#ídicí struktury: v#tvení, cykly.

Obsah

P#íkazy a #ídicí struktury v Jav#

- P#íkazy v Jav#
- P#i#azení
- Volání metod a návrh z nich
- #ídicí p#íkazy (v#tvení, cykly)

P#íkazy v Jav#

V Jav# máme následující p#íkazy:

- P#i#azovací p#íkaz = a jeho modifikace (kombinované operátory jako je += apod.)

- #ízení toku programu (v#tvení, cykly)
- Volání metody
- Návrat z metody - p#íkaz **return**
- P#íkaz je ukon#en st#edníkem **;**
- v Pascalu st#edník p#íkazy *odd#luje*, v Jav# (C/C++) *ukon#uje*

P#i#azení

P#i#azení v Jav#

- Operátor p#i#azení = (assignment)
- na levé stran# musí být prom#nná
- na pravé stran# výraz *p#i#aditelný* (assignable) do této prom#nné
- Rozlišujeme p#i#azení primitivních hodnot a odkaz# na objekty

P#i#azení primitivní hodnoty

- Na pravé stran# je výraz vracející hodnotu primitivního typu
- #íslo, logická hodnotu, znak (ale ne nap#. #et#zec)
- Na levé stran# je prom#nná téhož nebo širšího typu jako p#i#azovaná hodnota
- nap#. `int` `[http://www.google.com/search?q=int]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=int] lze p#i#adit do `long` `[http://www.google.com/search?q=long]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=long]
- P#i zužujícím p#i#azení se také provede konverze, ale m#že dojít ke ztrát# informace
- nap#. `int` `[http://www.google.com/search?q=int]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=int] -> `short` `[http://www.google.com/search?q=short]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=short]
- P#i#azením primitivní hodnoty se hodnota zduplikuje ("opíše") do prom#nné na levé stran#.

P#i#azení odkazu na objekt

Konstrukci `=` `[http://www.google.com/search?q==]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search==] lze použít i pro p#i#azení do objektové prom#nné:

```
Person z1 = new Person(); [http://www.google.com/search?q=Person z1 =
```

`new Person();]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Person z1 = new Person();]

Co to ud#lalo?

1. vytvo#ilo nový objekt typu `Person [http://www.google.com/search?q=Person]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Person] (`new Person() [http://www.google.com/search?q= new Person() ]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search= new Person()])

2. p#i#adilo jej do prom#nné `z1 [http://www.google.com/search?q=z1]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=z1] typu `Person [http://www.google.com/search?q=Person]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Person]

Nyní m#žeme *odkaz* na tentýž vytvo#ený objekt znovu p#i#adit - do `z2 [http://www.google.com/search?q=z2]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=z2]:

`Person z2 = z1; [http://www.google.com/search?q=Person z2 = z1;]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Person z2 = z1;]

Prom#nné `z1 [http://www.google.com/search?q=z1]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=z1] a `z2 [http://www.google.com/search?q=z2]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=z2] ukazují nyní na stejný objekt typu osoba!!!

Prom#nné objektového typu obsahují *odkazy* (reference) na objekty, ne objekty samotné!!!

Volání metod a návrat z nich

Volání metody

Metoda objektu je vlastn# procedura/funkce, která realizuje svou #innost primárn# s prom#nnými objektu.

Volání metody ur#itého objektu realizujeme:

identifikaceObjektu.názevMetody(skute#né parametry)

- **identifikaceObjektu**, jehož metodu voláme
- **.** (te#ka)
- **názevMetody**, jíž nad daným objektem voláme
- v závorách uvedeme *skute#né parametry* volání (záv. m#že být prázdná, nejsou-li parametry)

Návrat z metody

Návrat z metody se d#je:

1. Bu#to automaticky posledním p#íkazem v t#le metody
2. nebo explicitn# p#íkazem **return** ***návratováhodnota*** **[http://www.google.com/search?q=return]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=return]

zp#sobí ukon#ení provád#ní t#la metody a návrat, p#i#emž m#že být specifikována *návratová hodnota*

typ skute#né návratové hodnoty musí korespondovat s deklarovaným typem návratové hodnoty

