

## Zásobník znakov

Máme definovaný zásobník celých čísel

```
type
  TPrvok = char;
  TStack = class
    st:array of TPrvok;
    constructor Create;
    procedure push(p:TPrvok);
    procedure pop(var p:TPrvok);
    function top:TPrvok;
    function empty:boolean;
  end;
...
function stack.top:TPrvok;
begin
  if empty then chyba('Prázdny
zásobník');
  Result:=st[High(st)];
end;
```

V nasledujúcich úlohách predpokladáme, že máme k dispozícii jednu znakovú premennú, jeden znakový zásobník a čítame jeden riadok textového súboru (ukončený napr. znakom #13). Úlohou je napísať funkciu, ktorá zistí, či vstupný riadok súboru je nejakého konkrétneho tvaru:

<b>1. najprv len a, potom rovnaký počet len b</b>
$a^n b^n$ , $0 \leq n$ t.j. či sa vstupný riadok skladá sa len z a-čok a b-čok, ktorých je rovnaký počet a najskôr idú a-čka a potom b-čka. Napr. pre <code>aabb#13</code> , <code>ab#13</code> , <code>#13</code> vráti funkcia <code>true</code> , pre <code>bbaa#13</code> , <code>abb#13</code> , <code>aaebb#13</code> vráti <code>false</code> .
<b>2. najprv len a, potom rovnaký alebo väčší počet b</b>
$a^n b^m$ , $0 \leq n \leq m$ t.j. či sa vstupný riadok skladá sa len z a-čok a b-čok, najskôr idú a-čka a potom b-čka, ktorých je rovnako alebo viac ako a-čok. Napr. pre <code>aabb#13</code> , <code>aabbbb#13</code> , <code>bb#13</code> vráti funkcia <code>true</code> , pre <code>aab#13</code> , <code>bab#13</code> , <code>aabbbx#13</code> vráti <code>false</code> .
<b>3. je palindrom</b>
$wcw^r$ , kde $w$ je z abecedy $\{a,b\}$ , $w^r$ je zrkadlový obraz slova $w$ . Napríklad správne slovo je <code>abaabcbaaba</code> .
<b>4. rovnaký počet a a b</b>
v slove je rovnaký počet a-čok aj b-čok (môžu ísť v ľubovoľnom poradí), ak je na vstupe iný znak ako a,b tak vráti <code>false</code>
<b>5. rovnaký počet a, b a c</b>
rovnaký počet <b>a</b> , <b>b</b> a <b>c</b> (podobne ako 4) – môžete použiť dva zásobníky a jednu znakovú pomocnú premennú.

## Zásobník celých čísel

Máme definovaný zásobník celých čísel, t.j. `TPrvok = integer`;

<b>6. vyhod' všetky nulové prvky</b>
Napíšte procedúru <code>vyhod0(s:TStack)</code> , ktorá pomocou pomocného zásobníka (t.j. druhého, nie toho, čo je na vstupe) spracuje čísla v zásobníku <code>s</code> tak, že vyháďže všetky nulové prvky.
<b>7. nulové prvky zásobníka na spodok:</b>
Napíšte procedúru <code>presun0(s:TStack)</code> , ktorá pomocou pomocného zásobníka spracuje čísla v zásobníku tak, že všetky nulové prvky premiestni na spodok zásobníka.
<b>8. triedenie</b>
Naprogramujte takéto triedenie, ktoré využije dva zásobníky: <ul style="list-style-type: none"><li>• na začiatku máme čísla v prvom zásobníku</li><li>• vyberieme prvé dve čísla, porovnáme ich - menšie z nich vložíme do druhého zásobníka</li><li>• vyberieme ďalšie číslo z prvého zásobníka - porovnáme ho s číslom, ktoré nám ostalo pri predchádzajúcom porovnávaní - menšie číslo vložíme do druhého zásobníka</li></ul>

- opakujeme, až kým prvý zásobník nie je prázdny (pri tomto presýpaní, sme mohli spočítať počet triedených čísel)
- teraz máme v druhom zásobníku všetky čísla, pritom na vrchu je najväčšie z nich
- podobne presypeme čísla z druhého zásobníka do prvého, ale teraz vkladáme do prvého zásobníka väčšie z porovnávaných čísel
- po dostatočnom opakovaní tohto algoritmu budú čísla v zásobníku utriedené

## Aritmetické výrazy

<b>9. postfix</b>
Vyhodnoťte aritmetický výraz $7\ 4\ *\ 9\ 2\ 4\ -\ *\ 6\ /\ +$ . Prepíšte ho do prefixu a infixu.
<b>10. prefix</b>
Vyhodnoťte aritmetický výraz $\ +\ *\ -\ +\ 1\ 2\ 3\ 4\ *\ 5\ -\ 6\ +\ 7\ 8$ . Prepíšte ho do postfixu a infixu.
<b>11. infix</b>
Vyhodnoťte aritmetický výraz $3*(4+5)-6*7/(8+6)$ . Prepíšte ho do prefixu a postfixu.