Programování 6.L, 2.A

Úkol od 14. 5. do 20. 5.

Zhodnocení minulého úkolu (průběh quicksortu na papír)

Většina si s hodnotami proměnných pěkně pohrála, a tím doufám přišla na podstatu této metody: podle nějaké výchozí hodnoty se rozdělí všechna čísla na dvě hromádky – menší než výchozí hodnota a větší než výchozí hodnota. Ty hromádky nemusejí být stejně početné a dál se řadí stejným způsobem – rozdělením na ještě menší hromádky podle nové výchozí hodnoty. Pro rychlost této metody je klíčová volba výchozí hodnoty. V našem programu to byla hodnota prostředního prvku hromádky čísel `x:=s[(d+h) div 2]`, tzn. počítá se *průměr indexů*. Někteří ale `x` počítali jinak – jako *průměr prvního a poslední čísla* hromádky, v programu by to vypadalo `x:=(s[d]+s[h]) div 2`, což je rozdíl, ale pro fungování metody kupodivu nepodstatný. Ani jedna z obou možností není ideální (pro našich 10 čísel je to jedno, ale třeba v databázi 500.000 kusů zboží už by to jedno nebylo, zákazník v e-shopu nebude čekat 10 minut, než se mu to seřadí), proto bývá quicksort doplňován pomocnou metodou pro zjištění výhodné výchozí hodnoty (u čísel je nejlepší *medián*, pokud znáte z matematiky).

Práce se souborem

Čtení a zapisování dat do souboru jsme už dělali, ale připomeneme si a shrneme základní poznatky, protože to budeme v následujících úkolech potřebovat.

Textový a binární soubor

Rozlišujeme 2 typy souborů: *textový* soubor a *binární* soubor. Textový soubor obsahuje textové znaky (písmena, číslice, interpunkční znaménka a několik znaků, které nejsou přímo viditelné, ale formátují text, např. znak konce odstavce). Čistě textový soubor si můžeme vytvořit a číst třeba v Poznámkovém bloku a je dobré mu dávat příponu .txt (pozor – soubor z Wordu není textový, obsahuje kromě textu obrovské množství formátovacích kódů).

Binární soubor obsahuje počítačové kódy, které pro člověka nemusí být na první pohled dešifrovatelné a jejich struktura souvisí s uspořádáním dat v paměti počítače. Slouží k uložení jakýchkoli dat. K jejich zápisu nebo čtení potřebujeme zvláštní program (např. Word pro čtení souboru .docx, nebo grafický editor pro práci se soubory .jpg). Binární soubory, které budeme vytvářet pomocí Pascalu, mohou mít různou příponu, záleží na nás, jakou si zvolíme, ale pro soubor s daty je dobré používat příponu .dat.

Typy proměnných pro práci se souborem

Budeme-li pracovat s *textovým* souborem, bude proměnná zastupující tento soubor typu `Text`. Proměnná zastupující *binární* soubor bude typu `file of …` , kde za tři tečky dáme *integer*, *real*, *string*, apod., podle toho, jaký typ dat chceme do souboru uložit. Jeden soubor tedy obsahuje data jednoho, přesně určeného, typu. Příště si ukážeme, že soubor může obsahovat i data našeho vlastního typu, který si sami vytvoříme.

Příklad deklarace proměnných:

`var f:file of integer`	proměnná `f` zastupuje soubor celých čísel
`var g:Text`	proměnná `g` zastupuje textový soubor
`var sx2:file of string[20]`	proměnná `sx2` zastupuje soubor 20znakových řetězců
`var ub:file of mujtyp`	proměnná `ub` zastupuje soubor vlastního typu `mujtyp`

Příkazy pro práci se souborem

K práci se souborem využijeme příkazy `assign`, `reset`, `rewrite`, `close`. Ukážeme si na příkladech.

