Manual compiladores

Guilherme Gonçalves, Dionathan Tomaz e Allan Nicolete

Terminais

Código	Token
1	while
2	void
3	string
4	return
5	numerointeiro
6	numerofloat
7	nomevariavel
8	nomedochar
9	nomedavariavel
10	nomedastring
11	main
12	literal
13	integer
14	inicio
15	if
16	Î
17	for
18	float
19	fim
20	else
21	do
22	cout
23	cin
24	chat
25	callfuncao

26	>>
27	>=
28	>
29	==
30	=
31	<=
32	<<
33	<
34	++
35	+
36	}
37	{
38	;
39	:
40	/
41	,
42	*
43	)
44	(
45	$
46	!=
47	--
48	-

Não-terminais

Código	Token
49	BLOCO
50	DLCVAR
51	DCLFUNC

52	CORPO
53	REPIDENT
54	TIPO
55	LDVAR
56	REPIDEN
57	LID
58	TIPO_RETORNO
59	DEFPAR
60	VALORRETORNO
61	PARAM
62	LPARAM
63	COMANDO
64	REPCOMANDO
65	EXPRESSAO
66	PARAMETROS
67	TPARAM
68	REPPAR
69	COMPARACAO
70	ELSEPARTE
71	CONTCOMPARACAO
72	INCREMENTO
73	SEQCOUT
74	SEQUENCIA
75	EXPSIMP
76	REPEXPSIMP
77	TERMO
78	REPEXP
79	FATOR
80	REPTERMO

33	<
34	++
35	+
36	}
37	{
38	;
39	:
40	/
41	,
42	*
43	)
44	(
45	$
46	!=
47	--
48	-

Manual de linguagem

Estrutura

- A sintaxe do programa é:

  ```
  program ident ; DECLARACOES BLOCO .
  ```

- Isso significa que um programa começa com a palavra-chave `program`, seguida de um identificador (`ident`), um ponto e vírgula (`;`), depois a parte de declarações (`DECLARACOES`) e finalmente o bloco de comandos (`BLOCO`), que termina com um ponto (`.`).

Comentários

Comentários são trechos de texto ignorados pelo compilador, usados para documentar o código, explicar lógicas ou marcar sessões importantes. Eles não afetam a execução do programa.

- **Comentário de Linha**
- **Sintaxe**: \\ comentário aqui
  Tudo que estiver após \\ na mesma linha será ignorado pelo compilador.

- **EXEMPLO:**
  \\ Este é um comentário de linha
  var idade : integer; \\ Variável para armazenar a idade

- **Comentário de Bloco**
- **Sintaxe**: Cada linha começa com \\
  Para escrever comentários em várias linhas, basta iniciar cada uma com \\.Explicações detalhadas, instruções ou documentação maior.

- **EXEMPLO:**
  \\ Este procedimento realiza a saudação personalizada.
  \\ Ele recebe um nome como parâmetro e exibe uma mensagem.
  procedure saudacao(nome : string) ;
  inicio
  print { "Olá, ", nome, "!" };
  fim ;

Tipo de dados

Os tipos de dados definem a natureza dos valores que podem ser armazenados.

- Integer para inteiros (1,2,3)
- Real para números decimais (1.5, 2.6, 3.7)
- String para cadeia de caracteres (exemplo, demonstração)

Declarações

O programa pode ter três tipos de declarações:

- **CONSTANTES**: Declarações de constantes.
- **VARIAVEIS**: Declarações de variáveis.
- **PROCEDIMENTOS**: Declarações de procedimentos (funções ou sub-rotinas).

Constantes

- **CONSTANTES**: São definidas com a palavra-chave `const`, seguida de um identificador, um sinal de igual, e um valor inteiro (`nint`), seguido por um ponto e vírgula. Constantes podem ser seguidas por outras constantes.

  EXEMPLO:

  const TAMANHO_MAX = 100;

  const VELOCIDADE = 5;

  const VIDAS_INICIAIS = 3;

Variáveis

A sintaxe para declarar variáveis é: *var LISTAVARIAVEIS : TIPO ; LDVAR*

- Isso define variáveis a partir de uma lista (`LISTAVARIAVEIS`), seguida de um tipo (`TIPO`) e um ponto e vírgula. Também pode haver uma parte adicional (`LDVAR`) que permite declarar mais variáveis em seguida.

  EXEMPLO:

  var contador, limite : nint;

  var nome, sobrenome : texto;

  var ativo : logico;

LISTAVARIAVEIS: É uma lista de identificadores separados por vírgulas (como `ident, ident, ...`).

TIPO: Pode ser `integer`, `real` ou `string`, que são os tipos de dados permitidos.

- **INTEGER:**

  Descrição: Números inteiros positivos ou negativos, sem casas decimais.

  Separadores: Não possui separadores. É escrito diretamente com os dígitos.

  Limites: Dependem da implementação, mas geralmente entre `-32.768` e `32.767`

  EXEMPLO:

  var idade : integer;

  idade := 25;

  const LIMITE_MAXIMO = 1000;

- **REAL:**

  Descrição: Números reais com parte decimal, positivos ou negativos.

