Pointers
what are pointers?

• Up until now, we have only considered variables that represent values.
  • A variable is a named box in memory that contains a value.

```c
int x
```
what are pointers?

- Up until now, we have only considered variables that represent values
  - A variable is a named box in memory that contains a value

- But what if the box can contain the address of another box?
what are pointers?

• Up until now, we have only considered variables that represents values
  • A variable is a named box in memory that contains a value

• But what if the box can contain the address of another box?
what are pointers?

• Up until now, we have only considered variables that represent values.
  • A variable is a named box in memory that contains a value.

• But what if the box can contain the address of another box?
pointers vs references

• A **pointer** is a variable that holds an *address*
• That address can be treated as a *value* that can be computed over

```c
int * p = &a // p gets the address of a
int * q = p + 1 // q gets 4 + the address of a
```

• Some languages only have references instead of pointers
  • A reference refers to another memory location (under the hood: holds the address of another location in memory)
  • But cannot do pointer arithmetic
two key operations

• Two new unary operations:

• & : address-of operation
  • Returns the address of a variable
    \[ p = \&a \quad \text{//store the address of a in } p \]

• * : pointer dereference operation:
  • Let you load from, or store to, a pointer
    \[ \ast p = 7 \quad \text{//store to the address stored in } p \]
    \[ x = \ast p \quad \text{//load from the address stored in } p \]
• How do we build pointers into our type system?
• Can thin of types as being defined by a grammar!

\[
T \to \text{int} \mid \text{float} \\
T \to * T
\]

• Type is either a **base type** or a **pointer to** another type
• What are the type rules for our two unary operations?

• * expr :
  If *expr* has type *T* then *expr* has type *T*

• & expr :
  If expr has type T then & expr has type *T*
next: code generation for pointers