# Algoritmizace a programování

**Algoritmus –** návod či postup, jak lze vyřešit daný typ úlohy, v informatice je to předpis, pomocí kterého PC provádí výpočty realizující požadované řešení problému

Aby to byl opravdu algoritmus, musí mít tyto vlastnosti:

* Konečnost: musí skončit v konečném počtu kroků, pro každý jednotlivý vstup musí být tento počet konečný
* Determinovanost: každý krok musí být přesně a jednoznačně popsán
* Vstup (univerzálnost): vstupy mají definované množiny hodnot, kterých mohou nabývat
* Výstup (rezultativnost): algoritmus má alespoň jeden výstup
* Obecnost: řeší obecnou třídu obdobných problémů, nejen jeden konkrétní
* Efektivita: operace prováděná algoritmem musí být dostatečně jednoduchá, aby bylo možné ji provést v principu pomocí tužky a papíru

Způsoby zápisu:

1. Slovním vyjádřením, obdoba kuchyňského receptu (zadej číslo a, zadej číslo b, …)
2. Pomocí diagramu, znázorňuje postup řešení (jednoduchý nákres, vývojový diagram)
3. V programovacím jazyce

**Metoda top-down** – začíná celkovým problémem a postupně se rozkládá na menší a jednodušší podproblémy, často spojena s rekurzivním způsobem řešení problémů -> nejdříve celkový problém se rozdělí na menší podproblémy, které se dále mohou dělit nebo se již vytváří algoritmy právě pro tyto podproblémy

**Ladící (debug) cyklus** – k odhalování a opravě chyb v kódu, využívá se většinou vývojové prostředí a jeho funkce

**Řídící struktury:**

* **Sekvenční (Linear):** Kódy se provádějí postupně od začátku do konce
* **Podmíněné (Conditional):** Určuje větvení programu na základě podmínky (např. if, else, switch)
* **Cyklické (Iterative):** Opakuje kód v závislosti na podmínce (např. for, while, do-while)

**Datové struktury:**

* **Pole (Array):** Skupina prvků stejného typu uložených na po sobě jdoucích paměťových místech
* **Seznam (List):** Dynamická datová struktura, která může růst nebo klesat v délce
* **Ukazatel (Pointer):** Proměnná, která obsahuje adresu v paměti jiné proměnné
* **Struktura (Structure):** Skupina různých prvků (datových typů) sdružených pod jedním názvem

**Proměnná** – pojmenovaná úložiště pro uchování dat v programu, mohou obsahovat různá data

**Identifikátory** – názvy proměnných, funkcí, tříd a dalších entit v programu

**Datové typy** – definují typ dat, který proměnná může uchovávat (např. celé číslo, desetinné číslo, znak, řetězec)

**Aritmetické operace** – k manipulaci s proměnnými (sčítání, odečítání, násobení, dělení)

**Logické operace** – negace, logický součet, logický součin, rovnost, nerovnost

**Číslicová reprezentace datových typů:**

* **Celé číslo (Integer):** Celá čísla jsou reprezentována bez desetinné části. Mohou být záporná, nula nebo kladná. Většina programovacích jazyků má různé délky celých čísel (např. 32-bit, 64-bit).
* **Desetinné číslo (Float/Double):** Desetinná čísla jsou reprezentována pomocí pohyblivé řádové čárky. Mohou reprezentovat desetinná místa a mají omezenou přesnost.
* **Řetězec (String):** Řetězce jsou složeny ze sekvencí znaků. Každý znak může být reprezentován pomocí kódové tabulky, jako např. ASCII nebo Unicode.
* **Logický (Boolean):** Logická hodnota je často reprezentována jako jeden bit (0 nebo 1, false nebo true).

**Číselné Soustavy:**

* **Desítková Soustava (Decimal):** Standardní soustava, kterou používáme v každodenním životě. Číslice jsou 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
* **Binární Soustava (Binary):** Používá číslice 0 a 1. Každá pozice v binární číselné hodnotě reprezentuje mocninu čísla 2. 1101 (13 v desítkové soustavě), 1010 (10 v desítkové soustavě).
* **Osmičková Soustava (Octal):** Používá číslice 0 až 7. Každá pozice v osmičkovém čísle reprezentuje mocninu čísla 8.
* **Hexadecimální Soustava (Hexadecimal):** Používá číslice 0 až 9 a písmena A až F pro reprezentaci čísel 10 až 15. Každá pozice v hexadecimálním čísle reprezentuje mocninu čísla 16. 1A (26 v desítkové soustavě), FF (255 v desítkové soustavě)

**Rekurzivní postupy** – program volá funkci ve funkci, tím pádem dochází k rekurzivnímu volání, může být neomezeně velké vnoření

**Iterační postupy** – program prochází iteracemi například v cyklu, který je podmíněn nějakou hodnotou proměnné, každé opakování cyklu se nazývá iterace

**Posuzování kvality algoritmů**

Funkce časové složitosti určuje, jak roste časová náročnost algoritmu vzhledem k množství vstupních dat, značí se velkým O a nazývá se asymptotická časová náročnost.

