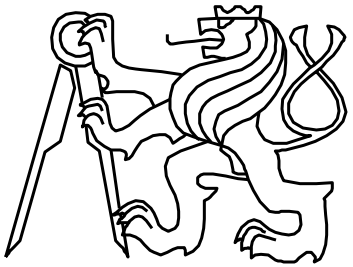
Příprava studijního programu Informatika je podporována projektem financovaným z Evropského sociálního fondu a rozpočtu hlavního města Prahy.
Praha & EU: Investujeme do vaší budoucnosti

Algoritmy



BI-PA1 Programování a algoritmizace 1, ZS 2012-2013
Katedra teoretické informatiky
© Miroslav Balík
Fakulta informačních technologií
České vysoké učení technické

Kdo je kdo v PA1?

- Ing. Miroslav Balík, Ph.D.
  - přednášející a garant předmětu
- Ing. Josef Vogel, CSc.
  - přednášející, proseminář
- Ing. Ladislav Vagner, Ph.D.
  - proseminář, Progtest
- Cvičící a asistenti
- Studenti

Organizace předmětu

- Výuka
  - 2 hod. přednášek
  - 2 hod. proseminář
  - 2 hod. cvičení týdně v počítačové učebně
- Studijní materiály a další informace budou postupně zveřejňovány na webu:

  http://edux.fit.cvut.cz/courses/BI-PA1/
- Doporučená literatura
  - Herout: Učebnice jazyka C, KOPP, Č. Budějovice
  - Virius: Jazyky C a C++, GRADA, Praha

Osnova předmětu

Přednášky

1. Algoritmus
2. Proměnné, vstup, výstup
3. Výrazy
4. Větvení, cykly
5. Funkce
6. Rekurze
7. Pole, řetězce
8. Ukazatele
9. Soubory
10. Složitost algoritmů
11. Struktury
12. Spojové struktury
13. Modulární programování

Cvičení

1. Učebna, číselné soustavy
2. Proměnné, vstup, výstup
3. Reálné typy, výrazy
4. Podmíněné příkazy
5. Cykly, posloupnosti
6. Funkce
7. Rekurze
8. Pole a řetězce
9. Ukazatele
10. Soubory
11. Složitost algoritmů
12. Struktury
13. Spojové struktury

Cíl předmětu

- Náš cíl
  - naučit Vás sestavovat algoritmy pro řešení jednoduchých problémů a zapisovat je v jazyku C
- Váš cíl
  - získat zápočet a zkoušku za A ☺

Hodnocení předmětu

Zdroje bodů pro hodnocení	Body
domácí úkoly	8*5 = 40b
soutěžní úloha	až 15 bodů
znalostní testy	4*5 = 20 b
test u počítače v semestru	20 b
písemný zkouškový test	20 b
Ústní zkouška	-10 +20 b

zápočet >= 40b

Klasifikace (*na základě bodového hodnocení*)				
klasifikace	*počet bodů*	číselně	slovně	
A	> 90	1	výborně	
B	80 - 89	1,5	velmi dobře	
C	70 - 79	2	dobře	
D	60 – 69	2,5	uspokojivě	
E	50 - 59	3	dostatečně	
F	< 50	4	nedostatečně	

Detaily na http://edux.fit.cvut.cz/courses/BI-PA1/classification/start

Algoritmus a jeho vlastnosti

- Definice
  - postup při řešení určité třídy úloh, který je tvořen seznamem jednoznačně definovaných příkazů a zaručuje, že pro každou přípustnou kombinaci vstupních dat se po provedení konečného počtu kroků dospěje k požadovaným výsledkům
- Vlastnosti
  - *hromadnost*
    měnitelná vstupní data
  - *determinovanost*
    každý krok je jednoznačně definován
  - *konečnost a resultativnost*
    pro přípustná vstupní data se po provedení konečného počtu kroků dojde k požadovaným výsledkům

Algoritmus

- Algoritmus – syntetický model postupu řešení obecných úloh
- Prostředky pro zápis algoritmu
  - přirozený jazyk, vývojové diagramy, struktogramy, pseudojazyk, programovací jazyk - C

Problém číslo 1, myš na Zemi a ve vesmíru

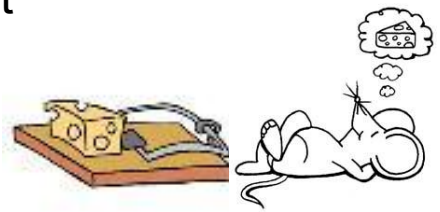
- Problém: Předpokládejme, že naše Země je omotána na rovníku tlustou mašlí. Otázka zní, pokud tuto mašli prodloužíme o jeden metr, vznikne dostatečná mezera, aby se pod ní protáhla myš? Jak se změní odpověď pokud nebudeme uvažovat Zemi, ale Měsíc?

Země(Apollo 17):

Myš:

Řešení 1. velmi naivní algoritmus

1. Chytnu myš ,raději živou, bude muset prolézat
2. seženu velmi dlouhou mašli,
3. natáhnu ji po rovníku,
4. přidám k ní jeden metr,
5. zkontroluji, zda je všude povolená stejně,
6. vhodně motivovanou myš požádám, aby zkusila prolézt,
7. zapíši výsledek.



