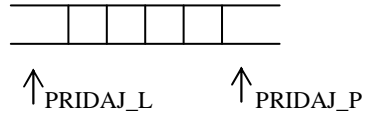


Príklad 1: Napíšte algebraickú špecifikáciu ADT (abstraktné dátové typy) reťazec (string).

reťazec =



Riešenie:

sorts: abeceda

reťazec

opns: a, b, ..., z : \longrightarrow abeceda

PRÁZDNY : \longrightarrow reťazec

VYTVOR : abeceda \longrightarrow reťazec

SPOJ : reťazec reťazec \longrightarrow reťazec

PRIDAJ_L : abeceda reťazec \longrightarrow reťazec

PRIDAJ_P : reťazec abeceda \longrightarrow reťazec

eqns: $\forall x \in \text{abeceda}; \forall s, s_1, s_2, s_3 \in \text{reťazec}$

SPOJ (s, PRÁZDNY) = s

SPOJ (PRÁZDNY, s) = s

SPOJ (SPOJ (s₁, s₂), s₃) = SPOJ (s₁, SPOJ (s₂, s₃))

PRIDAJ_L (x, s) = SPOJ (VYTVOR (x), s)

PRIDAJ_R (s, x) = SPOJ (s, VYTVOR (x))

Je možné definovať aj testovacie operácie:

JE_PRÁZDNY : reťazec \longrightarrow boolean

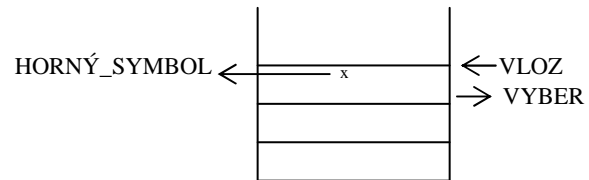
s axiómami :

JE_PRÁZDNY (PRÁZDNY) = true

JE_PRÁZDNY (VYTVOR (x)) = false

Príklad 2: Napíšte algebraickú špecifikáciu ADT zásobník (stack)

zásobník =



Riešenie:

sorts: abeceda

zásobník

opns: $A_1, \dots, A_n : \longrightarrow \text{abeceda}$

PRÁZDNY : \longrightarrow zásobník

VLOŽ : abeceda zásobník \longrightarrow zásobník

VYBER : zásobník \longrightarrow zásobník

HORNÝ_SYMBOL : zásobník \longrightarrow abeceda

CHYBA : \longrightarrow abeceda

eqns: $\forall x \in \text{abeceda}, s \in \text{zásobník}$

VYBER (VLOŽ(x, s)) = s

HORNÝ_SYMBOL (VLOŽ(x, s)) = x

VYBER (PRÁZDNY) = PRÁZDNY

HORNÝ_SYMBOL (PRÁZDNY) = CHYBA

Príklad 3: Napíšte algebraickú štruktúru ADT binárny strom.

Riešenie:

bin_strom =

sorts: abeceda

bin_strom

opus: $A_1, \dots, A_n: \rightarrow \text{abeceda}$

LIST: $\text{abeceda} \rightarrow \text{bin_strom}$

L_PODSTROM: $\text{bin_strom abeceda} \rightarrow \text{bin_strom}$

P_PODSTROM: $\text{abeceda bin_strom} \rightarrow \text{bin_strom}$

LP_PODSTROM: $\text{bin_strom abeceda bin_strom} \rightarrow \text{bin_strom}$

ŠÍRKA: $\text{bin_strom} \rightarrow \text{nat}$

HRANY: $\text{bin_strom} \rightarrow \text{nat}$

UZLY: $\text{bin_strom} \rightarrow \text{nat}$

LEN_LIST: $\text{bin_strom} \rightarrow \text{bool}$

equs: $\forall x \in \text{abecedy}, b, b_1, b_2, : \text{bin_strom}$

ŠÍRKA (LIST(x)) = SUCC (0)

ŠÍRKA (L_PODSTROM(b,x)) = ŠÍRKA(b)

ŠÍRKA (P_PODSTROM(x,b)) = ŠÍRKA(b)

ŠÍRKA (LP_PODSTROM(b₁,x,b₂)) = ADD(ŠÍRKA(b₁),ŠÍRKA(b₂))

HRANY (LIST(x)) = 0

HRANY (L_PODSTROM(b,x)) = SUCC (HRANY(b))

HRANY (P_PODSTROM(x,b)) = SUCC (HRANY(b))

