáticos deve ser realizada de maneira correta, respeitando a precedência. Não é necessário gerar a resposta da expressão, basta cuidar da precedência entre os operadores matemáticos através da gramática;
- As atribuições também devem ser realizadas;
    - É necessário verificar se é possível realizar as operações, devido aos tipos das variáveis e ao seu escopo.
- Os comandos de leitura do teclado (**`scanf`**) e de impressão na tela (**`printf`**) devem ser disponibilizados.
- O compilador tem que aceitar números decimais.
- A cada utilização de uma variável, é necessário verificar se ela já foi declarada.

O compilador deve fazer a conversão de um programa desenvolvido na Linguagem definida pelo grupo para a Linguagem Java.

A verificação da corretude do programa será realizada compilando o arquivo gerado pelo compilador desenvolvido.

Seu compilador deverá receber como entrada um arquivo contendo um programa escrito na Linguagem definida pelo grupo e gerar uma forma equivalente em Java, que deverá ser compilada no compilador javac, executada na JVM e não deverá conter erros.

OBS: a gramática não pode conter recursividade à esquerda e produções vazias (que porventura venham a surgir). Caso seja necessário, efetue sua fatoração à esquerda.

Exemplo de um Compilador

A descrição a seguir ilustra um exemplo de um Compilador que faz a conversão de um programa desenvolvido em uma linguagem fictícia para uma forma equivalente na linguagem C.

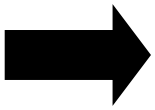
Os termos em negrito significam palavras reservadas. Preste muita atenção aos sinais de pontuação.

```
Prog → programa Declara Bloco fimprog.
Declara → (inteiro | decimal) Id (, Id)* .
Bloco → (Cmd )+
Cmd → CmdLeitura | CmdEscrita | CmdExpr | CmdIf
CmdLeitura → leia ‘(‘ Id ‘)’.
CmdEscrita → escreva ’(‘ Texto | Id ‘)’.
CmdIf → if ‘(‘ Expr Op_rel Expr ‘)’ ‘{‘ Cmd+ ‘}’ (else ‘{‘ Cmd+ ‘}’ )?
CmdExpr → Id ‘:’’=’ Expr .
Op_rel → ‘<’ | ‘>’ | ‘<’’=’ | ‘>’’=’ | ‘!’’=’ | ‘=’’=’
Expr → Expr ‘+’ Termo | Expr ‘-‘ Termo | Termo
Termo → Termo ‘*’ Fator | Termo ‘/’ Fator | Fator
Fator → Num | Id | ‘(‘ Expr ‘)’
Texto → ’ “ ’(0..9 | a..z | A..Z | ‘ ‘ )+ ’ ” ’
Num → (0..9)+
Id → (a..z | A..Z) (a..z | A..Z | 0..9)*
```

OBS: espaços em branco, tabs e enter podem aparecer e devem ser eliminados.

Entrada:input.in

```
programa

inteiro a,b,c.
decimal d.
escreva(“Programa Teste“).
escreva (“Digite A“).
leia (a).
escreva (“Digite B“).
leia (b).

if (a<b)
{
   c := a + b.
}else
{
   c := a - b.
}

escreva (“C e igual a “).
escreva (c).

d := c / (a + b).

escreva (“D e igual a “).
escreva (d).

fimprog.
```



Saída:input.c

```
#include <stdio.h>

void main(void)
{
int a,b,c;
double d;
printf(“Programa Teste“);
printf(“Digite A“);
scanf(“%d“,&a);
printf(“Digite B“);
scanf(“%d“,&b);

if (a<b)
{
   c = a + b;
}else
{
   c = a - b;
}

printf(“C e igual a “);
printf(“%d“,c);

d = c / (a + b);

printf(“D e igual a “);
printf(“%lf“,d);
}
```