Programování

Pokročilá algoritmizace

Opakování základů algoritmizace

- **Algoritmus nám poskytuje řešení konkrétního problému**
- Daný problém musí být splnitelný, aby se dalo uvažovat o sestrojení algoritmu
- Vytvářený algoritmus musí splňovat požadavky na **jednoznačnost, konečnost** a **rezultativnost** -> „algoritmus dělá co má a chová se dle očekávání“
- Samotný algoritmus se skládá ze sledu jednoduchých instrukcí, které mají své pevné pořadí – říkáme jim kroky
- Cílem je vytvořit co nejefektivnější algoritmus
- U tvorby algoritmu máme na paměti otázku složitosti a náročnosti:
  - Nároky na paměťový prostor
  - Nároky na čas
  - Celková přehlednost a srozumitelnost

Zápis algoritmu

- Pro znázornění algoritmu můžeme použít několik přístupů
- **Matematické vyjádření** – algoritmus je popsán jako rovnice
- **Textový zápis** – obdoba pracovního postupu nebo receptu
- **Diagramy a schémata** – lze použít i pro popis modelu jednotlivých částí systému
- **Pseudokód** – kombinace textového zápisu s užitím programových primitiv

```
if číslo je větší než nula
    vypiš: "Číslo je kladné"
else
    vypiš: "Číslo je záporné"
end if
```

- **Programovacím jazykem** – nejpřesnější zápis (může být hůře pochopitelný)

Metody řešení návrhu algoritmu

- Velká část algoritmů pracuje se vstupem o určitém počtu hodnot – pole, seznam, …
- V našich algoritmech se s takovými typy často nesetkáváme, ale můžeme těchto přístupů využít i pro jednodušší algoritmy
- **Rozděl a panuj**
  - Způsob řešení takových algoritmů spočívá v dělení vstupní množiny na menší částí
  - Tyto menší části jsou pak řešeny samostatně
- **Hladový algoritmus**
  - Vstupní množina je zpracovávána v pořadí v jakém přišla na vstup
  - U každé z procházených podmnožin se sleduje, zda vyhovuje řešení nebo ne
- **Hledání s návratem**
  - Metoda používána pro problémy, které jsou řešené nad datovou strukturou stromu

Zjednodušení obecného postupu řešení algoritmu

- Definování očekávaného výsledku – čeho chceme dosáhnout
- Zjištění vstupních hodnot – co máme k dispozici pro výsledek
- Pokud předpokládáme vyšší složitost/komplexnost řešení rozdělíme jej na menší části
  - Výsledek menšího algoritmu je následně vstupme pro další menší algoritmus
- Sestavení sekvence kroků vedoucí k úplnému řešení nebo mezivýsledku
- Zpětná revize algoritmu – zvážení okrajových případů řešení (co by kdyby)
- Testovací průchod pro dostupné možnosti vstupů
- TIP: před samotným vytvořením diagramu nebo zdrojového kódu si algoritmu kreslíme jako tzv. stavový – zjišťujeme stav a hodnoty částí před a po provedení některého z kroků

Algoritmy k procvičení

- Algoritmus, který spočítá počet slov ve větě – věta je ukončena tečkou.
- Rozdělení náhodné množiny na dvě podmnožiny podle znaménka
- Zjištění vnějších úhlů pravidelného n-uhelníku
- Hledání maxima v množině (poli) čísel
- Rozklad celého čísla na součin prvočísel
- Nalezení vadných pixelů na monitoru
- Přejití přes přechod se semaforem
- Založené nové karty pacienta u doktora
- Přihlášení uživatele do systému