Modifikace pro Měsíc:

1. seženu grant,
2. koupím vesmírnou loď, skafandr pro sebe a myš,
3. pokračuji jako na Zemi

Požadavky na algoritmus, aneb je tento naivní algoritmus algoritmem?

- **Konečnost** - Každý algoritmus musí skončit v konečném počtu kroků. Tento počet kroků může být libovolně velký (podle rozsahu a hodnot vstupních údajů), ale pro každý jednotlivý vstup musí být konečný.
  - **Ano, skončí.**
- **Determinovanost** - Každý krok algoritmu musí být jednoznačně a přesně definován; v každé situaci musí být naprosto zřejmé, co a jak se má provést, jak má provádění algoritmu pokračovat.
  - **S rezervou, ale ano.**
- **Vstup** - Algoritmus obvykle pracuje s nějakými vstupy, veličinami, které jsou mu předány před započetím jeho provádění, nebo v průběhu jeho činnosti. Vstupy mají definované množiny hodnot, jichž mohou nabývat.
  - **Vstupem jsou myš a planeta (mají velikost), 1metr.**
- **Výstup** - Algoritmus má alespoň jeden výstup, veličinu, která je v požadovaném vztahu k zadaným vstupům, a tím tvoří odpověď na problém, který algoritmus řeší. (Algoritmus vede od zpracování hodnot k výstupu - Resultativnost)
  - **Ano, projde x neprojde.**

Požadavky na algoritmus II, aneb je tento naivní algoritmus algoritmem?

- **Efektivita -** Obecně požadujeme, aby algoritmus byl efektivní, v tom smyslu, že požadujeme, aby každá operace požadovaná algoritmem, byla dostatečně jednoduchá na to, aby mohla být alespoň v principu provedena v konečném čase pouze s použitím tužky a papíru. (tj. byla elementární)
  - **Tady je největší slabina navrženého algoritmu**
- **Obecnost (Hromadnost)** Algoritmus neřeší jeden konkrétní problém (např. „jak spočítat 9×7“), ale obecnou třídu obdobných problémů (např. „jak spočítat součin dvou celých čísel“).
  - **Ano.**
- V praxi jsou proto předmětem zájmu hlavně takové algoritmy, které jsou v nějakém smyslu kvalitní. Takové algoritmy splňují různá kritéria, měřená např. počtem kroků potřebných pro běh algoritmu, nebo jednoduchost či elegance algoritmu. Problematikou efektivity algoritmů, tzn. metodami, jak z několika známých algoritmů řešících konkrétní problém vybrat ten nejlepší, se zabývají odvětví informatiky nazývané algoritmická analýza a teorie složitosti.Výpočetní proces je posloupnost akcí nad daty uloženými v paměti počítače

Řešení 2. - výpočetní algoritmus

1. Zjistím velikost poloměru příslušné planety, označím jej **r,**
2. velikost myši označím **mys**, budu ji uvádět v metrech,
3. vypočtu rovníkový obvod planety podle vztahu **obvod = 2*π*r**,
4. vypočtu poloměr **r2** povolené mašle podle vztahu **r2 = (obvod + 1)/(2* π)**,
5. jestliže **mys + r < r2** odpověď je **Ano,** jinak **Ne**

Modifikace pro Měsíc:

1. Změním poloměr a postup opakuji.

- Konečnost, determinovanost, vstup, výstup, efektivita, obecnost - Hromadnost
- Determinovanost si vynucuje jednoznačný zápis, např. v C. K tomu potřebujeme proměnné, výrazy, vstup a výstup.

Algoritmy - NSD

- Úloha: najděte největšího společného dělitele čísel 6 a 15 (co je největší společný dělitel dvou přirozených čísel?)

Řešení:

- Popišme postup tak, aby byl použitelný pro dvě libovolná přirozená čísla, nejen pro 6 a 15:
    - označme zadaná čísla *x* a *y* a menší z nich *d*
    - není-li *d* společným dělitelem *x* a *y*, pak zmenšíme *d* o 1, test opakujeme a skončíme, až *d* bude společným dělitelem *x* a *y*

Poznámka:

- Význam symbolů *x*, *y* a *d* použitých v algoritmu:
    - jsou to **proměnné** (paměťová místa), ve kterých je uložena nějaká hodnota, která se může v průběhu výpočtu měnit

NSD 2

- Úloha: najděte největšího společného dělitele
- *Přesnější popis:*

  Vstup: přirozená čísla *x* a *y*

  Výstup: *nsd*(*x*,*y*)

  Postup:

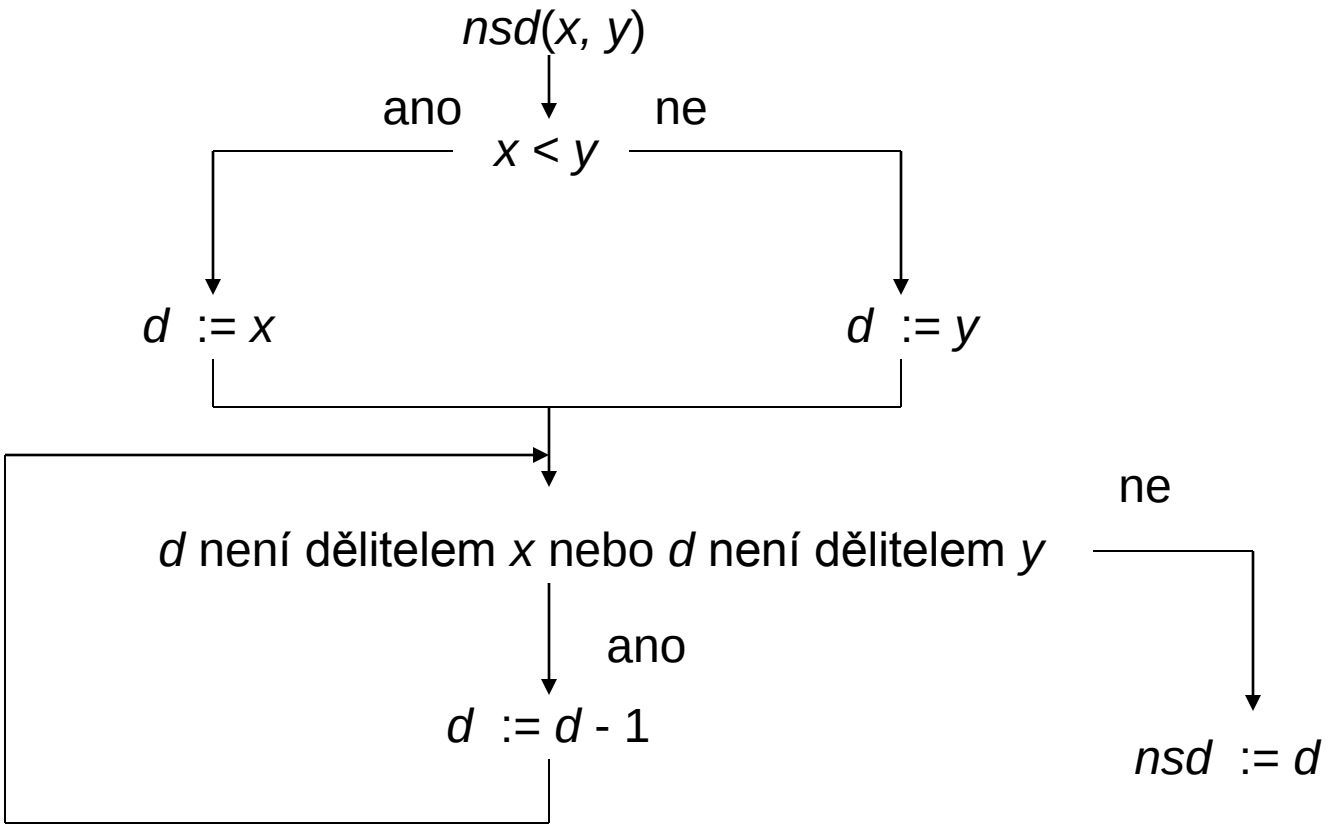
  1. Je-li *x*<*y*, pak *d* má hodnotu *x*, jinak *d* má hodnotu *y*
  2. Opakuj krok 3, pokud *d* není dělitelem *x* nebo *d* není dělitelem *y*
  3. Zmenši *d* o 1
  4. Výsledkem je hodnota *d*
- Sestavili jsme algoritmus pro výpočet největšího společného dělitele dvou přirozených čísel

