Lenguajes de Programación

Dr. Carlos A. Coello Coello
Departamento de Computación
CINVESTAV-IPN
ccoello@cs.cinvestav.mx

Implementación de los Lenguajes Estructurados

- Esto significa que tenemos que dividir la representación del estado de un procedimiento en dos partes:

  1) Una parte fija del programa.

  2) Una parte variable del registro de activación.

  A continuación analizaremos las partes de un registro de activación.

Implementación de los Lenguajes Estructurados

- Al igual que en los lenguajes naturales, en los lenguajes de programación, una misma sentencia puede tener un significado diferente, dependiendo del contexto. Por ejemplo, en FORTRAN:

```
X = A(I)
```

Denota el acceso al elemento de un arreglo, si 'A' se declaró como tal, o podría ser una invocación a una función, si 'A' se declaró como una función.

Implementación de los Lenguajes Estructurados

- Por tanto, es crucial que el compilador sea capaz de interpretar correctamente los constructores en el contexto adecuado.

- ¿Cómo se relaciona esto con los registros de activación? En Pascal, los procedimientos son constructores que *definen contextos*. Esto significa que las sentencias y expresiones dentro de un procedimiento se interpretan en un *contexto* diferente que aquellas sentencias y expresiones que se encuentran fuera de él.

Implementación de los Lenguajes Estructurados

- Por lo tanto, para conocer el estado de la activación de un procedimiento, no es suficiente conocer la sentencia que se está ejecutando actualmente; se requiere también conocer el contexto en el cual debe ejecutarse dicha sentencia.

- El contexto se define mediante la *parte del ambiente* (conocida también como *apuntador al ambiente*) de un registro de activación.

Implementación de los Lenguajes Estructurados

- Un registro de activación consta, por tanto de dos partes:

1) La **parte del ambiente** (*environment part*, o ***ep***), que define el contexto a usarse para esta activación del procedimiento.

2) La **parte de la instrucción** (*instruction part*, o ***ip***), que designa la instrucción actual siendo ejecutada (o por ejecutarse) en esta activación del procedimiento.

Implementación de los Lenguajes Estructurados

- El contexto de las sentencias contenidas en un procedimiento es simple: consta simplemente de los nombres declarados en ese procecimiento, junto con los nombres declarados en los procedimientos circundantes.

- Debemos agregar la condición de que si algún nombre se declara en más de uno de estos procedimientos, entonces se verá la asociación más interna.

Implementación de los Lenguajes Estructurados

- Esto lo podemos expresar de una forma más procedural: si deseamos localizar un nombre, entonces lo buscamos en el ambiente local. Si se encuentra ahí, entonces realizamos la asociación correspondiente. En caso contrario, buscamos en el ambiente que se encuentra alrededor del actual.

- Este proceso continúa hasta encontrar la asociación correspondiente. Si ésta no se encuentra, entonces el nombre correspondiente se considera *no declarado* y se genera un mensaje de error.

Implementación de los Lenguajes Estructurados

- Para implementar procedimientos recursivos y las funciones de asignación dinámica de memoria de Algol y Pascal, se usa un registro de activación para guardar las variables locales y los parámetros pasados a cada procedimiento.

- Estos registros de activación se crean y se borran cuando entramos y salimos de los procedimientos correspondientes, lo que enciende la activación y la desactivación de almacenamiento para las variables locales.

Implementación de los Lenguajes Estructurados

- Esto proporciona acceso inmediato a la parte del contexto (las variables locales), puesto que esta información se almacena directamente en el registro de activación.

- La parte del ambiente del registro de activación también debe considerar el acceso a las partes no locales del contexto.

Implementación de los Lenguajes Estructurados

- Por tanto, la parte del ambiente tiene dos componentes:

1) El contexto local (las variables locales)

2) El contexto no local (las variables globales)

Implementación de los Lenguajes Estructurados

- De nuestro análisis anterior se desprende que se usa la siguiente representación para la activación de los procedimientos:

I. Parte fija del programa
II. Parte variable del registro de activación
    A. Parte de la instrucción
    B. Parte del ambiente
        1. Contexto local
        2. Contexto no local

Ahora veremos cómo se proporciona acceso no local.

Implementación de los Lenguajes Estructurados

- Cada variable es local a algún procedimiento. Por lo tanto, cada variable puede hallarse en el registro de activación de algún procedimiento.

- De esto se infiere que para proporcionar acceso a las variables no locales de un procedimiento, es necesario proporcionar acceso a los registros de activación de los procedimientos para los cuales estas variables son locales.

