

8. Datový typ Pole

Tento datový typ lze použít všude tam, kde vystupují indexované proměnné. V matematice se jedná především o vektory, posloupnosti, matice apod.

Příklad 1: Tisk ASCII tabulky Win codepage 852. Vraťme se nyní k výše slíbenému příkladu na tisk ASCII tabulky znaků. Uživatelské rozhraní bude tvořit jen objekt, do kterého se tabulka vytiskne. Tímto objektem je StringGrid („mřížka“ či tabulka, která je schopna zobrazovat řetězce). Tento objekt najdeme v liště **Additional**. Je to obecně dvojrozměrná tabulka, jejíž rozměry určují vlastnosti **FixedCols** resp **FixedRows**, tj. počet pevných sloupců resp. řádků (zde oba parametry nastavíme na jedničku) a dále **ColCount** resp. **RowCount**, tj. počet sloupců resp. řádků. Zde nastavíme celkem jedenáct sloupců (deset plus jeden pevný – ColCount=11) a dvacet sedm řádků (dvacet šest plus jeden pevný – RowCount=27). Obsah polí je určen maticí **Cells**, jejíž prvky jsou **indexovány od nuly**, a to v pořadí **sloupec-řádek** (tedy naopak, než jsme zvyklí u matic). V Object Inspectoru můžeme opět měnit nejružnější vlastnosti této mřížky: Rozměry polí pomocí proměnných **DefaultColWidth** resp. **DefaultRowHeight**. Program sestavíme tak, že ve sloupcích tabulky se ASCII kód bude zvyšovat po jedné, v řádcích pak po deseti. Proceduru, která tabulku vyplní, můžeme spojit s událostí `onCreate` formuláře. Jediným úkolem takového programu je zobrazení tabulky, které se provede automaticky po spuštění programu.

```
procedure TForm1.FormCreate(Sender: TObject);  
var i,j:Byte;  
begin                                     {vyplnění pevného sloupce}  
    for i:=0 to 25 do StringGrid1.Cells[0,succ(i)]:=IntToStr(10*i);  
                                     {vyplnění pevného řádku}  
    for j:=0 to 9 do StringGrid1.Cells[succ(j),0]:=IntToStr(j);  
    for i:=0 to 24 do                   {vyplnění čtyřadvaceti „kompletních“ řádků}  
        for j:=0 to 9 do                 {funkce Chr vrací znak dle argumentu aktuální ASCII tabulky}  
            StringGrid1.Cells[succ(j),succ(i)]:=Chr(10*i+j);  
        for j:=0 to 5 do                 {vyplnění posledního „nekompletního“ řádku}  
            StringGrid1.Cells[succ(j),26]:=Chr(250+j);  
end;
```

[Zdrojový kód](#)

[Spustitelný kód](#)

Tímto způsobem ovšem můžeme vytisknout pouze ASCII tabulku znakové sady, kterou máme implicitně nastavenou. Používáme-li operační systém Windows, pak je to pravděpodobně Win codepage 852. Řada českých mutací nejružnějšího software však používá kódování jiná. Abychom mohli pracovat s jiným kódováním, je třeba především znát jeho ASCII tabulku.

Jak takovou tabulku vytisknout? Je třeba především porovnat obě tabulky a najít prostředek, jak standardní ASCII Win codepage 852 předefinovat. ASCII tabulka je v každém případě uspořádaná posloupnost 255 hodnot. Pro tabulku Latin2 tedy vyhradíme proměnnou Latin typu `array[0..255] of byte`; do které je třeba implicitní tabulku „přeskládat“. Podívejme se např. na znak „ů“. Ve Win 852 je ASCII(ů) = 249, v Latin2 je ASCII(ů) = 133. Pro tento znak je tedy třeba položit Latin[133]:=249. Podobně je třeba postupovat u všech znaků ASCII tabulky. Kompletní kód pro tisk tabulky Latin2 tedy vypadá následovně:

ASCII tabulka stránky Win codepage 852										
	0	1	2	3	4	5	6	7	8	9
0										
10										
20										
30				!	"	#	\$	%	&	'
40	()	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[\]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{		}	~			
130	,	.	„	…	†	‡	‰	Š	Š	Š
140	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š
150	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š
160	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š
170	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š
180	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š
190	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š
200	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š
210	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š
220	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š
230	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š
240	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š
250	Š	Š	Š	Š	Š	Š	Š	Š	Š	Š

