

5.4 Arrays

Oft müssen viele Werte gleichen Typs gespeichert werden.

Idee:

- ▶ Lege sie konsekutiv ab!
- ▶ Greife auf einzelne Werte über ihren Index zu!

Feld:	17	3	-2	9	0	1
Index:	0	1	2	3	4	5

Beispiel

```
1 int [] a; // Deklaration
2 int n = read();
3
4 a = new int[n]; // Anlegen des Felds
5 int i = 0;
6 while (i < n) {
7     a[i] = read();
8     i = i+1;
9 }
```

Einlesen eines Feldes

5.4 Arrays

Oft müssen viele Werte gleichen Typs gespeichert werden.

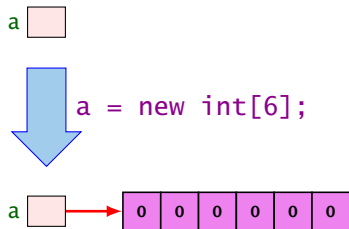
Idee:

- ▶ Lege sie konsekutiv ab!
- ▶ Greife auf einzelne Werte über ihren Index zu!

Feld:	17	3	-2	9	0	1
Index:	0	1	2	3	4	5

Beispiel

- ▶ `type[] name;` deklariert eine Variable für ein Feld (`array`), dessen Elemente vom Typ `type` sind.
- ▶ Alternative Schreibweise:
`type name[];`
- ▶ Das Kommando `new` legt ein Feld einer gegebenen Größe an und liefert einen `Verweis` darauf zurück:



Beispiel

```
1 int [] a; // Deklaration
2 int n = read();
3
4 a = new int[n]; // Anlegen des Felds
5 int i = 0;
6 while (i < n) {
7     a[i] = read();
8     i = i+1;
9 }
```

Einlesen eines Feldes

Was ist eine Referenz?

Eine Referenzvariable speichert eine Adresse; an dieser Adresse liegt der eigentliche Inhalt der Variablen.



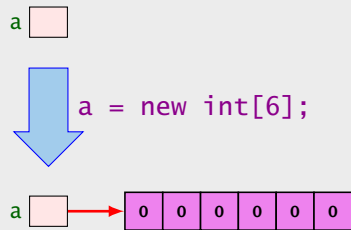
Wir können die Referenz nicht direkt manipulieren (nur über den `new`-Operator, oder indem wir eine andere Referenz zuweisen).

Adresse	Inhalt
⋮	⋮
0000 0127	
a: 0000 0128	0000 012C
0000 0129	
0000 012A	
0000 012B	
0000 012C	0000 0004
0000 012D	0000 0003
0000 012E	0000 0000
0000 012F	0000 0009
0000 0130	
⋮	⋮

A red bracket on the right side of the table spans from the row with address 0000 0128 to the row with address 0000 012C, indicating the range of memory addresses that the variable 'a' points to.

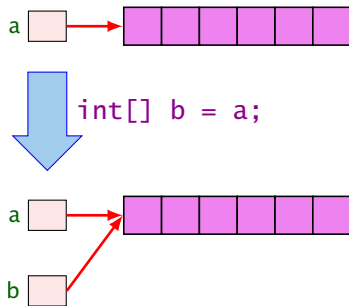
Beispiel

- ▶ `type[] name;` deklariert eine Variable für ein Feld (`array`), dessen Elemente vom Typ `type` sind.
- ▶ Alternative Schreibweise:
`type name[];`
- ▶ Das Kommando `new` legt ein Feld einer gegebenen Größe an und liefert einen `Verweis` darauf zurück:



5.4 Arrays

- ▶ Der Wert einer Feld-Variable ist also ein Verweis!!!
- ▶ `int[] b = a;` kopiert den Verweis der Variablen `a` in die Variable `b`:



- ▶ **Alle nichtprimitive Datentypen sind Referenztypen, d.h., die zugehörige Variable speichert einen Verweis!!!**

Was ist eine Referenz?

