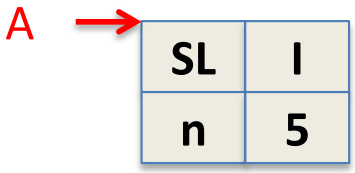
# Ambiente
# Run-time & Run-time Simulation

# Un esempio

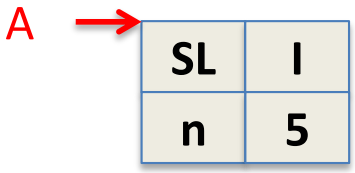
```
let  n = 5;;
let h = fun x -> n + x ;;
let rec f g n = if n = 1 then g(n) else n * f g (n-1);;
f h 2;;
```

# Run-time Stack




```
let  n = 5;;
let h = fun x -> n + x ;;
let rec f g n = if n = 1 then g(n) else n * f g (n-1);;
f h 2;;
```

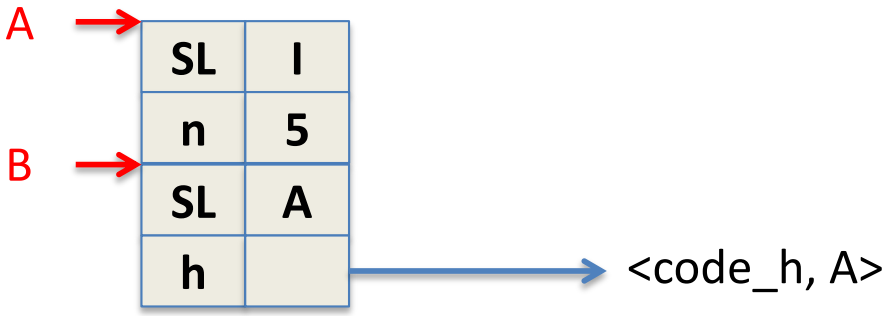
# Run-time Stack: simulation




Env_A(n) = 5
Env_A(m) = unbond
for all m != n

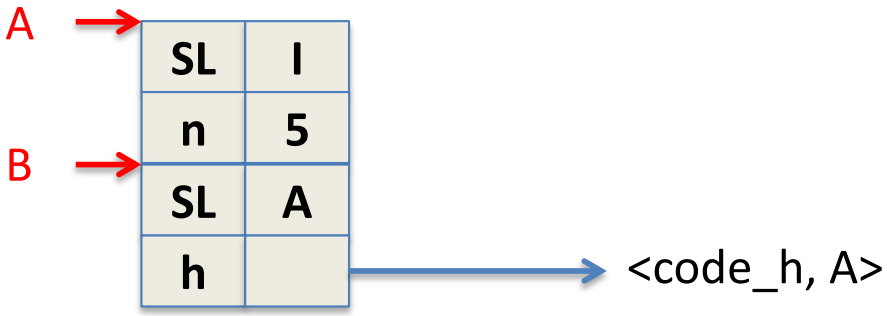
```
let  n = 5;;
let h = fun x -> n + x ;;
let rec f g n = if n = 1 then g(n) else n * f g (n-1);;
f h 2;;
```

# Run-time Stack




```
let  n = 5;;
let h = fun x -> n + x ;;
let rec f g n = if n = 1 then g(n) else n * f g (n-1);;
f h 2;;
```

# Run-time Stack: simulation

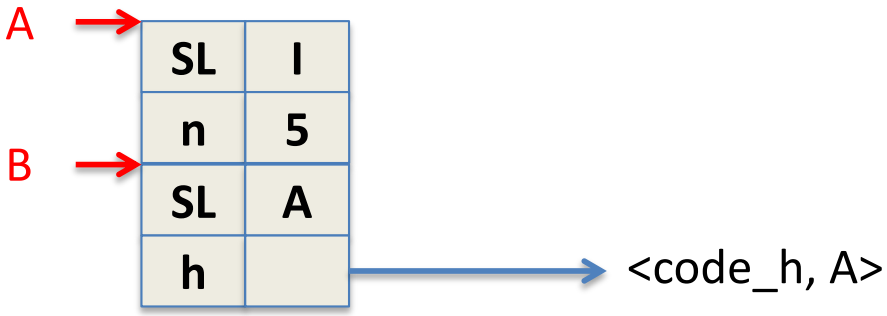



Env_A(n) = 5
Env_A(m) = unbond
for all m != n

Env_B (n) = 5
Env_B(h) = <code_h, Env_A>

```
let  n = 5;;
let h = fun x -> n + x ;;
let rec f g n = if n = 1 then g(n) else n * f g (n-1);;
f h 2;;
```

