

5. 9. 2022

## Informatika úvod

- oboř: - zkoumá strukturu, chování a interakce navržených a přirozených výpočetních systémů
- člověk: - orientace na výsledek s cílem získat co je požadováno (deklarativní přístup)
- počítač: - nástroj provádějící výpočetní proces určující jak dojít k výsledku (imperativní přístup)
- zaměření: - podle oblasti související s informatikou a s příklady na různé úrovni abstrakce
  - automatické zpracování a transformace dat, informací a znalostí pro účely inteligence
- automatické: - materiálová věda, technické vybavení počítače CPU, GPU, TPU, ...
- zpracování: - spočetnost, automaty, počítače, proces, OS, FW, ovladače, OS

15. 9. 2022

## Pokračování

- transformace: - logika, formální jazyky, algoritmy, programování, programy, řízení, virtualizace
- data: - kódování, šifrování, datorová věda a inženýrství, databáze, grafika, SQL, HTML, JSON
- informace: - teorie informace, informační věda a inženýrství, systémy a technologie
- znalosti: - knihověda, znalostní věda a inženýrství, expertní systémy a strojové učení (ML)
- inteligence: - kognitivní věda, modely, inteligence, neuronové sítě, člověk a umělá inteligence
- využití: - informatiku může inteligence použít v různých situacích vyžadujících rozhodování
  - výhodnocení: - poskytnout odpověď na předložené turzení týkajícího se předmětu dotazu
  - předpověď: - odhad možného budoucího stavu podle aktuálního stavu a dalších informací
  - rizika: - posouzení a minimalizace možných rizik a negativních vlivů pro budoucí vývoj
  - řízení: - ovlivňování aktuálního stavu systému s cílem dosáhnout požadovaný stav
  - plánování: - návrh budoucího stavu s ohledem na požadované vlastnosti
  - rozvrhování: - přiřazených dostupných zdrojů požadavkům a jejich správa a řízení
  - zdokonalování: - zvyšení efektivity a schopnosti systému poskytující nové možnosti využití

- úrovňě: - rozdělení zaměření podle úrovňě abstrakce (obecnosti), použitých nástrojů a výstupů
- teorie: - formálně definuje, popisuje a dokazuje základní vlastnosti matematických objektů
- věda: - na základě teorie zkoumá možnosti a limity objektů nebo způsoby, jak je tvorit
- nástroje: - samostatně fungující prvek umožňující provádět akci nebo operaci s objektem
- systémy: - soubor nástrojů specializujících se na různé oblasti a poskytuje služby
- technologie: - propojení nástrojů a systémů do jednoho celku využitelného pro aplikaci
- využití: - nasazení kombinace vybraných technologií do praxe se stanoveným cílem

19. 9. 2022

## Pokračování

- specializace: - informatika podle úrovni abstrakce, použitých nástrojů
  - matematická: - matematika využitá pro informatiku
  - teoretická: - základ pro výzkum
  - inženýrská: - tvorba jednotlivých nástrojů a samostatně fungujících součástí
  - praktická: - návrh celých systémů
  - aplikovaná: - informační technologie, jako soubor fyzického vybavení i programů
  - technická: - návrh HW, tj. fyzických přímo zpracovávajících informace
- činnosti: - provádění mezi různými úrovňemi specializace
  - předpoklady: - určení toho co je nebo není možné dosáhnout
  - výzkum: - cílem je najít neznámý postup
  - vývoj: - vytvoření nástrojů používající postupy od výzkumu
  - realizace: - využití nástrojů pro vytvoření technologie řešící typ úlohy
  - aplikace: - přizpůsobení a nasazení technologie do praxe

## Základy Informatiky

- počítač: = programy jako nedělitelná část hardwaru
- informatika: = zaměřuje se na přenos, zpracování a využití informací
- větva: = výstupy větví na sebe navazují, tj. matematická → teoretická → praktická aplikace
- informace: = základ pro rozhodování a možná výhoda proti ostatním

26. 9. 2022

## Historie Informatiky

- obrázek: = reprezentuje grafický znělý popisovaného objektu na různé úrovni detailů
- písmo: = uchování informací a významu v symbolické podobě pro přenos prostorem a časem
  - potřeba = sdílený a spolehlivý způsob přenosu informací
- kódování: = přenos informace do podoby strojově zpracovatelného kódu
- proces: = transformace kódované informace výpočetním procesem pomocí výpočetního modelu

## Programovací jazyky s imperativním přístupem

- programovací: - umožní popsat transformaci dat i jejich struktury definovanými pravidly
- algoritmus: - návod pro provedení výpočetní procedury s cílem získat požadovaný výsledek
  - obraz vstupní hodnoty transformuje na obraz výstupní hodnoty
- program: - realizace algoritmu v programovacím jazyce s možností generovat výpočetní proces
- paradigma = přístupy a nástroje použitelné pro popis výpočetního procesu algoritmem
- imperativní: - program určuje (řídí) počítací změnu stavu od počátečního (vstup) na koncový (výstup)
  - změna stavu je docílena změnou hodnoty v paměti počítací příkazy → vedlejší efekt
- nestrukturované = řízení výpočetního procesu příkazy skoku GOTO (nepodmíněný a podmíněný)
- strojový kód = (1883) A. Lovelace, C. Babbage, mechanický počítací Analytical Engine
- (absolutní) = řídí činnost hardware řadičem překládajícího kód na operace
- asembler = (1947) první abstrakce od strojového kódu k přirozenému jazyku
- (relativní) = absolutní kódy instrukcí a míst v paměti nahrazeny jejich názvy (obrazy)
- blokové = (1950) kódy příkazů sestupeny do sekvence s určitým významem

