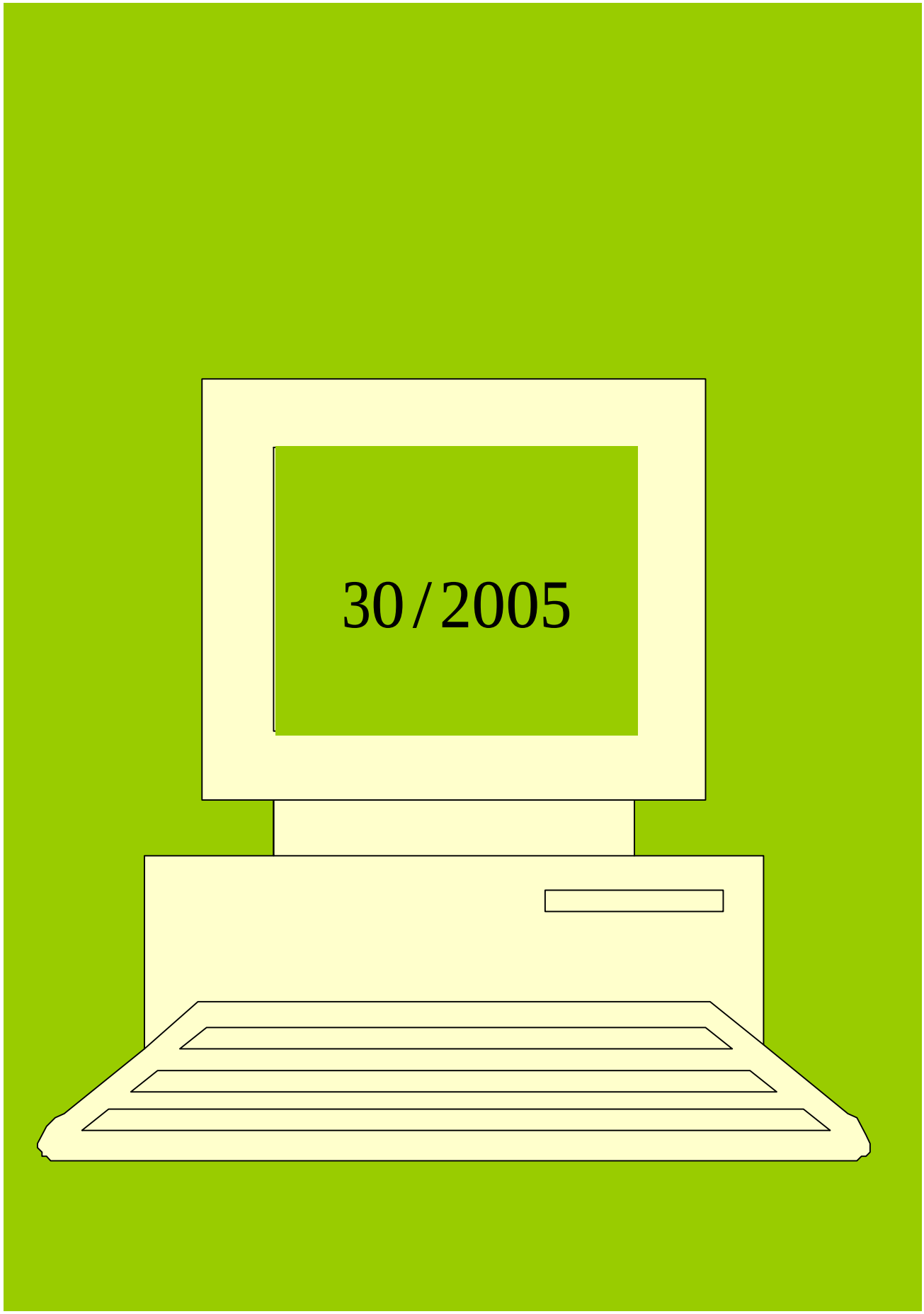

30 / 2005

Informatika v škole č. 30 2005

Informačné periodikum
O teoretických, metodických otázkach a skúsenostiach z praxe pri uplatňovaní informatiky a výpočtovej techniky v základných a stredných školách

Predseda redakčnej rady: PhDr. Peter ZVERKA
Výkonná redaktorka: Ing. Alžbeta MEGOVÁ
Jazyková redaktorka: Mgr. Gabriela AICHOVÁ

Vydáva
Ústav informácií a prognóz školstva v Bratislave

Adresa redakcie: Ústav informácií a prognóz školstva
Staré grunty 52
842 44 Bratislava
e-mail: megova@uips.sk

OBSAH

GRAF V PREZENTÁCII POWER POINT

Vo vyučovacom procese sa čoraz viac dostáva do pozornosti využívanie prezentácií na skvalitnenie vyučovacieho procesu. Pri tvorbe prezentácií je asi najviac využívaný prostriedok spadajúci do balíka MS Office – Power Point. Jeho použitie je užívateľsky veľmi príjemné, užívateľ nemusí mať výnimočné znalosti s tvorbou prezentácie v tomto prostriedku.

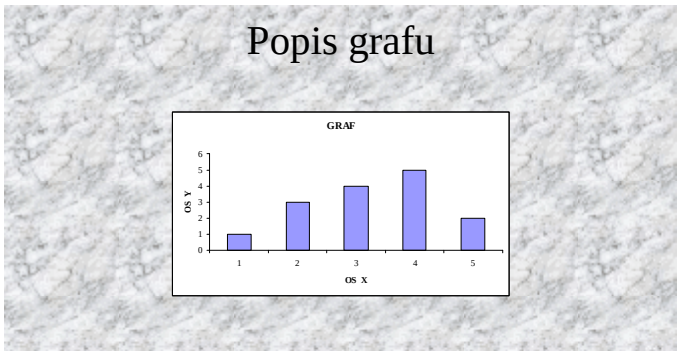
V príspevku by som však chcela poukázať na niektoré problémy a možnosti pri tvorbe prezentácie v prípade, že potrebujeme prezentovať grafické výstupy z aplikácie MS Excel.

Zobrazenie grafu v prezentácii

Jednoduché prenesenie grafu z prostriedku MS Excel do prezentácie má za následok, že daný graf sa zobrazuje s bielym pozadím, bez ohľadu na pozadie snímku. To môže pôsobiť rušivo pri samotnom predvádzaní prezentácie. Ak by sme chceli zmeniť vlastnosť daného objektu cez „Formát objektu“, dostaneme taký výpis vlastností, ktoré nezodpovedajú našej požiadavke prispôsobenia pozadia grafu s pozadím prezentácie.

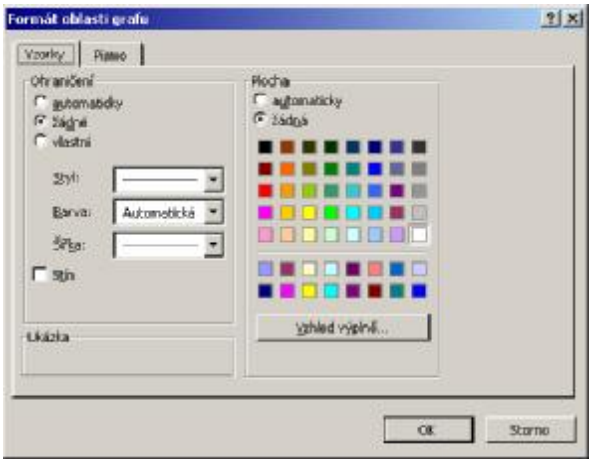
Z daného obrázku je vidieť, že objekt (graf) aj keď má vlastnosť bez pozadia, sa javí ako priehľadný, avšak výsledok v prezentácii je opačný.

Obr. 1



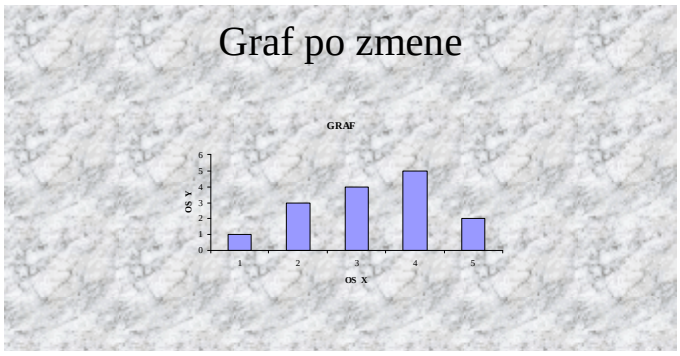
Zmenu dosiahneme dvojitým kliknutím na objekt grafu, čím sa dostaneme do prostredia možnosti zmeny vlastností grafu a opätovným dvojitým kliknutím dostaneme okno vlastností, v ktorom nastavíme plochu ako priehľadnú a tým sa zmení aj zobrazenie daného grafu.

Obr. 2



V okne vlastnosti zmeníme vlastnosť plochy na „žádná“. Ak chceme odstrániť aj rámček okolo grafu, zmeníme vlastnosť „ohraničení na žádné“. Snímok s grafom potom bude vyzerať nasledovne.

Obr. 3

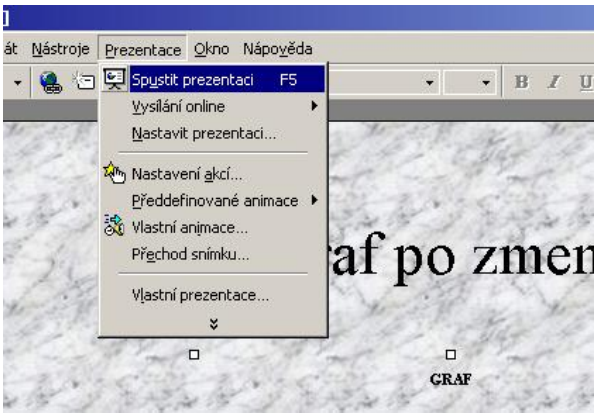


Podľa zobrazenia daného grafu môžeme zmeniť vlastnosti ako sú výška a hrúbka písma, prípadne zvýrazniť hrúbku čiar pre osi využívajúc formát osi.

Animácia grafu

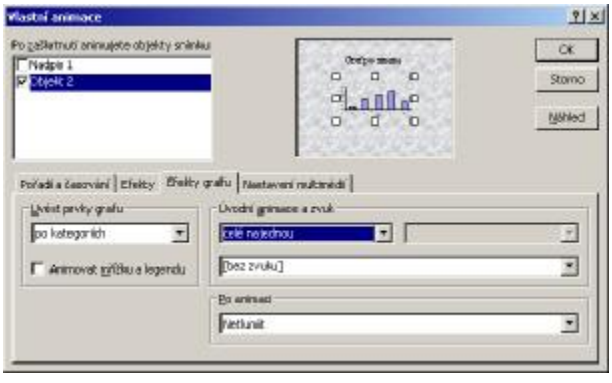
Pri uvedenom type grafu, ktorý je zobrazený na snímke je možnosť nastavenia aj tzv. vlastnej animácie, kde môžeme predvádzať jednotlivé stĺpce grafu postupne po jednom. Po označení objektu (grafu) klikneme na „Prezentace“, alebo klikneme pravým tlačidlom a vyberieme si možnosť „vlastní animace“.

Obr. 4



Po potvrdení tejto možnosti dostaneme okno vlastnosti, ktoré je znázornené na obr. 5. V jednotlivých kategóriách daných vlastností môžeme nastaviť požadované vlastnosti.

Obr. 5



Nastavením vlastností v tomto okne dosiahneme efekt, keď sa budú jednotlivé stĺpce postupne zobrazovať na obrazovke buď po stlačení klávesu Enter, alebo ak nastavíme časovanie, tak podľa daného časovania.

Na záver len pripomeniem, že to, ako bude vyzerať naša prezentácia záleží len na nás. Mali by sme však mať na zreteli, aby prílišné efekty, ktoré nám Power Point ponúka, neboli na ujmu zrozumiteľnosti a neodvádzali pozornosť od prezentovaných informácií.

Literatúra

WAGNER, D., DVOŘÁKOVÁ, E.: PowerPoint 7 pro Windows 95. Kompletní kapesní průvodce, GRADA, 1997.

Ing. Stella HREHOVÁ, PhD.
Katedra informatiky, matematiky a kybernetiky
FVT TU v Košiciach so sídlom v Prešove
hrehova@fvt.sk

POROVNÁVANIE OPERÁCIE KOPÍROVANIA V DVOCH PROGRAMOVACÍCH JAZYKOCH

V príspevku sa zaoberáme porovnávaním niektorých vlastností programov v jazyku C, resp. v strojovo orientovanom jazyku (ďalej len SOJ). Zameriame sa hlavne na rozdiel v rýchlosti spracovania kódu napísaného v týchto jazykoch pri operácii presunu údajov z jedného do druhého poľa. Použijeme rôzne techniky kopírovania. Výsledky porovnáme a zhodnotíme.

Úvod

Kopírovanie údajov je jedna z operácií, ktorá dobre ukazuje, akým spôsobom pracujú programy v C a SOJ s pamäťou a cyklom. Pre lepšie hodnotenie výsledkov zvolíme viac možných spôsobov kopírovania v oboch jazykoch. Už tradične jazyk C poskytuje viac možností, ale aj v SOJ môžeme kopírovať viacerými technikami. Pre vysokú rýchlosť kopírovania v oboch jazykoch použijeme prístup cez cyklus, kde sa daný reťazec presúva viackrát. V C uplatníme nasledovné prístupy:

1. Kopírovanie cez pole:
   - cez jeden cyklus pomocou premennej typu long,
   - cez dva cykly pomocou premenných int do dynamicky alokovaného poľa,
   - cez dva cykly pomocou premenných typu int do staticky alokovaného poľa.
2. Kopírovanie pomocou funkcie strcpy:
   - cez jeden cyklus pomocou premennej typu long,
   - cez dva cykly pomocou premenných typu int do poľa.

V SOJ použijeme nasledovné techniky:

1. kopírovanie cez cyklus (pole),
2. kopírovanie pomocou inštrukcie movsb.

Aby boli výsledky smerodatné, vykonáme testy pre dva rôzne počty kopírovaní (prvýkrát 6553500, druhýkrát 655350).

V ďalších častiach priblížime jednotlivé prístupy. Výsledky budú zobrazené v tabuľkách, významné porovnania v grafoch. Kopírovanie budeme sledovať na viacerých reťazcoch rôznej dĺžky (10, 20, 30, 47 znakov), pre každý 10-krát. V tabuľkách je uvedený priemer za jednotlivé behy, stredná doba kopírovania jedného znaku a celkový priemer na znak za všetky behy.

1. Kopírovanie v C pomocou premennej typu long cez pole

Sledujeme túto časť programu:

```
long i;
for ( i=0; i < 6553500 ; i++)
    for ( int j=0 ; j < 47; j++)
          p_txt[j] = txt[j];
```

kde p_txt je cieľové pole, txt je zdrojové pole.

