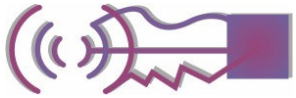
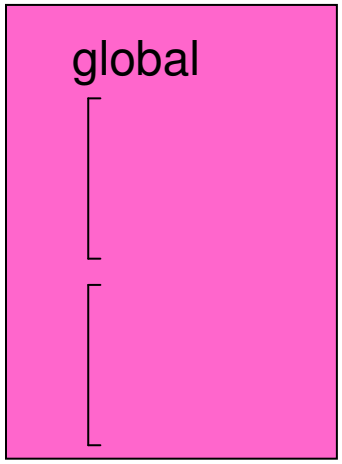
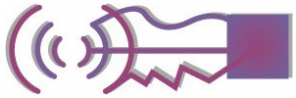


C3
C2 + recursión y valor de retorno

- rutinas con capacidad de recursion

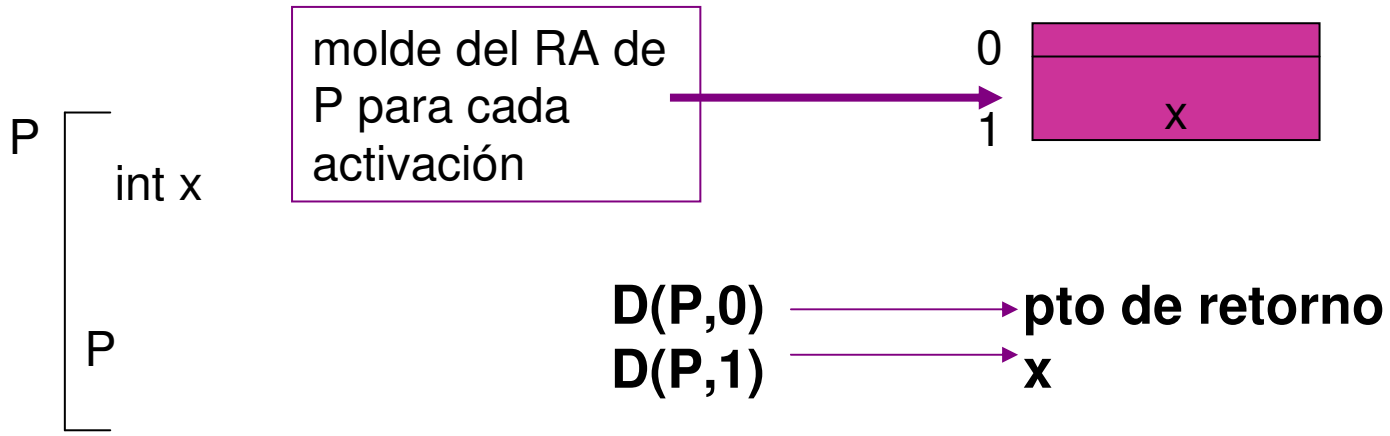
- rutinas con capacidad de devolver valores.

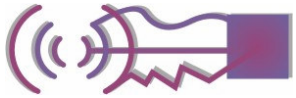




C3

- **no se sabe** cuantas instancias de cada unidad se necesitarán
- el compilador puede ligar cada variable con su desplazamiento
- **cada nueva invocación aloca un nuevo registro de activación y nuevas ligaduras**

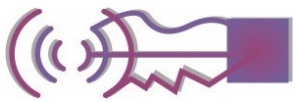




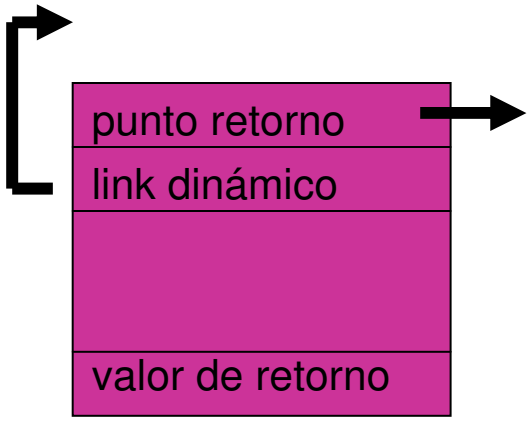
C3

nuevos elementos:

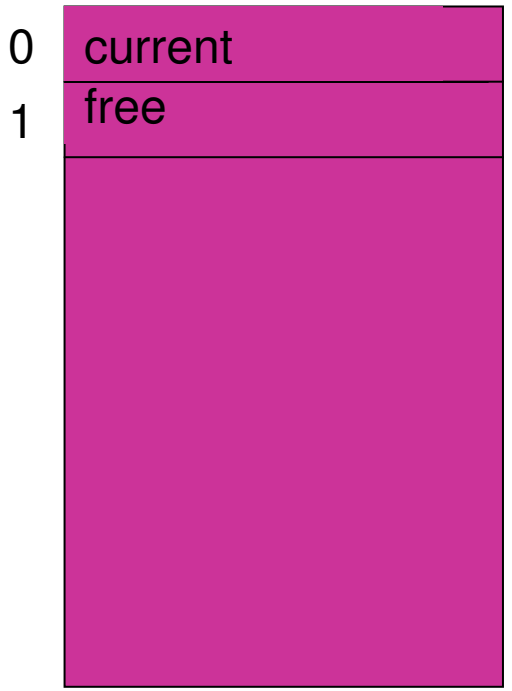
- **current:** dirección base del registro de activación de la unidad que se este ejecutando actualmente
- **free:** próxima dirección libre en la pila
- **link dinámico** puntero a la dirección base del registro de activación de la rutina llamadora
- **valor de retorno**

