Clase 5 – Pilas

Sigue con esquemas de Ejecucion

En este caso va de C3 a C6, estos son los casos basado en pila.

C3 (C2 + Recursion y valor de retorno): Utiliza rutinas con capacidad de llamarse a si mismas (recursion directa) o de llamar a otra rutina en forma recursiva (recursion indirecta). Ademas de utilizar rutinas con la capacidad de devolver valores, osea funciones.

Funcionamiento: El registro de activacion de cada unidad sera de tamaño fijo y conocido, pero no se sabra cuantas instancias de cada unidad se necesitaran durante la ejecucion. Igual que en C2 el compilador puede ligar cada variable con su desplazamiento(estatico) dentro del correspondiente registro de activacion.

La dirección donde se cargará el registro de activación, es dinámica, por lo tanto, la ligadura con la direcciones absolutas en la zona de Datos de la memoria, solo puede hacerse en ejecución. Cada nueva invocación aloca un nuevo registro de activación y se establecen las nuevas ligaduras ente el segmento de código y el nuevo registro de activación.

En ejecucion, cuando se invoca por primera vez a la rutina A si el RA de A se carga en la zona de memoria 1000 entonces la var X tomara la direccion absolta 1004 y cuando se invoca por segunda vez a la rutina A si el RA de A se carga en la zona de memoria 2050, entonces la var X tomara la direccion absoluta 2054.

En C3 se debe tener en cuenta que las unidades pueden devolver valores y esos valores no deberian perderse cuando se desactive la unidad, pero tambien que cuando la instancia actual de la unidad termine de ejecutarse, su registro de activacion no se necesitara mas, por lo tanto se puede liberar el espacio ocupado por su registro de activacion y dejar el espacio disponible para nuevos registros de activacion. *(Entiendo que se libera el espacio que contiene las variables locales y el punto de retorno, el valor que retorna la funcion se debe mantener por lo cual entiendo que este no se almacena en el RA de la funcion).*

Para manejar la alocacion dinamica necesitamos nuevos elementos:

.Valor de retorno: al terminar una rutina se desaloca su RA, por lo tanto la rutina llamante debe guardar en su RA el valor de retorno de la rutina llamada.

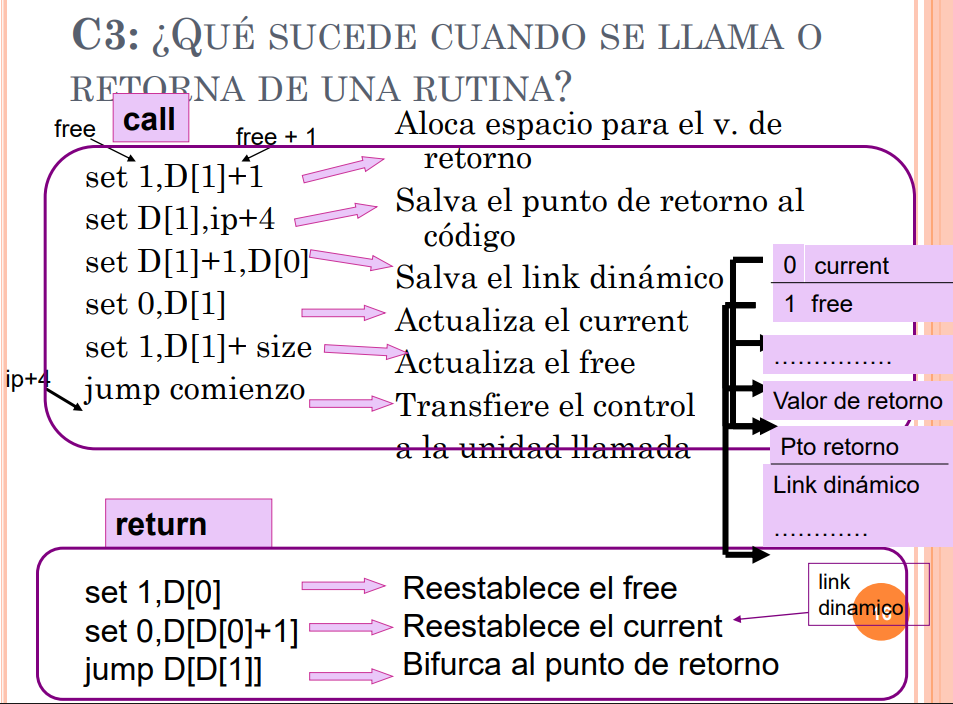
.Link dinamico: contiene un puntero a la direccion base del registro de activacion de la rutina llamadora.

