Compiladores
Análise Sintática
BNF *TINY*

Prof. Dr. Luiz Eduardo G. Martins
(adaptado por Profa Dra Ana Carolina Lorena)
UNIFESP

A Linguagem *TINY*

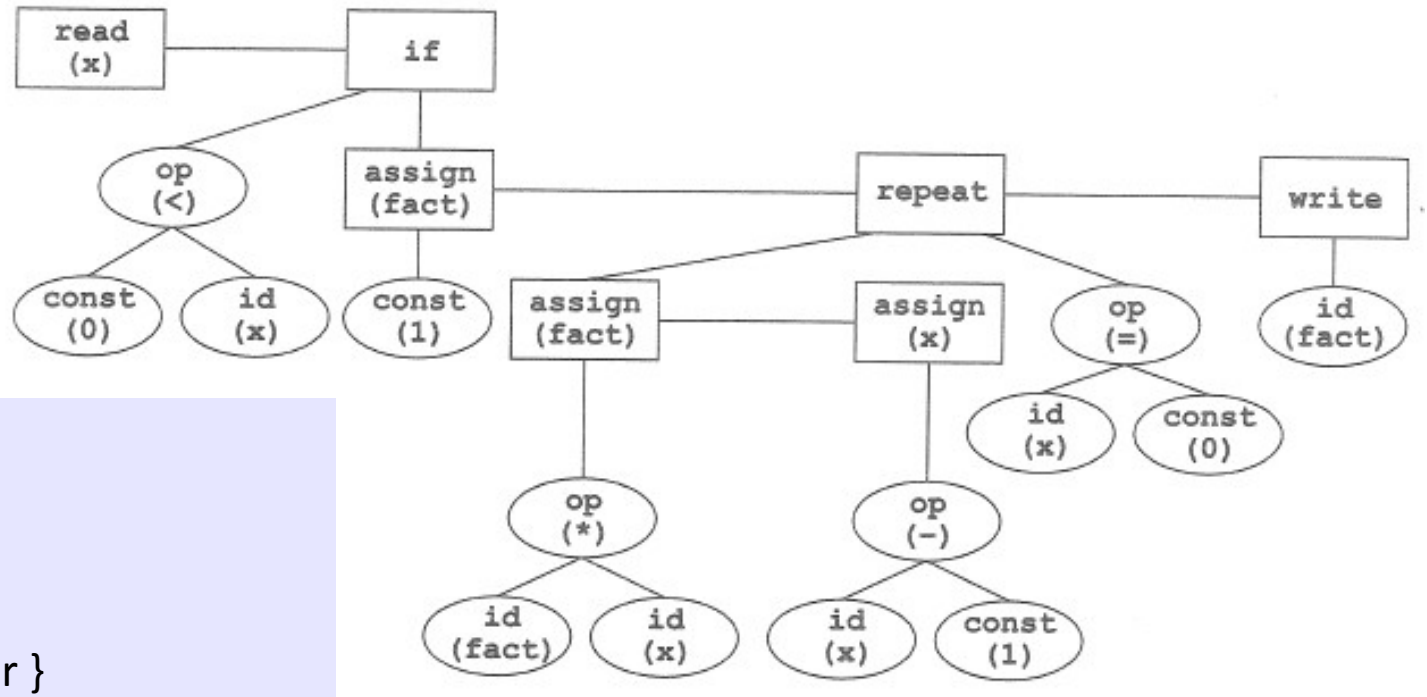
- Programa TINY é uma sequência de declarações
  - Cinco tipos de declarações: condição, repetição, leitura, escrita e atribuição
  - end no bloco if para não ter problema do else pendente
  - Expressões aritméticas associativas à esquerda e com precedências usuais
  - Uma operação de comparação por expressão
  - Sequências de declarações precisam ser separadas por ;
  - Não há ; após declaração final de uma sequência