17.10.2022

## Pokračování

- 1. překladač = (1952) A. Glennie, překlad z vyšších jazyků do asembleru a strojového kódu
- COBOL = (1959) G. Hopper, databáze a finanční aplikace
- BASIC = (1964) Beginners All-purpose Symbolic Instruction Code
- strukturované = proces jako sekvence, rozhodování, opakování a rekurence příkazů
  - FORTRAN = (1957) J. Backus, FORMula TRANslation, matematické výpočty
  - C = (1972) D. Ritchie, vyšší jazyk, systémové programování, nezávislý na hardware
  - MATLAB = (1978) MATrix LABoratory, výkonné matematické programy
  - Perl = (1987) skriptovací jazyk, zpracování výstupů, operační systém UNIX
  - R = (1995) statistická analýza a data mining, informatika
- objektové (OOP) = (1973) objekt obsahuje data i procedury pro práci - přístup, čtení, změna dat
  - Objective-C = (1983) jazyk C s podporou OOP, pro vývoj programů v Apple
  - C++ = (1983) jazyk C s podporou OOP, zaměření na výkon programů
  - Python = (1991) standardní knihovna a snadné připojení dalších knihoven
  - Java = (1995) virtuální stroj JVM, multiplatformní, automatická správa paměti

## Pokračování

- měření: - C. Shannon (1948) matematicky popsal jak měřit množství informace ve zprávě
  - jednotka = bit jako množství informace nutné k rozhodnutí mezi dvěma výsledky
- fyzika: - propojení informace a termodynamiky popisující přenos energie ve formě tepla
- médium = informaci vždy přenáší a uchovává fyzický systém
- Maxwellův démon = J.C. Maxwell (1867), experiment získání energie pomocí informace
- Limity:
  - odvozená omezení realného světa podle poznání fyzikálních zákonů
    - R. Landautem (1961) = minimum energie pro přečtení 1bitu
    - H.J. Bremermann (1962) = maximální rychlosť fyzického výpočetního systému
    - J. Bekenstein (1981) = maximum informace dokonale popisující fyzický systém

## Jazyky přirozené a formální pro komunikaci a data

- jazyk:
  - symbol kóduje informaci a umožňuje přenést význam mezi informačními systémy
  - abeceda = sada symbolů jazyka, např. binární, číslice 0-9, latinka, ...
  - slova = sestaveny z abecedy jazyka a přenášejí jistý dohodnutý a sdílený význam

3.10. 2022

## Pokračování

- gramatika = sada pravidel jazyka pro sestavování struktury zprávy
- kontext = může měnit význam slova v závislosti jiných slov
- druhý jazyků: - jazyky lze rozdělit podle způsobu vzniku na přirozené a formální
  - přirozený = využívá se postupně při komunikaci mezi lidmi
  - formální = N. Chomsky (1956) matematicky definoval 4 typy gramatik,
  - typy = regulární (typ-3), bezkontextové (2), kontextové (1), rekurevně spočetné (0)
  - zpracovatelnost = strojově (3), konečný automat (2), zásobníkový automat (1), Turingův stroj (0)
- komunikace: - lidé (přirozený), člověk-počítací (informační, programovací), počítací (datové, strojový kód)
  - přirozená = vrozené prostředky pro komunikaci s okolím
  - symbolická = přirozené a strojově zpracovatelné jazyky užívající symboly
  - člověk-počítací = pro člověka snadnější čitelnost, pochopení a upravování informací
  - počítací-počítací = efektivní výměna informací mezi počítači nebo aplikacemi binárním kódem
- protokoly: - zajišťující přenos informací mezi systémy a jeho řízení dle dohodnutého protokolu
  - ITA1, ITA2 = É. Baudot (1880), CCITT (1924), telegraf, binární kód
  - ASCII = ASA/ANSI (1961), American standard code for Inf. Interchange
  - TCP, IP = R. Kahn, V. Cerf (1974), komunikace mezi počítači v decentralizované síti
  - HTTP = T. Berners-Lee (1989), přenos souborů mezi klientem a serverem na webu
- jazyky: - poskytují jistou úrovně abstrakce od binárního kódu a mechanismů technologie
  - nižší = popisují chování stroje a generování procesu změnou stavů (blížší strojovému kódu)
  - vyšší = popisují požadavky na výstup (blížší přirozenému jazyku) - specifikace  $\Rightarrow$  deklarativní

