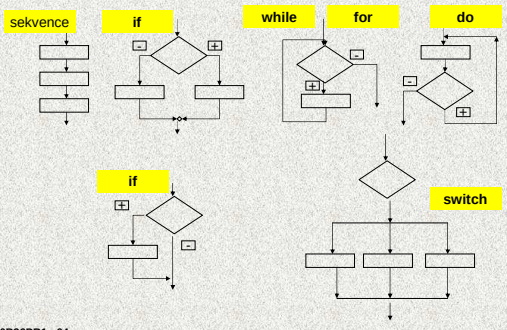
Řídící konstrukce

A0B36PR1-Programování 1
Fakulta elektrotechnická
České vysoké učení technické

Řídicí struktury

- Řídicí struktura je programová konstrukce, která se skládá z dílčích příkazů a předepisuje pro ně způsob provedení
- Tři druhy řídicích struktur:
  1. *posloupnost*, předepisující postupné provedení dílčích příkazů
  2. *větvení*, předepisující provedení dílčích příkazů v závislosti na splnění určité podmínky
  3. *cyklus*, předepisující opakované provedení dílčích příkazů v závislosti na splnění určité podmínky

Řídicí struktury



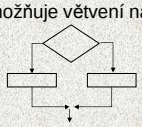
Řídicí struktury

- Budeme používat následující složené příkazy:
  1. složený příkaz nebo blok pro posloupnost
  2. příkaz **`if`** pro větvení
  3. příkazy **`while`**, **`do`** *nebo* **`for`** pro cyklus
- Řídicí struktury mají obvykle formu strukturovaných příkazů
- Další strukturované příkazy jazyka Java:
  - Složený příkaz: **{** <posloupnost příkazů> **}**
  - Blok: **{** <posloupnost deklarací a příkazů>**}**

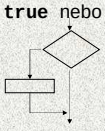
Pozn.:*Deklarace jsou v bloku lokální, tzn. neplatí vně bloku*

Příkaz **`if`**

- Příkaz **`if`** (podmíněný příkaz) umožňuje větvení na základě podmínky
- Má dva tvary:
  - if (*podmínka*) *příkaz*$_1$ else *příkaz*$_2$
  - if (*podmínka*) *příkaz*$_1$



kde *podmínka* je logický výraz
(výraz, jehož hodnota je typu **`boolean`**, tj. **`true`** nebo **`false`**)



Příkaz **`if`**

- Příklad (do *min* uložit a pak vypsat menší z hodnot *x* a *y*):

```
// 1. varianta
if (x < y) min = x; else min = y;
System.out.println(min);

// 2. varianta
int min = x;
if (y < min) min = y;
System.out.println(min);
```

Příkaz if

- Jestliže v případě splnění či nesplnění podmínky má být provedeno více příkazů, je třeba z nich vytvořit složený příkaz nebo blok
- Příklad: jestliže *x* < *y*, vyměňte hodnoty těchto proměnných
- Špatné řešení:

```
if (x < y)
  pom = x; // provede se pro x<y
  x = y; // provede se vždy !!!
  y = pom; // a co toto?
```

Správně

```
if (x < y)
{
   pom = x;
   x = y;
   y = pom;
}
```


Příkaz if

- Příklad: do *min* uložte menší z čísel *x* a *y* a do *max* uložte větší z čísel
- Špatné řešení:

```
if (x < y)
  min = x;
  max = y;
else
  min = y;
  max = x;
```

Správně

```
if (x < y) {
    min = x;
    max = y;
} else {
    min = y;
    max = x;
}
```


Příkaz if

- Do příkazu ***if*** lze vnořit libovolný příkaz, tedy i podmíněný příkaz
- Příklad: do *s* uložte –1, 0 nebo 1 podle toho, zda *x* je menší než nula, rovno nule nebo větší než nula

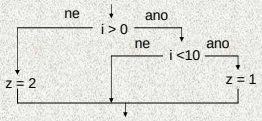
```
if (x < 0)s = -1;
else if (x == 0)s = 0;
else s = 1;
```