Eine Referenzvariable speichert eine Adresse; an dieser Adresse liegt der eigentliche Inhalt der Variablen.



Wir können die Referenz nicht direkt manipulieren (nur über den `new`-Operator, oder indem wir eine andere Referenz zuweisen).

Adresse	Inhalt
⋮	⋮
0000 0127	
a: 0000 0128	0000 012C
0000 0129	
0000 012A	
0000 012B	
0000 012C	0000 0004
0000 012D	0000 0003
0000 012E	0000 0000
0000 012F	0000 0009
0000 0130	
⋮	⋮

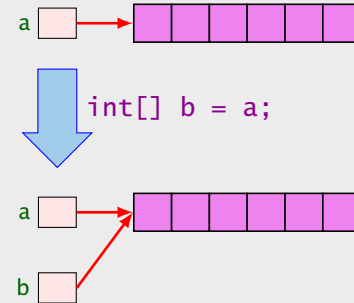
A red bracket on the right side of the table highlights the memory addresses 0000 0128 and 0000 012C, with an arrow pointing from the value 0000 012C in the 'Inhalt' column to the value 0000 0004 in the 'Inhalt' column at address 0000 012C.

5.4 Arrays

- ▶ Die Elemente eines Feldes sind von 0 an durchnummeriert.
- ▶ Die Anzahl der Elemente des Feldes `name` ist `name.length`.
- ▶ Auf das i -te Element greift man mit `name[i]` zu.
- ▶ Bei jedem Zugriff wird überprüft, ob der Index erlaubt ist, d.h. im Intervall $\{0, \dots, \text{name.length}-1\}$ liegt.
- ▶ Liegt der Index außerhalb des Intervalls, wird eine `ArrayIndexOutOfBoundsException` ausgelöst (↑**Exceptions**).

5.4 Arrays

- ▶ Der Wert einer Feld-Variable ist also ein Verweis!!!
- ▶ `int[] b = a;` kopiert den Verweis der Variablen `a` in die Variable `b`:



- ▶ **Alle nichtprimitive Datentypen sind Referenztypen, d.h., die zugehörige Variable speichert einen Verweis!!!**

Mehrdimensionale Felder

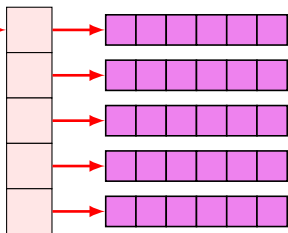
- ▶ **Java** unterstützt direkt nur eindimensionale Felder.
- ▶ ein zweidimensionales Feld ist ein Feld von Feldern...

a



`a = new int[5][6];`

a



5.4 Arrays

- ▶ Die Elemente eines Feldes sind von **0** an durchnummeriert.
- ▶ Die Anzahl der Elemente des Feldes `name` ist `name.length`.
- ▶ Auf das *i*-te Element greift man mit `name[i]` zu.
- ▶ Bei jedem Zugriff wird überprüft, ob der Index erlaubt ist, d.h. im Intervall $\{0, \dots, \text{name.length}-1\}$ liegt.
- ▶ Liegt der Index außerhalb des Intervalls, wird eine `ArrayIndexOutOfBoundsException` ausgelöst (↑**Exceptions**).

Der new-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
new	new	Typ, Konstruktor	links	1

Erzeugt ein Objekt/Array und liefert eine Referenz darauf zurück.

1. Version: Erzeugung eines Arrays (Typ ist Arraytyp)

- ▶ `new int[3][7];` oder auch
- ▶ `new int[3][];` (ein Array, das 3 Verweise auf `int` enthält)
- ▶ `new String[10];`
- ▶ `new int[]{1,2,3,};` (ein Array mit den `ints` 1, 2, 3)

2. Version: Erzeugung eines Objekts durch Aufruf eines Konstruktors

- ▶ `String s = new String("Hello World!");`

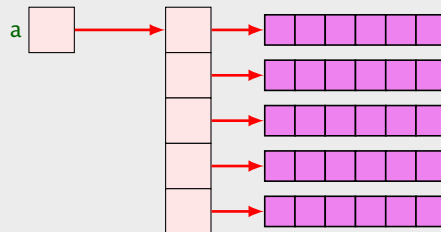
Mehrdimensionale Felder

- ▶ `Java` unterstützt direkt nur eindimensionale Felder.
- ▶ ein zweidimensionales Feld ist ein Feld von Feldern...

a 



`a = new int[5][6];`



Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
<code>[]</code>	index	array, int	links	1

Zugriff auf ein Arrayelement.