10.10. 2022

## Pokračování

- datové: - symbolicky přenášejí strojově zpracovatelný obsah a strukturu informace
  - HTML = T. Berners-Lee (1989), popis obsahu a struktury hypertextových dokumentů
  - CSS = H. Lie, B. Bos (1996), popis vzhledu a řízení vykreslování struktury HTML dokumentu
  - XML = T. Bray (1996), výměna strukturovaných informací v textové podobě mezi aplikacemi
  - SVG = W3C (2001), popis vektorové grafiky pomocí XML
  - JSON = R. Hickey (2012), datový formát pro textový přenos dat objektů mezi aplikacemi

## Pokračování

31. 10. 2022

- události = řízené = nadstavba OOP při změně dat objektu může být spuštěn kód programu
- Visual Basic = (1991) Microsoft, grafické aplikace částečně pomocí vizuálních nástrojů a IDE
- JavaScript = (1995) aplikace pro webový prohlížeč, dnes na server i pro OS
- C# = (2000) Windows aplikace, knihovna .NET, IDE visual studio
- Scratch = (2003) výukový nástroj pro děti, blokový vizuální programování
- aspektové = rozdělení programu od samostatných modulů s vymezenou funkcionálitou
  - Groovy = (2003) programovací a skriptovací jazyk pro Java (JVM)
- souběžné = (1965) vykonávání více toků instrukcí zároveň v jednom okamžiku bez blokování
  - Go = (2009) obdoba C, produktivita v síťích a u vícejádrových systémů
  - Rust = (2010) výkonný, také funkcionální a s automatickou správou paměti
  - Kotlin = (2011) multiplatformní, propojen s Java, aplikace pro Android
  - Swift = (2014) propojení s Objective-C knihovnami, Apple, Win 10, Android

## Programovací jazyky s deklarativními a jinými přístupy

- nástroj = cokoliv co není součástí těla a umožní splnění úlohy (fyzický předmět i logický posun)
- používané = i zvířata používají nástroje pro řešení problémů ve svém prostředí (např. mravenec)
- výroba = vytvoření díky naučené znalosti pomocí dostupných prostředků (materiál)
- znalost = není součástí nástroje, ale umožní popsat proces vytváření či používání nástroje
- technologie = soubor nástrojů a znalostí jak je vytvořit a použít v prostředí
- lidé = pomocí nástrojů vytvářejí další nástroje, kterými transformují okolní prostředí
- jazyk = psaný (dříve mluvený) mů schopnost měnit způsob jakým lidé myslí a řeší problémy
- matematika = logika poskytuje objektivní a univerzální způsob jak dojít k řešení úlohy
  - $\lambda$  calculus = A. Church (1930), formální systém pro popis výpočetního procesu pomocí funkcí

7.11. 2022

## Pokračování

- formální = podle využití rozlišujeme Domain Specific Languages a General Purpose Languages
  - DSL = specializované na určitý typ úloh = doménu (např. Logo, HTML, CSS, MATLAB)
  - GPL = obecně použitelné pro různé domény (např. XML, C, Java, Python)

- deklarativní = program v jazyce pomocí dat a jejich struktury specifikuje požadavek na výsledek
- funkcionální = výsledek y je ze vstupní hodnoty x získán transformací pomocí funkce f:  $y = f(x)$ 
  - funkce : - definice vstupní a výstupní hodnotou nebo složením z již definovaných funkcí
    - např. logická, aritmetická, textová, informační (vyhledávání, řadiči), statická
- LISP : - J. McCarthy (1958) LISP Processing, základ  $\lambda$ -calculus, výzkum umělé inteligence
  - program popsán jako datová struktura a vyhodnocením pomocí definic získáme výsledek
  - interaktivní vývoj software pomocí nástroje REPL místo překládání
  - metaprogramování - rozšíření syntaxe jazyka pomocí make - program generující program
- Scheme : - G. Steele, G. Sussmann (1978), odvozen z LISPu  $\rightarrow$  dialekt, IEEE standard
  - tail-call optimalizace pro rekurezivní programy a lexikální užívání
  - first-class continuation umožní práci s nekonečnými hodnotami pomocí generování sekvencí
- ML = R. Milner (1973), Meta Language, tvorba překladačů, formální kontrola programů
- Common LISP = ANSI (1984) dialekt jazyka LISP, ANSI standard, podpora mnoha různých přístupů
- Haskell = (1990) čistě funkcionální, výzkum, vývoj a průmyslové aplikace, systém datových typů
- Racket = PLT Inc. (1995), dialekt Scheme, hárk programovacích jazyků a jejich realizace
- F# = Microsoft (2005), vychází z ML, pro .NET Framework a virtuální stroj CLR
- a další... = JavaScript, R, XQuery, SQL, C++, C#, Kotlin, Perl, GO, Rust, Java 8
- souběžný = více toků (vlákna) vykonávají, pracuje souběžně se sdílenými daty
- Erlang / OTP : - Ericsson (1986), open telecom platform, ústředny a mobilní sítě GPRS, 3G, LTE
  - systémové programování (distribuované - mnoho uzlů v síti, RTC - běh v reálném čase)
  - spolehlivost (tolerující chyby, vysoká dostupnost, nezastavující oprava programu za běhu)

