# Strings in C (or lack thereof)

↳ arrays of individual characters

```
char   a = 'z';
```

→ single quotes (really just a number)

---

word

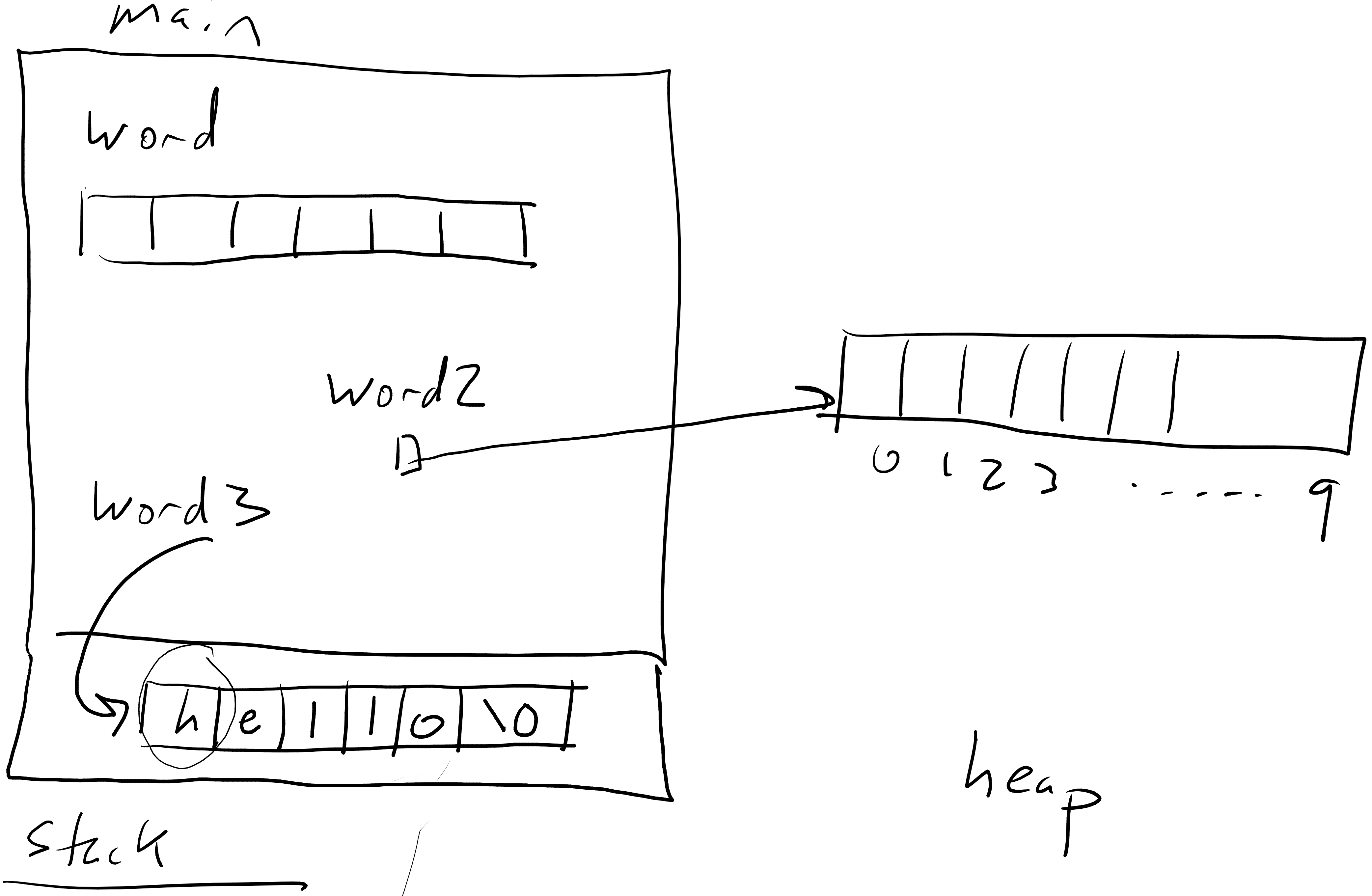
| h | e | l | l | o |  |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