# Run-time Stack: simulation



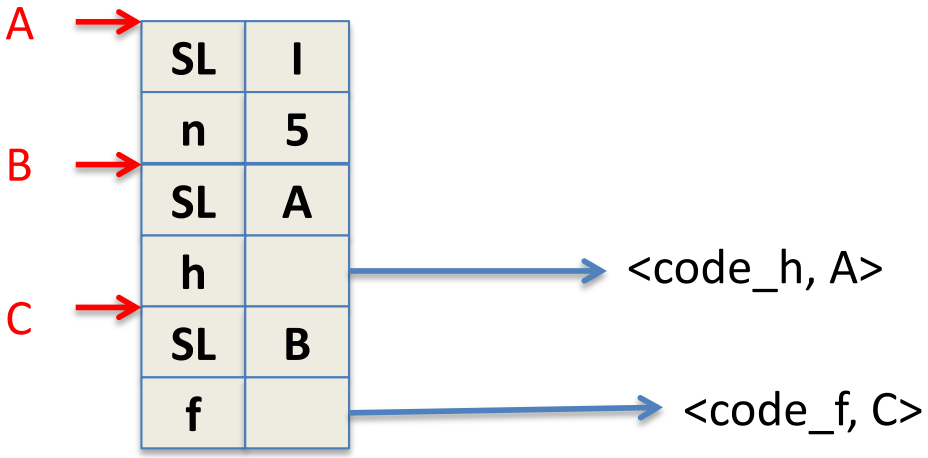

Env_A(n) = 5
Env_A(m) = unbond
for all m != n

Env_B (n) = 5
Env_B(h) = <code_h, Env_A>

```
:
Let(h, fun(x, Plus(n,x)), ebody) ->
 let fclosure = eval fun(x, Plus(n,x)), env_A) in
  eval ebody (bind env_A h fclosure)
(* fclosure= Closure(x, Plus(n,x), env_A) *)
```

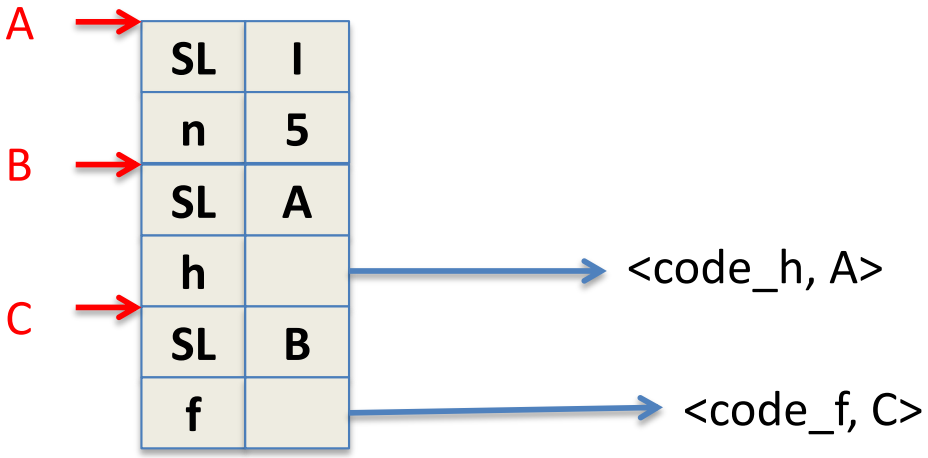
```
let  n = 5;;
let h = fun x -> n + x ;;
let rec f g n = if n = 1 then g(n) else n * f g (n-1);;
f h 2;;
```