14.11. 2022

### Pokračování

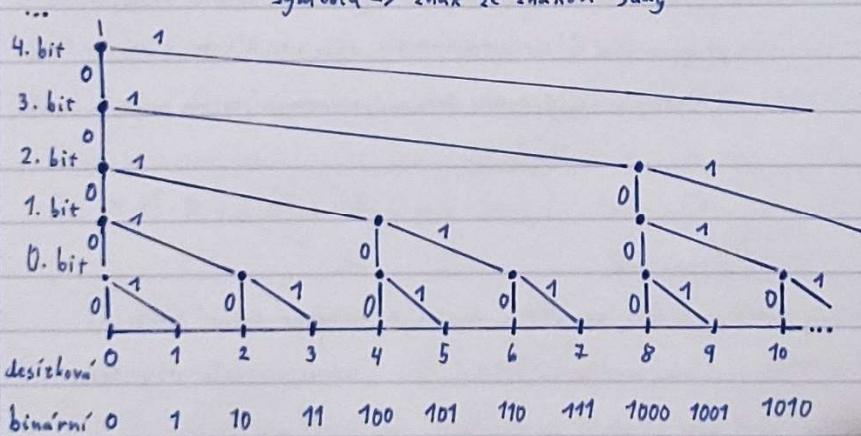
- Clojure = R. Hickey (2007), z LISPu, pro JVM a JavaScript, neměnné datové struktury
- Elixir = J. Valim (2012), systémový, nadstavba Erlang s jeho výhodami, virtuální stroj Beam
- dotazovací = program jako specifikace požadavku na databázový nebo informační systém
- SQL = D. Chamberlin, R. Boyce (1974) práce s daty a databázemi pomocí systému dotazů
- XPath = W3C (1999) výrazový jazyk popisující dotazování a transformaci XML
- LINQ = Microsoft (2003) dotazování nad datovými strukturami pro C# a .NET Framework

- omezující = programu zadaná sada omezujících pravidel pro hledání řešení
- logické = program tvorí sada logických pravidel umožňující odvozování a odpovídání na otázky
  - Prolog = A. Colmerauer, R. Kowalski (1972), vývoj umělé inteligence a výpočetní lingvistiky
- ontologické = kódování a výpočetní zpracování informací a znalostí pomocí pravidel a odvozování
- datové toky = program jako orientovaný graf, kde uzel představuje funkce a hrany přenáší datu
- reaktivní = prostup změn dat v datových proudech a jejich vzájemné závislosti
- VHDL, Verilog = IEEE (1984), jazyk HDL (Hardware Description Language), popis hardwaru
- SystemVerilog = (2002) jazyk pro specifikaci (model, návrh) a ověřování (simulace testování) HW
- Simulink = (2002), MATLAB, tvorba modelů, simulaci a analýzy dynamických systémů
- automatové = různé typy automatů, např. konečný, zásobníkový nebo buňkový
- současně = model aktori, agentové programování (např. více spolu pracujících bot)
- ne deterministické = dopředu není určen tok programu a vyučívá se mu náhodné rozdělování
- paralelní = podobně jako současně, více programů vykonáváho v jedné okamžik

21. 11. 2022

### Binární Kódování

- abeceda  $\Rightarrow$  dva symboly (např. 0 a 1, zdroba a čárka,...)
- kódování významu: - hodnotu čísla  $\Rightarrow$  velikost (směr)
- symboly  $\Rightarrow$  znak ze znakové sady



- řetězec 1 a 001 vyjadřují stejnou hodnotu

16.01.2023

### Množina

- Prázdná množina - neobsahuje žádny prvek (objekt)
- má strukturu  $\Rightarrow$  umožňuje operace  
(např. vkládání a dálší)
- zápis  $\Rightarrow \emptyset \{ \}$

- Velikost množiny - počet prvků v množině  $n \in N^*$   
(celé číslo včetně nuly)

$$M = \{a, b, c\} \\ |M| = 3 \\ |\emptyset| = |\{\}| = 0 \quad \left. \right\} \text{konečné množiny (např. soubor)}$$

$$|N| = \infty \quad \left. \right\} \text{neukončená (nekonečná) množina (např. stream)}$$

23.01.2023

022

$$\{1, 2, -5, 4, 4, 7\}$$

• Seřadit  $(\{1, 2, -5, 4, 4, 7\}, f(x, y) = x < y)$

$$= (-5, 1, 2, 4, 4, 7) \quad \leftarrow \text{uspořádaná } n\text{-tice} \quad \left. \right\} \text{že zvolit dle} \\ = [-5, 1, 2, 4, 4, 7] \quad \leftarrow \text{vektor} \quad \left. \right\} \text{pořadí}$$

$$(-5, 1, 2, 4, 4, 7)_3 = 4$$

- množinná funkce provede na sekvenci (uspořádaná n-tice)
  - pomocí funkce která určí pořadí dvou zadaných (libovolných) prvků
- $f(x, y)$   
 $x < y$   
 $x > y$

