

# Apontamentos sobre Pascal

---

## Estrutura de um programa Pascal

```
PROGRAM ProgramName (FileList);

CONST
  (* Constant declarations *)

TYPE
  (* Type declarations *)

VAR
  (* Variable declarations *)

  (* Subprogram definitions *)

BEGIN
  (* Executable statements *)
END.
```

A linha PROGRAM é opcional em Pascal.

Comentários em Pascal podem ser do tipo `(* *)` ou do tipo `{ }`

## Exemplos de Programa Pascal

```
program Hello;
begin
  writeln ('Hello, world.');
  readln;
end.
```

Análise deste código:

- Keywords: program, begin, end, '(', ')', função, string, id
- função: writeln, readln
- string: 'Hello, world.'
- id: Hello

## Identificadores

Identificadores são nomes que permitem referir valores guardados. Além disso, todos os programas devem ser identificados por identificadores. Regras para os identificadores:

- Deve começar por a-z ou A-Z ou um underscore(\_)

- Pode ser seguido de 0 ou mais letras (a..Z), dígitos(0..9), ou underscores(\_)
- Não pode ser igual às palavras-chave: begin, end, for, while, if, etc..
- Não pode conter nenhum dos caracteres especiais: `:";'<>? , . / | \` ou espaço)

## Constantes

As constantes são referidas pelos identificadores, e podemos dar logo um valor no início do programa. O valor de uma constante não pode ser alterado. As constantes podem ser: Scalars, Records ou Arrays.

### Scalars

Identificador único com um valor único.

```
const
  Identifier1 = value;
  Identifier2 = value;
  Identifier3 = value;
```

Também é permitido algo do tipo:

```
const
  a : real = 12;
```

Que devolve o valor 12.0 em vez do inteiro 12.

### Records

Records são criados ao criar um record type com um ou mais campos. A forma geral é: `const identifier: record_type = ( field_values ); identifier ...` onde:

- identifier é o nome do record
- é o nome do record type usado para descrever esta constante record
- field\_values é uma lista de valores

Exemplo:

```
type
  complex = record
    R,I: real;
  end;
const
  Pi = 3.14159267;
  C1: Complex = (R:3; I:1783.5);
  C2: Complex = (R:9.6; I:1.62);
  C3: Complex = (R:17; I:115);
  C4: Complex = (R:1.9e56; I:72.43);
  C5: Complex = (R:22.1; I:Pi);
```

## Array

**\*\* Array de uma dimensão \*\*** O formato geral é: `const identifier: array[low_value .. high_value] of type = ( values ); identifier ...`

Onde:

- `identifier` é o nome
- `low_value` é o valor mais baixo do array
- `high_value` é o maior
- `type` é o tipo dos elementos que estão no array
- `values` é uma lista com os valores em que cada item está separado por uma vírgula.

Exemplo:

```
const
  Alphabet: array [1..26] of char =
    ('A','B','C','D','E','F','G','H','I',
     'J','K','L','M','N','O','P','Q','R',
     'S','T','U','V','W','X','Y','Z'  );
```

Um array de uma dimensão pode consistir em records. Exemplo:

```
type
  complex = record
    R,I: real;
  end;
const
  Pi = 3.14159267;
  C2: complex = ( R: 96; I:1.62);
  C_Low = 1; C_High = 5;
  C: array [C_Low .. C_High] of complex = (
    (R:3; I:1783.5),
    (R:96; I:1.62),
    (R:17; I:115),
    (R:1.9e56; I:72.43),
    (R:102.1; I:Pi)
  );
```

É possível também fazer um array de **duas e três** dimensões. Exemplos:

- **Duas dimensões**

```
const
  DemoArray: array [1..2, 1..3] of integer = (
    (11,12,13),
```

```
(21,22,23)
);
```

- **Três Dimensões**

```
const
  Demo3: array [1..2, 1..3, 1..4] of integer =
  (
    (
      (111,112,113,114),
      (121,122,123,124),
      (131,132,133,134)
    ),
    (
      (211,212,213,214),
      (221,222,223,224),
      (231,232,233,234)
    )
  );
```

## Variáveis e Data Types

Variáveis são similares às constantes, mas o valor delas pode se ir alterando ao longo da execução do programa.

```
var
  IdentifierList1 : DataType1;
  IdentifierList2 : DataType2;
  IdentifierList3 : DataType3;
  ...
```

## Operações e atribuição de valores

Depois de definirmos uma variável é possível atribuir-lhe um valor ou uma expressão. Ex.: `some_real := 37573.5 * 37593 + 385.8 / 367.1;`

- As operações aritméticas são feitas com os operadores "normais" das outras linguagens de programação: +, -, \*, /, div, mod

## Funções Standard do Pascal

- abs - absolute value
- sin, cos, tan, arcsin, etc.
- exp, ln, sqr, sqrt
- round - arredondar
- trunc - arredondar para baixo