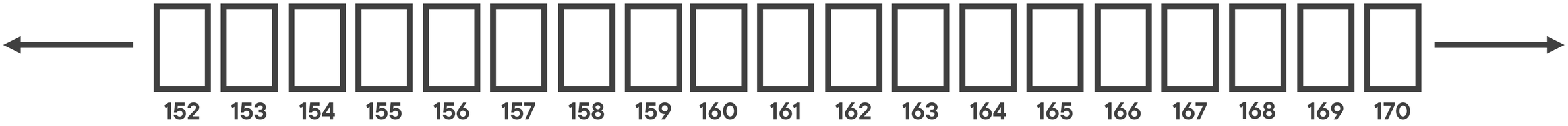
Using Pointers to Access Array Elements

Mateo Prigl
Software Developer

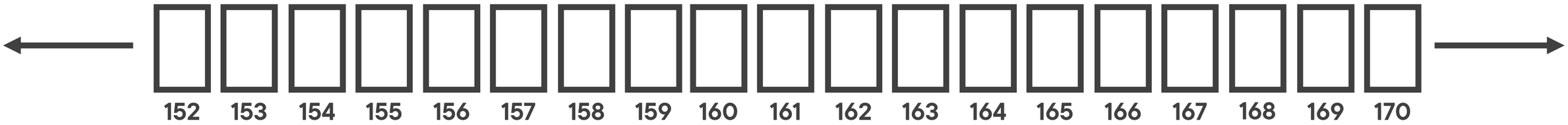
```
int x;
```

Memory (Byte Sequence)



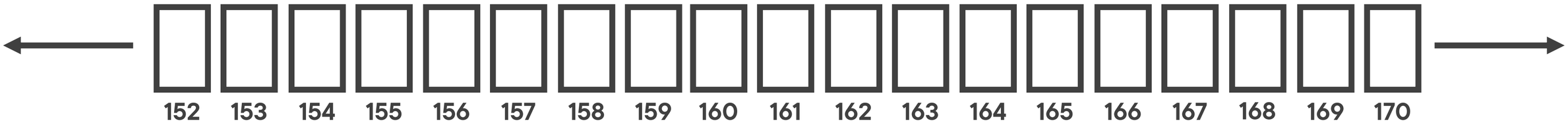
```
int x; // 1 int -> 4B
```

Memory (Byte Sequence)



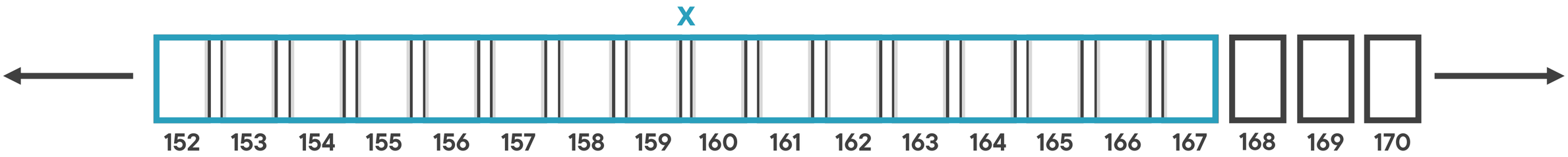
```
int x; // 4 int -> 16B
```

Memory (Byte Sequence)



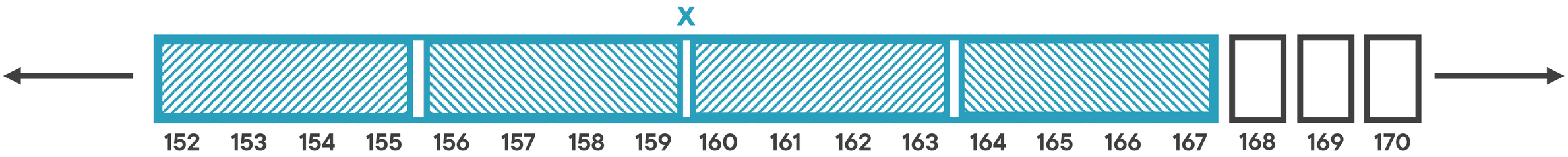
```
int x[4]; // 4 int -> 16B
```

Memory (Byte Sequence)



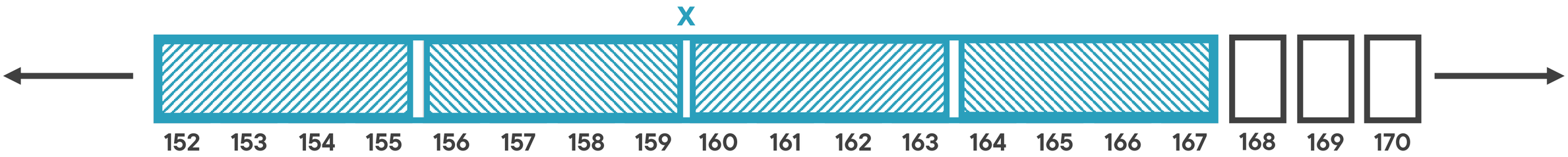
```
int x[4]; // 4 int -> 16B
```

Memory (Byte Sequence)