- Rovnost množin  $\Rightarrow$  na počtu prvků a pořadí NEZAŽEJÍ

$$\emptyset = \{\}$$

$$\begin{aligned}\{1, 3\} &= \{3, 1\} \\ \{1, 1, 3\} &= \{3, 1\} \\ \{\{1\}\} &\neq \{1\} \\ \{1, \{2\}\} &\neq \{\{2\}, \{1\}\} \\ \{x\} &\neq \{y\}\end{aligned}$$

• odebrání prvků (objektů) z množiny

$$\begin{aligned}\{4\} \setminus \{1, 2, 3, 4\} &\Rightarrow \{1, 2, 3\} \\ 4 \cup \{1, 2, 3\} &\Rightarrow \{1, 2, 3, 4\} \\ x \cup A &= B \\ \{x\} \cup A &= B\end{aligned}$$

jednoprvká množina  
obsahující objekt  $x$

$\Rightarrow x \in B \iff x \text{ je obsažen v } B$

$$\begin{aligned}\{x\} \cup \{y\} &= \{x\} \quad (\text{komutativní}) \\ \{x\} \cup \emptyset &= \{x\}\end{aligned}$$

30.01.2023

Kartézský součin (Cartesian product)

- množina všech uspořádáních dvojic, které vzniknou kombinacemi prvků z množin  $A \times B$

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$$

$$A = \{1, 2, 3\}$$

$$B = \{x, y\}$$

$$A \times B = \{(1, x), (1, y), (2, x), (2, y), (3, x), (3, y)\}$$

$$A \Delta B = (A \setminus B) \cup (B \setminus A)$$

$$A \oplus B = A \Delta B$$

$$\begin{aligned} \{A, H, O, J\} \Delta \{H, O, N, Z, O\} \\ = \{N, Z\} \cup \{A, J\} \\ = \{N, Z, A, J\} \end{aligned}$$

$\Rightarrow$  Výsledná množina obsahuje prvky které nejsou v průniku množin  $A, B$

### Rozdíl množin (set difference)

-výsledkem je množina

$$A \setminus B = \{x \mid x \in A \wedge x \notin B\}$$

$$A - B = A \setminus B$$

-množina která obsahuje prvky z množiny prvků  $A$  množiny  $B$  - bez prvků z množiny  $B$

$$\{H, O, N, Z, O\} \setminus \{A, H, O, J\} = \{N, Z\}$$

$$\{A, H, O, J\} \setminus \{H, O, N, Z, O\} = \{A, J\}$$

### Sjednocení (union)

$$A \cup B = \{x \mid x \in A \vee x \in B\}$$

-výsledkem sjednocení množin  $A, B$   $A \cup B$  je množina obsahující všechny prvky z  $A$  i z  $B$

$$\{A, H, O, J\} \cup \{H, O, N, Z, O\} = \{A, H, O, J, N, Z\}$$

### Přenik (intersection)

$$A \cap B = \{x \mid x \in A \wedge x \in B\}$$

-výsledkem průniku množin  $A, B$  ;  $A \cap B$  je množina obsahující prvky které jsou v množině  $A$  a zároveň v množině  $B$

$$\{A, H, O, J\} \cap \{H, O, N, Z, O\} = \{H, O\}$$

### Podmnožina

$$\Rightarrow A \subset B \Rightarrow A \text{ je/není podmnožina } B$$

$A \not\subset B$  (není)

• operace (funkce) pracuje se dvěma množinami  $A, B$   
a je výsledkem  $\Rightarrow$  výsledkem je pravda/nepravda (true/false)

•  $A \subset B$  platí (výsledek pravda) když množina  $B$  obsahuje  
všechny prvky z  $A$

$$A = \{1, 3\}$$

$$B = \{1, 3, 5, 7\}$$

$$C = \{3, 1\}$$

$$\Rightarrow A \subset B, B \not\subset A \Rightarrow B \neq A$$

$$A \subset A \quad B \subset B$$

$$\Rightarrow A = C \Rightarrow A \subset C, C \subset A$$

06.02.2023

$$\begin{aligned} & \{a_1, a_2, a_3\} \quad (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1) \\ & P(A) = 2^4 = \left\{ \begin{array}{l} \{ \} \\ \{a_1\}, \{a_2\}, \{a_3\}, \{a_1, a_2\}, \{a_1, a_3\}, \{a_2, a_3\}, \{a_1, a_2, a_3\}, \{a_1, a_2, a_3, a_4\} \end{array} \right\} \\ & |P(A)| = 2^{|A|} = 2^{|\{a_1, a_2, a_3\}|} = 2^3 = 8 \rightarrow \text{velikost potenciální množiny} \end{aligned}$$

- využití - kolika různými způsoby lze poskládat objednávku  
z  $N$  položek
  - kolik existuje variant konfigurací zapnuté/vypnuté pro  $N$  různých voleb

