Compiladores:
Ambientes para Ejecución

Pontificia Universidad Javeriana Cali
Ingeniería de Sistemas y Computación
Prof. María Constanza Pabón

Procedimientos

- **Definición de un Procedimiento**: es una declaración que asocia un identificador –el **Nombre del Procedimiento**-, con un conjunto de instrucciones –el **Cuerpo del Procedimiento**-. Los identificadores de los parámetros que aparecen en la declaración del procedimiento son los **Parametros Formales**.

- Los procedimientos que retornan valores normalmente son llamados **Funciones**.

- Cuando el nombre del procedimiento aparece en la ejecución de una instrucción, decimos que el procedimiento esta siendo L**lamado,** y se ejecuta el cuerpo del procedimiento. Los argumentos que pasan al procedimiento son los **Parámetros Actuales**.

Asumpciones sobre el flujo de control entre procedimientos

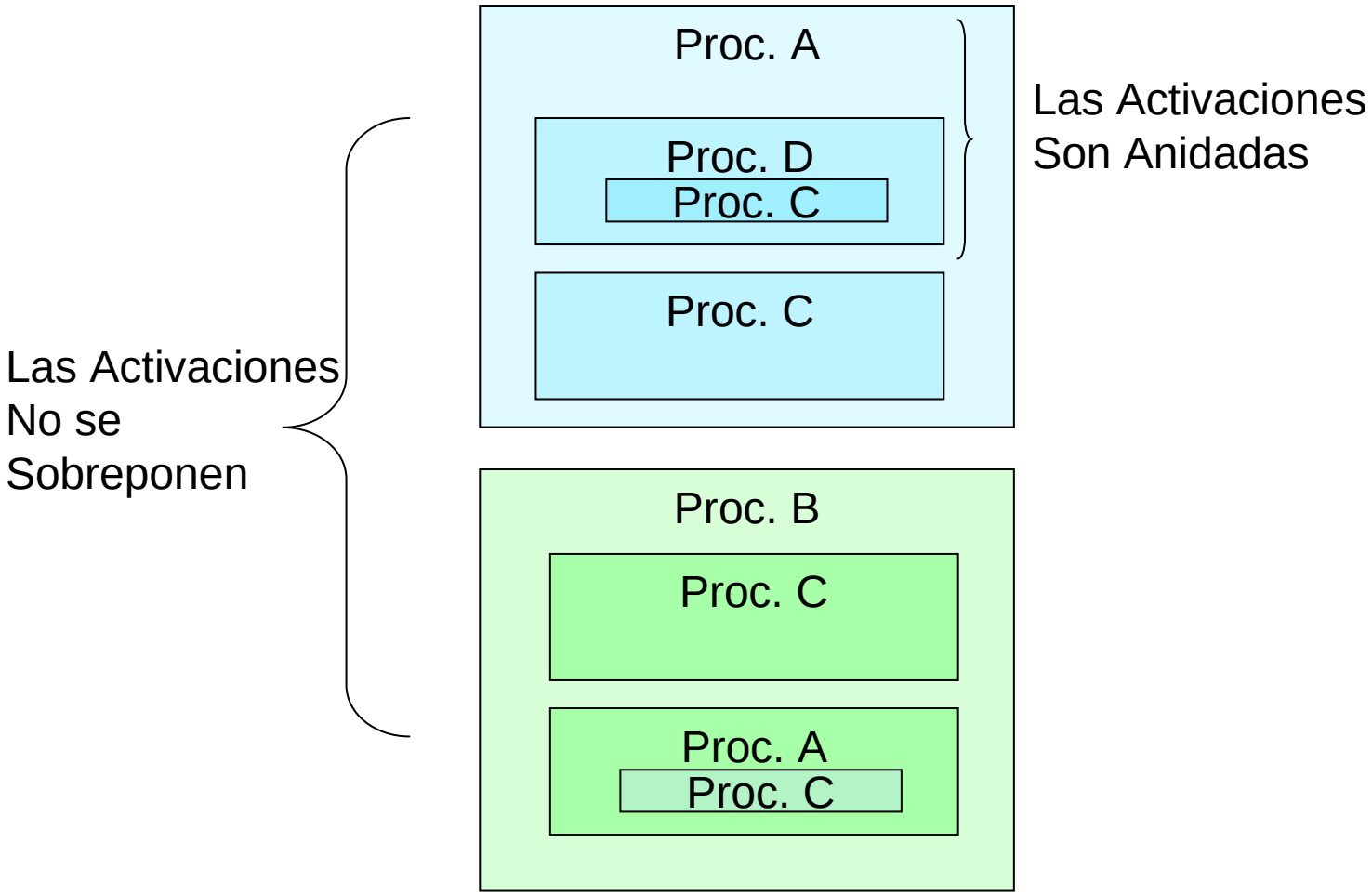
- El control fluye secuencialmente, la ejecución de un programa consta de una secuencia de pasos, en cada paso el control está en un punto específico del programa