**Asymptotická časová složitost:**

* **O(1) - Konstantní časová složitost:** Algoritmus má konstantní časovou náročnost, bez ohledu na velikost vstupních dat.
* **O(log n) - Logaritmická časová složitost:** Rostoucí pomalu při zvyšující se velikosti vstupních dat.
* **O(n) - Lineární časová složitost:** Časová složitost roste lineárně s velikostí vstupu.
* **O(n log n) - Lineárně logaritmická časová složitost:** Často viděná v efektivních algoritmech řazení a vyhledávání.
* **O(n^2), O(n^3),... - Kvadratická, kubická atd. časová složitost:** Časová náročnost roste s kvadrátem, kubem a tak dále.
* **O(2^n), O(n!) - Exponenciální a faktoriální časová složitost:** Časová náročnost roste velmi rychle s velikostí vstupních dat.

**Strukturované datové typy:**

**Pole** – obsahuje prvky stejného bázového typu (typ pole), mohou být vícerozměrná, statická nebo dynamická

**Struktura** – strukturovaná proměnná z položek většinou různého typu, k položkám se může přistupovat jednotlivě pomocí jejich názvů ve struktuře (např. clovek.age)

**Abstraktní datové typy:**

**Zásobník (stack)** – k dočasnému uložení dat, char. Způsob manipulace s daty – LIFO „last in, first out“, k manipulaci s uloženými položkami využívá ukazatel, který si drží pozici poslední uložené položky

**Fronta** – typ FIFO „first in, first out“, opak zásobníku

**Seznam** – obsahuje posloupnost prvků

**Množina** – schopna uložit určité hodnoty bez jakéhokoliv pořadí a bez opakujících se hodnot

**Strom (binární)** – slouží k efektivnímu vyhledávání, třídění, ukládání a mazání, má kořen, který má maximálně dva potomky

**Tvorba uživatelských funkcí**

* Cílem je poskytnutí uživatelsky přívětivého prostředí a funkčního rozhraní
* Výzkum -> Definování cílů -> Návrh UI -> Vytvoření Wireframů -> Vývoj funkcí -> Testování -> Dokumentace -> Vydání, monitoring a aktualizace

**Princip práce se soubory**

* soubor se musí nejdříve otevřít, pak se čte (buď po řádcích nebo celý soubor najednou), zápis do souboru, zavření souboru k uvolnění systémových prostředků
* většinu těchto funkcí poskytují základní knihovny

**Přidělování paměti**

* statická, dynamická a automatická přiřazování

Statická – přidělování během kompilace programu a zůstává pevně přidělena po celý běh programu

Dynamická – za běhu, program žádá o alokaci paměti na haldě (heap), lze se dotazovat pomocí příkazu v části kódu a také tento prostor uvolnit

Automatická – spojena s proměnnými, které se vytváří za běhu programu, většinou uvnitř funkce, proměnné se alokují na zásobníku

**Základní algoritmy třídění**

Bubble sort – porovnává a vyměňuje sousedící prvky, dokud nejsou ve správném pořadí, může být neefektivní pro velké záznamy

Insertion sort – postupně vkládá prvky na správná místa tak, aby prvky nalevo od aktuální pozice seřazené, efektivní u malých a téměř seřazených seznamů

Selection sort – vybírá nejmenší prvek a umisťuje ho na začátek seznamu, postupně opakuje tento postup pro zbytek seznamu

Dále Quicksort, Mergesort, Heapsort

**Základní algoritmy hledání**

Lineární hledání – prochází všechny prvky postupně a porovnává je s hledaným prvkem, dokud nenajde shodu

Binární hledání – předpokládá, že je seznam seřazený, porovnává hledaný prvek s prvkem se středu seznamu a pokračuje v hledání v polovině, kde by prvek mohl být

Interpolační hledání – prvky musí být rovnoměrně rozložené, odhaduje polohu hledaného prvku na základě jeho hodnoty a aktuálních hodnot na krajích

**Numerická derivace** – derivace v bodě pomocí konečných derivací, diference se používají k aproximaci sklonu, základní jsou diference vpřed a vzad

**Numerická integrace** – metoda lichoběžníků – aproximuje plochu jako součet ploch lichoběžníků, metoda pravoúhlých obdélníků – pomocí obdélníků, jejichž výška je rovna hodnotě funkce v levém konci intervalu

**Aproximace metodou nejmenších čtverců** – často v regresní analýze a aproximaci funkcí, nalezení parametrů modelu tak, aby byla minimalizována kvadratická odchylka mezi daty a predikovanými hodnotami