null

'\0'

null terminator

When C processes a string, it doesn't remember how long it is, so everything intenally is done "until null"



doublequotes indicate a string

"hello" string (not really)

stored constant

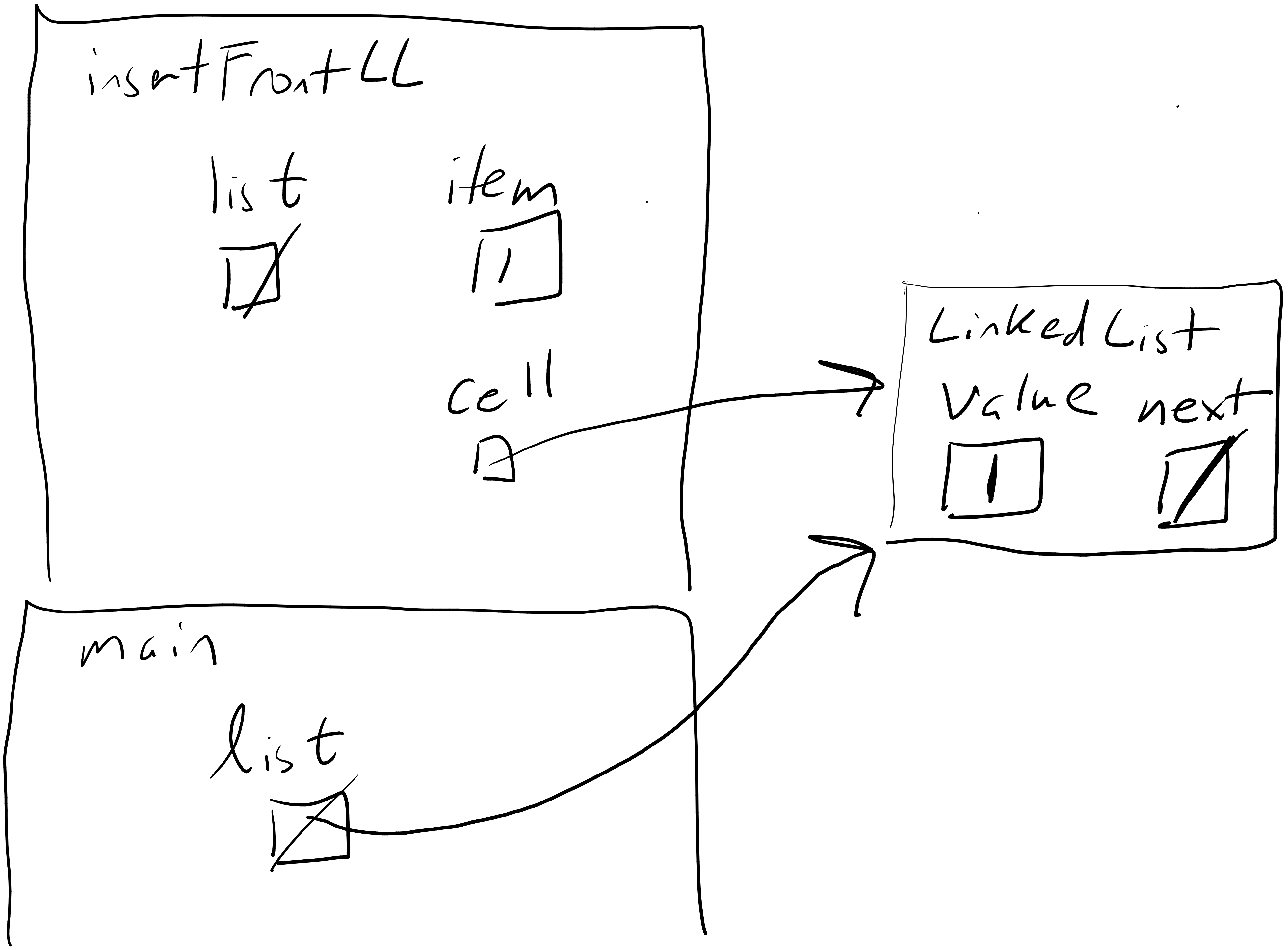
there in

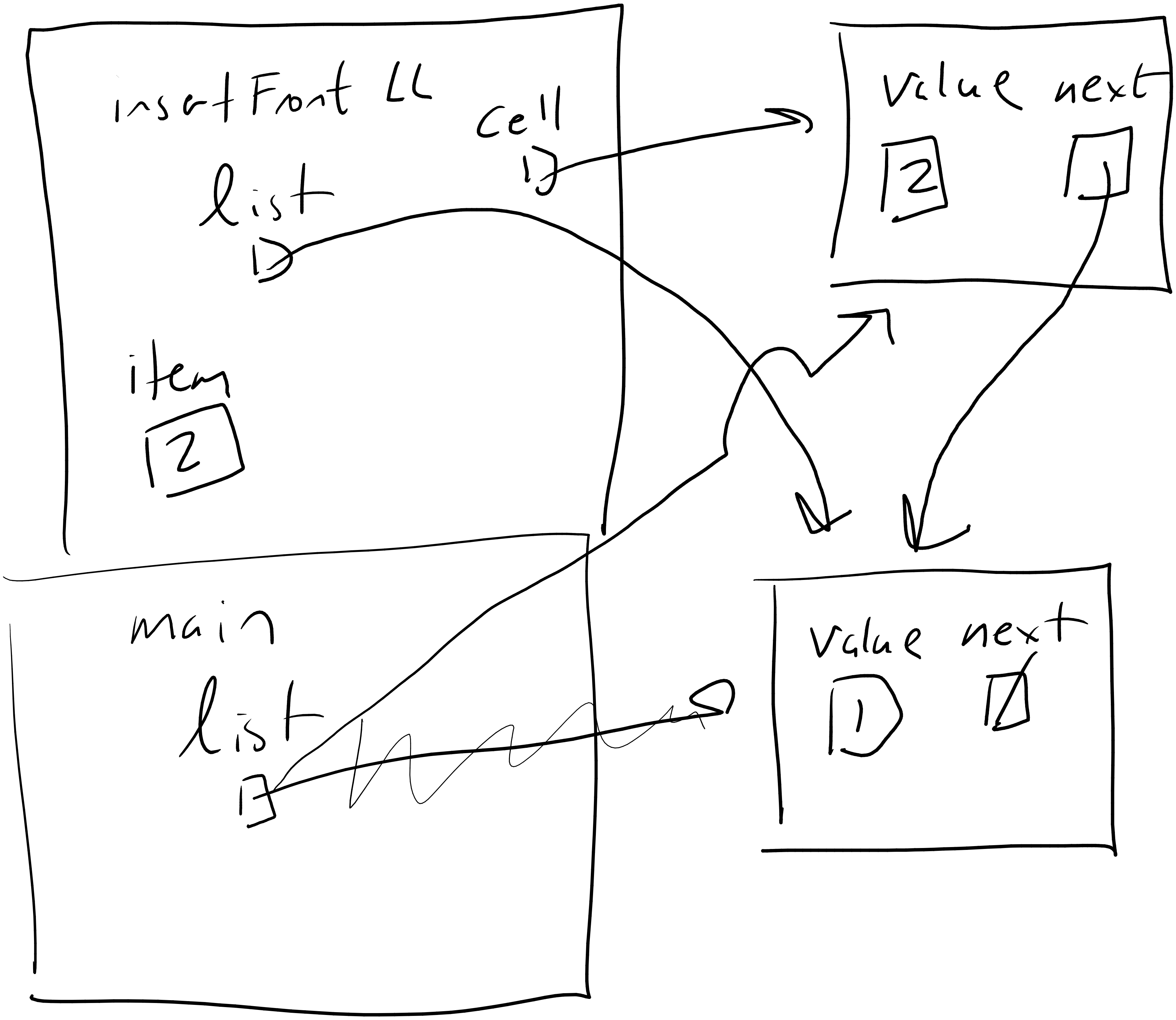
"static" memory