HRANY (LP_PODSTROM(b₁,x,b₂)) = SUCC (SUCC (ADD (HRANY(b₁), HRANY(b₂))))

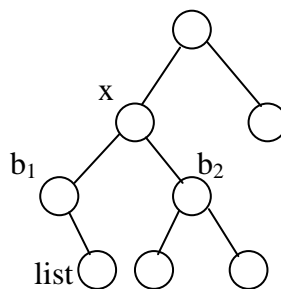
UZLY (b) = SUCC (HRANY (b))

LEN_LIST (LIST (x)) = TRUE

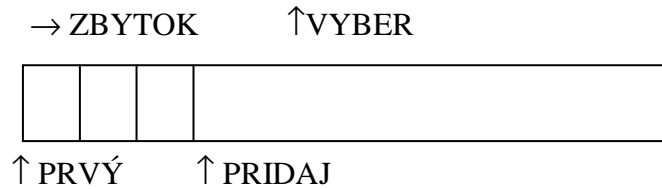
LEN_LIST (L_PODSTROM (b,x)) = LEN_LIST (b)

LEN_LIST (P_PODSTROM (x,b)) = LEN_LIST (b)

LEN_LIST (LP_PODSTROM (b₁,x,b₂)) = FALSE



Príklad 4: Napíšte algebraickú špecifikáciu ADT jednosmerný zoznam (list 1).



Riešenie:

zoznam1 =

sorts: abeceda
zoznam1

opus: $A_1, \dots, A_n \rightarrow \text{abeceda}$

VYTVOR : $\text{abeceda} \rightarrow \text{zoznam1}$

PRIDAJ : $\text{zoznam1 } \text{abeceda} \rightarrow \text{zoznam1}$

VYBER : $\text{zoznam1} \rightarrow \text{abeceda}$

PRVÝ : $\text{zoznam1} \rightarrow \text{abeceda}$

DĹŽKA : $\text{zoznam1} \rightarrow \text{nat}$

ZBYTOK : $\text{zoznam1} \rightarrow \text{zoznam1}$

PRÁZDNY : $\rightarrow \text{zoznam1}$

equs: $\forall x \in \text{abeceda}, z \in \text{zoznam1}$

DĹŽKA (PRÁZDNY) = 0

DĹŽKA (VYTVOR(x)) = SUCC(0)

DĹŽKA (PRIDAJ (z,x)) = SUCC (DĹŽKA (z))

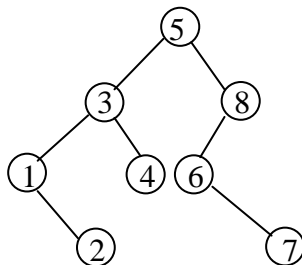
VÝBER (PRIDAJ (z,x)) = x

PRVÝ (VYTVOR (x)) = x

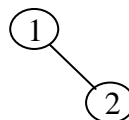
ZBYTOK (VYTVOR (x)) = PRÁZDNY

·
·
·

Príklad 5: Napíšte algoritmus pre prehľadávanie binárneho stromu metódou in order s použitím rekúzie.



Pojmy: napr. 5 - koreň
 3, 8 - synovia
 2, 4, 7 - listy
 vertex (1) - je koreň podstromu



hlbka (4) - je dĺžka cesty od koreňa ku 4 = 2
 výška stromu = dĺžka najdlhšej cesty od koreňa po list

Riešenie: Postup: 1. prejdeme do ľavého podstromu koreňa r (ak existuje)
 2. prejdeme do r
 3. prejdeme do pravého podstromu koreňa r (ak existuje)

```

procedure INORDER (VERTEX);
  begin
    if LSYN [VERTEX] ≠ 0 then INORDER (LSYN [VERTEX]);
    LABEL [VERTEX] ← POČÍTADLO;
    POČÍTADLO ← POČÍTADLO + 1;
    if PSYN [VERTEX] ≠ 0 then INORDER (PSYN [VERTEX]);
  end;

  begin
    POČÍTADLO ← 1;
    INORDER (ROOT);
  end.
  
```

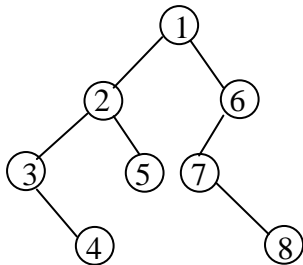
Príklad 6: Napíšte algoritmus pre prehľadávanie (značenie) binárneho stromu metódou in order bez použitia rekurzie.