- Příklad: do *max* uložte největší z čísel *x*, *y* a *z*

```
if (x > y)
   if (x > z) max = x; else max = z;
else
   if (y > z) max = y; else max = z;
```

Příkaz if

- Pozor na vnoření neúplného *if* do úplného *if*
- Příklad: zapište příkazem *if* následující větvení:



- Špatně:

```
if (i > 0)
if (i < 10) z = 1;
  else z = 2;
```

Správně

```
if (i > 0) {
  if (i < 10) z = 1
} else z = 2;
```


Pořadí podmínek a rychlost programu

- … číslo od nuly do 9999 určí, kolik … desítkový zápis

Lze to „rychleji“:
(int)Math.log10(x+0.1)+1;
// + 0.1 kvůli 0, zkuste to odstranit a co dostanete?

7 ... jedno místo
58 ... dvě
1101 ... 4 místa

```
// řešení 1.
if(x<10) p = 1;
else if(x<100) p =
else if(x<1000)p =
else p = 4;
```

```
// řešení 2.
if(x>999) p = 4;
else if(x>99) p = 3;
else if(x>9) p = 2;
else p = 1;
```

Ob… …ytu… …ak
st…
S… …999

číslo 0 – 9, jedno porovnání
číslo 10 - 99, dvě
číslo 100 – 999, tři
číslo 1000 – 9999, čtyři
10*1+90*2+900*3+9000*3 = 29890 porovnání

číslo 0 – 9, čtyři
číslo 10 - 99, tři
číslo 100 – 999, dvě
číslo 1000 – 9999, jedno
10*4+90*3+900*2+9000*1 = 11110 porovnání

Druhé řešení je téměř 3 krát rychlejší !!!

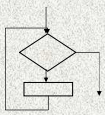
Příklad

- Program, který pro zadaný rok zjistí, zda je přestupný
- Přestupný rok je dělitelný 4 a buď není dělitelný 100 nebo je dělitelný 1000

```
import java.util.*;

public class Rok {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int rok;
    System.out.println("Zadejte rok");
    rok = sc.nextInt();
    System.out.println("rok " + rok);
    if (rok%4==0 && (rok%100!=0 || rok%1000==0))
      System.out.println(" je přestupný");
    else
      System.out.println(" není přestupný");
  }
}
```

Příkaz while



- Základní příkaz cyklu, který má tvar
  while (*podmínka*) *příkaz*
- Příklad:

```
q = x;
while (q>=y) q = q-y;
```

(jsou-li hodnotami proměnných *x* a *y* přirozená čísla, co je hodnotou proměnné *q* po skončení uvedeného cyklu?)

Vyzkoušejte pro x = 10, y = 3.

Příkaz while

- Má-li se opakovaně provádět více příkazů, musí tvořit složený příkaz
- Příklad:

```
q = x;
p = 0;
while (q>=y) {
  q = q-y;
  p = p+1;
}
```

(jsou-li hodnotami proměnných *x* a *y* přirozená čísla, co je hodnotou proměnných *p* a *q* po skončení uvedeného cyklu?)

Vyzkoušejte pro x = 10, y = 3.

Příklad

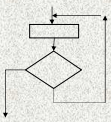
Výpočet faktoriálu přirozeného čísla *n* (*n*! = 1 × 2 × ... × *n*)

```
public class Faktorial {
  public static void main(String[] args) {
    System.out.println("zadejte přirozené číslo");
    int n = sc.nextInt();
    if (n<1) {
      System.out.println(n + " není přirozené číslo");
      System.exit(0);
    }
    int i = 1;  int f = 1;
    while (i<n) {
      i = i+1;
      f = f * i;
    }
    System.out.println (n + "! = " + f);
  }}
```

Příkaz do

- Příkaz cyklu *do* se od příkazu *while* liší v tom, že podmínka se testuje až za tělem cyklu
- Tvar příkazu:
  - do *příkaz* while (*podmínka*);
- Vnořeným příkazem je nejčastěji složený příkaz
- Příklad (faktoriál):

```
f = 1; i = 0;
do {
  i = i+1;   f = f*i;
} while (i<n);
```



