Além dos vetores (e matrizes) podemos ter outros tipos de dados no Pascal.

Outros tipos de dados predefinidos podem ser organizados em tipos de dados complexos, denominados *tipos estruturados*.

A linguagem Pascal oferece, basicamente, quatro destes tipos:

1. Enumerações ;
2. Ponteiros ;
3. Registros ;
4. Vetores ;

- Tipos de dados enumerados são utilizados para denotar um conjunto de constantes.

Declaração de enumerações

```
Var nomeEnumeracao : ( identificador, ...., identificador ) ;
```

Onde identificador denota um identificador válido na linguagem Pascal.

Exemplo

```
Program Pzim ;
var diasSemana : (domingo, segunda, terca, quarta, quinta, sexta, sabado) ;
Begin
  writeln( 'Depois de segunda vem quinta? ' , succ(segunda) = quinta  );
  writeln( 'Depois de segunda vem terca? '  , succ(segunda) = terca  );
  writeln( 'Antes de quinta vem quarta? '   , pred(quinta) = quarta  );
  writeln( 'Antes de quinta vem segunda? '  , pred(quinta) = segunda  );
End.
```

Exemplo

```
Program Pzim ;
Type diaSemana = ( domingo, segunda, terca, quarta, quinta, sexta,
sabado ) ;
Var dia : diaSemana ;
Begin
   for dia := domingo to sabado do
   begin
     case ( dia ) of
        domingo: writeln( 'O dia é domingo' );
        segunda: writeln( 'O dia é segunda' ) ;
        terca  : writeln( 'O dia é terca' ) ;
        quarta : writeln( 'O dia é quarta' ) ;
        quinta : writeln( 'O dia é quinta' ) ;
        sexta  : writeln( 'O dia é sexta' ) ;
        sabado : writeln( 'O dia é sabado' ) ;
      end;
   end;
   readkey;
End.
```

Ponteiros são variáveis que podem armazenar o endereço de uma outra variável.

Declaração de ponteiros

```
Var nomePonteiro : ^tipoDados ;
```

O simbolo ^ deve ser lido como o *ponteiro para...*

Na declaração acima temos que *nomePonteiro* é um ponteiro para variáveis do tipo *tipoDados*.

Exemplo

```
Var ponteiro: ^integer ;
```

Exemplo

```
Type TAluno = Record
                nome: String ;
                matricula: String ;
              End ;

Var ponteiroAluno: ^TAluno ;
```

Operações sobre ponteiros

- Guardar no ponteiro endereço de uma variável:

```
ponteiro := @variável ;
```

- Guardar no ponteiro o endereço armazenado em um outro ponteiro:

```
ponteiro := outroponteiro ;
```

- Dizer que o ponteiro não guarda nenhum endereço:

```
ponteiro := nil ;
```

- Referenciar o dado apontado pelo ponteiro (o elemento que tem o tipo de dados definido pelo ponteiro, e que está no endereço de memória que o ponteiro armazena):

```
ponteiro^
```

Exemplo

```
Program Ponteiros ;
Var  a: integer;
   p: ^integer;
Begin
    a := 8 ;    // Guardamos o valor 8 em a
    p := nil;   // O ponteiro não guarda nenhum endereço
    writeln( 'Valor armazenado em a: ' , a );
   // Guardamos no ponteiro o endereço da variável a
   p := @a ;
   writeln( 'Valor apontado por p: ' , p^ );
   // Esse comando é equivalente a "a:= 2 * a ;" , pois p
   // aponta para o endereço de a
   a:= 2 * p^ ;
   writeln( 'O valor de a agora: ' , a );     // Imprime 16
   writeln( 'Valor apontado por p: ' , p^ );  // Imprime 16
   readln ;
End.
```

Alocação Dinâmica de Memória

É possível alocar, dinamicamente, espaço na memória para um ponteiro. A quantidade de memória é determinada pelo tipo do ponteiro.

Sintaxe

```
new( ponteiro ) ;
```

Deve-se tomar cuidado para que a memória alocada com um *new* seja liberada antes do programa terminar.