NSD - příklad

Úloha: najděte největšího společného dělitele čísel 6 a 15

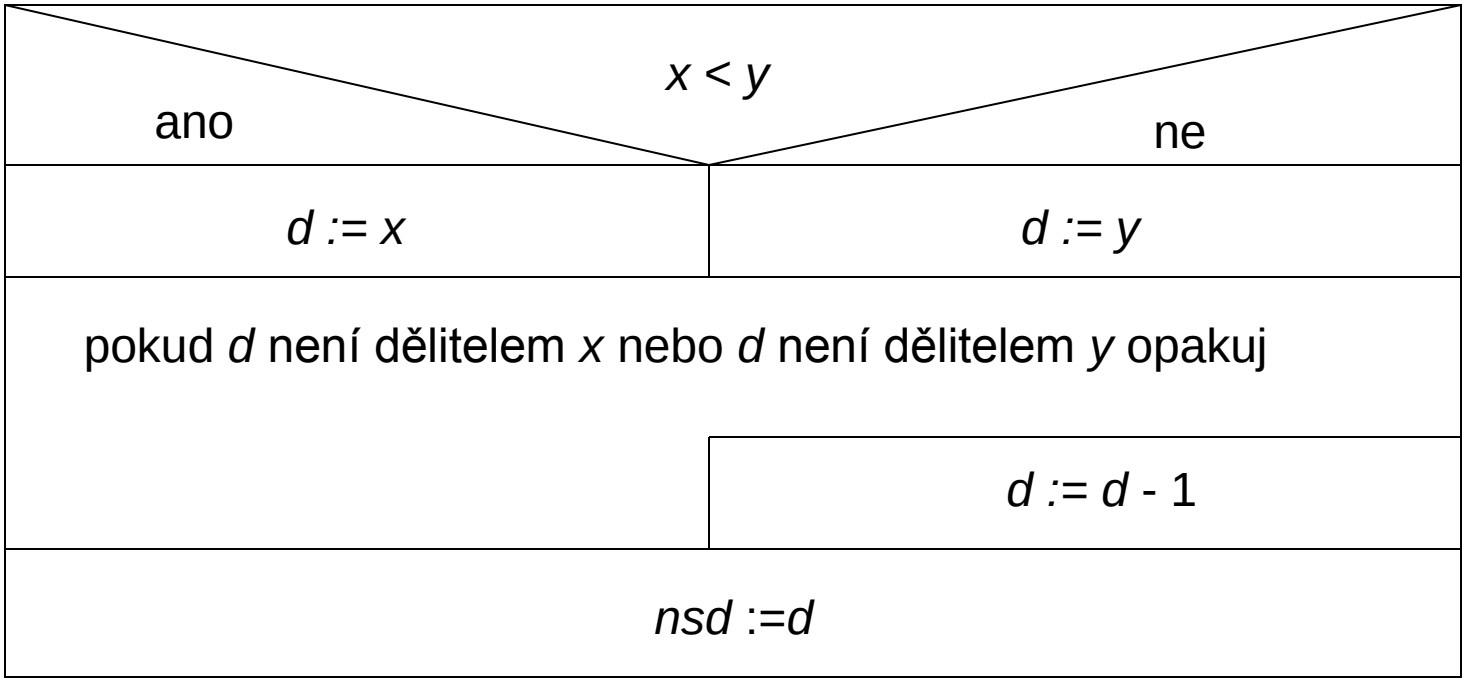
krok	x	y	d	poznámka
	6	15	?	zadání vstupních dat
1	6	15	6	
2	6	**15**	**6**	**d není dělitelem y**, proveď krok 3
3	6	15	5	
2	**6**	15	**5**	**d není dělitelem x,** proveď krok 3
3	6	15	4	
2	**6**	**15**	**4**	**d není dělitelem x ani y,** proveď krok 3
3	6	15	3	
2	**6**	**15**	**3**	**d je dělitelem x i y,** proveď krok 4
4	6	15	3	výsledkem je číslo **3**

NSD – vývojový diagram



NSD - Struktogram

nsd(*x*, *y*)



NSD - pseudojazyk

Zápis algoritmu v pseudojazyku

```
nsd(x,y):
    if x<y then d:=x else d:=y;
    while d „není dělitelem“ x or d „není dělitelem“ y do
      d:=d-1;
    nsd:=d;
```

Zápis algoritmu v programovacím jazyku C

```
int nsd(int x, int y)
{
  int d;
  if (x<y) d=x; else d=y;
  while (x%d!=0 || y%d!=0) d--;
  return d;
}
```

Programy a programovací jazyky

- Program je předpis pro provedení určitých akcí počítačem zapsaný v programovacím jazyku (je to vždy zápis algoritmu?)
- Programovací jazyky
    - strojově orientované
        - strojový jazyk = jazyk fyzického procesoru
        - assembler (jazyk symbolických adres)
    - vyšší jazyky
        - **imperativní** (příkazové, procedurální)
        - neimperativní (např. funkcionální, deklarativní,...)
- Hlavní rysy imperativních jazyků (např. **C**, C++, Java**,** Pascal, Basic, ...)
  - zpracovávané údaje mají formu datových objektů různých typů, které jsou v programu reprezentovány pomocí proměnných resp. konstant
  - program obsahuje deklarace a příkazy
  - deklarace definují význam jmen (identifikátorů)
  - příkazy předepisují akce s datovými objekty nebo způsob řízení výpočtu