• příklad:  $A = \{x, y\}$

$$P(A) = ?$$

$$|P(A)| = ?$$

## Uchovávání paměti v PC

13.2. 2023

- Paměť: - skládá se z paměťových buněk uchovávající jednotlivé bity informace
- Operační: - nejmenší průměrně adresovatelnou velikostí paměti je 1 bajt (8bitů)
- Typy paměti:
  - Podle možnosti zpracovávání dat ponocí CPU
    - Primární (hlavní)
      - Prístupná přímo CPU
      - Cache, registry, RAM
    - Sekundární (vedlejší)
      - nepřímo přístupná CPU
      - HDD, SSD, USB Flash, ...
    - Tertiální (doplňková)
      - nepřímo přístupná ponocí CPU
      - Sítě, cloud, webové služby

1)

27.2. 2023

2.3) - Typy dat a jejich struktura

- Primitivní typy

- Jednoduchá vnitřní struktura → definovanou pevnou velikostí
- Uchovávají hodnotu, jako řetězce bitových hodnot
- Úzká výužitkovost na používání HW

- Boolean

- bool

- logická hodnota, rozlišuje pouze dva možné stavy hodnot

- datová velikost 1b, typicky vše 1B

- Integer

- int

- celé číslo v zadáném rozsahu, běžná datová velikost 1-4B

- rozsah  $0-2^N-1$  (kladné č.)  $-2^{N-1}$  až  $2^N-1$  (kladné i záporné) ( $N = \text{bity}$ )

### - Enumerated

- enum

- Výpočtový typ, konečná množina všech hodnot

- Hodnoty jsou do paměti kódována jako jednotlivá řetězce čísel

### - Charakter

- char

- znak nebo symbol ze znakové sady

- datová velikost ASCII = 1B, UTF-8 podle znaku 1-4B

6.3. 2023

### - Floating Point

- float

- reálné desetinné číslo se základní přesností (7 číslic a exponent)

- číslo → plavoucí fiótkou např.  $1,234 \cdot 10^{-5}$

- je uloženo jako základ, súčin a exponent

- přesnost je konečná

- běžná datová velikost 4B

### - Double, Real

- double

- Reálné číslo s dvojitou přesností

- větší rozsah a přesnost od Floating pointu

- datově většinou 2x větší než Floating point

### - Složené datové typy

- pevně zadání struktura dat s volitelnou velikostí

- Mohou uchovávat hodnoty libovolného předmětu definovaného typu

13. 3 2023

- Array

- pole, pevně zadáný počet položek jednoho zvoleného datového typu

- Record, struct, tuple

- záříznam zadaný počet položek libovolného typu v definovaném pořadí

- položky různého typu a přístup pomocí názvů

- popřípadě pomocí čísla dle pořadí

- Union, variant, var

- uchovává hodnotu v různých datových typech v jeden okamžik jen jednu

- Tagged Union, variant

- Jako Union, navíc informace o aktuálně používaném typu

- String

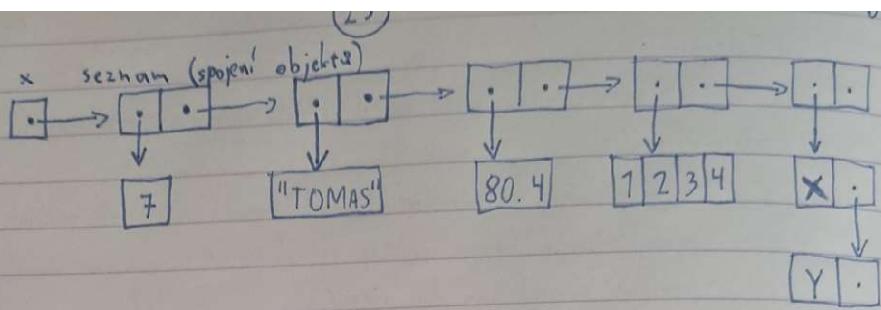
- Řetězec, jedná se o pole, znaků, pevná velikost (256B)

- Abstraktní typy

- volně zadaná struktura dat s proměnnou velikostí

- Mohou uchovávat hodnoty libovolného zvoleného typu

- Uchovávané položky zvané Items v datové struktuře



- container, class, object  $\Rightarrow$  kontejner, třída, objekt

- uchovává data i algoritmy

- přístupné oprávnění k položkám (např. private, protected, public)

```
let souradnice = {x:3, y:-5, ← data
                  souradnice: function(){
                    return math.floor(x*x + y*y)
                  }
                }
```

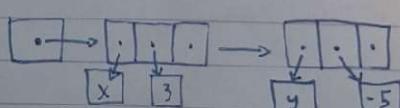
- List - seznam, uchovává hodnoty zvoleného datového typu

- uplatnění podobný jako pole, proměnná velikost

- abstrakcionalizace array, map - asociální pole, flash tabulka, slovník, učebníkův seznam polícií

- položka je uspořádaná dvojice (unikátní klíč - hodnota, key - value)

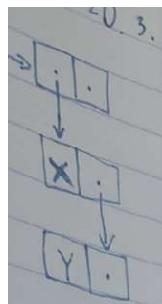
- multimap = jako map, jeden klíč může mít více hodnot



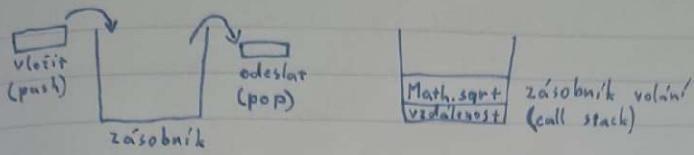
- set, enum - množina, výpočetový typ, sada více odlišitelných položek

- každá položka má přiřazenou unikátní hodnotu (celé číslo)

- multiset, bag - multimnožina, batoh, položka může být v množině vícekrát



- stack → zásobník, paměť typu LIFO (last-in first-out)
- např. zásobník volání v OS



- quul → fronta, paměť typu FIFO (first-in first-out)
- např. fronta událostí v OS

- priority quul → odbovení záleží na prioritě položky (vyšší priorita = přednostnější)
- např. fronta procesů v OS



(x · x + y · y)

(26)

27. 3.