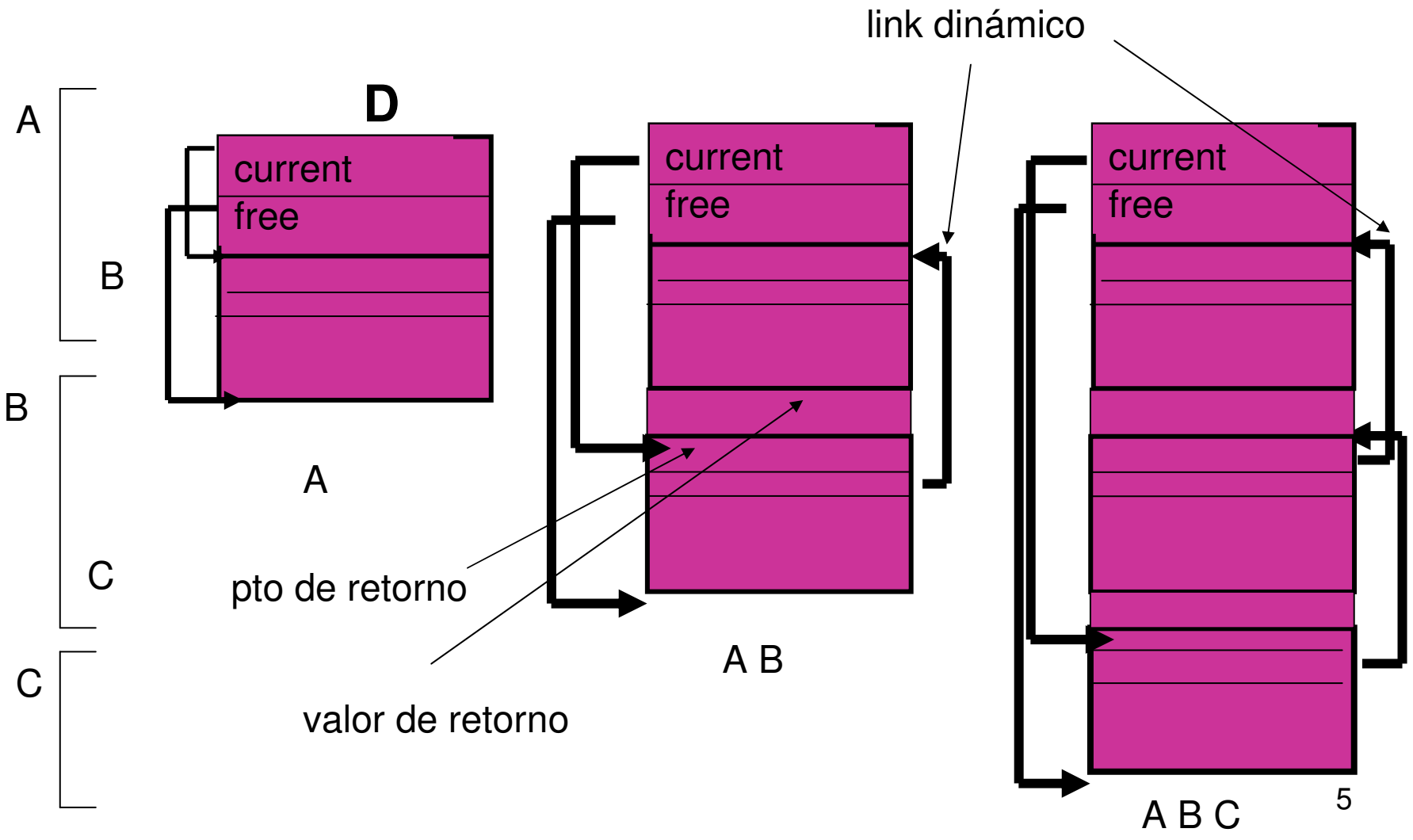


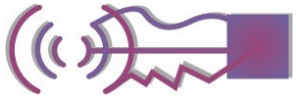
C3: moldes


punto retorno
link dinámico
valor de retorno

D


0
current
1
free


link dinámico
A
B
B
C
C
D
current
free
A
pto de retorno
valor de retorno
current
free
A B
current
free
A B C



C3: semántica del call - return

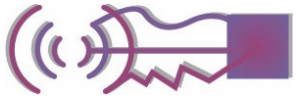
call

set 1,D[1]+1	aloca espacio para el v. de retorno
set D[1]+ip+5	salva el punto de retorno
set D[1]+1,D[0]	salva el link dinámico
set 0,D[1]	actualiza el current
set 1,D[1]+ size	actualiza el free
jump comienzo	transfiere el control a la unidad llamada

ip+5

set 1,D[0]	reestablece el free
set 0,D[D[0]+1]	reestablece el current
jump D[D[1]]	bifurca al punto de retorno

return

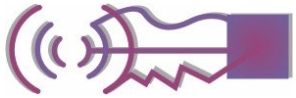


C4: estructura de bloque

C4´
permite que dentro de las sentencias compuestas aparezcan declaraciones locales

C4´´
permite la definición de una rutina dentro de otras rutinas.
(anidamiento de rutinas)

- controla el alcance de las variables,
- define el tiempo de vida de las variables
- divide el programa en unidades mas pequeñas.



C4': anidamiento via sentencias compuestas

- bloque:

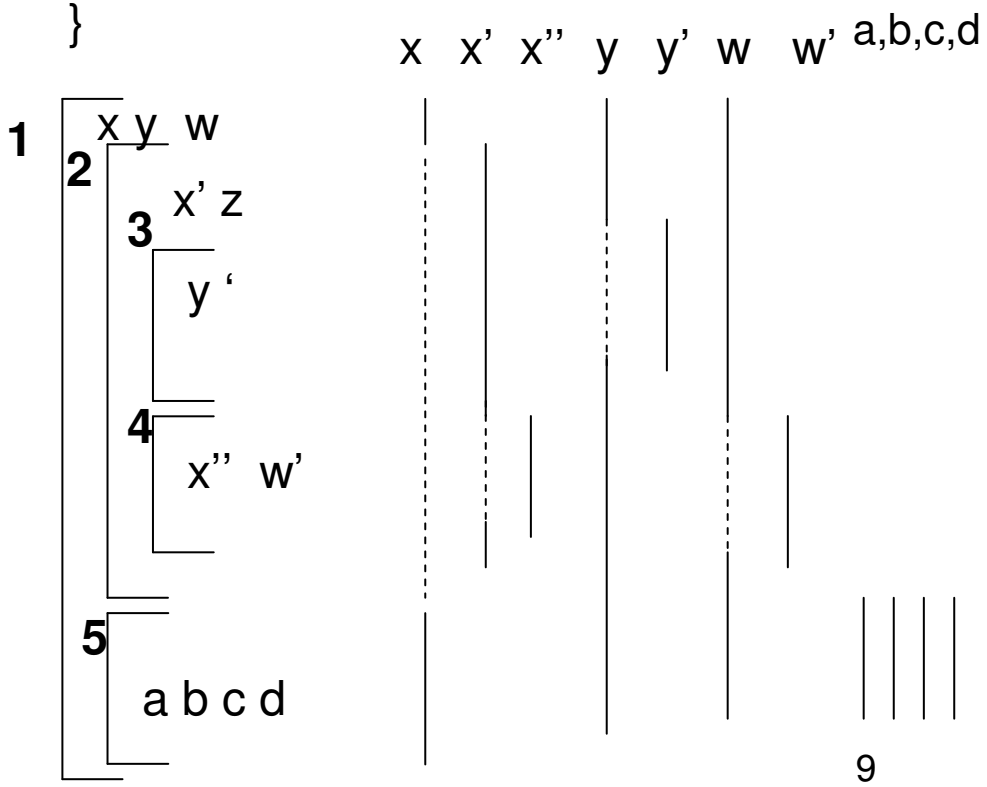
  {<lista de declaraciones>;<lista de sentencias>}

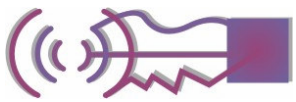
- las variables tienen alcance local
- si hay una nueva declaración de un nombre, la declaración interna enmascara la externa

C4’:sentencias compuestas

```
    int f();
{
    int x,y,w:
    while (……..)
    {
        int x,z:
        while (….)
        {
            int y;
        }
        if(…..)
        {
            int x, w;
        }
    }
```

```
if (…)
    {
        int a,b,c,d
    }
}
```