ASCII tabulka - kódování LATIN2										
	0	1	2	3	4	5	6	7	8	9
0										
10										
20										
30				!	"	#	\$	%	&	'
40										
50										
60										
70										
80										
90										
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{		}	~		Ç	ü
130	é	â	ä	û	ó	ç	í	ê	Ö	š
140	î	ó	Ä	Ć	É	Ł	ŕ	ô	Ł	Ł
150	Š	š	Ö	Ü	Ť	ť	Ł	×	č	č
160	á	í	ó	ú	Ą	ą	Ż	ż	Ę	ę
170	ż	Č	š	«	»	„	”	”	”	”
180	Á	Ă	Ė	Š	•	”	•	”	Ž	Ž
190	ž	μ	™	¶	•	”	”	”	Ā	ā
200	…	ı	-	-	•	”	”	”	đ	Đ
210	Đ	đ	Ń	Í	İ	ě	”	”	”	•
220	•	ı	Ů	™	Ó	ß	Ô	Ň	ń	ň
230	Š	š	Ř	Ú	ř	Ú	ý	Ý	ť	’
240	•	”	•	•	š	÷	•	•	”	”
250	•	•	•	•	•	•	•	•	•	•

Příklad 2: Tisk ASCII tabulky Latin2. Tato tabulka je využívána velmi často. Prvních 127 znaků je, jak již bylo řečeno, standardních, rozmístění znaků s ASCII 128 – 255 je zřejmé z připojeného obrázku.

procedure TForm1.ASCII_Tabulka_LATIN2(Sender: TObject);

var i,j :byte;

Latin :array[0..255] of byte;

begin

for i:=0 to 127 do Latin[i]:=i; {prvních 127 znaků se nemění}

Latin[128]:=199;	Latin[129]:=252;	Latin[130]:=233;	Latin[131]:=226;
Latin[132]:=228;	Latin[133]:=249;	Latin[134]:=230;	Latin[135]:=231;
Latin[136]:=179;	Latin[137]:=235;	Latin[138]:=213;	Latin[139]:=245;
Latin[140]:=238;	Latin[141]:=243;	Latin[142]:=196;	Latin[143]:=198;
Latin[144]:=201;	Latin[145]:=197;	Latin[146]:=190;	Latin[147]:=244;
Latin[148]:=246;	Latin[149]:=188;	Latin[150]:=144;	Latin[151]:=140;
Latin[152]:=156;	Latin[153]:=214;	Latin[154]:=220;	Latin[155]:=141;
Latin[156]:=157;	Latin[157]:=163;	Latin[158]:=215;	Latin[159]:=232;

Latin[160]:=225;	Latin[161]:=237;	Latin[162]:=243;	Latin[163]:=250;
Latin[164]:=165;	Latin[165]:=185;	Latin[166]:=142;	Latin[167]:=158;
Latin[168]:=202;	Latin[169]:=234;	Latin[170]:=160;	Latin[171]:=159;
Latin[172]:=200;	Latin[173]:=186;	Latin[174]:=171;	Latin[175]:=187;

Latin[176]:=128;	Latin[177]:=129;	Latin[178]:=131;	Latin[179]:=136;
Latin[180]:=144;	Latin[181]:=193;	Latin[182]:=195;	Latin[183]:=204;
Latin[184]:=170;	Latin[185]:=149;	Latin[186]:=189;	Latin[187]:=183;

```

Latin[188]:=132;   Latin[189]:=143;   Latin[190]:=159;   Latin[191]:=181;
Latin[192]:=153;   Latin[193]:=182;   Latin[194]:=176;   Latin[195]:=172;
Latin[196]:=168;   Latin[197]:=166;   Latin[198]:=195;   Latin[199]:=227;
Latin[200]:=133;   Latin[201]:=152;   Latin[202]:=151;   Latin[203]:=150;
Latin[204]:=149;   Latin[205]:=146;   Latin[206]:=145;   Latin[207]:=164;

Latin[208]:=239;   Latin[209]:=208;   Latin[210]:=207;   Latin[211]:=240;
Latin[212]:=210;   Latin[213]:=205;   Latin[214]:=206;   Latin[215]:=236;
Latin[216]:=146;   Latin[217]:=147;   Latin[218]:=148;   Latin[219]:=149;
Latin[220]:=150;   Latin[221]:=222;   Latin[222]:=217;   Latin[223]:=153;

Latin[224]:=211;   Latin[225]:=223;   Latin[226]:=212;   Latin[227]:=209;
Latin[228]:=241;   Latin[229]:=242;   Latin[230]:=138;   Latin[231]:=154;
Latin[232]:=192;   Latin[233]:=218;   Latin[234]:=224;   Latin[235]:=218;
Latin[236]:=253;   Latin[237]:=221;   Latin[238]:=254;   Latin[239]:=146;

Latin[240]:=151;   Latin[241]:=189;   Latin[242]:=178;   Latin[243]:=161;
Latin[244]:=162;   Latin[245]:=167;   Latin[246]:=247;   Latin[247]:=184;
Latin[248]:=149;   Latin[249]:=168;   Latin[250]:=183;   Latin[251]:=251;
Latin[252]:=216;   Latin[253]:=248;   Latin[254]:=255;   Latin[255]:=160;

