```
program Treegen;
(* Generates the language defined by a  regular  expression*)
(* Program written by A. Kreczmar 1982
                 proof   written by A. Salwicki    1990   *)


 unit REGEXP:coroutine;
            (* an object in the hierarchy of subtypes  of type REGEXP represents a regular expression *)
(*
```

Theorem

For every object o in the hierarchy of classes that inherit from Regexp class the program *Pr* (see below), when executed will print all the words of the regular language represented by the object o and then it will stop.

```
Pr:  I:=0;
                  do
                   attach(o);
                   (* print the WORD *)
                          for J:=1 to I
                          do
                            write(WORD(J))
                          od;
                          writeln;
                   if W.B then exit fi
                  od
```

Lemma

Let i0 be the value of the variable I. Suppose that the some words of the language L(o) were generated by the earlier activations of the **coroutine** o.
An execution of command **attach**(o) has the following effect: the subsequent word of the language L(o) is concatenated to the content of the WORD(1), ... , WORD(I); i.e. the *new* word is placed beginning of the position WORD(I+1). The value of B attribute becomes true iff all the words of the language L(o) were shown.

PROOF of the lemma is distributed in the subclasses of the class regexp, i.e. the proof goes by induction with respect to the length of a regular expression *)

```
    var B:BOOL; (* B ≡ all the words of the language were shown *)
 begin
   return
   inner;
   B := true
 end REGEXP;




  unit ATOM: REGEXP class(C:CHAR);
 (* an atomic regular expression consists of a letter
      Proposition. An execution of attach statement applied to this object will place the letter C on I+1-th place
                   in the table WORD and the value of B will be assigned to true. In this way the whole regular
                  language is displayed at once.
    in this way we proved the base of the induction proof of Lemma. *)
 begin
   do
    I:=I+1; (* update the position *)
    WORD(I):=C;
    B:=TRUE;
    detach
   od
 end ATOM;



 unit UNION: REGEXP class(L,R:REGEXP);
          (* represents the expression (L ∪ R) i.e. the union *)
```

(* **Proposition**. *Assume that objects L and R enjoy the property expressed by the Lemma then any time this coroutine will be attached we obtain a new word of the union of the languages L and R*.

Consider, a regular expresion of the length k. By our definition it is either a union object or a concatenation object.

Let o be a union object i.e. o **is** UNION. The structure of its commands assures the following

```
while not exhausted(L)
do
        attach(L)               -- by induction hypothesis this command returns a word of L language
od
(* L.B = true *)                --  the exhaustion mark for L
while not exhausted(R)
do
        attach(R)               -- by induction hypothesis this command returns a word of R language
od
(* R.B = true
   B = true *)
```

It is evident that in this way by repeated execution of attach(o) one obtains a sequence of words composed from the all words of L language followed by the sequence of all words from the R language. *)

```
var M: INTEGER;
begin
 do     (* repeat : store I; generate one word (first from L next from R; detach; restore I until exhausted *)
  M:=I;
      (* I is the position of the lastly generated letter.              *)
      (* M+1 is the position where the current UNION object   *)
      (* will place the letters of the currently generated word.    *)
  do
   attach(L);    (* by the inductive assumption this statement causes that one word will be generated of the
                 language L and it will be concatenated to the content of WORD(1) , ... , WORD(I)  *)
   if L.B then exit fi;
   detach;
   I:=M  (* reestablish the position in the table WORD for the next word *)
  od;
  L.B:=FALSE;  (* restart language L *)
  do
   detach;
   I:=M;  (* reestablish the position in the table WORD for the next word  *)
   attach(R);    (* by the inductive assumption this statement causes that one word will be generated of the
                 language R and it will be concatenated to the content of WORD(1) , ... , WORD(I)  *)
   if R.B then exit fi;
  od;
  R.B:=FALSE;  (* restart language R *)
  B:=TRUE;
  detach;
 od;
end UNION;
```

```
unit CONCATENATION: REGEXP class(L,R:REGEXP);
      (* represents the concatenation (L•R) of the languages represented by the regular expressions L and R *)
   (* Suppose the object o is of the class CONCATENATION.
```

