

## 6. Изрази, операции (прости типове)

Лекционен курс “Програмиране на Java”  
проф. д-р Станимир Стоянов

# Структура на лекцията


---

- ▶ Въвеждащ пример и проблеми
- ▶ Синтаксис на изразите
- ▶ Странични ефекти
- ▶ Преглед на операциите
- ▶ Претоварване
- ▶ Автоматично и явно преобразуване на типове
- ▶ Приоритети
- ▶ Типизиране

# Пример: тест за високосна година

---

```
((year % 4) == 0)
&& ((year % 100) != 0)
|| ((year % 400) == 0)
```



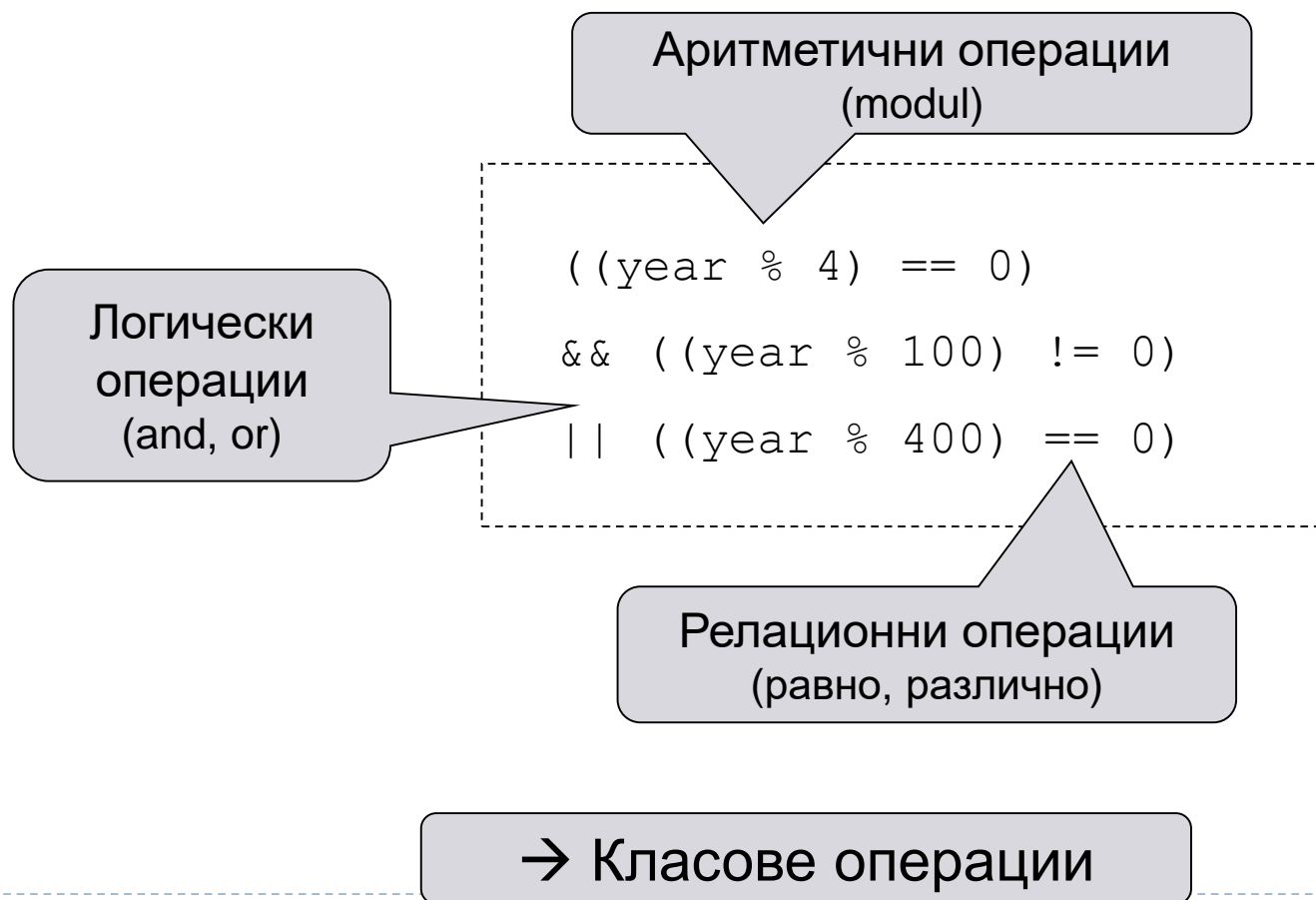
Какви проблеми  
трябва да се  
изяснят ?