A Linguagem *TINY*

programa → *decl-seqüência*
decl-seqüência → *decl-seqüência* **;** *declaração* | *declaração*
declaração → *cond-decl* | *repet-decl* | *atrib-decl* | *leit-decl* | *escr-decl*
cond-decl → **if** *exp* **then** *decl-seqüência* **end**
| **if** *exp* **then** *decl-seqüência* **else** *decl-seqüência* **end**
repet-decl → **repeat** *decl-seqüência* **until** *exp*
atrib-decl → **identificador** **:=** *exp*
leit-decl → **read identificador**
escr-decl → **write** *exp*
exp → *exp-simples comp-op exp-simples* | *exp-simples*
comp-op → **<** | **=**
exp-simples → *exp-simples soma termo* | *termo*
soma → **+** | **-**
termo → *termo mult fator* | *fator*
mult → * | /
fator → **(***exp***)** | **número** | **identificador**

Figura 3.6 Gramática da linguagem TINY em BNF.

A Linguagem *TINY*



ura 3.9 Árvore sintática para o programa TINY da Figura 3.8.

```
{ Sample program
  in TINY language -
  computes factorial
}
read x; { input an integer }
if 0 < x then { don't compute if x <= 0 }
  fact := 1;
  repeat
    fact := fact * x;
    x := x - 1
  until x = 0;
  write fact  { output factorial of x }
end
```

A Linguagem *TINY*

```
typedef enum {StmtK,ExpK} NodeKind;
typedef enum {IfK,RepeatK,AssignK,ReadK,WriteK}
             StmtKind;
typedef enum {OpK,ConstK,IdK} ExpKind;

/* ExpType é utilizado para verificação de tipos */
typedef enum {Void,Integer,Boolean} ExpType;

#define MAXCHILDREN 3

typedef struct treeNode
   {  struct treeNode * child[MAXCHILDREN];
      struct treeNode * sibling;
      int lineno;
      NodeKind nodekind;
      union { StmtKind stmt; ExpKind exp;} kind;
      union { TokenType op;
              int val;
              char * name; } attr;
      ExpType type; /* para verificação de tipos de expressões */
   } TreeNode;
```

Figura 3.7 Declarações em C para um nó de árvore sintática TINY.

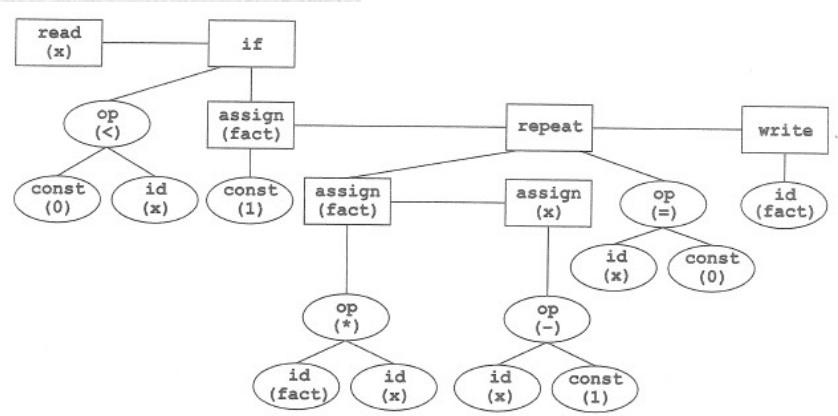


Figura 3.9 Árvore sintática para o programa TINY da Figura 3.8.

A Linguagem *TINY*

Tipos principais de nós: `statement (StmtK) e expression (ExpK)`

Em util.c:

```
TreeNode * newStmtNode(StmtKind kind)
{ TreeNode * t = (TreeNode *) malloc(sizeof(TreeNode));
  int i;
  if (t==NULL)
    fprintf(listing,"Out of memory error at line %d\n",lineno);
  else {
    for (i=0;i<MAXCHILDREN;i++) t->child[i] = NULL;
    t->sibling = NULL;
    t->nodekind = StmtK;
    t->kind.stmt = kind;
    t->lineno = lineno;
  }
  return t;
}
```