- Poznámka k sémantice: příkaz *do* provede tělo cyklu alespoň jednou, nelze jej tedy použít v případě, kdy lze očekávat ani jedno provedení těla cyklu

Příkaz for

- Cyklus je často řízen proměnnou, pro kterou je stanoveno:
  - jaká je počáteční hodnota
  - jaká je koncová hodnota
  - jak změnit hodnotu proměnné po každém provedení těla cyklu
- Příklad:

```
f = 1; i = 1;  // počáteční hodnota řídicí proměnné
while (i<=n) { // podmínka určující koncovou hodnotu
  f = f*i;     // tělo cyklu
  i = i+1;     // změna řídicí proměnné
}
```

- Cykly tohoto druhu lze zkráceně předepsat příkazem *for*:

```
f = 1;
for (i=1; i<=n; i=i+1) f=f*i;
```

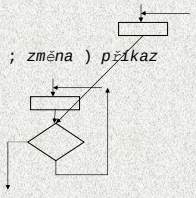

Příkaz for

- Tvar příkazu *for*:

```
for ( inicializace ; podmínka ; změna ) příkaz
```

- Provedení příkazu *for* :

```
inicializace;
while (podmínka) {
  příkaz
  změna;
}
```



- Změnu řídicí proměnné přičtením resp. odečtením 1 lze zkráceně předepsat pomocí operátoru inkrementace resp. dekrementace:
  - *x*++ ***x* se zvětší o 1**
  - *x*-- ***x* se zmenší o 1**

Příkaz for

Příklad:

```
f = 1;
for (i=1; i<=n; i++) f=f*i;
```

Příklad 2a

```
for (int i=1; i<10; i++) {
  System.out.println(i);
}
```

Příklad 2a

```
for (int i=1; i<10; ++i) {
   // místo i++ je zde ++i, jak se změní výstup?
  System.out.println(i);
}
```


Zpracování posloupností

- Příklad: program pro součet posloupnosti čísel
- Hrubé řešení:

```
suma = 0;
while (nejsou přečtena všechna čísla) {
  dalsi = přečti celé číslo;
  suma = suma+dalsi;
}
```

- Jak určit, zda jsou přečtena všechna čísla?
- Možnosti:
  1. počet čísel bude vždy stejný, např. 5
  2. počet čísel bude dán na začátku vstupních dat
  3. vstupní data budou končit „zarážkou“, např. nulou

Zpracování posloupností

- Jak určit, zda jsou přečtena všechna čísla?
- Možnosti:
  1. počet čísel bude vždy stejný, např. 5
  2. počet čísel bude dán na začátku vstupních dat
  3. vstupní data budou končit „zarážkou“, např. nulou
- Struktura vstupních dat formálně:
  1. $a_1\ a_2\ a_3\ a_4\ a_5$
  2. $n\ a_1\ a_2\ \ldots\ a_n$
  3. $a_1\ a_2\ \ldots\ a_5\ 0$ kde $a_i \neq 0$

Zpracování posloupností

- Řešení 1: vstupní data: $a_1\ a_2\ a_3\ a_4\ a_5$

```
public class Suma1 {
  public static void main(String[] args) {
   Scanner sc = new Scanner(System.in);
   int dalsi, suma, i;
    System.out.println ("zadejte 5 čísel");
    suma = 0;
    for (i=1; i<=5; i++) {
      dalsi = sc.nextInt();  suma = suma+dalsi;
    }
    System.out.println ("součet je " + suma);
  }
}
```


Zpracování posloupností

- Řešení 2 vstupní data: $n\ a_1\ a_2\ \ldots\ a_n$

```
public class Suma2 {
public static void main(String[] args) {
int dalsi, suma, i, n;
System.out.println("zadejte počet čísel");
n = sc.nextInt();
System.out.println("zadejte " + n +" čísel");
suma = 0;
for (i=1; i<=n; i++) {
dalsi = sc.nextInt(); suma = suma+dalsi;
  }
System.out.println("součet je " + suma);
  }
}
```