Високосна година:

- Всички години, които се делят на **4 са** високосни
- Изключение 1: всички, които се делят на **100 не са** високосни
- Изключение 2: всички, които се делят на **400 са** високосни

# Проблеми (1)

- ▶ Притежават ли операндите (напр. 'year') коректните типове ?



# Проблеми (2)

---

## ► Коректно свързване на подизрази?

```
((year % 4) == 0)
&& ((year % 100) != 0)
|| ((year % 400) == 0)
```

1.

2.

3.

- Приоритети на операции
- Повече скоби?
- По-малко скоби?

# Проблеми (3)

- Ако подизразите са определили вече общата стойност: да бъде ли оценен остатъчният израз?
- Практически съществени: ефективност, определяемост на остатъка

```
((year % 4) == 0)
&& ((year % 100) != 0)
|| ((year % 400) == 0)
```

- `((year % 4) == 0)` е невярно  
→ ...`&&`... е неверен
- `((year % 4) == 0) && ((year % 100) != 0)` е вярна  
→ ...`||`... е вярна

Решение в Java?

Java:	<code>&amp;&amp;</code> , <code>  </code>	със съкратено оценяване
	<code>&amp;</code> , <code> </code>	с пълно оценяване

# Синтаксис на изразите в Java: странични ефекти

---

- Изрази в Java
  - Литерали (числа, символни низове, логически стойности, ...)
  - Променлива (+ параметър)
  - Извиквания на методи
  - Съставни изрази с аритметични, релационни, логически и побитови операции
  - Присвоявания (!!!)
  - ...

Критика (към Java, C):

Няма ясно разделение между изрази и оператори !

→ Причина за грешки !

# Пример

```
class Assignment {  
    public static void  
        main (String[] args) {  
        int x = 2, y = 5, z = 1;  
        x = (z++ - (y = y + x));  
    }  
}
```

Крайни стойности:	x =	-6
	y =	7
	z =	2

Избягване : странични ефекти в изрази!  
(заедно с изчисляването на стойност:  
промяна на стойностите на променливите  
- причина за грешки)



# Странични ефекти: $z++$ и $++z$

$x = (z++ - (y = y + x));$

Стойност =  $z$

След това:  $z = z + 1$

Резултат:  $x = -6, y=7, z=2$

1-7

$x = (++z - (y = y + x));$

първо:  $z = z + 1$

стойност =  $z + 1$

резултат:  $x = -5, y=7, z=2$

2-7

# Странични ефекти: принципно избягвани

Но: съществуват смислени приложни случаи !

Задача на един израз:

- Да: изчисляване на стойности
- Не: промяна на стойности

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

```
y = y + x;  
x = z - y;  
z++;
```

Изрично задаване на стойностите на променливите, които трябва да се променят (последователност!)

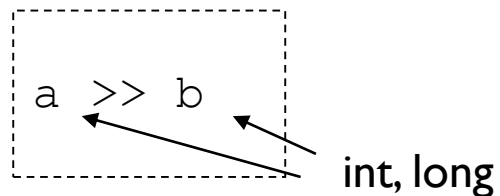
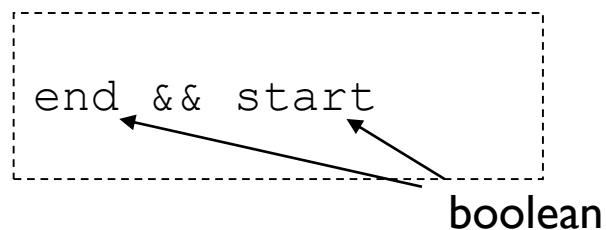
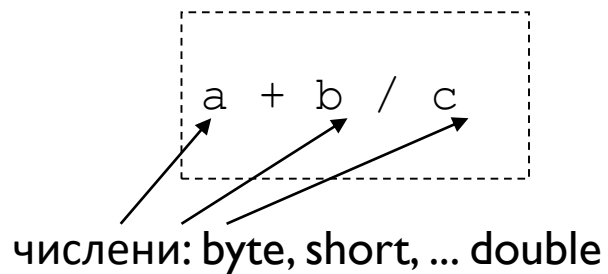
```
x = (++z - (y = y + x));
```

```
++z;  
y = y + x;  
x = z - y;
```