	Kopírovanie 6553500-krát v C pomocou premennej typu long			
	47 znakov	30 znakov	20 znakov	10 znakov
	8,186813	5,384615	3,791209	2,142857
	8,186813	5,384615	3,791209	2,142857
	8,186813	5,384615	3,791209	2,142857
	8,186813	5,384615	3,791209	2,142857
	8,186813	5,384615	3,791209	2,142857
	8,186813	5,384615	3,736264	2,142857
	8,351468	5,384615	3,736264	2,142857
	8,351468	5,384615	3,736264	2,142857
	8,131868	5,43956	3,736264	2,197802
	8,131868	5,32967	3,736264	2,087192
priemer	8,208755	5,384615	3,7637365	2,142785
na znak v micros	0,026650547	0,027387986	0,028715469	0,032696803
priemer na znak v ms	0,028862701			

Je zaujímavé, že pri tomto spôsobe kopírovania nezáleží na tom, či je cieľ alokovaný na halde alebo staticky. Pri ostatných prípadoch sa presvedčíme o opaku.

2. Kopírovanie v C cez dva cykly pomocou premennej typu int do dynamicky alokovaného poľa

Sledujeme túto časť programu:

```
unsigned int k , i;
for ( int k = 0 ; k < 100; k++)
  for( i = 0 ; i < 65535 ; i++)
    for ( int j=0; j < 47; j++)
          p_txt[j] = txt[j];
```

pričom txt je statické pole a p_txt je dynamicky alokované pole.

V tejto tabuľke, ani v ďalších, neuvádzame číselné hodnoty jednotlivých desiatich meraní, ale len posledné tri riadky s vypočítaným priemerom.

	Kopírovanie 6553500-krát v C pomocou premennej int do dyn.poľa			
	47 znakov	30 znakov	20 znakov	10 znakov
priemer	5,318681	3,489011	2,428571	1,3461535
na znak v micros	0,017267632	0,017746298	0,018528809	0,020540986
priemer na znak v ms	0,018520931			

Tento spôsob je ukážkou toho, ako sa dá pri správnej práci s pamäťou zrýchliť kopírovanie. Ak namiesto jednej premennej typu long zvolíme dve typu int, prístup do pamäti pri vykonávaní cyklov sa urýchli a výsledky sú určite lepšie.

3. Kopírovanie v C cez dva cykly pomocou premennej typu int do staticky alokovaného poľa

Kopírovanie sa vykoná zo statického poľa txt do statického poľa txt2

```
unsigned int k , i;
for ( int k = 0 ; k < 100; k++)
  for( i = 0 ; i < 65535 ; i++)
    for ( int j=0; j < 47; j++)
          txt2 = txt[j];
```

	Kopírovanie 6553500-krát v C pomocou premennej int do stat.pola			
	47 znakov	30 znakov	20 znakov	10 znakov
priemer	2,824176	1,9505465	1,4120875	0,8681302
na znak	0,009168971	0,009921144	0,010773537	0,013246818
priemer na znak v ms	0,010777617			

Jedinou zmenou oproti predchádzajúcemu príkladu je cieľ (ten je alokovaný staticky). Zrýchlenie je takmer dvojnásobné a oproti kopírovaniu cez premennú typu long takmer trojnásobné. Keďže toto riešenie vychádza zo všetkých skúmaných možností najlepšie, je porovnávané aj s výsledkami dosiahnutými s programami v SOJ, ktoré budú priložené na grafoch.

4. Kopírovanie v C pomocou funkcie strcpy cez premennú typu long

Sledujeme túto časť programu:

```
long i;
for(i=0;i<6553500;i++)
          strcpy(txt2,p_txt);
```

	Kopírovanie 6553500-krát v C pomocu strcpy s long premennou			
	47 znakov	30 znakov	20 znakov	10 znakov
priemer	7,3681315	5,1923075	4,153846	2,8736265
na znak	0,023921379	0,026409845	0,031691814	0,04384873
priemer na znak v ms	0,031467942			

Kopírovanie pomocou špecializovanej funkcie je pomalšie oproti predchádzajúcemu príkladu (do statického poľa cez int), takmer trojnásobne. Zrejme to bude závislé na častom volaní funkcie strcpy, ktoré môže celý beh spomaliť.

5. Kopírovanie v C pomocou funkcie strcpy cez premenné typu int

Jedná sa o túto časť programu:

```
for(int k=0;k<100;k++)
  for(i=0;i<65535;i++)
          strcpy(p_txt,p_txt);
```

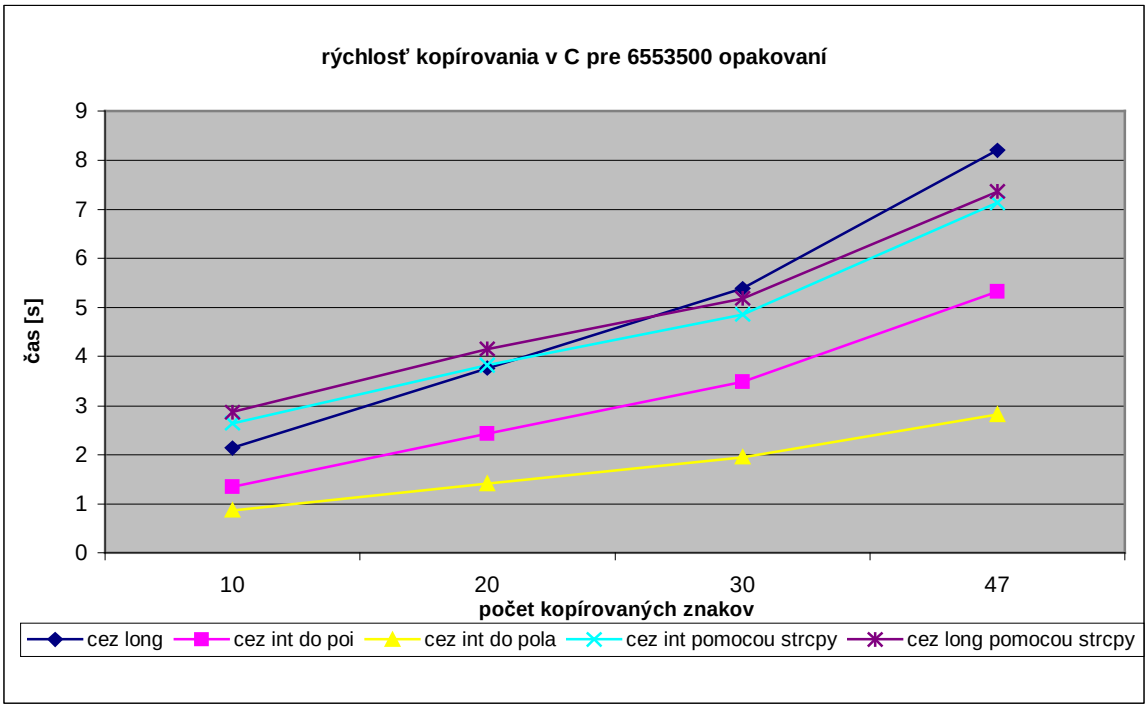
	Kopírovanie 6553500-krát v C pomocu strcpy s integer (je jedno do coho)			
	47 znakov	30 znakov	20 znakov	10 znakov
priemer	7,131868	4,8626375	3,824176	2,6378575
na znak	0,023154326	0,024733031	0,029176593	0,040251125
priemer na znak v ms	0,029328769			

Tento prípad je rýchlejší ako predchádzajúci, ale zrýchlenie nie je výrazne významné. Ostatné typy kopírovania ho podstatne predbiehajú.

Je zaujímavé všimnúť si, že pri kopírovaní cez strcpy takmer vôbec nezávisí na tom, akým spôsobom je alokovaný cieľ (dynamicky alebo staticky). Porovnávania nie sú práve z týchto dôvodov zahrnuté, aj keď boli vykonávané.

Zhrnutie výsledkov kopírovania v jazyku C

Na záver časti o spôsoboch kopírovania v programoch napísaných v jazyku C si zhrňme dosiahnuté výsledky v prehľadnom grafe:



Z grafu je vidieť, že najmenej času sa spotrebovalo pri kopírovaní obsahu statických polí za použitia premennej typu int (spodná žltá čiara).

Môžeme vyvodiť nasledujúce závery:

– Pri kopírovaní (ak sa deje v cykloch) je výhodnejšie používať premenné typu int, prípadne short, spracovanie v C sa deje podstatne rýchlejšie ako napr. s premennou typu long (je to dané veľkosťou typu).

– Je výhodnejšie používať statické pole, ak nepotrebujeme použiť dynamické z iných príčin.

Aj keď horeuvedené výsledky sú podstatne odlišné, museli sme použiť extrémne veľký počet kopírovaní, aby boli výsledky vôbec merateľné. Pri presúvaní menšieho množstva dát nemá spôsob kopírovania v podstate na samotnú rýchlosť takmer žiadny vplyv (alebo len minimálny).

7. Kopírovanie v SOJ cez pole

Testovaný bol nasledujúci program, napísaný v strojovo orientovanom jazyku, ktorý presúval obsah poľa txt2 do poľa txt3.

```
mov ax, @data
mov ds, ax
mov es, ax

        mov cx,100                              ; kopirujem to 100 * 65535 krat
        mak:
                mov ax, cx
                mov cx,65535                    ;tolkotokrat opakujem vnutorny cyklus
                makaj:
                        mov si,offset txt2
                        mov di,offset txt3
                        add di,50               ; na takuto adresu retazec zkopcime
                        mov bx,cx               ; odlozim cx lebo ho potrebujem pre retazec

                      kopiruj:
                                mov cl,[si]     ; a tu kopirujeme po jednom
                                mov [di],cl
                                inc si
                                inc di
                                cmp cl,'$'      ;zistim ci uz skoncil retazec
                                jne kopiruj

                        mov cx,bx               ; a tu ho loadujem pre pouzitie v cykle
                        loop makaj
                mov cx,ax
                loop mak
```

Pri kóde sú zobrazené komentáre, ktoré upresňujú činnosť programu. Pracuje sa v dvoch cykloch, ktoré zabezpečujú dostatočné množstvo kópií, aby ich bolo možné zaznamenať. Vnútorný cyklus slúži na samotné kopírovanie.

Trvanie kopírovania pre rôzny počet kopírovaných znakov je zapísané v nasledujúcej tabuľke

	Kopírovanie 6553500 krát v SOJ cez cyklus			
	47 znakov	30 znakov	20 znakov	10 znakov
priemer	2,7197785	1,80276918	1,42857	0,8791208
na znak v micros	0,008830034	0,009169498	0,01089929	0,013414524
priemer na znak v ms	0,010578337			

8. Kopírovanie v SOJ pomocou inštrukcie movsb

Tabuľka za týmto kódom uvádza sledované časy:

```
mov ax, @data
mov ds, ax
mov es, ax

mov cx,100                                      ; kopirujem to 100 * 65535 krat
mak:
        mov ax, cx
        mov cx,65535
```

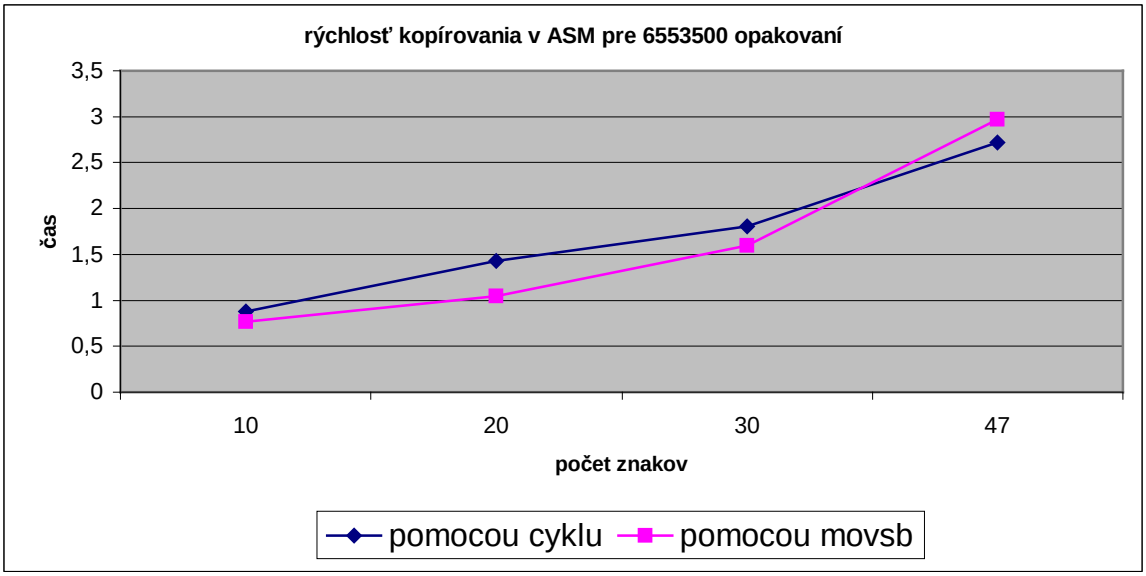
```
makaj:
        mov si, offset txt2
        mov di,offset txt3              ; na tuto adresu budeme kopirovat

        mov bx,cx               ; odlozim cx lebo ho potrebujem pre retazec
        mov cx,47               ; a kopirujeme 47 znakov
        rep movsb
        mov cx,bx               a tu ho loadujem pre pouzitie v cykle
 loop makaj

mov cx,ax
loop mak
```

	Kopírovanie 655350 krát v SOJ cez movsb			
	47 znakov	30 znakov	20 znakov	10 znakov
priemer	0,274725	0,10989	0,082417	0,05494505
na znak v micros	0,008919223	0,00558938	0,006288014	0,008384077
priemer na znak v ms	0,007295173			

Rozdiel oproti predchádzajúcemu príkladu je v inštrukcii movsb (namiesto vnútorného cyklu). Tento spôsob je rýchlejší približne 1,5-krát. Kopírovanie pomocou inštrukcie movsb je určite lepšie optimalizované ako náš kód.

