UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC
CURSO DE CIÊNCIA DA COMPUTAÇÃO

VICTOR FARIAS CASAGRANDE
GIOVANNI DOS SANTOS BERTI

COMPILADORES – MANUAL GRAMÁTICA

CRICIÚMA
2023

- **Estrutura do Programa**

Para começar algum tipo de aplicação com essa linguagem de programação devemos começar com a palavra chave bloco “program”, em seguida nome do programa que queremos criar, “;” para delimitar o fim da declaração do programa, seguido das declarações de variáveis e/ou procedimentos e do bloco lógico. Ao final do programa deve haver um ponto (“.”) para indicar o fim do arquivo.

Sintaxe:

```
program ident;
DECLARAÇÕES
BLOCO.
```

Exemplo:

```
program ProgExemplo;
var seuNome:string;
begin
    print{“Qual o seu nome?”}
    read(seuNome)
    print{“Olá “+seuNome+”, como vai você?”
end.
```

- **Tipos de Variáveis**

A linguagem possui três tipos de valores que podem ser usados e atribuídos. São eles:

integer: tipo usado para números inteiros, podem conter números de -2,147,483,648 a 2,147,483,647 (incluso).

Exemplos de integer: `3`

`11`

`49`

`9786`

real: tipo usado para números reais, que podem conter partes menores que um inteiro sendo esta parte não inteira separada com o símbolo “.”.

Exemplos de real: `3.33`

`11.13`

`49.97`

`9786.12345`

string: tipo usado para armazenar texto com caracteres alfanuméricos e símbolos. Strings podem ser delimitadas por aspas simples.

Exemplos de string: `‘esta é um valor possível para uma string’`

- **Comandos de entrada e saída de dados**

O comando para imprimir informações na tela é chamado a partir da palavra reservada “print” seguida do que deve ser impresso entre chaves (“{}”) .

Para aguardar e ler informações digitadas no terminal é usado o comando read seguido da variável que deve receber o que foi lido entre parênteses(“()”)

Exemplo:

```
program ProgExemplo;
var seuNome:string;
begin
    print{“Qual o seu nome?”}
    read(seuNome)
    print{“Olá “+seuNome+”, como vai você?”
end.
```

Execução do exemplo:

```
Qual o seu nome?
Victor
Olá Victor, como vai você?
```

- **Comentários**

Os comentários são utilizados para explicar o código, e são indicados por duas barras (“//”) para iniciar e para terminar. Tudo que for digitado entre as duplas barras é considerado um comentário e será ignorado pelo compilador.

É possível fazer um comentário de apenas uma linha usando três barras, tudo que for digitado até o final da linha é considerado um comentário.

Exemplo:

```
program ProgExemplo;
var seuNome:string;
begin
    // Isto é um comentário //
    print{“Qual o seu nome?”}
    read(seuNome)
    // comentários só acabam quando
    é encontrado outra barra dupla //
    print{“Olá “+seuNome+”, como vai você?”
end.
```

- **Criação de Procedimentos (Funções)**

É possível definir procedimentos na parte de declaração do código que podem ser usados várias vezes durante o programa possibilitando reuso de código e evitando redundância e inconsistência no programa. Para tal indicamos o início da declaração com procedure seguida de seu identificador, e seus parâmetros entre parênteses (“()”).

Exemplo:

```
program ProgExemplo;
var seuNome:string;
var valorSoma : integer;
procedure soma( valorA, valorB : integer);
    begin
        valorSoma := valorA + valorB;
        print{valorSoma};
    end;
begin
    soma(1,2);
    soma(7,8);
end.
```

- **Comandos Matemáticos**

A linguagem possui símbolos reservados para realizar as 4 operações matemáticas básicas, como indicado abaixo.
Adição: +
Subtração: -
Multiplicação: *
Divisão: /

Exemplo:

```
program ProgExemplo;
var numA, numB, numC : integer;
begin
    numA := 2;
    numB := 3;
    numC := numB - numA;
end.
```

- **Comandos de Comparação e Atribuição**

Como já visto nos exemplos anteriores usamos o símbolo “:=” para realizar atribuições a uma variável. Abaixo estão indicados também os símbolos referentes a cada comparador relacional usado nas expressões relacionais da linguagem.

Igual: =
Diferente: <>
Maior que: >
Menor que: <
Maior ou igual que: >=
Menor ou igual que: <=

- **Estruturas de Controle:**
    - **if/else**

Nesta linguagem é possível realizar ações condicionais usando a estrutura de controle if/else da seguinte forma:

Sintaxe:

```
if [EXPRESSÃO] then begin
    //comandos executados caso a [EXPRESSÃO] seja resolvida para
    “verdadeiro”//
end
else begin
    //comandos executados caso a [EXPRESSÃO] seja resolvida para
    “falso”//
end;
```

Exemplo:

```
if (numA >= numB) then begin
    print{numA-numB};
end
else begin
    print{numB-numA};
end;
```

    - **while/do**

Esse comando é uma estrutura de repetição, onde existe uma condição, e enquanto ela for verdadeira, deverá continuar repetindo a operação até que a expressão se torne falsa.

Sintaxe:

```
while [EXPRESSÃO RELACIONAL] do begin
    //código executado em cada iteração do while//
end;
```

    - **for**

Este também é uma estrutura de repetição, porém o for recebe um identificador que tem uma expressão atribuída a ele e deve ser iterado até que outra expressão se torne falsa de forma similar ao comando while/do.

Sintaxe:

```
for nomeForm := EXPRESSÃO to EXPRESSÃO do begin
    //código executado em cada iteração do for//
end;
```

- **Caracteres especiais**

A linguagem possui caracteres com funções especiais, além dos já descritos anteriormente. São estes

Ponto e Vìrgula (“;”): Usado ao final de cada comando como delimitador do fim do comando

Dois Pontos (“:”): usado para indicar que o tipo descrito a direita se aplica ao identificador de variável escrito a esquerda do símbolo

Parênteses (“()”): Podem ser usados em expressões para indicar precedências matemáticas e lógicas. Também é usado para delimitar as expressões usadas pelas estruturas de controle.

- **Regras Léxicas**

integer: podem conter números de 0 a 2,147,483,647 (incluso).

real: podem conter números de 0.00 a 2,147,483,647.99 (incluso)
aceitando até duas casas depois do separador decimal que é o ponto (“.”)

string: são delimitadas por aspas simples (“ ‘ “) no começo e no final.

literal: são delimitados por aspas duplas (“ “ “) no começo e no final.

- **Erros Léxicos**

Qualquer sequência de caracteres que não se encaixe a um token previsto nas regras acima.

Variáveis não podem ser identificadas pelas palavras reservadas (“real”, “integer”, “for”, “do”, “if”, “while”, “print”) nem pelos caracteres especiais("{", "(", "+", "-", "*", "/", ";", ".", ",", "?")

- **Erros Semânticos**

  Variáveis e não podem ter o mesmo nome e nível;

  Procedures não podem ter o mesmo nome e nível;

  Variáveis e Procedures não podem ter o mesmo nome;

  Variáveis e Procedures precisam ser declaradas precisa;