Vlastnosti programovacích jazyků

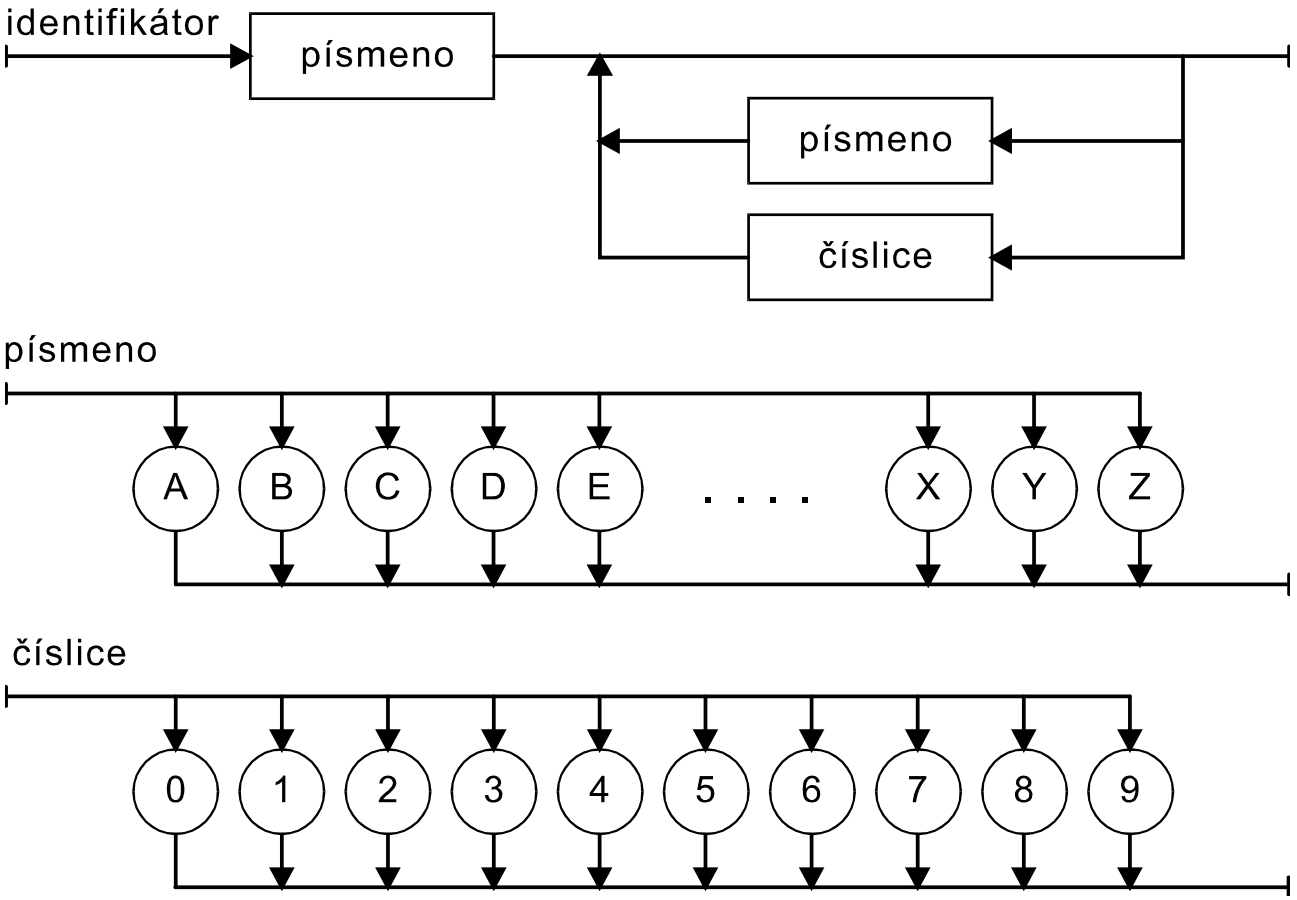
- Syntaxe
  - souhrn pravidel udávajících přípustné tvary dílčích konstrukcí a celého programu
- Sémantika
  - udává význam jednotlivých konstrukcí

- Prostředky pro popis syntaxe
  - syntaktické diagramy
  - různé formy Backus-Naurovy formy

- Sémantika je obvykle popsána slovně

Syntaktické diagramy

- Příklad: identifikátor je posloupnost písmen a číslic začínající písmenem



Rozšířená BNF

- Rozšířená Backus-Naurova forma – EBNF
- Příklad: identifikátor

  identifikátor = písmeno {písmeno | číslice}

  písmeno = 'A' | 'B' | 'C' | 'D' | ... | 'X' | 'Y' | 'Z'

  číslice = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '9'
- Neterminály:

  identifikátor, písmeno, číslice
- Terminály:

  ‘A’, ‘B’, ...
- Význam metasymbolů:

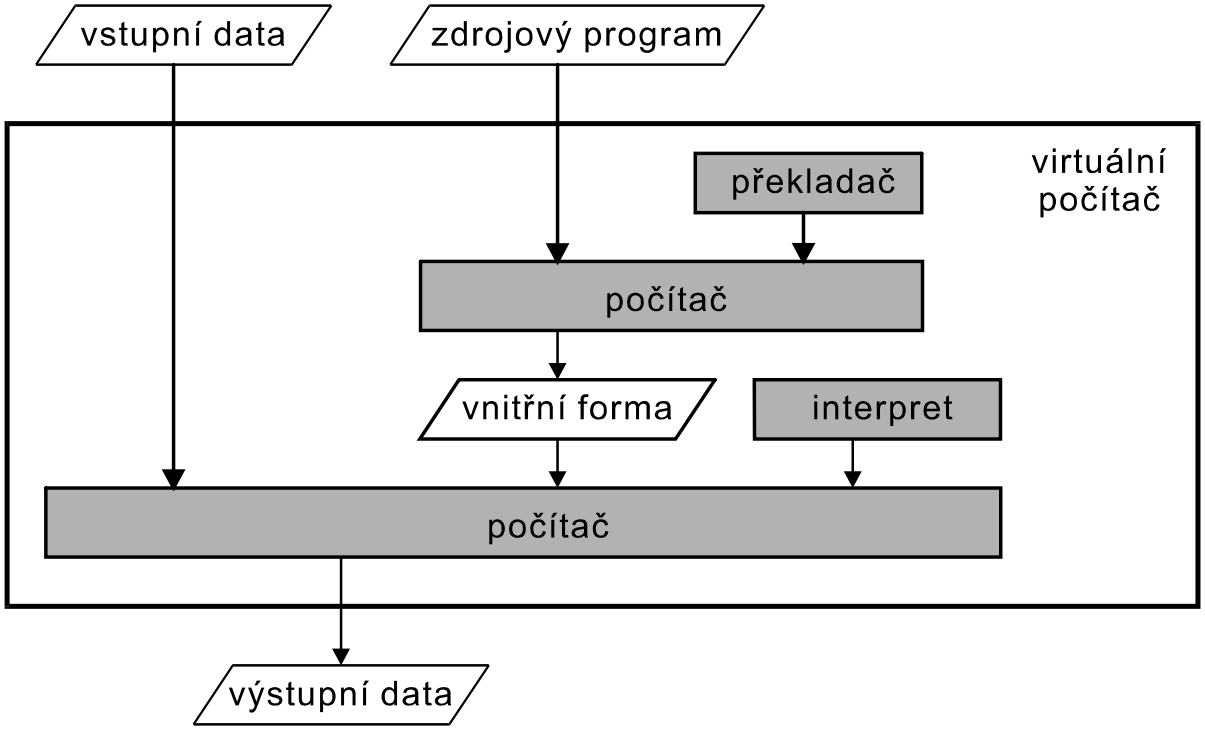
  {*x*} žádný nebo několik výskytů *x*

  x | *y* *x* nebo *y*

  [*x*] žádný nebo jeden výskyt *x*

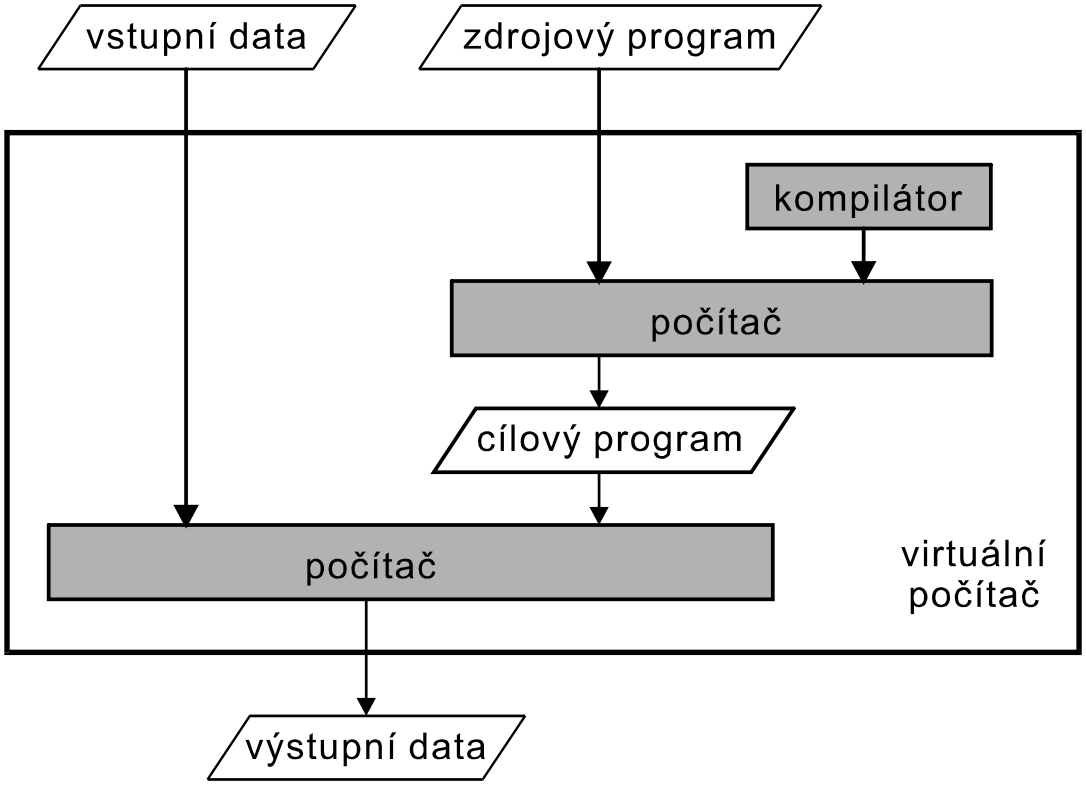
Implementace programovacích jazyků

- Dvě základní metody:
  - interpretační
  - kompilační
- Interpretační metoda:



Implementace programovacích jazyků

- Kompilační metoda:



Úvod do jazyka C

- Pro prezentaci, návrh a ověřování algoritmů použijeme jazyk C
- Z historie jazyka C

  1972 D. Ritchie Bell Laboratories PDP11

  1978 Kernighan, Ritchie: The C Programming Language (K&R C)

  Systémový jazyk operačního systému Unix

  1982 ANSI pracovní skupina X3J11

  ANSI X3.159-1989 (ANSI C, C89)

  převzatá ISO/IEC 9899:1990

  1999 nová norma ISO 9899:1999 (C99)

- Budeme používat ANSI C
- Objektově orientovaným rozšířením C je jazyk C++, se kterým se seznámíme v předmětu PA2