Sintaxe

```
dispose( ponteiro ) ;
```

Exemplo

```
Program AlocacaoDinamica ;
Var p: ^integer;
    v : integer ;
Begin
  new( p );    // Aloca espaço para armazenar um inteiro
  p^ := 10 ;   // Guarda um inteiro na posição apontada por p

  writeln( 'Valor armazenado na posicao de memoria: ', p^ );

  v:= p^ ;     //Guardamos em v o valor apontado por p

  writeln( 'Valor armazenado em v: ', v );

  dispose( p );  // Liberamos a memoria associada a p
  readln ;
End.
```

Exemplo

```
// ------------------------------------------------------------
// Este programa mostra ilustra a utilização de listas lineares
// usando ponteiros.
//
// Problema. Construir uma lista linear e imprimir seus dados.
// ------------------------------------------------------------

Program PercorrendoLista ;

// Definição de um tipo para representar um nó da lista
type TNo = record
              dado : integer ;  // Dado armazenado pelo nó
              prox : ^TNo ; // Ponteiro p/ próximo nó
           end ;

Var pinicio: ^TNo;    // Guarda endereço 1° nó da lista
    p1: ^TNo;         // Auxiliar. Guarda endereço de um nó
    resposta : char ; // Auxiliar. Controla repetição.

Begin
  pinicio := nil ;

  // Repetição que define os nós da lista
  repeat
     new( p1 );
     write( 'Entre com novo dado: ' );
     readln( p1^.dado ) ;
     p1^.prox := pinicio ;
     pinicio := p1 ;
     write( 'Novo dado(S/N)?' );
     readln( resposta );
     resposta := upcase( resposta );
  Until resposta = 'N' ;

  // Percorrer a lista, imprimindo seus elementos
  p1 := pinicio ;
  while( p1 <> nil ) do
   Begin
      writeln( 'Achei: ' , p1^.dado );
      p1 := p1^.prox ;
   End;

  // Percorrer a lista, desalocando memória para os elementos
  while( pinicio <> nil ) do
  Begin
     p1 := pinicio ;
     pinicio := pinicio^.prox ;
     dispose( p1 );
  End;

  readln ;
End.
```

Um *registro* é um tipo composto por um conjunto formado por dados de tipos diferentes, onde cada um dados é definido como sendo um *campo.*

Um tipo *registro* é definido através da palavra reservada **record**, seguida por uma série de declaração de *campos*. A palavra reservada **end** seguida de um ponto e vírgula encerra a definição de um registro.

A sintaxe genérica para definição de *registros* segue o seguinte formato:

Sintaxe

```
Record
  Identificador de campo  : tipo;
  Identificador de campo  : tipo;

  Identificador de campo  : tipo;
End;
```

Exemplo. Declaração de um registro simples:

```
var dados : Record
              numero : integer;
              caracter : char;
              preenchido : boolean;
            End;
```

Exemplo. Declaração de um registro contendo registros aninhados:

```
var umRegistro : Record
                   numero : integer ;
                   dado : Record
                            caractere : char ;
                          End;
                   preenchido: boolean ;
                 End;
```

A referência a um campo de um registro é feita através do nome da variável do tipo registro seguida por um ponto e pelo nome do campo, como por exemplo,

```
   Read(umRegistro.numero);
Writeln(umRegistro.numero);
```

A definição de um novo tipo é feita na seção de definição de tipos, contida na seção de definição e declaração de dados.

O início da seção de definição de tipos é indicada através da palavra reservada **Type**. A palavra reservada **Type** deve aparecer uma única vez dentro da seção de definição e declaração de dados.

Sintaxe

```
type
  nomeTipo = tipoDefinido ;
```

onde tipoDefinido é um dos tipos estruturados *vetor, registro, ponteiro* ou outro tipo de dados simples.

Exemplo

```
type
  intList = array[1..100] of integer ;
  matrix = array[0..9, 0..9] of real ;
  pInt = ^integer ;
```