

16. Grafické značky používané při prezentaci algoritmů, třídící algoritmy

Algoritmus

- Přesný postup, kterým lze vyřešit daný typ úlohy
- Slovo pochází z příjmení perského matematika z 9. století
- Algoritmus může být např. recept na nějaké jídlo, nebo návod, jak postavit dům
- Video pro vysvětlení algoritmů: [zde](#)

Vlastnosti

- **Elementárnost** – Algoritmus se skládá z konečného počtu jednoduchých a snadno srozumitelných kroků, tedy příkazů
- **Konečnost** – Každý algoritmus musí skončit v konečném počtu kroků, tento počet kroků může být libovolně velký, pro každý jednotlivý vstup ale musí být konečný
- **Obecnost** – Algoritmus neřeší jeden konkrétní problém, ale obecnou „třídu“ problémů, má široké množství možných vstupů
- **Determinovanost** – Algoritmus je determinovaný, pokud za stejných podmínek, nabízí stejný výstup, využívá se dost často, ale někdy se také může využívat náhodnost
- **Výstup** – Algoritmus má alespoň jeden výstup, tvoří odpověď na problém, který algoritmus řeší (algoritmus vede od zpracování hodnot k výstupu)

Způsoby zápisu

Textové vyjádření algoritmu

- Dnes nejpoužívanější forma zápisu
- Výhody: snadný přepis do programovacího jazyku, přehlednost zápisu a jeho jednoznačnost
- Zápis je zapisován pomocí prostřednictvím formalizovaného jazyka, je využíván tzv. pseudokód

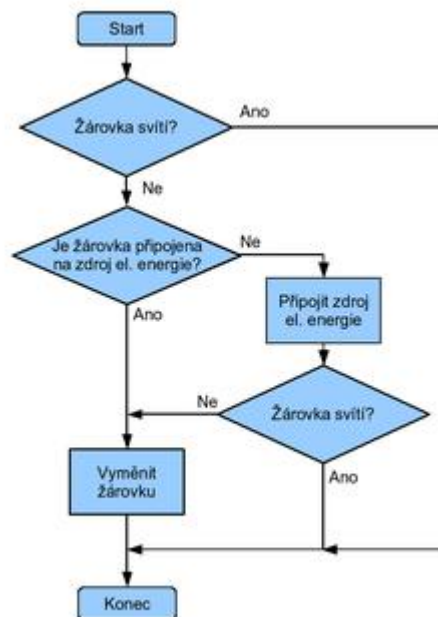
Pseudokód:

```
pokud je číslo kreditní karty platné
    proved' přenos, založený na číslech kreditní karty a objednávky
jinak
    zobraz chybové hlášení
konec podmínky
```

Grafické vyjádření algoritmu

- Algoritmus je popsán formalizovanou soustavou grafických symbolů
- Používány vývojové diagramy nebo strukturogramy
- Výhody: přehlednost, znázornění struktury problému, poskytuje informace o postupu jeho řešení

- Nevýhody: není vhodné pro rozsáhlé a složité problémy, náročnost konstrukce grafických symbolů, obtížná možnost dodatečných úprav












Vývojové diagramy

- Jeden z nejčastěji používaných prostředků pro znázorňování algoritmů
- Vývojové diagramy jsou grafické zobrazení algoritmu pomocí symbolů a propojení, které popisují sekvenci příkazů, větvení a cykly
- Tvořeny značkami ve formě uzavřených obrazců, do kterých jsou vepisovány slovní formou jednotlivé operace
- Tvary a velikosti značek jsou dány normami
- Značky jsou spojeny přímými nebo lomenými čarami a znázorňují tak posloupnost jednotlivých kroků
- Čáry mohou být orientované zavedením šipek, neměly by se křížit
- Pokud již ke křížení dojde, měly by být čáry zvýrazněny tak, aby bylo jednoznačně patrné, odkud a kam směřují
- Vývojový diagram čteme ve směru shora dolů
- **Výhody:** názornost, přehlednost
- **Nevýhody:** pracnost a složitost konstrukce

Sekvence příkazů, větvení, cyklus a další značky

- Sekvence příkazů se používá pro postupné vykonávání příkazů bez větvení nebo cyklů
- Větvení a cykly se používají k rozhodování, který kód se má vykonat na základě určitých podmínek nebo pro opakování určitého bloku kódu
- **Větvení:** slouží k větvení programu na základě podmínky, která je uvedena uvnitř, v případě splnění, pokračuje program větví označenou znaménkem + v opačném případě větví označenou znaménkem –
- **Cyklus:** označuje začátek cyklu o známém počtu opakování

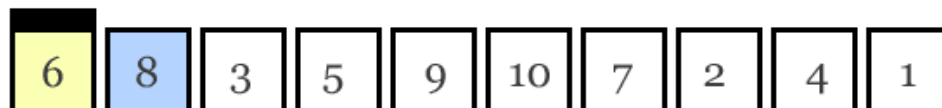
	Konec a začátek algoritmu
	Běžný příkaz
	Podmíněný výraz
	Cyklus s určeným počtem opakování
	Cyklus s podmínkou na konci
	Cyklus s podmínkou na začátku
	Ruční vstup
	Zobrazení výstupu
	Zpracování souboru

Třídící algoritmy, metody třídění, popis vybraného typu

- Třídící algoritmy jsou postupy pro třídění prvků v seznamu nebo poli podle určitých kritérií
- Existuje mnoho metod třídění, jako jsou Bubble sort, Insertion sort, Selection sort, Merge sort, Quick sort a další
- **Insertion sort** – Řazení vkládáním, je jednoduchý řadící algoritmus založený na porovnávání, algoritmus pracuje tak, že prochází prvky postupně a každý další neseřazený prvek zařadí na správné místo do již seřazené posloupnosti
- **Bubble sort** – Algoritmus opakovaně prochází seznam, přičemž porovnává každé dva sousedící prvky, a pokud nejsou ve správném pořadí, prohodí je, Pro praktické účely je neefektivní, využívá se hlavně pro výukové účely či v nenáročných aplikacích
- **Merge sort** – Merge sort je algoritmus, založený na tzv. principu rozděl a panuj, to znamená, že pokud nějaký problém neumíme vyřešit v celku, rozložíme si ho na více menších a jednodušších problémů, ten samý postup aplikujeme i na tyto problémy
- **Selection sort** – Myšlenka spočívá v nalezení minima, které se přesune na začátek pole (nebo můžeme hledat i maximum, a to dávat na konec), v prvním kroku tedy nalezneme nejmenší prvek v poli a ten poté přesuneme na začátek, v druhém kroku již nebudeme při hledání minima brát v potaz dříve nalezené minimum, po dostatečném počtu kroků dostaneme pole seřazené, algoritmus má nepříliš výhodnou časovou složitost a není stabilní, je však velice jednoduchý na pochopení i implementaci
- **Quick sort** – Jeden z nejrychlejších běžných algoritmů řazení založených na porovnávání prvků, paměťově nenáročný, funguje dobře na malých i velkých polích

Zobrazit v WORD

Selection Sort



Yellow is smallest number found

Blue is current item

Green is sorted list

Bubble Sort

6 5 3 1 8 7 2 4

Insetion Sort

6 5 3 1 8 7 2 4