Algoritmus a vlastnosti algoritmov

Mgr. Ing. Roman Horváth, PhD.
Katedra matematiky a informatiky
Pedagogická fakulta
Trnavská univerzita v Trnave
roman.horvath@truni.sk

Algoritmus

- konečná postupnosť krokov (**postup**),
- dostatočne elementárnych,
- ktorých vykonávanie nevyžaduje „vhľad“, inteligenciu či intuíciu
- a ktorých vykonávanie dokáže ***vyriešiť*** predložený ***problém*** v **konečnom čase**.

- postupnosť elementárnych krokov vedúca k riešeniu v konečnom čase…

Algoritmus vstávania do školy

- zastav budík,
- vstaň z postele,
- umy sa,
- vyzleč si pyžamo,
- obleč si šaty,
- naraňajkuj sa,
- obuj sa,
- zober tašku,
- vyjdi pred dvere,
- zamkni byt.

(neriešime úplne elementárne úkony typu: otvor dvere, daj zubnú pastu na kefku atď.)

Vlastnosti algoritmov

- **Elementárnosť** – postup je zložený z elementárnych krokov (príkazov).

- **Determinovanosť** – jednoznačnosť – postup má byť zostavený tak, aby bolo v každom okamihu jeho vykonávania zrejmé, čo má nasledovať ďalej a či sa postup už skončil.

- **Rezultatívnosť** – pri správnom algoritme musí byť zaručené, že sa dopracujeme k správnemu výsledku, a že postup dáva rovnaké výsledky pri rovnakých vstupných údajoch.

Vlastnosti algoritmov

- **Konečnosť** – postup skončí v konečnom čase, po vykonaní konečného počtu činností.
- **Hromadnosť** – postup je použiteľný pre celú dostupnú množinu prípustných vstupných hodnôt – pre ľubovoľné vstupné údaje.
- **Efektívnosť** – postup skončí v čo najkratšom čase, s využitím čo najmenšieho počtu činností (zdrojov, prostriedkov…).

Vstup a výstup

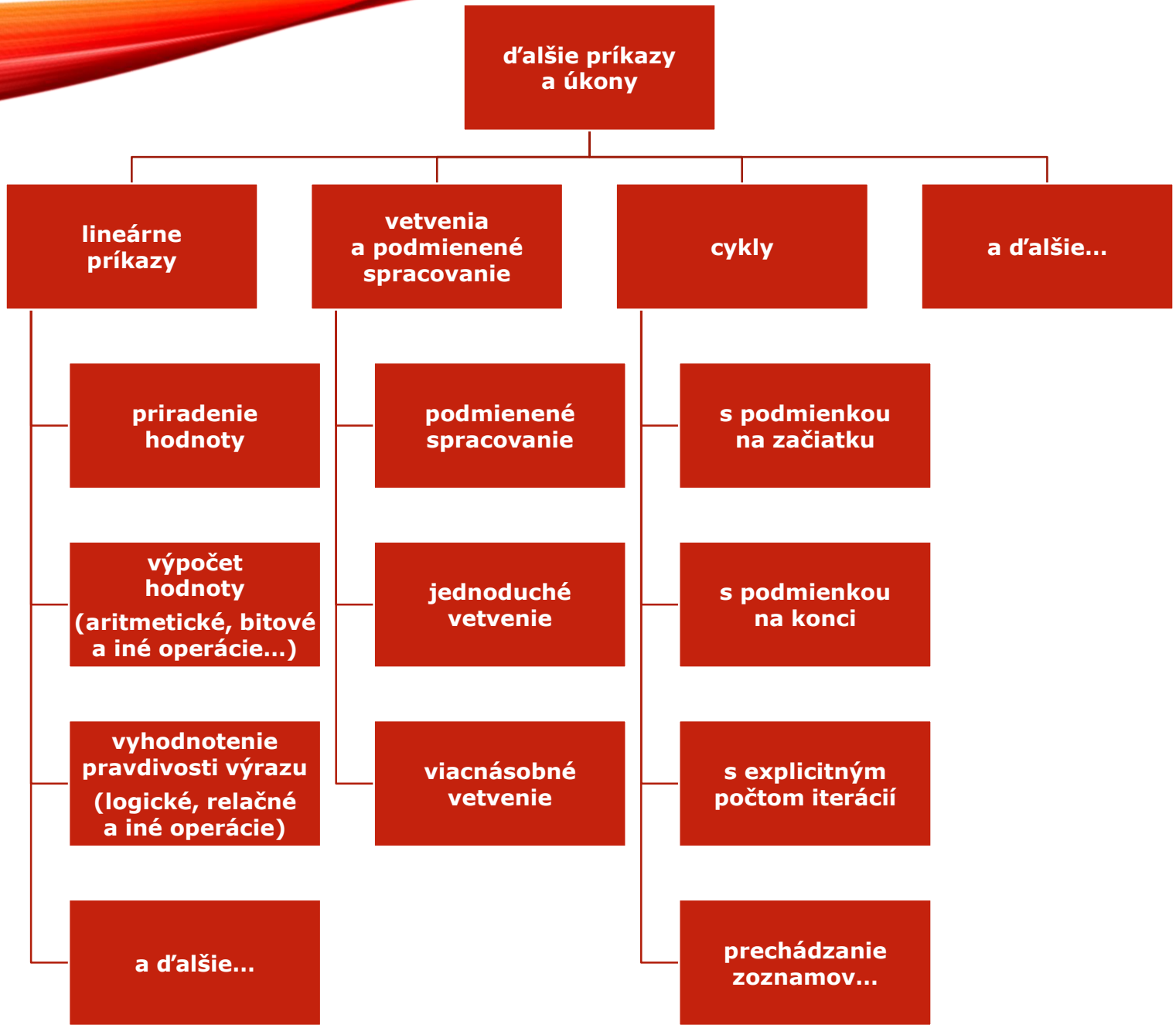
- vstup, či výstup údajov môže byť vykonávaný rôznymi spôsobmi:
  - klávesnica / monitor,
  - súbor,
  - databáza,
  - iný proces...

