Programování I

Martin Pergel, perm@kam.mff.cuni.cz

2. října 2012

Informace o přednášce, cvičeních a Praktiku z programování

- Kurz je zakončen zápočtem, zkouška bude v létě.
- Podmínky zápočtu:
  - Zápočtový test (praktický),
  - zápočtový program (domácí práce),
  - další dle požadavků cvičících.
- Praktikum z programování:
  - Volitelný předmět,
  - v rozvrhu letos maskováno jako druhé cvičení (zpravidla od 19:00 + Po 12:20 K11),
  - začne v týdnu od 15. října zjištěním zájmu o jednotlivá cvičení,
  - do té doby cvičící vytipují kandidáty.

Cíle předmětu:

- Ovládání prostředí Borland Pascal,
- jazyk Pascal,
- algoritmy,
- a teorie s nimi související.

Jednotlivé položky budou probírány paralelně!

Proč Pascal:

- Mýtus: Pascal je zastaralý!
- Skutečnost: Pascal je osvědčený.
- Java, C#, Jazyk C... – pěkné ale komplikované.
- Pascal: Nevýhoda: Věkovitost Výhoda: Jednoduchost.
- My: Borland Pascal (Free Pascal, GNU Pascal, Delphi).

Organizační záležitosti

- Stroj CodEx (alias Code Examiner a účet na něm)
- Účty v příslušných počítačových učebnách (Karlín, Malá Strana)
- Existují dvě paralelní přednášky, které jsou ekvivalentní, nikoliv totožné.
- Pozor, programování je dovednost náročná na čas!
- Literatura:
  Pavel Töpfer: Algoritmy a programovací techniky,
  Pavel Satrapa: Pascal pro zelenáče,
  Niklaus Wirth: Algorithms + Data Structures = Programs (slovenský překlad "Algoritmy a Štruktury údajov") [poměrně věkovitější, zajímavější jsou v ní algoritmy než jazyk]
- Dotazy? [pokud ano, ptejte se ihned]

Algoritmy:

Definition

Algoritmy jsou přesně definované a záměrně vytvořené postupy.

- Algoritmus je postup, jak řešit určitý problém.
- Realizací algoritmu dojdeme od zadaných (vstupních) dat k požadovanému výsledku.
- Sestává z "kroků" zvaných příkazy (příklad - alg. sčítání čísel).
- Správný algoritmus musí být:
  - konečný (pro každý vstup doběhne v konečném čase)
  - a parciálně správný (pokud doběhne, odpoví správně).

Způsoby zápisu algoritmu

Karel:

```
krok
krok
krok
vlevobok
krok
```

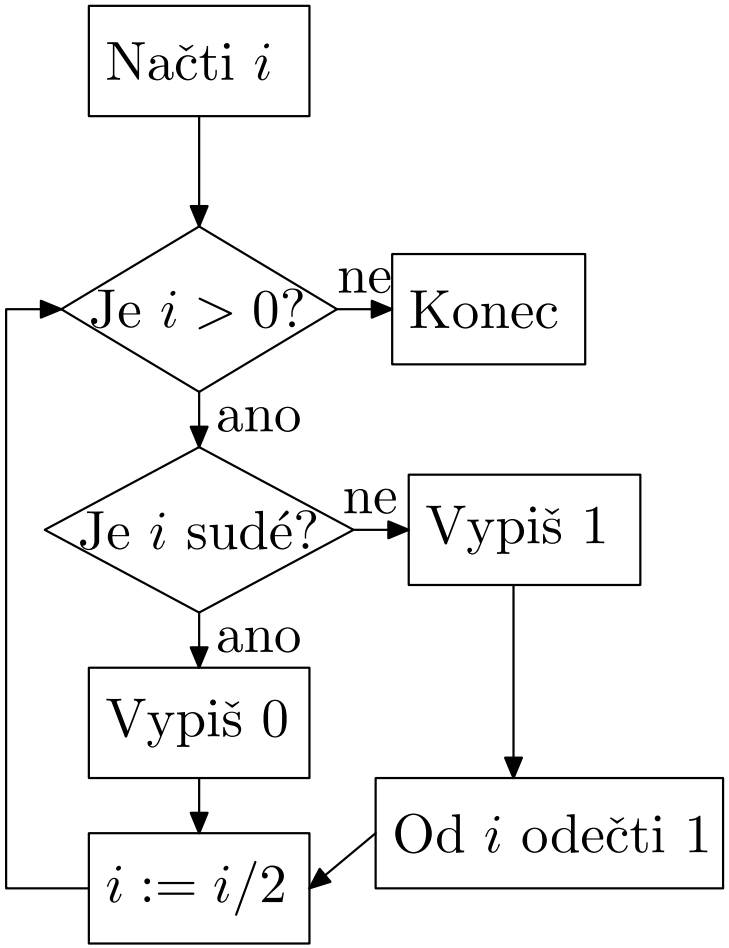
Jazyk C:

```
while(i)
{ if(i%2) printf(1);
  else printf(0);
  i/=2;
}
```