Zhrnutie kopírovania v SOJ

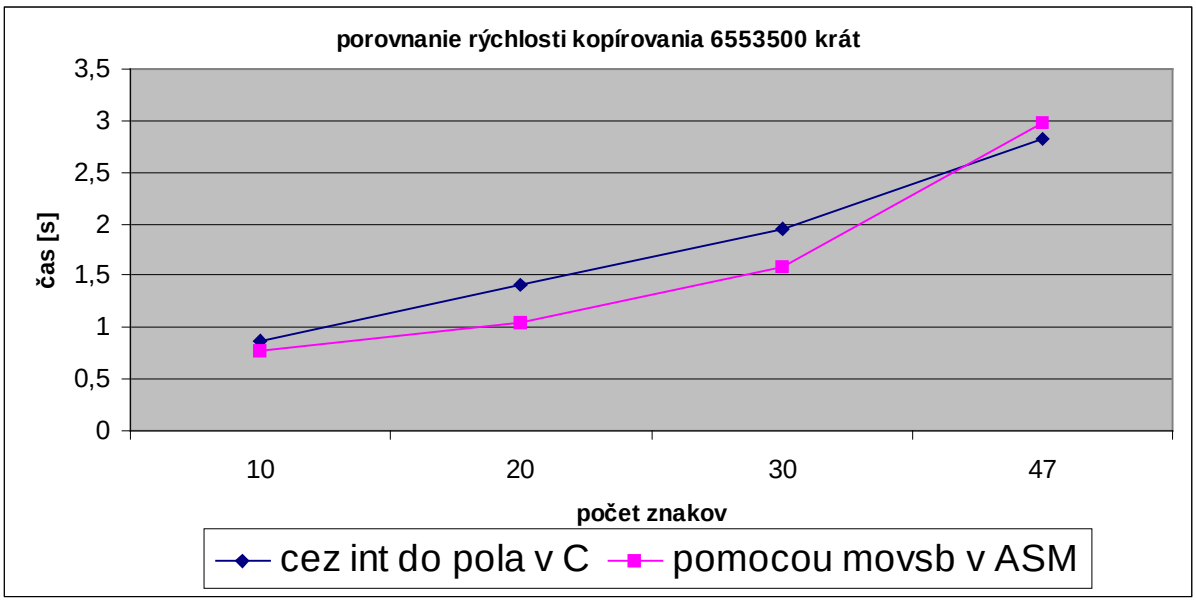
Pri sledovaní spotrebovaného času programami strojovej úrovne sme sa zaoberali iba dvoma druhmi kopírovania: pomocou cyklu a pomocou inštrukcie movsb. Výsledky ich porovnávania sú zakreslené v grafe:



Je zaujímavé všimnúť si, že pri zvyšovaní množstva kopírovaných znakov movsb stráca na výkonnosti a do popredia sa dostáva klasické kopírovanie. Podobne ako v jazyku C, nemá zmysel rozhodovať sa pri kopírovaní menšieho počtu údajov, ktorý spôsob použijeme, pretože výsledky sú prakticky identické. Samotné rozhodovanie nám však môže uľahčiť relatívne prehľadnejší zápis pri movsb.

Porovnanie kopírovania v C a SOJ

Zobrazenie všetkých výsledkov do jedného grafu by bolo neprehľadné, preto sme zvolili porovnávanie iba najvýkonnejších spôsobov v oboch jazykoch.



Z grafu je vidieť, že pri správnom prístupe v programovaní v jazyku C môžeme dosiahnuť takmer identickú rýchlosť s programami v symbolickom jazyku (hovorovo asembler - ASM). Keď si uvedomíme, že C je nízkoúrovňový jazyk, výsledky sú pochopiteľné.

Programy v jazyku C boli realizované v prostredí Borlandc. Strojovo orientované programy boli spracované prekladačom TASM a linkerom TLINK.

Jana PARÍZKOVÁ
Fakulta informatiky a informačných technológií STU
Bratislava

POROVNANIE RÝCHLOSTI ZOBRAZENIA BODOV OBRAZOVKY V DVOCH PROGRAMOVACÍCH JAZYKOCH

V tomto príspevku sa zaoberáme porovnávaním niektorých vlastností programov v jazyku C a v strojovo orientovanom jazyku (ďalej len SOJ). Zameriame sa na rozdiel v rýchlosti spracovania kódu napísaného v týchto jazykoch pri zobrazovaní pixelov v grafickom móde. Výsledky porovnáme a zhodnotíme.

Úvod

V predchádzajúcom článku „Porovnávanie operácie kopírovania v dvoch programovacích jazykoch" sme sa zaoberali rýchlosťou spracovania údajov pri ich kopírovaní. Porovnávali sme programy napísané v jazyku C a v strojovo orientovanom jazyku (hovorovo asembler). V tomto príspevku sa sústredíme na rýchlosť spracovania programu, ktorý zobrazuje body obrazovky v grafickom režime.

Pri zobrazovaní pixelov na obrazovku je toľko variácií, ako pri kopírovaní reťazcov nemáme. Na porovnanie rýchlosti vykonania programu v oboch jazykoch, si však vystačíme s niekoľkými nameranými hodnotami. Testovanie budeme realizovať v grafickom móde nastavenom na rozlíšenie 640x480 bodov. Túto obrazovku budeme prekresľovať od 20 do 500-krát ako bude uvedené vo výsledkových tabuľkách. Pod jednotlivými nameranými hodnotami (10-krát pre každý beh) uvedieme priemer za beh, priemer na pixel za beh a priemer na pixel za všetky behy.

Najskôr uvedieme výsledky zobrazovania v jazyku C, potom budú nasledovať výsledky zobrazovania v SOJ.

Rýchlosť vykresľovania pixelov programom v jazyku C

V tabuľke uvádzame výsledky nasledovného programu, v ktorom je zahrnutá inicializácia grafického režimu a testovanie jeho správneho nastavenia. Je to podstatná časť kódu. Nie je tu uvedený časovač, ktorý bol použitý.

```
int gdriver = DETECT, gmode, errorcode;
  initgraph(&gdriver, &gmode, "e:\\borlandc\\bgi");
  errorcode = graphresult();

  if (errorcode != grOk)
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();

    exit(1);
  }
  for(int k=0;k<20;k++)                    //20 – počet obrazoviek (mení sa až po 500)
      for(i=0;i<640;i++)
           for(int j=0;j<480;j++)
             putpixel(i,j,k);
```

Zobrazovanie pixelov v C pre vzorku 640x480					
	500-krát	**100-krát**	**80-krát**	**50-krát**	**20-krát**
	1452,214289	289,2278579	232,1228577	144,9321432	57,857143
	1452,214289	289,2278579	232,1228577	144,9321432	57,857143
	1452,214289	289,2278579	232,1228577	144,9321432	57,857143
	1448,157246	289,2278579	232,1228577	144,9321432	57,857143
	1448,157246	289,2278579	232,1228577	144,9321432	57,857143
	1448,157246	289,2278579	232,1228577	144,9321432	57,857143
	1448,157246	289,2278579	232,1228577	144,9321432	57,857143
	1448,157246	289,2278579	232,1228577	144,9321432	57,857143
	1453,657128	289,2278579	233,1285772	145,9343215	57,857143
	1453,657128	289,2278579	233,1285772	145,9343215	57,857143
Priemer	1450,474335	289,2278579	232,3240016	145,1325789	57,857143
Priemer na pixel ms	0,009443192	0,009414969	0,009453288	0,009448736	0,009416853
Priemer za všetky behy	0,009435408				

Grafická knižnica jazyka C nie je preslávená výkonmi, čo sa potvrdilo aj v tomto teste. Aj keď vykreslenie jedného pixelu sa môže zdať relatívne rýchle, v prípade, že vykresľujeme väčší počet bodov, funkcia putpixel nie je použiteľná. Ako ukážka môžu slúžiť hodnoty v prvom stĺpci – vykreslenie 500 obrazoviek trvalo viac ako 24 minút! (testy naozaj prebehli). Ak chceme v C vykresľovať, musíme sa uchýliť k iným taktikám, napr. priame zapisovanie do videopamäte, videostránok a podobne. V týchto prípadoch však musíme v určitom kroku práce použiť strojovo orientovaný jazyk (testy prebehli, ich aspoň čiastočné porovnanie nájdete v časti „porovnanie rýchlosti zobrazovania v C a SOJ“).

Rýchlosť vykresľovania pixelov programom v SOJ

Uvádzame zodpovedajúci kód v strojovo orientovanom jazyku podľa predchádzajúceho programu. Ani tu nie je uvedený časovač, ktorý bol nastavovaný pred vykreslením pixelu a po jeho zobrazení.

```
mov ah,0        ;nahodime graficky mod
mov al,13h
int 10h

mov x,0         ;urcuju suradnice pixelu
mov y,0

 mov cx,20      ;pocet vykresleni obrazoviek
chod:
          mov bx,cx
           mov x,        ;urcuju suradnice pixelu
           mov y,0

mak:
makaj:                   ;postupne kreslime cez x a y
         mov ax,40960    ;inicializacia pre vykreslovanie
```

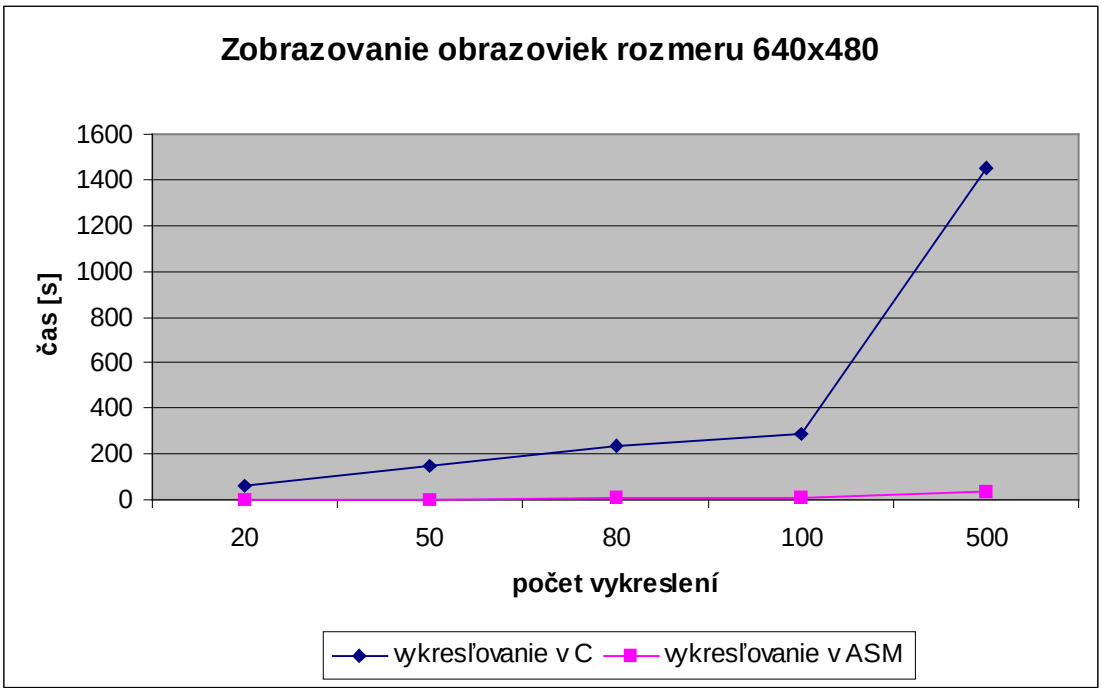
```
        mov es,ax
        mov di,0
        mov di,y
        mov ax,di

        mov cl,6
        shl di,cl

        mov cl,8
        shl ax,cl
        add di,ax
        add di,x
        mov al,bl          ;farba pixelu v bl
        mov es:[di],al

        inc x
        cmp x,640
        jne makaj        ;fungujeme v dvoch cykloch

        mov x,0
        inc y
       cmp y,480
        jne mak
        mov cx,bx
        loop chod

mov ah,0                 ;nazad nahodime povodny mod
mov al,2h
int 10h
```

Zobrazovanie pixelov v SOJ pre vzorku 640x480					
	500-krát	100-krát	80-krát	50-krát	20-krát
	34,94505495	6,978021978	5,604395604	3,516483516	1,373626374
	34,94505495	6,978021978	5,604395604	3,516483516	1,373626374
	34,94505495	6,978021978	5,604395604	3,516483516	1,373626374
	34,94505495	6,978021978	5,604395604	3,516483516	1,373626374
	34,94505495	6,978021978	5,604395604	3,516483516	1,373626374
	34,94505495	6,978021978	5,604395604	3,461538462	1,373626374
	34,94505495	7,087912088	5,604395604	3,461538462	1,373626374
	34,94505495	7,087912088	5,604395604	3,461538462	1,428571429
	35,05494505	7,087912088	5,604395604	3,461538462	1,428571429
	35,05494505	7,142857143	5,604395604	3,461538462	1,428571429
Priemer	34,96703297	7,027472527	5,604395604	3,489010989	1,39010989
Priemer na pixel ms	0,00022765	0,000228759	0,000228043	0,000227149	0,000226255
Priemer za všetky behy	0,000227571				

Aj v programoch v strojovo orientovanom jazyku existujú viaceré možnosti vykresľovania pixelov na obrazovku. Ich princíp však zostáva približne rovnaký, ako bol načrtnutý v predchádzajúcom kóde. Inicializujeme grafický mód, inicializujeme potrebné premenné, nastavíme sa na začiatok videopamäte (mov ax, 4096), zistíme, na ktorú pozíciu začíname vykresľovať a na dané miesto zapíšeme farbu pixelu. Po ukončení zápisu opustíme grafický mód.

Porovnanie rýchlosti zobrazovania pixelov programom v jazyku C a SOJ

Zistené časy, ktoré určujú dobu trvania zobrazenia bodu obrazovky po vykonaní oboch programov sme znázornili vo forme grafu.



Z grafu je evidentné zrýchlenie vykresľovania programom v strojovo orientovanom jazyku (ASM) oproti času vykresľovania programom v jazyku C. Dosahuje približne 41-násobok. V jazyku C však existujú aj iné možnosti vykreslenia, ako bolo spomínané vyššie. Pri použití týchto spôsobov bolo zrýchlenie programov v SOJ iba 3,5-násobné, čo je obrovský rozdiel. V zásade však platí, že ak sa snažíme zobraziť väčšie množstvo pixelov (napr. vykresľujeme obrázok), program v jazyku C stráca na výkone a do popredia sa dostáva strojovo orientovaný program. V C programe je preto výhodné zadefinovať funkciu, resp. službu operačného systému na zobrazenie pixelov.

Programy v jazyku C boli realizované v prostredí Borlandc. Strojovo orientované programy boli spracované prekladačom TASM a linkerom TLINK.

Jana PARÍZKOVÁ
Fakulta informatiky a informačných technológií STU
Bratislava

ZOBRAZOVANIE ČÍSEL V PEVNEJ RÁDOVEJ ČIARKE

Informatika v počítači – to je predovšetkým práca s číslami. Aritmetické operácie s nimi nie sú v princípe zložité, no už jednoduché „ručné“ spočítanie dvoch čísel robí študentom problémy: bez použitia kalkulátora často robia chyby. S odpočítavaním je to ešte horšie, niekedy nedokážu nájsť v primeranom čase správny výsledok takej operácie. Preto tu chceme poskytnúť vyučujúcim i študentom niekoľko názorných príkladov, ako to robí počítač, v ktorom sa čísla (okrem iných zobrazení) zapisujú vo formáte, ktorý označujeme ako **pevná rádová čiarka**. Článok poskytuje dosť námetov na praktické precvičovanie problematiky.

Je známe, že v počítači sa nerobí operácia odčítania samostatnou odčítačkou, ale volí sa také zobrazenie čísel, ktoré umožní použiť bežnú sčítačku aj na odčítanie. Tu uvedieme tri spôsoby zobrazenia, pre ktoré sa zaužívali názvy **priamy, doplnkový a inverzný kód**.