По-добре:  
z = z + 1;

# Операции: свързани с типове

---



# Обобщение: примитивни типове данни

---

Тип	Дължина (Byte)	Област на стойности
boolean	1	true, false
char	2	Всички Unicode символи
byte	1	$-2^7 \dots 2^7 - 1$
short	2	$-2^{15} \dots 2^{15} - 1$
int	4	$-2^{31} \dots 2^{31} - 1$
long	8	$-2^{63} \dots 2^{63} - 1$
float	4	+ / - $3.4028234738 * 10^{38}$
double	8	+ / - $1.797693134862315703 * 10^{308}$

# Аритметични операции

- Числови операнди, тип на резултат: числов
- Конвертиране в обхващащия тип при операнди с различни типове

+	Положителен	$n$
-	Отрицателен	$-n$
+	Сума	$a + b$
-	Разлика	$a - b$
*	Умножение	$a * b$
/	Деление	$a / b$
%	Остатък от деление	$a \text{ modul } b$
++	Предварително увеличение	++a става $a+1$ , увеличава $a$ с 1
++	Последващо увеличение	$a++$ става $a$ , увеличава $a$ с 1
--	Предварително намаление	--a става $a-1$ , намалява $a$ с 1
--	Последващо намаление	$a--$ става $a$ , намалява $a$ с 1

Забележка: % също за числа с плаваща запетая !

# Операции: претоварване (Overloading)

$x = a + b$

+: int	x	int	→	int
+: float	x	float	→	float
+: double	x	...		

'+': различни операции

математически

хардуерни

"Overloading": еднакво име за  
различни операции

# Операции: изискват определено позициониране на операндите

---

Infix–операция:

<code>a + b</code>	<code>a &lt;&lt; b</code>
--------------------	---------------------------

Postfix–операция:

<code>a ++</code>
-------------------

Prefix–операция:

<code>++ a</code>	<code>! true</code>	<code>~ a</code>	<code>^ a</code>
-------------------	---------------------	------------------	------------------

Други (3-позиционни):

<code>a &gt; b ? a : b</code>
-------------------------------

3 – позиционна Infix-операция

# Релационни операции

- ▶ За числени операнди (също смесени)
- ▶ (Не) Равенство също за типове на обекти (сравняване на адреси)
- ▶ Тип на резултата: `boolean`

<code>==</code>	равно	<code>a == b</code>
<code>!=</code>	неравно	<code>a != b</code>
<code>&lt;</code>	по-малко	<code>a &lt; b</code>
<code>&lt;=</code>	по-малко равно	<code>a ≤ b</code>
<code>&gt;</code>	по-голямо	<code>a &gt; b</code>
<code>&gt;=</code>	по-голямо равно	<code>a ≥ b</code>



# Тип 'boolean'

---

## ► Стойности: true, false

```
boolean ready;  
ready = false;  
ready = year > 1999;
```

Операции:

!  
&  
|

отговарят на тези в логиката

Отрицание

Конюнкция ('and')

Дизюнкция ('or')

# Логически операции

- ▶ За логически типове (boolean)
- ▶ Тип на резултата: boolean
- ▶ „Частично оценяване”: следващите, в дясно стоящи подизрази не се оценяват, ако стойността вече е известна
  - ▶ напр.  $a \ \&\& \ b \rightarrow \text{false}$ , ако  $a$  е вече false  
 $\rightarrow b$  не се оценява

!	отрицание	$\sim a$
&&	AND с частично оценяване	$a \wedge b$
	OR с частично оценяване	$a \vee b$
&	AND с пълно оценяване	$a \wedge b$
	OR с пълно оценяване	$a \vee b$
^	EXCLUSIVE-OR (или ... или)	$a \otimes b$

