Programování

Programování s řetězci

There are
10 types
of people
in the world:

Those who
understand binary
and those
who don't.

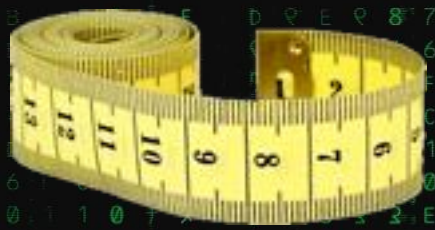
Martin Urza

Motivace

- Programování s textovými řetězci patří k nezbytným součástem dovednosti tvorby programů, bez které se prakticky nedá stvořit žádná rozumná aplikace.
- Programů, které pracují s textem (byť okrajově) je v praxi mnohem víc, než programů, které například něco velkolepého počítají.
- Řetězcové operace (minimálně ty základní) využijete např. při každém rozumnějším formátování ne zcela triviálního výstupu.
- V Pascalu (ale i v Delphi) jsou operace s řetězci velmi pohodlně vyřešeny (zejména třeba oproti čistému C), pro programátory je práce s nimi opravdu snadná.

Typ string

- String je pole znaků indexované od 0 do 255.
  - type string = array [0..255] of char;
- V případě potřeby lze vyrobit řetězec i kratší; jeho délku uveďte do hranaté závorky (např. string[32]).
  - type string[*n*] = array [0..*n*] of char;
- Přiřadíte-li textový řetězec do proměnné typu string, vyplní tuto proměnnou po znacích od indexu 1 dál.
  - Je-li s proměnná typu string a přiřadíme-li do ní text 'ahoj', pak s[1] = 'a', s[2] = 'h', s[3] = 'o' a s[4] = 'j'.
- Na indexu 0 je uložena délka řetězce. Řetězec délky n obsahuje na indexu 0 znak, který je n-tý v ASCII.
  - Délka řetězce se nastavuje automaticky.



Délka řetězců

- Jak bylo uvedeno na minulém slajdu, délka řetězce je uložena ve znaku na nultém indexu řetězce.
- Typ char je ordinální; má jednoznačné uspořádání.
  - Pořadí znaku v tomto uspořádání (v Pascalu shodou okolností ASCII) lze zjistit pomocí funkce ord, jež má jediný parametr typu char; vrací hodnotu typu integer.
    - Je-li s string a přiřadíme-li mu 'ahoj', pak ord(s[0]) = 4.
- Když změníte obsah řetězce, znak na nultém indexu se sám modifikuje tak, aby délka odpovídala (o to se nemusí programátor nijak starat).
- Délku řetězce lze zjistit i funkcí length, jež má jediný parametr typu string a vrací jedno délku.

Cvičení

- Napište program, který přečte ze vstupu nějaký text (slovo, větu, číslo, shluk znaků, ...) a ten pak vypíše na výstup. Délka textu bude menší než 256 znaků.

```
program readAndWrite;

var
  word : string;

begin
  writeln('Napiste nejaky text:');
  readln(word);
  writeln();

  writeln('Napsal jste:');
  writeln(word);
  writeln();

  writeln('Pro ukonceni programu stisknete ENTER.');
  readln();
end.
```

- ** Upravte program, aby daný text vypsal n-krát, kde n je délka onoho textu ve znacích.*
- *** Modifikujte program tak, aby před každý z výpisů textu napsal jeho pořadové číslo. Navíc ať program nevypisuje žádný text více než 8x.*

Přístup k jednotlivým znakům řetězce

- Jak již bylo řečeno, řetězec (string) je pole znaků. To znamená, že se k němu tak může programátor ve všech ohledech chovat.
- Potřebujeme-li přistupovat k jednotlivým znakům řetězce, lze toho (pro čtení i zápis) docílit pomocí indexů znaků v poli.
- Je-li s proměnná typu string, pak je *n*-tý znak textu v proměnné přístupný přes identifikátor s[*n*].
  - Chcete-li tento znak změnit, přiřaďte do něj jiný:
    - s[*n*] := 'x';
  - Podobně lze ten znak přiřadit do proměnné typu char:
    - c := s[*n*];