Der new-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
<code>new</code>	new	Typ, Konstruktor	links	1

Erzeugt ein Objekt/Array und liefert eine Referenz darauf zurück.

1. Version: Erzeugung eines Arrays (Typ ist Arraytyp)
 - ▶ `new int[3][7];` oder auch
 - ▶ `new int[3][];` (ein Array, das 3 Verweise auf `int` enthält)
 - ▶ `new String[10];`
 - ▶ `new int[]{1,2,3,};` (ein Array mit den `ints` 1, 2, 3)
2. Version: Erzeugung eines Objekts durch Aufruf eines Konstruktors
 - ▶ `String s = new String("Hello World!");`

Beispiel: $x = a[3][2]$

`x = a [3] [2]`

Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
<code>[]</code>	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

x = a [3] [2]

Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
[]	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

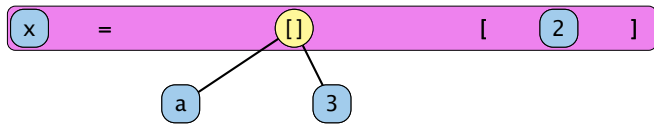


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
$[]$	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

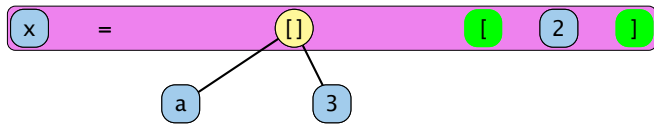


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
$[]$	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

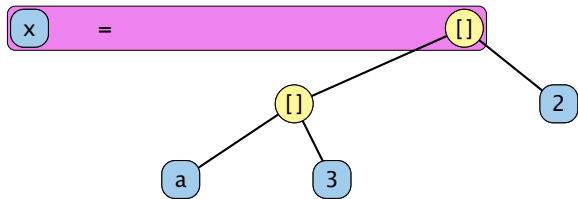


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
$[\]$	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

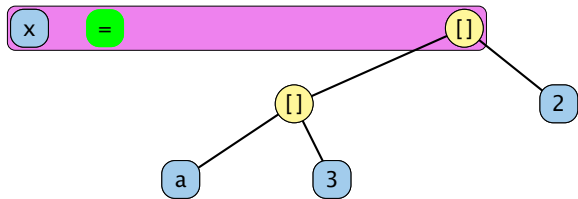


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
[]	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

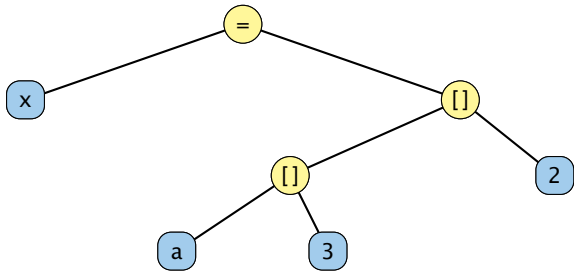


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
[]	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

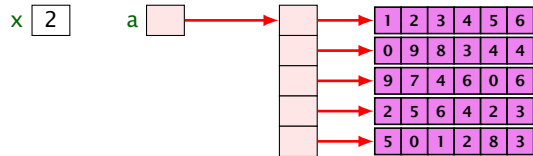
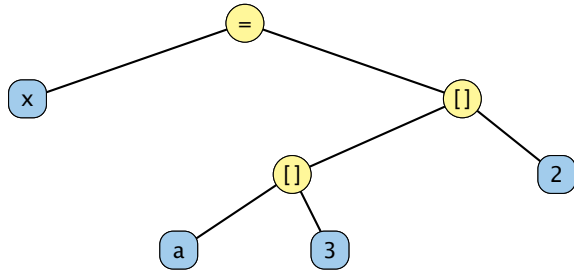