#ízení toku uvnit# metod - v#tvení, cykly

#ízení toku programu v t#le metody

P#íkaz (neúplného) v#tvení **if** **[http://www.google.com/search?q=if]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=if]

`if (logický výraz) p#íkaz` [http://www.google.com/search?q=if (logický výraz) p#íkaz] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=if (logický výraz) p#íkaz]

platí-li logický výraz (má hodnoty true), provede se p#íkaz

P#íkaz úplného v#tvení **if - else [http://www.google.com/search?q=else] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=else] [http://www.google.com/search?q=if -]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=if -]

```
if (logický výraz)
   p#íkaz1
else
   p#íkaz2
```

platí-li `logický výraz` [http://www.google.com/search?q= logický výraz] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search= logický výraz] (má hodnoty `true` [http://www.google.com/search?q=true] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=true]), provede se `p#íkaz1` [http://www.google.com/search?q= p#íkaz1] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search= p#íkaz1]

neplatí-li, provede se `p#íkaz2` [http://www.google.com/search?q= p#íkaz2] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search= p#íkaz2]

V#tev **else** **[http://www.google.com/search?q=else]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=else] se **nemusí uvád#t**

Cyklus s podmínkou na za#átku

T#lo cyklu se provádí tak dlouho, **dokud** platí podmínka

obdoba **while** **[http://www.google.com/search?q=while]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=while] v Pascalu

v t#le cyklu je jeden jednoduchý p#íkaz ...

```
while (podmínka)
   p#íkaz;
```

... nebo p#íkaz složený

```
while (podmínka) {
   p#íkaz1;
   p#íkaz2;
   p#íkaz3;
   ...
}
```

T#lo cyklu se nemusí provést ani jednou - pokud už hned na za#átku podmínka neplatí

Doporu#ení k psaní cykl#/v#tvení

V#tvení, cykly: doporu#uji vždy psát se **složeným p#íkazem v t#le** (tj. se složenými závorkami)!!!

jinak hrozí, že se v t#le v#tvení/cyklu z neopatrnosti p#i editaci objeví n#co jiného, než chceme, nap#.:

```
while (i < a.length)
   System.out.println(a[i]); i++;
```

Provede v cyklu jen ten výpis, inkrementaci ne a program se zacyklí.

Pišme proto vždy takto:

```
while (i < a.length) {
    System.out.println(a[i]); i++;
}
```

P#íklad použití "while" cyklu

Dokud nejsou p#e#teny všechny vstupní argumenty:

```
int i = 0;
while (i < args.length) {
   //"p#e#ti argument args[i]"
   i++;
}
```

Dalším p#íkladem je použití **while [http://www.google.com/search?q=while]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=while] pro realizaci celo#íselného d#lení se zbytkem:

P#íklad: Celo#íselné d#lení se zbytkem [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/DeleniOdcitanim.java]

Cyklus s podmínkou na konci

T#lo se provádí **dokud** platí podmínka (vždy aspo# jednou)

obdoba **repeat [http://www.google.com/search?q=repeat]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=repeat] v Pascalu (podmínka je

ovšem *interpretována opa#n#*)

Relativn# málo používaný - je mén# p#ehledný než **while [http://www.google.com/search?q=while]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=while]

Syntaxe:

```
do {
   p#íkaz1;
   p#íkaz2;
   p#íkaz3;
   ...
} while (podmínka);
```

P#íklad použití "do-while" cyklu

Dokud není z klávesnice na#tena požadovaná hodnota:

```
String vstup = "";
float number;
boolean isOK;
// create a reader from standard input
BufferedReader in = new BufferReader(new InputStream(System.in));
// until a valid number is given, try to read it
do {
   String input = in.readLine();
   try {
      number = Float.parseFloat(input);
      isOK = true;
   } catch (NumberFormatException nfe) {
      isOK = false;
   }
} while(!isOK);
System.out.println("We've got the number " + number);
```

P#íklad: Na#ítej, dokud není zadáno #ísło [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/DokudNeniZadano.java]

Cyklus "for"

obecn#jší než **for [http://www.google.com/search?q=for]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=for] v Pascalu, podobn# jako v C/C++

De-facto jde o rozší#ení **while [http://www.google.com/search?q=while]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=while], lze jím snadno nahradit