Riešenie:

```
begin
    POČÍTADLO  $\leftarrow$  1;
    VERTEX  $\leftarrow$  ROOT;
    ZÁSOBNÍK  $\leftarrow$  prázdny;
L:   while LSYN [VERTEX]  $\neq$  0 do
        begin
            push VERTEX onto ZÁSOBNÍK;
            VERTEX  $\leftarrow$  LSYN [VERTEX];
        end;
H:   LABEL [VERTEX]  $\leftarrow$  POČÍTADLO;
    POČÍTADLO  $\leftarrow$  POČÍTADLO + 1;
    if PSYN [VERTEX]  $\neq$  0 then
        begin
            VERTEX  $\leftarrow$  PSYN [VERTEX];
            goto L;
        end;
    if ZÁSOBNÍK  $\neq$  prázdny then
        begin
            VERTEX  $\leftarrow$  prvok na vrchole ZÁSOBNÍKa;
            pop ZÁSOBNÍK;
            goto H;
        end;
    end.
```

Popis: Prechádzame po ľavom podstrome koreňa a ešte neoznačené uzly vkladáme do ZÁSOBNÍKa. List ľavého podstromu očísľujeme, ideme hore a do pravého podstromu atď.

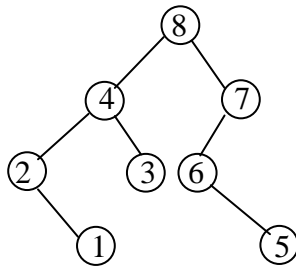
Príklad 7: Napíšte algoritmus pre značenie binárnych stromov metódou pre order s použitím rekúzie.



Riešenie:

```
procedure PREORDER (VERTEX);  
  begin  
    LABEL [VERTEX]  $\leftarrow$  POČÍTADLO;  
    POČÍTADLO  $\leftarrow$  POČÍTADLO + 1;  
    if LSYN [VERTEX]  $\neq$  0 then  
      PREORDER (LSYN [VERTEX]);  
    if PSYN [VERTEX]  $\neq$  0 then  
      PREORDER (PSYN [VERTEX]);  
  end;  
  
  begin  
    POČÍTADLO  $\leftarrow$  1;  
    PREORDER (ROOT);  
  end.
```

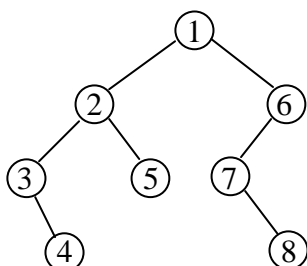
Príklad 8: Napíšte algoritmus pre značenie binárnych stromov metódou post order s použitím rekúzie.



Riešenie:

```
procedure POSTORDER (VERTEX);  
  begin  
    if LSYN [VERTEX]  $\neq$  0 then  
      POSTORDER (LSYN [VERTEX]);  
    if PSYN [VERTEX]  $\neq$  0 then  
      POSTORDER (PSYN [VERTEX]);  
    LABEL [VERTEX]  $\leftarrow$  POČÍTADLO;  
    POČÍTADLO  $\leftarrow$  POČÍTADLO + 1;  
  end;  
  
  begin  
    POČÍTADLO  $\leftarrow$  1;  
    POSTORDER (ROOT);  
  end.
```


Príklad 9: Napíšte algoritmus pre značenie binárnych stromov metódou preorder bez použitia rekurzie.



Riešenie:

```

begin
    POČ ← 1
    VER ← ROOT
    ZÁS ← prázdny
K:  LABEL[VER] ← POČ;
    POČ ← POČ + 1;
L:  while LSYN[VER] ≠ 0 then
        begin
            push VER onto ZÁS;
            VER ← LSYN[VER];
            goto K;
        end
    if PSYN[VER] ≠ 0 then
        begin
            VER ← PSYN[VERTEX];
            goto K
        end
    if ZÁS ≠ prázdny then
        begin
            pop ZÁS;
            VER ← prvok na vrchole ZÁS
            goto L
        end
    end
end.
  
```

Príklad 10: Napíšte algoritmus pre značenie binárnych stromov metódou postorder bez použitia rekurzie.