Text:

Načti hodnotu i.
Dokud je $i > 0$:
    Je-li je i liché vypiš "1"
    jinak vypiš "0"
    Vyděl i dvojkou.

Vývojové diagramy:



Největší společný dělitel

Možnosti hledání:

- Najít prvočíselné rozklady a porovnat,
- Eukleidův algoritmus.

Pozorování: Jsou-li $a \geq b$ přirozená čísla dělitelná (rovněž přirozeným) číslem k, pak i $a - b$ je dělitelné k.

Eukleidův algoritmus: 1. varianta (s odčítáním)

```
Načti a, b.                          read(a); read(b);
1: Pokud b > a, prohoď hodnoty a a b.
Pokud b je nula, vypiš a             if b=0 then write(a);
a ukonči algoritmus.
Od a odečti b.                          a:=a-b;
GOTO 1:
```

Eukleidův algoritmus: 2. varianta (se zbytkem po dělení)

Načti a, b.
1: Pokud $b > a$, prohoď a a b.
Pokud b je nula, vypiš a a konec.
Do a přiřaď zbytek po dělení hodnoty a hodnotou b.
GOTO 1:

Magické čtverce lichého řádu

Snadný algoritmus, leč důkaz správnosti je komplikovaný:

6	1	8
7	5	3
2	9	4

15	8	1	24	17
16	14	7	5	23
22	20	13	6	4
3	21	19	12	10
9	2	25	18	11

1. Začneme uprostřed horního řádku.
2. Postupujeme šikmo doleva nahoru a vyplňujeme čísla v rostoucím pořadí.
3. Najedeme-li na vyplněné políčko, vrátíme se zpět a postoupíme o políčko dolů.

Stabilní párování

- Instance: N pánů a N dam. Každá postava má seznam "přijatelnosti" (všech) příslušníků opačného pohlaví. Problém: Vytvořte N koedukovaných dvojic, aby vzniklé párování bylo stabilní.
- Párování je stabilní, pokud neexistuje dvojice $I\iota$ tak, že $I\kappa$ a $K\iota$, ale kdybychom je "přepojili" na $I\iota$ a $K\kappa$, jak I, tak ι by si polepšili.
- Ne zcela snadný algoritmus, zkuste si do příště rozmyslet, příště o něm pohovoří kolega Kryl.

Rozklad na prvočinitele

- Nápady
- Naivní algoritmus: Hledej postupně prvočísla až do n a zkoušej jimi dělit.
- Pokus o méně naivní algoritmus: `Hledej prvočísla do` $\sqrt{n_i}$ `(kde` n_i `je hodnota, kterou zbývá faktorizovat).` nebude fungovat. Proč?
- Ještě méně naivní algoritmus: Zkoušej všechna čísla až do $n/2$. Proč toto funguje?

K Eratosthenovu sítu

Jak nagenerovat všechna prvočísla menší než n?

- Naivní algoritmus (generuj a testuj),
- Méně naivní algoritmus generuj a zkoušej dělit jen již nagenerovanými prvočísly.
- Eratosthenes: Nageneruj čísla $2 \ldots n$. Pro i od 2 do $\sqrt{n}$ pokud je i prvočíslo, proškrtej všechny jeho násobky.