

Balancing HTML tags

`` I am boldface, `<i>` I am bold face and italic. `` I am just italic. `</i>`

`<i>`
`` ``

* error: mismatch
`<i>` ``

Balancing HTML tags

`` I am boldface, `<i>` I am bold face and italic. `` I am just italic. `</i>`

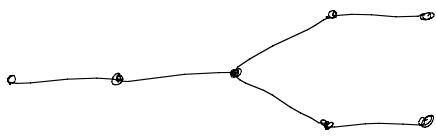
This is incorrect html (though most web browsers will interpret it correctly).

Instead you should add tags to have proper nesting

`</i>` `<i>`

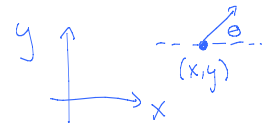
Example 3: Stacks in Graphics

Define a "programming language" for drawing simple figures like:



Suppose I can draw unit length lines starting at current position (x, y) and in any direction that is a multiple of 45° e.g. $\theta = 0^\circ, 45^\circ, 90^\circ, 135^\circ, 180^\circ, \dots$

The state of my pen is (x, y, θ) .



The initial state is $(0, 0, 0)$.

Let instructions be:

D - draw unit length line in direction θ (changes x, y)

R - turn 45° clockwise i.e. right (changes θ)

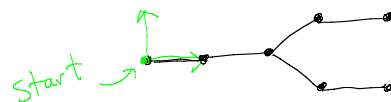
L - turn 45° counterclockwise i.e. left

[- push state

] - pop state

Starting state is $(x, y, \theta) = (0, 0, 0)$

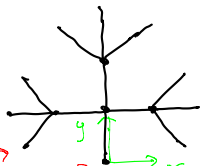
D - draw
 R - turn right
 L - turn left
 [- push state
] - pop state



DD [RDL D] LDRD

(2, 0, 0)

3D Graphics - OpenGL



finish start state is $(x, y, 0) = (0, 0, 0)$

Program for drawing the above shape

```
LLD [RRD [RD] [D] LD]
    [D [RD] [D] LD]
LLD [RD] [D] LD
```

sequence of instructions for

```
glPushMatrix();
glTranslatef(-10.0, -2.0, 0.0);
glRotatef(-90.0, 1.0, 0.0, 0.0); // rotate 90°
for (i = 0; i < N; i++){          // about axis
    glPushMatrix();              (1, 0, 0)
    for (j = 0; j < N; j++){
        drawSquare(1.0);
        glTranslatef(0.0, 1.0, 0.0);
    }
    glPopMatrix();
    glTranslatef(1.0, 0.0, 0.0);
}
glPopMatrix();
```

Example 4: Parsing a program (e.g. compilers)

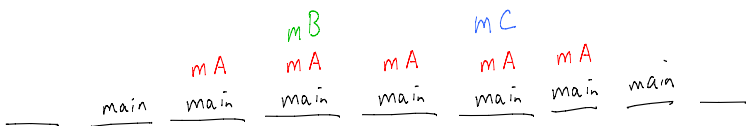
```
{ if (i > 3) then { if (j < 4) then
tmp = 2 else { if (bool) then
tmp = 4 else tmp = 5 } } else tmp = 1 }
```

The brackets $\{, \}$ illustrate the correct "parsing (but are not there). Inferring the brackets requires a stack. (Details omitted.)

Example 5 - "Call Stack"

```
class My Program {
void mA() {
    mB();
    mC();
}
void mB() { ... }
void mC() { ... }
void main() {
    mA();
}
}
```

```
class My Program {
void mA() {
    mB();
    mC();
}
void mB() { ... }
void mC() { ... }
void main() {
    mA();
}
}
```



```
class My Program {
void mA() {
    mB();
    mC();
}
void mB() { ... }
void mC() { ... }
void main() {
    mA();
}
}
```

Stack "Frame"

- contains all the information you need to run the method
- parameters passed
- local variables
- "return address"

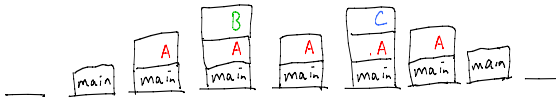
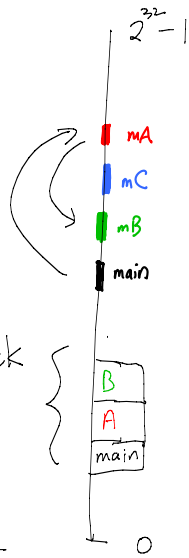


```

class My Program {
void mA() {
  mB();
  mC();
}
void mB() {...}
void mC() {...}
void main() {
  mA();
}
}

```

code has addresses too!



Summary : Stack ADT

push (element)
pop ()

Next lecture

Queue ADT