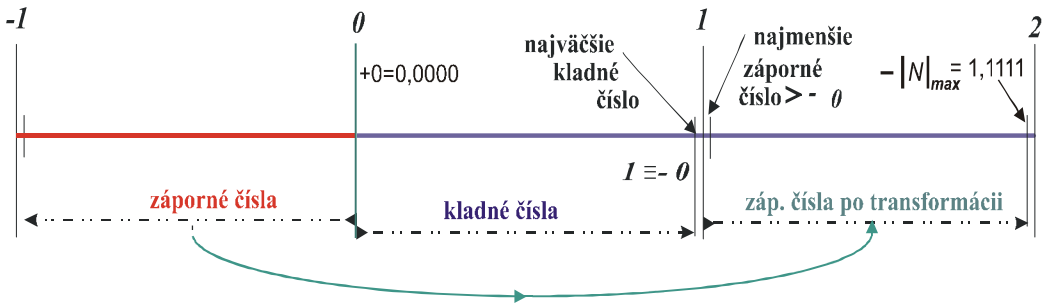
1. Priamy kód

Ide o zobrazenie informačného slova typu číslo, v pevnej rádovej čiarke, označme ho x, pre ktoré platí obmedzenie

$$|x| < 1 \qquad (1)$$

$$x_{PR} = \begin{cases} x, & \text{pre } x \geq 0 \\ 1 - x, & \text{pre } x \leq 0 \end{cases} \qquad (2)$$

Z uvedeného vyplýva, že číslo, napr. + *5/16* bude v dvojkovom zápise 0, 0101, a číslo – *5/16* bude vyjadrené 1, 0101. Prvý bit vľavo od rádovej čiarky sa označuje ako znamienkový, a ako vidno, kladné číslo má v ňom nulu, záporné jednotku, bity mantisy sú zhodné. Takéto zobrazenie čísel priamym kódom sa používa na ukladanie čísel do pamäti. Presvedčte sa na číselnej osi *(obr. 1)*, že zobrazenie jednotky je zhodné s obrazom zápornej nuly.



Obr. 1: Číselná os k priamemu kódu

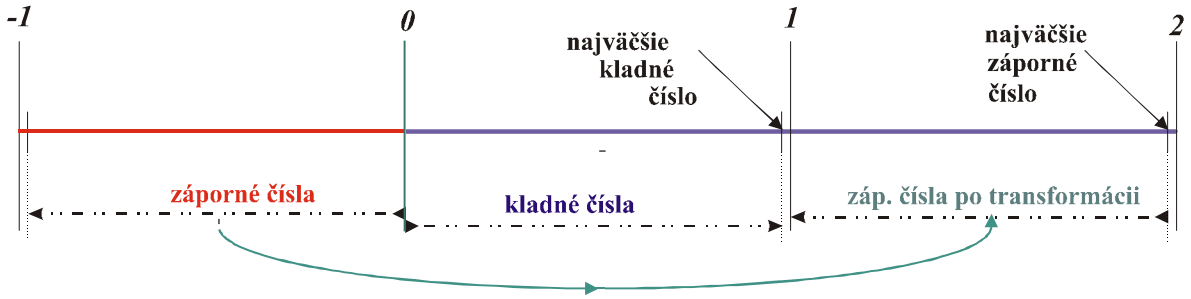
2. Doplnkový kód

Ide opäť o zobrazenie informačného slova typu číslo v číselnej sústave o základe 10, v pevnej rádovej čiarke, označme ho x, pre ktoré platí obmedzenie

$$|x| < 1 \qquad (3)$$

$$x_{cno} = \begin{cases} x, & \text{pre } x \geq 0 \\ 10 + x, & \text{pre } x < 0 \end{cases} \qquad (4)$$

Uvedené vzťahy vedú na zobrazenie číselnej osi podľa obr. 2. Doplnkový kód kladného čísla x je opäť rovný tomu kladnému číslu x. Doplnok záporného čísla dostaneme tak, že do znamienkového bitu zapíšeme hodnotu 1, na všetkých ďalších nahradíme jednotky nulami a nuly jednotkami a k najnižšiemu rádu pripočítame jednotku. Je to priamy dôsledok vzťahu (4.) a zároveň ukazuje, odkiaľ kód dostal svoj názov: obraz záporného čísla je totiž doplnkom do základu sústavy (teda tu do dvojky).



Obr. 2: Číselná os k doplnkovému kódu

Ako vidno, nula je vždy kladná +0,00000, maximálne záporné číslo je 1,00000 a nemá kladný ekvivalent. Dá sa dokázať, že súčet dvoch čísel v doplnkovom kóde dáva opäť doplnok súčtu: $x_{dop} + y_{dop} = (x + y)_{dop}$.

Príklad 1

x =	**0,1101**	x_{cno} = **0,1101**
y =	**0,0001**	y_{cno} = **0,0001**
$x+y$ =	**0,1110**	$(x+y)_{cno}$ = **0,1110**

Príklad 2

x = +	**0,1101**	x_{cno} = **0,1101**
y = -	**0,0001**	y_{cno} = **1,1111**
$x+y$ =	**0,1100**	$(x+y)_{cno}$ = **10,1110**

neuvažuje sa

Príklad 3 $x = -\mathbf{0,1101}$ $x_{cno} = \mathbf{1,0011}$

$y = +\mathbf{0,0001}$ $y_{cno} = \mathbf{0,0001}$

$x+y = \mathbf{0,1110}$ $(x+y)_{cno} = \mathbf{1,0100}$

Príklad 4 $x = -\mathbf{0,1101}$ $x_{cno} = \mathbf{1,0011}$

$y = -\mathbf{0,0001}$ $y_{cno} = \mathbf{1,1111}$

$x+y = -\mathbf{0,1110}$ $(x+y)_{cno} = \mathbf{11,0010}$

neuvažuje sa ⇑

Príklad 5 $x = -\mathbf{0,1101}$ $x_{cno} = \mathbf{1,0011}$

$y = -\mathbf{0,0111}$ $y_{cno} = \mathbf{1,1001}$

$x+y = -\mathbf{1,0100}$ $(x+y)_{DOP} = \mathbf{10,1100}$

⇑ indikuje, že ide o kladný výsledok

Príklad 5 ukazuje prípad vzniku chybného výsledku, ktorý by mal byť isto záporný, lebo oba sčítance sú záporné. Táto chyba vznikla pretečením číselného rozsahu bitov mantisy do znamienkového bitu. Pripomeňme, že ak platí vzťah (3) pre sčítance, musí platiť aj pre ich súčet. (V tomto príklade sa hľadal súčet čísel –13/16 a –7/16 čo je –20/16 a je zrejmé že na to nestačia 4 bity mantisy, ale musí ich byť aspoň päť. Preto vzniklo pretečenie a výsledok je nepoužiteľný.)

3. Inverzný kód

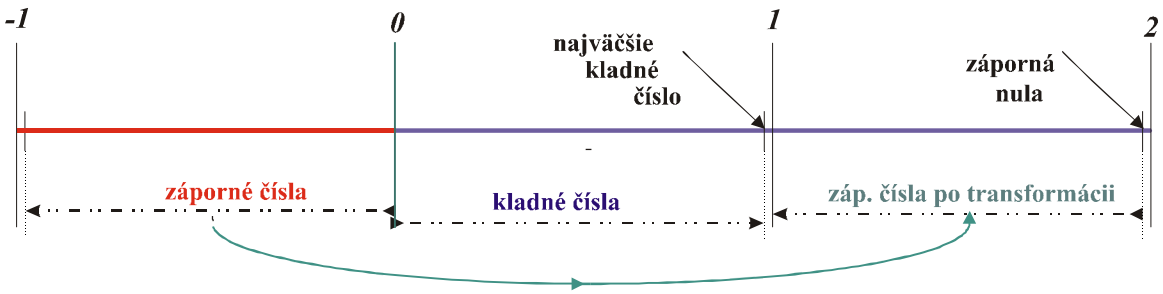
Rovnako ako vyššie ide o zobrazenie informačného slova typu číslo v číselnej sústave o základe *10*, v pevnej rádovej čiarke, označme ho *x*, pre ktoré platí obmedzenie

$$|x| < 1 \qquad (5)$$

$$x_{hm} = \begin{cases} x, & \text{pre } x \geq 0 \\ 10 + x - 1.10^{-m}, & \textbf{pre } x \leq 0 \end{cases} \qquad (6)$$

Ak porovnáte vzťahy (4 a 6) ľahko zistíte, že inverzný kód pre záporné *x* dáva číslo, ktoré je o jednotku najnižšieho rádu menšie než u doplnkového kódu: nie je to doplnok do základu ústavy, ale do základu sústavy zmenšeného o jednotku najnižšieho (*n-tého*) rádu. Pre dvojkovú sústavu ide o doplnok do čísla $\boldsymbol{10 - 1.10^{-n} = 1,111...1.}$

Uvedené vzťahy vedú na zobrazenie číselnej osi podľa *obr. 3.* Inverzný kód kladného čísla *x* je opäť rovný tomu kladnému číslu *x*. Inverzný kód záporného čísla dostaneme tak, že vo všetkých bitoch nahradíme jednotky nulami a nuly jednotkami. Je to priamy dôsledok vzťahu (6) a zároveň ukazuje, odkiaľ kód dostal svoj názov: obraz záporného čísla vznikne totiž inverziou pôvodných hodnôt v jednotlivých bitoch.



Obr. 3: Číselná os k inverznému kódu

Kladná nula dáva obraz 0,00000, záporná nula 1,11111. Pre operáciu sčítania v inverznom kóde musíme uvažovať aj takýto možný prípad:

$$\begin{array}{rr} & 10 + x - 1.10^{-m} \\ + & 10 + y - 1.10^{-m} \\ \hline & 100 + (x+y) - 2.10^{-m} \end{array} \qquad (7)$$

Ľahko poznáme, že vzťah (7) nie je zhodný so vzťahom (6) definujúcim inverzný kód, ale že vznikol iný kód, ktorý predstavuje o dve jednotky menšie číslo než má mať obraz inverzného kódu. Preto musíme v takomto prípade robiť korekciu, t.j. pripočítať k výsledku jednu jednotku najnižšieho rádu, aby aj pre vyjadrenie súčtu bol zachovaný rovnaký definičný vzťah. Takáto korekcia sa robí formou tzv. **kruhového prenosu,** keď sa využije fakt, že v znamienkovom bite vznikol prenos do vyššieho rádu, ktorý možno pripočítať k najnižšiemu rádu.

Príklad 6

$x = 0,1101$ $\quad x_{inv} = 0,1101$

$y = 0,0001$ $\quad y_{inv} = 0,0001$

$x+y = 0,1110$ $\quad (x+y)_{inv} = 0,1110$

Príklad 7

$x = -0,1101$ $\quad x_{inv} = 1,0010$ $\quad 1,1111 (ZÁKL-10^{-4})$

$y = +0,0001$ $\quad y_{inv} = 0,0001$ $\quad -0,1101$ $\quad (X)$

$x+y = 0,1110$ $\quad (x+y)_{inv} = 1,0011$ $\quad 1,0010$ $\quad (X_{inv})$

inverzia

Príklad 8

$X = +0,1101$ $\quad x_{inv} = 0,1101$

$Y = -0,0001$ $\quad y_{inv} = 1,1110$

$x+y = 0,1100$ $\quad (x+y)_{inv} = 10,1011$

kruhový prenos 1

$(x+y)_{inv} = 0,1100$

Príklad 9

$x = -0,1101$ $\quad x_{inv} = 1,0010$

$y = -0,0001$ $\quad y_{inv} = 1,1110$

$x+y = -0,1100$ $\quad (x+y)_{inv} = 11,0000$

kruhový prenos 1

$(x+y)_{inv} = 1,0001$

4. Modifikované doplnky

Všimnime si teraz doplnkový kód. Povedali sme, že nula je tu vždy kladná: 0,000...00 a záporná nula nie je definovaná. Vzhľadom na to a tiež ak vieme, že je jediný bit pre znamienko, je celkový rozsah zobrazení záporných čísel väčší než rozsah kladných čísel a to o jednotku najnižšieho rádu. Maximálne záporné číslo 1,00...00 totiž nemá kladný ekvivalent: jeho pravý dvojkový doplnok je opäť 1,00...00, t.j. zase záporné číslo alebo číslo kladné, v ktorom sa prekročil (pretiekol) rozsah, vymedzený pre kladné čísla.

Rozsah môže pretiecť tiež pri spočítaní dvojice kladných alebo záporných čísel a prejaví sa to tým, že pri spočítaní dvojice kladných čísel dostávame v znamienkovom bite jedničku (čo je znak záporného čísla) a naopak, pri spočítaní dvoch záporných čísel nulu, (čo je znak kladného čísla). Ak chceme mať korektný výsledok, museli by sme si pamätať znamienka sčítancov a porovnávať ich so znamienkom výsledku. Lepšie je však pracovať s tzv. **modifikovanými doplnkami**.

Hlavná myšlienka spočíva v tom, že za základ číselnej sústavy zoberieme jej vyššiu mocninu: namiesto 10 (dvojka) budeme pracovať so základom 100 (štvorka). Ak zobrazujeme v súlade s predpokladom číslo x < 1, potom doplnok sa vytvára do štvorky a prestavuje dva bity. Kladné číslo má potom oba tieto bity nulové (00,...) a záporné jednotkové (11,...). Prípad´, v ktorom v znamienkových bitoch vznikne podoba (01,...) alebo (10,...) indikuje **stav pretečenia**. Znamienko v tomto prípade určuje vyšší znamienkový rád, a nižší bit v tej dvojici sa označuje aj ako rád pretečenia.

V uvažovanom prípade potom kódové obrazy vytvoríme podľa vzťahov (8) pre doplnkový kód a (9) pre inverzný kód:

$$x_{cno} = \begin{cases} x, & \text{pre } x \geq 0 \\ 100 + x, & \text{pre } x < 0 \end{cases} \qquad (8)$$

$$x_{Imu} = \begin{cases} x, & \text{pre } x \geq 0 \\ 100 + x - 1.10^{-m}, & \text{pre } x \leq 0 \end{cases} \qquad (9)$$

Príklad 10

x = - **0,1101** x_{cno} = **11,0011**

y = - **0,0001** y_{cno} = **11,1111**

$x+y$ = - **0,1110** $(x+y)_{cno}$ = **11,0010**

neuvažuje sa

Príklad 11

x = – **0,1101** x_{Imu} = **11,0010**

y = – **0,0001** y_{Imu} = **11,1110**

$x+y$ = – **0,1110** $(x+y)_{Imu}$ = **111,0000**

kruhový prenos +**1**

$(x+y)_{Imu}$ = **11,0001**

5. Desiatkový a deviatkový doplnok

V desiatkovej sústave sú operácie s číslami celkom analogické. Pre doplnkový kód platia vzťahy (3) a (4) s tým, že sa za základ sústavy volí desiatková desiatka. Kódový obraz záporného čísla je teda **desiatkový** doplnok.

