GRAFICKÉ MODELY ŠTANDARDNÝCH ÚDAJOVÝCH ŠTRUKTÚR A ICH VÝZNAM VO VYUČOVANÍ PROGRAMOVANIA

STOFFOVÁ Veronika – UDVAROS József, SR

Resumé

Článok prináša opis grafických modelov štandardných údajových štruktúr, ktoré umožňujú budúcemu programátorovi nielen vytvoriť správnu predstavu o ich implementácii v pamäti počítača, ale pomáhajú mu aj pri manipulácii s ich jednotlivými elementmi a ich položkami v programe. Správna predstava o štruktúre a zoskupení údajov určitého typu do väčších celkov, ako aj o ich hierarchickej výstavbe umožňujú pochopiť pravidlá o prístupe k ich jednotlivým elementom, adresovanie tak jednotlivých elementov ako aj zložitejších údajových celkov, ich správne používanie v programoch na riešenie rôznych úloh. Modely sú založené na lineárnej štruktúre pamäte (teda pamäť je chápaná ako usporiadaná množina adresovateľných jednotiek – bajtov, príp. väčších celkov).

Kľúčová slová: grafické modely, údajové štruktúry, súbor

GRAPHICAL MODELS OF STANDARD DATA STRUCTURES AND THEIR IMPORTANCE IN TEACHING PROGRAMMING

Abstract

The article gives a graphic description of the models of standard data structures, which allow the beginner programmer to create the right idea about their implementation in computer memory. True understanding of the structure of a particular type of data clusters into larger units, their hierarchical construction permit to understand the rules of access to their individual elements, and the possibility of addressing individual elements as well as more complex data files allows their proper use in programs to solve various tasks. The models are based on the linear structure of memory (memory is viewed as an ordered set of addressable units – bytes).

Key words: graphical models, data structures, file

1 Úvod

Každý programovací jazyk ponúka určitú paletu štandardných údajových typov a údajových štruktúr, ktoré má programátor k dispozícii pri riešení programátorských úloh. Na základe štandardných údajových typov a údajových štruktúr môže v programe definovať aj vlastné údajové typy a štruktúry, ktoré mu uľahčia, príp. zjednodušia riešenie problému.

Údajová štruktúra je súhrn viacerých elementov rovnakého, príp. aj rôzneho typu, ktoré sú kvôli výhodnej manipulácii združené pod jedným menom (premennou) a sú obyčajne určitým spôsobom usporiadané, aby prístup k jednotlivým jej elementom bol jednoduchý a jednoznačný a aby bolo možné ich priamo adresovať. Premennou sa tu myslí inštancia určitého údajového typu. Pre každú údajovú štruktúru je implicitne daná jej vnútorná reprezentácia a je definovaná množina základných operácií pre prácu s ňou (Stoffová, 2003).

2 Štandardné údajové štruktúry

Medzi štandardné údajové štruktúry patria: pole, množina, záznam, súbor a pod. Vzhľadom na obmedzený rozsah práce podrobne opíšeme len grafické modely poľa a množiny.

Zoznamom sa budeme zaoberať len ako údajovou štruktúrou, ktorý môže byť elementom poľa (Végh, 2008 Végh – Csízi, 2009).

3 Pole

Pole je údajová štruktúra obsahujúca prvky rovnakého typu, ktoré sú usporiadané (akoby v sústave súradníc určitého rozmeru). Teda pole je homogénna údajová štruktúra. Prvky poľa sú prístupné na základe hodnoty indexu (často hovoríme, že sú indexované). Poznáme jednorozmerné pole (ktoré je často stotožňované s matematickou údajovou štruktúrou vektor), dvojrozmerné pole (ktoré je často stotožňované s matematickou údajovou štruktúrou matica) a viacrozmerné pole. Pole môže mať všeobecne ľubovoľný nezáporný celočíselný rozmer. Rozmer poľa ako aj hodnota indexu môžu byť definované ľubovoľnou hodnotou, (príp. premennou) ordinálneho typu. Ordinálne typy sú také údajové typy, kde počet hodnôt, ktoré môže premenná nadobudnúť je konečný, a hodnoty samotné je možné usporiadať, teda každá hodnota má svoje poradové číslo v usporiadanej postupnosti. Tieto skutočnosti evokujú predstavu o implementácii jednorozmerného poľa, ktoré vychádza zo základnej definície usporiadanosti operačnej pamäte počítača, podľa ktorej operačná pamäť je usporiadaná množina adresovateľných jednotiek – bajtov, príp. iných väčších pamäťových celkov.

