ИЗРАЗИ

ЛЕКЦИОНЕН КУРС "ПРОГРАМИРАНЕ НА JAVA"

Distributed @learning center

СТРУКТУРА НА ЛЕКЦИЯТА

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

ИЗРАЗИ

- Могат да се образуват изрази, които съдържат:
  - Литерали
  - Променливи
  - Символни константи
  - Операции
- Съществуват правила за приоритети на операциите
  - Аналогично като в математиката
- Изразите могат да съдържат извиквания на методи
  - Могат да се появяват в дясната част на операторите за присвояване
    - С актуални параметри, които могат да бъдат изрази
  - Подобно на изразите те произвеждат стойности когато се обработват

ИЗРАЗИ

1 Валидно ли е присвояването в израза от примера?

- Java винаги знае какъв тип на стойността имат изразите
  - **Много внимателно**: те да отговарят на типовете на променливите, в които се записват стойностите

```
int i;
double x;

i = 10.3*x;
```

Присвояването невалидно

ИЗРАЗИ

- Съществуват изключения
  - Например: x = i + 10;
- Внимание с конвертирането на изрази, съдържащи различни типове данни
  - Възможно е автоматично конвертиране
    - Когато не се губи информация
  - При загуба на информация
    - Автоматично конвертиране не се извършва
- Можем да предизвикаме конвертиране
  - Оператор cast
  - Напр.: i = (int) (10.3 * x);

ПРИМЕР

1. Какво прави изразът?

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

Високосна година:
- Всички години, които се делят на 4, но не се делят на 100
- Всички години, които се делят на 400

2. Какви проблеми трябва да се изяснят?

ПРОБЛЕМ 1

Притежават ли операндите (напр. ‘year') коректните типове и свързаните с тях операциии ?

1 Какви операции?

- Аритметични операции (modulo)
- Логически операции (and, or)
- Релационни операции (равно, различно)

Класове операции

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

ПРОБЛЕМ 2

Коректно ли са свързани подизразите?

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

Приоритети на операции
Повече скоби?
По-малко скоби?

ПРОБЛЕМ 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 грешна → . . . && . . . е грешна
- ((year % 4) == 0) && ((year % 100) != 0) е вярна → . . . || . . . е вярна

Решение в Java?

Java: &&, || със съкратено оценяване
& | с пълно оценяване

СИНТАКСИС НА ИЗРАЗИТЕ В JAVA: СТРАНИЧНИ ЕФЕКТИ

- Изрази в Java
  - Литерали (числа, символни низове, логически стойности, ...)
  - Поменлива (+ 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

1 Крайни стойности?

```
class Assignment {

   public static void
                main (String[] args) {

        int x = 2, y = 5, z = 1;

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

   }

}
```