A Linguagem *TINY*

Tipos principais de nós: `statement (StmtK) e expression (ExpK)`

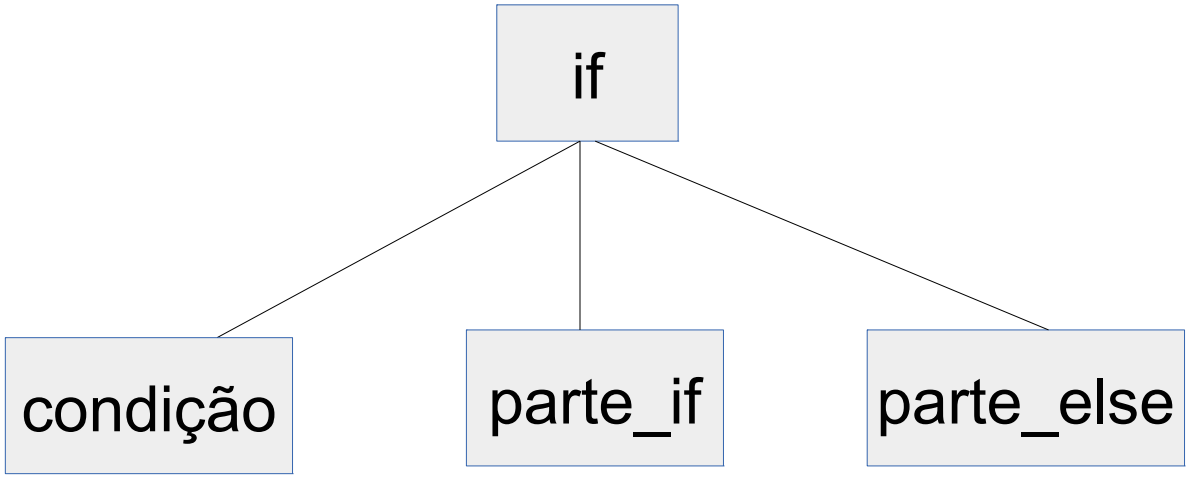
Em util.c:

```
TreeNode * newExpNode(ExpKind kind)
{ TreeNode * t = (TreeNode *) malloc(sizeof(TreeNode));
  int i;
  if (t==NULL)
    fprintf(listing,"Out of memory error at line %d\n",lineno);
  else {
    for (i=0;i<MAXCHILDREN;i++) t->child[i] = NULL;
    t->sibling = NULL;
    t->nodekind = ExpK;
    t->kind.exp = kind;
    t->lineno = lineno;
    t->type = Void;
  }
  return t;
}
```