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
<code>[]</code>	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

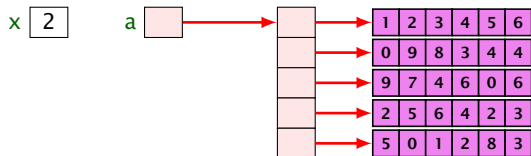
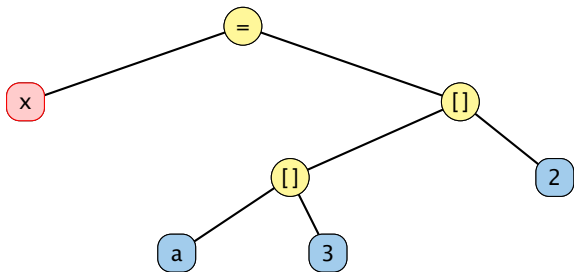


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
<code>[]</code>	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

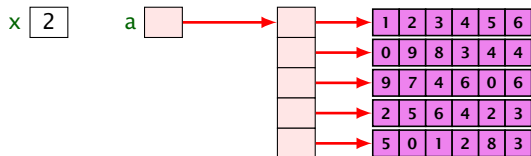
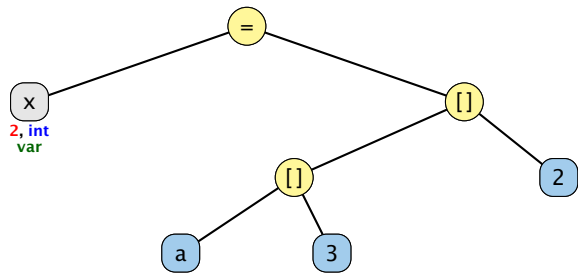


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
<code>[]</code>	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

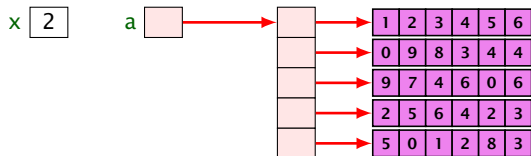
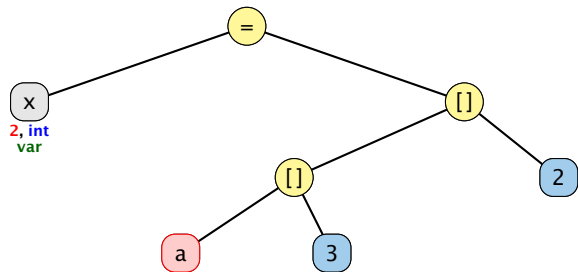


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
[]	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

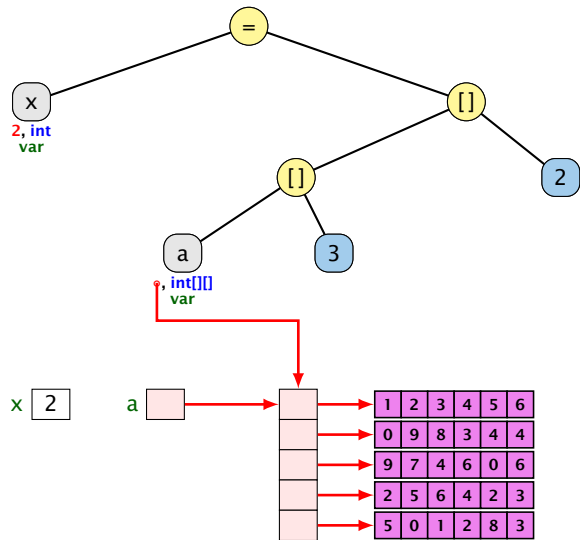


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
[]	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