3.1 Jednorozmerné pole

Jednorozmerné pole si možno predstaviť ako pevný počet škatúľ (zásuviek) na skladovanie vecí (informácií, hodnôt), ktoré sú spojené jedna po druhej do väčšej jednotky nazvanej spoločným menom – menom deklarovaného poľa. Vektor môže obsahovať daný počet definovaných hodnôt (premenných), ktoré má programátor deklarovať. Vzhľadom na dopredu definovaný počet elementov, pole je statická údajová štruktúra. Veľkosť poľa počas behu programu sa nedá zmeniť. Jednorozmerné pole možno deklarovať všeobecne nasledovne:

var
***názov_poľa* : array *[1 .. N]* of *typ_elementov*;**

napr. `PoleCelychCisel: array [1 .. 20] of integer;`

V príklade array [1 .. 20] definuje veľkosť poľa (počet elementov poľa) a *integer* definuje typ elementov poľa (v našom prípade sú to celé čísla).

Ak chceme určiť hodnotu (uložiť hodnotu) pre konkrétny index v poli, urobíme to nasledovne:

názov_poľa [index]: = hodnota;

```
a[5]: = 10;
a[1]: = 25;
```

3.2 Dvojrozmerné pole

Dvojrozmerné polia sú údajové štruktúry, do ktorých ukladáme hodnoty v riadkoch a stĺpcoch a majú rozmer N * M. Môžeme si ich predstaviť ako plošné 2D útvary, kde pomocou indexu riadku a indexu stĺpca možno jednoznačne nájsť konkrétny prvok. Najbližšie sú k nim predstavy o maticiach, kde sú prvky usporiadané do stĺpcoch a do riadkov.

Dvojrozmerné pole je formálne deklarované nasledovne:

názov_poľa:* array *[1 .. N, 1 .. M]* of *typ_údajov;

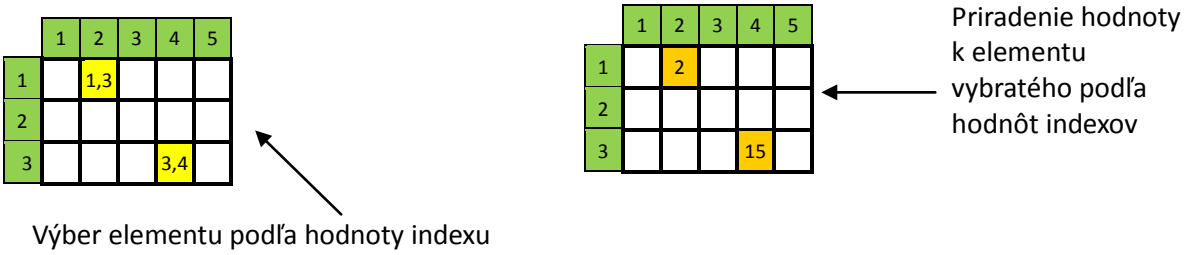
Napr. `b: array [1 .. 3, 1 .. 5] of integer;`

Vo vyššie uvedenej deklarácii, pole **b** obsahuje 15 čísiel v troch riadkoch a piatich stĺpcoch. Teda na dvojrozmerné pole môžeme pozerať ako na maticu, kde miesto prvku (samotný prvok

údajovej štruktúry) jednoznačne identifikujeme s indexmi, ako napr.: b[i,j], b[1,3], b[3,4]. Ak chceme priradiť hodnotu na konkrétne miesto v štruktúre dvojrozmerného poľa, ktoré určíme pomocou indexov, urobíme to nasledovne:

názov_poľa [index1, index2]: = hodnota;