C4': sentencias compuestas

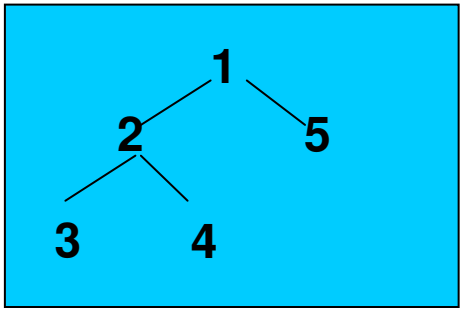
- implementación

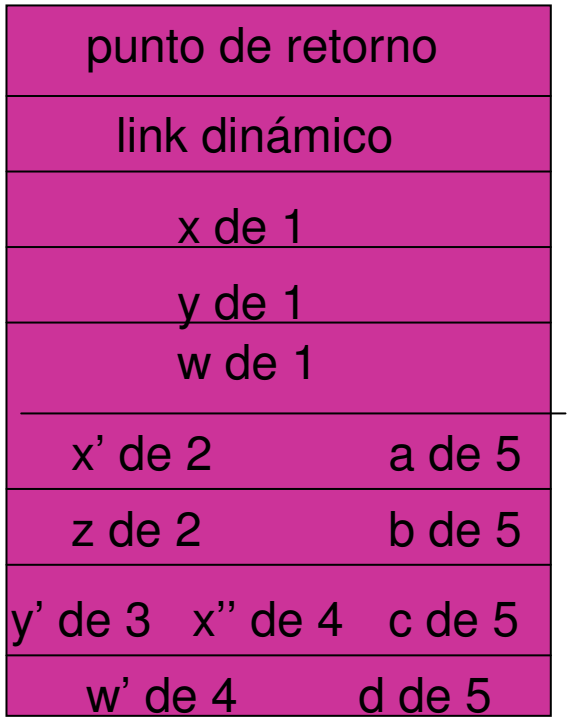
estático

dinámico

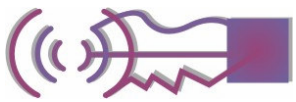
simple y eficiente en tiempo

eficiente en espacio





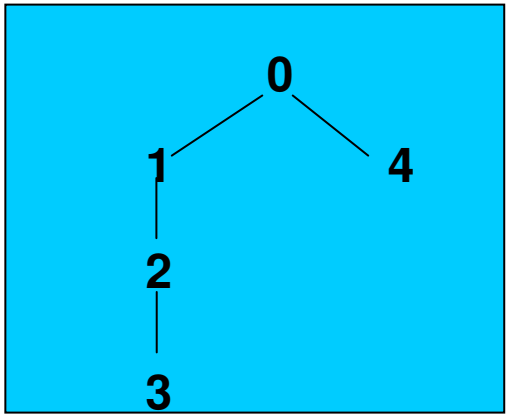
overlay para bloques disjuntos

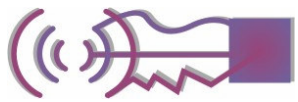


C4”: rutinas anidades

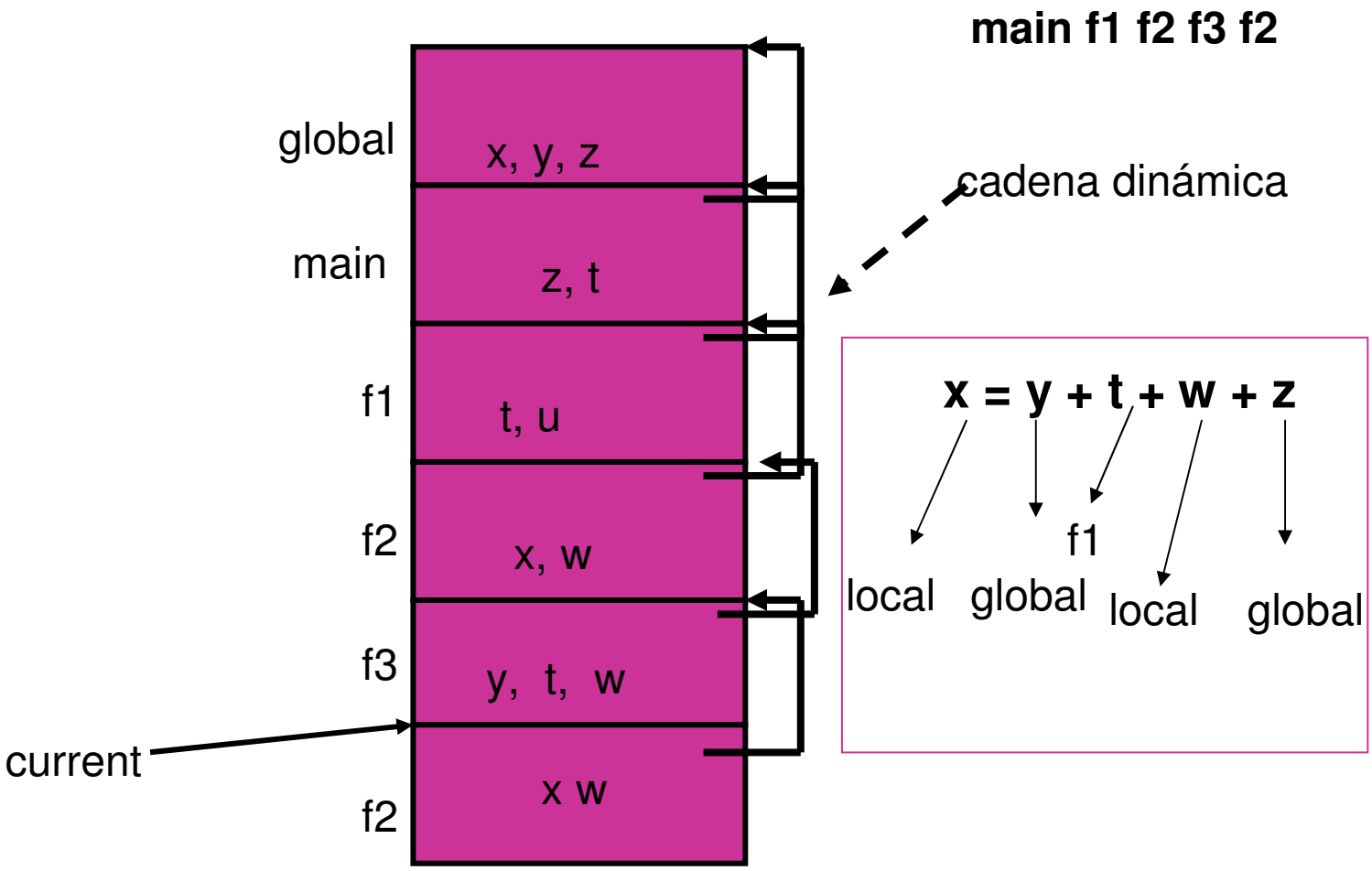
```
//file
int x,y,z;
f1()
{
    int t,u;
    f2()
    {
        int x,w;
        f3()
        {
            int y,w,t;
        }
         x = y + t + w + z;
    }
}
```

