Řídicí struktury

- Řídicí struktura je programová konstrukce, která se skládá z dílčích příkazů a předepisuje pro ně způsob provedení
- Tři druhy řídicích struktur:
  - *posloupnost*, předepisující postupné provedení dílčích příkazů
  - *větvení*, předepisující provedení dílčích příkazů v závislosti na splnění určité podmínky
  - *cyklus*, předepisující opakované provedení dílčích příkazů v závislosti na splnění určité podmínky
- Řídicí struktury mají obvykle formu strukturovaných příkazů
- Budeme používat následující složené příkazy:
  - složený příkaz nebo blok pro posloupnost
  - příkaz *if* pro větvení
  - příkazy *while nebo for* pro cyklus

  (Pro zájemce uvedeme i zbývající strukturované příkazy jazyka Java)
- Složený příkaz: **{** posloupnost příkazů **}**
- Blok: **{** posloupnost deklarací a příkazů**}**

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

```
// 1. varianta
if (x < y) min = x; else min = y;
Sys.pln(min);

// 2. varianta
min = x;
if (y < min) min = y;
Sys.pln(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

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

- Špatné řešení:

```
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

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


- Špatné řešení:

```
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

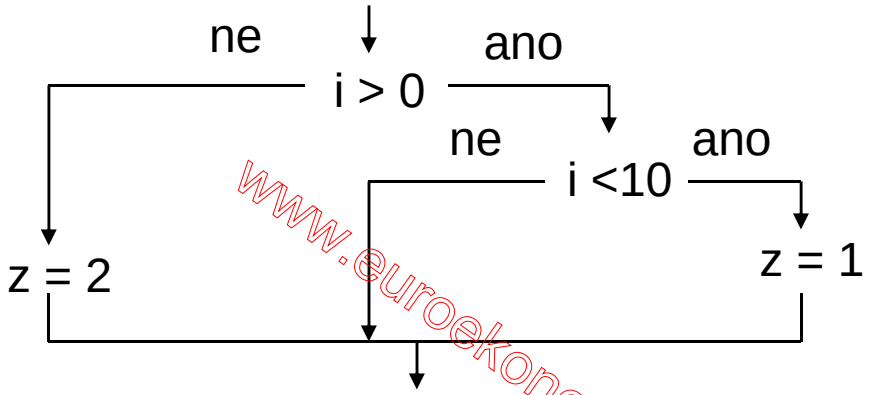
```
if (x < 0)
  s = -1;
else if (x == 0)
  s = 0;
else
  s = 1;
```


- 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:



- Dobře:

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

- Špatně:

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

Příklad

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

```
package alg3;

import sugar.Sys;

public class Rok {
  public static void main(String[] args) {
    int rok;
    Sys.pln("zadejte rok");
    rok = Sys.readInt();
    Sys.p("rok "+rok);
    if (rok%4==0 && (rok%100!=0 || rok%1000==0))
      Sys.pln(" je přestupný");
    else
      Sys.pln(" 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?)
- 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?)

Příklad

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

```
public class Faktorial {
  public static void main(String[] args) {
    Sys.pln("zadejte přirozené číslo");
    int n = Sys.readInt();
    if (n<1) {
      Sys.pln(n + " není přirozené číslo");
      System.exit(0);
    }
    int i = 1;
    int f = 1;
    while (i<n) {
      i = i+1;
      f = f * i;
    }
    Sys.pln(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 syntaxi: příkaz *do* je jediným strukturovaným příkazem, který končí středníkem
- 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

- Příklad:

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

```
package alg3;
import sugar.Sys;

public class Suma1 {
  public static void main(String[] args) {
    int další, suma, i;
    Sys.pln("zadejte 5 čísel");
    suma = 0;
    for (i=1; i<=5; i++) {
      dalsi = Sys.readInt();
      suma = suma+dalsi;
    }
    Sys.pln("součet je " + suma);
  }
}
```

Zpracování posloupností

- Řešení 2

  vstupní data: $n\ a_1\ a_2\ \ldots\ a_n$

```
package alg3;
import sugar.Sys;

public class Suma2 {
  public static void main(String[] args) {
    int další, suma, i, n;
    Sys.pln("zadejte počet čísel");
    n = Sys.readInt();
    Sys.pln("zadejte " + n +" čísel");
    suma = 0;
    for (i=1; i<=n; i++) {
      dalsi = Sys.readInt();
      suma = suma+dalsi;
    }
    Sys.pln("součet je " + suma);
  }
}
```

Zpracování posloupností

- Řešení 3

  vstupní data: $a_1\ a_2\ \ldots\ a_n\ 0$ kde $a_i \neq 0$

```
package alg3;
import sugar.Sys;

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

Příkazy break a 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)
- K tomu slouží příkaz *break* vnořený do podmíněného příkazu

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


- Příkaz *continue* předepisuje předčasné ukončení těla cyklu
- Příklad: následující příkaz vypíše čísla od 1 do 100 s výjimkou dělitelných 10:

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

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

Konečnost cyklů

- Příklad:

```
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*

- Vstupní podmínku konečnosti cyklu lze určit ke každému cyklu (někdy je to velmi obtížné)
- Splnění vstupní podmínky konečnosti cyklu musí zajistit příkazy předcházející příkazu cyklu
- Zásada: zabezpečený program testuje přípustnost vstupních dat

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 $příkazy_{def}$

Příkaz switch

- Příklad:

```
switch (n) {
  case 1: Sys.p("*"); break;
  case 2: Sys.p("**"); break;
  case 3: Sys.p("***"); break;
  case 4: Sys.p("****"); break;
  default: Sys.p("---");
}
```

- 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: Sys.p("*");
  case 2: Sys.p("**");
  case 3: Sys.p("***");
  case 4: Sys.p("****");
  default: Sys.p("---");
}
```

Příklad

- Program pro výpočet pořadového čísla dne v roce

```
public class Den {
  public static void main(String[] args) {
    Sys.pln("zadejte den, měsíc a rok");
    int den = Sys.readInt();
    int mesic = Sys.readInt();
    int rok = Sys.readInt();
    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.pln(den+"."+mesic+"."+rok+" je "+n+". den v roce");
  }
}
```