Obsah

20 Vývojový diagram, programovací jazyky

20.1 Vývojový diagram

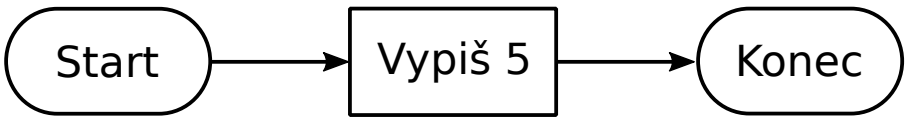
- grafické vyjádření procesu (algoritmus, postup práce)
- reprezentace kroků pomocí tvarů a šipek
- analýza procesu, návrh, dokumentace
- programy – MS Visio, Lucid chart, Inkscape
- preferovaný směr shora dolů, zleva doprava

20.1.1 Start a konec programu

- vyznačeno oválem
- šipky ukazují směr pokračování procesu

20.1.2 Dílčí krok

- obdélník
- popis dílčí akce, kroku



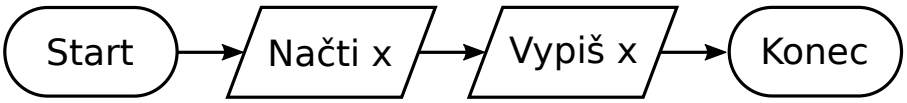
Obr. 20.1: Diagram jednoho kroku

20.1.3 Podprogram

- obdélník se svislými čárami
- více kroků označených jedním symbolem
- vyjádření funkce

20.1.4 Vstup/výstup

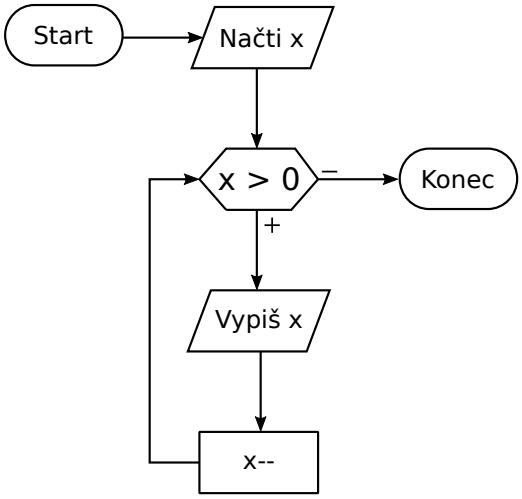
- rovnoběžník
- komunikace s uživatelem



Obr. 20.2: Diagram se vstupem a výstupem

20.1.5 Cykly

- šestiúhelník
- probíhají pouze za platné podmínky, jinak se pokračují dál



Obr. 20.3: Diagram cyklu

20.1.6 Podmínka

- kosočtverec
- pokud podmínka, splní se kroky

20.1.7 Spojovací značka

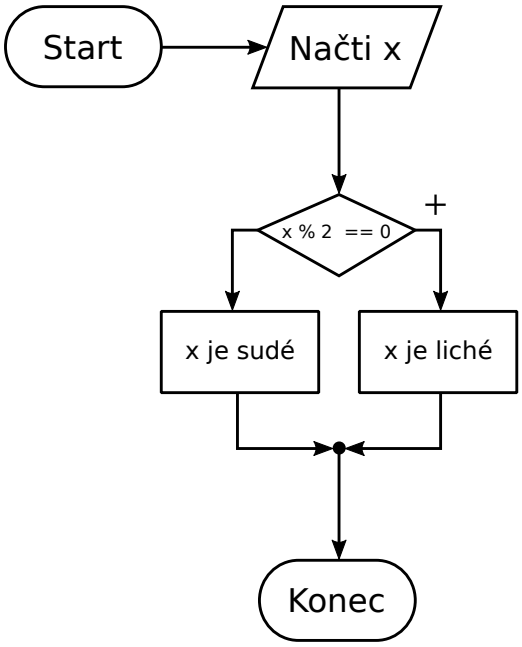
- kruh
- spojení více toků procesu do jednoho

