Building a list library: obj.c
(lab 8)

- list stores data_t objects
as in lab 2: in C:

```
typedef struct data_type
{
    char e_name[16];
    int  e_id;
} data_t;
```

- in Obj-C, define data_t object:

```
@interface data_t : NSObject
{
    ... // { an int num
            ?
            NSString *str;
            (or name)
}
```

what they're called

not super important, but NSString is new

- need the following member functions:

naming conv.

- (id) init: (NSString *)_str:
  (int)_num;

- (int) num; //accesor

- (id) print: (FILE *)f;

- the print function prints

string	num	pair

- "%s %d\n", with this accessor function for NSString:

[str UTF8String]

↑ corresponds to (char *)

- what should print
  return type: (id)

```
return self;
```

- main() for lab 8 is given
- Usage:

- NSString usage:
  - declaration:

NSString *str;

(the temporary str that gets used to init data_t)

- recall every object is a pointer
- here we don't alloc str

- there's probably a way for NSString to "read itself"
- for familiarity, we'll read in a

```
char nam[16];
```

via:

```
scanf("%s%d", nam,
                &num);
```

- then, alloc & init the NSString:

```
str =
[[NSString alloc]
    initWithUTF8String nam];
```

"camelCase" as opposed to TitleCase → first letter lower case

- note usage of list add:

```
[list add:
    [(data_t*)[data_t alloc]
        init: str: num]];
```

- other list function usages:

```
[list reset];
```

(as before, reset current pointer)

```
while( ![list end]) {
    [[list data] print: stderr];
```

data_t object

data_t function

```
    [list next];
}
```

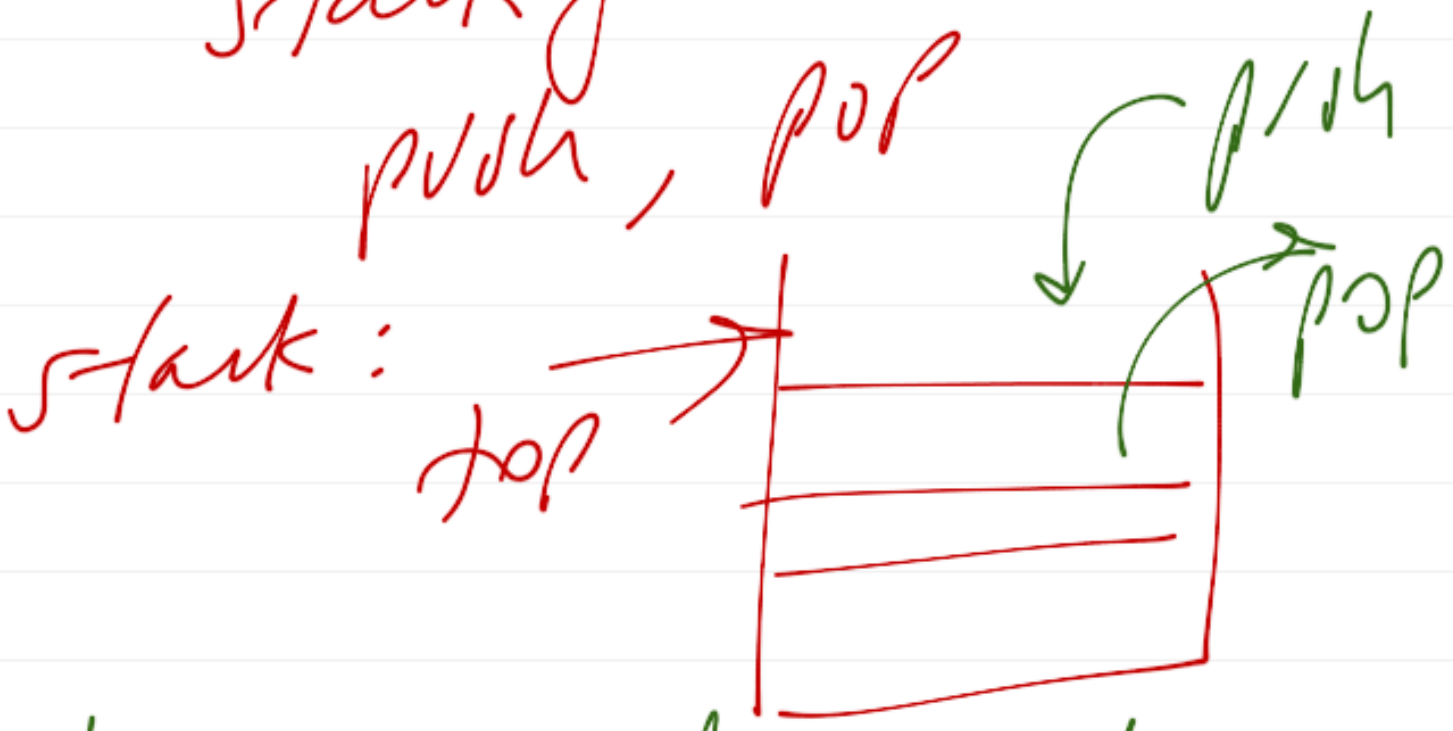
---

can do this as a for loop (more compact) →

```
for ([list reset];
     ![list end];
     [list next]) {
    [[list data] print: stdout];
}
```