Implementación de los Lenguajes Estructurados

- ¿Cómo podemos proporcionar acceso al registro de activación del procedimiento circundante?
- La forma más simple es mantener un apuntador al mismo.
- Veamos un ejemplo (clase).

Implementación de los Lenguajes Estructurados

- El diagrama de contorno que dibujamos para este programa, muestra el contexto cuando se ha entrado a los procedimientos ‘a’, ‘b’ y ‘c’.

- Para localizar una variable, tal como N, comenzamos en el contexto local actual y empezamos a buscar la variable, hacia afuera a través de todos los contextos.

- Por tanto, debemos proporcionar una liga desde cada contorno hacia el contorno previo circundante.

Implementación de los Lenguajes Estructurados

- Estas ligas, en su conjunto, forman una cadena que conduce del registro de activación actual al más exterior (o sea, el entorno global).

- A esta liga se le denomina **liga estática**, SL (y a su cadena correspondiente se le llama **cadena estática**), porque refleja la estructura estática del programa (es decir, la forma en que se anidan los procedimientos).

Implementación de los Lenguajes Estructurados

- En clase veremos un esquema que ilustra la implementación típica de los registros de activación de los procedimientos.

- En dicho esquema, suponemos que estamos dentro del procedimiento ‘c’.

Implementación de los Lenguajes Estructurados

- En el esquema mostrado, puede verse que hay un registro de activación en la pila para cada uno de los procedimientos ‘a’, ‘b’ y ‘c’, y que la liga estática apunta desde cada registro hacia el registro anterior en la pila.

- Cada registro de activación contiene un campo con la **liga estática** (SL) que contiene la liga estática y espacio para las variables locales.

Implementación de los Lenguajes Estructurados

- El registro **EP** (apuntador al ambiente o *environment pointer*) apunta al registro de activación del contexto que está activo en ese momento. A éste lo denominaremos como el registro de activación actual.

- El registro **SP** (*stack pointer*, o apuntador de la pila) siempre apunta a la parte superior de la pila.

- Llamaremos al par de registros **IP-EP** el lugar de control, porque estos dos registros definen juntos la instrucción y el contexto actual de ejecución de la misma.

Implementación de los Lenguajes Estructurados

- Evidentemente, este formato no es el único posible. El formato adoptado por cada computadora en particular dependerá del conjunto de instrucciones de la misma y de algunas otras características de su arquitectura.

- Por ejemplo, en este caso hemos hecho que las ligas estáticas (incluyendo el registro EP) apunten hacia la base de los registros de activación (específicamente, al campo que contiene sus ligas estáticas). Esto facilita la construcción de la cadena de un registro de activación a otro, cuando intentamos acceder a una variable.

Implementación de los Lenguajes Estructurados

- Resulta evidente que sería muy ineficiente realizar la localización de variables de manera literal, pues se requeriría ubicar los nombres de cada variable en tiempo de ejecución, cada vez que éstas fueran referenciadas.

- De tal forma, es deseable contar con un mecanismo que haga más eficiente la localización de las variables, a fin de evitar este costo adicional.

Implementación de los Lenguajes Estructurados

- Como vimos anteriormente, el compilador de FORTRAN asigna ubicaciones fijas de memoria a cada variable y luego usa las direcciones de esas ubicaciones para acceder a las variables en tiempo de ejecución.

- Este mecanismo de asociación de las variables a ubicaciones específicas de memoria la realiza el compilador y se almacena en la denominada **tabla de símbolos.**

- En clase veremos un ejemplo de una tabla de símbolos.

Implementación de los Lenguajes Estructurados

- La tabla de símbolos se desecha al terminar el proceso de compilación, puesto que en ese punto, todas las referencias a las variables han sido reemplazadas por direcciones absolutas de memoria, por lo cual la tabla ya no se requiere.

- Este tipo de asociación estática que usa FORTRAN no funcionará en Pascal, porque a las variables en este lenguaje se les asignan posiciones de memoria en tiempo de ejecución, además de que pueden haber varias instancias de la misma variable a la vez.

Implementación de los Lenguajes Estructurados

- Para entender mejor esto, regresemos al programa que vimos anteriormente.

- En este caso, la variable ‘val’ es local al procedimiento ‘c’, por lo cual está contenida en el registro de activación actual y no tenemos que seguir la cadena estática para obtenerla.

- La variable ‘sum’ está contenida en el procedimiento ‘b’ el cual es circundante al procedimiento ‘c’. Por tanto, debemos seguir la cadena estática durante una distancia de una liga para obtener el registro de activación que contiene ‘sum’.