Napr. `a[3,4]: = 15; a[1,2]: = 2;`

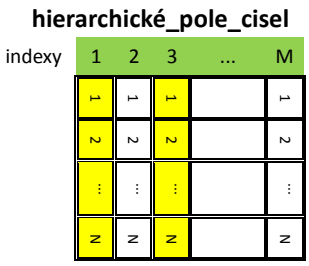


Viac ako dvojrozmerné polia sa používajú zriedkavo. Naša priestorová predstavivosť končí pri trojrozmernom poli. Viacrozmerné pole môžeme abstrahovať (predstaviť) ako určitý počet o jeden rozmer jednoduchších polí. Takto možno definíciu (predstavu) rekurzívne rozvíjať, kým sa nedostaneme k jednorozmernému poľu. Viacrozmerné pole (jeho prvky) možno rozvinúť do lineárnej pamäťovej štruktúry tak, že **posledný index elementu sa mení najrýchlejšie**. Teda v prípade dvojrozmerného poľa ako keby boli prvky uložené po riadkoch v rovine (Stoffová – Végh, 2006, 2008).
Jednorozmerné, dvojrozmerné, príp. trojrozmerné polia sa používajú najčastejšie, lebo prvé dva útvary si vieme predstaviť v rovine a tretí v priestore (3D). Štvorrozmerné pole je vlastne určitý počet trojrozmerných polí. Počet trojrozmerných polí udáva štvrtý index. (Je to rozdiel medzi hornou a dolnou hranicou štvrtej dimenzie.)

V špeciálnych prípadoch môžeme definovať pole aj hierarchicky, keď prvkami poľa nie sú elementy jednoduchého typu. V takomto prípade elementy poľa sú štruktúrované a sú na vyššej hierarchickej úrovni než štandardné údajové typy. Prvkom hierarchicky budovaného (definovaného) poľa môže byť, napr. jednorozmerné pole alebo záznam. Takto definovaný neštandardný typ je ďalej deliteľný. Keď prvkami budú jednorozmerné polia musíme definovať nový údajový typ ako jednorozmerné pole, s ktorým určíme – deklarujeme typ prvkov jednorozmerného poľa ako adresovateľnú jednotku. Takéto dvojrozmerné pole je deliteľné na adresovateľné stĺpce.

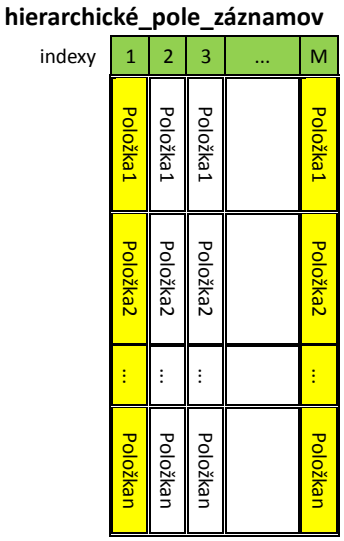
type *typ_poľa* = array *[1 .. N]* of *typ_údaja;*

var
hierarchické pole :* array *[1 .. M]* of *typ_poľa;
Prvky hierarchického poľa jednoznačne identifikujeme s dvomi indexami: hierarchické pole [i][j], kde *i* určí prvok hierarchického poľa a *j* poradie prvku v prvku hierarchického poľa.
V nasledujúcej úlohe ukážeme prácu
s hierarchickým poľom.




```
Program pole;
Uses crt;
Type a=array [1..5] of byte;
Var
b:array[1..3] of a;
i,j:byte;
```

```
Begin
Randomize; {vygenerovanie hodnôt}
For i:=1 to 3 do
    For j:=1 to 5 do b[i][j]:=random(50);
{výpis hodnôt poľa}
For i:=1 to 3 do
    begin
    For j:=1 to 5 do write(b[i][j],' ');
    writeln;
    end;
readln;
end.
```



Na predchádzajúcom obrázku sú jednoznačne explicitne identifikované stĺpce 1. a 3. (a v nich indexy položiek stĺpca). Na ďalšom obrázku sú žltou farbou zafarbené a stĺpce 1 a M. Kde prvkami jednorozmerného poľa sú záznam. Najprv ale deklarujeme údajový typ záznam. Potom deklarujeme jednorozmerné pole tak, aby jeho prvky boli typu záznam.