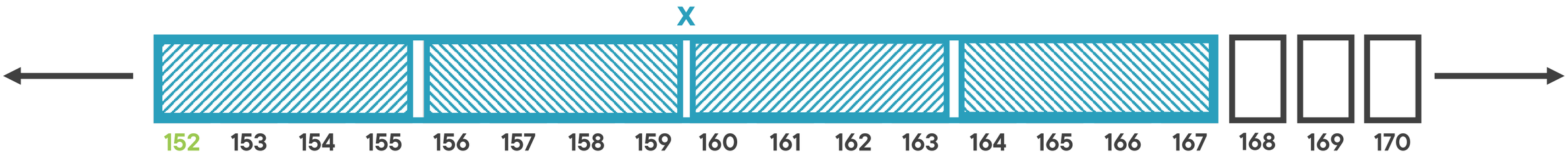
```
int x[4]; // 4 int -> 16B

x // 152
```

Memory (Byte Sequence)



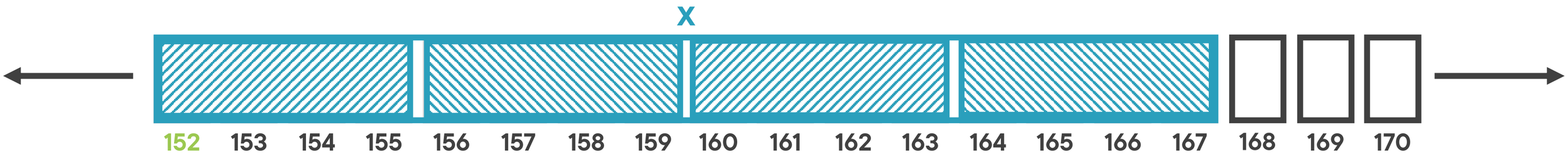
```
int x[4]; // 4 int -> 16B

x // 152
```

Memory (Byte Sequence)



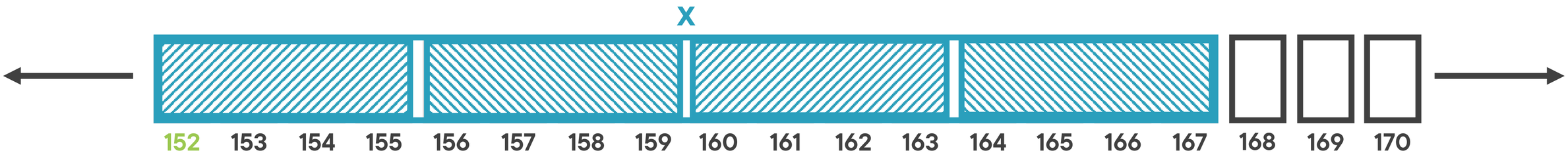
```
int x[4]; // 4 int -> 16B

x // 152

*x // garbage value (no value set)
```

Memory (Byte Sequence)



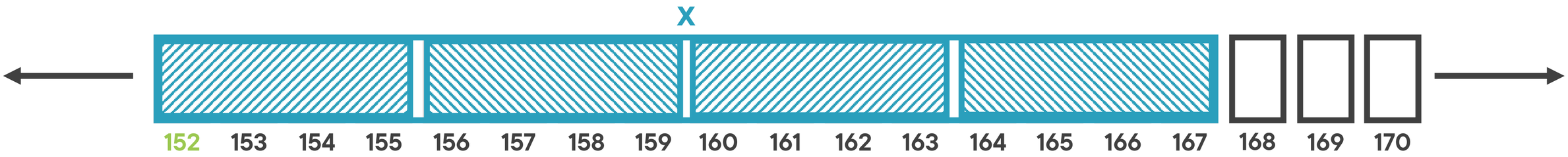
```
int x[4]; // 4 int -> 16B

x // 152

*x // garbage value (no value set)

*x = 3;
```

Memory (Byte Sequence)



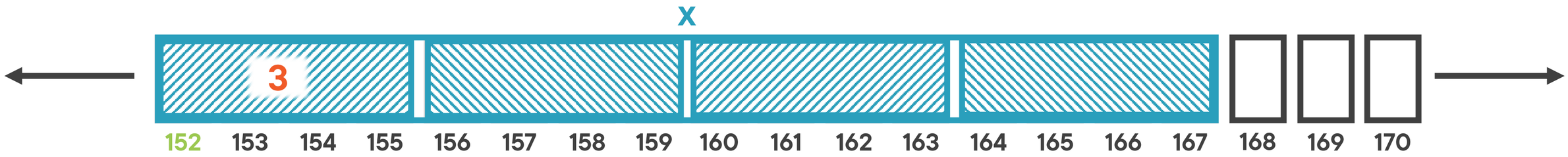
```
int x[4]; // 4 int -> 16B

x // 152

*x // garbage value (no value set)

*x = 3;
```

