Nerekurzívní zápis operace Insert nad AVL stromem.

Převzato z přílohy skript Honzík J.,a kolektiv"Programovací techniky".

PŘÍLOHA A

PROGRAM AVLSTROM pro demonstraci nerekurzivního zápisu operace INSERT v AVL stromu

```
PROGRAM AVLSTROM(INPUT,OUTPUT);

(*
   PROGRAM DEMONSTRUJE VYTVARENI
   AVL - STROMU  S  VYROVNAVANIM
   NEREKURZIVNI  METODOU.  JEHO
   CINNOST  JE  NAPLNI PROCEDURY
   ZARAD.

   DATUM: 29. 5. 1983
   AUTOR: P. LAMPA, K. SOLNICKY
          VUT BRNO FE, III-EP
*)

TYPE
   PUZEL=^TUZEL;                                  (* UKAZATEL NA UZEL STROMU *)
   TSTAV=(LEVY_DELSI,STEJNE,PRAVY_DELSI); (* STAVY PODSTROMU UZLU *)
   TPODSTROM=(LEVY,PRAVY);                        (* TYP PODSTROMU *)
   TKOREN=ARRAY [TPODSTROM] OF PUZEL;             (* UKAZATELE NA LEVY A PRAVY
                                                     PODSTROM *)
   TPOLOZKY=RECORD
              PREDCHOZI: PUZEL;       (* UKAZATEL NA PREDCHOZI UZEL *)
              STRANA: TPODSTROM;      (* TYP PODSTROMU VE KTEREM JSME *)
            END;

   TZASOBNIK=ARRAY[1..16] OF TPOLOZKY; (* ZASOBNIK PRO CESTU STROMEM *)

   TUZEL=RECORD
            HODNOTA: INTEGER; (* HODNOTA UZLU *)
            KOREN: TKOREN;    (* UKAZATELE NA PODSTROMY *)
            STAV: TSTAV;      (* VYROVNANOST PODSTROMU *)
         END;

VAR
   HLAVICKA: PUZEL;  (* VRCHOL STROMU *)
   DATA: INTEGER;    (* UKLADANA HODNOTA *)
   NALEZENO: BOOLEAN;

PROCEDURE ZARAD(DATA: INTEGER; VAR P: PUZEL; VAR NASEL: BOOLEAN);

(* TATO PROCEDURA ZARADI DO STROMU S HLAVICKOU P ZADANA DATA.
   POKUD JIZ VE STROMU TYTO DATA JSOU, NABUDE HODNOTY FALSE,
   JINAK DATA ZARADI, ZAPISE CESTU DO ZASOBNIKU A V PRIPADE
   POTREBY UPRAVI STROM NA AVL STROM. *)

   VAR
      PP,                     (* POMOCNY UKAZATEL *)
      PREDPOSLEDNI: PUZEL; (* NEJBLIZSI VYSSI UZEL NEZ HLEDANY *)
      STRANA: TPODSTROM;   (* TYP PODSTROMU NEJBLIZSIHO VYSSIHO UZLU *)
      ZASOBNIK: TZASOBNIK;
      VRCHOL: INTEGER;     (* VRCHOL ZASOBNIKU *)

   (* NASLEDUJICI PROCEDURY SLOUZI K UPRAVE OKOLI UZLU PODSTROMU.