Príklady príkazov vstupu

- Pascal:
  - read(…);
  - readln(…);
- Java:
  - InputStreamReader vstup = **new** InputStreamReader(System.in);
  - BufferedReader čítač = **new** BufferedReader(vstup);
  - try { riadok = čítač.readLine(); } catch …
- GRobot:
  - Long l = Svet.čítajCeléČíslo("Počet:");
  - Double d = Svet.čítajReálneČíslo("Číslo:");

Príklady príkazov výstupu

- Pascal:
  - write(…);
  - writeln(…);
- Java:
  - System.out.print("…");
  - System.out.println("…");
- GRobot:
  - Svet.vypíš("Nezadali ste: ");
  - Svet.vypíšRiadok("Hra na ozvenu…", riadok);


ďalšie príkazy a úkony
lineárne príkazy
vetvenia a podmienené spracovanie
cykly
a ďalšie...
priradenie hodnoty
výpočet hodnoty (aritmetické, bitové a iné operácie...)
vyhodnotenie pravdivosti výrazu (logické, relačné a iné operácie)
a ďalšie...
podmienené spracovanie
jednoduché vetvenie
viacnásobné vetvenie
s podmienkou na začiatku
s podmienkou na konci
s explicitným počtom iterácií
prechádzanie zoznamov...

Spôsoby vyjadrovania algoritmov

Mgr. Ing. Roman Horváth, PhD.
Katedra matematiky a informatiky
Pedagogická fakulta
Trnavská univerzita v Trnave
roman.horvath@truni.sk

Spôsoby vyjadro-vania algoritmov

- textové
  - slovne,
  - zdrojovým kódom…
- grafické
  - vývojové diagramy,
  - Nassi-Shneidermanove diagramy = = štruktogram (štruktúrogram)
  - …

Podmienené spracovanie a jednoduché vetvenie

- ak *«podmienka»* tak
  «príkaz(y)»
  …

- ak *«podmienka»* tak
  «príkaz(y)»
  …
  inak
  «príkaz(y)»
  …

Podmienené spracovanie a jednoduché vetvenie

- if *«podmienka»* then
  begin
    «príkazy»;
    …
  end;

- if *«podmienka»* then
  begin
    «príkazy»;
    …
  end
  else
  begin
    «príkazy»;
    …
  end;

Podmienené spracovanie a jednoduché vetvenie

- if («podmienka»)
  {
    «príkazy»;
    …
  }

- if («podmienka»)
  {
    «príkazy»;
    …
  }
  else
  {
    «príkazy»;
    …
  }

Podmienené spracovanie a jednoduché vetvenie

- if *a* <> *0* then
  c := b / a;

- if *D = 0* then
```
begin
    x1 := (–b + sqrt(D)) / (2 * a);
    x2 := (–b – sqrt(D)) / (2 * a);
end
else
begin
    writeln('Nehľadáme riešenie ',
        'v množine komplexných čísel');
end;
```

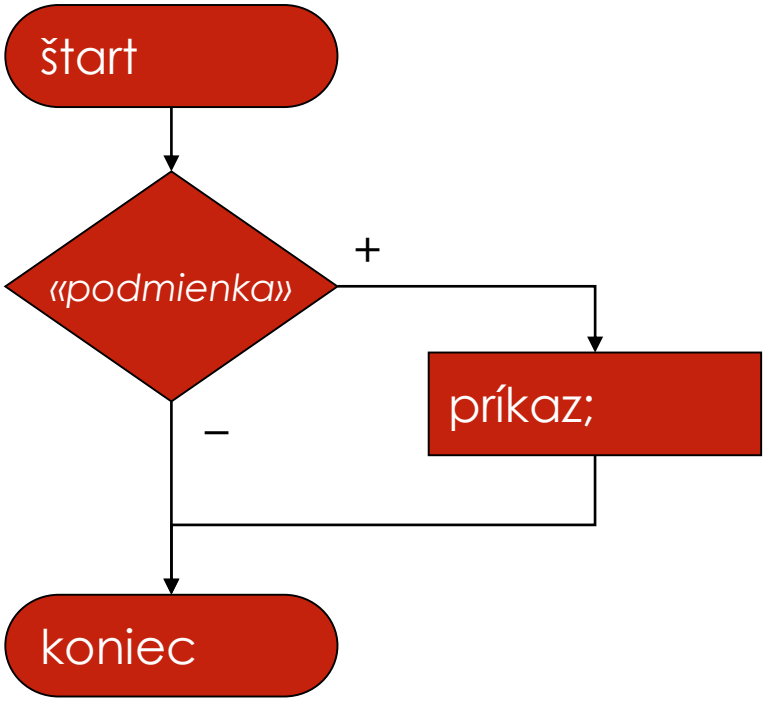
Podmienené spracovanie a jednoduché vetvenie