Príklad 12 x = + **0,4257** x_{cno} = **0,4257** **10,0000** (základ)

y = - **0,0316** y_{cno} = **9,9684** - **0,0316** (y)

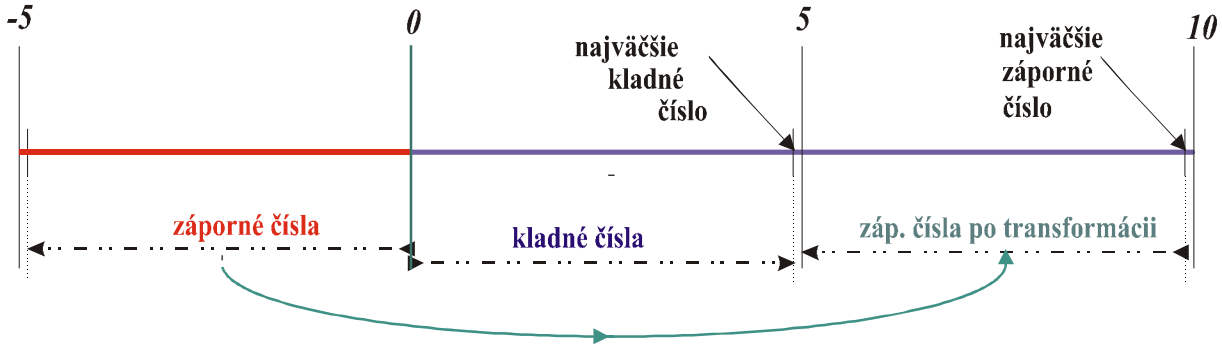
$x+y$ = + **0,3941** $(x+y)_{DOP}$ = **10,3941** **9,9684** (y_{cno})

neuvažuje sa desiatkový doplnok

Všimnite si, že kladné číslo má v znamienkovom mieste nulu, záporné deviatku. V desiatkovej sústave sa však používa desať číslic 0, 1, 2, ...9, čo umožňuje uvoľniť obmedzenie podľa vzťahu (3) takto:

$$I\,x\,I < 5 \qquad (10)$$

čo rozdeľuje číselnú os na dva intervaly: ‹0, 5) pre kladné čísla a ‹5, 10) pre čísla záporné, pozri obr. 4. Nula je vždy kladná (+0,00...00). Najväčšie záporné číslo je 5,00...00 a nemá kladný ekvivalent.



Obr. 4: Číselná os pre desiatkové doplnky

Príklad 13 x = + **4,4257** x_{cn} = **4,4257** **10,0000** (základ)

y = - **2,6966** y_{cno} = **7,3034** - **2,6966** (y)

$x+y$ = + **1,7291** $(x+y)_{cno}$ = **11,7291** **7,3034** (y_{cno})

neuvažuje sa

Vo všeobecnosti platí, že číslice 0,1,2,3,4 zapísané v znamienkovom mieste indikujú číslo kladné, číslice 5,6,7,8,9 znamenajú že ide o číslo záporné, vyjadrené desiatkovým doplnkom. Ale uvedomme si tiež, že v týchto úvahách sa ukazuje, že číslica zapísaná v znamienkovom mieste (prvé vľavo od rádovej čiarky), v zmysle vzťahu (10) už nenesie len informáciu o znamienku daného čísla, ale spoluvytvára celú jeho číselnú hodnotu. **To je dôležité pre spätný prevod do priameho kódu.**

Príklad 14

x = – **4,265** x_{cn} = **5,745** **10,000** (základ) **10,000** (základ)

y = – **3,431** y_{cno} = **6,569** –**3,431** (y) - **4,265** (x)

$x+y$ = – **7,696** $(x+y)_{cno}$= **12,314** **6,569** (y_{cno}) **5,745** (x_{cno})

indikuje kladné číslo, takže ide o **chybu pretečením.**

Inverzný kód v desiatkovej sústave nazývame **deviatkovým doplnkom**. Vyplýva to rovno zo vzťahu (9) : 10 – 1. $^{0/}\vdash^{m}$ = 9,999...99. Tu je skrytý postup tvorby kódových obrazov.

Príklad 15

x = - 3,4253	x_{hmu} = 6,5746	9,9999	(základ–$^{0/}$)
y = +1,1926	y_{hmu} = 1,1926	–3,4253	(x)
$x+y$ = –2,2327	$(x+y)_{hmu}$ = 7,7572	6,5746	(x_{hmu})

deviatkový doplnok

Výsledok je – ako vidno – záporný a je to teda deviatkový doplnok záporného čísla, ktoré je súčtom oboch pôvodných čísel vyjadrených v inverznom kóde. Ak urobíte spätný deviatkový doplnok z tohoto doplnku, dostanete „viditeľnú“ kontrolu toho, že nedošlo k žiadnej chybe:

– 9,9999 + 7,7572 = –2,2327.

Pripomeňme, že vo všeobecnosti treba robiť korekciu pri súčte v tom istom duchu ako pri sústave dvojkovej a uplatniť kruhový prenos.

Príklad 16

x = – 4,3425	x_{hmu} = 5,6574	9,9999	(**základ**–$^{0/}$)
y = – 3,4192	y_{hmu} = 6,5807	–3,4192	(Y)
$x+y$ = – 7,7617	$(x+y)_{hmu}$ =12,2381	6,5807	(y_{hmu})
		+1	
		2,2382	

indikuje kladné číslo, takže ide o **chybu pretečením.**

Príklad 17

x = – 1,5412	x_{hmu} = 8,4587	9,9999	(**základ**–$^{0/}$)
y = – 1,0374	y_{hmu} = 8,9625	– 1,4587	(x)
$x+y$ = – 2,5789	$(x+y)_{hmu}$ =17,4212	8,5412	(x_{hmu})
		+1	
		7,4213	znamienko aj výsledok je správny

Uvedené pojmy sú jednoduché, sú vhodnou témou na cvičenia s číslami v pevnej rádovej čiarke a v podobnom duchu pripravíme i témy s číslami v pohyblivej rádovej čiarke. Teraz však treba objasniť i prácu s číslami x, ktoré nespĺňajú podmienku IxI<1 danú napr. vzťahom (1), ale sú v absolútnej hodnote väčšie ako 1. Je celkom prirodzené, že i takéto čísla musia byť počítačom spracovateľné aj v pevnej rádovej čiarke.

Problém nie je nijak zložitý ak zavedieme pojem *merná jednotka*. Uvažujme napr. číslo $x = +13/16$. Čitateľ tohto zlomku, t.j. číslo *13* je väčšie ako *1* a výraz môžeme čítať ako *trinásť šestnástin*. Menovateľ *16* je tiež väčší ako *1*, ale hodnota zlomku *13/16* je určite menšia ako *1*, lebo menovateľ je väčší ako čitateľ. Z toho plynie, že číslo *13* bolo transformované do podoby vyhovujúcej vzťahu (1) pre zobrazenie v pevnej rádovej čiarke a mernou jednotkou sa stali *šestnástiny*. Dvojkovo sa uvažované $x = +13/16$ zapíše napríklad v priamom kóde ako 0,1101. Uvedomte si, že váhy v bitoch napravo od rádovej čiarky postupne vyjadrujú *poloviny, štvrtiny, osminy, šestnástiny*..., t.j. 2^{-0}, 2^{-1}, 2^{-2}, 2^{-3}, ... , 2^{-k} , kde k je počet bitov napravo od rádovej čiarky a predstavuje mernú jednotku.

V praktických úlohách treba voliť takú hodnotu k, ktorá by minimalizovala možnosť vzniku pretečenia pri realizácii operácií, v spracovaní počítačom sa automaticky volí tá merná jednotka, ktorá prislúcha poslednému pravému bitu v slove.

Objasnime ešte pojem mernej jednotky pre desiatkovú sústavu. Nech je číslo $x = 653$. Číslo samo o sebe je bezrozmerné, ale predpokladajme, že predstavuje finančnú hodnotu vyjadrenú v korunách. (Mohli by sme tiež hovoriť, že ide o *653* jabĺk, *653* rokov, *653* litrov, atď., ale z hľadiska mernej jednotky sú peňažné názvy najnázornejšie.) Môžeme zapísať takéto vyjadrenia:

		merná jednotka
653	korún	jednotky
65,3	desaťkorún	desatiny
6,53	stokorún	stotiny
0,653	tisíckorún	tisíciny
0,0653	desaťtisíckorún	desaťtisíciny
0,00653	stotisíckorún	stotisíciny

Číselný zápis je vo všetkých riadkoch rovnaký, ale formálny zápis vyhovujúci podmienke vzťahu (1) je vhodný až od štvrtého riadku zoznamu. Teda mernou jednotkou musia byť prinajmenšom *tisíciny* alebo ešte menšie hodnoty, aby bolo možné vyjadriť x v pevnej rádovej čiarke v niektorom z preberaných kódov. Procesu prepisu čísla do podoby z vhodnou mernou jednotkou sa hovorí **zmena mierky.** V praktickej podobe tomu zodpovedá *posuv čísla doprava* o potrebný počet miest (v registri, v ktorom je číslo uložené). Po vykonaní operácie sa treba vrátiť k pôvodnej mierke.

Príklad 18

Vypočítajte, koľko rokov má človek v roku 2005, keď sa narodil v roku 1962.

Označme $x' = 2005$, $y' = -1962$, zmeňme mierku posuvom čísel o 4 miesta doprava a použime napríklad doplnkový kód. Potom

$x = +$ **0,2005**	$x_{cno} =$ **0,2005**	**10,0000** (základ)
$y = -$ **0,1962**	$y_{cno} =$ **9,8038**	**- 0,1962** (y)
$x+y = +$ **0,0043**	$(x+y)_{cno} =$ **10,0043**	**9,8038** (y_{cno})
	neuvažuje sa	

Výsledok operácie je v prostrednom stĺpci, ide o číslo kladné a po návrate k pôvodnej mierke možno odpovedať: v roku 2005 dosiahol človek z príkladu 43 rokov.

Prof. Ing. Ján KOLENIČKA, PhD.
Ing. Jarmila ŠKRINÁROVÁ, PhD.
Katedra informatiky FPV UMB
Banská Bystrica

kolenick@fpv.umb.sk
skrinar@fpv.umb.sk

EFEKTÍVNOSŤ PROGRAMOVACIEHO JAZYKA PRI VÝPISE ČÍSLA

V tomto príspevku sa zaoberáme porovnávaním programov v jazyku C a v strojovo orientovanom jazyku (ďalej len SOJ). Zameriame sa na rozdiel v rýchlosti spracovania kódu napísaného v týchto jazykoch, ktorý bude vypisovať na obrazovku celé čísla. Výsledky porovnáme a zhodnotíme.

Úvod

Zobrazovanie čísel je tiež jedna z možností ako otestovať efektívnosť dvoch programovacích jazykov, ktoré tieto čísla spracovávajú.

V predchádzajúcich príspevkoch „Porovnávanie operácie kopírovania v dvoch programovacích jazykoch" a „Porovnanie rýchlosti zobrazenia bodov obrazovky v dvoch programovacích jazykoch" sme sa zaoberali rýchlosťou spracovania údajov pri ich kopírovaní a rýchlosťou zobrazenia bodov obrazovky programom, zakódovaným v dvoch odlišných programovacích jazykoch. Porovnávali sme výsledky programov, ktoré boli napísané v jazyku C a v strojovo orientovanom jazyku (hovorovo asembler). V tomto príspevku nás bude zaujímať čas, potrebný na vytlačenie dvoch rádovo odlišných celých čísel.

Podobne ako v predchádzajúcich prípadoch sme zvolili počet opakovaní jednotlivých testov 10 pre každý beh (55555, 40000, 31765, 20000-krát). Pre účely porovnávania sme zvolili dve čísla **`60665`** a **`3578`**.

Výpis čísla `60665` programom v jazyku C

Po príklade krátkeho kódu pre 40000-krát opakované tlačenie celého čísla uloženého v premennej *cis* nasleduje tabuľka, kde sú uvedené časy výpisu celého čísla aj pre ďalší počet opakovania výpisu.

```
unsigned int cis = 60665;
for (i=0; i< 40000; i++)
        printf("%u",cis");
```

Zobrazovanie čísla 60665 v C				
	55555-krát	**40000-krát**	**31765-krát**	**20000-krát**
	15,769231	11,318681	8,901099	5,549451
	15,769231	11,318681	8,901099	5,549451
	15,769231	11,318681	8,901099	5,549451
	15,769231	11,318681	8,901099	5,549451
	15,769231	11,318681	8,901099	5,549451
	15,769231	11,318681	8,956044	5,549451
	15,714286	11,318681	8,956044	5,549451
	15,714286	11,263736	8,956044	5,549451
	15,714286	11,263736	8,956044	5,549451
	15,824176	11,153846	8,956044	5,549451
priemer	15,758242	11,2912085	8,9285715	5,549451
priemer na cifru v ms	0,056730239	0,056456043	0,056216411	0,05549451
priemer za všetky behy	0,056224301			

Z tabuľky sa dá vyčítať viacero zaujímavých výsledkov:

1. Pri jednom teste sa vyskytovalo len málo rozdielnych hodnôt, pri teste s 20000 opakovaniami dokonca ani jedna rozdielna hodnota. Pri dokonalom prostredí a spôsobe vykonávania by mal test vždy trvať rovnako. Z týchto meraní je vidieť, že jazyk C má relatívne rovnaký výkon aj pri opakovanom spustení programu.
2. Jazyk C nestráca na výkone ani so vzrastajúcim počtom opakovaní (je to viditeľné v riadku 'priemer na cifru v ms', kde ani s narastajúcim počtom nenarastá doba zobrazenia jednej cifry čísla, alebo narastá len minimálne)

Výpis čísla `3578` programom v jazyku C

Kód je totožný s tým v predchádzajúcej kapitole, zmena je len v hodnote premennej *cis*. Keď sa pozrieme na priemer za všetky behy, priemer na jednu cifru čísla (jeden zobrazovaný znak) a porovnáme ju s touto hodnotou z predchádzajúcej kapitoly, zistíme, že výsledky sú takmer totožné - mierne rýchlejšie je zobrazovanie pri čísle `60665`, ale rozdiel je naozaj minimálny.

Zobrazovanie čísla 3578 v C				
	55555-krát	**40000-krát**	**31765-krát**	**20000-krát**
	12,802198	9,230769	7,197802	4,505495
	12,802198	9,230769	7,197802	4,505495
	12,802198	9,230769	7,197802	4,505495
	12,802198	9,230769	7,197802	4,505495
	12,802198	9,120879	7,197802	4,505495
	12,802198	9,120879	7,197802	4,505495
	12,802198	9,120879	6,978022	4,450549
	12,802198	9,175824	6,978022	4,450549
	12,802198	9,175824	6,978022	4,450549
	12,802198	9,175824	7,2527474	4,450549
priemer	12,802198	9,1813185	7,13736254	4,4835166
priemer na cifru v ms	0,057610467	0,057383241	0,056173167	0,056043958
priemer za všetky behy	0,056802708			

Aby bolo možné vykonať porovnávanie s výsledkami programu v jazyku strojovej úrovne, jednotlivé testy spriemerujeme, čím sa získa približná hodnota dĺžky trvania zobrazenia cifry programom v jazyku C. Spriemerovanie je uvedené v nasledujúcej tabuľke.

Spriemerovanie hodnôt pre C				
číslo 60665	0,056730239	0,056456043	0,056216411	0,05549451
číslo 3578	0,057610467	0,057383241	0,056173167	0,056043958
priemer za oba testy	0,057170353	0,056919642	0,056194789	0,055769234

Priemer je vykonávaný z priemerov na cifru v jednotlivých testoch.