.Current: direccion base del registro de activacion de la unidad que se esta ejecutando actualmente.

.Free: proxima direccion libre en la pila.

.Cadena dinamica: cadena de links dinamicos originada en la secuencia de registros de activacion activos. Representa la secuencia dinamica de unidades activadas.

.Moldes: lo entiendo como lo que tiene que tener cada metodo/funcion que seria el punto de retorno, el link dinamico y el valor de retorno (esto estaria en el espacio de codigo) y el current y free (en el esapacio de datos).



C4 (Estructura de bloque):

C4’: permite que dentro de las sentencias compuestas aparezcan declaraciones locales.

C4’’: permite la definicion de una rutina dentro de otras rutinas. (anidamiento de rutinas.

Estas caracteristicas conforman el concepto de estructura de bloque:

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

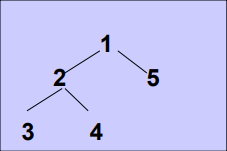
Los bloques pueden ser disjuntos (no tiene porcion comun) o anidados (un bloque esta completamente contenido en otro).

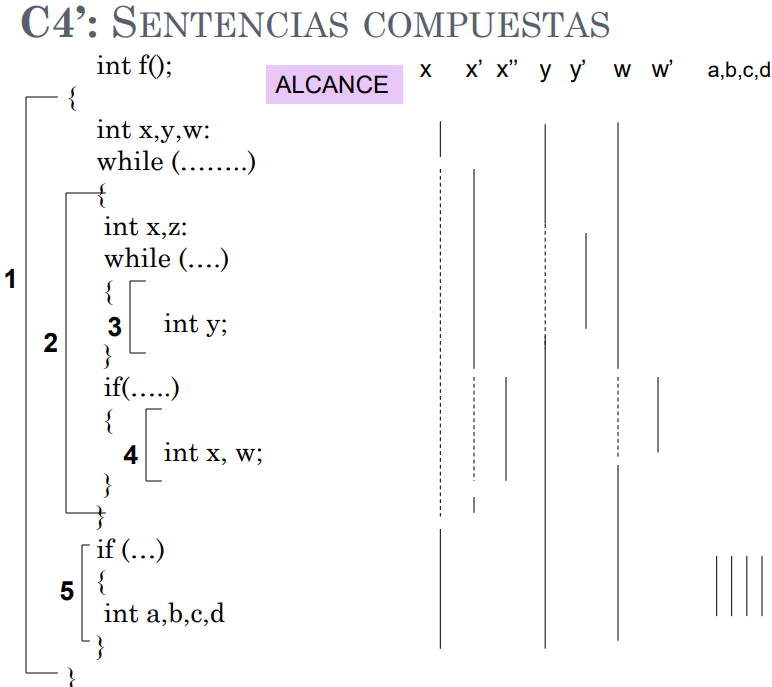
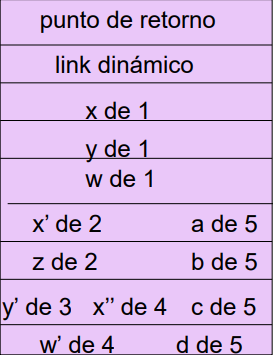
C4’ (Anidamiento via sentencias compuestas):

.Un bloque tiene forma de una sentencia compuesta: {<lista de declaraciones>;<lista de sentencias>}

.Las variables tienen alcance local, son visibles dentro de la sentencia compuesta, incluyendo cualquier sentencia compuesta anidada en ella.

.Si en el anidamiento, hay una nueva declaracion de un nombre, la declaracion interna enmascara la externa del mismo nombre.