Zpracování posloupností

- Řešení 3 vstupní data: $a_1\ a_2\ \ldots\ a_n\ 0$, kde $a_i \neq 0$

```
public class Suma3 {
  public static void main(String[] args) {
int dalsi, suma;
System.out.println("zadejte řadu čísel zakončenou
                                    nulou");
    suma = 0;
    dalsi = sc.nextInt();
    while (dalsi!=0) {
      suma = suma+dalsi; dalsi = sc.nextInt();
    }
 System.out.println("součet je " + suma);
  }
}
```

Příkaz continue

- Příkazy *while* a *for* testují ukončení cyklu před provedením těla cyklu
- Příkaz *do* testuje ukončení cyklu po provedení těla cyklu
- Někdy je třeba ukončit cyklus v nějakém místě uvnitř těla cyklu (které je v tom případě tvořeno složeným příkazem)
- Příkaz *continue* předepisuje předčasné ukončení průchodu těla cyklu
- Příklad:

```
for (int i=1; i<=100; i++) {
  if (i%10==0) continue;
  System.out.println(i);
}
```

- příkaz vypíše čísla od 1 do 100 s výjimkou dělitelných 10

Příkaz break

- Příkaz *break* vnořený do podmíněného příkazu ukončí předčasně příkaz, schematicky:

```
while (...) {
  ...
  if (ukončit) break;
  ...
}
```

- Příkaz `break` předepisuje předčasné ukončení těla cyklu
- Příklad:

```
for (int i=1; i<=100; i++) {
  if (i%10==0) break;
  System.out.println(i);
}
```

- příkaz vypíše čísla od 1 do 9

Konečnost cyklů

- Aby algoritmus byl konečný, musí každý cyklus v něm uvedený skončit po konečném počtu kroků
- Nekonečný cyklus je častou chybou
- Základní pravidlo pro konečnost cyklu:
  - provedením těla cyklu se musí změnit hodnota proměnné vyskytující se v podmínce cyklu
- Triviální příklad špatného cyklu:

```
while (i!=0) j = i -1;
```

  Tento cyklus se buď neprovede ani jednou, nebo neskončí
- Uvedené pravidlo konečnost cyklu ještě nezaručuje
- Konečnost cyklu závisí na hodnotách proměnných před vstupem do cyklu

```
double x=0; int i = -1;
while (i<0) {x = x + Math.sin(i* 0.6); i--;}
```

Konečnost cyklů

```
while (i!=n) {
  P;     // příkaz, který nezmění hodnotu proměnné i
  i++;
}
```

- Otázka: co musí splňovat hodnoty proměnných *i* a *n* před vstupem do cyklu, aby cyklus skončil?
- Odpověď – vstupní podmínka konečnosti cyklu:
  i <= n (pro celá čísla, jak je pro typ double)
- Vstupní podmínku konečnosti cyklu lze určit téměř ke každému cyklu (někdy je to velmi obtížné, až nespočetné)
- Splnění vstupní podmínky konečnosti cyklu musí zajistit příkazy předcházející příkazu cyklu

Zabezpečený program testuje přípustnost vstupních dat

Konečnost cyklů – problém 3x+1

Problém se nazývá též syrakuský, Collatzův,...

Vstup: přirozené číslo n

```
while(n!=1){
  if(n%2==0) n = n/2;
  else n = 3*n + 1;
}
```

problém: skončí tento algoritmus a po kolika krocích?

pro n = 6 ... 6, 3, 10, 5, 16, 8, 4, 2, 1

pro n = 27 ... 27, 82, 41, 124, 62, 31, 94, 47, 142, 71, 214, 107, 322, 161, 484, 242,..., 9232, ..., 4, 2, 1 (111 iterací)