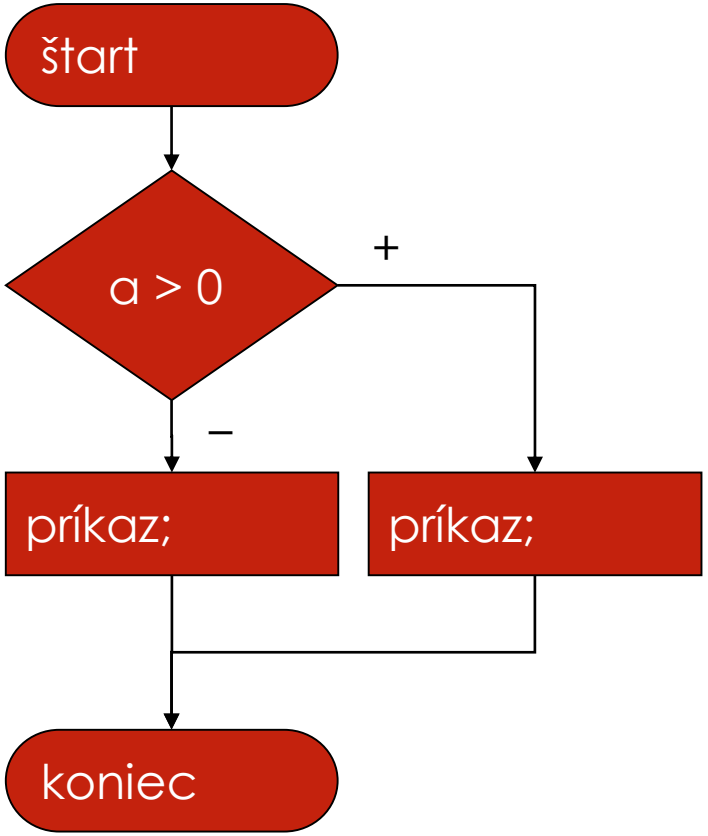
- if (*a != 0*)
  c = b / a;

- if (*D == 0*)

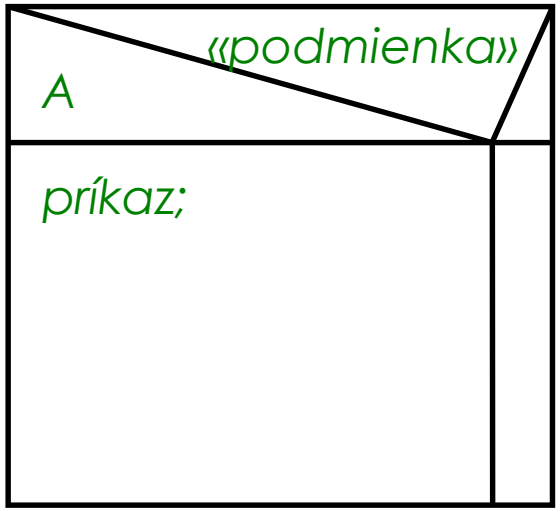
```
{
    x1 = (–b + Math.sqrt(D)) / (2 * a);
    x2 = (–b – Math.sqrt(D)) / (2 * a);
}
else
{
    vypíšRiadok("Nehľadáme riešenie " +
        "v množine komplexných čísel");
}
```

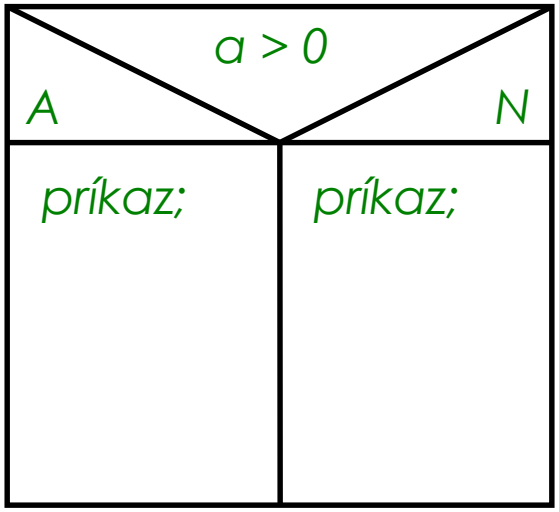
Podmienené spracovanie a jednoduché vetvenie





Podmienené spracovanie a jednoduché vetvenie





Príklady použitia „if“ (a „if-else“):

- write('Meno: '); readln(meno);
  write('Heslo: '); readln(heslo);

```
if (meno = 'admin') and (heslo = 'asdfg') then
begin
    writeln('Vitajte!');
    ...
end;

writeln('Dovidenia...');
```

- write('Zadaj a: '); readln(a);
  write('Zadaj b: '); readln(b);

```
if b <> 0 then
     writlen('Podiel je: ', a / b);
else
     writlen('Nemôžem deliť nulou! ');
```

Príklady použitia „if“ (a „if-else“):

- String meno = čítajReťazec("meno");
  String heslo = čítajReťazec("heslo");

```
if (meno.equals("admin") && heslo.equals("asdfg"))
{
    vypíšRiadok("Vitajte!");
    …
}

vypíšRiadok("Dovidenia...");
```

- int a = čítajČíslo("zadaj a");
  int b = čítajČíslo("zadaj b");

```
if (b != 0)
     vypíšRiadok("Podiel je: " + (a / b));
else
     vypíšRiadok("Nemôžem deliť nulou!");
```