# Run-time Stack




```
let  n = 5;;
let h = fun x -> n + x ;;
let rec f g n = if n = 1 then g(n) else n * f g (n-1);;
f h 2;;
```

# Run-time Stack: simulation



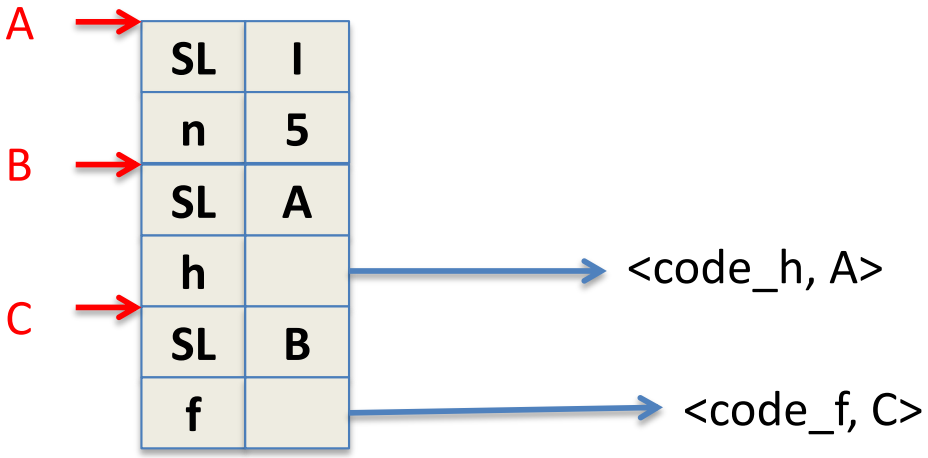

Env_A(n) = 5
Env_A(m) = unbond
for all m != n

Env_B (n) = 5
Env_B(h) = <code_h, Env_A>

Env_C(f) = <code_f, Env_C>
Env_C(h) = <code_h, Env_A>
Env_C(n) = 5

```
let  n = 5;;
let h = fun x -> n + x ;;
let rec f g n = if n = 1 then g(n) else n * f g (n-1);;
f h 2;;
```

# Run-time Stack: simulation




Env_A(n) = 5
Env_A(m) = unbond
for all m != n

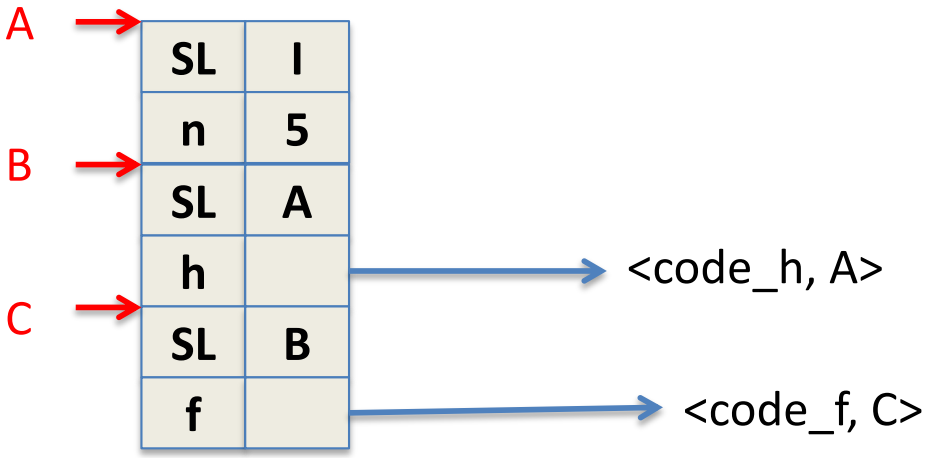
Env_B (n) = 5
Env_B(h) = <code_h, Env_A>

Env_C(f) = <code_f, Env_C>
Env_C(h) = <code_h, Env_A>
Env_C(n) = 5

Definizione ricorsiva:

```
let  n = 5;;
let h = fun x -> n + x ;;
let rec f g n = if n = 1 then g(n) else n * f g (n-1);;
f h 2;;
```