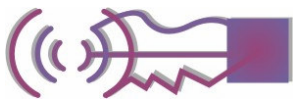
```
main ();
{
    int z,t;

}
//end file
```





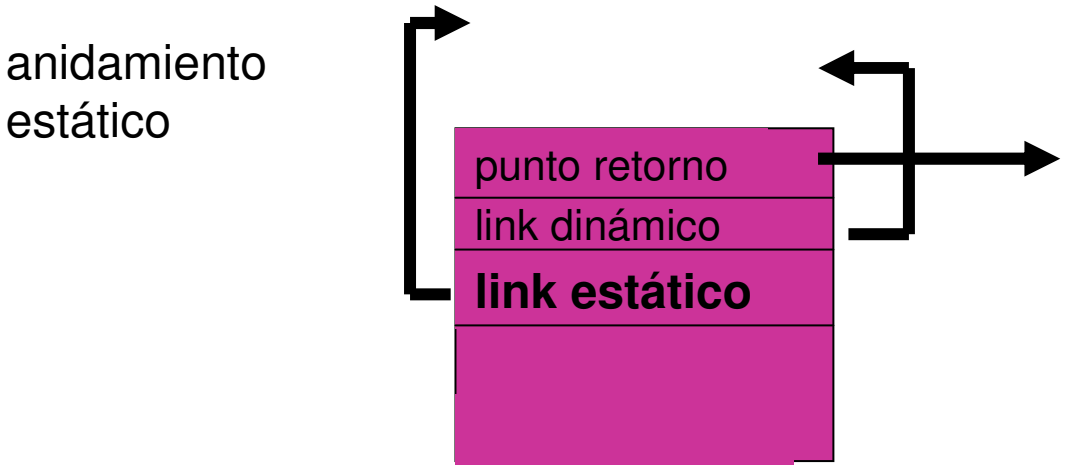
C4” rutinas anidadas acceso al ambiente no-local



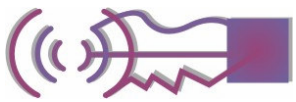


C4” acceso al ambiente no local

- link estático: apunta al registro de activación de la unidad que estáticamente la contiene



- la secuencia de links estáticos se denomina cadena estática



C4'' acceso al ambiente no local