```

```

for i:=0 to 25 do StringGrid1.Cells[0,succ(i)]:=IntToStr(10*i);
  for j:=0 to 9 do StringGrid1.Cells[succ(j),0]:=IntToStr(j);
for i:=0 to 24 do for j:=0 to 9 do
  StringGrid1.Cells[succ(j),succ(i)]:=Chr(Latin[10*i+j]);
for j:=0 to 5 do StringGrid1.Cells[succ(j),26]:=Chr(Latin[250+j]);
end;

```

Tento program sám o sobě příliš velký význam nemá. Do zdrojového kódu jsme v podstatě jen pracně přepsali jakousi tabulku, kterou jsme museli mít někde vytištěnou a program nám ji pouze zreprodukoval. Toto překódování nicméně později využijeme při automatickém rozpoznávání kódování češtiny, což je věc, kterou dosud žádný komerční software neumí. K tomu ovšem potřebujeme znalosti týkající se práce s externími soubory, kterým se budeme věnovat později. Pro zájemce:

Automatické rozpoznávání kódování češtiny: [Zdrojový kód](#) [Spustitelný kód](#)

V tuto chvíli se tedy soustředíme na další práci s polem dat, při níž tyto znalosti prozatím potřebovat nebudeme.

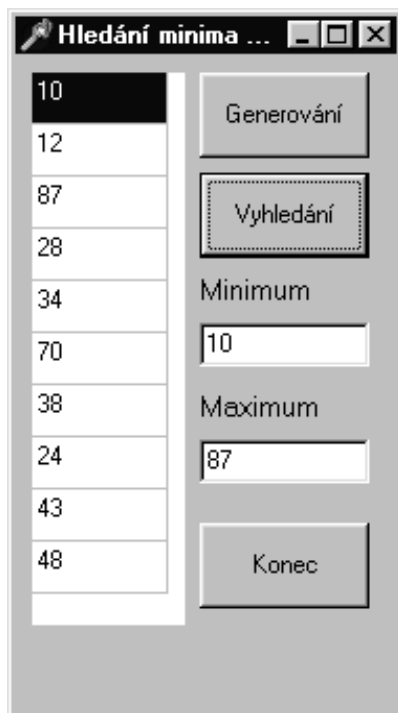
Příklad 3: Vyhledání minimálního a maximálního prvku v posloupnosti čísel: Nejdříve sestavíme formulář dle připojeného obrázku. Pole, které se má prohledávat, budeme opět zapisovat do StringGrid. počet pevných sloupců resp. řádků Počet pevných řádků a sloupců tentokrát nastavíme na nulu a celkový počet sloupců (**ColCount**) na jedničku a celkový počet řádků (**RowCount**) na deset. Náš program bude fungovat tak, že na stisk tlačítka s titulkem Generování vygeneruje posloupnost náhodných dvojčiferných čísel a na stisk Vyhledávání vyhledá minimum a maximum. Tlačítko s titulkem generování je pak třeba OnClick spojit s procedurou Generovani a tlačítko s titulkem Vyhledávání pak OnClick s procedurou Vyhledavani. Celý kód pak může vypadat následovně:

```
implementation
{$R *.DFM}
```

