Operátory

Základy programování 1

Martin Kauer

Operátory a jejich vlastnosti

- Základní konstrukce (skoro) každého jazyka
- Z daných operandů vytvoří výsledek, který je možné dále využívat
- Arita – udává počet operandů (vstupů)
- Některé operátory mají i tzv. vedlejší efekt
- Příklady výrazů s operátory:
  - `a + b` (binární sčítání)
  - `a - b` (binární odčítání)
  - `-a` (unární mínus)
  - `a <= b` (binární menší nebo rovno)
  - `a--` (unární dekrementace)
  - `prumer = (a + b) / 2;` (přiřazení, sčítání a dělení v 1 výrazu)

Priorita a asociativita operátorů

- **Priorita** určuje pořadí, ve kterém se operátory vyhodnocují
- Znáte ji vlastně už z matematiky
- `1 + 2 / 3 + 4` = ?
- Vyhodnocování výrazu lze ovlivnit použitím (kulatých) závorek
  `(1 + 2) / (3 + 4)` `1 + (2 / 3 + 4)` `(1 + 2) / 3 + 4`
- **Asociativita** udává směr, ve kterém se vyhodnocují binární operátory se stejnou prioritou
- Zleva nebo zprava
- `1 - 2 - 3 - 4` = ? ? a = b = c ?
- Opět lze ovlivnit závorkami
  `((1 - 2) -3) - 4` `(1 - 2) - (3 - 4)` `1 - (2 - (3 - 4))`

Aritmetické operátory

- Unární operátory `+` a `-`
- Binární operátory `+`, `-`, `*` a `/` s obvyklým významem
- Asociativita zleva
- Příklady:

```
stejneCislo = +cislo;
opacneCislo = -cislo;
delka = vetsi - mensi;
prumer = (prvni + druhy + treti) / 3;
obsah = 2 * pi *polomer * polomer;
```

- Binární operátor `%` pro určení zbytku po celočíselném dělení (modulo)
- Příklad:

```
int delenec = 13, delitel=5, podil, zbytek;
podil = delenec / delitel;
zbytek = delenec % delitel;
```

Aritmetické operátory

- Pozor na typy operandů
- Pokud jsou všechny operandy celočíselné, je i výsledek celočíselný
- Jinak je výsledek desetinné číslo
- Například při dělení toto může ovlivnit i hodnotu výsledku

```
int cislo = 15;
double polovina;
polovina = cislo / 2;
```

- Lze obejít přetypováním operandu nebo desetinným zápisem konstanty

```
polovina = (double)cislo / 2;
polovina = cislo / 2.0;
```

Přiřazení

- Pomocí operátoru `=`
- Zápis ve tvaru *`identifikatorPromenne = JakykoliVyraz`*
- Příklady použití
```
cislo = 15;
druheCislo = 2*cislo;
```
- Vedlejší efekt – do proměnné uvedené vlevo uloží výsledek výrazu vpravo
- Asociativní zprava
```
první = druhy = treti = 42;
(první = (druhy = (treti = 42)));
```
- Další přiřazovací operátory
`+=` `-=` `*=` `/=` `%=` atd.
- Význam
```
cislo += 5;      cislo = cislo+5;
```

Inkrementace a dekrementace

- Aritmetické operátory, které v matematice nemáme
- Mají vedlejší efekt
- Inkrementace (++) zvyšuje hodnotu operandu o 1

```
cislo++;     cislo += 1;     cislo = cislo + 1;
```

- Dekrementace (--) snižuje hodnotu operandu o 1

```
cislo--;     cislo -= 1;     cislo = cislo - 1;
```

- Mohou být v tzv. prefixovém nebo postfixovém tvaru
- Prefixové a postfixové použití se odlišuje výsledkem

```
int vysledek1, vysledek2;
int cislo = 5;
vysledek1 = cislo++;
vysledek2 = ++cislo;
```

- Nedoporučuji používat přiřazení, inkrementaci a dekrementaci ve složitějších výrazech!

```
y = (x++ - 5 + (z = y - 2));
```

Podmínkové operátory

- Operátory pro porovnávání
  `<` `>` `<=` `>=` `==` (rovná se) `!=` (nerovná se)
- Binární operátory
- Výsledkem je logická hodnota – v jazyku C se jedná o celé číslo!
- Příklady:

```
prvni >= druhe + treti
cislo != - cislo
dalsiCislo == cislo + 10
```