Úvod do jazyka C

- Přednosti jazyka C
  - jde o vyšší, obecně použitelný programovací jazyk
  - neobsahuje „černé schránky“, veškeré konstrukce jsou srozumitelné a snadno ilustrovatelné
  - umožňuje operace blízké strojovému kódu (operace s adresami)
  - překladače optimalizují, výsledný kód je velice efektivní
  - vytvořené programy jsou dobře přenositelné (portabilní), pokud se nepoužívají počítačově závislé triky
  - syntaxi výrazů a příkazů převzala řada dalších jazyků
- Hlavní nevýhoda překladačů jazyka C pro začátečníka
  - nízká zabezpečenost (při běhu programu se neprovádějí žádné kontroly)

První program v jazyku C

- Příklad programu, který vypíše daný text na obrazovku:

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  printf("\nNazdar, toto je prvni program\n");
  system("PAUSE");
  return 0;
}
```

vložení deklarací knihovních funkcí

definice hlavní funkce

oddělovač řádků

příkaz pro výpis řetězce

- Po překladu a spuštění se na obrazovku vypíše

```
Nazdar, toto je prvni program
Pokračujte stisknutím libovolné klávesy ...
```

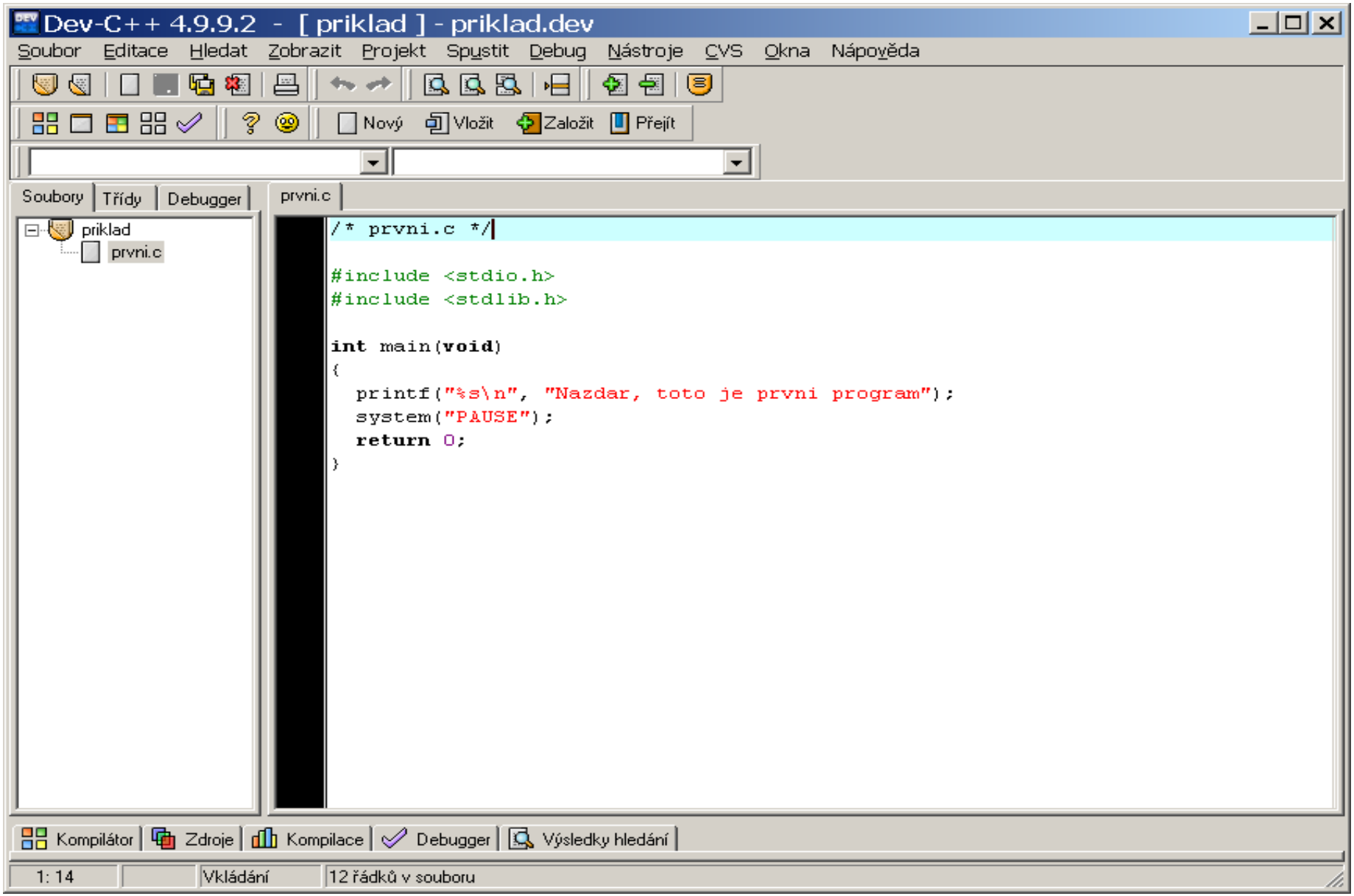
- Nejjednodušší zdrojový program