# Логически операции

## ► Пример

```
boolean A = true;  
boolean B = false;  
int x = 0;  
int y = 0;
```

```
boolean C = A || (++x < 0)
```

```
boolean D = B & (++y < 0)
```

C = true  
x = 0

D = false  
y = 1

# Побитови операции

► Манипулации с битове за `int` съотв. `long`

<code>~</code>	Единичен комплимент	<code>~a</code> : инвертиране битовете на <code>a</code>
<code> </code>	Побитов OR	<code>a   b</code> : побитов $a_i \vee b_i$
<code>&amp;</code>	Побитов AND	<code>a &amp; b</code> : побитов $a_i \wedge b_i$
<code>^</code>	Побитов XOR	<code>a ^ b</code> : побитов $a_i \otimes b_i$
<code>&gt;&gt;</code>	Отместване на дясно със знак	<code>a &gt;&gt; b</code> : битовете на <code>a</code> се отместват с <code>b</code> позиции надясно, знак като при <code>a</code>
<code>&gt;&gt;&gt;</code>	Отместване на дясно без знак	<code>a &gt;&gt;&gt; b</code> : битовете на <code>a</code> се отместват с <code>b</code> позиции надясно, попълване с 0, припокриване на знак (0)
<code>&lt;&lt;</code>	Отместване на ляво	<code>a &lt;&lt; b</code> : битовете на <code>a</code> се отместват с <code>b</code> позиции наляво, попълване с 0, знак като при <code>a</code>

При това: `b modul 32` (`int`) съотв. `64` (`long`)

<code>a &lt;&lt; 1</code>	Умножение с 2
<code>a &lt;&lt; 2</code>	Умножение с 4
<code>a &lt;&lt; n</code>	Умножение с $2^n$

# Оператори за присвояване (1)

- Присвояванията са изрази във формата  
**ЛяваСтрана оператор\_за\_присвояване Израз**
  - напр.: **x = x + y**
- ЛяваСтрана означава памет (в общия случай променлива)

Разлика за "x" в **x = x + y**;

Адрес

Стойност

- Тип на присвояването = Тип на ЛяватаСтрана
- Стойност на присвояването = стойност на израза

Смесване "=" / "==",  
(само) при boolean:

```
boolean x = true, y = false;  
System.out.println( y == x );  
System.out.println( y = x );
```

# Оператори за присвояване (2)

EBNF:

Оператор за присвояване ::= = | += | \*= | -= | /=  
| %= | &= | ^=  
| <<= | >>= | >>>=.

Комбинация на "=" с аритметични и побитови операции във формата  
"Операция=" за операциите

+ | \* | - | / | % | & | ^ | << | >> | >>>

Ефект за  $x$  операция =  $y$  както  
 $x = x$  операция  $y$

$x += 100$ както	$x = x + 100$
$x <<= 2$ както	$x = x << 2$

# Други оператори (1)

Въпросителен знак оператор:

ЛогичекиИзраз ? израз1 : израз2

доставя:

израз1, ако ЛогичекиИзраз == true

израз2, ако ЛогичекиИзраз == false

напр. `max = (x > y ? x : y);`

Свързване на низове:

За символни низове (Strings): `string1 + string2`

```
x = 1; y = 2;  
System.out.println(x + y);  
System.out.println(x + " " + y);
```

→ Изход: "3"

→ Изход: "12"

# Други оператори (2)

---

new – оператор:

Създаване на инстанции:

```
new Type ([ArgumentList])
```

С конструктура *Type*(*[ArgumentList]*)

За инициализиране на една инстанция

instanceof – оператор:

```
InstanceName instanceof ClassName
```

Тип на резултата boolean:

true, ако InstanceName е инстанция на класа ClassName съотв. означава един подклас на ClassName