Viacnásobné vetvenie

- v prípade, že *«premenná»*

  má hodnotu *«hodnota1»* tak
  «príkaz(y)»
  …

  má hodnotu *«hodnota2»* tak
  «príkaz(y)»
  …

  inak
  «príkaz(y)»

Viacnásobné vetvenie

- case *«premenná»* of

  «hodnota»: *«príkaz»*;

  else
    «príkazy»;

  end;

- case *c* of

  1: a := 8;

  -3, *2*: a := c * 14;

  else
    a := –2 * c;

  end;

Viacnásobné vetvenie

- switch («premenná»)
  {
  case «hodnota»:
    «príkazy»;
    break;

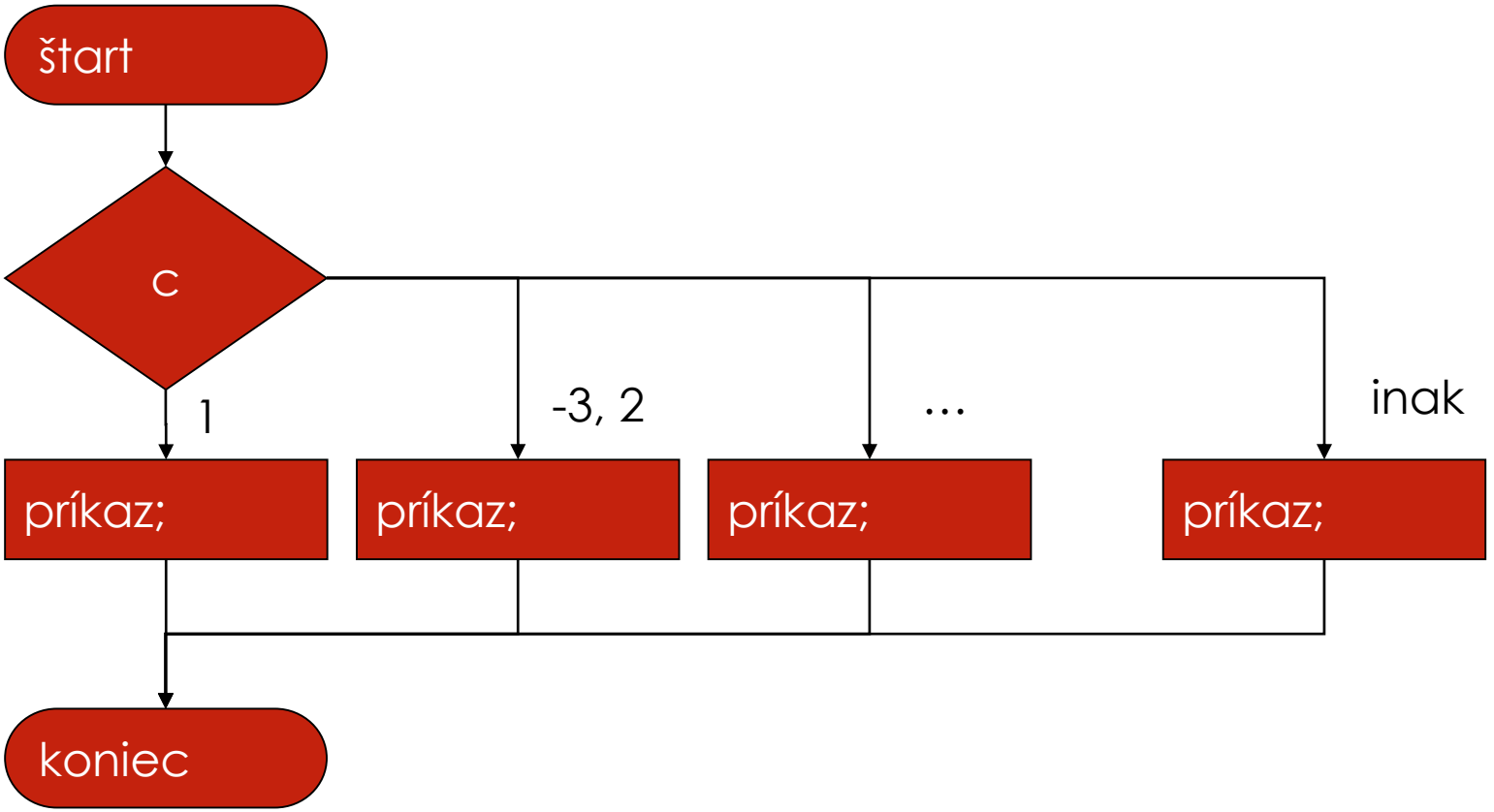
  default:
    «príkazy»;
  }

- switch (c)
  {
  case 1:
    a = 8;
    break;

