Capítulo 4 – Estruturas de Dados

Embora os tipos primitivos sejam suficientes para lidar com informações simples como a idade de uma pessoa (**inteiro**), o salário de um funcionário (**real**), o nome de um cliente (**caractere**) e até mesmo para saber se uma conta foi ou não paga (**lógico**), algumas informações requerem tipos mais complexos, como uma data, um vetor, uma matriz, etc. Para esses tipos de dados complexos, utilizamos as estruturas de dados.

Uma estrutura de dados é uma composição formada por tipos primitivos e possivelmente outras estruturas de dados. Sua principal finalidade é agrupar dados relacionados no intuito de facilitar na construção dos algoritmos.

Por exemplo, ao invés de lidarmos diretamente com uma variável para o dia, uma para o mês e outra para o ano toda vez que uma data se faz necessária, pode-se definir uma estrutura de dados do tipo **data** e utilizar essa estrutura sempre que necessário.

```
tipo data = registro
              inteiro: dia, mês, ano;
            fim-registro;

data: dataNascimento, dataAtual, dataCompra;
```

Vetores

Um vetor é uma coleção unidimensional composta por dados de um mesmo tipo. Os elementos de um vetor são acessados por meio de seu índice.

Sintaxe: **tipo** novo-tipo = **vetor** [i..j] **de** tipo-existente;

i: índice inicial
j: índice final
i ≤ j

Exemplo…

```
início
  tipo vmeses = vetor [1..12] de caractere;
  vmeses: meses;
  inteiro: mês;
  meses[1] ← "Janeiro";
  meses[2] ← "Fevereiro";
  meses[3] ← "Março";
  meses[4] ← "Abril";
  meses[5] ← "Maio";
  meses[6] ← "Junho";
  meses[7] ← "Julho";
  meses[8] ← "Agosto";
  meses[9] ← "Setembro";
  meses[10] ← "Outubro";
  meses[11] ← "Novembro";
  meses[12] ← "Dezembro";
  leia(mês);
  se (mês >= 1 e mês <= 12) então
    escreva(meses[mês]);
  fim-se;
fim.
```

Exemplo: Algoritmo para calcular a média aritmética das notas de uma turma de dez alunos e imprimir a média e a quantidade de notas acima da média.

```
início
  real: n1, n2, n3, n4, n5, n6, n7, n8, n9, n10; // as notas da turma
  real: média;
  inteiro: notasAcimaMédia;
  leia(n1, n2, n3, n4, n5, n6, n7, n8, n9, n10);
  média ← (n1 + n2 + n3 + n4 + n5 + n6 + n7 + n8 + n9 + n10) / 10;
  notasAcimaMédia ← 0;
  se (n1 > média) então
    notasAcimaMédia ← notasAcimaMédia + 1;
  fim-se;
  se (n2 > média) então
    notasAcimaMédia ← notasAcimaMédia + 1;
  fim-se;
  se (n3 > média) então
    notasAcimaMédia ← notasAcimaMédia + 1;
  fim-se;
  se (n4 > média) então
    notasAcimaMédia ← notasAcimaMédia + 1;
  fim-se;
  se (n5 > média) então
    notasAcimaMédia ← notasAcimaMédia + 1;
  fim-se;
  se (n6 > média) então
    notasAcimaMédia ← notasAcimaMédia + 1;
  fim-se;
  se (n7 > média) então
    notasAcimaMédia ← notasAcimaMédia + 1;
  fim-se;
  se (n8 > média) então
    notasAcimaMédia ← notasAcimaMédia + 1;
  fim-se;
  se (n9 > média) então
    notasAcimaMédia ← notasAcimaMédia + 1;
  fim-se;
  se (n10 > média) então
    notasAcimaMédia ← notasAcimaMédia + 1;
  fim-se;
  escreva(média);
  escreva(notasAcimaMédia);
fim.
```

Utilizando vetores…

```
início
  tipo vnotas = vetor [1..10] de real;
  vnotas: notas;
  real: soma, média;
  inteiro: notasAcimaMédia, i;
  soma ← 0;
  notasAcimaMédia ← 0;
  para i de 1 até 10 passo 1 faça
    leia(notas[i]);
    soma ← soma + notas[i];
  fim-para;
  média ← soma / 10;
  para i de 1 até 10 passo 1 faça
    se (notas[i] > média) então
      notasAcimaMédia ← notasAcimaMédia + 1;
    fim-se;
  fim-para;
  escreva(média);
  escreva(notasAcimaMédia);
fim.
```