Memory (Byte Sequence)



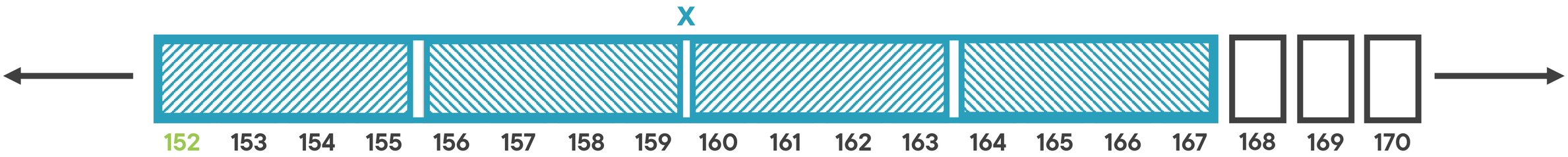
```
int x[4]; // 4 int -> 16B

x // 152

*x // garbage value (no value set)

*x = 3;
```

Memory (Byte Sequence)



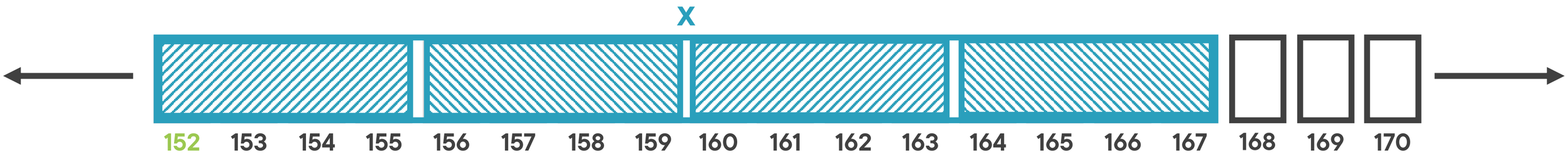
```
int x[4]; // 4 int -> 16B

x // 152
```

Memory (Byte Sequence)



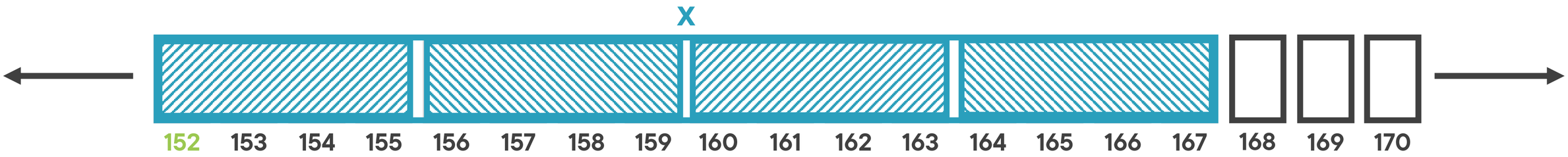
```
int x[4]; // 4 int -> 16B

x // 152

x + 1
```

Memory (Byte Sequence)



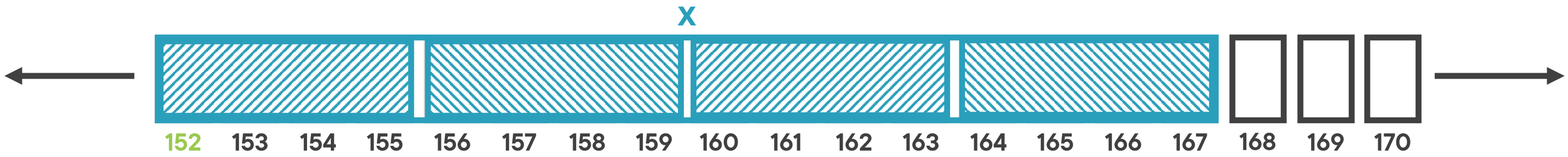
```
int x[4]; // 4 int -> 16B

x // 152

x + 1 // 156
```

Memory (Byte Sequence)