Výpis čísla `60665` programom v SOJ

Kód k tejto časti je príliš dlhý, a preto nebude v tomto dokumente kompletne zahrnutý. Uvedieme niektoré významné časti. Napríklad nasledujúca časť kódu realizuje 40000-krát výpis čísla 60665 pomocou procedúry *vypcisl*.

```
;vypis 40000 krat cislo 60665
  mov cx,40000
makaj:
  xor dx,dx
  mov ax,60665
  push cx
  call vypcisl
  pop cx
  loop makaj
```

Procedúra *vypcisl*, ktorá zabezpečí výpis čísla po cifrách na obrazovku je definovaná v nasledujúcej nekompletnej časti programu

```
.model Tiny
stack 100H
.data
  txt0 db 'prva hodnota: $'
  txt1 db ' druha hodnota: $'
 zero dw (?)
 max dd (?)
 max2 dd (?)
 pp db (0)
 pom dw (?)
 zvysok dw (?)
 pom2 dw (?)
 prenos dw (0)

.code
;PROCEDURA VYPISU CISLA
proc vypcisl near
mov bx,65535
div bx
mov zvysok,dx
mov cx,ax
xor dx,dx
xor bx,bx
mov bx,10
mov ax,65535
mov pom,65535
mov prenos,0
mov pp,0
rob:
   inc pp
   div bx                ;delim 65535/10
   mov pom,ax            ;podiel 6553 do pom
   mov ax,dx             ; zvysok do ax
   mul cx                ;vynasobi, 5*cx,cx-kolkokrat sa 65535 naxadza
                          ;v cisle
   mov pom2,ax         ;odlozim si vysledok
   xor dx,dx
   mov ax,zvysok         ;zvysok do ax,potrebujem iba desiatky-> div bx
   cmp ax,0
   je potom
   div bx
   add pom2,dx           ;pridam do pom2
   mov zvysok,ax
potom:
   mov ax,prenos
   add pom2,ax
   xor ax,ax
```

```
   xor dx,dx
   mov ax,pom2
   div bx
   mov prenos,ax
   push dx
   xor dx,dx
   mov ax,pom
   cmp ax,0
 jne rob
xor bx,bx
xor dx,dx
mov ax,prenos
xor cx,cx
dass:
inc cl
mov bx,10
div bx
push dx
xor dx,dx
cmp ax,0
jne dass
mov zero,0
add cl,pp
xor bx,bx
mov bx,1
xor dx,dx
pak:
  pop ax
  div bl
  mov dl,al
  add dl,48
  cmp zero,1
  je VYP
  cmp dl,'0'
  je NULA
VYP:
  mov ah,02h
  int 21h
  mov zero,1
NULA:
  loop pak
ret
 vypcisl endp
;koniec procedury vypcisl

start:
 mov ax,@data
 mov ds,ax
 xor ax,ax
 mov ah,0        ;zapneme casovac a ziskame hodnotu
 int 1Ah
 mov ax,dx
 mov dx,cx
 mov word ptr [max],ax             ;tu si zatial ulozime
 mov word ptr [max]+2,dx

     .
     .
     .
```

;program vypisuje postupne cifry daného celého čísla a nakoniec vypíše
;hodnotu časovača po ukončení výpisu celého čísla

```
    .
    .
    .
mov ax,4c00h
int 21h
end start
```

Zobrazovanie čísla 60665 v SOJ				
	55555-krát	40000-krát	31765-krát	20000-krát
	2,307692308	1,648351648	1,318681319	0,824175824
	2,307692308	1,648351648	1,318681319	0,824175824
	2,307692308	1,648351648	1,318681319	0,824175824
	2,307692308	1,648351648	1,318681319	0,824175824
	2,307692308	1,648351648	1,318681319	0,824175824
	2,307692308	1,648351648	1,318681319	0,824175824
	2,307692308	1,703296703	1,318681319	0,824175824
	2,362637363	1,703296703	1,318681319	0,824175824
	2,362637363	1,703296703	1,318681319	0,824175824
	2,362637363	1,703296703	1,318681319	0,824175824
priemer	2,324175824	1,67032967	1,318681319	0,824175824
priemer na znak v ms	0,008367117	0,008351648	0,008302731	0,008241758
priemer za všetky behy	0,008315814			

Výpis čísla 3578 programom v SOJ

Uvádzame len výsledky po spustení programu v strojovo orientovanom jazyku.

Zobrazovanie čísla 3578 v SOJ				
	55555-krát	40000-krát	31765-krát	20000-krát
	1,593406593	1,098901099	0,879120879	0,549450549
	1,593406593	1,098901099	0,879120879	0,549450549
	1,593406593	1,098901099	0,879120879	0,549450549
	1,593406593	1,098901099	0,879120879	0,549450549
	1,593406593	1,098901099	0,879120879	0,549450549
	1,593406593	1,098901099	0,879120879	0,549450549
	1,593406593	1,098901099	0,879120879	0,549450549
	1,593406593	1,098901099	0,879120879	0,549450549
	1,593406593	1,098901099	0,879120879	0,549450549
	1,593406593	1,153846154	0,879120879	0,549450549
priemer	1,593406593	1,104395604	0,879120879	0,549450549
priemer na cifru v ms	0,007170401	0,006902473	0,006918943	0,006868132
priemer za všetky behy	0,006964987			

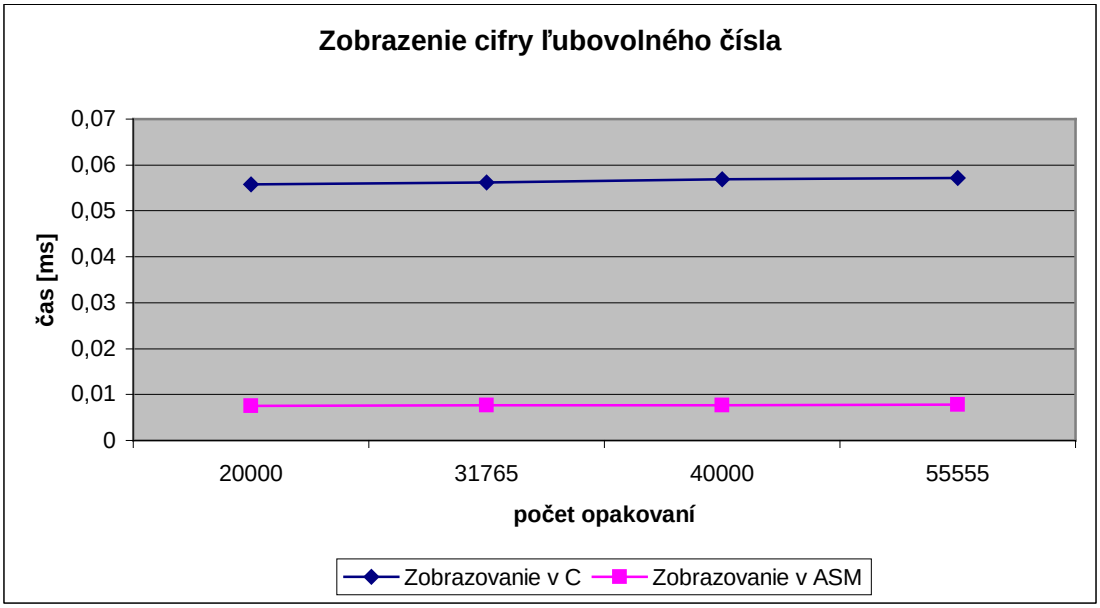
Aby sme získali relevantné výsledky, aj hodnoty dosiahnuté programom v SOJ budú spriemerované a tento priemer nasledovne porovnaný s údajmi získanými z programu v jazyku C.

Spriemerovanie hodnôt pre SOJ				
číslo 60665	0,008367117	0,008351648	0,008302731	0,008241758
číslo 3578	0,007170401	0,006902473	0,006918943	0,006868132
priemer za oba testy	0,007768759	0,00762706	0,007610837	0,007554945

Spriemerované sú priemery na cifru z jednotlivých testov (pre číslo `60665` aj pre `3578`). Je zaujímavé, že tieto hodnoty sa tak výrazne líšia (pre číslo `60665` je priemer `0,008315814`, pre číslo `3578` je `0,006964987`). Túto dosť veľkú odchýlku si nevieme vysvetliť.

Porovnanie rýchlosti výpisu čísel programom v jazyku C a v SOJ

V uvedenom grafe môžeme sledovať spriemerovaný čas výpisu celého čísla. Skratka ASM znamená asembler. Príslušná spodná čiara teda zakresľuje výsledky platné pre strojovo orientovaný program.



Z predchádzajúcich kapitol je jasné, ktorý jazyk je vo výpise celých čísel rýchlejší. V tomto smere vedie jednoznačne strojovo orientovaný jazyk, ktorý je výkonnejší približne 9,5-krát. V tomto teste sa však porovnávali len čísla do veľkosti `65535` (teda v rozsahu 16-bitového registra). Predpokladáme, že pri zobrazovaní čísel typu double alebo long by strojovo orientovaný jazyk dosahoval podobné, ak nie aj lepšie výsledky v porovnaní s jazykom C. Na druhej strane, ak sa pozrieme na dĺžku kódu, ktorý je nutné vytvoriť v oboch jazykoch, rozdiel je priepastný. V jazyku C na zobrazenie čísla stačí napísať jeden riadok, v strojovo orientovanom jazyku to môžu byť aj stovky! Aj keď je jazyk C v zobrazovaní relatívne pomalý, jednoduchosť kódu je v tomto prípade asi rozhodujúcejšia. Ak navyše nepotrebujeme v krátkom čase zobrazovať veľké kvantá dát, zvolíme použitie jazyka C.

Ako sme už naznačili v predchádzajúcich kapitolách a ako je viditeľné z grafov, oba jazyky si držia približne rovnakú výkonnosť pri zobrazovaní čísel, čo svedčí o veľmi dobrom výkone kompilátorov a spôsobe akým pracujú.

Na záver treba ešte dodať, že do určitej malej miery záleží od hodnoty zobrazovaného čísla, ale dôležitejší je počet cifier. V konečnom dôsledku oba jazyky rozdelia číslo na znaky, a tie sa potom zobrazujú. Obe čísla boli vybraté náhodne, s ohľadom na počet cifier.

Programy v jazyku C boli realizované v prostredí Borlandc. Strojovo orientované programy boli spracované prekladačom TASM a linkerom TLINK.

Literatúra

1. CAPEK, R.: TurboAssembler 3.0, Grada, 1992.

2. HEROUT, P.: Učebnice jazyka C, Kopp, České Budejovice, 1998.

Jana PARÍZKOVÁ
Fakulta informatiky a informačných technológií STU
Bratislava

VPLYV IKT NA ZÁUJEM ŽIAKOV O BIOLÓGIU Z POHĽADU UČITEĽOV

Úvod

Biológia sa stala jednou z najvýznamnejších vied súčasnosti. Tento rozmach vedie k zvyšovaniu biológie ako vedy aj ku zvyšovaniu úrovne výchovno-vzdelávacieho procesu. Pokiaľ sa v minulosti kládol dôraz na pasívne zvládnutie poznatkov, dnes už je vyučovanie zamerané na to, aby žiaci vedeli poznatky aplikovať v praxi. To si vyžaduje uplatniť vo vyučovaní také prostriedky, metódy a formy, ktoré vedú k zvyšovaniu efektívnosti a úrovne vyučovania (NAGYOVÁ – UŠÁKOVÁ 2003).

V súčasnej dobe existuje niekoľko možností ako niekoho niečo naučiť. Jedným spôsobom je klasický spôsob výučby, kde učiteľ vysvetľuje, skúša, opakuje... Tento spôsob výučby je v dnešnej dobe najrozšírenejší, ale pomaly vzrastá popularita vyučovania pomocou informačných a komunikačných technológií (IKT). Tento spôsob výučby má aj opodstatnenie v tom, že súčasná generácia študentov sa snaží nájsť materiál k štúdiu v elektronickej podobe (TURČÁNI – BAUEROVÁ 2001).

Vzdelávanie by sa malo oslobodiť od konzervatívnych a neproduktívnych tradícií, od časových a priestorových obmedzení. Škola prestáva byť hlavným zdrojom informácií a silno jej v tomto smere začínajú konkurovať atraktívnejšie média ako televízia, video... a elektronické zdroje ako sú osobné počítače, notebooky, internet... V súčasnosti sa snahy o zefektívnenie vzdelávania sústreďujú najmä na vyučovací proces, ktorý prebieha v triedach a miestnostiach upravených na praktické cvičenia (HORVÁTHOVÁ – VÍTKO 2003).

IKT sa najskôr zavádzali vo forme jednotlivých strojov v kabinetoch učiteľov, neskôr vo forme lokálnej a celoškolskej siete. Zo začiatku ako učitelia, tak aj žiaci využívali výpočtový potenciál strojov. V súčasnosti je snaha presunúť počítač do úlohy nástroja pre získavanie informácií, ich následné spracovanie a takisto využívanie na komunikáciu na lokálnej a medzinárodnej úrovni. Vyučovanie pomocou IKT mení tradičnú formu vzdelávacieho procesu, kde učiteľ prestáva byť iba odovzdávateľom informácií, usmerňuje cieľavedomé získavanie informácií a spolu so žiakom sa podieľajú na tvorbe, realizácii a hodnotení práce študenta.

Rozvoj IKT vyplýva z ich postavenia v spoločnosti a z ich vlastností – dokážu sa prispôsobiť takmer všetkým podmienkam a profesiám. Prudký rozvoj rôznych ľudských odvetví a hlad po informáciách sú hlavnou príčinou rýchleho rozvoja IKT. Pri vyslovení slova IKT máme na mysli najmä modernú techniku, napojenie na internet, výmenu informácií a komunikáciu na lokálnej, národnej a medzinárodnej úrovni, aplikácie CD ROM, spracovávanie informácií textovými, tabuľkovými a grafickými editormi... V súčasnosti existuje veľa možností pre zavádzanie IKT do vyučovania. Od využitia jednoduchej elektronickej pošty, cez diskusné kluby, on-line učebnice, vzdelávacie webové stránky, až po kompletné popisy študijných plánov. Študenti aj učitelia majú veľa možností ako využiť IKT vo vyučovacom procese (NAGY – BRESTENSKÁ 2001).

Zavedenie IKT do vzdelávacieho procesu predstavuje určitú revolúciu v spôsoboch učenia sa a takisto aj v spôsobe vyučovania. Využívanie nových technológií si vyžaduje aj nový spôsob hodnotenia. Väčšina učiteľov však nemá dostatok skúseností v tom, ako hodnotiť používanie IKT vo vyučovacom procese (GADUŠOVÁ – MALÁ 2003).

Pomocou IKT si žiaci vedia predstaviť aj predmety a javy, ľudským okom nezachytiteľné. Z toho vyplýva, že s IKT úzko súvisí aj vizualizácia, ktorá umožňuje symbolicky vyjadriť abstraktné pojmy a tým ich lepšie a ľahšie pochopiť, uľahčuje a urýchľuje proces poznávania, aktivizuje celú osobnosť žiaka tým, že uľahčuje jeho samostatné myslenie, púta jeho pozornosť a záujem, pôsobí na city a zároveň oživuje vyučovanie. Prispieva tiež k rozvoju zmyslových orgánov, zdokonaleniu poznávacích procesov a rozvoju pozorovacích schopností žiakov (HALÁKOVÁ – PROKŠA – ŽOTANIOVÁ 2004).