- how to delete a list:
  use function 'pop'
  (like list_front_del)

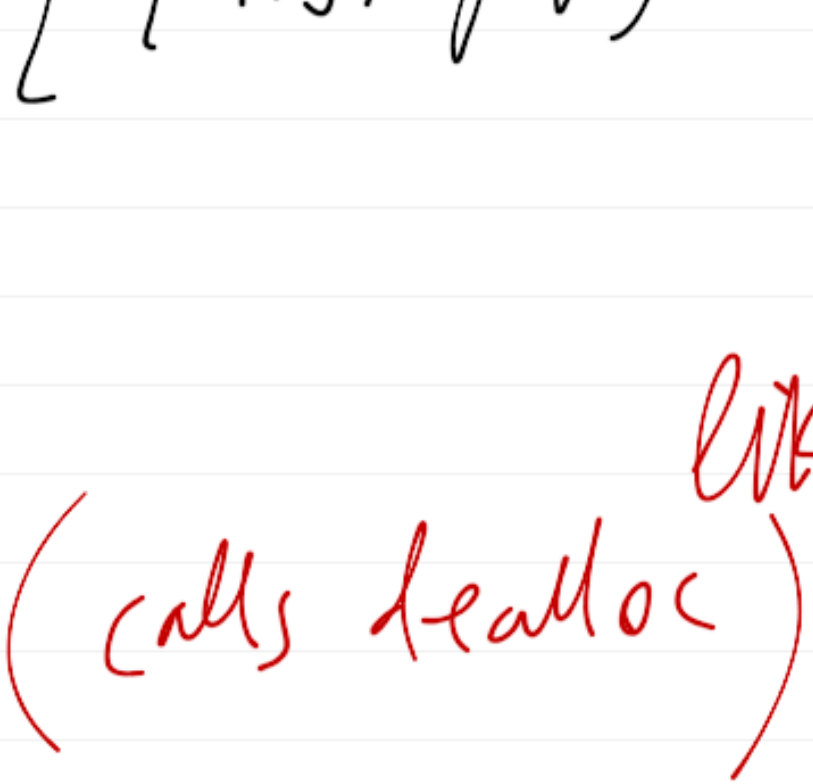
stack function names
push, pop



stack: a kind of data structure

- deleting list:

```
[list reset];
while(![list end])
    [[list pop] release];
```



- how to write the list class:
- need a link_t object as before
- link_t has as data members:

```
id data;
link_t *next;
```

- link_t methods:

```
-(id) init;
-(id) init: (id) _data;
-(id) init: (id) _data:
        (link_t *) _next;
```

Why the _ ?

Convention for arg. names so not to confuse with data members

- other link_t methods:

//accessor/mutators:

- (id) data;

- (link_t *) next;

- (void) setNext: (link_t *) _next;

- (void) setData: (id) _data;

@end (of interface of data_t)

- list_t interface:

data types:

link_t *first;

link_t *last;

link_t *current;

---

can't seem to access object internals via → pointer like in C; Use accessor/mutators

- list of functions:

```
- (id) init; //constructor
- (id) data; //accessor

//operators
- (void) add: (id) _data;
- (id) pop;
```

- members:
  - (void) reset;
  - (bool) empty;
  - (bool) end;
  - (void) next;

- material_t, pixel_t objects (rgb)

- material object similar to Sphere:

```
int cookie;
NSString *name;
pixel_t *ambient;
pixel_t *diffuse;
pixel_t *specular;
```

- functions:

-(id) init; // constructor

//accessor/mutators

(NSString *) setname:
(NSString *)_name;

(pixel_t *) setamb:
(pixel_t *)_amb;

can be void — no need to return when setting

```
//accessor
- (pixel_t *) getamsn;
...
//members
- (bool) matches:
        (NSString *)_name;
-(void) read: (FILE *)_file;
-(void) write: (FILE *)_file;
```

- the matches function:

user NSString isEqualToString function

```
-(bool)matches:(NSString *)_name
{
    return [name isEqualToString:
                _name];
}
```

NSString object (belong to this material_t)

```
- (void) read: (FILE *) _ih
{
    int count;
    char _name[256];
    count = fscanf(_ih, "%s",
        _name);
[self setname:
    [[NSString alloc]
initWithUTF8String: _name]];
```

use own mutator function

- continue reading in
  '{', pixel data
- use pixel_t to
  read itself ...
- when writing, use
  [name UTF8String]
  to write name as "0x"
- let pixels write themselves

- pixel_t object very similar to vec_t
- Constructors:
  - (id) init;
- accessor / mutator:
  - (bool) horizon;
  - (double) get: (int) i;
  - (void) set: (int) i : (double) di

```
// operators:
- (pixel_t *) plus:
                (pixel_t *) p;
- (pixel_t *) minus:
                (pixel_t *) p;
- (pixel_t *) scale:
                (double) d
```

// "friends"

- (void) read: (FILE *) in;

- (void) write: (FILE *) out:
(char *) msg;

- pixel_t data member still

drgb_t drgb;

as before, where

typedef double drgb_t[3];

(should really be using

NSImage,

NSColor

and/or NSNumber ??)