Now the loop of commands of object o assures basically the following

```
while not exhausted
do
    store (I);
    attach(L);          -- a word from L
    attach(R);          -- followed by a word from R
    detach;             -- hence a word of (L R) is given
    restore(I)
od
```

with the necessary reactions to a case when one language (L or R) ends.
It is clear that if the object L and R enjoy the property mentioned in the Lemma then the object o enjoys it too*)

```
var N,M:INTEGER;
begin
  do
    M:=I; (*begin of first language word position *)
    do
      attach(L);
      N:=I; (* begin of the second language word position *)
      do
        attach(R);
        if R.B then if L.B then exit exit else exit fi fi;
        detach; I:=N (* restart language R word generation position *)
      od;
      R.B:=FALSE; (* restart language R *)
      detach; I:=M (* restart language L word generation position *)
    od;
    R.B,L.B:=FALSE; B:=TRUE; detach
  od;
end CONCATENATION;
```

```
const N=50;   (* DIMENSION FOR ARRAY WORD *)

var     A,B,C,D,E,W,V,L,O,G,II,NN:REGEXP,
        I,J,N,M:INTEGER;
          (* I = GLOBAL POSITION POINTER FOR ARRAY WORD *)
var WORD: arrayof CHAR; (* BUFFER FOR WORDS GENERATION *)

begin
      writeln(" LANGUAGE GENERATOR USING COROUTINES");
      writeln(" LANGUAGE IS REPRESENTED AS A TREE WITH OPERATIONS IN NODES");
      writeln(" OUR OPERATIONS ARE SET THEORETICAL JOIN AND CONCATENATION OF");
      writeln(" LANGUAGES");writeln;
      A:=new ATOM('A'); B:=new ATOM('B'); C:=new ATOM('C');
      D:=new ATOM('D'); E:=new ATOM('E');
      L:=new ATOM('L'); G:=new ATOM('G');
      II:=new ATOM('I'); NN:=new ATOM('N');
      O:=new ATOM('O');
      W:=new UNION(A,L);
      W:=new CONCATENATION(W,new UNION(D,O));
      V:=new CONCATENATION(II,C);
      V:=new UNION(V,new CONCATENATION(L,new CONCATENATION(A,NN)));
      V:=new CONCATENATION(G,V);
      V:=new UNION(A,V);
      W:=new CONCATENATION(W,V);
      writeln(" WE HAVE LANGUAGE DEFINED BY THE FOLLOWING EXPRESSION");
      writeln;
      writeln(" (A∪L)•(D∪O)•(A∪G•(I•C∪L•A•N))");
      writeln; writeln;
      array WORD dim(1:N);
      do
```

```
            attach(W);
            write(" ");
            for J:=1 to I
            do
                write(WORD(J))
            od;
            writeln;
            if W.B then exit fi
        od
      end
```

(*

Theorem

For every object o in the hierarchy of classes that inherit from Regexp class the program Pr (see below), when executed will print all the words of the regular language represented by the object o and then it will stop.

```
Pr:  I:=0;
     do
       attach(o);
       for J:=1 to I
       do
         write(WORD(J))
       od;
       writeln;
       if W.B then exit fi
     od
```

Lemma

Let i0 be the value of the variable I. Suppose that the some words of the language L(o) were generated by the earlier activations of the **coroutine** o.

An execution of command **attach**(o) has the following effect: the subsequent word of the language L(o) is concatenated to the content of the WORD(1), ... , WORD(I); i.e. the *new* word is placed beginning of the position WORD(I+1). The value of B attribute becomes true iff all the words of the language L(o) were shown.

Proof.

Induction with respect to the length of the expression represented by the object o.

Base. Suppose the actual type of o is ATOM. Then the thesis of the lemma is satisfied.

Induction step. Suppose the lemma holds for every regular expression shorter than an integer k. Consider, a regular expresion of the length k. By our definition it is either a union object or a concatenation object.

case A. Let o be a union object i.e. o **is** UNION. The structure of its commands assures the following

```
        while not exhausted(L)
        do
                attach(L)               -- by induction hypothesis this command returns a word of L language
        od
        L.B := true                     -- set the exhaustion mark for L
        while not
        do
                attach(R)               -- by induction hypothesis this command returns a word of R language
        od
        L.R := true
        B := true
```

It is evident that in this way by repeated execution of attach(o) one obtains a sequence of words composed from the all words of L language followed by the sequence of all words from the R language.

case B Suppose the object o is of the class CONCATENATION.

Now the loop of commands of object o assures basically the following

```
while not exhausted
do
    store (I);
    attach(L);          -- a word from L
    attach(R);          -- precedes a word from R
    detach;             -- hence a word of (L R) is given
    restore(I)
od
```

with the necessary reactions to a case when one language (L or R) ends.
It is clear that if the object L and R enjoy the property mentioned in the Lemma then the object o enjoys it too.

This ends the proof of the Lemma. ♦