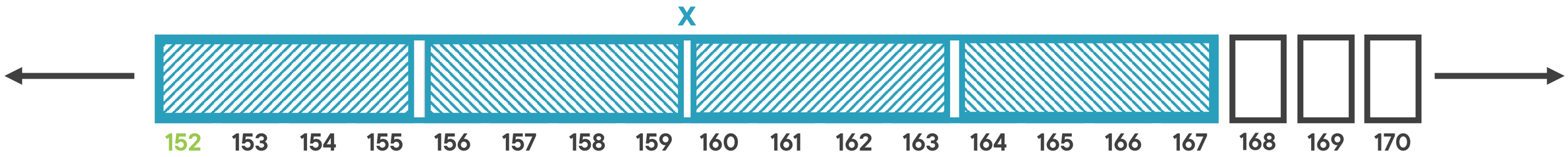
```
int x[4]; // 4 int -> 16B

x // 152

x + 1 // 156
x + 2 // 160
```

Memory (Byte Sequence)



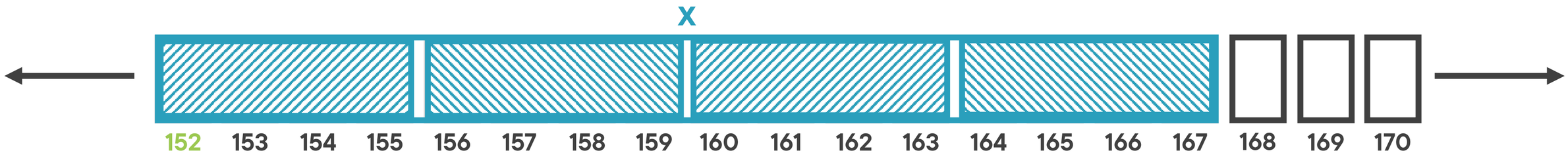
```
int x[4]; // 4 int -> 16B

x // 152

x + 1 // 156
x + 2 // 160
x + 3 // 164
```

Memory (Byte Sequence)



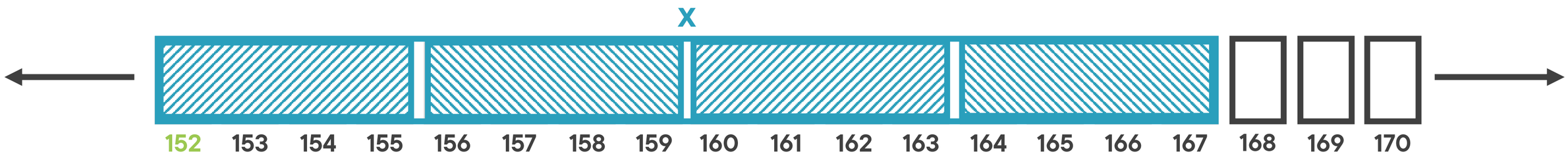
```
int x[4]; // 4 int -> 16B

x // 152

x + 1 // 156
x + 2 // 160
x + 3 // 164
```

Memory (Byte Sequence)



```
int x[4]; // 4 int -> 16B

x // 152
```

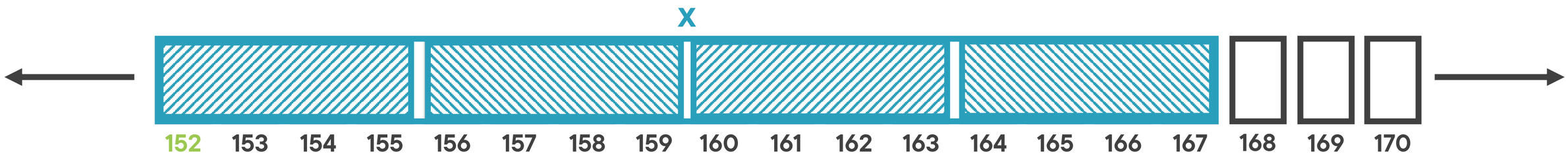
Memory (Byte Sequence)



Elements address: x x + 1 x + 2 x + 3

```
int x[4]; // 4 int -> 16B

x // 152
```

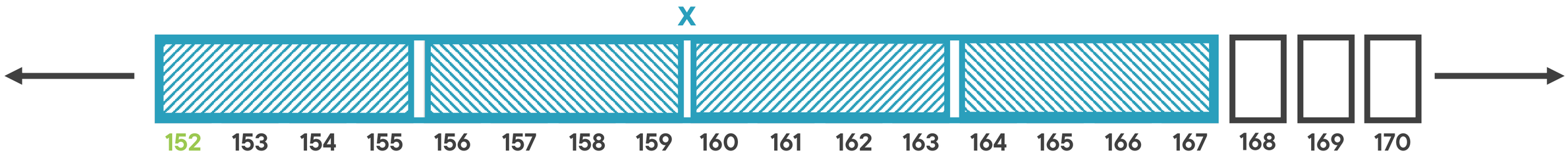
Memory (Byte Sequence)



Elements address: x + 0 x + 1 x + 2 x + 3

```
int x[4]; // 4 int -> 16B

x // 152
```