20.2 Programovací jazyky

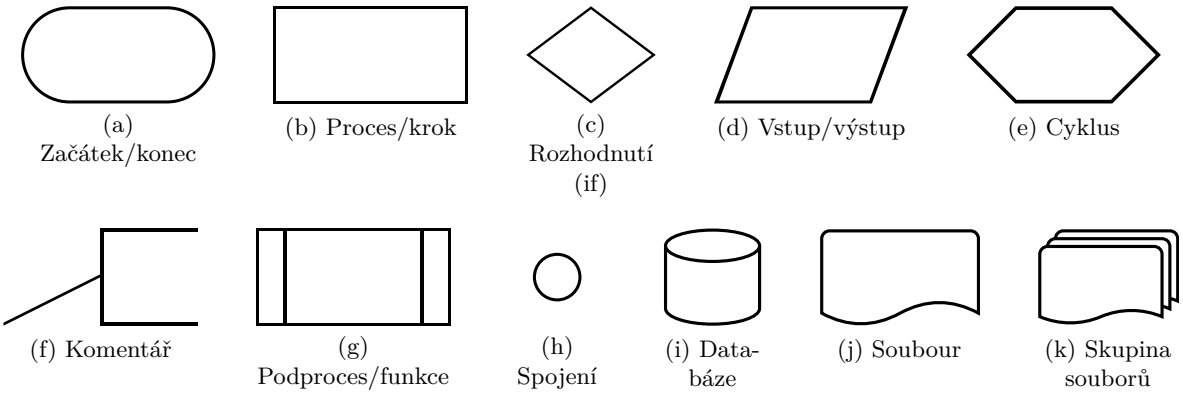
- způsob zapsání kódu a jeho následné použití v počítači
- rozdělení na vyšší a nižší

20.2.1 Nižší programovací jazyky

- strojový kód či mu velice blízko
- jednoduché na převod do binárního záznamu
- základ vyšších jazyků
- přímá kontrola nad registry, pamětí, pointery…
- strojový kód, assembly, občas za něj považované i C



Obr. 20.4: Diagram podmínky



20.2.2 Vyšší programovací jazyky

- použití přirozeného jazyku
- velké použití abstrakcí
- překládány do nižších jazyků
- velký počet předdefinovaných funkcí
- menší kontrola nad samotným hardwarem

20.2.3 Procedurální vs neprocedurální

- procedurální/imperativní
  - popis výpočtu pomocí příkazů, určení přesného postupu
  - Fortran, ALGOL, BASIC, C
- neprocedurální
  - specifikace cíle namísto postupu získání cíle
  - SQL, PROLOG, LIPS

20.2.4 Strukturované vs objektově orientované

- strukturované – jeden vstup a výstup, vytvoření algoritmu z řídíc struktur a funkcí
  - C, Pascal
- objektově orientované – využití objektů, jejich funkcí, dědičnosti…
  - Java, C#
- kombinované – strukturované programování s podporou objektů
  - Python, JavaScript, C++

20.2.5 Funkcionální vs logický

- funkcionální
  - vyhodnocování matematických funkcí
  - využití lambda funkcí
  - SQL, Mathematica, Haskell
- logický
  - použití matematické logiky k programování
  - PROLOG

20.2.6 Kompilované vs interpretované

- kompilované
  - jazyky přeloženy do binárního souboru, který je následně spuštěn
  - C, C++, Rust
- interpretované
  - interpreter interpretuje jazyk v reálném čase
  - flexibilnější, ale pomalejší
  - chyby jsou zachyceny až při spuštění kódu
  - Python, JavaScript, Java, R

20.2.7 Statické vs dynamické

- statické programovací jazyky
  - typ proměnných je znám v čase kompilace
  - typ proměnných zadán programátorem (Java, C, C++) nebo odvozen kompilátorem (Haskell, Rust)
- dynamicky psané programovací jazyky
  - typ dat specifikován pro konkrétní hodnotu, ne pro proměnnou
  - možno změnit typ proměnné v průběhu programu

- Python, JavaScript, PHP, Ruby