# Преобразуване на типове: Cast-Оператор

Присвояване

```
x = y; x = a + b;
```

Израз

автоматично

Проблем:  
Операнди от  
различен тип

byte

short

char

int

long

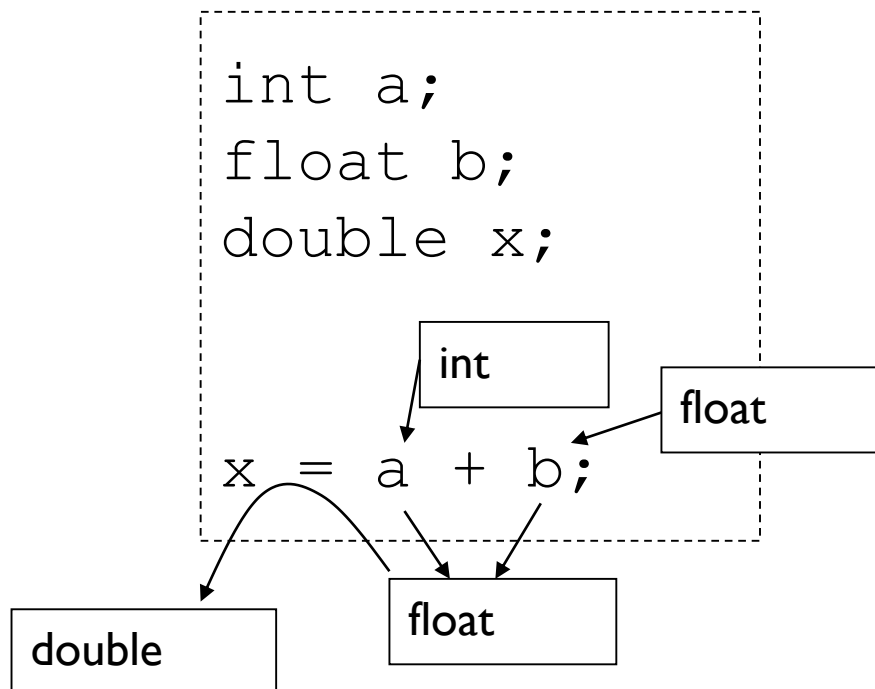
float

double

cast

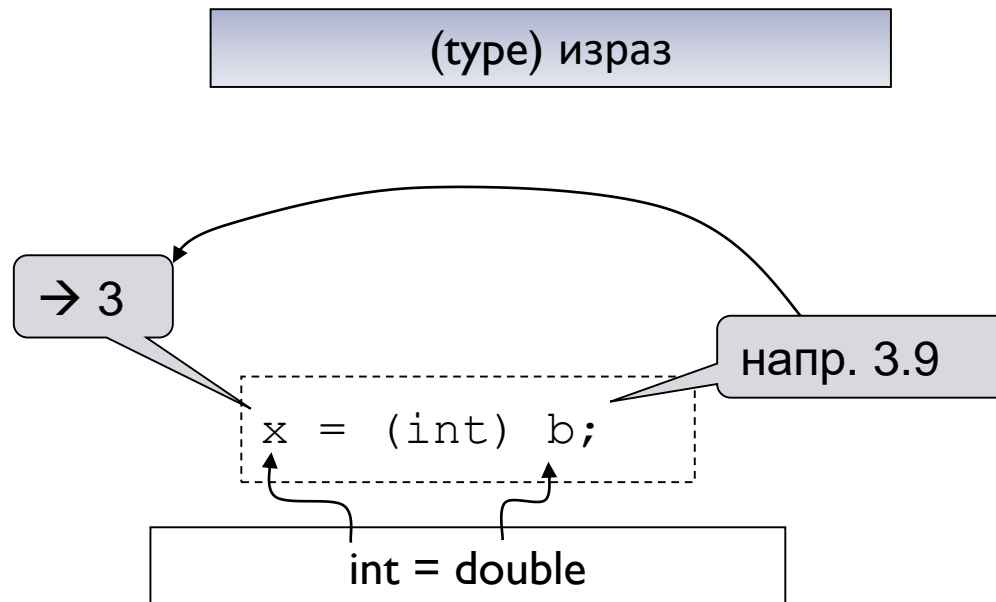
# Автоматично преобразуване на типове

- ▶ Автоматично преобразуване в “по-големия” тип: няма загуба на информация



# Явно преобразуване на типове: Type-Cast-оператор

- Преобразува изрази в нов израз от тип *type* - ев. със загуба на информация



# Изход на Unicode-символи 0020 - 00FF

```
// Windows: Output of Unicode-symbols in Console-Window Fenster requires Codepage 1252.
// Activate by Command: 'chcp 1252' in DOS-Window.
// Additionally in DOS-Window 'Lucida Console' selected.

import java.awt.*;

class Unicode {

    public static void main (String [] args) {

        for (char code = '\u0020' ; code <= '\u00FF' ; code = (char)(code + 1)) {
            String fill = "";
            if (code < '\u0064')
                fill = " " ;                                // 2-position decimalnumber right bundled

            if (code >= '\u007F' && code <= '\u009F')
                // '?' for unprinted symbols
                System.out.print
                    (fill + Integer.toString(code) + " " + '?' + " ");
            else
                System.out.print
                    (fill + Integer.toString(code) + " " + Character.toString(code) + " ");

            if (code%10 == 0)
                System.out.println();
        }
        System.out.println();
    }
}
```

0000 .. 001F:  
Управляващи символи  
(без визуален образ)

# Изход на програмата

32	33	!	34	"	35	#	36	\$	37	%	38	&	39	'	40	(			
41	)	42	*	43	+	44	,	45	-	46	.	47	/	48	0	49	1	50	2
51	3	52	4	53	5	54	6	55	7	56	8	57	9	58	:	59	;	60	<
61	=	62	>	63	?	64	@	65	A	66	B	67	C	68	D	69	E	70	F
71	G	72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O	80	P
81	Q	82	R	83	S	84	T	85	U	86	V	87	W	88	X	89	Y	90	Z
91	[	92	\	93	]	94	^	95	_	96	`	97	a	98	b	99	c	100	d