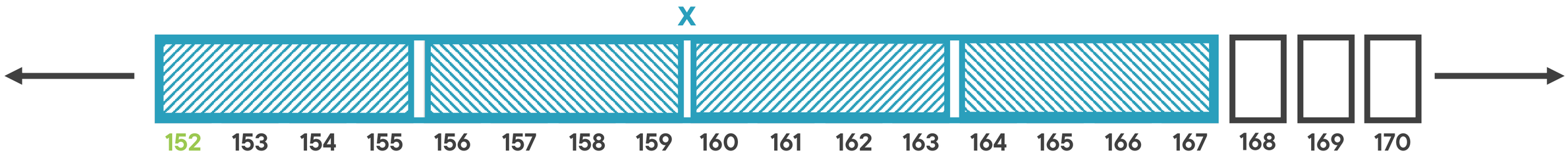
Memory (Byte Sequence)



Elements address:	x + 0	x + 1	x + 2	x + 3
Elements value:	*(x + 0)	*(x + 1)	*(x + 2)	*(x + 3)

```
int x[4]; // Array

x // 152
```

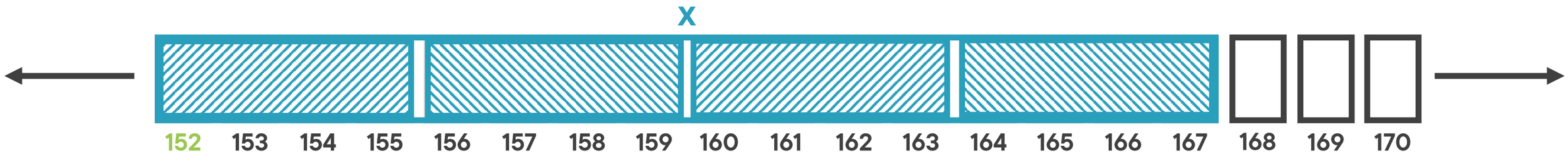
Memory (Byte Sequence)



Elements address:	x + 0	x + 1	x + 2	x + 3
Elements value:	*(x + 0)	*(x + 1)	*(x + 2)	*(x + 3)

```
int x[4]; // Array

x // 152
```

Memory (Byte Sequence)

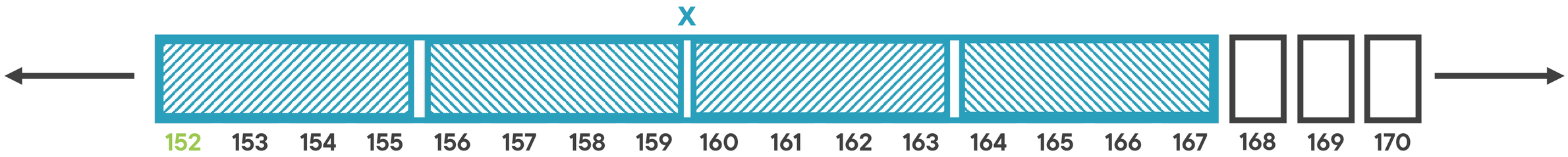


Elements address: x + 0 x + 1 x + 2 x + 3

Elements value: *(x + 0) or x[0] *(x + 1) or x[1] *(x + 2) or x[2] *(x + 3) or x[3]

```
int x[4]; // Array

x // 152
```

Memory (Byte Sequence)

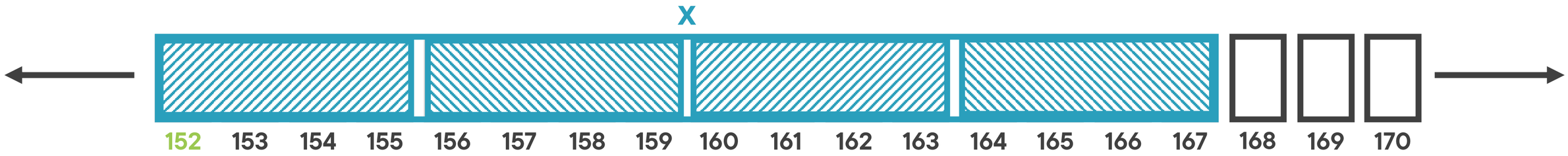


Elements address: x + 0 or &x[0] x + 1 or &x[1] x + 2 or &x[2] x + 3 or &x[3]

Elements value: *(x + 0) or x[0] *(x + 1) or x[1] *(x + 2) or x[2] *(x + 3) or x[3]

```
int x[4]; // Array

x // 152

&x // 152
```

Memory (Byte Sequence)