Desafio "Selection Sort"! Escreva um algoritmo para colocar em ordem crescente dez números digitados pelo usuário. Utilize a seguinte estratégia: Encontre o menor dos números e coloque este número na primeira posição. Certifique-se de que o número que estava na primeira posição troque de lugar com aquele que agora está ocupando sua posição. Repita esses passos considerando-se agora a segunda posição, depois a terceira e assim por diante.

```
início
  tipo vnúmeros = vetor [1..10] de inteiro;
  vnúmeros: números;
  inteiro: i, j, k, selecionado;
  para i de 1 até 10 passo 1 faça
    leia(números[i]);
  fim-para;
  para i de 1 até 9 passo 1 faça
    k ← i;
    selecionado ← números[k];
    para j de i + 1 até 10 passo 1 faça
      se (números[j] < selecionado) então
        início
          k ← j;
          selecionado ← números[k];
        fim;
      fim-se;
    fim-para;
    números[k] ← números[i];
    números[i] ← selecionado;
  fim-para;
  para i de 1 até 10 passo 1 faça
    escreva(números[i]);
  fim-para;
fim.
```

Desafio "Bubble Sort"! Escreva um algoritmo para colocar em ordem crescente dez números digitados pelo usuário. Utilize a seguinte estratégia: Compare sempre pares de elementos adjacentes, permutando-os quando estiverem fora de ordem.

```
início
  tipo vnúmeros = vetor [1..10] de inteiro;
  vnúmeros: números;
  inteiro: i, aux;
  lógico: ordenado;
  para i de 1 até 10 passo 1 faça
    leia(números[i]);
  fim-para;
  ordenado ← F;
  enquanto (não ordenado) faça
    ordenado ← V;
    para i de 1 até 9 passo 1 faça
      se (números[i] > números[i + 1]) então
        início
          aux ← números[i];
          números[i] ← números[i + 1];
          números[i + 1] ← aux;
          ordenado ← F;
        fim;
      fim-se;
    fim-para;
  fim-enquanto;
  para i de 1 até 10 passo 1 faça
    escreva(números[i]);
  fim-para;
fim.
```

Matrizes

Uma matriz é uma coleção multidimensional composta por dados de um mesmo tipo. Os elementos de uma matriz são acessados por meio de seus índices.

Sintaxe: **tipo** novo-tipo = **matriz** [$i_1..j_1$, $i_2..j_2$, …] **de** tipo-existente;

i_1, i_2, …: índice inicial da respectiva dimensão da matriz
j_1, j_2, …: índice final da respectiva dimensão da matriz
$i_k \leq j_k$

Exemplo…

```
início
  tipo matriz2x2 = matriz [1..2, 1..2] de inteiro;
  matriz2x2: m;
  m[1, 1] ← 1;
  m[1, 2] ← 2;
  m[2, 1] ← 3;
  m[2, 2] ← 4;
  // Processamento dos dados da matriz m
fim.
```

Registros

Um registro é uma estrutura de dados que agrupa campos não necessariamente do mesmo tipo. Os campos (variáveis) de um registro são acessados por meio de seus respectivos nomes (identificadores).

Sintaxe:

```
tipo novo-tipo = registro
                   tipo-existente: campo1;
                   tipo-existente: campo2, campo3, …;
                   …
                 fim-registro;
```

Exemplo…

```
início
  tipo Data = registro
                inteiro: dia, mês, ano;
              fim-registro;
  tipo Notas = vetor [1..4] de real;
  tipo Aluno = registro
                 caractere: ra, nome;
                 Data: dataNascimento;
                 Notas: português, matemática;
               fim-registro;
  Aluno: aluno;
  real: somaPortuguês, somaMatemática, médiaPortuguês, médiaMatemática;
  inteiro: i;
  leia(aluno);
  somaPortuguês ← 0;
  somaMatemática ← 0;
  para i de 1 até 4 faça
    somaPortuguês ← somaPortuguês + aluno.português[i];
    somaMatemática ← somaMatemática + aluno.matemática[i];
  fim-para;
  médiaPortuguês ← somaPortuguês / 4;
  médiaMatemática ← somaMatemática / 4;
  escreva(aluno);
  se (médiaPortuguês > médiaMatemática) então
    escreva(aluno.nome, " é melhor em português.");
  senão
    se (médiaMatemática > médiaPortuguês) então
      escreva(aluno.nome, " é melhor em matemática.");
    fim-se;
  fim-se;
fim.
```

Tarefa: Ler ~~capítulo 5~~ capítulo 7 do livro para a próxima aula