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++;
```

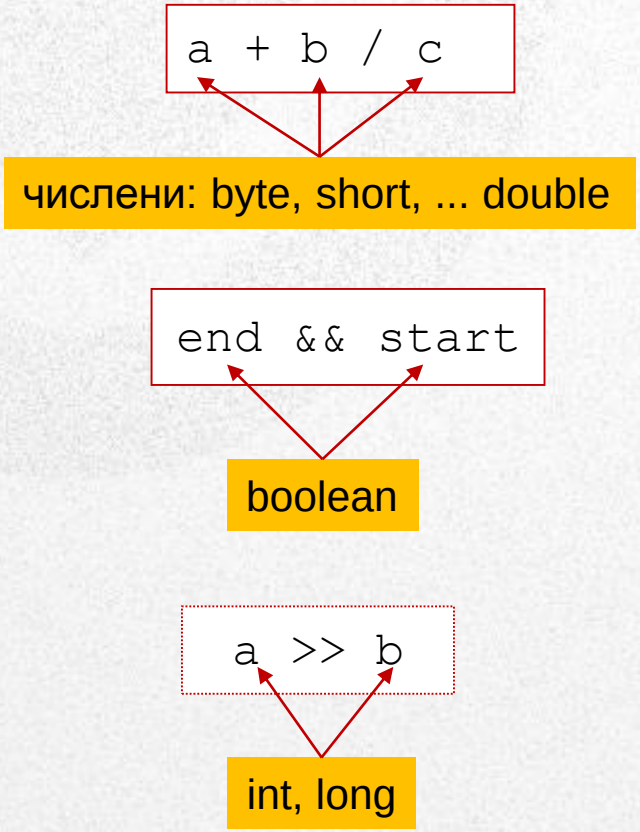
Експлицитно задаване на стойностите на променливите, които трябва да се променят (последователност!)

По-добре:
z = z +1;

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

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

ОПЕРАЦИИ: СВЪРЗАНИ С ТИПОВЕ



ОБОБЩЕНИЕ: ПРИМИТИВНИ ТИПОВЕ ДАННИ

Тип	Дължина (Byte)	Област на стойности
boolean	1	true, false
char	2	All Unicode-Zeichen
byte	1	$-2^7 \ldots 2^7 - 1$
short	2	$-2^{15} \ldots 2^{15} - 1$
int	4	$-2^{31} \ldots 2^{31} - 1$
long	8	$-2^{63} \ldots 2^{63} - 1$
float	4	$+ / - 3.4028234738 * 10^{38}$
double	8	$+ / - 1.79769313486231570 3 * 10^{308}$

АРИТМЕТИЧНИ ОПЕРАЦИИ

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

+	Positive	n
-	Negative	–n
+	Sum	a + b
-	Difference	a – b
*	Product	a * b
/	Quotient	a / b
%	Restvalue	a modulo b
++	Preincrement	++a става a+1, увеличава a с 1
++	Postincrement	a++ става a, увеличава a с 1
--	Predecrement	--a става a-1, намалява a с 1
--	Postdecrement	--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	...		

"Overloading": еднакво име за различни операции

ОПЕРАЦИИ: ИЗИСКВАТ ОПРЕДЕЛЕНО ПОЗИЦИОНИРАНЕ НА ОПЕРАНДИТЕ

Infix–операция:

```
a + b
```

```
a << b
```

Postfix–операция:

```
a ++
```

Prefix–операция:

```
++ a
```

```
! true
```

```
~ a
```

```
^ a
```

Други (3-позиционни):

```
a > b ? a : b
```

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

==	равно	a == b
!=	неравно	a != b
<	по-малко	a < b
<=	по-малко равно	a ≤ b
>	по-голямо	a > b
>=	по-голямо равно	a ≥ b

ТИП 'BOOLEAN'

- Стойности: true, false