Pro zajímavost

Cyklus for – programujeme efektivně

Příklad – zjištění, zda x je prvočíslo

```
boolean jePrvocislo = true;
for (int i=2; i<=(int)Math.sqrt(x); i++){
    if(x%i==0){
   jePrvocislo = false;
      break;
    }
}
```

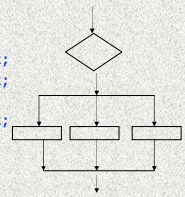
- Při nalezení prvního dělitele je zbytečné pokračovat ve zkoušení dalších hodnot
- Výraz `(int)Math.sqrt(x)` je počítán v každém cyklu i když se jeho hodnota nemění:

```
final  int HORNI_MEZ = (int)Math.sqrt(x);
 for (int i=2; i<=HORNI_MEZ ; i++){...
```

Příkaz switch

- Příkaz *switch* (přepínač) umožňuje větvení do více větví na základě různých hodnot výrazu (nejčastěji typu *int* nebo *char*)
- Základní tvar příkazu:

```
switch (výraz) {
  case konstanta1 : příkazy1 ; break;
  case konstanta2 : příkazy2 ; break;
  ...
  case konstantan : příkazyn ; break;
  default : příkazydef ;
}
```



kde *konstanty* jsou téhož typu, jako *výraz*
příkazy jsou posloupnosti příkazů

- Sémantika (zjednodušeně):
  - vypočte se hodnota *výrazu* a pak se provedou ty *příkazy*, které jsou označeny *konstantou* označující stejnou hodnotu
  - není-li žádná větev označena hodnotou výrazu, provedou se $příkazy_{def}$

pro zájemce

Příkaz switch

```
public class PrikazSwitch {}:
    switch (n) {
      case 1: System.out.print ("*"); break;
      case 2: System.out.print ("**"); break;
      case 3: System.out.print ("***"); break;
      case 4: System.out.print ("****"); break;
      default: System.out.print ("---");
    }
```

```
***
```

- Příkaz *break* dynamicky ukončuje větev
- Pokud jej neuvedeme, pokračuje se v provádění další větve !!!
- Příklad: co se vypíše, má-li *n* hodnotu 3 a příkaz *switch* zapíšeme takto:

```
switch (n) {
  case 1: System.out.print ("*");
  case 2: System.out.print ("**");
  case 3: System.out.print ("***");
  case 4: System.out.print ("****");
  default: System.out.print ("---");
}
```

```
***
****
---
```


pro zájemce

Příklad – den v roce

Program pro výpočet pořadového čísla dne v roce

```
public class Den {
public static void main(String[] args) {
        System.out.println("zadejte den, měsíc a rok");
        int den = sc.nextInt();
        int mesic = sc.nextInt();
        int rok = sc.nextInt();
        int n = 0;
        switch (mesic) {
          case 1: n = den; break;
          case 2: n = 31+den; break;
          case 3: n = 59+den; break;
          case 4: n = 90+den; break;
          case 5: n = 120+den; break;
          case 6: n = 151+den; break;
          ...
          case 12: n = 334+den; break;
        }
    if (mesic>2 && rok%4==0 && (rok%100!=0 || rok%1000==0))
          n = n+1;
    Sys...print (den+"."+mesic+"."+rok+" je "+n+". den v roce");
      }
    }
```


pro zájemce

Příkaz **for** - detaily I

```
Nevhodná řešení :
for (int i=1; i<=n; i++) f=f*i;//Systém.out.print(i); nezná i!

int i;
for (i=1; i<=n; i++) f=f*i; Systém.out.print(i); // i?

int i=0;
for (; i<=n; i++) f=f*i;          //nevhodně mimo

int i=0;
for (; i<=n;) f=f*i++;            // inkrementace mimo

int i=0;
for (i=1; i<=n; f=f*i) i++;       // přehozené
```


Ještě k příkazu **for II**

```
int i=0;
for (; i<=n; f=f*i++);            // smíšené

int i=1;
for (; ;) {
if (i>n) break; f=f*i++;          // nesmyslné
 }


int i=1;
for (; i<=n; f=f*i, i++);         // nesmyslné
```