Cvičení

- Napište program, který přečte text ze vstupu, nahradí každý druhý znak písmenem 'x' a pak vše vypíše na výstup.
- ** Upravte program, aby každý sudý znak nenahrazoval 'x', ale znakem, který je v řetězci před ním, např. 'Jadzia Dax' změní na 'JJddii aa'.*
- *** Modifikujte předchozí program tak, aby upravený text (tedy ten se zduplikovanými znaky z minulého příkladu) vypsal pozpátku.*
  Hint: Řetězec nevypisujte jako řetězec, ale po znacích; využijte cyklu.

```
program modifyChars;

var
  word  : string;
  loop1 : integer;

begin
  writeln('Zadejte text:');
  readln(word);

  for loop1 := 1 to length(word)
    do
    if (0 = (loop1 mod 2))
      then word[loop1] := 'x';

  writeln(word);

  readln();
end.
```

Stringové operátory

- Jazyk Pascal umožňuje velmi snadnou manipulaci s řetězci; operátory fungují velmi intuitivně.
  - Operátor zřetězení je plus (+). Spojí dva řetězce do jednoho. Např. str := str + ' ' + str; přidá k řetězci str mezeru a za ni str znovu zkopíruje (bez mezery).
    - 'Seven ' + 'of ' + 'Nine' = 'Seven of Nine'
  - Stringy lze porovnávat logickými operátory = a <> jako čísla a funguje to intuitivně (stejné stringy se rovnají).
    - Toto opět v mnohých jazycích intuitivně nefunguje.
  - Operátory < a > fungují na řetězce taky. Zjednodušeně lze říci, že řetězec „je menší“ než jiný, pokud by se při zařazení řetězců do telefonního seznamu nacházel blíže začátku (tedy „větší“ jsou dále v abecedě).

Funkce copy

- function copy (str : string; idx, cnt : integer) : string;
  - Vrátí řetězec, který vznikne tak, že se z kopíruje cnt znaků z řetězce str počínaje znakem na pozici idx.
  - Například copy('Seven of Nine', 2, 9) = 'even of N'.
- Tuto funkci lze dobře kombinovat s funkcí length.
- Funkci copy lze použít i k odmazání několika znaků.
  - word := copy(word, 3, length(word) - 2);
    - Smaže dva první znaky. (Je jasné proč?)
- Nesmyslné hodnoty nezpůsobí pád programu (což je oproti mnohým jiným jazykům luxus).
  - copy('The Doctor', 5, 128) = 'Doctor'
  - copy('Kathryn Janeway', -2, 7) = 'Kathryn'

Procedura delete

- procedure delete (var str : string; idx, cnt : integer);
  - Procedura smaže z řetězce str odpovídající část, která začíná na indexu idx a je dlouhá cnt znaků.
  - Proměnná str je proceduře předávána odkazem, aby mohla být změněna; parametr nemůže být konstanta!
  - Je-li v proměnné name uloženo 'T Pol', pak po zavolání delete(name, 2, 2) v name zbyde 'Tol'.
  - Pochopitelně nelze zavolat delete('T Pol', 2, 2), protože konstantu není možné předat odkazem.
- Ani procedura delete nezpůsobí pád programu při zadání nesmyslných parametrů.
- Může být nahrazena funkcí copy (viz. minulý slajd).

Procedura insert

- pr. insert (src : string; var dst : string; idx : integer);
  - Procedura vloží (celý) řetězec src do řetězce dst tak, aby nově vložená část začínala na pozici idx.
  - Do src procedura nezasahuje, do dst ano, proto je src předáván hodnotou a dst odkazem (viz. minulý slajd).
  - Je-li v proměnné name 'Seven Nine', pak po volání insert('of ', name, 7) bude v name 'Seven of Nine'.
- Procedura insert opět není nezbytná a lze ji nahradit pomocí funkce copy a operátoru zřetězení, kterým je možno spojovat jednotlivé návratové hodnoty funkcí copy.
- Ani procedura insert nezpůsobuje pády programu.

