```
{
   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 0.2.0
}
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;

//
// procedimento para exibir mensagens de erro
//
procedure sherror(code  : integer);
```

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

```
132     divisão inteira - não se pode dividir um número por 0
133     resto da divisão inteira - não se pode dividir um número por 0
134     ...
135
136     Nestes casos (das exceções) o algoritmo precisa primeiro verificar a exceção e,
137     se ela não ocorrer, aí a operação pode ser executada.
138
139     Quando ocorrer uma exceção, o algoritmo deve sinalizar o código de erro para que o
        procedimento sherror mostre a mensagem de erro correspondente.
140 }
141
142 // dezena retorna o valor da dezena formada pelos dois dígitos juntos
143 // recebe duas exprns de 1 dígito, concatena e avalia o valor resultante
144 function dezena(inp1, inp2 : exprn) : exprn;
145 var v , w : integer;
146   s : string;
147 begin
148   s := inp1.texto + inp2.texto;
149   val(s, v, w);
150   if w<>0 then
151   begin
152     inp1.texto := 'erro';  // padronização do tratamento de erro
153     inp1.valor := -996;
154     control := 204;         // código para sherror()
155   end
156   else
157   begin
158     inp1.texto := s;
159     inp1.valor := v;
160   end;
161   dezena := inp1;
162 end;
163
164 //
165 // encaminhamento da avaliação de um par de exprns
166 //
167 function eval2e(i1, i2 : exprn; var ctl : integer) : exprn;
168 begin
169   inc(ctl);
170   case ctl of // seleciona qual função "par" deve ser avaliada
171     1 : eval2e := dezena(i1, i2);
172     else
173       ctl := 0; // não restam mais funções de "par"
174   end;
175 end;
176
177
178 //
179 // processamento de igualdades entre pares
180 // exp1 op exp2 = exp3 op exp4
181 // versão 0.0.0
182 //
183 procedure test2by2(inp : a4exp);
184 var eleft, eright : exprn;
185   cleft, cright : integer;
186
187 begin
188 // lado esquerdo
189   cleft := 0;
190   repeat
191     eleft := eval2e(inp[1], inp[2], cleft);
192 // lado direito
193     cright := 0;
194     repeat
195       eright := eval2e(inp[3], inp[4], cright);
196 // quando os lados esquerdo e direito são válidos e há igualdade entre eles
197       if (cleft>0) and (cright>0) and (eleft.valor = eright.valor) then
```

```
        writeln('Permutação ',linha:2,' => ',eleft.texto,' = ',eright.texto)
      else
        if control>0 then
          sherror(control);
    until cright = 0;
  until cleft = 0;
end;


// o programa principal
// versão 0.2.0
var entrada : string;
  input, prova : a4exp;
  p, i : integer;

begin
  { inicializando a M.V. do algoritmo }
  linha := 1;
  for i := 1 to MAXDIG do
    a4num[linha,i] := i;
  mistura(a4num[linha],1);

  // iniciar a interação do algoritmo
  repeat
    write('Entre o número da placa, na forma de uma sequência de quatro dígitos: ');
    readln(entrada);
    if entrada='fim' then
      break;

    for i := 1 to MAXDIG do
    begin
      input[i].texto := entrada[i];
      input[i] := valn(input[i]);
      if DEBUG then
        showexpr(input[i]);
    end;
    if control = 0 then
    begin
      linha := 1;
      while linha < 19 do
      begin
        for i := 1 to MAXDIG do
          prova[i] := input[a4num[linha][i]];
{   teste de igualdade entre 2 e 2 dígitos }
        test2by2(prova);
        inc(linha);
        if linha>6 then inc(linha);
      end;
    end;

    if control<>0 then
      sherror(control)
    else
      writeln;
  until FALSE;

end.

```