{následující proměnné budou používány v celém programu, je třeba je deklarovat globálně}

```
Const Pocet = 10;                                     {počet prvků v posloupnosti}
var   Prvek:array [1..Pocet] of Byte;                  {deklarace pole}
```

```
                                     {generování posloupnosti dvojčiferných náhodných čísel}
procedure TForm1.Generovani(Sender: TObject);
var i :Integer;
begin
  For i:=1 to Pocet do
  begin
    Prvek[i]:=Trunc(Random*90+10);{funkce Random vrací náhodné číslo z intervalu <0;1)}
    StringGrid1.Cells[0,pred(i)]:=IntToStr(Prvek[i]);
  end;
end;
```



```
procedure TForm1.Vyhledani(Sender: TObject);
var Max,Min,i :Byte;
    PrevodRetezec :String;
```

```
begin
  Max:=Prvek[1];Min:=Max;
  For i:=2 to Pocet do
    if Prvek[i]>Max then Max:=Prvek[i]
    else if Prvek[i]<Min then
      EditMin.Text:= IntToStr(Min);
      EditMax.Text:= IntToStr(Max);
  end;
```

Tento příklad je doplněn algoritmem na hledání zadané hodnoty v zadaném poli:

[Zdrojový kód](#)

[Spustitelný kód](#)

Příklad 4: Uspořádání číselné posloupnosti. Číselnou posloupnost bychom mohli uspořádat metodou opakovaného hledání maxima popř. minima, tedy opakováním postupu z předchozího příkladu. Tato metoda je však značně neefektivní a pracná. Ukážeme zde metodu bublinovou, která je často používána, i když ani ona nepatří k nejefektivnějším:

```
procedure TForm1.Generovani(Sender: TObject);
var i :Integer;
begin
  StringGrid1.Cells[1,0]:='Původní';
  For i:=1 to Pocet do
```

Bublinová metoda

Posloupnost	Původní	Uspořádaná
1	100	100
2	128	128
3	874	153
4	282	163
5	345	173
6	704	245
7	386	282
8	245	345
9	435	386
10	483	435
11	173	483
12	527	527
13	163	704
14	856	856
15	153	874

Generování Uspořádání

Konec

```

begin
  Prvek[i]:=Trunc(Random*900+100);
  StringGrid1.Cells[0,i]:=IntToStr(i);
  StringGrid1.Cells[1,i]:=IntToStr(Prvek[i]);
end;
end;

procedure TForm1.BublinovaMetoda
(Sender: TObject);

var i,Trezor:Integer;
    Zmena:Boolean;
begin
  StringGrid1.Cells[2,0]:='Uspořádaná';
  Repeat
    Zmena:=False;
    For i:=1 to Pocet-1 do
      if Prvek[i]>Prvek[succ(i)] then
        begin
          Trezor:=Prvek[i];
          Prvek[i]:=Prvek[Succ(i)];
          Prvek[Succ(i)]:=Trezor;
          zmena:=True;
        end;
    until not Zmena;

    For i:=1 to Pocet do
      StringGrid1.Cells[2,i]:=IntToStr(Prvek[i]);

end;

```

Příklad 5 – hledání prvočísel: Program má za úkol najít všechna prvočísla menší než zadané n. Nejdříve budeme postupovat tzv. metodou postupného dělení. Uživatelská funkce, která touto metodou testuje, zda dané číslo je prvočíslo či nikoli, vypadá následovně:

```

Function Prvocislo(Cislo:DWord):Boolean;
var i:DWord;
begin
  i:=1;
  Repeat
    inc(i);
  Until (Cislo mod i = 0) or (i>Trunc(sqrt(Cislo)));
  if Cislo mod i = 0 then Prvocislo:=False else Prvocislo:=True;
end;

```

Touto funkcí je pak potřeba testovat všechna čísla, která jsou menší nebo rovna danému n:

```

Procedure TForm1.Hledej_Prvocisla(Sender:TObject);
const PocetRadku=50;PocetSloupcu=50;
      {pro StringGrid, do kterého budeme vypisovat nalezená prvočísla}
var DolniMez,HorniMez,i:LongInt;
    PrevodRetezec:String;

```

```

begin
    {načtení mezí z edit okének - typ TEdit, jejichž vlastnost Name byla změněna}
    Val(EditDolniMez.Text,DolniMez,ErrorReport);           {na EditDolniMez}
    Val(EditHorniMez.Text,HorniMez,ErrorReport);           {resp. EditHorniMez}
    for i:=DolniMez to HorniMez do                         {procházej zadany interval}
    if Prvocislo(i) then                                    {najdeš-li prvočíslo}
    begin
        Str(i, PrevodRetezec);    {převeď ho na řetězce vyplň jím příslušnou buňku tabulky}
        StringGrid1.Cells[i div PocetRadku, j div PocetSloupcu]:=PrevodRetezec
    end;
end;