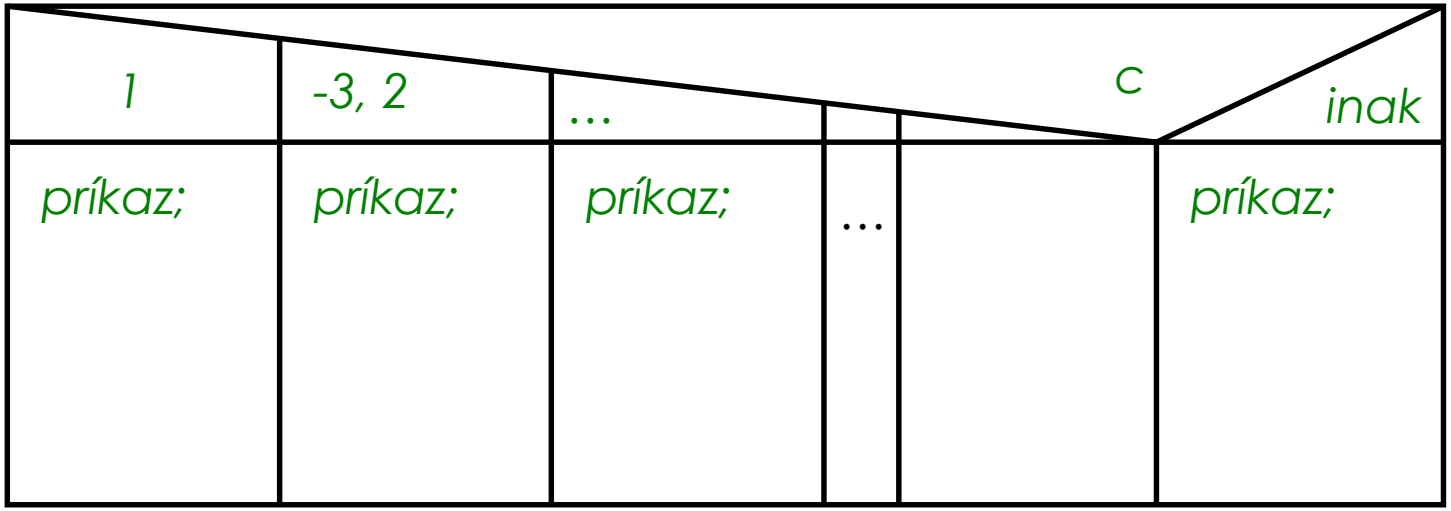
  case -3:
  case 2:
    a = c * 14;
    break;

  default:
    a = –2 * c;
  }

Viacnásobné vetvenie



Viacnásobné vetvenie



Príklad použitia „case-of“

```
writeln('Ponuka:');
writeln('  1. Vklad na účet');
writeln('  2. Výber v hotovosti');
writeln('  3. Žiadosť o hypotéku');

write('Zadaj voľbu: ');
readln(voľba);

case voľba of
    1: writeln('  -- zvolil si vklad na účet --');
    2: writeln('  -- zvolil si výber v hotovosti --');
    3: writeln('  -- zvolil si žiadosť o hypotéku --');

    else writeln('Neplatná voľba!');
end;
```

Príklad použitia „switch“

```
vypíšRiadok("Ponuka:");
vypíšRiadok("  1. Vklad na účet");
vypíšRiadok("  2. Výber v hotovosti");
vypíšRiadok("  3. Žiadosť o hypotéku");

int voľba = čítajČíslo("Zadaj voľbu");

switch (voľba)
{
    case 1: vypíšRiadok("  -- zvolil si vklad na účet --"); break;
    case 2: vypíšRiadok("  -- zvolil si výber v hotovosti --"); break;
    case 3: vypíšRiadok("  -- zvolil si žiadosť o hypotéku --"); break;

    default: vypíšRiadok("Neplatná voľba!");
}
```

Cykly s podmienkou na začiatku a s podmienkou na konci

- pokiaľ platí *«podmienka»* opakuj
  «príkaz(y)»
  …

  koniec cyklu

- opakuj
  «príkaz(y)»
  …
  pokiaľ platí *«podmienka»*

Cykly s podmienkou na začiatku a s podmienkou na konci

- while *«podmienka»* do
  begin
    «príkazy»;
    …
  end;

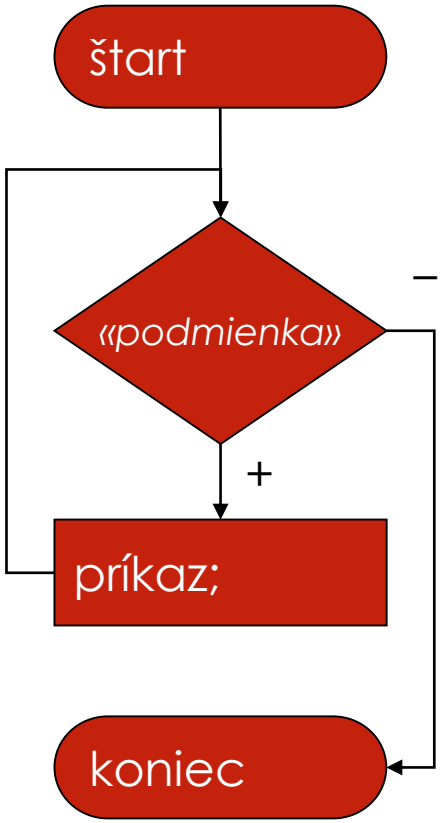
- repeat
    «príkazy»;
    …
  until *«negatívna podmienka»*;

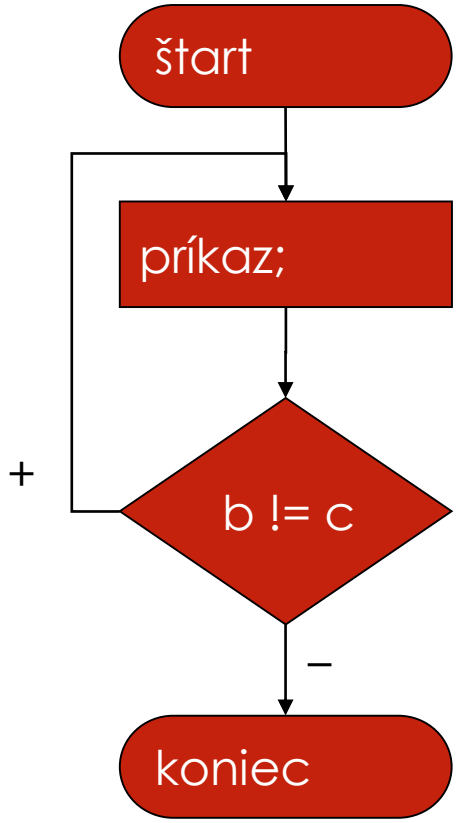
Cykly s podmienkou na začiatku a s podmienkou na konci