# Run-time Stack: simulation




Env_A(n) = 5
Env_A(m) = unbond
for all m != n

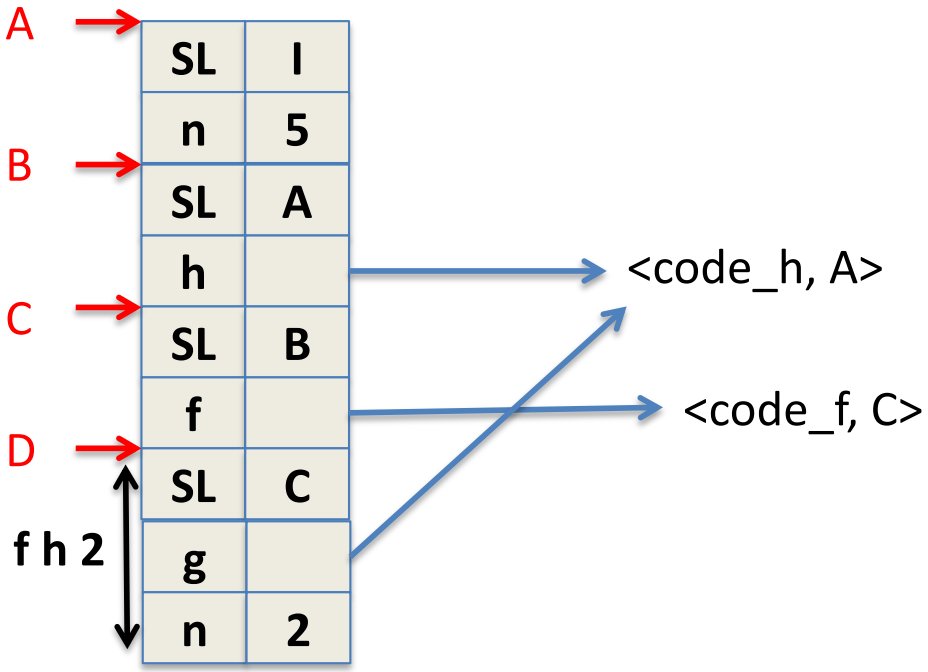
Env_B (n) = 5
Env_B(h) = <code_h, Env_A>

Env_C(f) = <code_f, Env_C>
Env_C(h) = <code_h, Env_A>
Env_C(n) = 5

```
:
| Letrec(f, par, fBody, letBody) ->
   let benv =
     bind(env_C, f, (Recfunval(f, par, fBody, env_C)))
      in eval(letBody, benv)
:
```

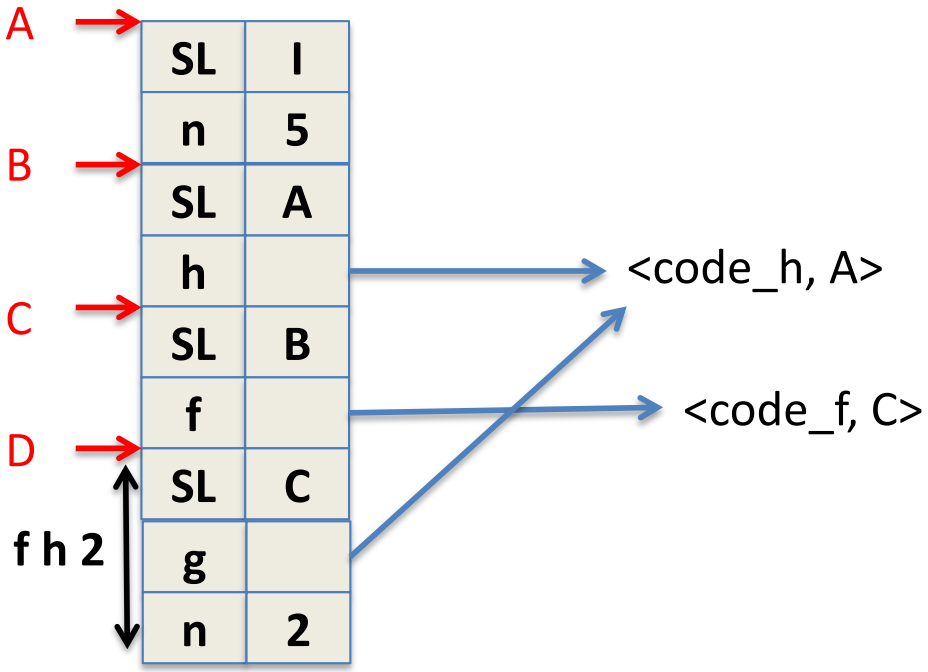
```
let  n = 5;;
let h = fun x -> n + x ;;
let rec f g n = if n = 1 then g(n) else n * f g (n-1);;
f h 2;;
```