`assign(f,'c:\pascal\moje\vysledky.dat')` spojení proměnné `f` s konkrétním souborem, jehož název (možno s celou cestou) napíšeme do apostrofů, dál už v programu nepracujeme s názvem souboru, ale jen s proměnnou

`reset(f)` otevření souboru pro čtení (soubor musí existovat)

`rewrite(f)` otevření souboru pro zápis (pozor – existující soubor se tím vymaže a bude se vytvářet nový)

`close(f)` uzavření souboru (Pascalu nevadí nechat otevřený soubor, ale jiným programům to vadit může, proto je dobré si zvyknout soubory hned po práci uzavírat)

`eof(f)` `eof` není příkaz, ale funkce typu Boolean, která zjistí, zda bylo dosaženo konce souboru (end of file), což je důležité při čtení souboru, kdy nevíme, kolik dat obsahuje; funkce vrací hodnotu *true* = když bylo dosaženo konce souboru, nebo *false* = když ještě není konec souboru

Čtení ze souboru a zápis do souboru

Máme-li soubor otevřený pro čtení, můžeme použít příkaz `read(f,x)` a ze souboru se z aktuální pozice (kterou přímo nevidíme, načítá se to postupně po jedné datové položce) přečte údaj, který se uloží do proměnné `x`. Příkaz `readln(f,x)` se používá jen u textových souborů a slouží k přečtení 1 řádku (line).

Máme-li soubor otevřený pro zápis, zapíšeme údaj příkazem `write(f,x)` – na konec souboru se vloží údaj z proměnné `x`. Příkaz `writeln(f,x)` vloží na konec textového souboru text z proměnné `x` a ukončí řádek.

Ukázkový program pro práci soubory

Program pracuje se dvěma soubory – z prvního, binárního souboru přečte všechna data (20znakové řetězce) a načte je do pole `x`, pak všechny řetězce z pole `x` uloží do textového souboru po řádcích. Jinak uživatel po spuštění programu vlastně neuvidí nic – není tam žádný výstup na obrazovku ani se po uživateli nechce žádný vstup z klávesnice. Pouze na disku by se měl v nějaké složce (není specifikováno v programu) objevit soubor radky.txt s řetězci z prvního souboru překopírovanými do řádků. Protože nevíme na začátku, kolik řetězců je v prvním souboru, použijeme cyklus `while`, který načítá řetězce tak dlouho, dokud není konec souboru. Proměnná `i` pak zjistí počet řetězců a uloží tento údaj do proměnné `n`.

Ukázkový program si projděte, zvlášť ti, kterým práce se souborem ještě nešla, řádek po řádku, a snažte se všechny kroky pochopit.

```
program Ukazkasoubor;
var   f:file of string[20];
      g:text;
      x:array of string[20];
      i,n: integer;
begin
   assign(f,`retezce.dat`);
   reset(f);
   i:=0;
   while not eof(f) do begin
      i:=i+1;
      read(f,x[i]);
   end;
   n:=i;
   close(f);
   assign(g,`radky.txt`);
   rewrite(g);
   for i:=1 to n do writeln(g,x[i]);
   close(g);
end.
```

Úkoly

1) Vytvořte program, který vygeneruje 100 náhodných celých čísel a uloží je do souboru – binární soubor typu `file of integer`.

2) Vytvořte program, který načte celá čísla ze souboru (z úkolu č. 1), seřadí je pomocí metody quicksort a seřazená je vypíše na obrazovku. Program by měl umět pracovat s libovolným počtem načtených čísel, nejen se 100.

3) Vytvořte program, ve kterém bude uživatel zadávat jména a příjmení (každé zvlášť) několika lidí (různý počet) a ty se budou ihned ukládat do textového souboru. Textový soubor si pak můžete otevřít v Poznámkovém bloku a zkontrolovat, jestli obsahuje všechna jména a příjmení na střídačku v řádcích.

4) Vytvořte program, který přečte z řádků textového souboru (z úkolu č. 3) jména a příjmení a vypíše na obrazovku vedle sebe vždy jméno a příjmení jednoho člověka.

Zdrojové kódy 4 programů **pošlete** na andziol@glp-plzen.cz nejpozději **ve středu 20. 5.**

Další úkol zveřejním v naší nové školní Google Učebně ve čtvrtek 21. 5.

Filip Andziol