      NA OBRAZCICH JE ZNAZORNEN STAV PODSTROMU PRED A PO UPRAVE. *)

   PROCEDURE LL_ROT(VAR P: PUZEL);

      (*       3                2
              2       =>       1 3
             1
      *)

      VAR
         P1: PUZEL;
```

```
    BEGIN
       P1:=P^.KOREN[LEVY];
       P^.KOREN[LEVY]:=P1^.KOREN[PRAVY];
       P1^.KOREN[PRAVY]:=P;
       P^.STAV:=STEJNE;
       P:=P1;
    END; (* LL_ROT *)

 PROCEDURE RR_ROT(VAR P:PUZEL);

    (*     1                  2
           2       =>       1 3
            3
    *)

    VAR
       P1: PUZEL;

    BEGIN
       P1:=P^.KOREN[PRAVY];
       P^.KOREN[PRAVY]:=P1^.KOREN[LEVY];
       P1^.KOREN[LEVY]:=P;
       P^.STAV:=STEJNE;
       P:=P1;
    END; (* RR_ROT *)

 PROCEDURE LR_ROT(VAR P:PUZEL);

    (*       3                2
            1       =>       1 3
             2
    *)

    VAR
       P1,P2: PUZEL;

    BEGIN
       P1:=P^.KOREN[LEVY];
       P2:=P1^.KOREN[PRAVY];
       P1^.KOREN[PRAVY]:=P2^.KOREN[LEVY];
       P2^.KOREN[LEVY]:=P1;
       P^.KOREN[LEVY]:=P2^.KOREN[PRAVY];
       P2^.KOREN[PRAVY]:=P;
       IF P2^.STAV = LEVY_DELSI THEN P^.STAV:=PRAVY_DELSI
                                ELSE P^.STAV:=STEJNE;
       IF P2^.STAV = PRAVY_DELSI THEN P1^.STAV:=LEVY_DELSI
                                 ELSE P1^.STAV:=STEJNE;
       P:=P2;
    END; (* LR_ROT *)

 PROCEDURE RL_ROT(VAR P:PUZEL);

    (*     1                  2
            3       =>       1 3
           2
    *)

    VAR
       P1,P2: PUZEL;

    BEGIN
       P1:=P^.KOREN[PRAVY];
       P2:=P1^.KOREN[LEVY];
       P1^.KOREN[LEVY]:=P2^.KOREN[PRAVY];
       P2^.KOREN[PRAVY]:=P1;
       P^.KOREN[PRAVY]:=P2^.KOREN[LEVY];
       P2^.KOREN[LEVY]:=P;

      IF P2^.STAV =PRAVY_DELSI THEN P^.STAV:= LEVY_DELSI
                               ELSE P^.STAV:=STEJNE;
      IF P2^.STAV =  LEVY_DELSI THEN P1^.STAV:=PRAVY_DELSI
                                ELSE P1^.STAV:=STEJNE;
      P:=P2;
END; (* RL_ROT *)
```

```
(* PRO UPLNE POCHOPENI TECHTO PROCEDUR JE NEJLEPE SI JEJICH
   CINNOST ODSIMULOVAT NA PAPIROVEM POCITACI *)


PROCEDURE NOVY_UZEL(DATA: INTEGER; VAR P: PUZEL);

   (* PROCEDURA VYTVORI NOVY UZEL A NASTAVI JEHO
      POLOZKY NA PRISLUSNE HOTNOTY *)

   BEGIN
      NEW(P);
      WITH P^ DO
         BEGIN
            HODNOTA:=DATA;
            KOREN[LEVY]:=NIL;
            KOREN[PRAVY]:=NIL;
            STAV:=STEJNE;
         END;
   END; (* NOVY_UZEL *)

PROCEDURE VYBALANCOVANI;

   (* PROCEDURA PROVEDE VYBALANCOVANI STROMU *)

   VAR
      BALANCOVAT: BOOLEAN;
      P: PUZEL;

   BEGIN
      BALANCOVAT:=TRUE; (* BUDEME BALANCOVAT *)
      WHILE BALANCOVAT AND (VRCHOL > 1) DO
         BEGIN
            P:=ZASOBNIK[VRCHOL].PREDCHOZI;
            IF ZASOBNIK[VRCHOL].STRANA=LEVY THEN (* PRIDALI JSME
                                                    LEVY UZEL *)
               CASE P^.STAV OF

               PRAVY_DELSI:

                  (* UZEL MEL PRAVY PODSTROM
                     A TAK SE SROVNAL *)

                  BEGIN
                     P^.STAV:=STEJNE;
                     BALANCOVAT:=FALSE;
                  END;

               STEJNE:

                  (* UZEL BYL KONCOVY
                     A JESTE TO UJDE *)

                  P^.STAV:=LEVY_DELSI;

               LEVY_DELSI:

                  (* UZEL JIZ MEL LEVY PODSTROM
                     A TAK JEJ MUSIME SROVNAT *)