101	e	102	f	103	g	104	h	105	i	106	j	107	k	108	l	109	m	110	n
111	o	112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w	120	x
121	y	122	z	123	{	124		125	}	126	~	127	?	128	?	129	?	130	?
131	?	132	?	133	?	134	?	135	?	136	?	137	?	138	?	139	?	140	?
141	?	142	?	143	?	144	?	145	?	146	?	147	?	148	?	149	?	150	?
151	?	152	?	153	?	154	?	155	?	156	?	157	?	158	?	159	?	160	
161	i	162	¢	163	£	164	¤	165	¥	166	¦	167	§	168	¨	169	©	170	ª
171	«	172	¬	173	-	174	®	175	¯	176	°	177	±	178	²	179	³	180	´
181	µ	182	¶	183	·	184		185	¸	186	¹	187	º	188	¼	189	½	190	¾
191	¿	192	À	193	Á	194	Â	195	Ã	196	Ä	197	Å	198	Æ	199	Ç	200	È
201	É	202	Ê	203	Ë	204	Ì	205	Í	206	Î	207	Ï	208	Ð	209	Ñ	210	Ò
211	Ó	212	Ô	213	Õ	214	Ö	215	×	216	Ø	217	Ù	218	Ú	219	Û	220	Ü
221	Ý	222	Þ	223	ß	224	à	225	á	226	â	227	ã	228	ä	229	å	230	æ
231	ç	232	è	233	é	234	ê	235	ë	236	ì	237	í	238	î	239	ï	240	ð
241	ñ	242	ò	243	ó	244	ô	245	õ	246	ö	247	÷	248	ø	249	ù	250	ú
251	û	252	ü	253	ý	254	þ	255	ÿ										

# Изход на Unicode-символи 0020 - 00FF

```
for (char code = '\u0020' ; code <= '\u00FF' ;  
      code = (char) (code + 1))  
{  
  
    if (code >= '\u007F' && code <= '\u009F')  
        // '?' for unprinted symbols  
        System.out.print(fill + Integer.toString(code)  
                          + " " + '?' + "  
    else  
        System.out.print(fill + Integer.toString(code)  
                          + " " + Character.toString(code) + "  
");  
  
    if (code%10 == 0)  
        System.out.println();  
}  
}
```

Предаване на параметри:char → int

```
char a = 'a';  
char a = '\u0061';
```

# Преглед на операциите: приоритети

Операция	Типизиране	Асоциативност	Означение
Група 1			
++	N	R	Увеличаване
--	N	R	Намаляване
+	N	R	Унарен плюс
-	N	R	Унарен минус
~	I	R	Onecomplement
!	L	R	Отрицание
(type)	A	R	Преобразуване
Група 2			
*	N,N	L	Умножение
/	N,N	L	Деление
%	N,N	L	Остатък
Група 3			
+	N,N	L	Събиране
-	N,N	L	Изваждане
+	S,S	L	Конкатенация на низове
Група 4			
<<	I,I	L	Изместване на ляво
>>	I,I	L	Изместване на дясно
>>>	I,I	L	Изместване на дясно с допълване с 0