  - Jeden soubor obsahující hlavní funkce **`main`**

Překladače a vývojová prostředí

- Co budeme používat pro tvorbu programů
  - NetBeans + GCC (Cygwin)
  - Dev-Cpp volně šiřitelný překladač (freeware)
  - PROC
    - interpret jazyka C s omezenou knihovnou napsaný v jazyce Java
    - provádí kontroly při běhu programu

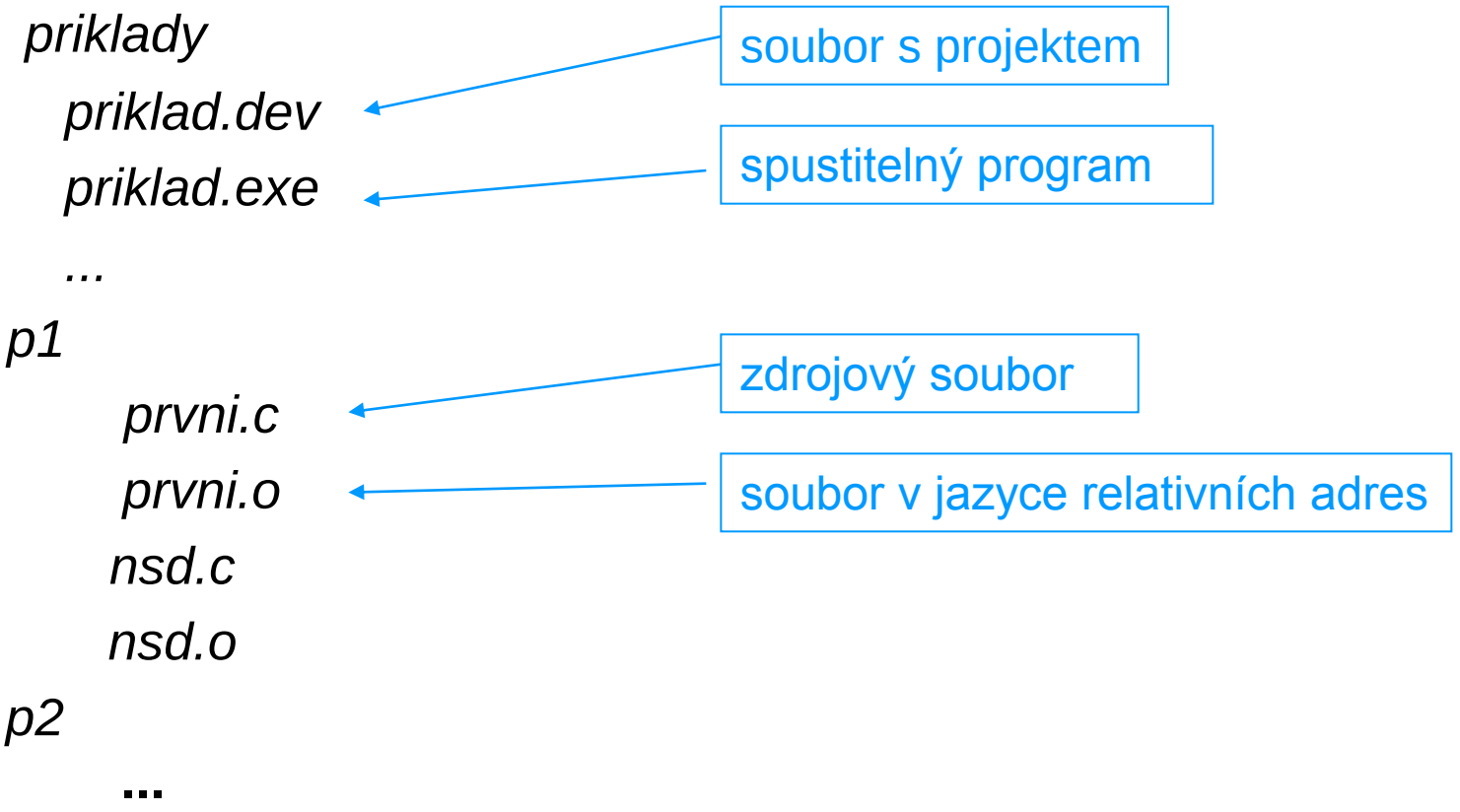
- Dále je možno použít
  - C++ Builder (Borland Inc.)
  - Visual C++ (Microsoft)

První program v Dev-Cpp



Příklady k přednáškám

- První program a všechny další, které budou prezentovány na přednáškách, jsou v adresáři *prednasky/priklady*
- Struktura adresáře:



Program pro výpočet NSD

```
#include <stdlib.h>
#include <stdio.h>
/* nsd.c */

int nsd(int x, int y)
{
  int d;
  if (x<y) d=x; else d=y;
  while (x%d!=0 || y%d!=0) d--;
  return d;
}
```

```
int main(void)
{
    int a, b, v;
    printf("\nNejvetsi spolecny delitel\n");
    printf("Zadej dve prirozena cisla\n");
    scanf("%d %d", &a, &b);
    if (a<=0 || b<=0)
       printf("cisla musi byt vetsi nez nula\n");
    else
       printf("nsd(%d,%d)=%d\n", a, b, nsd(a,b));
    system("PAUSE");
    return 0;
}
```