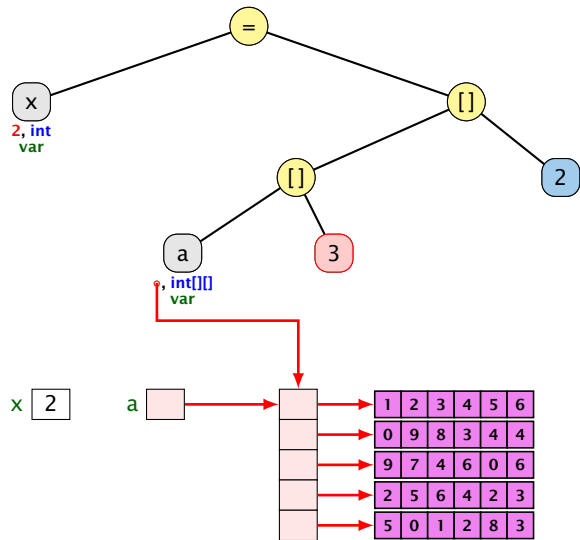


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
$[]$	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

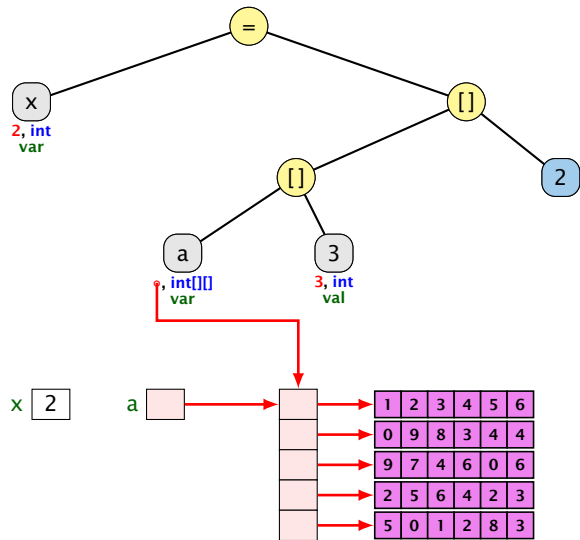


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
<code>[\]</code>	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

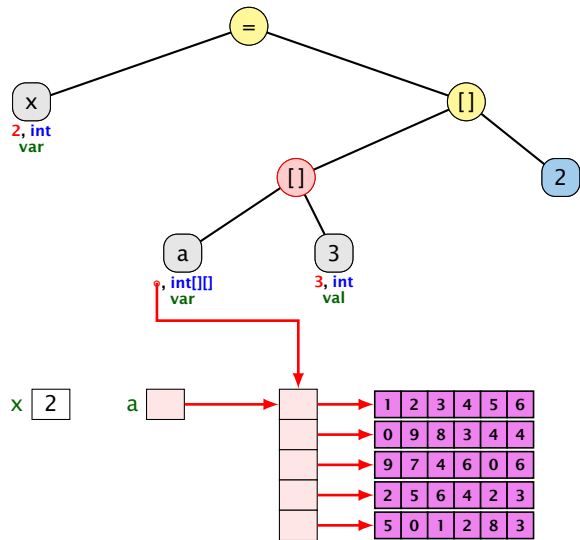


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
$[]$	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

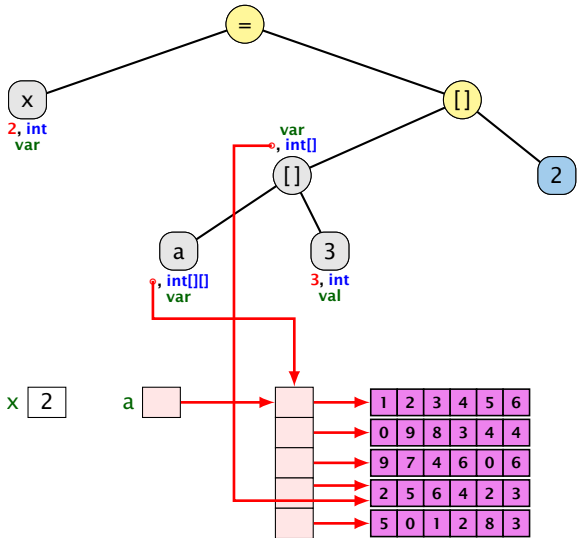


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
$[]$	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: x = a[3][2]