```
boolean ready, ...
ready = false;
ready = jahr > 1999;
```

Операции:	отговарят на тези в логиката
!	Отрицание
&	Конюнкция (‘and')
|	Дизюнкция (‘or')

ЛОГИЧЕСКИ ОПЕРАЦИИ

- За логически типове (boolean)
- Тип на реултата: boolean
- „Частично оценяване": следващите, в дясно стоящи подизрази не се оценяват, ако стойността вече е известна
  - напр. a && b → false, ако a е вече грешен
    → b не се оценява

!	отрицание	~a
&&	AND с частично оценяване	a ∧ b
||	OR с частично оценяване	a ∨ b
&	AND с пълно оценяване	a ∧ b
|	OR с пълно оценяване	a ∨ b
^	EXCLUSIVE-OR (или … или)	a ⊗ b

ПОБИТОВИ ОПЕРАЦИИ

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

~	Единичен комплимент	~a: инвертиране битовете на a
|	Побитов OR	a | b: побитов $a_i \vee b_i$
&	Побитов AND	a & b: побитов $a_i \wedge b_i$
^	Побитов XOR	a^b: побитов $a_i \otimes b_i$
>>	Писане дясно със знак	a >> b: битовете на a с b позиции надясно, знак като при a
>>>	Писане дясно без знак	a >>> b: битовете на a с b позиции надясно, попълване с 0, надписване на знак (0)
<<	Писане ляво	a << b: битовете на a с b позиции наляво, попълване с 0, знак като при a

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

a << 1	Умножение с 2
a << 2	Умножение с 4
a << n	Умножение с 2^n

ОПЕРАТОРИ ЗА ПРИСВОЯВАНЕ (1)

- Присвояванията за изрази във формата

  ЛяваСтрана оператор_за_просвояване Израз

  - напр.: **`x = x + y`**
- ЛяваСтрана означава памет (в общия случай променлива)

Разлика за "x" в **`x = x + y`** ;

- 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

+ | * | - | / | % | & | ˆ | << | >> | >>>

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)

? оператор:

ЛогическиИзраз ? израз1 : израз2

доставя:

израз1, ако ЛогичекиИзраз == true
израз2, ако ЛогичекиИзраз == false
напр. x > y ? max = x : max = 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 Typ ([ArgumentList])`

С конструктура *Typ([Argumentlist])*
За инициализиране на една инстанция

instanceof – оператор:

`InstanceName instanceof ClassName`

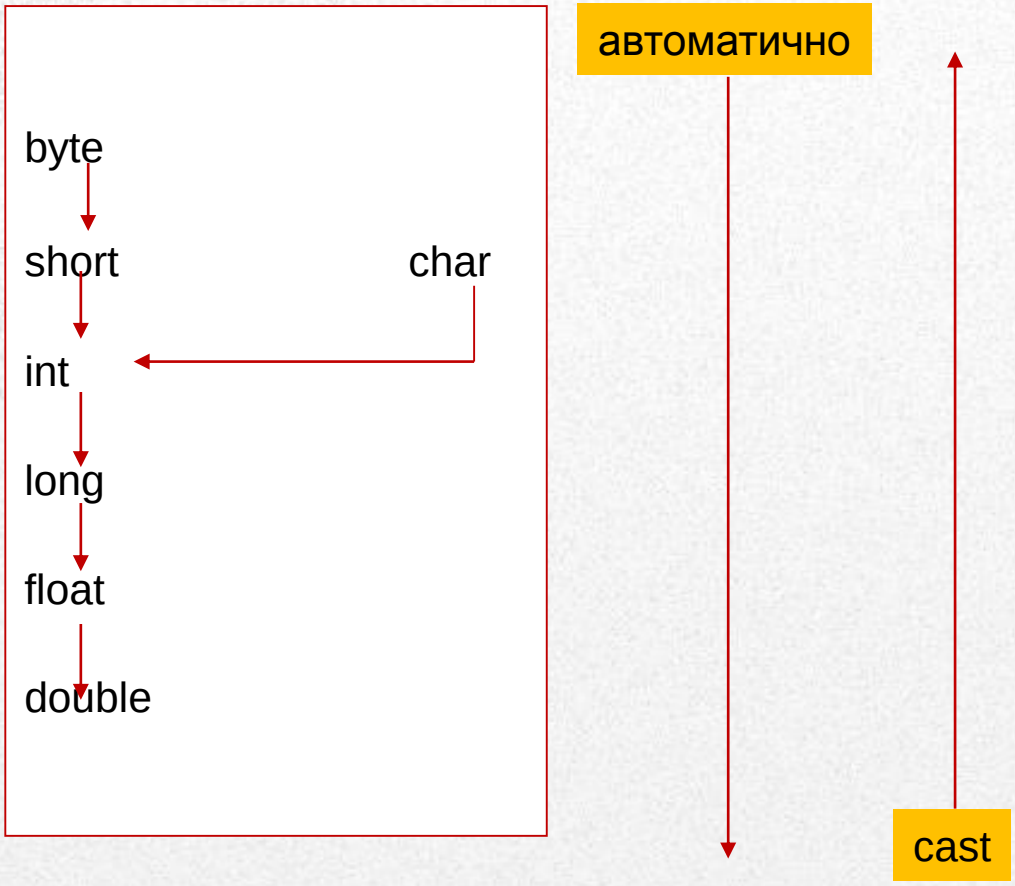
Тип на резултата **boolean**:

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

ПРЕОБРАЗУВАНЕ НА ТИПОВЕ: CAST-ОПЕРАТОР

Проблем:Операнди от различен тип

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



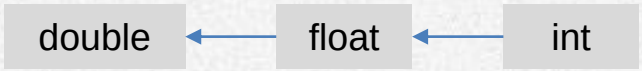
АТОМАТИЧНО ПРЕОБРАЗУВАНЕ НА ТИПОВЕ

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

1 Колко трансформации се извършват при изчисляване на израза от примера?

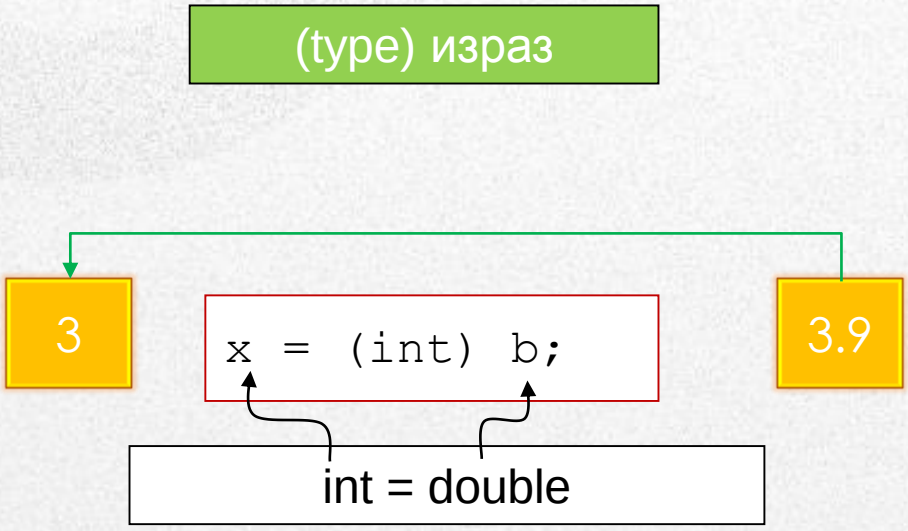
```
int a;
float b;
double x;