                  WITH ZASOBNIK[VRCHOL-1] DO
                     BEGIN
                        IF P^.KOREN[LEVY]^.STAV=LEVY_DELSI THEN
                           LL_ROT(PREDCHOZI^.KOREN[STRANA])
                        ELSE
                           LR_ROT(PREDCHOZI^.KOREN[STRANA]);
                        P:=PREDCHOZI^.KOREN[STRANA];
                        P^.STAV:=STEJNE;
                        BALANCOVAT:=FALSE;
                     END;
               END (* CASE *)
```

```
            ELSE (* PRIDALI JSME PRAVY UZEL *)
               CASE P^.STAV OF

               LEVY_DELSI:

                  (* UZEL MEL LEVY PODSTROM
                     A TAK SE SROVNAL *)

                  BEGIN
                     P^.STAV:=STEJNE;
                     BALANCOVAT:=FALSE;
                  END;

               STEJNE:

                  (* UZEL BYL KONCOVY
                     A JESTE TO UJDE *)

                  P^.STAV:=PRAVY_DELSI;

               PRAVY_DELSI:

                  (* UZEL JIZ MEL PRAVY PODSTROM
                     A TAK JEJ MUSIME SROVNAT *)

                  WITH ZASOBNIK[VRCHOL-1] DO
                     BEGIN
                        IF P^.KOREN[PRAVY]^.STAV=PRAVY_DELSI THEN
                           RR_ROT(PREDCHOZI^.KOREN[STRANA])
                        ELSE
                           RL_ROT(PREDCHOZI^.KOREN[STRANA]);
                        P:=PREDCHOZI^.KOREN[STRANA];
                        P^.STAV:=STEJNE;
                        BALANCOVAT:=FALSE;
                     END;
               END; (* CASE *)
            VRCHOL:=VRCHOL-1;
         END; (* WHILE *)
   END;  (* VYBALANCOVANI *)

BEGIN

   (* INICIALIZACNI PRIKAZY *)

   VRCHOL:=0;
   STRANA:=LEVY;
   PREDPOSLEDNI:=P;
   PP:=P^.KOREN[LEVY];
   IF PP=NIL THEN NOVY_UZEL(DATA,P^.KOREN[LEVY]) (* ZALOZENI STROMU *)
   ELSE
      (* ZARAZENI DO STROMU *)

      BEGIN
         NASEL:=FALSE;

         WHILE (PP<>NIL) AND NOT NASEL DO
            BEGIN

               (* ZAPISEME CESTU DO ZASOBNIKU *)

               VRCHOL:=VRCHOL+1;
               ZASOBNIK[VRCHOL].STRANA:=STRANA;
               ZASOBNIK[VRCHOL].PREDCHOZI:=PREDPOSLEDNI;
               PREDPOSLEDNI:=PP;

               (* ROZHODNEME SE KUDY DALE PUJDEME *)

               IF PP^.HODNOTA=DATA THEN NASEL:=TRUE
               ELSE
                  IF DATA<PP^.HODNOTA THEN
                     BEGIN
                        STRANA:=LEVY;
                        PP:=PP^.KOREN[LEVY];
                     END
```

```
                    ELSE
                       BEGIN
                          STRANA:=PRAVY;
                          PP:=PP^.KOREN[PRAVY];
                       END;
                 END; (* WHILE *)
              IF NOT NASEL THEN
                 BEGIN
                    NOVY_UZEL(DATA,PP);      (* VYTVORENI NOVEHO UZLU *)
                    PREDPOSLEDNI^.KOREN[STRANA]:=PP; (* NAVAZANI DO STROMU *)
                    VRCHOL:=VRCHOL+1;
                    ZASOBNIK[VRCHOL].STRANA:=STRANA;
                    ZASOBNIK[VRCHOL].PREDCHOZI:=PREDPOSLEDNI;
                    VYBALANCOVANI;
                 END;
           END; (* IF *)
        ZARAD:=NASEL;
     END; (* ZARAD *)

BEGIN

(* HLAVNI PROGRAM *)

   NOVY_UZEL(MAXINT,HLAVICKA);
   WHILE NOT EOF(INPUT) DO
      BEGIN
         READLN(DATA);
         ZARAD(DATA,HLAVICKA,NALEZENO);
      END;
END.
```