Computer Programming

Implementing an abstract datatype

Marius Minea

marius@cs.upt.ro

16 December 2014

Hiding / exposing the representation

C does not have polymorphism or parametric types
⇒ cannot declare, e.g., list of *arbitrary type*

Could do: `typedef int elemtype;` (or even a `#define`)
and have everything else use `elemtype`

But need to *recompile* everything when changing `elemtype`
binary code differs even for assignment/parameter passing
due to varying element size; even more so for addition, etc.)

Hiding / exposing the representation

Implementation is hidden if only a *pointer* to the data is exposed:
incomplete structure type: `typedef struct ilst *intlist_t`
or even a `void *` (only implementation knows what it points to)

Declaration of structure should be hidden in `.c` file
not exposed in `.h` file (which is included by all clients)

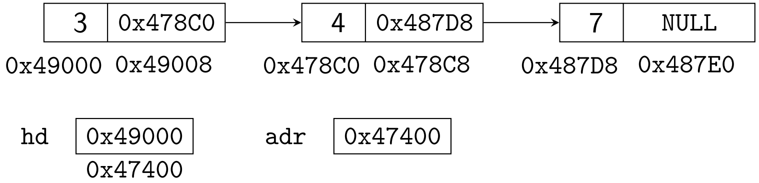
```
struct ilst {
  intlist_t nxt;
  int el;
};
```

Traversing linked list with address of pointer

When inserting/deleting into a linked list, must change link in cell *prior* to the one inserted/deleted

keep *address* of pointer to be changed (address of link field)

better than with address of previous element (may not exist)



```
intlist_t hd = cons(3, cons(4, cons(7, NULL)));
intlist_t *adr = &hd;
adr = &(*adr)->nxt;
```