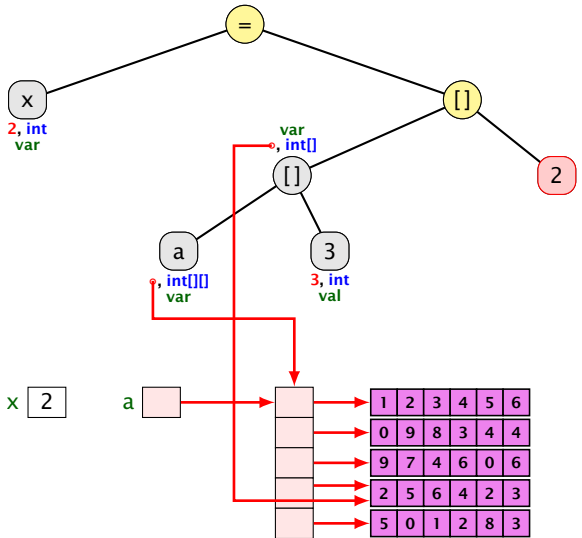


Der Index-Operator

symbol	name	types	L/R	level
[]	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: x = a[3][2]

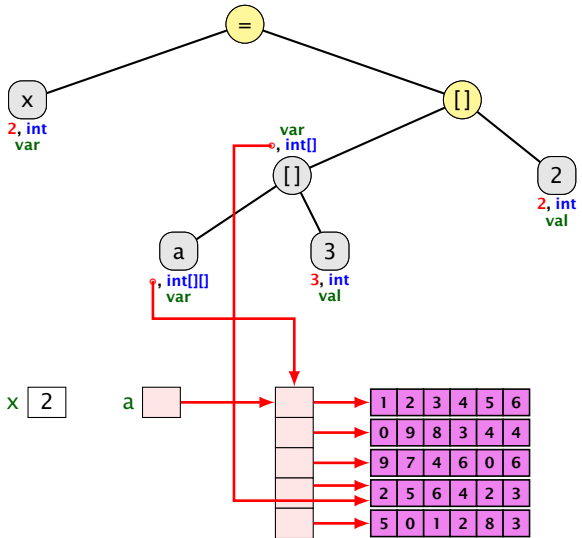


Der Index-Operator

symbol	name	types	L/R	level
[]	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: x = a[3][2]

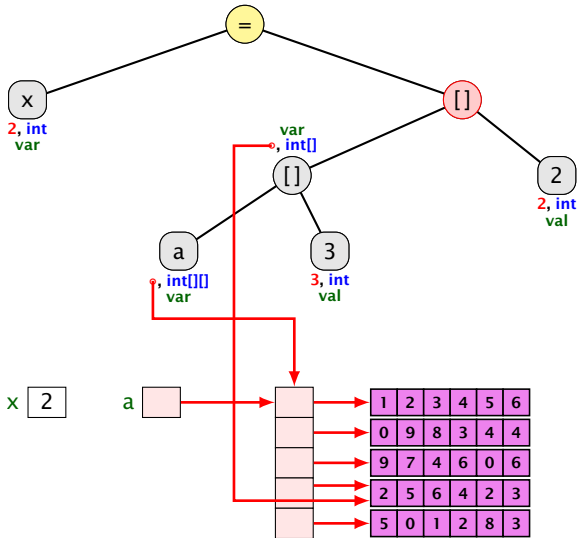


Der Index-Operator

symbol	name	types	L/R	level
[]	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: x = a[3][2]