A Linguagem *TINY*

Tipos de nós statment: `IfK, RepeatK, AssignK, ReadK, WriteK`

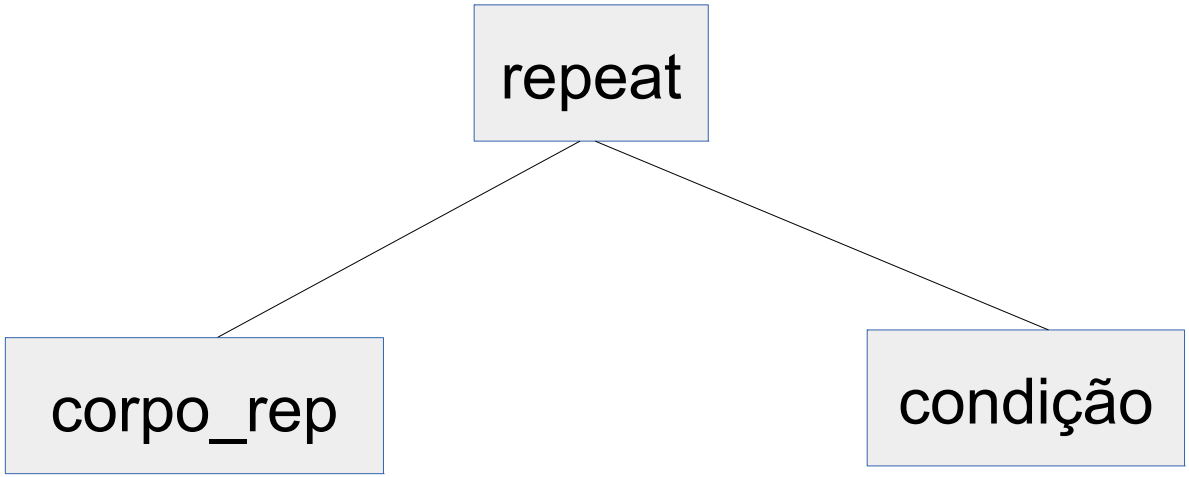
`IF:`



A Linguagem *TINY*

Tipos de nós statment: `IfK, RepeatK, AssignK, ReadK, WriteK`

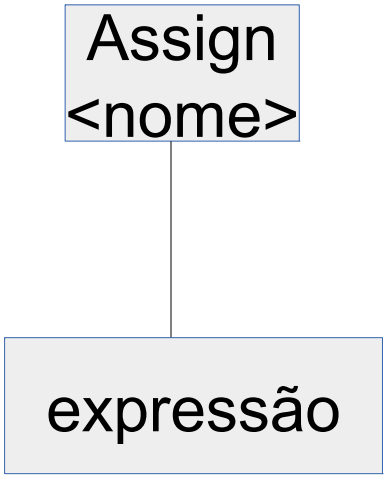
`REPEAT:`



A Linguagem *TINY*

Tipos de nós statment: `IfK, RepeatK, AssignK, ReadK, WriteK`

`ASSIGN:`



A Linguagem *TINY*

Tipos de nós statment: `IfK, RepeatK, AssignK, ReadK, WriteK`

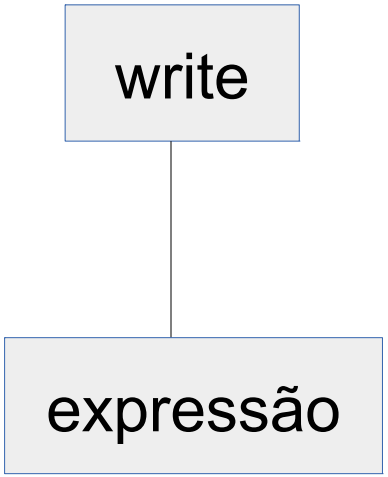
`READ:`

Read
<nome>

A Linguagem *TINY*

Tipos de nós statment: `IfK, RepeatK, AssignK, ReadK, WriteK`

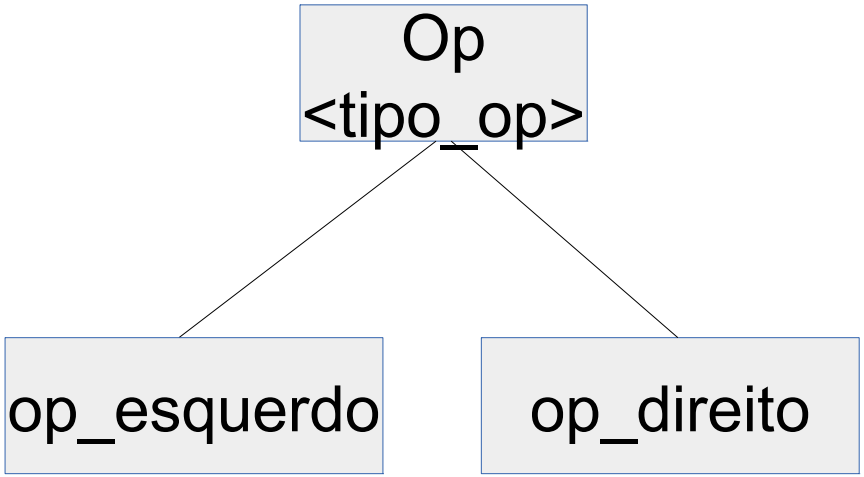
`WRITE:`



A Linguagem *TINY*

Tipos de nós expression: `OpK,ConstK,IdK`

`OP:`



A Linguagem *TINY*

Tipos de nós expression: `OpK,ConstK,IdK`

`Const:` para armazenar números

Const
<valor>

`Id:` para armazenar identificadores

Id
<nome>

A Linguagem *TINY*

- Bibliografia consultada

Capítulo 3: LOUDEN, K. C. **Compiladores: princípios e práticas.** São Paulo: Pioneira Thompson Learning, 2004