```

Elegantní metoda hledání prvočísel je tzv. metoda Eratosthénova síta. V prvním kroku jsou všechna čísla větší než jedna a menší nebo rovna daném n označena jako prvočísla. Pak je posloupnost procházena postupně od dvojky trojky atd. Je-li procházeným číslem prvočíslo, jsou v posloupnosti všechny jeho násobky vyškrtány, číslo samo ponecháno. Po testu posledního čísla zůstanou jen prvočísla (uvádíme opět jen rozhodující fragment kódu):

```

begin
    for i:=1 to Max do Prvocislo[i]:=True;{hledáme prvočísla v posloupnosti 2..Max}
                                         {Typ proměnná Prvocislo je array [1..Max] of boolean}
    for i:=2 to Max do                   {přičemž Prvocislo[i]:=True ⇔ i=prvočíslo}
    begin
        j:=2;
        while i*j<=Max do
            begin Prvocislo[i*j]:=False; inc(j);end;    {"vyškrtávání" j-násobků čísla i}
        end;
    .....                                     {výpis výsledků analogicky předchozímu příkladu}
end;

```

Poznámka: Tento algoritmus pochází od Eratosthena z Kyrény, * kolem 275 př. n. l., † 195 př. n. l., řeckého astronoma, matematika a geografa; správce alexandrijské knihovny, současníka Archimedova. Kromě prvočísel zkoumal řezy kužele, vlastnosti kuželoseček, souměřitelnost úseček, otázku zdvojení krychle. Z různé výšky Slunce nad obzorem v Alexandrii a Asuánu v den jarního slunovratu jako jeden z prvních (nezávisle na čínských učencích) usoudil, že Země je kulatá a dokonce spočítal délku zemského poledníku – asi 45 000 km, což je jen asi o 10% více, než skutečná hodnota.

[Zdrojový kód](#)

[Spustitelný kód](#)

Příklad 6 – výpočet hodnoty Lagrangeova interpolačního polynomu: Z numerické matematiky víme, že Lagrangeův interpolační polynom procházející množinou bodů $[x_i; y_i]; i = 1, \dots, n$ je

$$y = \sum_{i=1}^n y_i \frac{\prod_{j=1, j \neq i}^n (x - x_j)}{\prod_{j=1, j \neq i}^n (x_i - x_j)}$$

Tento vzorec je vyčíslíme uživatelskou funkcí Lagrange:

```

Function Lagrange(z:Double):Double
var i,j :Integer;
    Lagr,           {pomocná proměnná pro výpočet jednotlivých zlomků v součtu}
    Citatel,Jmenovatel :Double; {pom. proměnné pro čitatele a jmen. zlomků v součtu}
begin
    Lagr:=0;
    for i:=1 to n do
    begin
        Citatel:=1;Jmenovatel:=1;
        for j:=1 to n do
        if i<>j then begin
            Citatel:=Citatel*(z-x[j]);
            Jmenovatel:=Jmenovatel*(x[i]-x[j]);
        end;
        Lagr:=Lagr+y[i]*Citatel/Jmenovatel;
    end;
    Lagrange:=Lagr;
end;

```

Příklad 7 – výpočet součtu dvou vektorů: Výstupní proměnné uživatelských funkcí nemusí být jen jednoduchého typu. Jejich typ může dokonce definovat programátor tak, jak je zřejmé z následující jednoduché procedury, která počítá součet dvou vektorů (předpokládáme, že editovací okénka uživatelského rozhraní mají jména dle naší procedury):

```

procedure TForm1.VectorSoucet(Sender: TObject);
type TVector = array [1..3] of double;
var      u,v,w:TVector;

Function Soucet(u,v:TVector):TVector;
begin
    Soucet[1]:=u[1]+v[1];
    Soucet[2]:=u[2]+v[2];
    Soucet[3]:=u[3]+v[3];
end;

begin
    u[1]:=StrToFloat(EditU1.Text);
    u[2]:=StrToFloat(EditU2.Text);
    u[3]:=StrToFloat(EditU3.Text);
    v[1]:=StrToFloat(EditV1.Text);
    v[2]:=StrToFloat(EditV2.Text);
    v[3]:=StrToFloat(EditV3.Text);
    w:=Soucet(u,v);
    EditW1.Text:=FloatToStr(w[1]);
    EditW2.Text:=FloatToStr(w[2]);
    EditW3.Text:=FloatToStr(w[3]);
end;

```