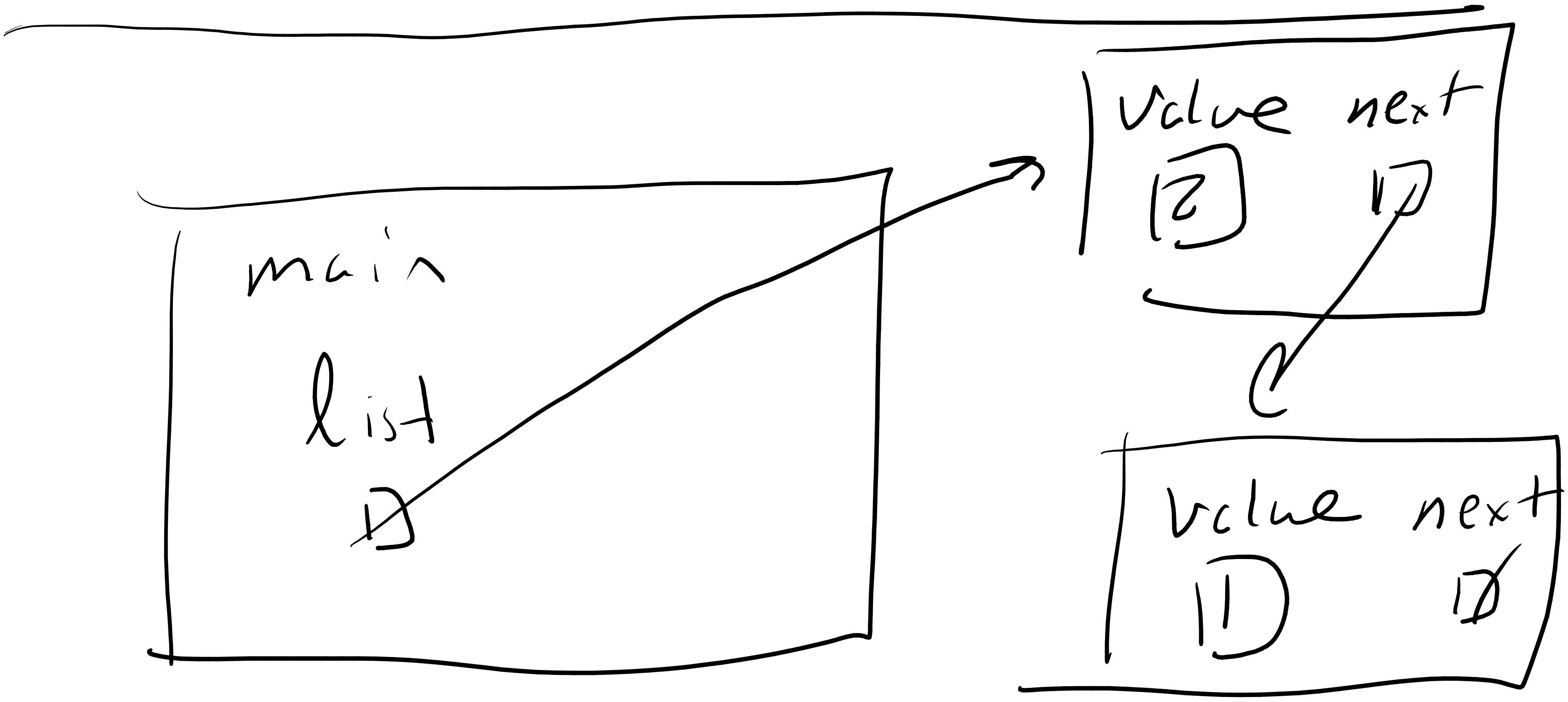
# Linked lists in C

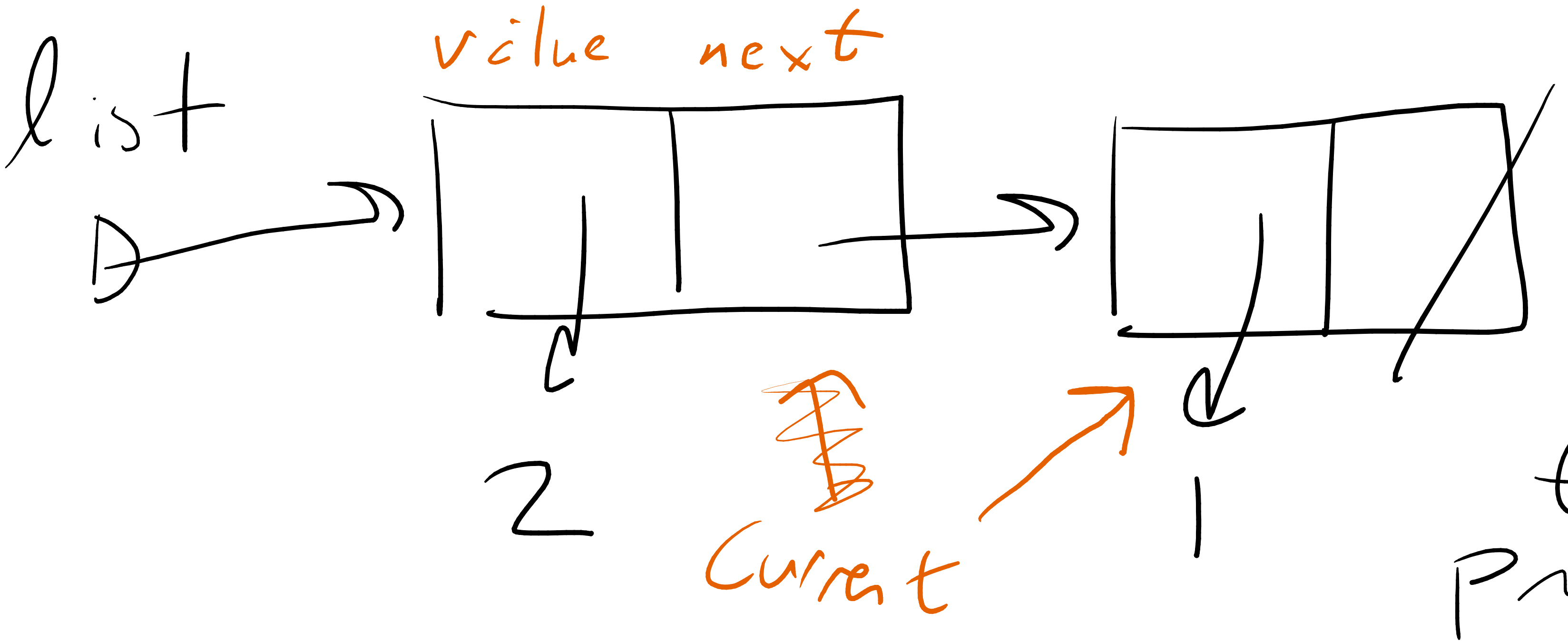
- we'll do a prototype here but a different version is part of the interpreter project

## Linked list code pictures




insertFront LL
cell
list
item
2
value next
2
main
list
value next
1


main
list
value next
2
value next
1



fun it
Problematic
code

```
Curent = list;
while (current != NULL) {

    curent = curent -> next;
    free(curent);
```

A difference in the Scheme interpreter

```
'(a b 3 "hello")
```

---

```
struct LinkedList {
  int value;          ???

  ..
```

we have to trick C into being able to store different types.

→ use a <u>union</u> in C

---

```
struct LinkedList {
   union {
      int i;
      double d;
   }
}
```

A union lets you stoe different types in the same memory.
(but better remember what you put in there).

```
LinkedList *list = malloc(—)
list -> i = 6;
printf("----", list -> i);
  :
  .
```

do the same for d

(occupy same memory so one will clobber the other)