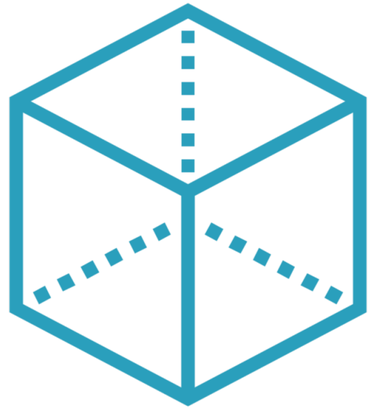


Elements address: x + 0 or &x[0] x + 1 or &x[1] x + 2 or &x[2] x + 3 or &x[3]

Elements value: *(x + 0) or x[0] *(x + 1) or x[1] *(x + 2) or x[2] *(x + 3) or x[3]

Linear Algebra in Programming

$$\begin{bmatrix} 4 & 7 \\ 1 & 5 \end{bmatrix}$$

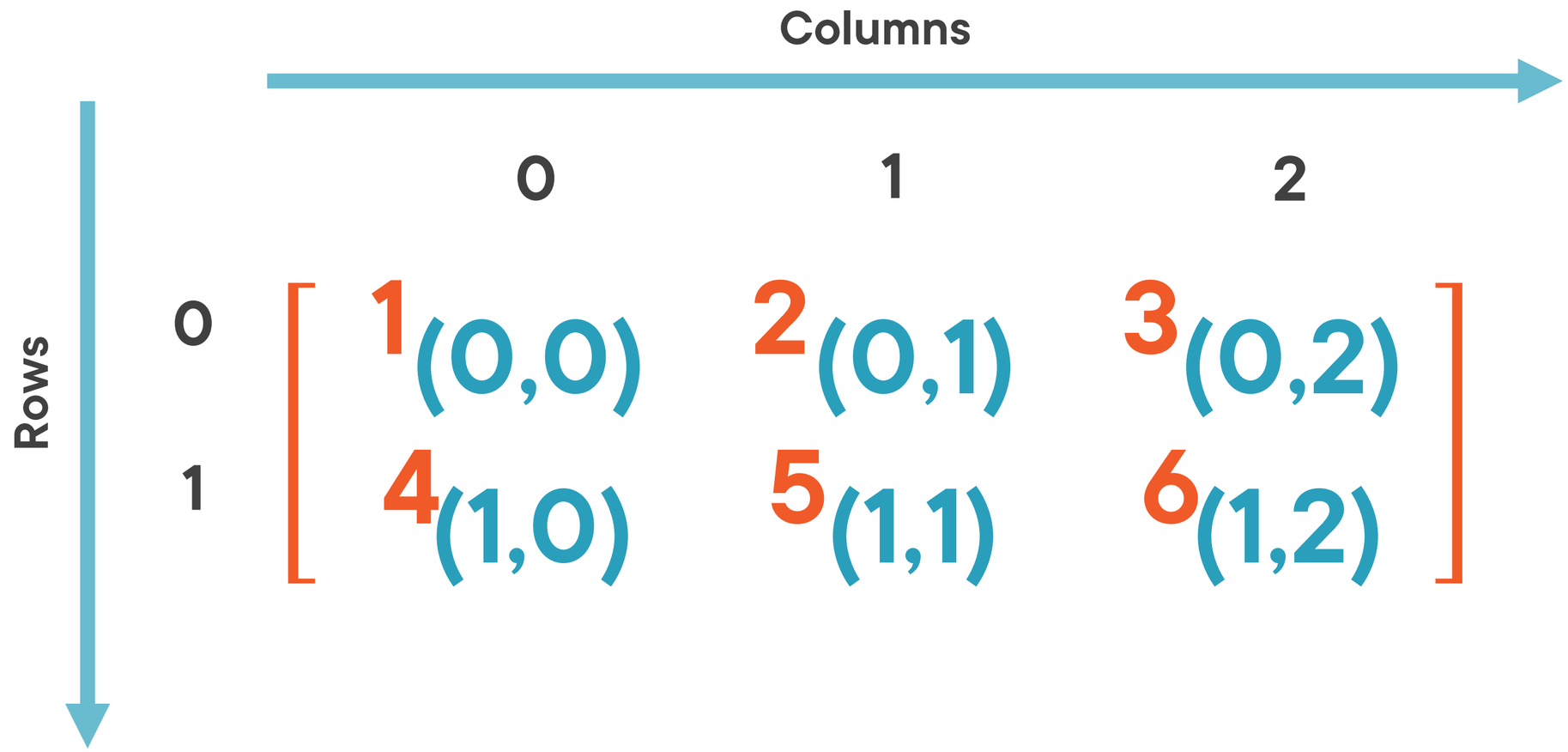


[1, 2, 3]