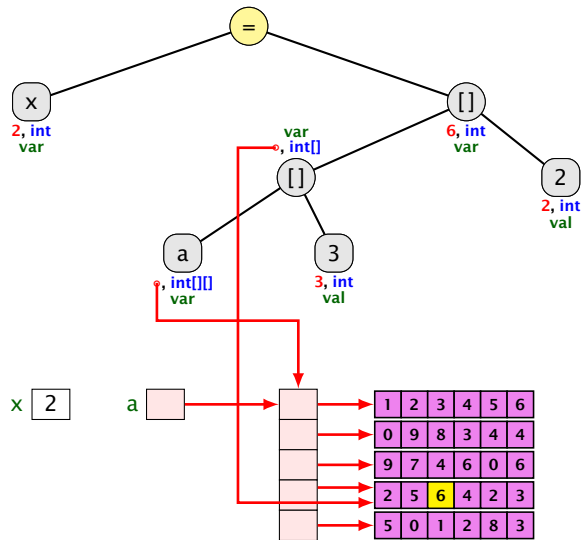


Der Index-Operator

symbol	name	types	L/R	level
[]	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: $x = a[3][2]$

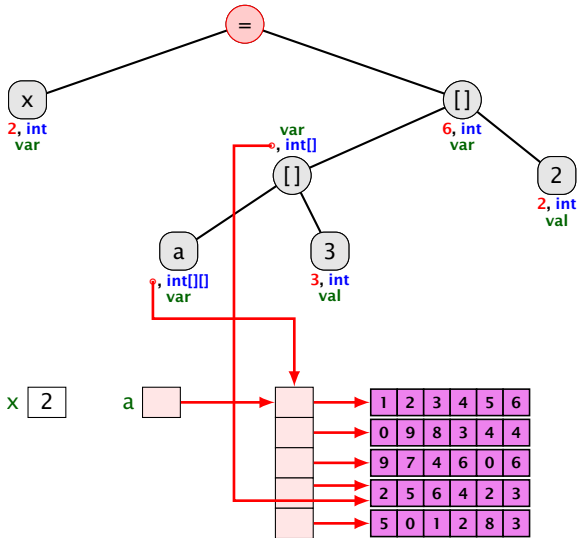


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
$[]$	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: x = a[3][2]

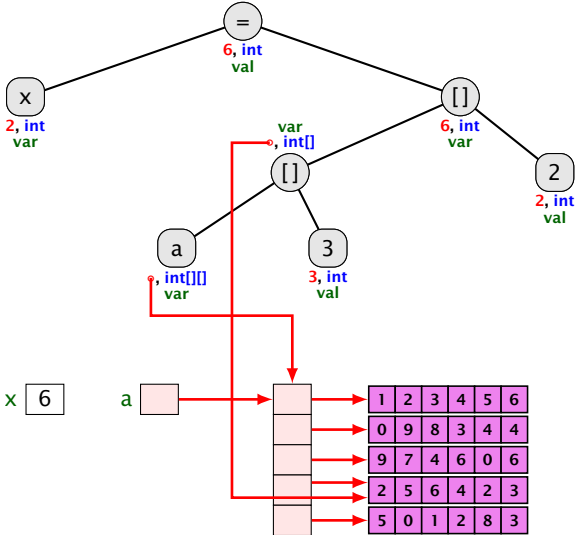


Der Index-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
<code>[]</code>	index	array, int	links	1

Zugriff auf ein Arrayelement.

Beispiel: x = a[3][2]



Der Index-Operator

symbol	name	types	L/R	level
[]	index	array, int	links	1

Zugriff auf ein Arrayelement.

