```
{
   projeto numbers-2026
   Projeto de estudo de algoritmos
   Este obra está licenciado sob Licença Creative Commons Atribuição-CompartilhaIgual 4.0 Internacional.
   Veja os termos desta licença na página https://creativecommons.org/licenses/by-sa/4.0/deed.pt-br
   Copyleft 2026 - Evandro Guglielmeli
   versão 0.0.0
}
program numbers;

type placa = array [1..4] of integer;

  exprn = record
            texto : string;
            valor : real;
          end;

  a4exp = array [1..4] of exprn;

const MAXDIG = 4;
  DEBUG = FALSE;

var a4num : array [1..24] of placa; //a4num : array [1..24,1..4] of integer
  control,
  linha : integer;

procedure mistura(input : placa; index : integer);
// input tem a série de digitos da placa
// index indica a posição que inicia o arranjo
var output : placa;
  p, i : integer;

begin
  if index=MAXDIG then
  begin
    a4num[linha] := input;
    linha := linha + 1;
  end
  else
  begin
    output := input;
    for p := index to MAXDIG do
    begin
      output[index] := input[p];
      for i := index to p - 1 do
        output[i + 1] := input[i];
      for i := p + 1 to MAXDIG do
        output[i] := input[i];
      mistura(output,index + 1);
    end;
  end;
end;

{ procedimentos de apoio }

{ procedimento para mostrar uma expr }
procedure showexpr(input : exprn);
begin
  writeln('"', input.texto, '" , ',input.valor);
end;

// o primeiro fonte vem até aqui
// aqui inicia a inclusão do segundo fonte

//
```

```
66 // procedimento para exibir mensagens de erro
67 //
68 procedure sherror(code  : integer);
69 begin
70   if (code>0) then
71   begin
72     if DEBUG then
73       case code of
74         201 :
75           writeln('A entrada é inválida. Apenas dígitos, de 0 a 9, são aceitos!');
76         202 :
77           writeln('Divisão por 0!');
78         203 :
79           writeln('Não posso calcular potência de valor igual ou menor que 0!');
80         204 :
81           writeln('Sequência numérica mal formatada!');   // erro introduzido pelo teste
                3x1
82       else
83         writeln('Código de controle: ',code);  // apenas para rastreamento - sair
84       end;
85     control := 0; // limpa o código de controle / erro
86   end;
87 end;
88
89
90 // funções de processamento dos dados fornecidos
91 // recebem um exprn e retornam outro
92 // ajustam o controle de erro, se for o caso
93
94 //
95 // funções "una" : recebe uma exprn e retorna uma exprn
96 //
97
98 // valn retorna um exprn de um dígito, erro se não for um número
99 //   o dígito entra pelo campo texto do parâmetro input
100 function valn(input : exprn) : exprn;
101 var v , w : integer;
102 begin
103   if (input.texto >= '0') and (input.texto <= '9') then
104   begin
105     val(input.texto,v,w);
106     if (w=0) then
107       input.valor := v
108     else
109       control := w;
110   end
111   else
112     control := 201;  // ver significado em sherror
113   valn := input;
114 end;
115
116
117 //
118 // funções "par" : processam duas exprns fornecidas, retornando uma exprn com o resultado
119 // funções de valor direto: dezena
120 // funções de aritmética comum:  nb_add, nb_sub, nb_mul, nb_div
121 // funções de aritmética de inteiros: nb_idiv, nb_mod
122 // funções exponenciais: nb_pow, nb_wop, nb_exp
123 // funções estatisticas: nb_mid, nb_max, nb_min
124 //
125
126
127 {   Todas as funções "par" recebem um par de exprn e retornam uma exprn
128     cada uma dessas funções executa apenas uma operação, dezena, por exemplo,
129     junta as duas exprns que recebeu e retorna a exprn que representa a
130     dezena formada pelos dois dígitos recebidos.
