Střední průmyslová škola a Vyšší odborná škola technická Brno, Sokolská 1

Šablona: Inovace a zkvalitnění výuky prostřednictvím ICT

Název: Základy programování a algoritmizace úloh

Téma: Třídění dat

Autor: Ing. Hodál Jaroslav, Ph.D.

Číslo: VY_32_INOVACE_26–04

Anotace: *Materiál popisuje principy třídění dat a související algoritmus hledání maxima.*
Materiál je určen pro 3. a 4. ročník oboru strojírenství a technické lyceum.
Vytvořeno: prosinec 2012.

Podpora digitalizace a využití ICT na SPŠ CZ.1.07/1.5.00/34.0632

4. Třídění dat

- třídění dat je jednou z nejčastějších úloh zadávanou počítačům
- existuje velké množství třídících algoritmů
- liší se od sebe **rychlostí** a **počtem operací**, které provádějí, ale také **náročností naprogramování**
- nelze jednoznačně vybrat jeden nejlepší třídící algoritmus, protože pro různé situace se hodí různé třídící postupy
- i ten nejpomalejší algoritmus může být v určité situaci vhodný, protože například pro setřídění malého množství dat by námaha spojená s psaním složitého kódu byla v nepoměru k zanedbatelné časové úspoře při třídění

Třídící algoritmy

- mezi nejjednodušší algoritmy, které sice data setřídí, ale zabere jim to hodně času, patří např. **třídění výběrem** (select sort) nebo **bublinové třídění** (bubble sort)
- velmi rychlé, ale náročné na naprogramování je tzv. **rychlé třídění** (quick sort)
- pro zatřiďování nových dat do již setříděné posloupnosti je vhodné např. **třídění vkládáním** (insert sort)
- speciální datovou strukturu pro uchovávání a zatřiďování dat používá např. **třídění haldou** (heap sort)

- další důležitou úlohou, která s tříděním dat často souvisí je **hledání maxima** (minima)

Hledání maxima / minima (1)

- nejedná se sice o klasický třídící algoritmus, protože jeho výstupem nejsou setříděná vstupní data, často je ale nejlepší hodnota ta jediná, která nás zajímá
- pak není potřeba zbytečně třídit všechna data
- místo složitého zpracování všech dat naráz, používá tento algoritmus (podobně jako člověk) **postupné prohlížení dat** s tím, že se snaží zapamatovat si **dosud nejlepší** nalezenou hodnotu
- k tomu používá pomocnou proměnnou (paměť), v níž si pamatuje jen tuto jednu hodnotu
- pokud při procházení hodnot nalezne lepší hodnotu než je zapamatovaná, zapamatuje si ji místo ní

Hledání maxima / minima (2)

- jakmile algoritmus projde všechny hodnoty, je dosud nejlepší nalezená hodnota zároveň **nejlepší hodnota**
- musíme ale také řešit problém vhodné inicializace pomocné proměnné (paměti) tak, aby její **počáteční hodnota** neovlivnila výsledek hledání
- uvažte například situaci, že nastavíme počáteční hodnotu 0 a uživatel nám zadá samá záporná čísla
- náš program bude tvrdit, že největší číslo je 0, i když mezi zadanými čísly vůbec nebylo
- proto se jako výchozí hodnota paměti běžně používá **první** (obecně lze kterákoliv) **ze zadaných** hodnot

Hledání maxima / minima (3)

```
var a:array[1..10] of byte;

...

for i := 1 to 10 do readln(a[i]);

max := a[1];
for i := 2 to 10 do
  if a[i] > max then
    max := a[i];

writeln('Největší hodnota je ', max);
```

Třídění výběrem

- využívá hledání maxima k nalezení největšího prvku, který následně vymění s prvkem, který se dosud nacházel na místě, kam potřebujeme umístit nalezenou hodnotu
- postupně také zmenšuje prohledávanou oblast

```
for i := 1 to 9 do begin
  max := a[i]; pozice := i;
  for j := i+1 to 10 do
    if a[j] > max then begin
      max := a[j]; pozice := j;
    end;

  a[pozice] := a[i];
  a[i] := max;
end;
```

Bublinové třídění

- využívá o něco složitější princip než třídění výběrem
- postupně porovnává jednotlivé dvojice sousedících hodnot a případně je zaměňuje, aby došlo k posunu lepší hodnoty dopředu
- postupně také zmenšuje prohledávanou oblast

```
for i := 1 to 9 do begin
  for j := 9 downto i do
    if a[j+1] > a[j] then begin
      x := a[j+1];
      a[j+1] := a[j];
      a[j] := x;
    end;
```