# Run-time Stack




```
let  n = 5;;
let h = fun x -> n + x ;;
let rec f g n = if n = 1 then g(n) else n * f g (n-1);;
f h 2;;
```

# Run-time Stack:simulation




Env_A(n) = 5
Env_A(m) = unbond
for all m != n

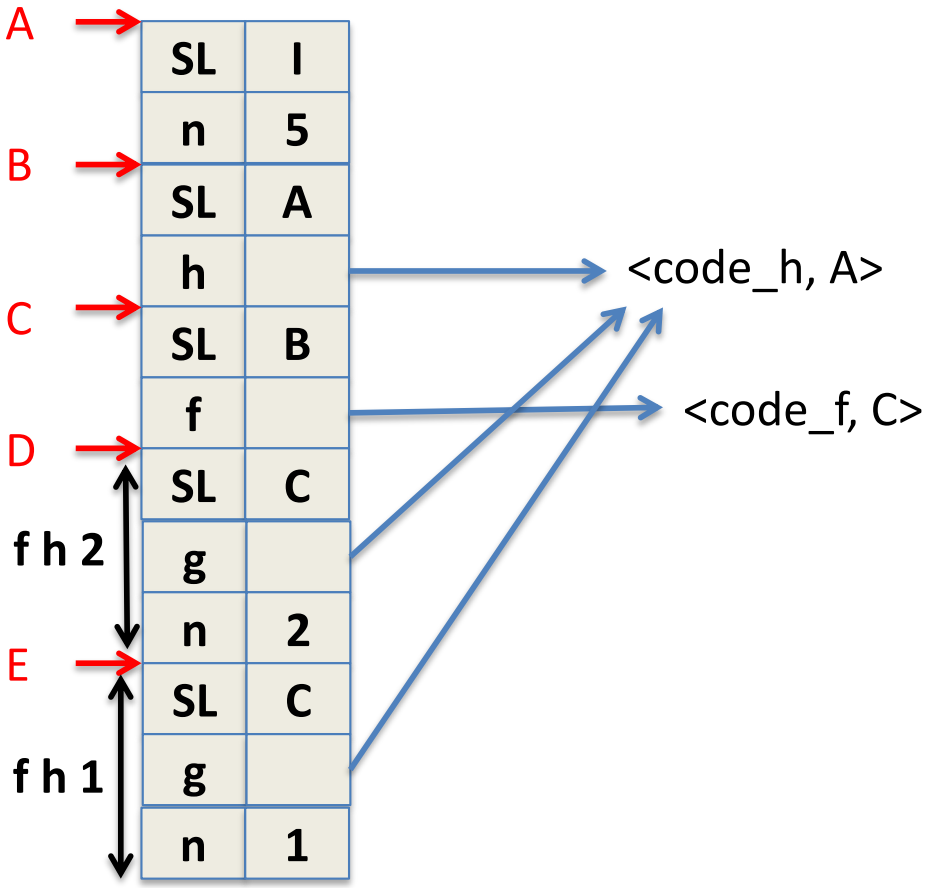
Env_B (n) = 5
Env_B(h) = <code_h, Env_A>

