Desugaring arrays
what is the syntactic sugar?

- An array is just a series of boxes stored consecutively in memory
  - In some languages arrays are objects (store length, etc.)
  - In C/C++, arrays are just regions of memory

```
int x[5]
```

- So how do we deal with arrays?
- Arrays are essentially pointers with special syntax!
allocating arrays

• An array is a **base pointer** plus a size
  • Base pointer is *just a pointer* that points to the beginning of the array
  • Size defines number boxes in array

• Allocating an array is just assigning a pointer

```c
int * p
p = malloc(10 * 4) //allocate an array of 10 integers
```
allocating arrays

• An array is a **base pointer** plus a size
  • Base pointer is *just a pointer* that points to the beginning of the array
  • Size defines number boxes in array

• You may see explicit array syntax for global/stack allocation:

  ```
  int p[10]; //allocate 10–integer array on stack
  ```

• In this case, p is still just an `int *` pointer with some extra compiler smarts (p == &p)
• Accessing arrays is very simple syntactic sugar:

\[ a[\text{expr}] \equiv * (a + 4 * \text{expr}) \]

size of data type pointed to by \( a \)
code generation for arrays

• Can generate code by implementing a desugaring pass
  • Before code generation, walk over AST, replace array nodes with corresponding pointer-based expression

• Can generate code by implementing desugaring during code generation
Desugaring composes!

\[
\begin{align*}
a[i][j] &= \star (a[i] + 4 \times j) \\
&\approx \star (\star (a + 4 \times i) + 4 \times j)
\end{align*}
\]
are arrays just pointers?

- Syntactic sugar can be complicated
  - In some sense, yes! Array accesses are explicitly equivalent to pointer arithmetic + a dereference, and pointers that point to a dynamically allocated array work as above

- But in another sense, no. If arrays are declared as arrays, with either local or global allocation, they are **array** type and C/C++ do some magic with them:

  - int a[4] vs int * b = malloc(16)
  - a refers to the whole box, a returns &a
  - b is a pointer that points to a separate array:
next: analyzing code