- graph → graf, data uloženy jako vrcholy, propojeny hrázdem
- např. webové stránky propojeny odkazy

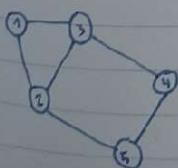
- tree → strom → graf který neobsahuje cykly
- např. souborový systém ve Windows

- heap → haldy je speciální varianta stromu určená pro rychlé vyhledávání
- např. správa operační paměti v OS

(27)

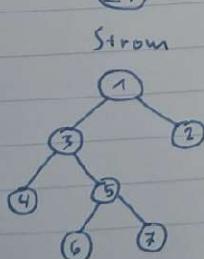
3. 4.

Graf

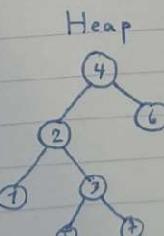


celých políček  
hodnota (celé čísla)  
množině vícekrát

Strom



Heap



## Podpora v programovacích jazycích

- nízko-úrovnové - assembly ASM, JSA
  - podpora datových struktur = chybi
  - základní typ = binární hodnota
  - složený typ = bitový řetězec o různé pěnně zadané délce,
    - 8bitů → 1 bajt, ...
  - zápis v hexadecimální soustavě
- uložené hodnoty = registry, dvojice registrů

- nižší jazyky
  - jazyk C, původně Visual Basic
  - podpora datových struktur = pole (array), záznam (record)
  - pole jednorozměrné i vícerozměrné
  - základní typ = celé číslo, desetinná čísla (float), znak
  - složený typ = textový řetězec (string), pole, záznam
  - uložené hodnoty = proměnné

- vyšší jazyky
  - objective C, Pascal, objective Pascal, C++, PHP
  - podpora datových struktur = stejná jako u nižších jazyků našiká
  - např. C++ má STL (Standard Template Library)
  - omezená podpora abstraktních datových struktur
  - základní typ = celé a desetinné číslo, znak, string, pole záznam
  - složený typ = objekt, třída

(28)

17. 4.

## Úvod do Algoritmů

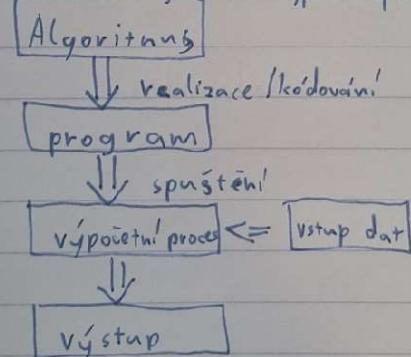
- algoritmus - nezávislý univerzální způsob pro popis a vývoj výpočetního procesu
- lze jej zakódovat do podoby instrukcí a provést jej
- spuštěním programu se zahají vývoj výpočetního procesu
- cíl  $\Rightarrow$  řešit efektivně tržidu problémů výpočetním procesem
- umožňuje automatizování provádět výpočty, zpracování dat a rozhodování
- algoritmus komprimuje a umožní generovat výpočetní proces

- výpočetní proces - venku prováděním algoritmu podle jeho instrukcí, nad/s hodnoty

- skládá se z jednotlivých proveditelných kroků - příkazů

- kroky lze strojově

- tím jak se výpočetní proces využije v krocích může generovat výstupní data



- data - zakódovaná informace zpracovávaná v krocích procesu

- algoritmus je kódován podle evolučního programovacího jazyka

- lze jej převést na strojový kód

(29)

11.5.

## Složitost Algoritmu

- algoritmus generuje výpočetní proces
- druhý složitosti → podle velikosti vstupu (počet hodnot)
- paměťová → kolik bude potřeba pracovní paměti
- časová → kolik kroků výpočtu bude potřeba pro dokončení výp. procesu
- nelesající funkci → větší vstup → stejná nebo vyšší složitost
- ovlivnění složitosti → velké vstupy nebo určité hodnoty
- např.: násobení 0 ⇒ výsledek vždy 0

- faktorování:

$$n=5 \quad \text{počet operací } n! \text{ (faktoriał)}$$

$$5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = \underline{120}$$

$$n=6 \quad 6! = 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = \underline{720}$$

$$n=7 \quad 7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = \underline{5040}$$

- ponížení: např. kolik existuje různých možností jak seřadit N polokek

- exponenciální

$$2^n = \underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{n\text{-krát}}$$

## Algoritmy

- efektivní metoda - proces generovaný algoritmem je v konečném čase a prostoru
- lze jej dokončit po konečném počtu kroků s konečnou vel. paměti
- formalizace - koncept algoritmu existuje již několik let, formalizován až v 20. st.
- formální podoba souvisí s matematickým popisem vlastnosti algoritmu

(30)

15.5.