Implementación de los Lenguajes Estructurados

- Finalmente, 'N' se declaró en el procedimiento 'a', por lo cual tenemos que seguir la cadena estática una distancia de dos ligas para obtener el registro de activación que contiene a 'N'.

- De aquí se infiere que si conocemos la "distancia" del uso de una variable a su declaración, entonces podemos atravesar esa cantidad de ligas de la cadena estática para obtener el ambiente de definición de dicha variable. Por lo tanto, tenemos que ver la forma en que podemos obtener esta distancia.

Implementación de los Lenguajes Estructurados

- Resulta claro que esta distancia depende de qué tan profundamente anidada se encuentre la variable a la hora de usarla, con respecto al ambiente donde se declaró. Esto es, si el procedimiento donde se usa está dos niveles más anidado que el procedimiento donde se declaró, entonces la distancia es de dos.

- El uso de cierta terminología de lenguajes de programación nos ayudará a definir esto.

Implementación de los Lenguajes Estructurados

- Llamaremos al número de niveles de **procedure-end** que contienen el uso o la declaración de un nombre, el **nivel de anidamiento estático**.

- Por ejemplo, y usando nuevamente el programa que vimos anteriormente, el nivel de anidamiento estático de la declaración de 'N' es uno, el de la declaración de 'sum' es dos y el de la declaración de 'val' es tres. El uso de 'Data' en el procedimiento 'c' es tres.

Implementación de los Lenguajes Estructurados

- Así mismo, la **distancia estática** entre dos constructores es la diferencia entre sus niveles de anidamiento estático.

- Por ejemplo, puesto que el nivel de anidamiento estático del uso de 'Data' en el procedimiento 'c' es tres y el nivel de anidamiento estático de la declaración de 'Data' (en el procedimiento 'b') es de dos, la distancia estática entre el uso y la declaración de 'Data' es de uno.

Implementación de los Lenguajes Estructurados

- Debe resultar claro que la distancia estática entre el uso de una variable y su declaración nos indica cuántas ligas estáticas deben recorrerse para llegar al registro de activación que contiene la variable en cuestión.

- Cabe señalar que resulta muy fácil para el compilador llevar un registro de esta información: necesita saber siempre el nivel de anidamiento estático del procedimiento que está compilando de manera que incremente este número cada vez que encuentre un **procedure** y lo decremente cuando encuentre un **end**.

Implementación de los Lenguajes Estructurados

- Así, cada vez que el compilador procese una declaración, debe registrar en la tabla de símbolos para la variable declarada, el nivel de anidamiento estático (**snl**) de su declaración.

- En clase veremos un ejemplo de esto.

Implementación de los Lenguajes Estructurados

- Para cada sentencia que contiene una variable, la diferencia entre el nivel de anidamiento estático de dicha sentencia y el nivel de anidamiento estático dado para esa variable en la tabla de símbolos es la distancia al registro de activación que contiene la variable.

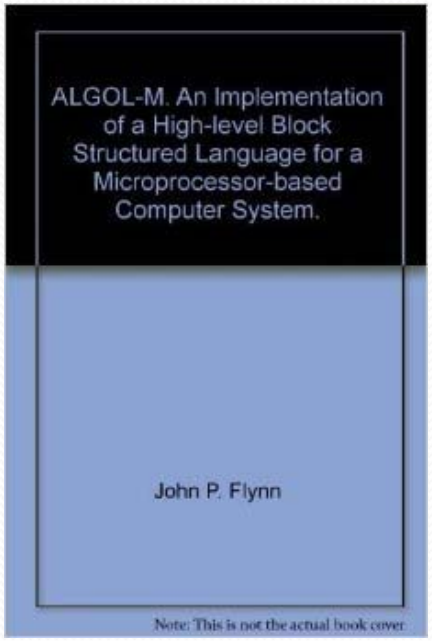
- Destaca el hecho de que no es suficiente conocer el registro de activación en el cual reside una variable en particular. También se necesita conocer su posición dentro del registro de activación.

Implementación de los Lenguajes Estructurados

- Ese es el propósito de la columna que dice “desplazamiento” en la tabla de símbolos mostrada en clase. Este valor nos indica la distancia desde la base del registro de activación hasta la variable que buscamos.

- De esto se desprende que estamos usando dos coordenadas para acceder a las variables. La primera coordenada es el nivel de anidamiento estático de la declaración de la variable, que nos da acceso al registro de activación donde se encuentra la variable. La segunda coordenada es el **desplazamiento** o posición de la variable dentro de ese registro de activación.

Implementación de los Lenguajes Estructurados



- Se hace notar que, aunque la posición del registro de activación puede variar en tiempo de ejecución, la posición de la variable dentro del registro de activación se mantiene fija.

