UNIVERSIDADE METODISTA DE ANGOLA
FACULDADE DE ENGENHARIA E AMBIENTE

Curso de Engenharia Informática
Disciplina de Tópicos Avançados de Compiladores
Professor Engº. Nanitamo António
Semestre 2021/2

Especificação da Linguagem Pascal Simplificado - COVIDLG

Sintaxe do COVIDLG

Programa e Bloco

1. <programa> ::= **program** <identificador> **;** <bloco>**.**
2. <bloco> ::=
   [<parte de declarações de variáveis>]
   [<parte de declarações de sub-rotinas>]
   <comando composto>

Declarações

3. <parte de declarações de variáveis> ::=
   <declaração de variáveis> {**;** <declaração de variáveis>}**;**
4. <declaração de variáveis>::= <tipo> <lista de identificadores>
5. <lista de identificadores> ::= <identificador> {**,** <identificador>}
6. <parte de declarações de subrotinas> ::= {declaração de procedimento> **;**}
7. <declaração de procedimento> ::= **procedure** <identificador> [<parâmetros formais>] **;** <bloco>
8. <parâmetros formais> ::=
   (<seção de parâmetros formais> { **;** <seção de parâmetros formais>} **)**
9. <seção de parâmetros formais> ::=
   [**var**] <lista de identificadores> **:** <identificador>

OBS: Todos os identificadores devem ser declarados antes de serem utilizados. Isto implica a impossibilidade de se utilizar recursão múltipla entre subrotinas quando elas não estão declaradas uma dentro da outra. São considerados **identificadores pré-declarados** os identificadores de tipo simples **int** e **boolean**, os identificadores de procedimentos **read** e **write** (usados respectivamente para leitura e impressão) e os identificadores de constantes **true** e **false**.

Comandos

10. <comando composto ::= **begin** <comando> { **;** <comando>} **end**
11. <comando> ::=
    <atribuição>
    | <chamada de procedimento>
    | <comando composto>
    | <comando condicional 1>
    | <comando repetitivo 1>
12. <atribuição>::= <variável> **:=** <expressão>
13. <chamada de procedimento> ::= <identificador> [**(** <lista de expressões> **)**]
14. <comando condicional 1> ::= **if** <expressão> **then** <comando>
    [**else** <comando>]
15. <comando repetitivo 1> ::= **while** <expressão> **do** <comando>

OBS: Os comandos de entrada (read) e saída (write) estão incluídos nas chamadas de procedimentos. Supomos que existe um arquivo padrão de entrada contendo apenas números lidos pelo comando read(v1,v2, ...vn), em que v1, v2, ... vn são variáveis inteiras. Os pormenores sobre o formato do arquivo de entrada serão ignorados. Analogamente, o comando write(e1,e2, ...en) imprimirá os valores das expressões inteiras e1, e2, ...em num arquivo padrão de saída.

Expressões

16. <expressão> ::= <expressão simples> [<relação> <expressão simples>]
17. <relação> ::= **=** | **<>** | **<** | **<=** | **>=** | **>**
18. <expressão simples> ::= [**+** | **-**] <termo> {(**+** | **-** | **or**) <termo>}
19. <termo> ::= <fator> {(***** | **div** | **and**) <fator> }
20. <fator> ::=
    <variavel>
    | <número>
    | **(** <expressão> **)**
    | **not** <fator>
21. <variável> ::= <identificador> | <identificador> **[** <expressão> **]**
22. <lista de expressões> ::= <expressão> {**,** <expressão>}

OBS: As expressões podem ser **inteiros** ou **booleanas** e a distinção deverá ser feita pelo compilador. Note que as constantes **true** e **false** são consideradas identificadores prédeclarados.

Números e Identificadores

23. <número> ::= <dígito> {<dígito>}
24. <dígito> ::= **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**
25. <identificador> ::= <letra> {<letra> | <dígito>}
26. <letra> ::= **_| a** -**z** |**A** - **Z**

Obs 1: As produções 24 a 27 estão incluídas para tornar a gramática completa. Será mais conveniente, entretanto, tratar números e identificadores como símbolos terminais da gramática. Será conveniente também impor o número máximo de caracteres que pode entrar na formação de números e identificadores.

EBNF:

[α] = significa um item opcional

{α} = repetição da cadeia α zero ou mais vezes α | β =

α ou β devem ser escolhidos

Não terminais aparecem entre < e > e terminais aparecem em negrito.

Obs 2: o comentário de código na linguagem **COVIDLG** inicia-se por **{** e finaliza-se com **}** para múltiplas linhas e **//** no caso de uma única linha.

Entrega

a) Desenvolver os analisadores que respondam as necessidades da Gramática acima descrita;
b) Gerar representação intermediaria, a com instruções ao vosso critério (P-Codigo, Três Endereços, Assembly x86, etc.)
c) Elaborar um MER e DER compilador a ser desenvolvido;
d) Elaborar a diagrama de Classe o compilador a ser desenvolvido;

Deve ser entregue um relatório do trabalho em formato A4, escrito com letra do tipo **Times** ou semelhante, um tamanho de letra de **12pt**, à excepção de títulos, identificação e início de secções (14pt **carregado - bold** - é suficiente). **NÃO** deve ser entregue folha de rosto ou qualquer tipo de encadernação (e.g. argolas, cola, furos, calha, ...). As folhas que constituem o relatório devem ser **agrafadas** no canto superior esquerdo. Convidam-se os alunos a serem sucintos, sendo penalizados relatórios exageradamente longos.

A fonte utilizada deverá ser, preferencialmente, de tamanho 12 e nunca deverá ser de tamanho inferior a 10 (excepto dos diagramas a serem mostrados, desde que perfeitamente legível).

O relatório deverá conter o nome da disciplina, o curso, a identificação do grupo e o **esforço (em horas)** de cada um dos seus elementos. Deve ocupar no máximo ¼ do topo da 1ª página - exemplo:

Tópicos Avançados de Compilação, 2021 Trabalho Prático

Grupo 0

Fulano, Beltrano, etc…

Penalização

As características deste projecto requerem uma equipa composta obrigatoriamente por cinco elementos. **Excepcionalmente**, se o número de elementos inscritos não permitir que todos os grupos tenham cinco alunos, poderão ser considerados grupos com um número superior de alunos, isto é, seis elementos, mas nunca sete elementos.

Para além da entrega do relatório do trabalho em papel no formato A4, deverão entregar também o Código fonte do projecto e Executável para posterior testes, o relatório numa Pendrive até às 12:00 de **18/06/2021**. Também podem utilizar o seguinte email para entrega: nanitamo19@gmail.com até às 23:59 de **18/06/2021**. E para cada dia de atraso será descontado 20% da cotação total do trabalho.

ANEXOS

PROGRAMA VALIDO

```
program correto;
int a, b, c;
boolean d, e, f;

{Comentário correto}

procedure proc(var a1 : int);
int a, b, c;
boolean d, e, f;
begin
        a:=1;
        while (a>1)
        begin
                if (b>10)
                  b:=2;
                  a:=a-1
        end

end;

end.
```

PROGRAMA INVALIDO

```
program correto;
int a, b, c;
boolean d, e, f;

{Comentário correto}

procedure proc(var a1 : int);
```

```
int a, b, c;
boolean d, e, f;
begin
        a:=1;
        while (a>1)
        begin
                if (b>10)@
                   b:=2;
                   a:=a-1
        end

end;

end.
```