- while («podmienka»)
  {
    «príkazy»;
    …
  }

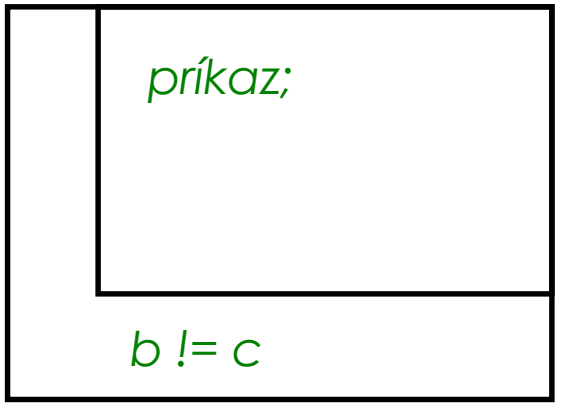
- do {
    «príkazy»;
    …
  } while («podmienka»)

Cykly s podmienkou na začiatku a s podmienkou na konci





Cyklus s podmienkou na začiatku a s podmienkou na konci



Príklad použitia „while-do“

```
domaciRozpocet := 0;
pocetPoloziek := 0;

writeln('Domáci rozpočet...');
write('Vložte prvú položku: ');
readln(ďalšiaHodnota);

while 0 <> ďalšiaHodnota do
begin
    domaciRozpocet := domaciRozpocet + ďalšiaHodnota;
    inc(pocetPoloziek);
    write('Vložte ďalšiu položku');
    readln(ďalšiaHodnota);
end;

writeln('Domáci rozpočet: ', domaciRozpocet);
writeln('Počet položiek:  ', pocetPoloziek);
```

Príklad použitia „while“

```
int domáciRozpočet = 0;
int početPoložiek = 0;

vypíšRiadok("Domáci rozpočet...");
int ďalšiaHodnota = čítajČíslo("Vložte prvú položku");

while (0 != ďalšiaHodnota)
{
    domáciRozpočet = domáciRozpočet + ďalšiaHodnota;
    ++početPoložiek;
    ďalšiaHodnota = čítajČíslo("Vložte ďalšiu položku");
}

vypíšRiadok("Domáci rozpočet: " + domáciRozpočet);
vypíšRiadok("Počet položiek:  " + početPoložiek);
```

Príklad použitia „repeat-until“

- var nevzniklaVynimka: boolean;

- …

- repeat
  nevzniklaVynimka := true;

  try
    write('Zadaj celé číslo: ');
    readln(retazec);
    cislo := StrToInt(retazec);
  except
    on e : EConvertError do
    writeln('Nastala výnimka!');
    nevzniklaVynimka := false;
  end;

  until nevzniklaVynimka;

- …

Príklad použitia „do-while“

- boolean vzniklaVýnimka;
- …
- do {
  vzniklaVýnimka = false;

```
try
{
    vypíšRiadok("Zadaj celé číslo: ");
    reťazec = vstup.readLine();
    číslo = Integer.parseInt(reťazec);
}
catch (Exception e)
{
    vypíšRiadok("Nastala výnimka!");
    vzniklaVýnimka = true;
}
```

  } while (vzniklaVýnimka);

- …

Cyklus s explicitným počtom iterácií (a cyklus na prechádzanie polí)

- pre každé «premenná» z rozsahu
  od *«spodná hranica»*
  do *«horná hranica»* opakuj

  «príkaz(y)»
  …

- pre každé *«premenná»* v poli *«pole»*
  opakuj

  «príkaz(y)»
  …

Cyklus s explicitným počtom iterácií

- for *«premenná»* := *«spodná hranica»* to *«horná hranica»* do
  begin
    «príkazy»;
    …
  end;

- for *«premenná»* := *«horná hranica»* downto *«spodná hranica»* do
  begin
    «príkazy»;
    …
  end;

- for i := 1 to 10 do
    writeln(i);

Cyklus s explicitným počtom iterácií

- for (*«inicializácia»*; *«podmienka»*; *«iteračný príkaz»*)
  {
  «príkazy»;
  …
  }

- for (int i = 0; i < 10; ++i)
  vypíšRiadok(i);