- La ejecución de un procedimiento inicia al principio del cuerpo del procedimiento, y termina cuando se retorna el control al punto inmediatamente siguiente a aquel donde el procedimiento fue llamado.

Activación de procedimientos

- La ejecución del cuerpo de un procedimiento es una **Activación** del procedimiento
- El **tiempo de vida** de una activación de un procedimiento, es la secuencia de pasos que transcurre entre el primero y el último paso en la ejecución del cuerpo del procedimiento.
- Asumiendo lo anterior, si a y b son activaciones de procedimientos, entonces sus tiempos de vida: No se sobreponen o Son anidados.
- Un procedimiento es **Recursivo** si una activación del procedimiento puede iniciar antes que finalice una activación anterior del mismo procedimiento.

Activación de procedimientos



Arbol de Activación

- Arbol que representa la forma en que el control entra y deja las activaciones
  - Cada nodo representa la activación de un procedimiento (una unica activación)
  - La raiz representa la activación del procedimiento principal (main)
  - El nodo *a* es padre del nodo *b*, si y solo si el control fluye de una activación de *a* a *b*.
  - El nodo *a* esta a la izquierda del nodo *b*, si y solo si el tiempo de vida de *a* ocurre antes que el tiempo de vida de *b*.

Pila de Control

- El control de flujo de un programa corresponde a un recorrido en profundidad del árbol de activación,
  - Visita el nodo antes que sus hijos
  - Recursivamente visita los hijos de izquierda a derecha

- Se usa una Pila de Control para llevar el rastro de los procedimientos activos, el nodo que corresponde a una activación entra en la pila cuando la activación inicia, y sale de la misma cuando la activación termina.

Ámbito de una declaración

- Una declaración es una construcción sintáctica que asocia un nombre con información. Pueden ser explicitas o implícitas.
- El mismo nombre puede ser declarado en diferentes partes del programa. Las **Reglas de Alcance** de un programa determinan cual declaración de un nombre aplica cuando el nombre es usado en el programa.
- El **Ambito** de una declaración es la porción del programa en la cual la declaración aplica. Dentro de un procedimiento los nombres pueden ser **Locales** o **No-Locales**.

Enlace de Nombres

- **Ambiente**: una función que mapea un nombre con una posición de almacenamiento
- **Estado**: una función que mapea una posición de almacenamiento con el valor que contiene.

Ambiente(Nombre) -> Almacenamiento

Estado(Almacenamiento) -> Valor

- A esto se refiere también la distinción del significado de un identificador al lado izquierdo o derecho de una asignación:
  - R-value: el lado derecho significa el valor (Estado)
  - L-value: el lado izquierdo la posición de almacenamiento (Ambiente)

Enlace de Nombres

- Cuando el ambiente enlaza una posición de almacenamiento s con un nombre x, se dice que **x esta enlazado a s.**

- **Estático vs. dinámico:**
  - La definición de un procedimiento es estática, su activación es dinámica
  - La declaración de un nombre es estática, su enlace es dinámico
  - El ámbito de una declaración es estático, el tiempo de vida es dinámico.

