Alogritmy

Algoritmus je **návod či postup jak vyřešit určitou úlohu**.

Příkladem algoritmu může být kuchyňský recept.

Matematický základ algoritmu:

- Lambda kalkul
- Turingův stroj - algoritmus je procedura proveditelná turingovým strojem

Poznámka: „Turingův stroj“ je teoretický model počítače vytvořený britským matematikem Alanem Turingem v polovině 20. Století. Jedná se konečný automat - program pro řešení jakékoli úlohy je zapsán v podobě pravidel s využitím několika základních instrukcí. Tato práce Alana Turinga se stala základem konstrukce dnešních počítačů (procesor vykonává všechny úlohy pomocí omezené sady základních instrukcí typu „přičti číslo“, „odečti číslo“, „skok na adresu v paměti“ atd.)

Algoritmus může být vyjádřen různými způsoby, například:

- Slovním popisem
- Vývojovým diagramem
- Pseudo-kódem
- Zdrojovým kódem
- Schémata, grafy

Vlastnosti algoritmů

Konečnost (finitnost)

- Musí skončit v konečném počtu kroků

Obecnost

- Neřeší jeden problém, ale třídu problémů

Determinovanost

- V každé situaci musí být jasné, co se má provést a jak má provádění pokračovat

Resultativnost (výstup)

- Musí mít aspoň jeden výstup (odpověď na řešený problém)

Elementárnost

- Skládá se z konečného počtu elementárních kroků

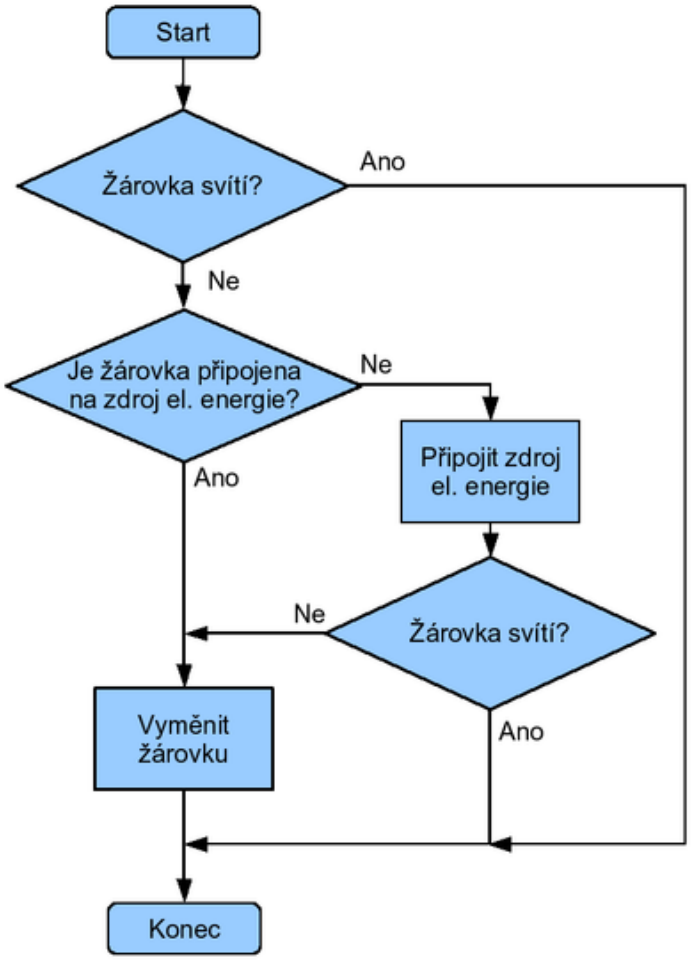
Složitost algoritmu

- **Algoritmická analýza** – zabývá se efektivitou algoritmů (jak z množiny možných algoritmů vybrat ten nejlepší)
- **Teorie složitosti** – otázka efektivity algoritmů, složitost – jak je algoritmus rychlý
- **Konečnost algoritmu** – lze úlohu vyřešit?

Některé typy algoritmů

- **Rekurzivní** – spouští ve smyčce sám sebe (dnes typické u webových aplikací – např. odeslání formuláře s daty)
- **„Hladové“** (Greedy Search) – řešení po jednotlivých rozhodnutích, která jsou-li učiněna, už nejsou revidována, lokální minima – problém „obchodního cestujícího“
- **Dynamické programování** – řeší od nejjednodušších částí po nejsložitější a využívá výsledky již vyřešených podproblémů - optimalizace
- **„rozděl a panuj“** (Divide and Conquer) – dělí problém na triviální části, které lze řešit přímo – FFT, Quick Sort, …
- **Pravděpodobnostní** (probabilistické) – binární výpočetní strom, v každém uzlu „hod mincí“
- **Genetické (evoluční)** – napodobování biologických evolučních procesů
- **Heuristické** – nemá za cíl nalezení přesného řešení, ale pouze vhodného přiblížení

Vyjádření algoritmu vývojovým diagramem (flowcharts)



Obr. 7.1 Ukázka algoritmu
[Zdroj: http://cs.wikipedia.org/wiki/Soubor:Vyvojovy_diagram_zarovka.png]

Programovací jazyky a algoritmy

Program - zápis algoritmu v programovacím jazyku

Programovací jazyk – soubor pravidel pro zápis algoritmu

Programovací jazyk má svoji **syntaxi** (souhrn pravidel) a **sémantiku** (množina slov a pravidla, která jim přiřazují význam)

Zdrojový text programu – algoritmus zapsaný v programovacím jazyku

Strojový kód – instrukce srozumitelné pro daný stroj

Instrukce – nejzákladnější příkaz, kterému rozumí CPU

Instrukční sada – soubor instrukcí, kterému rozumí dané CPU

Interpretované

- Interpretace
- Překlad do pseudokódu a jeho interpretace
- JIT (Just In Time) interpretace – před interpretací je program kompilován a optimalizován

Kompilované

Při **kompilaci** dochází k převodu programu zapsaného v programovacím jazyce do strojového kódu. Program je **kompilátorem** příslušného programovacího jazyka zkompilován pouze tehdy, pokud neobsahuje žádné syntaktické chyby. Kompilací vzniklý soubor ještě není přímo spustitelný, ale musí být sestaven do spustitelného tvaru pomocí **spojovače (linkeru)**. Linker spojí dohromady moduly a knihovní soubory podle závislostí, řeší globální proměnné a relativní adresaci. Výsledkem linkování je spustitelný program (v případě OS Windows soubor s příponou EXE).

Používané techniky optimalizace – odstranění mrtvého kódu, vkládání těl metod, rozbalení smyček, odstraněné shodných podvýrazů…

U **interpretace** je zdrojový kód čten řádek po řádku a ihned vykonáván. Výhodou je pružnost a možnost přepínání mezi vývojovým prostředím a běžícím programem. Vykonávání programu je ale pomalejší, protože na rozdíl od kompilace zde není možnost optimalizace kódu.

Chyby v programech

- **Syntaktické**
- **Chyby při zpracování**
- **Sémantické (logické) chyby**

Proces odstraňování chyb v programu se **nazývá ladění (debugging**). Prostředek pro procházení programu po jednotlivých příkazech a možností výpisu hodnoty proměnných se nazývá debuger.