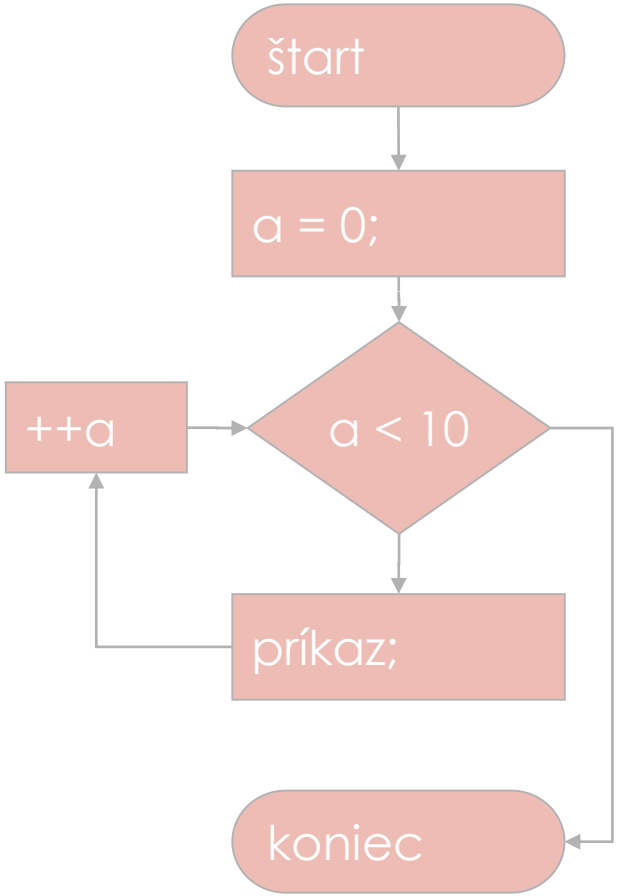
Cyklus na prechádzanie polí

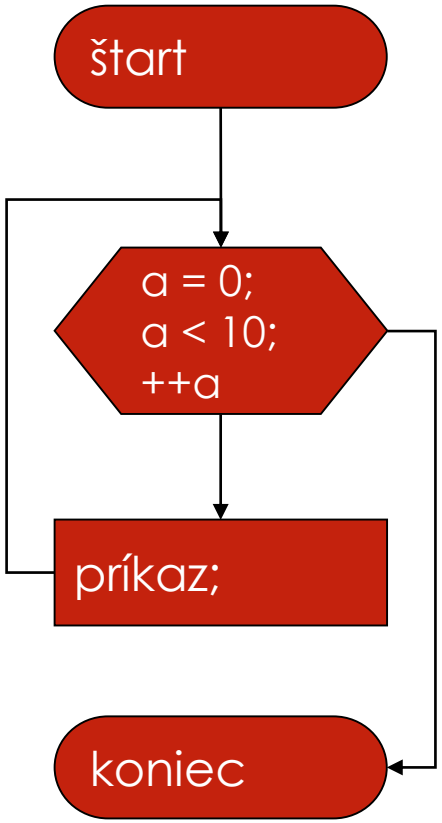
- for («premenná» : «pole»)
  {
  «príkazy»;
  …
  }

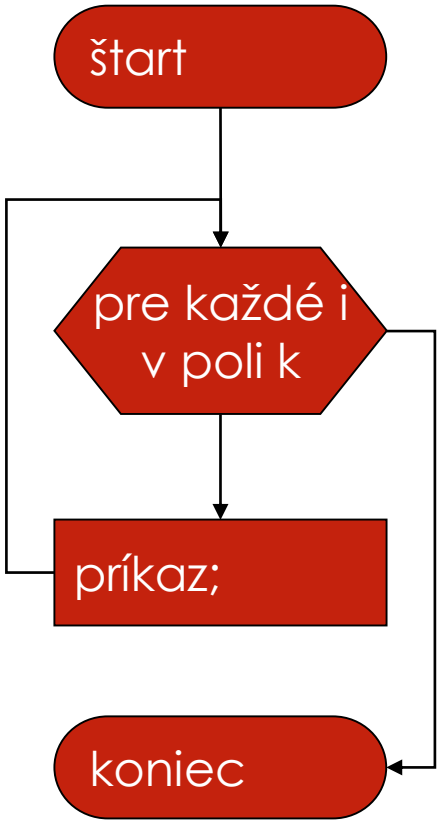
- int[] a;
  …
  int result = 0;

  for (int i : a) result += i;

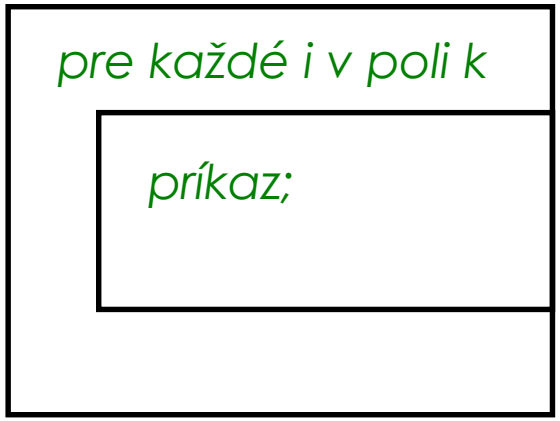
Cyklus s explicitným počtom iterácií (a cyklus na prechádzanie polí)