Organización de la Memoria en Tiempo de Ejecución

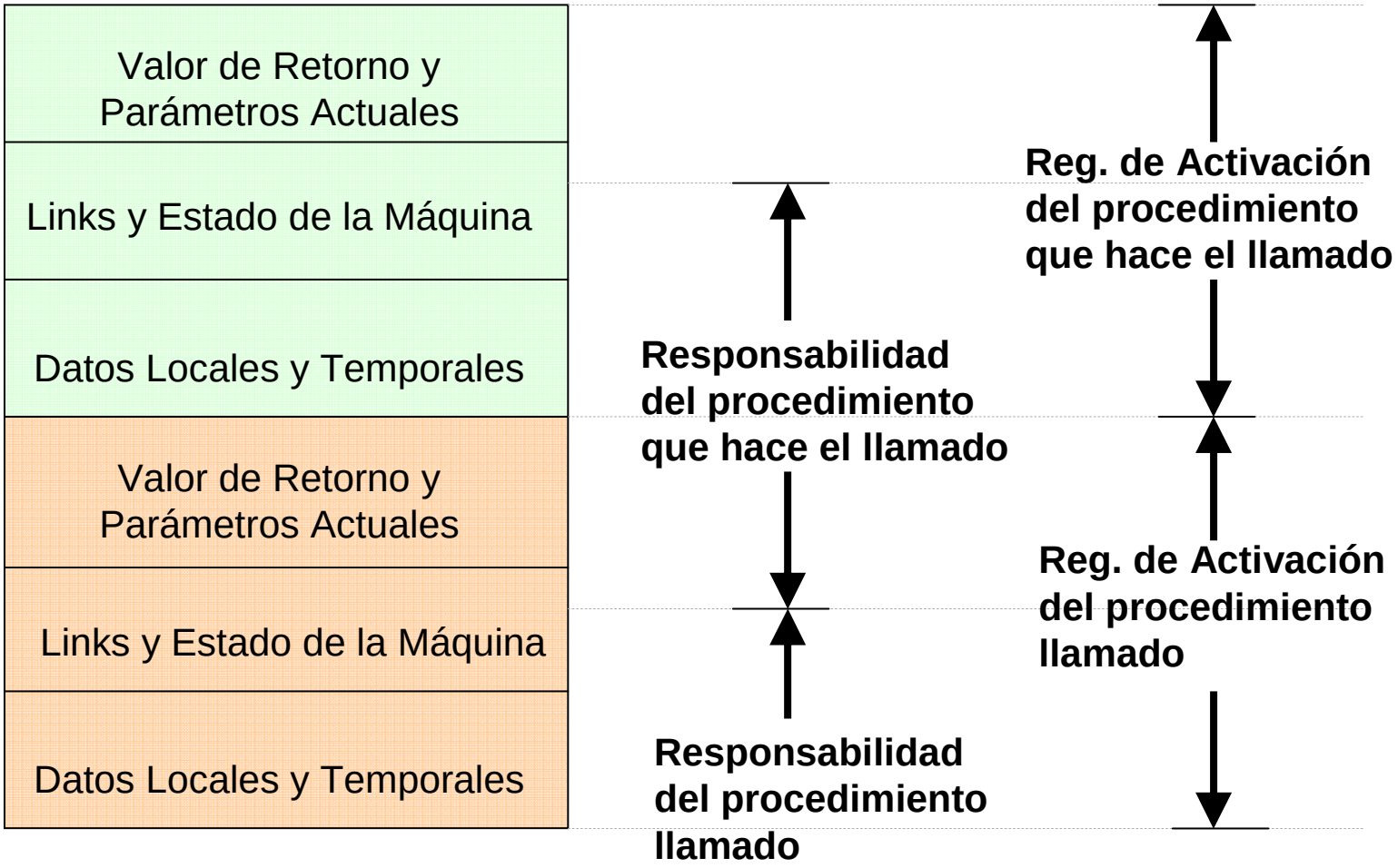
Registros de Activación

Asignación en Pila: Secuencias de Llamado y Retorno

- La secuencia de llamado a procedimientos se genera en el código destino. Debe asignar el espacio de memoria necesario para su activación e ingresar la información en sus campos.