Syntaxe:

```
for(po#áte#ní op.; vstupní podm.; p#íkaz po každém pr#ch.)
   p#íkaz;
```

anebo (obvyklejší, bezpe#n#jší)

```
for (po#áte#ní op.; vstupní podm.; p#íkaz po každém pr#ch.) {
   p#íkaz1;
   p#íkaz2;
   p#íkaz3;
```

```
   ...
}
```

P#íklad použití "for" cyklu

Provedení ur#ité sekvence ur#itý po#et krát

```
for (int i = 0; i < 10; i++) {
   System.out.println(i);
}
```

Vypíše na obrazovku deset #ádk# s #ísly postupn# 0 až 9

1. P#íklad: P#t pozdrav# [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/PetPozdravu.java]
2. P#íklad: Výpis prvk# pole objekt# "for" cyklem [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/PolozkyForCyklem.java]

Doporu#ení k psaní `for` [http://www.google.com/search?q=for] http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:S [earch?search=for] cykl# (1)

Používejte asymetrické intervaly (ostrá a neostrá nerovnost):

- podmínka daná po#áte#ním p#i#azením **i = 0 [http://www.google.com/search?q=i = 0]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=i = 0] a inkrementací **i++ [http://www.google.com/search?q=i++]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=i++] je *neostrou nerovností*, zatímco
- opakovací podmínka **`i < 10` [http://www.google.com/search?q=i < 10]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=i < 10] [http://www.google.com/search?q=] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=] je *ostrou nerovností* -> i už nesmí hodnoty 10 dosáhnout!

Vytvarujte se složitých p#íkaz# v hlavi#ce (kulatých závorkách) **for [http://www.google.com/search?q=for]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=for] cyklu -

- je lepší to napsat podle situace p#ed cyklus nebo až do jeho t#la

Doporu#ení k psaní `for` [http://www.google.com/search?q=for]

http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=for] cykl# (2)

N#kte#í auto#i nedoporu#ují psát deklaraci #ídicí prom#nné p#ímo do závorek cyklu

`for (int i = 0; ... [http://www.google.com/search?q=for (int i = 0; ...]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=for (int i = 0; ...]

ale rozepsat takto:

```
int i;
for (i = 0; ...
```

potom je prom#nná `i [http://www.google.com/search?q=i]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=i] p#ístupná ("viditelná") i mimo cyklus - za cyklem, což se však ne vždy hodí.

Vícecestné v#tvení "switch - case - default"

Obdoba pascalského **select - case - else [http://www.google.com/search?q=select - case - else]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=select - case - else]

V#tvení do více možností na základ# ordinální hodnoty

Syntaxe:

```
switch(výraz) {
   case hodnota1: prikaz1a;
                  prikaz1b;
                  prikaz1c;
                  ...
                  break;
   case hodnota2: prikaz2a;
                  prikaz2b;
                  ...
                  break;
   default:       prikazDa;
                  prikazDb;
                  ...
}
```

Je-li *výraz* roven n#které z hodnot, provede se sekvence uvedená za p#íslušným **case [http://www.google.com/search?q=case]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=case].

Sekvenci obvykle ukon#ujeme p#íkazem **break [http://www.google.com/search?q=break]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=break], který p#edá #ízení ("sko#í") na první p#íkaz za ukon#ovací závorkou p#íkazu **switch [http://www.google.com/search?q=switch]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=switch].

P#íklad: Vícecestné v#tvení [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/VicecestneVetveni.java]

Vno#ené v#tvení

V#tvení **if - else [http://www.google.com/search?q=if - else]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=if - else] m#žeme samoz#ejm# vno#ovat do sebe:

```
if(podmínka_vn#jší) {
   if(podmínka_vnit#ní_1) {
      ...
   } else {
      ...
   }
} else {
   if(podmínka_vnit#ní_2) {
      ...
   } else {
      ...
   }
}
```

Vno#ené v#tvení (2)

Je možné "šet#it" a neuvád#t složené závorky, v takovém p#ípad# se **else [http://www.google.com/search?q=else]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=else] vztahuje vždy k nejbližšímu neuzav#enému **if [http://www.google.com/search?q=if]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=if], nap#. znovu p#edchozí p#íklad:

```
if(podmínka_vn#jší)
   if(podmínka_vnit#ní1)
      ...
   else // vztahuje se k nejbližšímu if
        // s if (podmínka_vnit#ní_1)
      ...
else // vztahuje se k prvnímu if,
     // protože je v tuto chvíli
     // nejbližší neuzav#ené
   if (podmínka_vnit#ní_2)
      ...
   else // vztahuje se k if (podmínka_vnit#ní_2)
      ...
```

Tak jako u cykl# - tento zp#sob zápisu nelze v žádném p#ípad# doporu#it!!!