x = a + b;
```

2



ЯВНО ПРЕОБРАЗУВАНЕ НА ТИПОВЕ: TYPE-CAST–ОПЕРАТОР

- Преобразува израза 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:
Управлщващи 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	¡	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	¹	186	º	187	»	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										

ПРЕГЛЕД НА ОПЕРАЦИИТЕ:ПРИОРИТЕТИ

Операция	Типизиране	Асоциативност	Означение
		Група 1	
++	N	R	Increment
--	N	R	Decrement
+	N	R	Uneres Plus
-	N	R	Uneres Minus
~	I	R	Onecomplement
!	L	R	Negation
(type)	A	R	Type-Cast
		Група 2	
*	N,N	L	Multiplication
/	N,N	L	Division
%	N,N	L	Modulo
		Група 3	
+	N,N	L	Addition
-	N,N	L	Extraction
+	S,S	L	String concatenation
		Група 4	
<<	I,I	L	Left write
>>	I,I	L	Right write
>>>	I,I	L	Right write with Nullexpansion

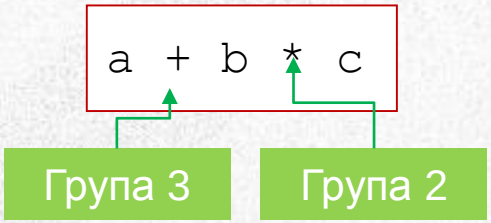
Операция	Типизиране	Асоциативност	Означение
		Група 5	
<	N,N	L	Smaller
<=	N,N	L	Smaller eq.
>	N,N	L	Bigger
>=	N,N	L	Bigger eq.
instanceof	R,R	L	Class membership
		Група 6	
==	P,P	L	Equally
!=	P,P	L	Unequally
==	R,R	L	Referece eq.
!=	R,R	L	Reference uneq.
		Група 7	
&	I,I	L	Bitw. AND
&	L,L	L	Log. AND
		Група 8	
^	I,I	L	Bitw. XOR
^	L,L	L	Log. XOR
		Група 9	
|	I,I	L	Bitw. OR
|	L,L	L	Log. OR
		Група 10	
&&	L,L	L	Log. AND, Short-Cut

Операция	Типизиране	Асоциативност	Означение
		Група 11	
||	L,L	L	Log. OR, Short-Cut
		Група 12	
? :	L,A,A	R	Conditional evaluation
		Група 13	
=	V,A	R	Assignment
+=	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



Еквивалентен на:

```
a + ( b * c )
```

Сравнение: (int) 3.1 + 3.9 и (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 “ИЗРАЗИ”

Distributed
@learning center