- La secuencia de Retorno restaura el estado de la máquina de tal forma que el procedimiento llamador pueda continuar su ejecución

Asignación en Pila: Secuencia de Llamado y retorno



Asignación en Pila: Secuencia de Llamado

- El procedimiento **Llamador**:
  - Evalúa los parámetros actuales
  - Guarda la dirección de retorno y el valor del tope de la pila

- El procedimiento **Llamado**:
  - Incrementa el apuntador del tope de la pila
  - Guarda el estado de la máquina
  - Inicializa los datos locales e inicia la ejecución

Asignación en Pila: Secuencia de Retorno

- El procedimiento **Llamado**:
  - Ubica el valor de retorno
  - Restaura el tope de la pila y el estado de la máquina

- El procedimiento **Llamado**:
  - Copia el valor de retorno a su propio registro de activación

Asignación en Pila: Datos locales de longitud variable

- Dentro del registro de activación solamente se guarda el puntero al dato
- La memoria para los datos de longitud variable se asigna en tiempo de ejecución inmediatamente después del registro de activación
- Las instrucciones que calculan la dirección relativa se generan en tiempo de compilación

Estrategias para Asignación de Memoria: Asignación Estática

- Los nombres se enlazan a las posiciones de memoria durante la compilación del programa.
- No requiere de un programa run-time
- Cada procedimiento tiene asignado un registro de activación, cuya posición de memoria es asignada en la compilación
- Los valores de las variables locales permanecen entre diferentes activaciones del mismo procedimiento
- Esto implica que:
  - El tamaño de los objetos de datos debe ser conocido en tiempo de compilación.
  - No es posible declarar procedimientos recursivos.
  - No pueden usarse estructuras de datos dinámicas.
- Ejemplo: Fortran

Estrategias para Asignación de Memoria: Asignación en Pila

- Implementa la pila de control: el almacenamiento es manejado como una pila, y los registros de activación entran a la pila cuando inicia la ejecución del procedimiento, y salen de ella cuando termina.
- Los valores de las variables locales se pierden cuando la activación termina.
- Un registro marca el tope de la pila, cuando se activa el procedimiento, se asigna el espacio para su registro de activación.

Estrategias para Asignación de Memoria: Asignación en Heap

- Usado para guardar datos no estáticos, cuyo tiempo de vida no está limitado por el tiempo de activación de un procedimiento.

- Viven hasta que el programa explícitamente los libera.

- Requiere un Administrador de Memoria (Memory Management) que localiza (allocation) y libera (deallocation) espacio en el heap, y un Recolector de Basura (Garbage Collection) que recupera espacios de memoria que ya no estan siendo utilizados.

Estrategias para Asignación de Memoria: Asignación en Heap

- Problemas que pueden surgir por la liberación explicita:
  - Garbage (basura): posiciones de memoria que han sido asignadas dinámicamente y no pueden ser referenciadas
  - Referencias suspendidas: una referencia a una posición de almacenamiento que ha sido liberada. Es un error lógico. Ej. Retornar direcciones de variables locales

Asignación en Heap: Asignación Dinámica de Memoria

Técnicas para implementarla

- Con asignación de memoria explicita:
  - Bloques de Tamaño Fijo. Los bloques de memoria se referencian entre si formando listas. Hay una lista de bloques disponibles.
  - Bloques de Tamaño Variable:
    - La memoria se fragmenta
    - Una técnica posible: Buscar el primer bloque libre de tamaño mayor o igual al espacio que se va a asignar. Compactar bloques consecutivos que son liberados.

Asignación en Heap: Asignación Dinámica de Memoria

Técnicas para implementarla

- Con asignación de memoria implícita:
  - Requiere un paquete run-time.
  - Los bloques de memoria tienen un formato definido (pueden ser de tamaño fijo o variable)
  - El run-time debe reconocer si un bloque de memoria continua en uso:
    - Cuenta de referencias: registra cuantas referencias al bloque hay
    - Técnicas de marca: suspende la ejecución del programa y revisa todas las referencias a memoria

Métodos para Paso de Parámetros

- **Por valor**: el procedimiento llamador evalúa los parámetros y pasa su r-valor al procedimiento llamado
  - Si el parámetro es una expresión esta se evalúa, y si es una variable se copia su valor.
  - El procedimiento llamado no afecta datos en el registro de activación del procedimiento llamador, a menos que se pasen como parámetros apuntadores.
  - Ejemplo: en C, el llamado proc_a(&a, &b);, y dentro del procedimiento se refiere como *a, *b.

Métodos para Paso de Parámetros

- **Por referencia**: el procedimiento llamador evalúa los parámetros y pasa su l-valor al procedimiento llamado (si es una expresión le asigna memoria, y pasa la dirección de memoria asignada).