  Separadores: Usa ponto (`.`) como separador decimal (e não vírgula).

  Limites: Precisão padrão até 6 ou 7 casas decimais.

- **STRING:**

  Descrição: Texto entre aspas duplas.

  Separadores: Os textos devem sempre ser delimitados por aspas duplas (").

  Limites: Pode variar, mas normalmente até 255 caracteres.

Procedimentos

PROCEDIMENTOS: A sintaxe de um procedimento é:
procedure ident PARÂMETROS ; BLOCO ; PROCEDIMENTOS

- Um procedimento tem um identificador, parâmetros (opcionais) e um bloco de comandos, e pode ser seguido por outro procedimento.
- **PARAMETROS**: São as variáveis que o procedimento recebe. Eles são listados entre parênteses, com cada um consistindo em um identificador e tipo.

- **EXEMPLO PROCEDIMENTO SEM PARÂMETROS:**

```
procedure mostrarMensagem ;
inicio
   escreva("Olá, mundo!");
fim ;
```

- **EXEMPLO PROCEDIMENTO COM PARÂMETROS:**

```
procedure saudacao(nome : string) ;
inicio
   escreva("Olá, ", nome, "!");
fim ;
```

Bloco de Comandos

- **BLOCO**: Um bloco de comandos é um conjunto de comandos entre `begin` e `end`.
- **COMANDOS**: Pode ter uma sequência de comandos, cada um seguido por um ponto e vírgula. O bloco pode ser vazio (`î` representa a ausência de comandos).

Comandos

- **print**: Imprime expressões ou valores (com a sintaxe `{ ITEMSAIDA REPITEM }`).
- EXEMPLO:

```
print { "Resultado: ", soma };
```

- **if EXPRELACIONAL then BLOCO ELSEOPC**: Uma estrutura condicional. Se a expressão relacional for verdadeira, o bloco é executado, e pode haver um `else` opcional.
- EXEMPLO:

```
if idade >= 18 then
inicio
   print { "Maior de idade" };
fim
else
inicio
   print { "Menor de idade" };
fim ;
```

- **ident CHAMADAPROC**: Chamada de procedimento com parâmetros.

- EXEMPLO:

  saudacao("Carlos");

- **for ident := EXPRESSAO to EXPRESSAO do BLOCO**: Um loop `for` com um índice de controle (`ident`), que vai de um valor de expressão a outro e executa um bloco de comandos.
- EXEMPLO:

  for i := 1 to 5 do

  inicio

    print { "Valor de i: ", i };

  fim ;

- **while EXPRELACIONAL do BLOCO**: Um loop `while`, que repete o bloco enquanto a expressão relacional for verdadeira.
- EXEMPLO:

  while contador > 0 do

  inicio

    print { "Contando: ", contador };

    contador := contador - 1;

  fim ;

- **read (ident)**: Leitura de um valor e atribuição a uma variável.
- EXEMPLO:

  read(idade);

Expressões e Termos

- **EXPRESSAO**: É uma combinação de termos e expressões aritméticas com operações de soma e subtração: *TERMO EXPR*

- **EXPR**: Operações aritméticas (como + ou -).
- **TERMO**: São fatores multiplicados ou divididos, como: FATOR TER
- **FATOR**: Pode ser um número inteiro (`nint`), um número real (`nreal`), um identificador (variável), ou uma expressão entre parênteses.

Expressões Relacionais

- **EXPRELACIONAL**: Compara duas expressões usando operadores como:
  - =, <>, <, >, <=, >=.

Estruturas de Controle

- **if-else**: Condicional onde um bloco de código é executado se a condição for verdadeira e outro pode ser executado no caso contrário.
- **for**: Loop de repetição com um índice controlado.
- **while**: Loop de repetição baseado em uma condição booleana.

Operadores

- **OPREL**: São os operadores relacionais, como igualdade (=) ou desigualdade (<>).
- **Operadores aritméticos**: Adição (+), subtração (-), multiplicação (*), divisão (/).

Literais e Identificadores

- **literal**: Refere-se a uma constante literal (um valor fixo).
- **ident**: Um identificador, geralmente usado para variáveis e procedimentos.

Erros Léxicos

- **Identificadores Mal Formados**

  1variavel — começa com número.

  vari@vel — contém caractere especial não permitido.

  minha_variavel_que_excede_o_limite_de_30_caracteres — excede o limite de comprimento permitido.

- **Números Mal Formados**

  123. — número real sem parte decimal.

  .123 — número real sem parte inteira.

  1.23.45 — múltiplos pontos decimais.

  123a — número com caractere alfabético.

  0xG — número hexadecimal com caractere inválido.

- **Literais de String Mal Formados**

  "Texto sem aspas finais — aspas finais ausentes.

  "Texto com "aspas" internas" — aspas internas não escapadas.

  "Texto com \n nova linha" — sequência de escape inválida.

- **Comentários Mal Formados**

// Comentário sem quebra de linha — comentário de linha única sem quebra de linha.

/* Comentário sem fechamento — comentário multilinha sem fechamento.

- **Símbolos Não Reconhecidos**

@ — símbolo não reconhecido.

— símbolo não reconhecido.

¢ — símbolo não reconhecido.