Doplňení 1a H (Doplňek množiny)

- complement  $A^c = A' = \{a \in U : a \notin A\}$  je množina všech možných  
 $\Rightarrow A \subseteq U, A^c \subseteq U$  (universal set)

- všechny průkly z množiny  $U$  které nejsou v množině  $A$   
 $A^c = U \setminus A = U - A$

### Asymptotická složitost

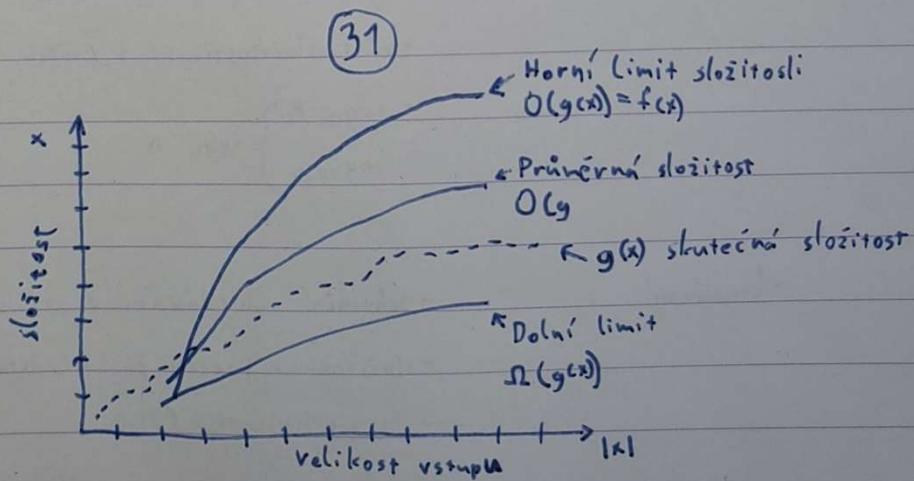
- určuje jak rychle - růadově roste složitost algoritmu s velikostí vstupu
- $O$ -omicron = horní limit (v nejhorsím případě)
- $\Omega$ -omega = dolní limit (v nejlepším případě)
- $\Theta$ -tetha = průměrná složitost

$\Rightarrow$  vstup  $x$ , velikost vstupu  $|x|$

$\Rightarrow$  funkce složitosti - porovnání

$g(x)$  - skutečná složitost

$f(x)$  - asymptotická složitost



$\Rightarrow$  platí:  $O(g(x)) = f(x) > g(x)$

$\Omega(g(x)) < g(x)$

(32)

22. 5.

- běžné třídy funkcí časové složitosti  $O, \Omega, \Theta$

- velikost vstupu - počet jednotlivých položek obsažených ve vstupu  $x$ , značeno  $n = |x|$

- běžné třídy funkcí - uvedené třídy jsou seřazeny od nejpomalejší k nejrychlejší růstancům

Označení třídy

 $O(1)$ 

Název třídy

- konstantní

Příklady operací

- slot na prvek v polí (array)

- hodnota složitosti se s velikostí vstupu nemění

- 30

$$1000 \Rightarrow O(1)$$

$$10^5$$

 $O(\log_a \log_b n)$ 

- dvojitě logaritmická

- mají některé vyhledávací algoritmy

- roste s počtem prvků nejpomalejší

- s počtem prvků se růst dramaticky zpomaluje

$$\frac{\log_2 \log_2 n}{\log_{10} \log_{10} n} \Rightarrow \log_a \log_b n$$

 $O(\log_a n)$ 

- Logaritmická

- vyhledávání půlením intervalu v seřazeném polí

- růst složitosti se s počtem prvků zpomaluje

$$\left. \begin{aligned} &\log_2 n \\ &\log_{10} n \end{aligned} \right\} \log_a n$$

 $O(n)$ 

- Lineární

- seřazení vyhledávání v seřazeném seznamu

- složitost  $\rightarrow$  počtem prvků roste konstantně

$$30n + 1000 \Rightarrow O(n)$$

 $O(n \cdot \log_a n)$ 

- lineárně-logaritmická

- seřazení podle prvků přímým porovnáváním

$$n \cdot \log_2 n + 1000 \Rightarrow O(n \cdot \log n)$$

$$n = 1000 \Rightarrow 1000 \cdot 10 + 1000$$

$$n = 1 \cdot 10^9 \Rightarrow 1 \cdot 10^9 \cdot 30 + 1000$$

-  $O(n^2)$

- kvadratická

- sčítání natic  $n+n$

- složitost s počtem prvků roste lineárně

-  $5n^2 + 1000n^1 + 1000n^0$  ( $n^0 = 1$ )

- většina „klouzajících“ řešicích programů

- neponuzitelné na rozsáhlé úlohy

-  $O(n^3)$

- kubická

- násobení dvou natic  $n \cdot n$