131
```

```
132     Em geral uma expressão "par" nunca vai gerar um erro, por que ela já recebe
133     os dígitos testados previamente; as exceções são:
134     divisão - não se pode dividir um número por 0
135     divisão inteira - não se pode dividir um número por 0
136     resto da divisão inteira - não se pode dividir um número por 0
137     ...
138
139     Nestes casos (das exceções) o algoritmo precisa primeiro verificar a exceção e,
140     se ela não ocorrer, aí a operação pode ser executada.
141
142     Quando ocorrer uma exceção, o algoritmo deve sinalizar o código de erro para que o
        procedimento sherror mostre a mensagem de erro correspondente.
143 }
144
145 // dezena retorna o valor da dezena formada pelos dois dígitos juntos
146 // recebe duas exprns de 1 dígito, concatena e avalia o valor resultante
147 function dezena(inp1, inp2 : exprn) : exprn;
148 var v , w : integer;
149   s : string;
150 begin
151   s := inp1.texto + inp2.texto;
152   val(s, v, w);
153   if w<>0 then
154   begin
155     inp1.texto := 'erro';  // padronização do tratamento de erro
156     inp1.valor := -996;
157     control := 204;         // código para sherror()
158   end
159   else
160   begin
161     inp1.texto := s;
162     inp1.valor := v;
163   end;
164   dezena := inp1;
165 end;
166
167 //
168 // encaminhamento da avaliação de um par de exprns
169 //
170 function eval2e(i1, i2 : exprn; var ctl : integer) : exprn;
171 begin
172   inc(ctl);
173   case ctl of // seleciona qual função "par" deve ser avaliada
174     1 : eval2e := dezena(i1, i2);
175     else
176       ctl := 0; // não restam mais funções de "par"
177   end;
178 end;
179
180
181 //
182 // processamento de igualdades entre pares
183 // exp1 op exp2 = exp3 op exp4
184 // versão 0.0.0
185 //
186 procedure test2by2(inp : a4exp);
187 var eleft, eright : exprn;
188   cleft, cright : integer;
189
190 begin
191 // lado esquerdo
192   cleft := 0;
193   repeat
194     eleft := eval2e(inp[1], inp[2], cleft);
195 // lado direito
196     cright := 0;
197     repeat
```

```
198       eright := eval2e(inp[3], inp[4], cright);
199 // quando os lados esquerdo e direito são válidos e há igualdade entre eles
200       if (cleft>0) and (cright>0) and (eleft.valor = eright.valor) then
201         writeln('Arranjo ',linha:2,' => ',eleft.texto,' = ',eright.texto)
202       else
203         if control>0 then
204           sherror(control);
205     until cright = 0;
206   until cleft = 0;
207 end;
208 // aqui termina a inclusão da segunda versão
209 //
210
211 // o programa principal
212 // versão 0.0.0
213 var entrada : string;
214   input, prova : a4exp;
215   p, i : integer;
216
217 begin
218   { inicializando a M.V. do algoritmo }
219   linha := 1;
220   for i := 1 to MAXDIG do
221     a4num[linha,i] := i;
222   mistura(a4num[linha],1);
223
224   // início da inclusão no segundo fonte
225   // iniciar a interação do algoritmo
226   repeat
227     write('Entre o número da placa, na forma de uma sequência de quatro dígitos: ');
228     readln(entrada);
229     if entrada='fim' then
230       break;
231
232     for i := 1 to MAXDIG do
233     begin
234       input[i].texto := entrada[i];
235       input[i] := valn(input[i]);
236       if DEBUG then
237         showexpr(input[i]);
238     end;
239     if control = 0 then
240     begin
241       linha := 1;
242       while linha < 19 do
243       begin
244         for i := 1 to MAXDIG do
245           prova[i] := input[a4num[linha][i]];
246 {   teste de igualdade entre 2 e 2 dígitos }
247         test2by2(prova);
248         inc(linha);
249         if linha>6 then inc(linha);
250       end;
251     end;
252
253
254
255     if control<>0 then
256       sherror(control)
257     else
258       writeln;
259   until FALSE;
260   // fim da inclusão no segundo fonte
261
262 end.
263
```