

Aufgabenblatt 3

Besprechungstermin: 03./04.05.2005

Aufgabe 1 (nach Löhr)

Hier sind einige Variable, deren Typ aus ihrem Namen hervorgeht, und ihre aktuellen Werte:

boolean1	char1	char2	short1	int1	long1	float1	double1
true	'a'	'b'	1	4711	12345L	0.0f	3.14

Analysieren Sie die unten angegebenen Ausdrücke:

1. Falls Syntaxfehler, bitte erläutern, sonst Syntaxbaum gemäß vereinfachter Java-Grammatik angeben und
2. falls nicht typkorrekt, bitte erläutern;
3. sonst Typ und Wert angeben.

- a) (1 P.) `short1 < int1 < long1`
- b) (2 P.) `char1 == char2 && char1`
- c) (3 P.) `++ int1 - int1 - - + + 1`
- d) (3 P.) `-(-1 - float1*char1)/2`
- e) (2 P.) `+ +3.7 - short1`
- f) (3 P.) `(double1) + + 3.0 - short1`

Aufgabe 2 Newtonsches Iterationsverfahren

Sei $f : \mathbb{R} \rightarrow \mathbb{R}$ eine stetige Funktion.

Gesucht: Nullstelle von f . Starte mit x_0 . Iteriere

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad i = 0, 1, \dots$$

Anforderung: f' muss im betrachteten Intervall von 0 verschieden sein! Berechnen Sie das Newtonsche Iterationsverfahren für

$$\begin{aligned} f(x) &= x^2 - 3 \\ f'(x) &= 2x \end{aligned}$$

Hinweis: Das Verfahren konvergiert für $x_0 > 1$. Programmieren Sie dieses Verfahren in Java.

Aufgabe 3 In der Kombinatorik heißt eine Teilmenge der Mächtigkeit k (von einer Grundmenge der Mächtigkeit n) eine Kombination k -ter Klasse. Beispiel: Grundmenge $G = \{a, b, c, d\}$. Kombinationen 2-ter Klasse von G sind: $\{a, b\}$, $\{a, c\}$, $\{a, d\}$, $\{b, c\}$, $\{b, d\}$, $\{c, d\}$.

Für die Anzahl C_n^k der Kombinationen k -ter Klasse gilt:

$$C_n^k = \binom{n}{k} = \frac{n!}{(n-k)! \cdot k!}$$

Schreibe eine Java-Funktion zur Berechnung von C_n^k .

Java-Operatoren mit Prioritäten in absteigender Ordnung und Assoziativitäten (Auszug)

einstellige Operatoren: +, -, ++, --, !, CAST (rechtsassoziativ)

(Punktrechnung)

zweitst. arithm. Operatoren: *, /, % (linksassoziativ)

(Strichrechnung)

zweitst. arithm. Operatoren: +, -

arithm. Relationen

(Vergleichsoperatoren): <, >, <=, >=

Gleichheitsrel.: ==, !=

logisches Und: &&

logisches Oder: ||

Vereinfachte Java-Grammatik (Teil I)

Java-Ausdrücke (Auszug)

PrimExp: Lit | Var | IncDecExp | (Exp) | MethodInvoc

Var: Id {[Exp]} Bsp. a[4][3][8]

Id: [Id.] Name Bsp. Math.sqrt, System.out.print

IncDecExp: Var IncDec | IncDec Var Bsp. b++, -c

IncDec: ++|--

Exp: UnaryExp | BinExp | CondExp | AssignExp | CreationExp

UnaryExp: PrimExp | UnaryOp UnaryExp

UnaryOp: + | - | ! | ... | Cast

Cast: (PrimType)

BinExp: Exp BinOp Exp

BinOp: + | - | * | / | && | || | : | < | > | != | ==

Cond: Exp ? Exp : Exp

AssignExp: Var AssignOp Exp

AssignOp: = | += | -= | *= | /= | ...

Method Invocation: Method ([ActualArgs])

Method: Id

ActualArgs: Exp {,Exp}

CreationExp: new ClassId([ActualArgs]) | ArrayCreationExp

ArrayCreationExp: new ID [Exp] {[Exp]} {[]}

PrimType: Name Beispiel: int, long, double