Riešenie:

```
begin POČ ← 1
      VER ← ROOT
      ZÁS ← prázdny
L:    while LSYN[VER] ≠ 0 then
      begin
            if top(ZÁS) = VER then goto Z else push VER onto ZÁS;
            VER ← LSYN[VER]
      end
P:    if PSYN[VER] ≠ 0 then
      begin
            if top(ZÁS) = VER then goto Z else push VER onto ZÁS;
            VER ← LSYN[VER]
            goto L
      end;
      LABEL[VER] ← POČ;
      POČ ← POČ + 1;
Z:    if ZÁS ≠ prázdny then
      begin
            VER ← top(ZÁS)
            LABEL[VER] ← POČ;
            POČ ← POČ + 1;
            pop ZÁS
            goto P
      end
end.
```

Príklad 11: Napíšte algoritmus dynamického programovania pre zistenie minimálneho počtu operácií pre násobenie n matic $M = M_1 \times M_2 \times \dots \times M_n$

Riešenie:

Zložit' $(M(p,q) \times N(p,r)) = p.q.r$

$\Rightarrow O(n^3)$

Označíme:

Matica $M_i (r_{i+1}, r_i)$

r_{i+1} - počet riadkov

r_i - počet stĺpcov

napr.: Pre súčin $M_1 (0,200 \times (M_2 (20,50) \times (M_3 (50,1) \times M_4 (1,100)))$ potrebujeme 125000 operácií, ale pre súčin $(M_1 (0,200 \times (M_2 (20,50) \times (M_3 (50,1)))) \times M_4 (1,100)$ potrebujeme len 2200 operácií.

Hľadáme algoritmus nie pre násobenie matic, ale pre zistenie minimálneho počtu operácií potrebných pre vynásobenie n matic.

Vstup: r_0, r_1, \dots, r_n

(počty riadkov a stĺpcov matic)

Výstup: m_{ij} – minimálna zložitosť násobenia $M_i \times M_{i+1} \times \dots \times M_j$ pre $1 \leq i \leq j \leq n$

$$m_{ij} = \begin{cases} 0 & \text{pre } i = j \\ \min_{i \leq k \leq j} (m_{ik} + m_{k+1,j} + r_{i-1} r_k r_j) & \text{pre } i \neq j, i \leq k \leq j \end{cases}$$

$M' = M_i \times \dots \times M_k \quad M'' = M_{k+1} \times \dots \times M_j \quad M'(r_{i-1}, r_k) \times M''(r_k, r_j)$

Algoritmus:

begin

for $i \leftarrow 1$ until n do $m_{ij} \leftarrow 0$;

for $l \leftarrow 1$ until $n-1$ do

for $i \leftarrow 1$ until $n-1$ do

begin

$j \leftarrow i+l$;

$m_{ij} \leftarrow \min_{i \leq k \leq j} (m_{ik} + m_{k+1,j} + r_{i-1} r_k r_j)$

end

write m_{in}

end

pre $M_1 \times M_2 \times M_3 \times M_4$:

$m_{11} = 0$	$m_{22} = 0$	$m_{33} = 0$	$m_{44} = 0$
$m_{12} = 10000$	$m_{23} = 1000$	$m_{34} = 5000$	
$m_{13} = 1200$	$m_{24} = 3000$		
$m_{14} = 2200$			

Príklad 12: Napíšte metódou divide-and-conquer algoritmus pre nájdenie maximálneho a minimálneho prvku množiny M.

Riešenie:

Nech $\|M\| = n - \text{párne}$

Divide-and-conquer metóda rozdelí problém na menšie časti, nájde riešenia a kombinuje ich do výsledného riešenia.

Množinu M rozdelíme do podmnožín M_1, M_2 , každá $n/2$ prvkov. Algoritmus hľadá maximum a minimum v oboch rekurzívnou aplikáciou algoritmu. Max. a min. prvok M sa vypočíta z oboch získaných maxím a miním.

Vstup: Množina M s párnym počtom prvkov $n \geq 2$

Výstup: Maximum a minimum M

Algoritmus:

```
Procedure MAXIMUM (M);  
begin  
    if  $\|M\| = 2$  then  
        begin  
            let  $S = \{a,b\}$   
            return (MAX(a,b),MIN(a,b))  
        end  
    else  
        begin  
            divide M to  $M_1, M_2$ ,  $\|M_1\| = \|M_2\| = n/2$  ;  
             $(\max_1, \min_1) \leftarrow \text{MAXIMUM}(M_1)$ ;  
             $(\max_2, \min_2) \leftarrow \text{MAXIMUM}(M_2)$ ;  
            return (MAX( $\max_1, \max_2$ ), MIN( $\min_1, \min_2$ ))  
        end  
    end
```

Zložitosť:

$$T(n) = \begin{cases} 1 & \text{pre } n = 2 \\ 2 T(n/2) + 2 & \text{pre } n > 2 \end{cases}$$

t.j. $T(n) = 3/2 \cdot (n-2)$.