Ejemplo C4’:



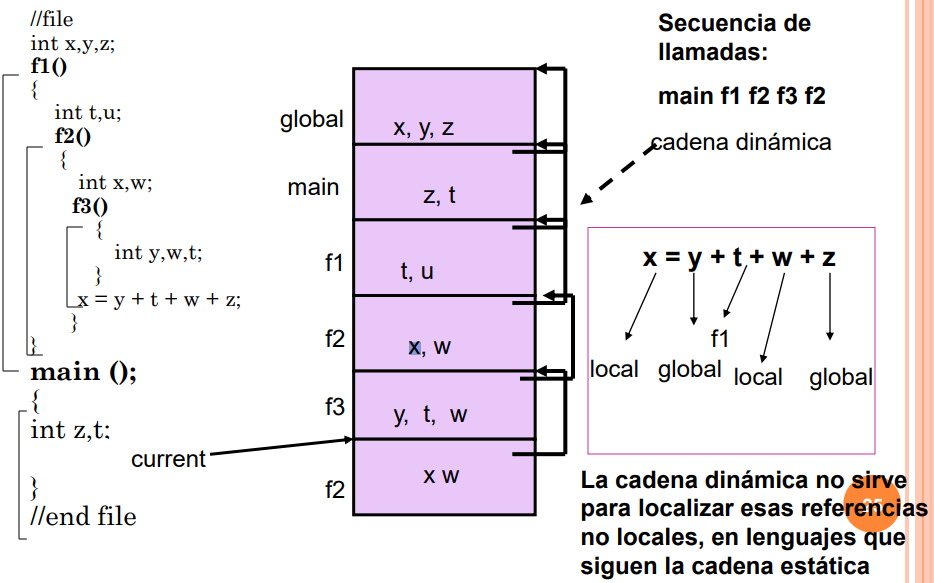
Alocacion:

Implemenatcion estatica: permite incluir las necesidades dentro del registro de activacion de la unidad a la que pertence. Esto lo vuelve simple y eficiente en cuanto al tiempo.

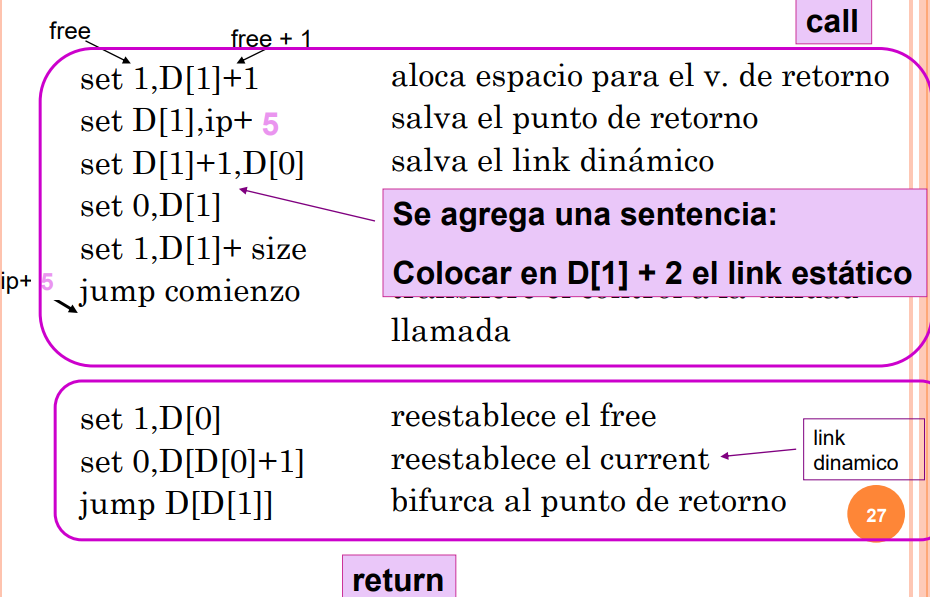
Implementacion dinamica: permite alocar el espacio dinamicamente cuando se ejecutan las sentencias. Esto lo vuelve eficente en cuanto al espacio.

C4’’ (Rutinas anidadas):

Acceso al ambiente no-local:



Link estatico: apunta al registro de activacion de la unidad que estaticamente la contiene. La secuencia de links estaticos se denomina cadena estatica.



***(El tema de la cadena dinamica es como lo que ya habia visto antes que seria como que la dinamica se queda con el ultimo valor que haya visto durante el recorrido entre las funciones y la estatica directamente toma el valor de la variable visible osea o es local o ya es del main).***

C5 (Datos mas dinamicos):

C5’: Registro de activacion cuyo tamaño se conoce cuando se activa la unidad (datos semidinamicos).

C5’’: Los datos pueden alocarse durante la ejecucion (datos dinamicos).