P#íklad: Vno#ené v#tvení [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/VnoreneVetveni.java]

#et#zené "if - else if - else"

N#kdy rozvíjíme pouze druhou (negativní) v#tev:

```
if (podmínka1) {
   ...
} else if (podmínka2) {
   ...
} else if (podmínka3) {
   ...
} else {
   ...
```

```
}
```

Neplatí-li podmínka1, testuje se podmínka2, neplatí-li, pak podmínka3...

neplatí-li žádná, provede se p#íkaz za posledním - samostatným - **else [http://www.google.com/search?q=else]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=else].

Op#t je dobré všude psát složené závorky!!!

P#íklad: #et#zené if [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/VicecestneVetveniIf.java]

P#íkazy "break"

Realizuje "násilné" ukon#ení pr#chodu cyklem nebo v#tvením **switch [http://www.google.com/search?q=switch]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=switch]

Syntaxe použití `break` [http://www.google.com/search?q=break] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=break] v **cyklu**:

```
for (int i = 0; i < a.length; i++) {
   if(a[i] == 0) {
      break; // skoci se za konec cyklu
   }
}
if (a[i] == 0) {
    System.out.println("Nasli jsme 0 na pozici "+i);
} else {
    System.out.println("0 v poli neni");
}
```

použití u `switch` [http://www.google.com/search?q=switch] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=switch] jsme již vid#li„Vícecestné v#tvení "switch - case - default"“

P#íkaz "continue"

Používá se v t#le cyklu.

Zp#sobí p#esko#ení zbylé #ásti pr#chodu t#lem cyklu

```
for (int i = 0; i < a.length; i++) {
   if (a[i] == 5)
      continue;
   System.out.println(i);
}
```

Výše uvedený p#íklad vypíše #ísla `1` [http://www.google.com/search?q=1] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=1], `2` [http://www.google.com/search?q=2] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=2], `3` [http://www.google.com/search?q=3] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=3], `4` [http://www.google.com/search?q=4]

http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=4], 6 [http://www.google.com/search?q=6] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=6], 7 [http://www.google.com/search?q=7] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=7], 8 [http://www.google.com/search?q=8] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=8], 9 [http://www.google.com/search?q=9] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=9], nevypíše hodnotu 5 [http://www.google.com/search?q=5] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=5].

P#íklad: #ízení pr#chodu cyklem pomocí "break" a "continue" [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/BreakContinue.java]

"break" a "continue" s náv#štím

Umožní ješt# jemn#jší #ízení pr#chodu vno#enými cykly:

- pomocí náv#ští m#žeme nazna#it, který cyklus má být p#íkazem **break [http://www.google.com/search?q=break]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=break] p#erušen nebo
- t#lo kterého cyklu má být p#esko#eno p#íkazem **continue [http://www.google.com/search?q=continue]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=continue].

P#íklad: Náv#ští [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/rizeni/Navesti.java]

Doporu#ení k p#íkaz#m break a continue

Poznámka

Rad#ji NEPOUŽÍVAT, ale jsou menším zlem než by bylo **goto [http://www.google.com/search?q=goto]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=goto] (kdyby v Jav# existovalo...), protože nep#edávají #ízení dále než za konec struktury (cyklu, v#tvení).

Poznámka

Toto však již neplatí pro **break [http://www.google.com/search?q=break]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=break] a **continue [http://www.google.com/search?q=continue]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=continue] na náv#ští!

Poznámka

Pom#rn# #asto se používá **break [http://www.google.com/search?q=break]** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=break] p#i sekven#ním vyhledávání prvku.