Type typ_záznam = record
Položka1:Typ1;
Položka2: Typ2;
...
PoložkaN:TypN);
end;
var
Pole : array[1..3] of typ_záznam;

V nasledujúcej úlohe ukážeme prácu s hierarchickým poľom, kde prvky jednorozmerného poľa sú záznamy. Jeden záznam obsahuje základné údaje o jednej knihe.

```
Program pole;
Uses crt;
Type kniha = record
     autor:string[24];
     rok:integer;
     nazov:string;
     end;
var
knihy : array[1..3] of kniha; i:byte;
begin
{zadavanie hodnot}
for i:=1 to 3 do begin
    write('Zadaj ',i,'. autora knihy:');readln(knihy[i].autor);
    write('Zadaj rok vydania ',i,'. knihy:');readln(knihy[i].rok);
    write('Zadaj názov ',i,'. knihy:');readln(knihy[i].nazov);
    writeln;
    end;
{vypis hodnot pola}
For i:=1 to 3 do begin
    writeln(i,'. kniha');
    writeln('Autor: ',knihy[i].autor);
    writeln('Rok vydania: ',knihy[i].rok);
    writeln('Názov: ',knihy[i].nazov);
    writeln;
```

```
    end;
readln;
end.
```

Práca s údajovou štruktúrou pole umožňuje pochopiť fungovanie riadiacej štruktúry cyklus. Dvoj- a viacrozmerné pole pomáha vysvetliť jednoduché vnorené, ale aj hierarchicky viacnásobne vnorené cykly.

4 Množina

Množina je neusporiadaný útvar elementov, v ktorom sa každý element vyskytuje presne raz. Na množine môžeme definovať nasledujúce operácie: rozdiel, prienik, zjednotenie (Angster, 1995, 1999). Typom množiny určíme, aké prvky môžeme do nej vkladať.

***názov_množiny* : Set Of *typ_prvkov*;**

Napr. `M : Set Of Byte;`
Elementy do množiny (uložíme) priradíme nasledovne:

```
M := [1, 3, 5, 7];      - zoznam
M := [1..7];            - interval
M := [1, 3..7];         - kombinácia intervalu a zoznamu
M := [];                - prázdna množina
```

Ďalšie prvky možno pridávať pomocou operátora „+“ (zjednotenie):

Napr. M := M + [10..5, 70];

Dôležité je, aby sme vedeli skúmať prvky množiny. Na zistenie či množina obsahuje hľadaný prvok, slúži operácia *IN*. V nasledovnom príklade ukážeme ako vkladáme veľké písmená do množiny, ktoré program pred svojim ukončením vypíšeme na monitor.

```
Program zber ;
Uses Crt ;
Var
Pismena : Set Of  'A' .. 'Z' ;
Ch : Char;
Begin
Pismena := [] ;
Ch := UpCase(ReadKey) ;
While Ch <> #27 Do
Begin
If  Ch  In  ['A'  ..  'Z']  Then
Pismena := Pismena + [Ch] ;
Ch := UpCase(ReadKey) ;
End ;

{výpis prvkov množiny}
For Ch := 'A' To 'Z' Do
If Ch In Pismena Then Write(Ch)
;
End.
```

Zaujímavé je, že začínajúci programátori používajú vo svojich programoch štruktúrovaný údajový typ *množina (set)* len zriedka. Možno ťažko nachádzajú analógiu medzi riešením problému a „fungovaním“ a vlastnosťami údajovej štruktúry množina.

