Algoritmizace

Úvodní sekce

V této kapitole se budeme věnovat principům logického převodu problému na kód, jeho zápisem a následným vyřešením. Tím se zabývá algoritmizace.

Co je to algoritmus?

Algoritmem rozumíme posloupnost konečného počtu přesně definovaných kroků potřebných k dosažení určitého výsledku či vyřešení určité úlohy. Příklad algoritmu pro opravu žárovky:

```
OpravaZarovky :
{
    Pokud: Svítí žárovka
    {
        Skoč na: Konec
    }

    Pokud: Není žárovka připojena na zdroj energie
    {
        Připojit žárovku na zdroj energie

        Pokud: Svítí žárovka
        {
            Skoč na: Konec
        }
    }

    Vyměnit žárovku

    Konec
}
```

Důležité vlastnosti algoritmu

Konečnost

Každý algoritmus musí vrátit výsledek v konečném počtu kroků.

Obecnost

Každý algoritmus by měl být univerzální. Například by cílem navržení algoritmu nemělo být vyřešit úlohu kolik je 3x4, ale implementace operace násobení pro jakýkoli vstup. Tedy pokud správně navrhneme algoritmus pro operaci násobení, pak by měl fungovat jak v případě, kdy jako vstupní data zadáme 3 a 4, ale také v případě, kdy zadáme čísla 23567 a 856.

Determinovanost (jednoznačnost výsledku)

Každý algoritmus musí vždy pro stejná vstupní data, vrátit stejná data výstupní.

Jednoznačnost (determinismus)

Každý algoritmus by měl být jednoznačný. Tedy každý jeho krok by měl být přesně a jednoznačně určen a mělo by být zcela zřejmé, co a jak který krok algoritmu provádí.

Resultativnost

Každý algoritmus by měl mít nějaký výstup, který je řešením zadaného problému pro vstupní data.

Elementárnost

Každý algoritmus by se měl skládat z konečného počtu jednoduchých kroků. Například by jeden krok neměl být “uvař vodu na kávu”, ale měl by obsahovat kroky jako vezmi konvici do ruky, otevři víko konvice atd.

> Ačkoli se všechny tyto vlastnosti mohou zdát jednoznačné a přirozeně jasné, je opravdu důležité na všechny tyto body při tvorbě programu myslet. Zejména elementárnost je velice důležitá pro další údržbu a změny v kódu. Na to však náš čtenář přijde postupem času sám.

Zápis algoritmu

V předchozí sekci jsme si představili dvě možnosti zápisu algoritmu. Prvnímu, spíše textovému způsobu se říká pseudokód. Druhý grafický způsob zápisu nazýváme vývojový diagram. Pseudokód je čistě slovní popis kroků, které je potřeba udělat k dosažení výsledku. Bývá členěn podobným způsobem, jako reálný kód zapsaný v programovacím jazyce. Jeho výhodou je, že po jeho napsání

je poměrně snadné přejít k reálnému kódu. Z popsaných kroků nám vzniknou názvy funkcí a z dalších postupů nám vzniknou komentáře. Je tedy také vhodný pro další převod do různých jazyků. Vývojový diagram se zase hodí k vizualizaci problému a kroků algoritmu a dosažení lepšího přehledu o algoritmu. Základními a nejčastěji používanými prvky vývojového diagramu jsou:

Obdélník se zakulacenými rohy

Používá se pro znázornění začátku a konce programu.

Obdélník s ostrými rohy

Používá se pro znázornění nějaké akce, funkce programu, či běhu nějakého podprogramu.

Kosočtverec posazený na hranu

Používá se pro znázornění nějaké podmínky nebo-li větvení běhu programu. Většinou z něj vedou dvě výstupní šipky. Jedna pro splněnou a jedna pro nesplněnou podmínku.

Obdélník se zkosenými stranami

Znázorňuje místo algoritmu, kde se načítá nějaký vstup, či vypisuje výstup.

> V praxi se pro menší programy používá buďto pouze jeden způsob zápisu algoritmu před převedením do reálného kódu nebo žádný. Pro větší aplikace se však většinou používají oba způsoby. Další možností zápisu algoritmu je samozřejmě zapsání ve námi vybraném programovacím jazyce. Této variantě se však budeme věnovat později.

https://skripta.ssps.cz/uploaded/IMG_5e396e467ede5.png

Shrnutí

Algoritmus je sada kroků užitých pro dosažení určitého výsledku. Základními vlastnostmi jsou konečnost, obecnost, determinovanost, jednoznačnost, resultativnost a elementárnost. K jeho zápisu se využívá pseudokód a vývojový diagram. Pseudokód v odborné programátorské literatuře naprosto převažuje a s vývojovým diagramem se setkáváme hlavně ve starší dokumentaci a v dokumentaci, která popisuje firemní procesy na obecnější úrovni. Nicméně je důležité umět číst obě notace - pseudokód i vývojové diagramy. Proto se je učíme. :)

Revision #6
Created 2025-05-02 12:31:10 UTC by Sklenář Maxim
Updated 2025-05-02 13:47:35 UTC by Sklenář Maxim