Implementación de los Lenguajes Estructurados

- Muchos objetos en los lenguajes estructurados se direccionan usando estas dos coordenadas:

1. Un **apuntador al ambiente** (EP), que es el nivel de anidamiento estático en este caso. Esto nos da acceso al registro de activación del ambiente donde se definió el objeto.

2. Un **desplazamiento relativo** (el de la variable, en este caso), que nos da acceso al objeto deseado dentro del registro de activación correspondiente.

Implementación de los Lenguajes Estructurados

- Puesto que se necesitan estas dos coordenadas para localizar una variable, es natural que se requieran dos pasos para acceder a una variable:

(1) Debe localizarse primero el registro de activación para el ambiente de definición de la variable.

(2) Debe localizarse la variable dentro de este registro de activación.

Implementación de los Lenguajes Estructurados

- Más específicamente:

1. En tiempo de ejecución, recorreríamos la cadena estática durante el número de ligas dados por la distancia estática, a fin de obtener el registro de activación en el que reside la variable. La distancia estática entre el uso y la declaración de la variable es una constante calculada por el compilador.

2. La dirección de la variable se obtiene añadiendo el desplazamiento fijo de la variable a la dirección obtenida en el paso 1. Este desplazamiento también es una constante calculada por el compilador.

Implementación de los Lenguajes Estructurados

- Veremos ahora un ejemplo, usando los formatos que vimos anteriormente.

- Supongamos que queremos acceder a la variable local 'val' desde dentro del procedimiento 'c'. Puesto que es local, la distancia estática a su registro de activación es cero, de manera que no tenemos que seguir la cadena.

- Por lo tanto, la dirección de 'val' está dada por EP+1, donde 1 es el desplazamiento fijo almacenado en la tabla simbólica que vimos anteriormente.

Implementación de los Lenguajes Estructurados

- El caso general se puede representar de la forma siguiente:

fetch M[EP + offset (*v*)]

donde *v* es cualquier variable local y 'offset (*v*)' es el desplazamiento constante almacenado para *v* en la tabla de símbolos.

Implementación de los Lenguajes Estructurados

- Ahora consideremos acceder a 'sum', que se encuentra a una distancia estática de uno. Esto significa que debe recorrerse una liga de la cadena estática para obtener la dirección del registro de activación de 'sum'. Esta dirección la mantendremos en un registro temporal AP (apuntador al registro de activación):

```
AP := M[EP];          recorrer la liga estática
fetch M[AP+1]         acceder la variable
```

puesto que 1 es el desplazamiento de 'sum'.

Implementación de los Lenguajes Estructurados

- Finalmente, consideremos el código necesario para acceder a 'N', que se encuentra a una distancia estática de 2.
- En este caso, se deben recorrer dos ligas de la cadena estática:

```
AP := M[EP];        recorrer la primera liga estática
AP := M[AP];        recorrer la segunda liga estática
fetch M[AP+1]       acceder la variable
```

puesto que el desplazamiento de 'N' es 1.

Implementación de los Lenguajes Estructurados

- Resumamos los pasos para acceder a una variable. Si la variable se encuentra a una distancia estática de cero, entonces puede ser accedida directamente usando 'M[EP + offset(v)]'. Si se encuentra a una distancia estática mayor que cero (*sd* > 0), entonces los pasos requeridos son:

AP := M[EP];	recorrer la primera liga estática
(*sd*-1) × AP := M[AP];	recorrer la segunda liga estática
fetch M[AP + offset(v)]	acceder la variable

donde *sd* es la distancia estática.

Implementación de los Lenguajes Estructurados

- El significado de la segunda línea del segmento de código del acetato anterior es que se repiten (*sd*-1) veces la instrucción 'AP := M[AP]'.

- Si *sd*=1, esta instrucción se omite y si *sd*=0, se omiten las dos primeras instrucciones y se usa EP en vez de AP.

- Se hace notar que cada liga estática que se recorre requiere una referencia de memoria (ignorando cualquier referencia de memoria requerida para decodificar las instrucciones mismas), de forma que se requieren *sd* referencias de memoria para obtener el registro de activación correspondiente.

Implementación de los Lenguajes Estructurados

- Se requiere una referencia de memoria adicional para leer o escribir la variable, una vez que se localiza, de tal forma que el número total de referencias de memoria que se requieren es: *sd*+1 (el valor exacto varía de acuerdo a la computadora utilizada).

- De tal forma, puede ser muy costoso acceder a variables que se encuentran a una enorme distancia estática, si bien el acceso a variables locales es muy económico.