typ Char

Tento typ slúži na uchovanie jedného znaku (napr. písmena, číslice alebo nejakého symbolu). Premenná tohto typu zaberá 1 bajt (8 bitov). Môžeme sa o tom presvedčiť SizeOf(Char) = 1. Do jedného bajtu sa vojde vždy iba jeden z 256 rôznych znakov. Znakové konštanty zapisujeme medzi dva apostrofy, pričom znakovú konštantu apostrof zapíšeme dvakrát. Napr. konštanty:
`'a', 'A', ' ', '7', ', '+', ...`
Vnútorne sú znaky reprezentované číslami od 0 do 255, pričom sa používa tzv. ASCII kódovanie. Podľa tohto kódovania, napr.

- 'a' má kód 97,
- 'A' má kód 65,
- '0' má kód 48,
- ' ' má kód 32,

Konštanty môžeme zapisovať aj inak ako symbolom medzi apostrofmi. Za znak # uvedieme ASCII kód, napr.

- #66 je to isté ako 'B'
- #49 je to isté ako'1'

So znakmi nie sú v pascale definované žiadne operácie (také ako súčet alebo súčin s číslami). Ale keďže znakový typ je tiež ***ordinálny***, platia tieto vlastnosti:

- funkcia **Ord()** vráti ordinálnu hodnotu, t.j. ASCII kód znaku, napr.
  - **Ord('C')** vráti 67
- funkcia **Char()** z čísla <0, 255> vyrobí znak, napr.
  - **Char(50)** vráti znak '2'

Znaky môžeme navzájom porovnávať, pričom platí, že nejaký znak je menší ako nejaký iný vtedy, keď ASCII kód prvého je menší ako ASCII kód druhého. Platí

- ' ' < ... < '0' < '1' < '2' < ... < '9' < ... < 'A' < ' B' < ... < 'a' < 'b' < ...

Ukážme malý príklad použitia vo for-cykle:

```
var
  Z: Char;

...
  for Z := 'A' to 'Z' do
    Image1.Canvas.TextOut(x,y, Z + '=' + IntToStr(Ord(Z)));
```

Znakové reťazce

Znakový reťazec je údajový typ, ktorý obsahuje nejakú ***postupnosť znakov*** (**Char**). Je to vlastne postupnosť znakových premenných, s ktorými môžeme pracovať ako s celkom a môžeme ich aj jednoducho modifikovať. Naproti tomu v súbore sú údaje najčastejšie uložené na disku a ich zmena je algoritmicky trochu náročnejšia.
Znaky v znakovom reťazci (t.j. v postupnosti znakov) sú očíslované od 1 až po momentálnu dĺžku reťazca. Túto dĺžku zistíme pomocou štandardnej funkcie **Length**. FreePascal si uchováva dĺžku v 4 bajtoch, t.j. teoreticky by maximálna dĺžka mohla byť 4 gigabajty - v skutočnosti je obmedzená možnosťami Windows: možno len 1 gigabajt - otestujte to na vašom počítači.
Premennú typu znakový reťazec deklarujeme takto:

```
var Retazec: string;
```

Premenná **Retazec** môže mať zatiaľ nedefinovanú hodnotu a preto jej musíme niečo priradiť, napr. reťazcovú konštantu:

```
Retazec := 'reťazec';
```

Premenná **Retazec** teraz obsahuje postupnosť znakov dĺžky 7: prvý znak je 'r', druhý 'e', atď. T.j. hodnota funkcie **Length(Retazec)** je teraz 7. Reťazcové konštanty sú uzavreté rovnako ako znakové konštanty v apostrofoch. Premenná typu **string** môže obsahovať aj prázdny reťazec:

```
Retazec := '';
```

Vtedy je jej dĺžka 0.
Môžeme pracovať aj s jednotlivými prvkami postupnosti, t.j. s jednotlivými znakmi v reťazci. Napr.

```
Retazec := 'abcdef';
Retazec[3] := '*';
```

Najprv sme do **Retazec** priradili 5-znakový reťazec a potom sme v ňom zamenili 3-tí znak (znak 'c') za hviezdičku '*'.
Nesmieme sa pritom odvolávať na znaky mimo rozsahu <1, momentálna dĺžka>, napr.

```
Retazec[10] := '+'; // reťazec má zatiaľ dĺžku len 6
```

spôsobí chybovú správu.
Znakové reťazce môžeme porovnávať pomocou relačných operácií (=, <>, <, >, <=, >=). Reťazce sú usporiadané pomocou tzv. **lexikografického usporiadania**. Napr. platí

```
'abc' > 'ABC'
'Adam' < 'Eva'
'jana' < 'jano'
'Jana' < 'jana'
```

Postupne sa pri tom porovnáva znak za znakom: kým sú v oboch reťazcoch rovnaké, pokračuje sa v porovnávaní; keď sa narazí na rozdielne znaky, tak sa tieto dva porovnajú navzájom a podľa toho sa nastaví celkový výsledok porovnania.
Porovnávanie dvoch znakov sa robí podľa pravidiel **Char**: t.j. menší je ten znak, ktorý má menší **ASCII**-kód.
Operácia '+' slúži na zreťazenie reťazcov (hovoríme, že + je polymorfný operátor, lebo jeho funkčnosť závisí od typu operandov: iný význam má pre čísla a iný pre reťazce). Napr.

```
Retazec := 'abc'+'def';      // Retazec='abcdef'
Retazec := '';
for I := 1 to 10 do         // Retazec='**********'
  Retazec := Retazec + '*';
```

```
var
  Retazec: string;
begin
  Retazec := 'abcd'+'efgh'+'ijkl'+'mnop';
```