Env_C(f) = <code_f, Env_C>
Env_C(h) <code_h, Env_A>
Env_C(n) = 5

Env_D(g) = <code_h, Env_A>
Env_D(n) = 2
Env_D(f) = <code_f, Env_C>
Env_D(h) <code_h, Env_A>

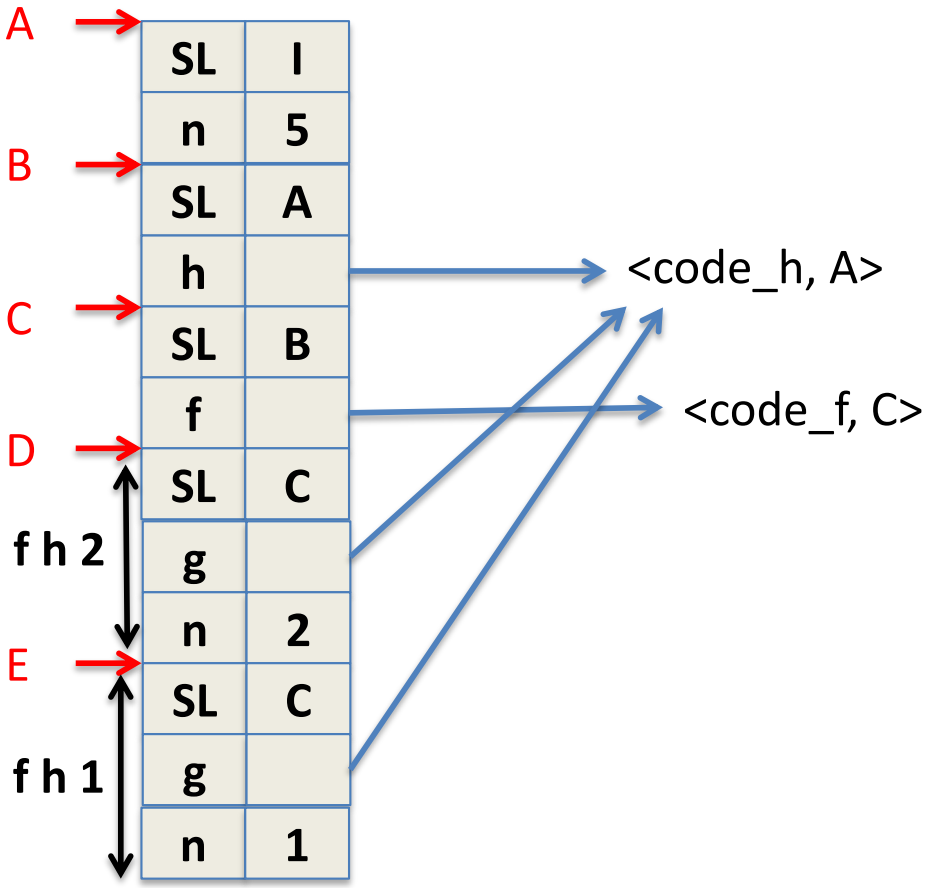
```
let  n = 5;;
let h = fun x -> n + x ;;
let rec f g n = if n = 1 then g(n) else n * f g (n-1);;
f h 2;;
```

# Run-time Stack




```
let  n = 5;;
let h = fun x -> n + x ;;
let rec f g n = if n = 1 then g(n) else n * f g (n-1);;
f h 2;;
```

# Run-time Stack: simulation




```
let  n = 5;;
let h = fun x -> n + x ;;
let rec f g n = if n = 1 then g(n) else n * f g (n-1);;
f h 2;;
```

Env_A(n) = 5
Env_A(m) = unbond
for all m != n

Env_B (n) = 5
Env_B(h) = <code_h, Env_A>

Env_C(f) = <code_f, Env_C>
Env_C(h) <code_h, Env_A>
Env_C(n) = 5

Env_D(g) = <code_h, Env_A>
Env_D(n) = 2
Env_D(f) = <code_f, Env_C>
Env_D(h) <code_h, Env_A>

Env_E(g) = <code_h, Env_A>
Env_E(n) = 1
Env_E(f) = <code_f, Env_C>
Env_E(h) <code_h, Env_A>