Funkce pos

- function pos (substr, str : string) : integer;
  - Funkce pos vrací pozici podřetězce substr v řetězci str; nenachází-li se substr ve str, vrací nulu. Vyskytuje-li se substr ve str víckrát, najde první výskyt.
  - Příklady:
    - pos('Dax', 'Jadzia Dax') = 8
    - pos('a', 'Jadzia Dax') = 2
    - pos('xxx', 'Jadzia Dax') = 0
    - pos('j', 'Jadzia Dax') = 0
  - Oba řetězce jsou předávány hodnotou, proto lze hledat i konstanty (či v konstantách).
- Tato funkce je velmi vhodná k parsování řetězců.

Další funkce

- Funkce concat má alespoň jeden (ale může jich mít mnoho) parametr typu string. Tyto řetězce spojí do jednoho a vrátí výsledek.
  - Stejného efektu se dá dosáhnout pomocí operátoru +, ale ten v některých verzích Pascalu nefunguje.
- Funkce lowercase změní všechna velká písmena v řetězci na malá, např. lowercase('BoRg') = 'borg'.
- Funkce upcase změní všechna malá písmena řetězce na velká, např. upcase('bOrG') = 'BORG'.
- Chcete-li porovnávat řetězce case-insensitive, oba změňte na lowercase/upcase a porovnejte. Totéž lze praktikovat při používání funkce pos.

Náročnost řetězcových operací

- Budete-li psát větší programy, které pracují s textem (obecně s řetězci), je třeba mít na paměti, že textové operace jsou výrazně časově i prostorově náročnější, než operace s čísly.
- Klasická proměnná typu integer v Pascalu zabírá jen dva bajty paměti. Řetězec potřebuje jeden bajt na každý znak (je výrazně větší). Funkce typu pos musí celý řetězec po znaku procházet.
- Sečtení dvou čísel typicky procesor zvládne v jednom tiku (taktu). Při sčítání řetězců jsou data kopírována z místa na místo; to trvá dlouho, jsou-li řetězce moc dlouhé.

Cvičení

- Napište program, který na vstupu dostane číslo a na výstup vrátí stejné číslo, avšak oddělené čárkami mezi každou trojicí cifer; existuje-li skupina méně než tří cifer, pak musí být tato skupina první.
- ** Upravte program tak, aby místo dělících čárek vkládal znaky udávající řád: tisíce jako k, miliony M, miliardy G, biliony T a dále P, E, Z, Y a pak už jen čárky. Například z čísla 4194304 udělá 4M194k304.*

```
program modifyChars;

var
  numIn  : string;
  numOut : string;

function max(a, b : integer) : integer;
begin
  if (a > b)
    then max := a
    else max := b;
end;

begin
  readln(numIn);
  while (0 < length(numIn))
    do
    begin
      numOut :=
        copy(numIn, length(numIn) - 2, 3) + numOut;
      delete(numIn, max(1, length(numIn) - 2), 3);
      if (0 < length(numIn))
        then numOut := concat(',', numOut);
    end;
  writeln(numOut);
end.
```

Rekapitulace

- Operace s řetězci patří k nejdůležitějším aspektům programování; drtivá většina aplikací nějak rozumně formátuje vstup (a k tomu využívá tyto operace).
- Řetězce jsou v Pascalu reprezentovány poli znaků, která mají na nultém indexu uloženou informaci o délce řetězce (nikoliv číselnou, nýbrž jako znak).
- Základní funkce (vestavěné v Pascalu) pro práci s řetězci jsou copy, delete, insert, pos, length, concat, lowercase a upcase (typicky jsou nahraditelné).
- Úkony s řetězci jsou většinou výrazně náročnější (na výkon procesoru i spotřebu paměti) než operace s čísly (což na malých datech nevadí).