Metodika

Výskum sme uskutočnili v stredných školách, väčšinou v gymnáziách. Ako merný nástroj sme použili dotazník vlastnej konštrukcie, v ktorom boli otázky škálované, otvorené, uzavreté a polouzavreté. Predtým, ako sme dotazník rozoslali respondentom, bol skontrolovaný kompetentými zaoberajúcimi sa konštrukciou a používaním dotazníkov, ktorí ho

označili za vhodný. Vyplnený dotazník nám vrátilo 72 respondentov zo šiestich krajov Slovenska, okrem Prešovského a Košického. Vek učiteľov sa pohyboval od 24 do 61 rokov.

Výsledky

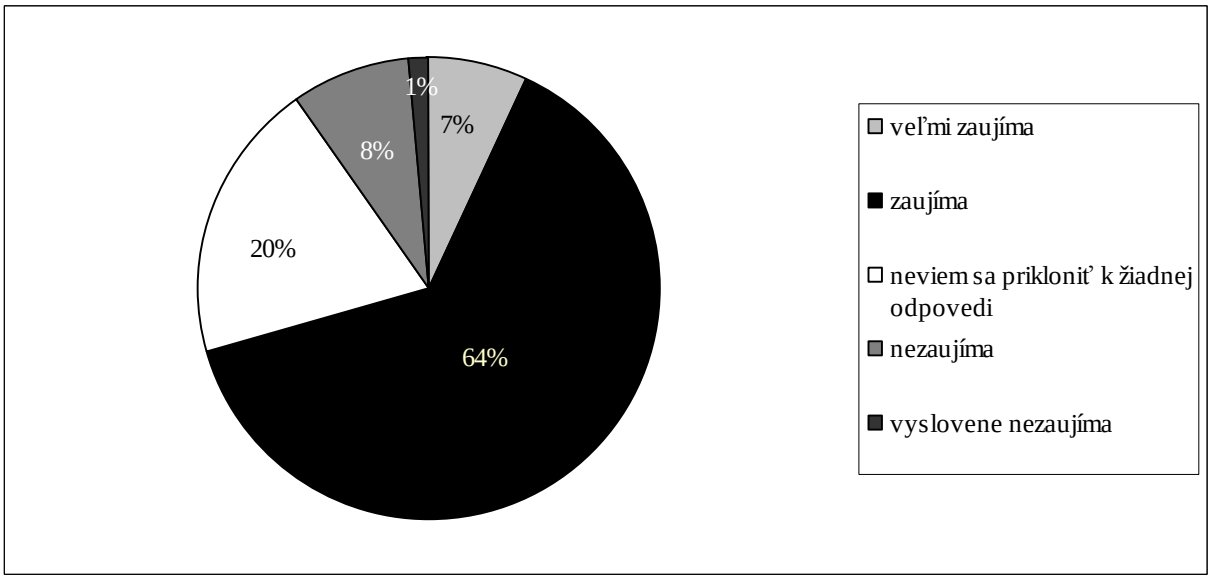
Ak sme chceli zistiť, či podľa učiteľov vplývajú IKT na záujem žiakov biológie, museli sme zistiť, čo si myslia učitelia o záujme žiakov na biológiu. Použili sme škálovanú otázku v dotazníku. Otázka znela:

Ø Učebný predmet biológia mojich žiakov:

a) veľmi zaujíma
b) zaujíma
c) neviem sa prikloniť k žiadnej z odpovedí a, b, d, e
d) nezaujíma
e) vyslovene nezaujíma

Najviac respondentov, ako je uvedené v grafe 1 označilo možnosť „zaujíma“ – 64 %, najmenej, len 1 % označilo možnosť „vyslovene nezaujíma“. A na základe vyhodnotenia Likertových škál bola odpoveď respondentov v priemere „zaujíma“ (koeficient 3,61).

Graf 1 Záujem žiakov o učebný predmet biológia z pohľadu učiteľov



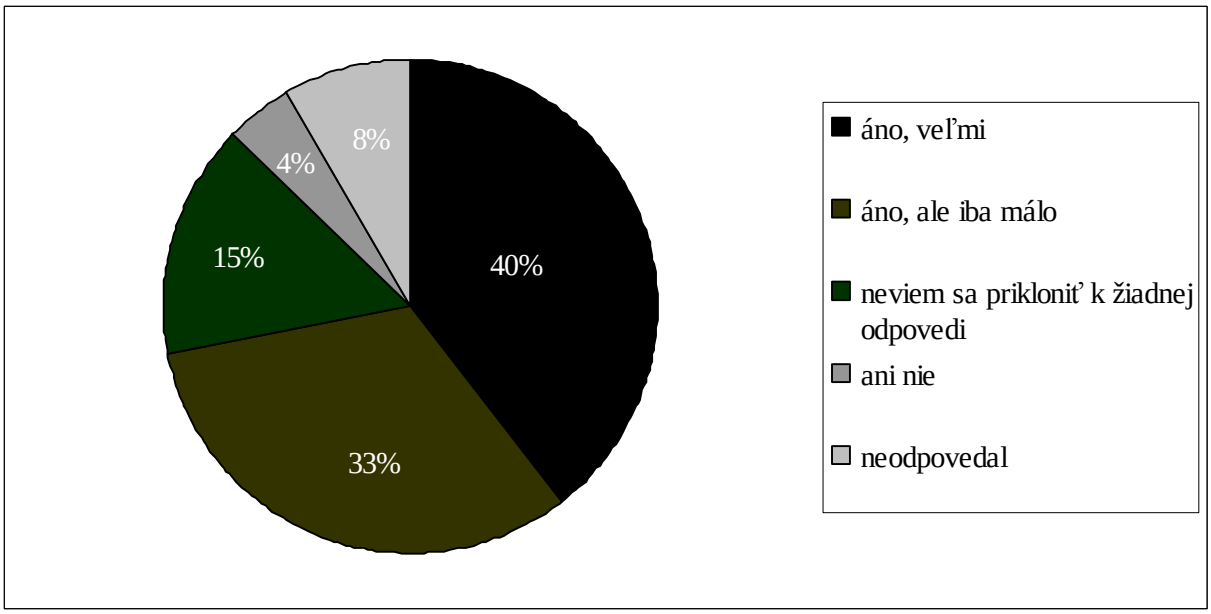
Na túto otázku sa dalo nadviazať otázkou týkajúcou sa IKT v biológii. Očakávali sme, že záujem žiakov o biológie prostredníctvom IKT v porovnaní s predchádzajúcou otázkou bude väčší. Tento predpoklad sa aj naplnil. Táto položka dotazníka znela:

Ø Podľa Vášho názoru zvyšuje používanie IKT na vyučovaní záujem žiakov o biológiu?

a) áno, veľmi
b) áno, ale iba málo
c) neviem sa prikloniť k žiadnej z odpovedí a, b, d, e
d) ani nie
e) vôbec nie

Najviac respondentov odpovedalo „áno, veľmi“ – 40 %. Takisto výraznú hodnotu nadobudla možnosť „áno, ale iba málo“ – 33 %. Možnosť „vôbec nie“ neoznačil žiaden respondent. Neodpovedalo 8 % respondentov a 15 % respondentov označilo možnosť „neviem, sa prikloniť k žiadnej z odpovedí a, b, d, e“ (graf 2). Pri tomto usudzujeme, že pre túto možnosť sa rozhodli učitelia, ktorí IKT na vyučovaní biológie nepoužívajú. Podľa vyhodnotenia Likertových škál vyšiel koeficient 4,01, ktorý zodpovedá odpovedi „áno, ale iba málo“.

Graf 2 Zvyšovanie záujmu o biológiu prostredníctvom IKT



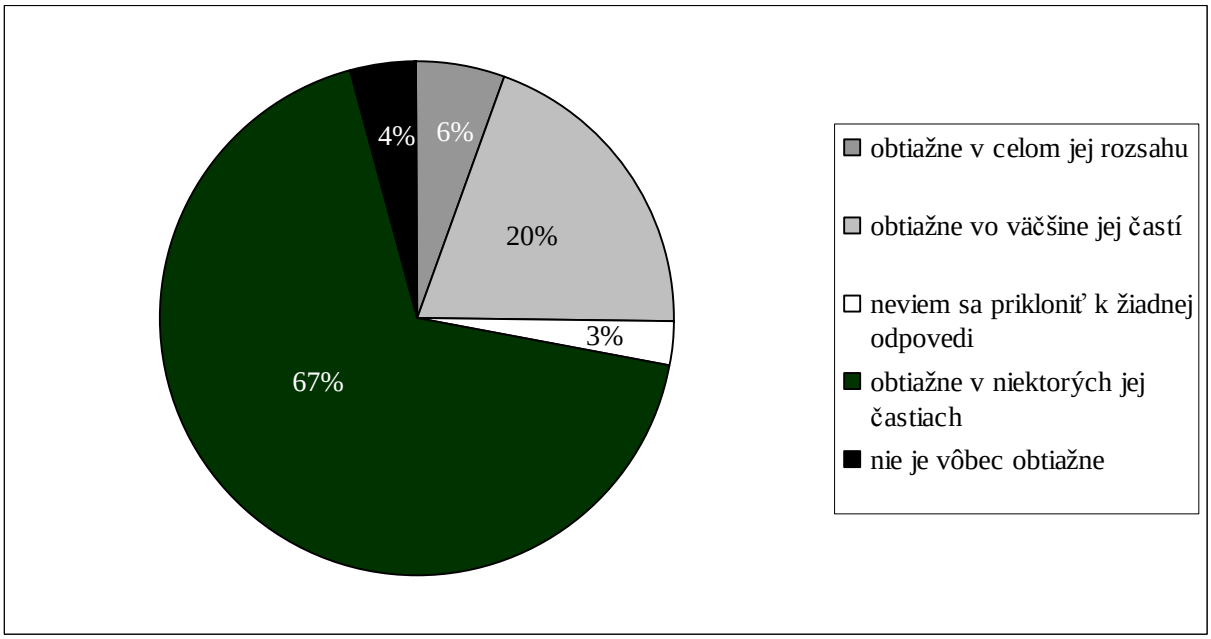
Ďalšími dvoma položkami sme sa snažili zistiť, že ako to je s porozumením učivu biológie bez používania IKT a s používaním IKT. Prvá položka znela:

Ø Porozumieť učivu biológie je pre mojich žiakov:

a) obtiažne v celom jej rozsahu
b) obtiažne vo väčšine jej častí
c) neviem sa prikloniť k žiadnej z odpovedí a, b, d, e
d) obtiažne v niektorých jej častiach
e) nie je vôbec obtiažne

Ako vidíme na grafe 3 najviac respondentov – 67 % označilo možnosť „obtiažne vo väčšine jej častí“. Najmenej, len 3 % opýtaných, „neviem sa prikloniť k žiadnej z odpovedí a, b, d, e“. Na základe vyhodnotenia Likertových škál bola odpoveď respondentov v priemere „neviem sa prikloniť k žiadnej z odpovedí a, b, d, e“ (koeficient 3,4)

Graf 3 Porozumenie učivu biológie žiakmi



Na túto otázku nadväzovala ďalšia, tak ako sme spomínali predtým. Učiteľov sme sa pýtali:

Ø Porozumieť učivu biológie pri vyučovaní pomocou IKT je pre mojich žiakov:

a) obtiažne v celom jej rozsahu

b) obtiažne vo väčšine jej častí

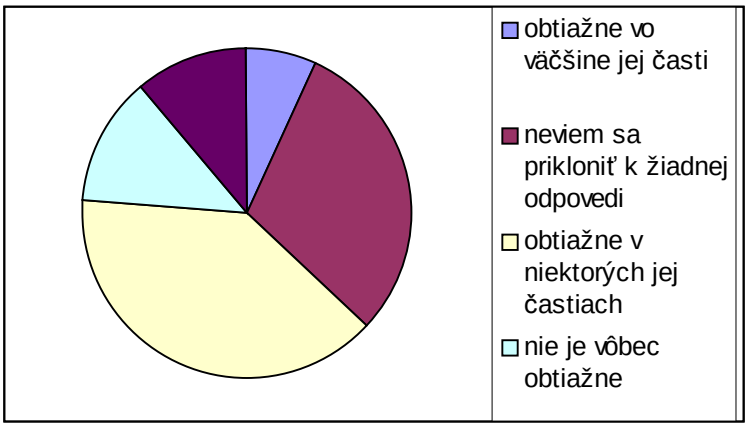
c) neviem sa prikloniť k žiadnej z odpovedí a, b, d, e

d) obtiažne v niektorých jej častiach

e) nie je vôbec obtiažne

Očakávali sme, že možnosti „d“ a „e“ dosiahnú vyšší percentuálny podiel v porovnaní s predchádzajúcou otázkou. Tu sa ale náš predpoklad nepotvrdil (graf 4). Možnosť „nie je vôbec obtiažne“ označilo len 13 % respondentov a možnosť „obtiažne v niektorých jej častiach“ 39 % respondentov. Potešujúce zistenie bolo, že žiaden respondent neoznačil možnosť „obtiažne v celom jej rozsahu“. Neodpovedalo až 11 % respondentov. Tu usudzujeme, že túto možnosť označili tí učitelia, ktorí IKT nepoužívajú.

Graf 4 Porozumenie učivu biológie žiakmi s pomocou IKT



Záver

Na základe zisteného môžeme povedať, že podľa názoru učiteľov vyučovací predmet biológia žiakov zaujíma. A používanie IKT na tomto vyučovacom predmete ich záujem ešte viac rastie. Predpokladáme, že to je práve interaktívnosť a väčšia motivácia, ktorá spôsobuje zvyšujúci sa záujem o tento predmet v porovnaní s klasickým spôsobom vyučovania. Motivácia a interaktívnosť takisto umožňuje študentom sa aktívne podieľať na vlastnom procese vzdelávania.

Tento príspevok vznikol za podpory Grantu UK 115/2005

Literatúra:

GADUŠOVÁ, Z.; MALÁ, E.: K otázkam hodnotenia využitia informačno-komunikačných technológií vo vzdelávaní. Technológia vzdelávania (príloha Slovenský učiteľ), roč. XI, 2003, č. 7, s. 2-4.

HALÁKOVÁ, Z.; PROKŠA, M.; ŽOTANIOVÁ, K.: Efektívnosť použitia prvkov vizualizácie v učebných úlohách z chémie. Chemické rozhľady, roč.V, 2004, č. 4, s. 246-252.

HORVÁTHOVÁ, D.; VÍTKO, P.: Rozvoj schopností študentov v ovládaní programového vybavenia prostredníctvom multimédií. Informatika v škole, 2003, č. 25, s.13-19.

NAGY, T.; BRESTENSKÁ, B.: Nové smerovanie prípravy učiteľov prírodovedných predmetov na prácu s IKT. Informatika v škole, 2001, č. 22, s. 24-30.

NAGYOVÁ, S.; UŠÁKOVÁ, K.: Niekoľko námetov na praktické cvičenia z biológie. Téma: „Nervová sústava človeka“(I.). Biológia – Ekológia – Chémia, roč. VIII, 2003, č. 2, s. 29-30.