Dokázat' indukciou:

1. pre $n = 2$ platí
2. ak platí pre $n = m$, $m \geq 2$, potom
 $T(2m) = 2 \cdot (3/2 \cdot m - 2) + 2 = 3/2(2m) - 2$

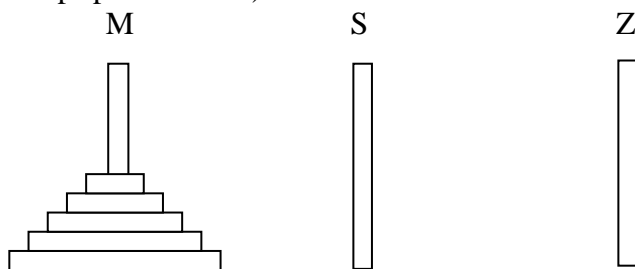
Príklad 13:

Príbeh: Pred kostolom v Hanoji stáli 3 stĺpy: medený, strieborný a zlatý. Ma medenou bolo 100 kruhov tak, že ich polomer sa zmenšoval z dola na hor. Podľa legendy, ak sa podarí preniesť preniesť kruhy na zlatý stĺp tak, aby ich polomer sa zmenšoval z dola na hor a naraz možno preniesť len 1 kruh a nemožno položiť kruh s väčším polomerom na kruh s menším polomerom, potom bude koniec sveta.

Jeden starý mních rozmýšľal a navrhol 3 kroky:

1. preniesie 99 kruhov z medeného stĺpa na strieborný
2. preniesie 1 kruh z medeného stĺpa na zlatý
3. preniesie 99 kruhov zo strieborného stĺpa na zlatý

Pretože bol múdry, zveril kroky 1. a 3. svojmu pomocníkovi. Ten opakoval postup a body 2., 3. Zveril ďalšiemu mníchovi. atď., kým úloha nebola vyriešená. (klasický prípad rekurzie)



Algoritmus: s_1, s_2, s_3 – stĺpy, n – počet kruhov

```
Procedure veže(u,  $s_1, s_2, s_3$ );  
  begin  
    if  $n > 1$  then  
      veže( $n-1, s_1, s_3, s_2$ );  
      preložkruh( $n, s_1, s_2$ );  
      if  $n > 1$  then  
        veže( $n-1, s_3, s_2, s_1$ );  
    end;  
  procedure preložkruh( $n$ , odkiaľ, kam)  
    begin  
      write(„prelož kruh“,  $n$ , „zo stĺpa“, odkiaľ, „na stĺp“, kam);  
    end;  
  begin  
    veže(100, M, S, Z);  
  end
```

Výstup:

```
pre n = 3:  
  prelož kruh    1 zo stĺpa    M na stĺp    S  
                2              M  
                :  
                :  
                atď.
```

Príklad 14: Umiestnite na šachovnicu 8 dám tak, aby ani jedna neohrozovala ostatné

Riešenie:

Dáma sa môže pohybovať vodorovne, zvisle, diagonálne ľubovoľný počet políček.

Postavenie dámy

$D[1..8, 1..8]$

↓ ↓
riadok stĺpec

redukcia rozhodovania: v každom riadku bude dáma, t.j. $D[1..8] = 1..8$

stĺpec
∧
riadok

procedure umiestneniedám (r);

r – riadok, s – stĺpec

begin

if r > 8 then vytlačvysledok

else

for s = 1 to 8 do

begin

if poleniejeohrozené then

begin

položdámu;

umiestnidámu (r+1);

end

end

end;

procedure položdámu;

begin

K[r] := s

end

function poleniejeobsadené: boolean;

var dáma;

begin

for dáma = to r-1 do

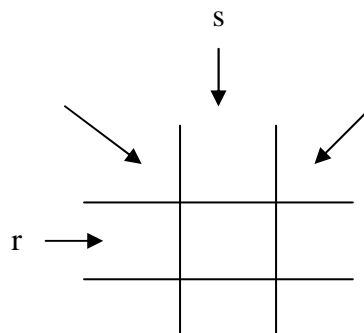
if (D[dáma] - s = 0 or

D[dáma] - s = r - dáma or

D[dáma] - s = dáma - r)

then false

else true



procedure vytlacvysledok : (podľa fantázie autora)

hlavý program:

begin

var D: array[1..8] of integer;

umiestnidámu(1)

end.