- Typické chyby:

```
prvni >= druhe >= treti
10 < cislo <= 20
```

Logické operátory

- Slouží pro konstrukci složitějších podmínek
- K dispozici máme operátory
  - konjunkce (a zároveň) `&&`
  - disjunkce (nebo) `||`
  - negace `!`
- Asociativita zleva
- Příklady:

```
(10 < cislo) && (cislo <= 20)
(cislo <= 10) || (20 < cislo)
!((10 < cislo) && (cislo <= 20))
```

a	b	a && b	a || b	!a
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Líné vyhodnocování

- Při vyhodnocování logických operátorů (konjunkce a disjunkce) se vyhodnocuje pouze část výrazu nezbytná pro určení výsledné pravdivostní hodnoty
- Příklady:

```
(2 < 1) && (10 < cislo/0)
(10 < cislo) && (cislo <= 20)
(10 < 12) || cokoli
(10 < cislo) || (cislo <= 0)
```

- Obzvláště zde tak platí doporučení neschovávat do složitějších podmínek operátory s vedlejším efektem (přiřazení, inkrementace, dekrementace).

Přetypování

- Změna typu výrazu na jiný typ
- Obecně:
  (nový_typ)výraz
- Příklady:

```
int delenec = 5, delitel = 2;
double podil;
podil = (double)delenec / delitel;
```

- Existuje i implicitní přetypování:

```
double cislo = 3.123;
int cela_cast;
cela_cast = cislo;
```

Podmínkový operátor

- Slouží pro větvení uvnitř výrazu
- Při složitějším použití nepřehledný
- Jediný ternární operátor
- Obecně:
  podmínka ? *výraz_splněno* : *výraz_nesplněno*
- Pokud platí *podmínka*, pak se operátor vyhodnotí na *výraz_splněno*, jinak na *výraz_nesplněno*.
- Příklady:

```
int cislo1, cislo2;
int min;
...
min = (cislo1 < cislo2) ? cislo1 : cislo2;
```

Čárka

- Slouží pro vytváření sekvencí
- Používá se např. v řídících částech cyklů, deklaraci více proměnných stejného typu
- Obecně:
  výraz_1, výraz_2
- Vyhodnotí se *výraz_1*, výsledek se zapomene, vyhodnotí se *výraz_2* a jeho výsledek udává výsledek celé sekvence
- Výrazy obvykle mívají nějaký vedlejší efekt
- Příklady:

```
int cis1 = 5, cis2 = 2;
int min, max;
...
(cis1 < cis2) ? (min = cis1, max = cis2) : (min = cis2, max = cis1);
```

Přehled operátorů

Priorita	Operátory	Asociativita	Arita
1	`() [] -> .`		
2	`! ~ ++ -- + - (typ) * & sizeof`		unární
3	`* / %`	zleva doprava	binární
4	`+ -`	zleva doprava	binární
5	`<< >>`	zleva doprava	binární
6	`< > <= >=`	zleva doprava	binární
7	`== !=`	zleva doprava	binární
8	`&`	zleva doprava	binární
9	`^`	zleva doprava	binární
10	`|`	zleva doprava	binární
11	`&&`	zleva doprava	binární
12	`||`	zleva doprava	binární
13	`? :`	zprava doleva	ternární
14	`= += -= *= /= %= >>= <<= &= |= ^=`	zprava doleva	binární
15	`,`	zleva doprava	binární

Cvičení 1

Vytvořte program, který spočítá objem a povrch pravidelného 4bokého hranolu. O potřebné údaje o konkrétním hranolu požádejte uživatele programu a **zkontrolujte zda jsou validní**. Výsledné hodnoty pak vypište uživateli do konzole.

Cvičení 2

Vytvořte program, který po načtení desetinného čísla a přesnosti vypíše zadané číslo po zaokrouhlení na zadanou přesnost. Při řešení použijte dnešní látku. Nesmíte použít žádné zaokrouhlovací funkce z dodatečných knihoven.

Příklady použití:

```
Číslo: 8.2483
Přesnost: 0.01
Zaokrouhleno: 8.25

Číslo: 8.2483
Přesnost: 0.1
Zaokrouhleno: 8.2
```