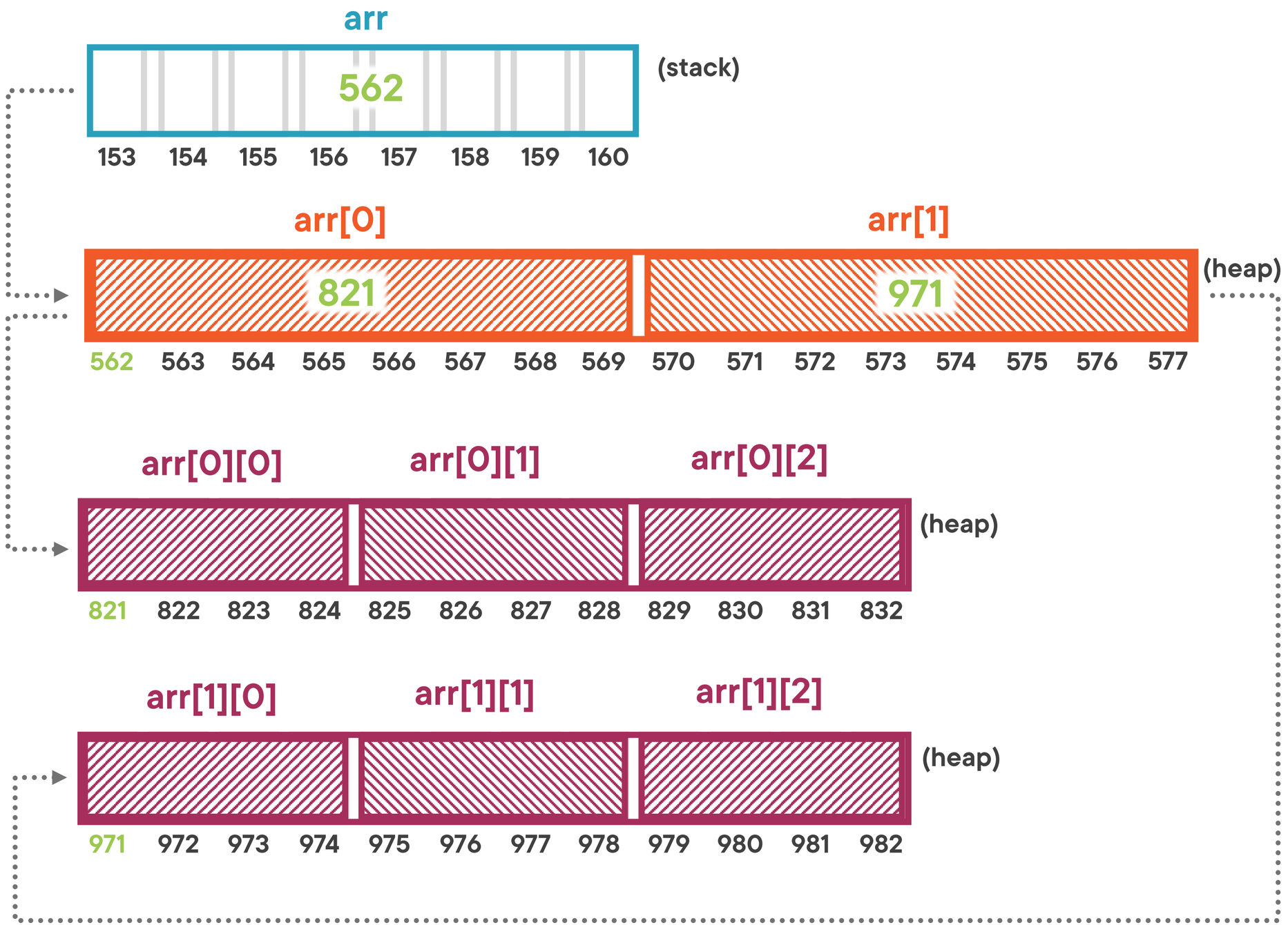
Matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

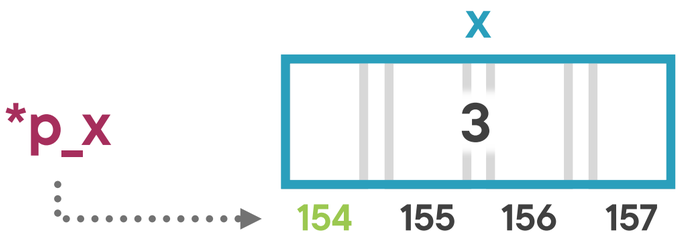
Matrix



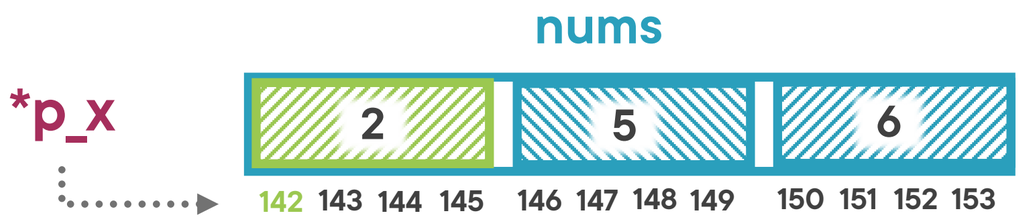
```
int **arr =
new int *[2];

arr[0] =
new int[3];

arr[1] =
new int[3];
```



```c
int x = 3;
int *p_x = &x;
*p_x; // (int) 3


int nums[3] = {2,5,6};
int *p_x = nums;
*p_x; // (int) 2


int *arr1[3];
```



