ZALG - 12. cvičení

Zpracování aritmetického výrazu

Aritmetické výrazy

- Výraz má určený typ a hodnotu
  - 23 typ int, hodnota 23
  - 14+16/2 typ int, hodnota 22
  - y=8 typ int, hodnota
- Přiřazení je výraz a jeho hodnotou je hodnota přiřazená levé straně
- Budeme uvažovat zjednodušené aritmetické výrazy. Tvoří je:
  - číselné konstanty, proměnné
  - binární operátory +,-,*,/
  - unární minus ~
  - závorky (,)
  - faktoriál

Aritmetické operátory

- Pro operandy typu int a double jsou definovány operátory
  - unární operátor změna znaménka −
  - binární sčítání + a odčítání −
  - binární násobení * a dělení /
  - binární zbytek po dělení %
- Jsou-li oba operandy stejného typu je výsledek aritmetické operace stejného typu
- Je-li jeden operand typu int převede se implicitní konverzí na hodnotu typu double a výsledek operace je hodnota typu double

- Unární operátory ++ a −− mění hodnotu svého operandu. Operand musí být l-hodnota, tj. výraz, který má adresu, kde je uložena hodnota výrazu (např. proměnná)
  - lze zapsat prefixově např. ++x nebo −−x
  - nebo postfixově např. x++ nebo x−−
- v obou případech se však liší výsledná hodnota výrazu!

Priorita operátorů

Precedence	Operator	Description	Associativity
1	::	Scope resolution	Left-to-right →
2	a++ a-- *type*() *type*{} a() a[] . ->	Suffix/postfix increment and decrement Functional cast Function call Subscript Member access	
3	++a --a +a -a ! ~ (*type*) *a &a sizeof co_await new new[] delete delete[]	Prefix increment and decrement Unary plus and minus Logical NOT and bitwise NOT C-style cast Indirection (dereference) Address-of Size-of[note 1] await-expression (C++20) Dynamic memory allocation Dynamic memory deallocation	Right-to-left ←
4	.* ->*	Pointer-to-member	Left-to-right →
5	a*b a/b a%b	Multiplication, division, and remainder	
6	a+b a-b	Addition and subtraction	
7	<< >>	Bitwise left shift and right shift	
8	<=>	Three-way comparison operator (since C++20)	
9	< <= > >=	For relational operators < and ≤ and > and ≥ respectively	
10	== !=	For equality operators = and ≠ respectively	
11	a&b	Bitwise AND	
12	^	Bitwise XOR (exclusive or)	
13	|	Bitwise OR (inclusive or)	
14	&&	Logical AND	
15	||	Logical OR	
16	a?b:c throw co_yield = += -= *= /= %= <<= >>= &= ^= |=	Ternary conditional[note 2] throw operator yield-expression (C++20) Direct assignment (provided by default for C++ classes) Compound assignment by sum and difference Compound assignment by product, quotient, and remainder Compound assignment by bitwise left shift and right shift Compound assignment by bitwise AND, XOR, and OR	Right-to-left ←
17	,	Comma	Left-to-right →

```
int a = 1;
int b = 4;
b += a -= 6;
```

- Oba operátory += a -= mají stejnou prioritu, ale jelikož asociativita je zprava doleva, tak se první vyhodnotí a-= 6 a až pak b+= a
- a tedy bude -5 a b bude -1

Reprezentace aritmetických výrazů

- textový řetězec
  - v infixové, postfixové nebo prefixové formě
- Binární strom
  - vnitřní vrcholy = operátory
  - listy = číselné konstanty nebo proměnné
  - podstrom = podvýraz

Infixová notace

- pro binární operátor nejprve levý operand, pak operátor a pak pravý operand, např. x + y
- pro unární operátor prvně operátor, pak operand, např ~x
- příklad: ((4-x)+y)/~(x*2)
- potřebujeme závorky
- pro člověka nejpřehlednější notace

Prefixová notace

- pro binární operátor prvně operátor, pak levý operand, pak pravý operand, např. +xy
- pro unární operátor prvně operátor, pak operand, např. ~x
- příklad: / + − 4 x y ~ ∗ x 2, stejné jako (/(+(− 4 x) y)(~ (∗ x 2)))
  - ((4-x)+y)/~(x*2)

Postfixová notace

- Pro binární operátor nejprve levý operand, pak pravý operand a pak operátor, např. AB+
- Pro unární operátor prvně operand a pak operátor, např. x~
- Příklad: 4 x - y + x 2 * ~ / , totéž jako ((4 x −) y +)((x 2 ∗) ~)/
  - ((4-x)+y)/~(x*2)

Typické operace s aritmetickými výrazy v programu

- vyhodnocení výrazu
- aritmetická kalkulačka

Zpracování prefixové notace přímo

Použijeme: zásobník na tokeny (čísla nebo operátory)

Bereme tokeny jeden po druhém:

- operátor → vložíme ho na zásobník
- číslo, proměnná →
  - proměnnou vyčíslíme → číslo v
  - **Cyklus:** podíváme se na vrchol zásobníku:
  1. pokud je zásobník prázdný nebo je na vrchu zásobníku binární operátor, číslo v vložíme na zásobník a ukončíme cyklus
  2. pokud je na vrchu zásobníku unární operátor →
     - vyjmeme ho, spočítáme výsledek operace, označíme ho jako v a opakujeme
  3. jinak (na vrchu zásobníku je číslo) →
     - vyjmeme ze zásobníku nejprve číslo (levý operand) a pak binární operátor, v bude pravý operand
     - spočítáme výsledek operace, výsledek označíme jako v a opakujeme

Zpracování postfixové notace přímo

Použijeme: zásobník na čísla

- Bereme tokeny jeden po druhém:
  - číslo → vložíme ho na zásobník
  - proměnná → vyčíslíme ji a číselnou hodnotu vložíme na zásobník
  - binární operátor →
    1. na vrchu zásobníku jsou jeho operandy (na vrchu pravý, pod ním levý), vyjmeme je
    2. spočteme výsledek operace a vložíme ho na zásobník
  - unární operátor →
    1. na vrchu zásobníku je jeho operand, vyjmeme ho
    2. spočteme výsledek operace a vložíme ho na zásobník
- na konci bude na zásobníku jedinný prvek = výsledek