Операция	Типизиране	Асоциативност	Означение
Група 5			
<	N,N	L	По-малко
<=	N,N	L	По-малко или равно
>	N,N	L	По-голямо
>=	N,N	L	По-голямо или равно
instanceof	R,R	L	Инстанция на клас
Група 6			
==	P,P	L	Равенство
!=	P,P	L	Неравенство
==	R,R	L	Равенство по реф.
!=	R,R	L	Неравенство по реф.
Група 7			
&	I,I	L	Побитов И
&	L,L	L	Логически И
Група 8			
^	I,I	L	Побитов XOR
^	L,L	L	Логически XOR
Група 9			
	I,I	L	Побитов ИЛИ
	L,L	L	Логически ИЛИ
Група 10			
&&	L,L	L	Логически И, частично



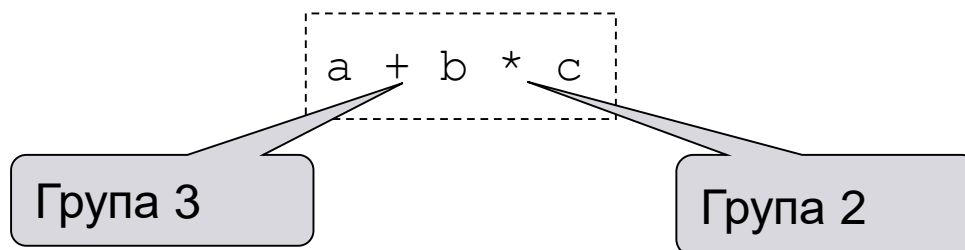
Операция	Типизиране	Асоциативност	Означение
Група 11			
	L,L	L	Логическо ИЛИ, част.
Група 12			
? :	L,A,A	R	Условно изчисление
Група 13			
=	V,A	R	Присвояване
+=	V,N	R	Addition assignment
-=	V,N	R	Extraction assignment
*=	V,N	R	Multiplication ssignment
/=	V,N	R	Division assignment
%=	V,N	R	Rest value assignment
&=	V,N	R	Bitw.-AND-Assignment
	V,L	R	Log.-AND-Assignment
=	V,N	R	Bitw.-OR-Assignment
	V,L	R	Log.-OR-Assignment
^=	V,N	R	Bitw.-XOR-Assignment
	V,L	R	Log.-XOR-Assignment
<<=	V,I	R	Left-Write-Assignment
>>=	V,I	R	Right-Write-Assignment
>>>=	V,I	R	Right-Write-Assignment with Nullexpansion

# Правила за приоритети (1):

---

Приоритет = Сила на свързване

- Таблица: по-ниска група с по-висок приоритет



→ така:

$a + (b * c)$

Сравнение:  $(\text{int}) 3.1 + 3.9$  и  $(\text{int}) (3.1 + 3.9)$

## Правила за приоритети (2):

- Вътре в една група: приоритет по асоциативност
  - Напр. лява асоциативност L:

$a - b - c$

както

$(a - b) - c$

$a - (b - c)$  би било грешно

$a == b == c$

както

$(a == b) == c$

true/false

Тип на a, b, c:  
int възможен ?

# Правила за приоритети (3):

---

- Вътре в една група: приоритет по асоциативност
  - напр. дясна асоциативност R:

$a = b = c$

както

$a = ( b = c )$

след това: a, b, c със стойност на c

избягване ?!

# Типизиране

---

- ( **N** ) Числен
- ( **I** ) Интегрален
- ( **L** ) Логически
- ( **S** ) Низ
- ( **R** ) Референция
- ( **P** ) Примитивен
- ( **A** ) Всички типове

(Типове на операндите)

Лява страна на едно присвояване: променлива V

**Благодаря за вниманието!**

Край лекция 6. “Изрази, операции  
(прости типове)”