4 Záver

Podľa našich skúseností študenti ľahšie pochopia prácu s údajovými štruktúrami ak majú k dispozícii grafický model. Tento model môžu využiť aj na znázornenie postupu riešenia a zobrazenie zmien hodnôt jednotlivých elementov. Grafický model, ktorý zjednoduší riešenie problému, možno použiť na myšlienkovú implementáciu ľubovoľnej štandardnej údajovej štruktúry. Pomocou grafickej reprezentácie je možné nielen pochopiť fungovanie selekcie elementu, spôsob adresovania, určenie indexov, explicitne definovaných stavebných elementov štandardnej údajovej štruktúry, ale si aj osvojiť prácu s nimi. Grafické zobrazenie údajovej štruktúry dáva priestor aj na sledovanie realizácie algoritmu a zápis zmien hodnôt

jednotlivých elementov. Používateľ – študent môže experimentovaním na grafickom modeli zadávaním rôznych vstupných údajov testovať správnosť svojho programu (časti programu), pochopiť aj zložitejšie operácie vykonané nad údajovými štruktúrami a odhaliť aj prípadné chyby a nedostatky programu (Stoffa, 2003).

Literatúra

1. ANGSTER, E.: *Programozás tankönyv I.* Martonvásár : Akadémiai nyomda, 1995. 253 s. ISBN 963-4509-56-8
2. ANGSTER, E.: *Programozás tankönyv II.* Martonvásár : Akadémiai nyomda, 1999. 1-153 s. ISBN 963-4509-57-6
3. STOFFA, V.: *Az elektronikus tanulás lehetőségei a programozás elsajátításában*. In: *AGRIA MEDIA 2002 : Az elektronikus tanulás a III. évezred pedagógiai kihívása*. Eger, 2003, s. 480 - 488. ISBN 963-9417-09-2.
4. SCHANDA, J. – SIK LÁNYI, C.: "Colour Correct Digital Museum on the Internet, Virtual Reality,." *Proc. HCII 2007, Lecture Notes in Computer Science, LNCS 4563,* 2007, pp. 708-717.
5. SIK LÁNYI, C. – GEISZT, Z. – KÁROLYI, P. – TILINGER, Á. – MAGYAR, V.: Virtual Reality in special needs early education, *The International Journal of Virtual Reality*, 2006, 5(4): 55-68. ISSN1081-1451: http://www.ijvr.org/issues/issue4/7.pdf
6. STOFFOVÁ, V. – VÉGH, L.: Szemléltető animációk a programozásban. In: *INFODIDACT 2010, 3. Informatika Szakmódszertani Konferencia*. Szombathely, 2010
7. STOFFOVÁ, V., 2003. *Computer-aided learning of programming.* In: Proceedings of the International Conference of Computer Systems and Technologies (e-Learning). Editors B. Rachev, A. Smrikarov, Sofia: Bulgarian Union of Automation and Informatics, 2003, s. IV.33-1- IV.33-6. ISBN 954-9641-33-3.
8. STOFFA, V. – VÉGH, L.: Guided animation of dynamic data structures. In *Third Central European Multimedia and Virtual Reality Conference*. Eger, Hungary 2006, 175–179. p. ISBN 963-9495-89-1.
9. VÉGH, L.: Elektronická podpora vyučovania dynamických údajových štruktúr. (Electronic support of teaching dynamic data structures). In Stoffa, J. – Stoffová, V. (editori) *XIX. DIDMATTECH 2006*. Komárno: Univerzita J. Selyeho 2007, 289–292. p. ISBN 978-80-89234-23-3.
10. VÉGH, L. – CSÍZI, L.: A programozás e-learninggel támogatott oktatása (E-learning supported teaching of programming). In Stoffová, V. (ed.): *XXI. DIDMATTECH 2008, 1st part*. Eger, Hungary : Eszterházy Károly Főiskola, 2009. 65-69. s. ISBN 978-963-9894-17-4.

Lektoroval: **Prof. Dr. Ing. Imrich Okenka, PhD.**

Kontaktné adresy:

Veronika Stoffová, Prof. Ing. CSc., tel. +421 35 3260 605, e-mail: NikaStoffova@seznam.cz

Univerzita J. Selyeho, Ekonomická fakulta, Katedra matematiky a informatiky,
Bratislavská cesta 3322, 945 01 Komárno, Slovenská republika

RNDr. József Udvaros, e-mail: udvarosj@atlas.sk
Faculty of Informatics, Eötvös Loránd University, Pázmány Péter sétány 1/c., 1117 Budapest,