TURČÁNI, M.; BAUEROVÁ, M.: Výučba prírodovedných predmetov s multimediálnou PC podporou. Technológia vzdelávania, roč. IX, 2001, č. 7, s. 14-16.

Mgr. Milan KUBIATKO

doc. RNDr. Katarína UŠÁKOVÁ, PhD.

Univerzita Komenského, Prírodovedecká fakulta
Katedra didaktiky prírodných vied, psychológie a pedagogiky
Mlynská dolina
842 15 Bratislava

EDUKAČNÝ DISK „RADY VTÁKOV EURÓPY“

Úvod

Didaktické prostriedky trvalo a tradične sprevádzajú vyučovanie a vzdelávanie. Časom sa však menia a vyvíjajú spolu s rozvojom kultúry a techniky. V súčasnej dobe sa pozornosť sústreďuje na tzv. moderné didaktické prostriedky, ako sú počítače, internet, CD, ... (MAŇÁK 1995).

Otázka využívania počítačov sa od svojich začiatkov po súčasnosť výrazne rozvinula. V súčasnej dobe sa objavuje tendencia, ktorá presadzuje využívanie hypermediálnych prostriedkov v oblasti vzdelávania.

Veľký prínos moderných didaktických prostriedkov spočíva v tom, že podnecujú k zmenám tradičných postupov z hľadiska obsahu, metód a organizačných foriem vyučovania. Hlavným činiteľom zmien nie sú počítače, ale učitelia (SKALKOVÁ 2004).

Počítače umožňujú prezentovať rôzny vizualizačný softvér. Výhodou softvéru je, že môže obsahovať veľké množstvo údajov na znázornenie príslušných javov, procesov a fyzikálnych objektov. Vizualizačný softvér môže byť uložený na kompaktnom disku (CD) alebo DVD *(Digital Versatile Disc).* Keďže ide o digitalizované formy záznamu, príslušné elementy problémových situácií možno meniť a kombinovať (VESELSKÝ 2005).

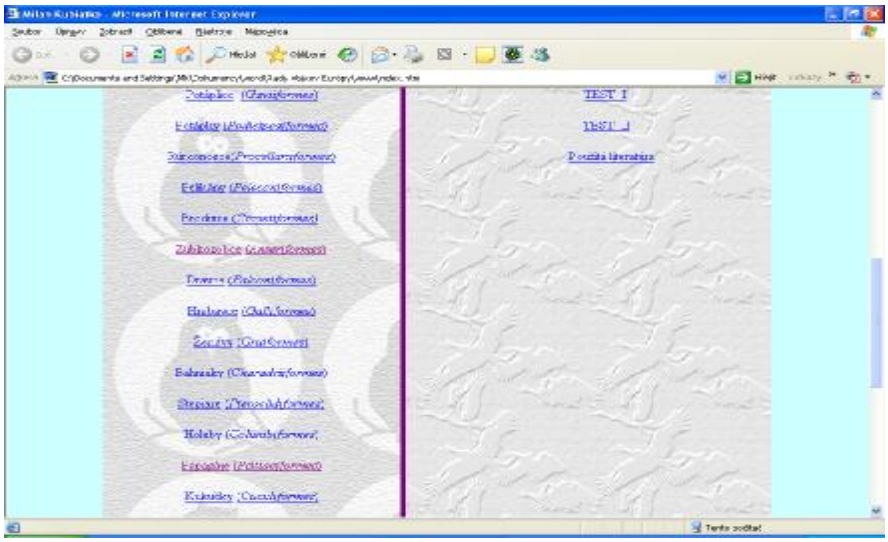
Pomocou edukačných diskov je možné zobraziť aj veci ľudským okom nezachytiteľné a tiež urýchliť jednotlivé deje. Z toho vyplýva, že s edukačným softvérom má veľmi úzky súvis vizualizácia, ktorá umožňuje symbolicky vyjadriť abstraktné pojmy a tým ich lepšie a ľahšie pochopiť, uľahčuje a zefektívňuje proces poznávania (HALÁKOVÁ – PROKŠA – ŽOTANIOVÁ 2004).

Edukačný disk

Edukačný kompaktný disk je digitalizovaná forma záznamu, ktorá dokáže žiakov vtiahnuť do mikrosveta, ktorý určitým spôsobom modeluje reálny svet. Edukačný disk s názvom „Rady vtákov Európy“ vznikol ako príloha diplomovej práce. Je sústredený na žiakov základných a stredných škôl, má slúžiť ako doplnkový materiál k vyučovaniu zoológie. Disk bol vytvorený v programe Microsoft FrontPage.

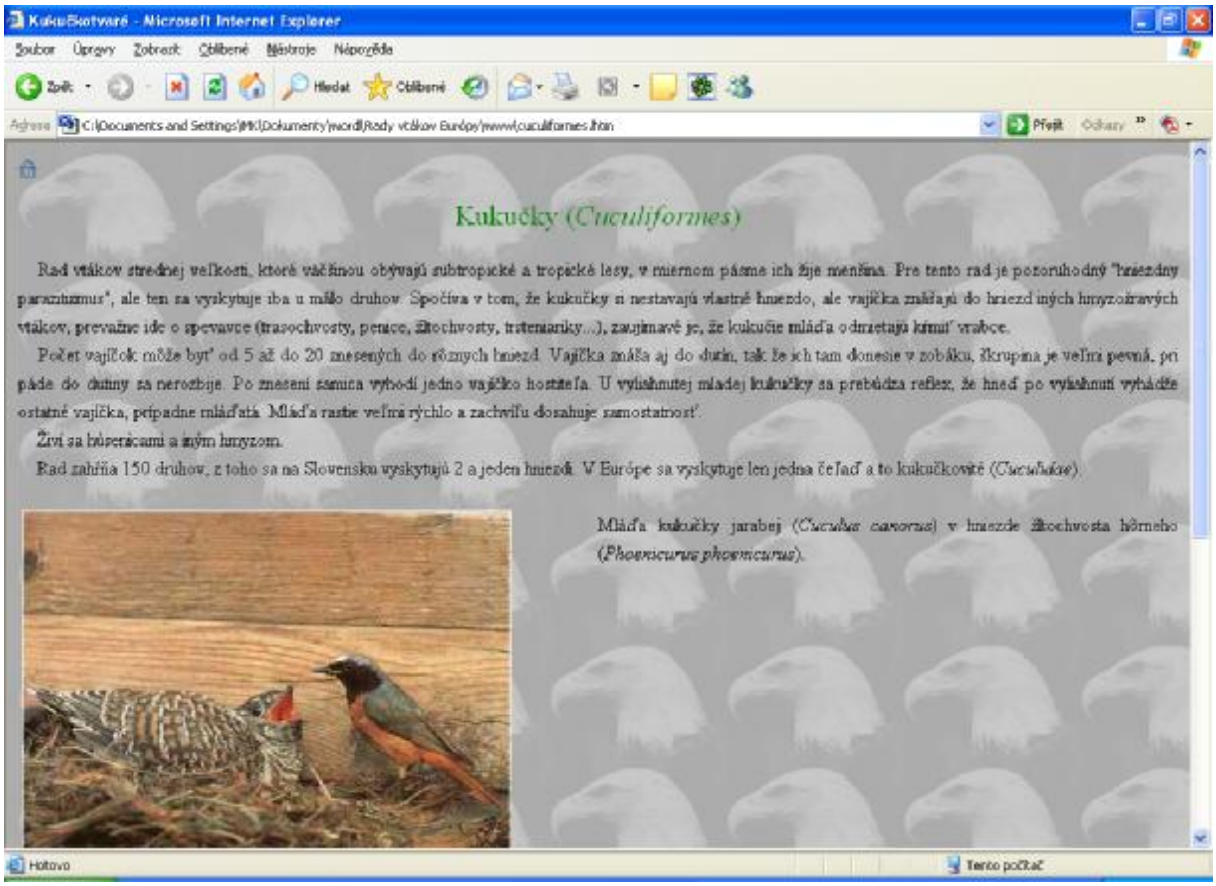
Úvodná stránka obsahuje odkazy na jednotlivé rady vtákov, spolu s ich latinskými názvami (obr. 1); odkaz, pomocou ktorého sa dostanú študenti ku všeobecnej charakteristike vtákov, ďalej tam bol uvedený odkaz na všeobecné informácie o autorovi. Tiež sa tam nachádzajú odkazy na testy súvisiace s informáciami, ktoré sme poskytli a použitá literatúra, pomocou ktorej sme vytvorili daný edukačný disk.

Obr. 1.: Ukážka úvodnej stránky edukačného disku



Pri prezentovaní jednotlivých radov sme ako prvé uviedli názov radu (slovenský a vedecký). Ďalej nasledovala charakteristika radu, ktorá zahrňovala informácie o geografickom rozšírení daného radu. Zväčša sme uvádzali informácie o hniezdení, t.j začiatku hniezdenia, o počte vajíčok, dĺžke sedenia na vajíčkach a dĺžke kŕmenia (obr. 2). Nakoniec nasledovali informácie o potravných nárokoch. Ak sa k nejakému radu žiadali uviesť špeciálne poznámky, tak bolo možné nájsť ich práve na danom mieste.. Potom nasledovala obrazová časť, kde sme uviedli obrázky typických zástupcov daného radu spolu s ich systematickým zaradením. Na každej strane bola znázornená ikona, po ktorej kliknutí bolo možné vrátiť sa späť na úvodnú stránku. Na každej stránke radu boli na konci vypísané odkazy na ďalšie rady.

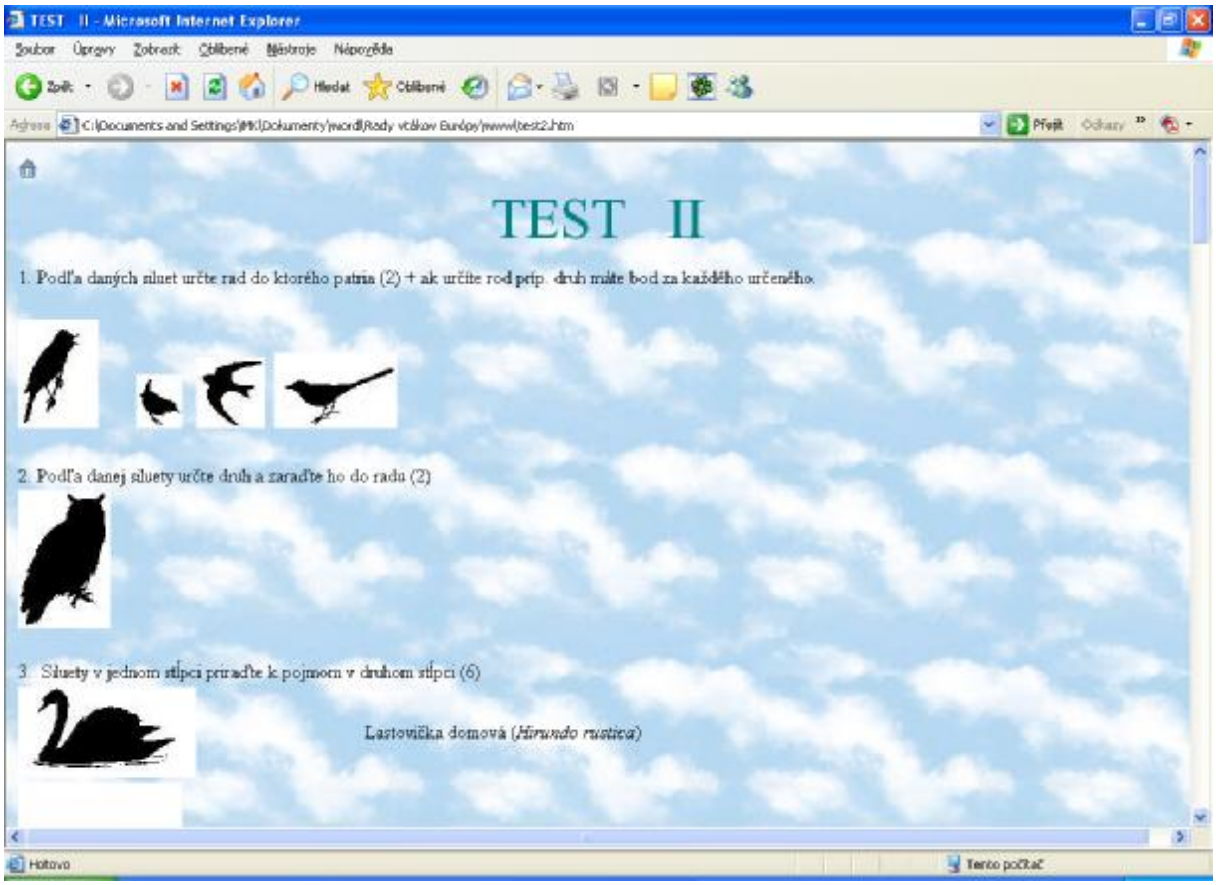
Obr. 2.: Ukážka zo stránky jedného radu



Po prezretí edukačného disku sa mohli žiaci, resp. študenti podrobiť testovaniu. Vytvorili sme dva typy testov. Prvý bol len písaný, kde boli rôzne typy otázok: priraďovacie, opisovacie, bolo potrebné opraviť chybnú vetu, ... K tomuto testu sme uviedli aj správne odpovede.

Druhý typ testu bol obrázkový, kde boli znázornené siluety vtákov, prípadne len ich nohy. Úlohou žiakov bolo určiť názov vtáka, ktorého reprezentuje daná silueta (obr. 3). K tomuto testu boli vypracované správne odpovede, ktoré sú uvádzané na disku.

Obr. 3.: Ukážka zo stránky testu II



Záver

Popísaný edukačný disk by mal slúžiť žiakom, študentom a učiteľom na doplnenie informácií týkajúcich sa zoológie. Prostredníctvom neho sme im ponúkli množstvo obrazového materiálu, ktorý by mal uľahčiť predstavu daného druhu vtáka. Veríme, že tento disk pomôže tým, ktorí sa zaujímajú o ornitológiu a vzbudí záujem o zoológiu u väčšieho počtu študentov.

Tento príspevok vznikol za podpory Grantu KEGA 3/3184/05.

Literatúra

HALÁKOVÁ, Z.; PROKŠA, M.; ŽOTANIOVÁ, K.: Efektívnosť použitia prvkov vizualizácie v učebných úlohách z chémie. Chemické rozhľady, roč. V, 2004, č. 4, s. 246-252.

MAŇÁK, J.: Nárys didaktiky. Brno, Masarykova Univerzita, 1995, 104 s. ISBN 80-210-1124-6.

SKALKOVÁ, J.: Pedagogika a výzvy nové doby. Brno, PAIDO, 2004, 160 s. ISBN 80-7315-060-3.

VESELSKÝ, M.: Pedagogická psychológia II: Teória a prax. Bratislava, Univerzita Komenského Bratislava, 2005, 168 s. ISBN 80-223-1911-2.

Mgr. Milan KUBIATKO

RNDr. Soňa NAGYOVÁ

Univerzita Komenského, Prírodovedecká fakulta
Katedra didaktiky prírodných vied, psychológie a pedagogiky
Mlynská dolina CH 2
842 15 Bratislava