Podprogramy s reťazcami

FreePascal ponúka sadu preddefinovaných (štandardných) podprogramov, ktoré pracujú s reťazcami. Medzi základné patria

- **Length**(*reťazec*)
  - táto funkcia vráti momentálnu dĺžku reťazca
- **Delete**(*reťazec*, *od*, *koľko*)
  - táto procedúra zmaže z daného reťazca podreťazec od príslušného indexu danej dĺžky

```
Retazec := 'abrakadabra'; Delete(Retazec, 4, 6); // Retazec='abrra'
Retazec := 'abrakadabra'; Delete(Retazec, 6, 20); // Retazec='abrak'
```

- **Insert**(*podreťazec,* reťazec, *index*)
  - táto procedúra slúži na vkladanie podreťazca do reťazcovej premennej od nejakého konkrétneho indexu

```
Retazec := 'abrakadabra'; Insert('a',Retazec,2); // Retazec=' aabrakadabra '
Retazec := 'abrakadabra'; Insert('xy',Retazec, 6); // Retazec='abrakxyadabra'
```

- **Copy**(*reťazec*, *od*, *koľko*)
  - táto funkcia vráti nový reťazec, ktorý je podreťazcom pôvodného
  - z pôvodného reťazca sa zoberú znaky počínajúc od hodnoty ***od*** a zoberie sa maximálne ***koľko*** znakov
  - ak je ***od*** mimo rozsahu indexov reťazca, funkcia vráti prázdny reťazec, t.j. '', napr.

```
Retazec := Copy('abcde', 2, 3); // Retazec='bcd'
Retazec := Copy('abcde', 7, 3); // Retazec=''
Retazec := Copy('abcde', 3, 5); // Retazec='cde'
```

  - ak potrebujeme podreťazec od nejakého indexu až do konca, nemusíme vypočítať presný počet, ale môžeme použiť konštantu **MaxInt**, napr.

Retazec := Copy('abcde', 3, **MaxInt**);

- **Pos**(*podreťazec*, *reťazec*)
  - táto funkcia vráti začiatočný index **prvého** výskytu podreťazca v danom reťazci alebo **0**, ak sa v reťazci podreťazec nenachádza, napr.

```
I := Pos('d', 'abcde'); // I=4
I := Pos('C', 'abcde'); // I=0
I := Pos('ba', 'abababab'); // I=2
```

Úlohy:

1. Načítajte nejaký reťazec od používateľa a napíšte jeho dĺžku
2. Načítajte nejaký reťazec od používateľa a vypíšte na obrazovku písmenká pod seba
3. Načítajte nejaký reťazec od používateľa, potom ešte jeden znak (char) a napíšte, ANO, alebo NIE, ak sa znak v reťazci nachádza, alebo nie.
4. Spočítaj počet slov v zadanej vete.
5. Napíšte program, ktorý z daného reťazca vynechá všetky samohlásky.
6. V zadanomn reťazci zistite, koľko je v ňom čísel, samohlások a spoluhlások.
7. V zadanom reťazci je prehodené y a z. Oprav.
8. Napíšte program, ktorý načíta od používateľa maximálne 12 – znakový reťazec a na obrazovku vypíše:

```
r            c
re           ec
ret          zec
reta         azec
retaz        tazec
retaze       etazec
retazec      retazec
retaze       etazec
retaz        tazec
reta         azec
ret          zec
re           ec
r            c
```

9. Na náhrdelníku sú navlečené korálky v tvare písmen. Vytvorte podobný náhrdelník, kde budú všetky korálky v tvare písmena A nahradené tromi guličkami v tvare hviezdičky.
10. Rodičia si zapisujú džavotanie svojho dieťaťa, ktoré začína rozprávať. Napíšte program, ktorý zistí, koľkokrát sa rodičmi zadaná slabika vyskytuje v celodennom zázname.
11. Pokúste sa naprogramovať vlastné procedúry zmaž znak z reťazca, vlož znak do reťazca na konkrétnu pozíciu, zisti pozíciu znaku v reťazci.
12. Vytvorte program, ktorý zašifruje alebo odšifruje zadaný text. Šifra je jednoduchá: za každý znak pôvodnej správy sa vloží náhodné písmeno.
13. Napíšte program, v ktorom sa bude šifrovať vložením konštantnej slabiky za každý znak pôvodnej správy.
14. Napíšte program, ktorý bude kódovať správy morzeovkou. Jednotlivé písmená morzeovky sú oddelené znakom /.
15. Napíšte program, ktorý vytvorí slová zo zadaného reťazca prešmyčkou napr. delo, lode, dole. (aj nezmyselné).
16. Zisti, či reťazec je palindróm. (číta sa rovnako spredu aj zozadu) napr. **jelenovi pivo nelej.**
    a) s medzerami
    b) bez medzier