C5’ (Datos semidinamicos): Utiliza variables cuyo tamaño se conoce en compilacion.

.Arreglos dinamicos:

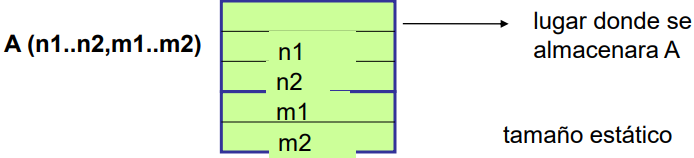
Type VECTOR is array (INTEGER range <>); define un arreglo con indice irrestricto

A: VECTOR (0..N); B: VECTOR (1..M);

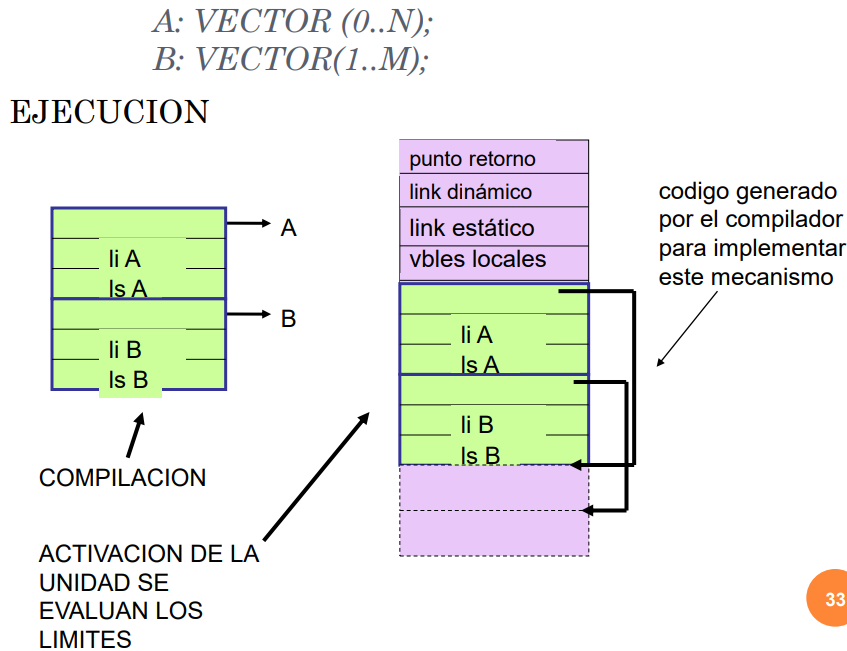
N y M deben ligarse a algun valor entero para que A y B puedan alocarse en ejecucion (referencia al ambiente no local o parametros).

Implementacion de arreglos dinamicos:

.Compilacion: se reserva lugar en el registro de activacion para los descriptores de los arreglos dinamicos.



Todos los accesos al arreglo dinamico son traducidos como referencias indirectas a traves del puntero en el descriptor, cuyo desplazamiento se determina estaticamente.



Ejecucion: el registro de activacion se aloca en varios pasos:

1. Se aloca el almacenamiento para los datos de tamaño conocido estaticamente y para los descriptores de los arreglos dinamicos.

2. Con la declaracion se calculan las dimensiones en los descriptores y se extiende el registro de activacion para incluir el espacio para la variable dinamica.

3. Se fija el puntero del descriptor con la direccion del area alocada.

C5’’ (Datos dinamicos): se alocan explicitamente durante la ejecucion mediante instrucciones de alocacion.

.El tiempo de vida no depende de la sentencia de alocacion, esta vivira mientras este apuntada.

.Se debe utilizar la heap para alocar.

C6 (Lenguajes dinamicos): se trata de aquellos lenguajes que adoptan mas reglas dinamicas que estaticas. Usan tipado dinamico y reglas de alcance dinamicas.

.Se podrian tener reglas de tipado dinamicas y de alcance estatico, pero en la practica las propiedades dinamicas se adoptan juntas. Una propiedad dinamica significa que las ligaduras correspondientes se llevan a cabo en ejecucion y no en compilacion.

Entonces:

.Variables estaticas C1 – C2.

.Variables semiestaticas o automaticas C3 – C4. Variables semidinamicas C5’. (Usan pila).

.Variables dinamicas C5’’. Tipos y alcance dinamico C6. (Usan heap).