  - En la traducción del código del procedimiento el parámetro formal se convierte en una referencia indirecta
  - Los arreglos son usualmente pasados por referencia
  - Ej. En Pascal, cuando se declara

    Procedure proc_a(var a, b : integer);

Métodos para Paso de Parámetros

- **Copiar-Restaurar**:
  - El procedimiento llamador evalúa los parámetros actuales, y pasa los r-valores al procedimiento llamado.
  - Cuando el control retorna, los r-valores de los parámetros formales son copiados en los parámetros actuales (de aquellos que tienen l-valor)
  - Ej. Algunas implementaciones de Fortran

Métodos para Paso de Parámetros

- **Por nombre**:
  - El procedimiento actúa como una macro, cuyo cuerpo se copia en el procedimiento llamador, los parámetros actuales sustituyen a los formales.

Acceso a Nombres no Locales

- Esta definido por las reglas de alcance del lenguaje
- Reglas de Alcance Estático (o Léxico): determina que declaración aplica a un nombre, examinando el programa (el texto).
- Reglas de Alcance Dinámico: determina que declaración aplica a un nombre, en tiempo de ejecución, de acuerdo con los procedimientos activos

Acceso a Nombres no Locales. Bloques

- Un bloque es un conjunto de instrucciones que tiene declaraciones locales. Está delimitado por marcas que señalan su inicio y final.
- Estructura de bloques: la propiedad de dos bloques que son: independientes uno del otro o anidados.
- Regla del anidamiento mas cercano: si un nombre no fue declarado en el bloque busca la declaración mas cercana
- La estructura de bloques se puede implementar con una pila

Acceso a Nombres no Locales: Alcance Léxico

- Sin Procedimientos Anidados.
  - Todo nombre no local es global (estático)
  - Se implementa facilmente el paso de procedimientos como parámetros, o el retorno de estos como resultado.

Acceso a Nombres no Locales: Alcance Léxico

- Con Procedimientos Anidados.
  - Para un nombre no local aplica la declaración más cercana de él (siguiendo la anidación en la declaración de los procedimientos)
  - La anidación en la declaración no implica anidación en el llamado. Entonces decidir que activación corresponde a una variable no local a un procedimiento es algo que se realiza en tiempo de ejecución.
  - **Profundidad de anidamiento**: en un procedimiento asocia el nombre no local con la profundidad a la cual fue declarado
  - El **link de acceso** apunta al registro de la activación mas reciente del procedimiento que lo anida
  - Ejemplo.

Acceso a Nombres no Locales: Alcance Léxico

- Con Procedimientos Anidados.
  - Recorriendo los links de acceso puede llegar al registro de activación que tiene el nombre (sabe en que nivel de anidación lo está usando y en que nivel fue declarado)
  - Si **p** tiene profundidad $\mathbf{n_p}$ y **q** profundidad $\mathbf{n_q}$:
    - Si **p** anida en **q**, $\mathbf{n_p > n_q}$
    - $\mathbf{n_p = n_q}$, ssi **p = q**
    - Si **p** accede a una variable **x** de **q**, $\mathbf{n_p} \geq \mathbf{n_q}$, se debe seguir el link de acceso ($\mathbf{n_p - n_q}$) veces (esta cantidad se conoce en tiempo de compilación)

Acceso a Nombres no Locales: Alcance Léxico

- Con Procedimientos Anidados.
  - Para asignar el link de acceso cuando **q** llama a **p**:
    - Si $\mathbf{n_p > n_q}$, entonces $\mathbf{n_p = n_q + 1}$, el link de acceso de **p** apunta a **q**.
    - Si $\mathbf{n_p = n_q}$, **p = q**, es recursivo, **q** copia el link de **p**.
    - Si $\mathbf{n_p < n_q}$, entonces **q** anida en un procedimiento **r**, debe saltar $\mathbf{n_q - n_p}$ + 1 links de acceso para encontrar la activación del procedimiento **r**
    - Si **p** es recibido por **q** como parámetro, pasar como información del parámetro el procedimiento y el link de acceso.

Acceso a Nombres no Locales: Alcance Dinámico

- Las reglas de alcance se basan en factores que solo se conocen en tiempo de ejecución.