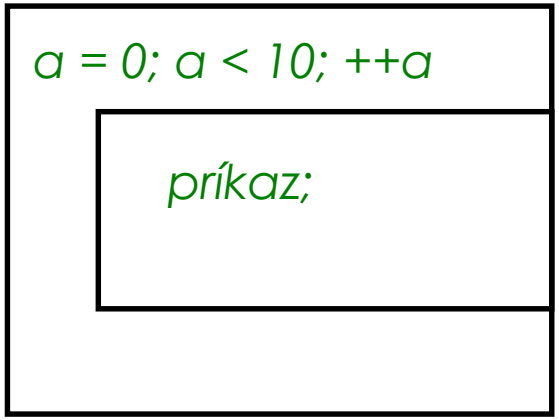




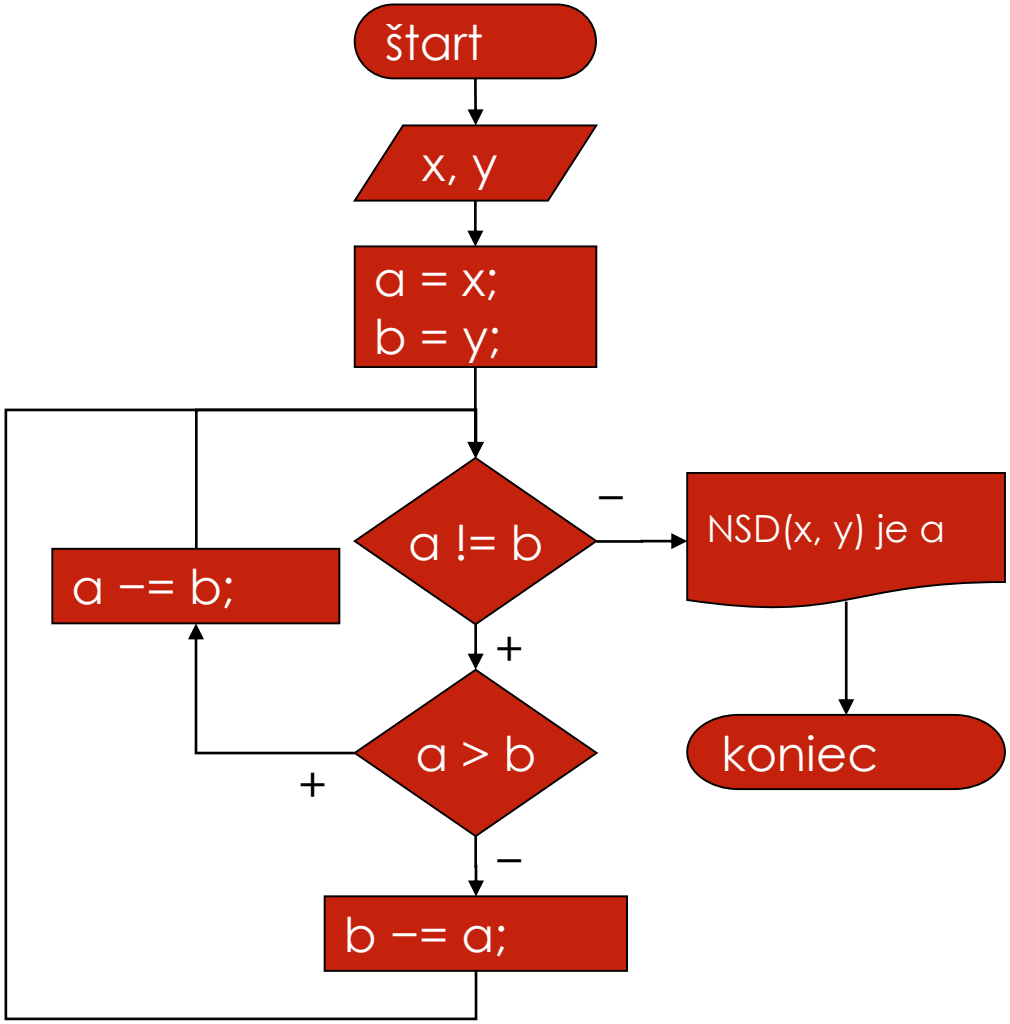


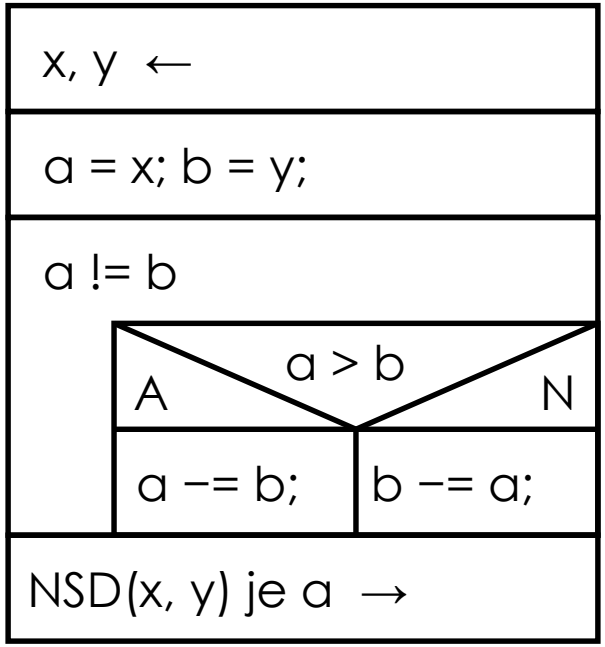
Cyklus s explicitným počtom iterácií (a cyklus na prechádzanie polí)





Najväčší spoločný deliteľ čísiel x, y





Najväčší spoločný deliteľ čísiel x, y

```
var x, y, a, b: integer;
begin
  write('x: '); readln(x);
  write('y: '); readln(y);

  a := x;
  b := y;

  while a <> b do
  begin
    if (a > b)
      a := a - b
    else
      b := b - a
  end;

  writeln('NSD(', x, ', ', y, ') je ', a);
end.
```

Najväčší spoločný deliteľ čísiel x, y

```
int x = čítajČíslo("x");
int y = čítajČíslo("y");

int a = x;
int b = y;

while (a != b)
{
    if (a > b)
        a -= b;
    else
        b -= a;
}

vypíšRiadok("NSD(", x, ", ", y, ") je ", a);
```