Pointer arithmetic: 4 B



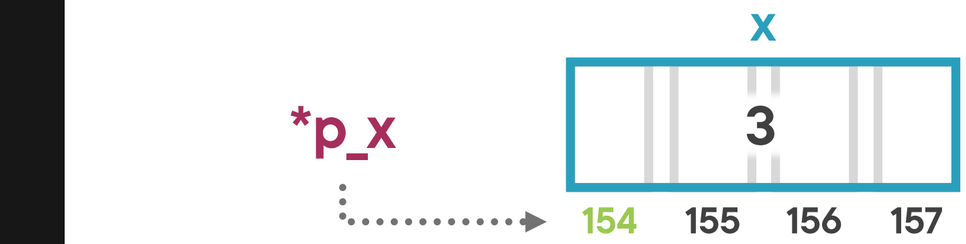
Pointer arithmetic: 4 B

```
int x = 3;
int *p_x = &x;
*p_x; // (int) 3
```

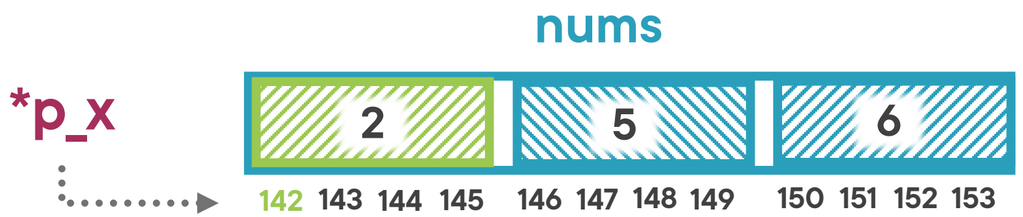
```
int nums[3] = {2,5,6};
int *p_x = nums;
*p_x; // (int) 2
```

```
int* arr1[3];
```

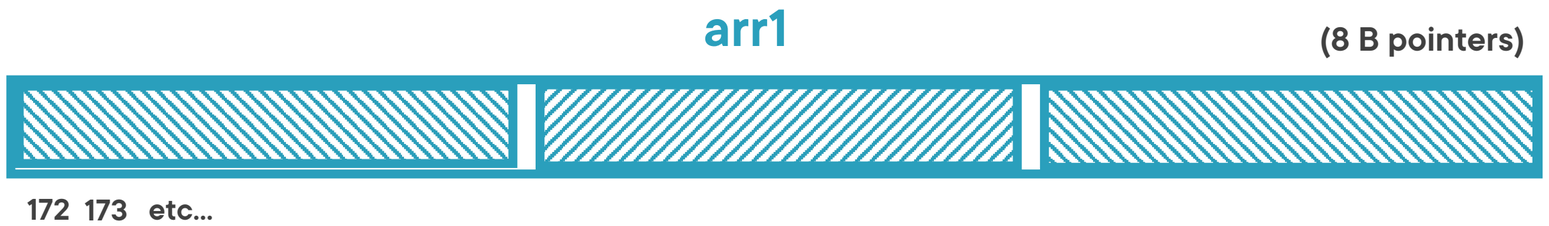
```
int (*arr2)[3];
arr2 = (int(*)[3])nums;
*arr2; // (int *) 142
```

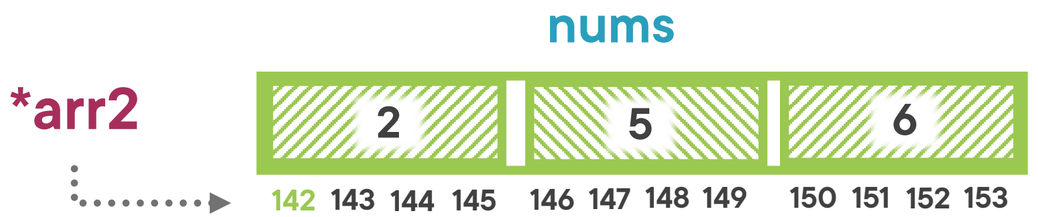


Pointer arithmetic: 4 B



Pointer arithmetic: 4 B





Pointer arithmetic: 12 B

* We need to type cast array name from (int *) to (int(*)[3])

Summary

Up Next:
Pointing to Class Members