Semantics of Functions - C

1. The references created at the moment of a function call are automatically removed from the memory and the end of the execution of the body of the function.
2. There is only one name space for functions and variables.
3. C compilers also evaluate a functions arguments from left to right. [but is formally unspecified!]

Recursion

Question. Is Σ correct if f contains a call to f in body?
Answer. Clearly fine if stratified: $f \longrightarrow g \longrightarrow h$
But for recursive definitions, the definition of the Σ function can be circular:

```
static void f (final int x) {f(x);}
```

Then the definition of $\Sigma(f(x);, e, m, G)$ uses the value of $\Sigma(f(x);, e, m, G)$.
We must therefore find another method of defining the Σ function.

Recursion and Induction

The factorial function is defined inductively:

```
static int fact (final int x) {
 if (x == 0) return 1;
 return x * fact(x - 1);}
```

Compute *fact*(4)...

The following function is not defined inductively.

```
static int f (final int n) {
 if (n <= 1) return 1;
 if (n % 2 == 0) return (1 + f(n / 2));
 return 2 * f(n + 1);}
```

Compute $f(11)$...

Recursion and Induction - Cont. (The Ackerman Function)

```
static int ack (final int x, final int y) {
 if (x == 0) return 2 * y;
 if (y == 0) return 1;
 return ack(x - 1,ack(x,y - 1));}
```

cannot be defined using nested definitions by induction.

Formalization

Unwind the recursive function p (Just like while)

n steps by creating n functions: $p_1, \cdots, p_n$

which each do 1 step (based on next call)

and n^{th} "give up"

Then $\Sigma(p, e, m, G) = lim_n \ \Sigma(p_n, e, m, G)$.

Fact: Recursion and Iteration are equivalent!

Fixed Point Equations

The function fact (factorial) is the unique function $f : \mathbb{N} \to \mathbb{N}$ that satisfies the equation:

$$f = (x \mapsto \text{if } (\mathrm{x} == 0) \text{ then } 1 \text{ else } x * f(x-1)).$$

This equation has the form $f = G(f)$, so it is a *fixed point equation*.

Remark. Any fixed point equation always has at least one solution for the set of partial functions on $\mathbb{N}$.

There is an alternative method of defining the Σ function (equivalent to the def on Slide 5)

Records

A *Tuple* (Mathematically): is a function $f : \{0, 1, \cdots, n-1\} \rightarrow Set$

Example:

$(5, -3.1, 6.0, 'c', "foo", 13, 14, 'd')$

bad style!:

Instead of implicit labels use explicit labels!

Tuples (Programming Languages): usually of *named fields*

tuple of named fields $\equiv$ *records*

Example:

labels: {latitude, longitude, altitude}

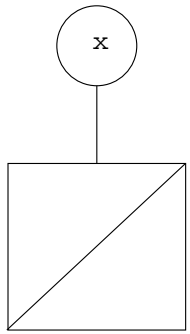
record: { latitude = 48.715, longitude = 2.208, altitude = 156}

Defining a Record (Java)

```
class Point {
 final double latitude;
 final double longitude;
 final double altitude;}
```

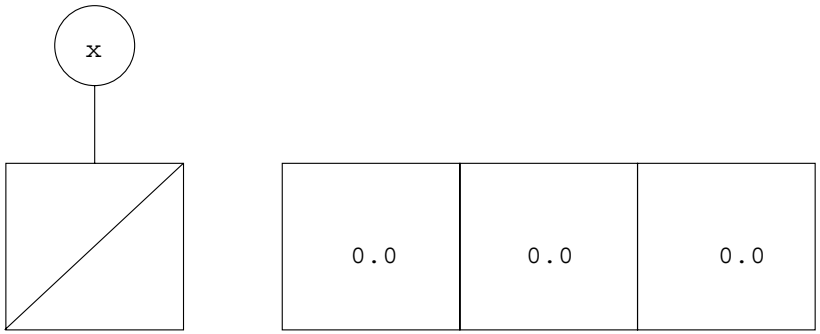
Records - Declaration

```
Point x;
```



r null

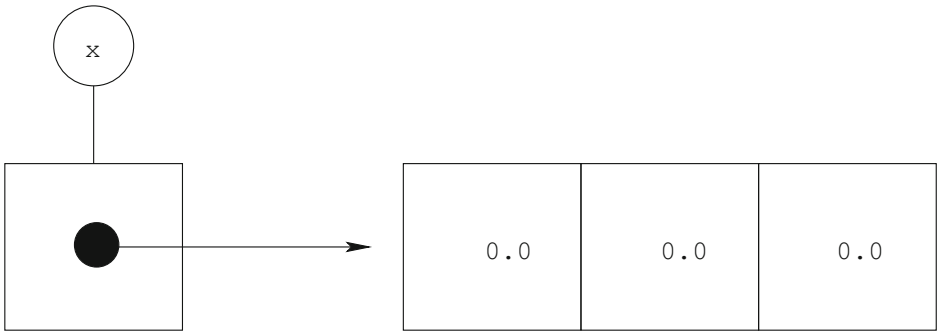
```
new Point();
```



The value of this expression is the reference
$r' = \{latitude = 0.0, longitude = 0.0, altitude = 0.0\}$

Records - Allocation

```
x = new Point();
```



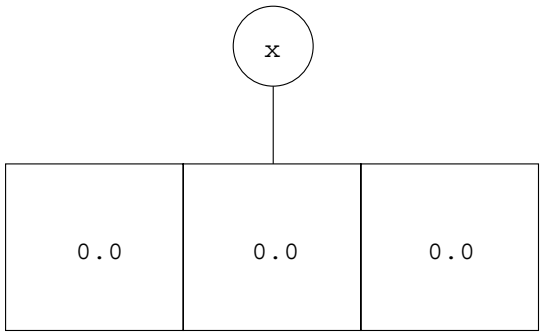
the environment is $e \oplus [x = r]$
the memory is $[r = r', r' = \{latitude = 0.0, longitude = 0.0, altitude = 0.0\}]$
You can also write:

```
Point x = new Point();
```

Records - Allocation - Cont.

The following statement creates the environment $[x = r']$ and the memory state $[r' = \{latitude = 0.0, longitude = 0.0, altitude = 0.0\}]$.

```
final Point x = new Point();
```



Records - Accessing Fields

```
x.latitude
x.thing : error at compile time
```