Der .-Operator

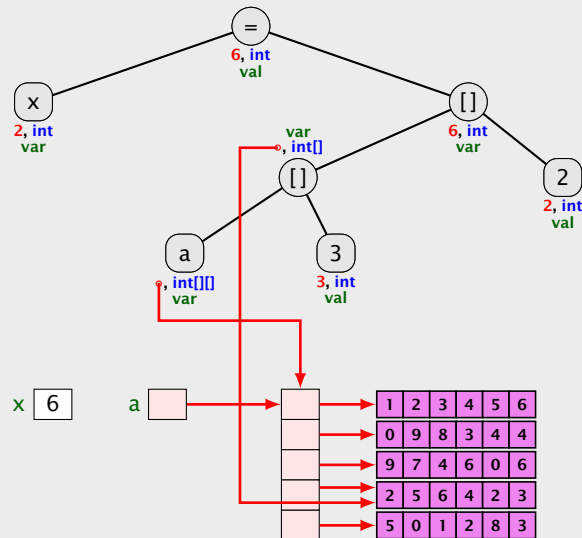
symbol	name	types	L/R	level
.	member access	Array/Objekt/Class, Member	links	1

Zugriff auf Member.

Beispiel:

- ▶ `x = new int[2][4].length`
x hat dann den Wert 2.

Beispiel: `x = a[3][2]`



Beispiel: `new int[2][4].length`

Das Parsing für den `new`-Operator passt nicht in das Schema:

```
new int [ 2 ] [ 4 ] . length
```

Der `.`-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
<code>.</code>	member access	Array/Objekt/Class, Member	links	1

Zugriff auf Member.

Beispiel:

- ▶ `x = new int[2][4].length`
x hat dann den Wert 2.

Beispiel: `new int[2][4].length`

Das Parsing für den `new`-Operator passt nicht in das Schema:

`new int [2] [4] . length`

Der `.`-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
<code>.</code>	member access	Array/Objekt/Class, Member	links	1

Zugriff auf Member.

Beispiel:

- ▶ `x = new int[2][4].length`
x hat dann den Wert 2.

Beispiel: `new int[2][4].length`

Das Parsing für den `new`-Operator passt nicht in das Schema:

`new int [2] [4] . length`

Der `.`-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
<code>.</code>	member access	Array/Objekt/Class, Member	links	1

Zugriff auf Member.

Beispiel:

- ▶ `x = new int[2][4].length`
x hat dann den Wert 2.

Beispiel: `new int[2][4].length`

Das Parsing für den `new`-Operator passt nicht in das Schema:



Der `.`-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
.	member access	Array/Objekt/Class, Member	links	1

Zugriff auf Member.

Beispiel:

- ▶ `x = new int[2][4].length`
x hat dann den Wert 2.

Beispiel: `new int[2][4].length`

Das Parsing für den `new`-Operator passt nicht in das Schema:



Der `.`-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
.	member access	Array/Objekt/Class, Member	links	1

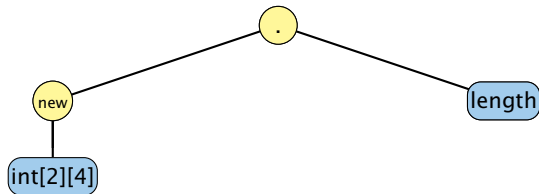
Zugriff auf Member.

Beispiel:

- ▶ `x = new int[2][4].length`
x hat dann den Wert 2.

Beispiel: `new int[2][4].length`

Das Parsing für den `new`-Operator passt nicht in das Schema:



Der `.`-Operator

<i>symbol</i>	<i>name</i>	<i>types</i>	<i>L/R</i>	<i>level</i>
.	member access	Array/Objekt/Class, Member	links	1

Zugriff auf Member.

Beispiel:

- ▶ `x = new int[2][4].length`
x hat dann den Wert 2.

Arrayinitialisierung

1. `int[] a = new int[3];`
`a[0] = 1; a[1] = 2; a[2] = 3;`
2. `int[] a = new int[]{ 1, 2, 3};`
3. `int[] a = new int[3]{ 1, 2, 3};`
4. `int[] a = { 1, 2, 3};`
5. `char[][] b = { { 'a', 'b' }, new char[3], {} };`
6. `char[][] b;`
`b = new char[][]{ { 'a', 'b' }, new char[3], {} };`
7. `char[][] b;`
`b = { { 'a', 'b' }, new char[3], {} };`

Beispiel: `new int[2][4].length`

Das Parsing für den `new`-Operator passt nicht in das Schema:

