Tema 8:
Representación de información en memoria

1. La tabla de símbolos (TDS)
   1. *Requisitos de la TDS*
   2. *Implementación de la TDS*
   3. *La TDS en un lenguaje con estructura de bloques*
2. Gestión de memoria en tiempo de ejecución
   1. *Conceptos básicos*
   2. *Asignación estática*
   3. *Gestión del montículo (heap)*
   4. *Gestión de la pila (stack) . Registros de activación*
3. Ejemplo de asignación de memoria

V. 16

1. La tabla de símbolos (TDS)

1.1. Requisitos de la TDS

- Operaciones básicas: Inserción, búsqueda y borrado.
- Ejemplo de información a incluir:

Tdsímbolos	Una entrada por cada objeto definido por el usuario.	
	Nombre	Lexema
	Objeto	Categoría del objeto: *variable*, *parámetro*, *función*...
	Tipo	Tipo del objeto: *tinteger, tarray, trecord, tvacio* o *terror*...
	Des	Desplazamiento relativo en el segmento de datos
	Niv	Nivel de anidamiento del objeto.
	ref	Referencia a la tabla auxiliar: **TdVectores**, **TdRegistros** **TdDominios**...

TdVectores	Una entrada para cada array definido. Cada entrada contiene el límite inferior (**min**) y superior (**max**) y el tipo de los elementos del array (**tipo**).	
TdRegistros	Una entrada por cada campo de los registros.	
	Nombre	Lexema
	Tipo	Tipo del campo.
	Des	Desplazamiento relativo del campo en el segmento correspondiente.
TdDominios	Una entrada por cada dominio definido en los bloques.	

- Tratamiento de palabras reservadas:
  - Por el analizador léxico
  - Como identificadores y tabla de palabras reservadas
- Almacenamiento de los lexemas:
  - Tabla de lexemas

1.2. Implementación de la TDS

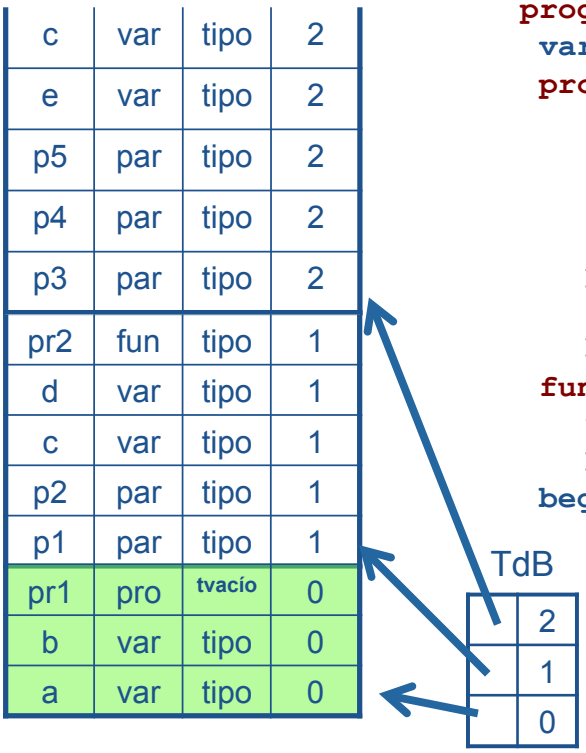
- *Array de registros*
- *Listas enlazada ordenadas. Listas ordenadas doblemente enlazadas*
- *Árboles equilibrados ordenados*
- *Tablas de dispersión (hash)*

1.3. TDS en lenguaje con estructura de bloques

```
program uno ;
  var
    a, b : Tipo ;
    procedure pr1 (p1, p2 : Tipo);
        var
        c, d: Tipo ;
        function pr2 (p3, p4, p5: Tipo): Tipo;
            var e, c: Tipo ;
            begin … end ;
        procedure pr3 (p6, p7: Tipo);
            begin pr2 … end ;
        begin pr3 … end ;
    function pr4: Tipo;
      var f : Tipo ;
      begin …  end ;
  begin pr1 … end
```

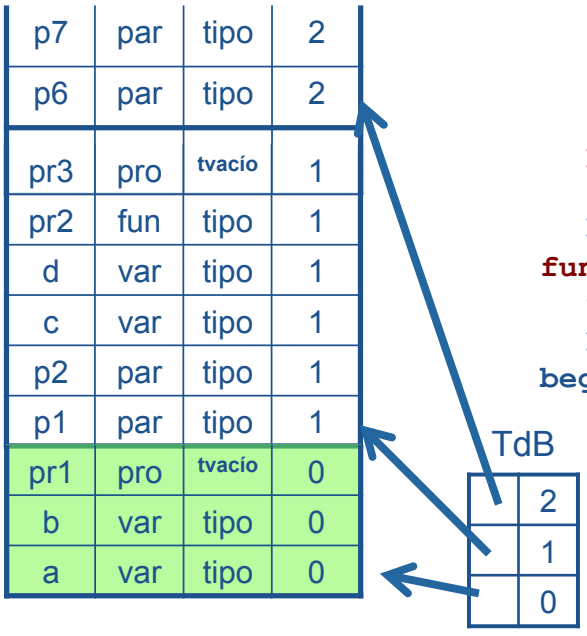

- *Problemas a resolver:*
  - *Control del alcance de cada declaración*
  - *Varios objetos con el mismo nombre accesibles*
- *Posibles soluciones:*
  - *Una subtabla para cada bloque.*
  - *Gestión como pila*

Ejemplo




```
program uno ;
 var  a, b : Tipo ;
 procedure pr1 (p1, p2 : Tipo);
    var  c, d: Tipo ;
    function pr2 (p3, p4, p5:Tipo):Tipo;
      var e, c: Tipo ;
      begin … end ;
    procedure pr3 (p6, p7: Tipo);
      begin pr2(3,4,5)  end ;
    begin pr3(6,7)  … end ;
 function pr4: Tipo;
    var f : Tipo ;
    begin …  end ;
 begin  pr1(1,2)  end
```

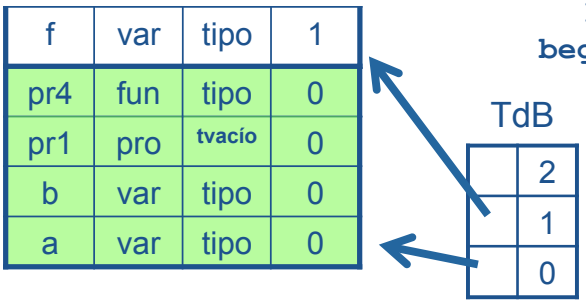

Ejemplo




```
program uno ;
 var  a, b : Tipo ;
 procedure pr1 (p1, p2 : Tipo);
    var  c, d: Tipo ;
    function pr2 (p3, p4, p5:Tipo):Tipo;
      var e, c: Tipo ;
      begin … end ;
    procedure pr3 (p6, p7: Tipo);
      begin pr2(3,4,5)  end ;
    begin pr3(6,7)  … end ;
 function pr4: Tipo;
    var f : Tipo ;
    begin …  end ;
 begin  pr1(1,2)  end
```

Ejemplo

```
program uno ;
 var  a, b : Tipo ;
 procedure pr1 (p1, p2 : Tipo);
    var  c, d: Tipo ;
    function pr2 (p3, p4, p5:Tipo):Tipo;
      var e, c: Tipo ;
      begin … end ;
    procedure pr3 (p6, p7: Tipo);
      begin pr2(3,4,5)  end ;
    begin pr3(6,7)  … end ;
 function pr4: Tipo;
    var f : Tipo ;
    begin …  end ;
 begin  pr1(1,2)  end
```



2. Gestión de memoria en tiempo de ejecución

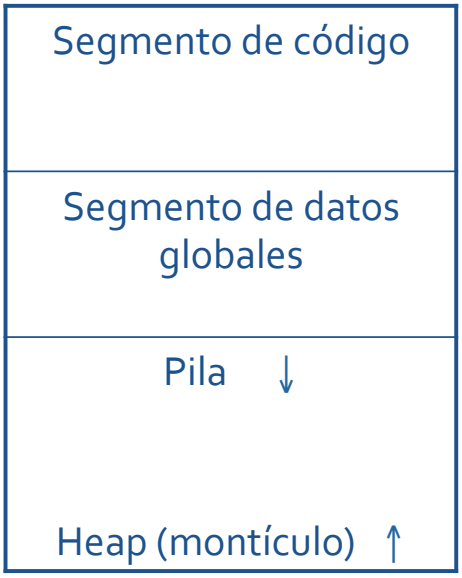
2.1. Conceptos básicos

Asignación dinámica:

- La localización del objeto solo se conocerá en tiempo de ejecución.

Asignación estática:

- La localización del objeto se conoce en tiempo de compilación.
- Requiere:
  - Conocer en tiempo de compilación el tamaño del objeto.
  - Conocer en tiempo de compilación el número de instancias simultaneas del objeto en ejecución.



2.2. Asignación estática

- A los objetos se les asigna una dirección absoluta que se mantiene durante la ejecución del programa.

Ejemplo:

- Variable global
- Variables locales estáticas (que mantienen su valor entre llamadas a la función),
- Cadenas de caracteres,...

- Si el lenguaje no dispone de recursión, se puede emplear asignación estática para las variables locales.

Ej. Fortran 90.

Ejemplo de asignación estática

P → { NIVEL = 0 ; DESP = 0 ; }
L_Decla

L_Decla → Decla | L_Decla Decla

Decla → DV

/**** Declaración de variables ****/

DV → T **id** ; { InsertarTds (id.nom, "variable", T.tipo, NIVEL, DESP);
DESP := DESP + T.talla ;}

T → **int** {T.tipo = Tentero; T.talla = TALLA_ENTERO; }
| **float** { T.tipo = Treal; T.talla = TALLA_REAL; }
| **bool** { T.tipo = Tlogico; T.talla = TALLA_LOGICO; }
| **struct** { C } { T.tipo := testructura (C.tipo); T.talla = C.talla }

C → T **id** { C.tipo := (id.nom x T.tipo); C.talla = T.talla }
| C_1 **;** T **id** { C.tipo := C1.tipo x (id.nom x T.tipo) ; C.talla = C1.talla + T.talla}

2.3. Gestión el montículo

- Montículo: Región de memoria en la que los subbloques pueden ser asignados y liberados en cualquier orden.
- Se debe usar siempre que un objeto pueda cambiar de tamaño.
- Liberación de bloques:
  - Explícita: Indicada por el programador.
  - Implícita: El bloque asignado a un objeto debe liberarse automáticamente cuando se detecte no se va a usar más el objeto. Requiere mecanismo recolector de basura (garbage collector) en tiempo de ejecución.
- Asignación de bloques: Problemas de fragmentación
  - Fragmentación interna:
    - El algoritmo de asignación asigna un bloque mayor del requerido para almacenar el objeto
  - Fragmentación externa:
    - Los bloques asignados se van dispersando: Puede haber espacio disponible pero repartido en trozos tan pequeños que puede no ser suficiente para almacenar un objeto entero.

Asignación de memoria del montículo

1. Usando lista de bloques de memoria libres. Inicialmente hay un único bloque (todo el montículo).
2. Ante petición de memoria para objeto de tamaño t:
   - Algoritmo **first-fit**: Asigna primer bloque de la lista de tamaño >= t
   - Algoritmo **best-fit**: Asigna bloque más pequeño de la lista de tamaño >=t
3. El bloque asignado se divide en dos para asignar solo un tamaño t. El resto se mete en la lista de bloques libres.
4. Al liberar un bloque se comprueba si puede fusionarse con los bloques adyacentes (si están libres).

- Mejora: Mantener varias listas de bloques libres en función del tamaño de éstos.
- Para *eliminar la fragmentación externa*: Compactar (moviendo bloques asignados)

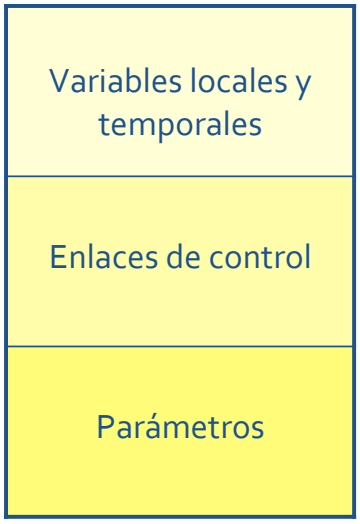
2.4. Gestión de la pila

- Los bloques de memoria se asignan y liberan en orden LIFO
- Llamamos activación de un bloque a cada una de sus ejecuciones.
- Tiempo de vida de una activación es la secuencia de pasos entre el primer y último paso de la ejecución.
  - Los tiempo de vida de dos activaciones o no se solapan, o uno incluye completamente al otro.
- Árbol de activación: Representación del tiempo de vida de las activaciones de un programa:
  - Cada nodo representa una activación.
  - A es padre de B si B se activa durante el tiempo de vida de A
  - A está a la izquierda de B si el tiempo de vida de A es anterior a B (A finaliza antes de comenzar B)
- El flujo de control del programa coincide con un recorrido en profundidad del árbol de activación

Registro de activación

- Espacio de la pila de ejecución asignado a una activación para almacenar sus datos locales.
- También llamados marcos (frames) de pila.
- Display: Bloque de punteros con una entrada por cada nivel de anidamiento, donde Display[n] apunta al último RA con objetos de nivel de anidamiento n cargado.

Estructura general



Ejemplo de RA y acceso

Acceso a Vble. local (niv , desp) :

vble.pos := display[niv] + desp

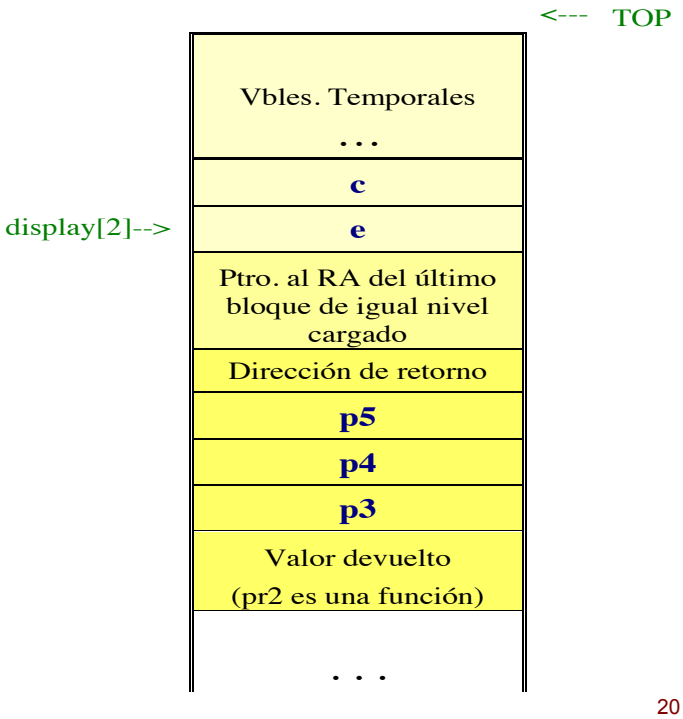
Acceso a Parámetro (niv , desp) :

param.pos :=

display[niv] + desp
- (Talla _de_parametros +
Talla_Enlaces_y_dir_retorno)

Depositar valor devuelto por una función de nombre (niv, -):

display[niv+1] - (Talla _de_parametros +
Talla_Enlaces_y_dir_retorno +
Talla_valor_devuelto)



Ejemplo de RA y acceso

Suponiendo elementos de talla 1

Acceso a Vble. local (niv , desp) :

vble.pos := display[niv] + desp

Acceso a Parámetro (niv , desp) :

param.pos :=

Display[niv] + desp

- (Num_parametros +

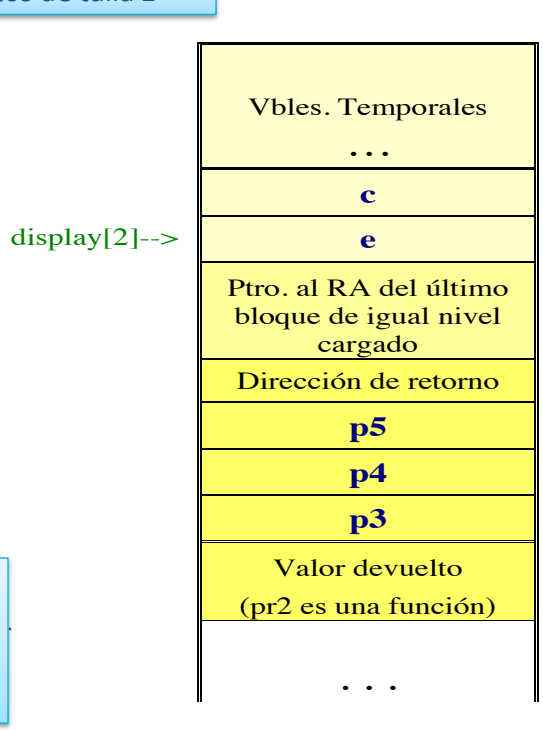
2)

Depositar valor devuelto por una función

Display[niv+1] - (Num_parametros +

2 +

1)



Ejemplo de Frame y acceso en C

- A los registros de activación se les suele denominar Frames

- No hay anidamiento de funciones:

  Display solo necesita 1 nivel -> FramePointer

  Nivel 0 = Global

  Nivel 1 = Local (apuntado por FramePointer)

Ejemplo de Frame y acceso en C

Acceso a Vble. local (desp) :

vble.pos := frame_pointer + desp

Acceso a Parámetro (desp) :

(apilados en orden inverso a declaración)

param.pos := frame_pointer -

(desp + Talla_Enlaces_y_dir_retorno + Talla_del_param)

Depositar valor devuelto por una función de nombre (niv, -):

frame_pointer – (Talla_Área_Parámetros + Talla_Área_Enlaces + Talla_valor_devuelto)

Ejemplo de Frame y acceso en C

Suponiendo elementos de talla 1

Acceso a Vble. local (desp) :

vble.pos := frame_pointer + desp

Acceso a Parámetro (desp) :

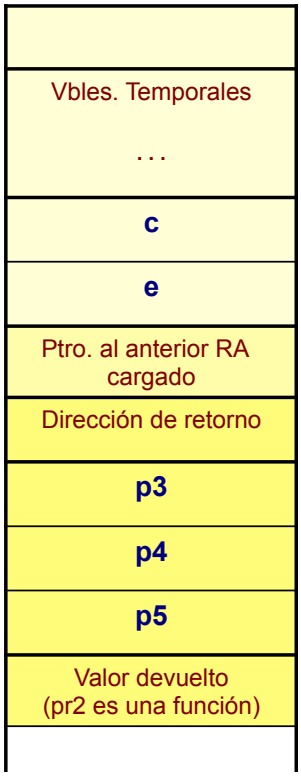
(apilados en orden inverso a declaración)

frame_pointer -
(desp + 2 +1)

Depositar valor devuelto por una función de nombre (niv, -):

frame_pointer – (Talla_Área_Parámetros + Talla_Área_Enlaces

frame_pointer – (Num_parámetros +
(2 + 1)



Secuencia de carga RA

Bloque llamador

1. Evaluar parámetros actuales
2. Si es una llama a función: Crear variable temporal para el valor devuelto
3. Apilar parámetros actuales
4. Apilar dirección de retorno (en general, el estado de la máquina) y saltar al código del bloque llamado (CALL)

Bloque llamado

1. Apilar enlace de control (Apilar Display[n])
2. Actualizar display (display[n] = TOP)
3. Reservar espacio para área de datos locales

Secuencia de descarga RA

Bloque llamado

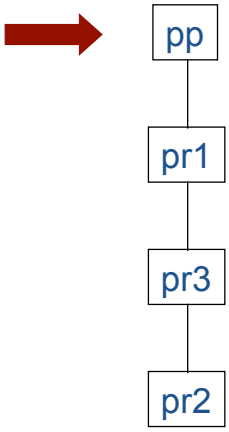
1. Desapilar área de datos (TOP= Display[n])
2. Desapilar enlace de control y restaurar valor de display (display[n]= POP)
3. Desapilar dirección de retorno y saltar a ella (RETURN)

Bloque llamador

1. Desapilar parámetros actuales
2. Si se llamó a función: Desapilar valor de retorno

Ejemplo (I)

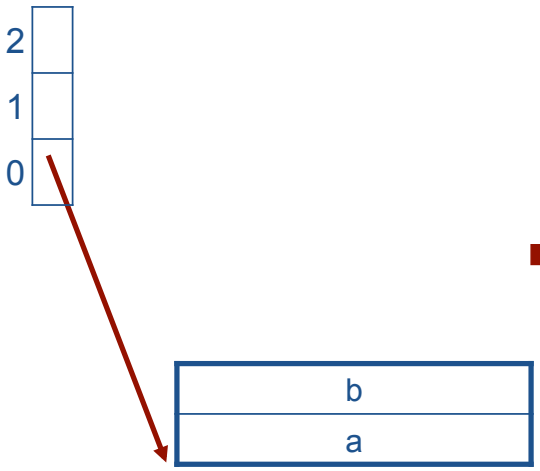
El árbol de activación es:




```
program ra ;
 var  a, b : Tipo ;
 procedure pr1 (p1, p2 : Tipo);
    var  c, d: Tipo ;
    function pr2 (p3, p4,p5:Tipo):T;
      var e, c: Tipo ;
      begin … end ;
    procedure pr3 (p6, p7: Tipo);
      begin pr2(3,4,5)  end ;
    begin pr3(6,7)  … end ;
 function pr4: Tipo;
    var f : Tipo ;
    begin …  end ;
begin  pr1(1,2)  end
```

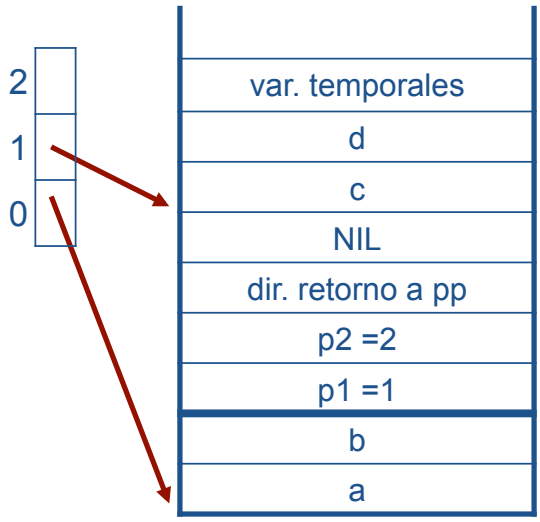

Ejemplo (II)

```
program ra ;
 var  a, b : Tipo ;
 procedure pr1 (p1, p2 : Tipo);
    var  c, d: Tipo ;
    function pr2 (p3, p4,p5:Tipo):T;
      var e, c: Tipo ;
      begin … end ;
    procedure pr3 (p6, p7: Tipo);
      begin pr2(3,4,5)  end ;
    begin pr3(6,7)  … end ;
 function pr4: Tipo;
    var f : Tipo ;
    begin …  end ;
begin  pr1(1,2)  end
```



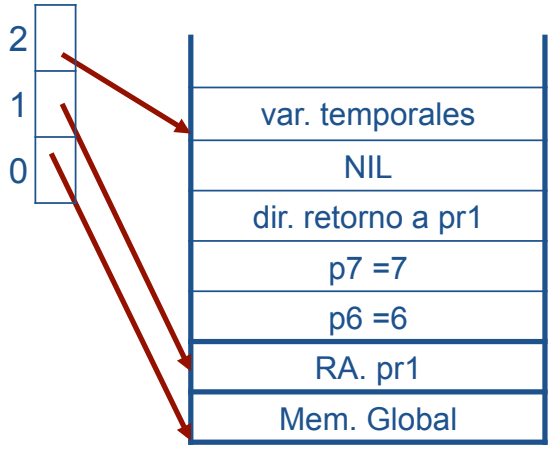
Memoria global

Ejemplo (III)



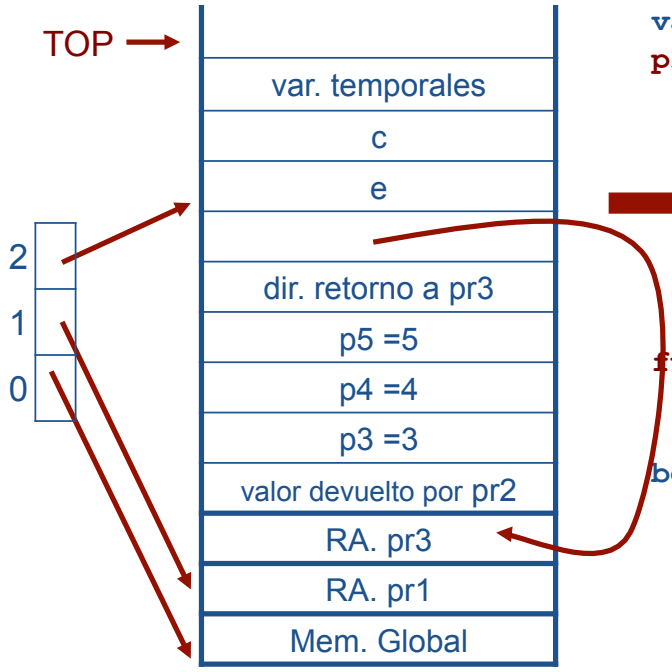

```
program ra ;
 var  a, b : Tipo ;
 procedure pr1 (p1, p2 : Tipo);
    var  c, d: Tipo ;
    function pr2 (p3, p4,p5:Tipo):T;
      var e, c: Tipo ;
      begin … end ;
    procedure pr3 (p6, p7: Tipo);
      begin pr2(3,4,5)  end ;
    begin pr3(6,7)  … end ;
 function pr4: Tipo;
    var f : Tipo ;
    begin …  end ;
 begin  pr1(1,2)  end
```


Ejemplo (IV)



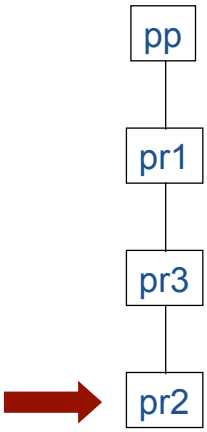

```
program ra ;
 var  a, b : Tipo ;
 procedure pr1 (p1, p2 : Tipo);
    var  c, d: Tipo ;
    function pr2 (p3, p4,p5:Tipo):T;
      var e, c: Tipo ;
      begin … end ;
    procedure pr3 (p6, p7: Tipo);
      begin pr2(3,4,5)  end ;
    begin pr3(6,7)  … end ;
 function pr4: Tipo;
    var f : Tipo ;
    begin …  end ;
 begin  pr1(1,2)  end
```

Ejemplo (V)



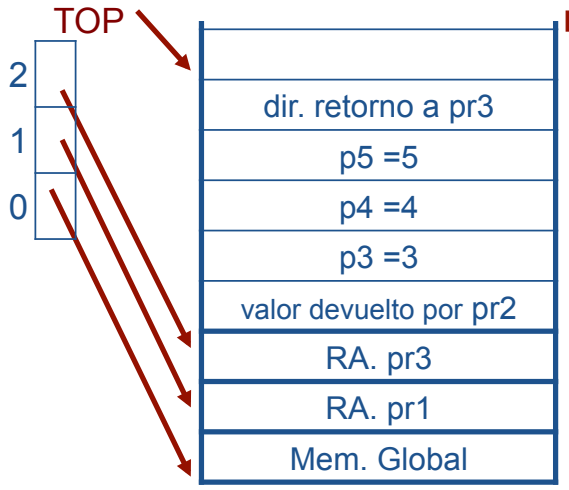

```
program ra ;
 var  a, b : Tipo ;
 procedure pr1 (p1, p2 : Tipo);
    var  c, d: Tipo ;
    function pr2 (p3, p4,p5:Tipo):T;
      var e, c: Tipo ;
      begin … end ;
    procedure pr3 (p6, p7: Tipo);
      begin pr2(3,4,5)  end ;
    begin pr3(6,7)  … end ;
 function pr4: Tipo;
    var f : Tipo ;
    begin …  end ;
begin  pr1(1,2)  end
```


Ejemplo (VI)



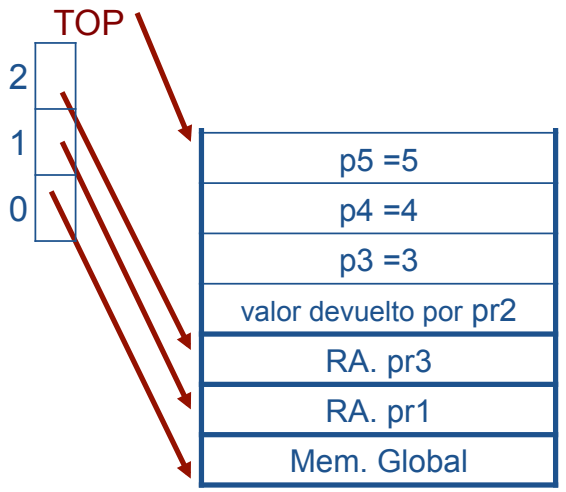

```
program ra ;
 var  a, b : Tipo ;
 procedure pr1 (p1, p2 : Tipo);
    var  c, d: Tipo ;
    function pr2 (p3, p4,p5:Tipo):T;
      var e, c: Tipo ;
      begin … end ;
    procedure pr3 (p6, p7: Tipo);
      begin pr2(3,4,5)  end ;
    begin pr3(6,7)  … end ;
 function pr4: Tipo;
    var f : Tipo ;
    begin …  end ;
begin  pr1(1,2)  end
```

Ejemplo (VII)



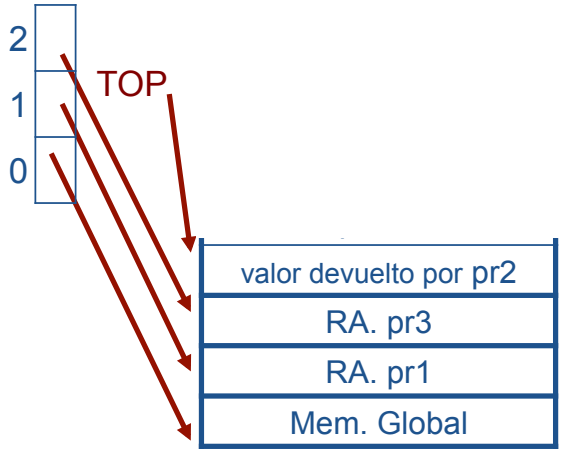

```
program ra ;
 var  a, b : Tipo ;
 procedure pr1 (p1, p2 : Tipo);
    var  c, d: Tipo ;
    function pr2 (p3, p4,p5:Tipo):T;
      var e, c: Tipo ;
      begin … end ;
    procedure pr3 (p6, p7: Tipo);
      begin pr2(3,4,5)  end ;
    begin pr3(6,7)  … end ;
 function pr4: Tipo;
    var f : Tipo ;
    begin …  end ;
 begin  pr1(1,2)  end
```


Ejemplo (VIII)



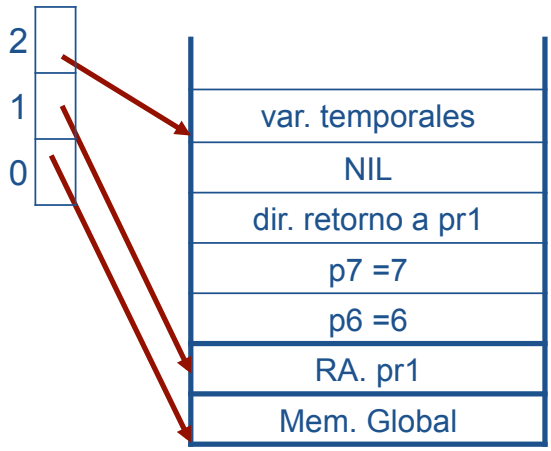

```
program ra ;
 var  a, b : Tipo ;
 procedure pr1 (p1, p2 : Tipo);
    var  c, d: Tipo ;
    function pr2 (p3, p4,p5:Tipo):T;
      var e, c: Tipo ;
      begin … end ;
    procedure pr3 (p6, p7: Tipo);
      begin pr2(3,4,5)  end ;
    begin pr3(6,7)  … end ;
 function pr4: Tipo;
    var f : Tipo ;
    begin …  end ;
 begin  pr1(1,2)  end
```

Ejemplo (IX)




```
program ra ;
 var  a, b : Tipo ;
 procedure pr1 (p1, p2 : Tipo);
    var  c, d: Tipo ;
   function pr2 (p3, p4,p5:Tipo):T;
     var e, c: Tipo ;
     begin … end ;
   procedure pr3 (p6, p7: Tipo);
     begin pr2(3,4,5)  end ;
   begin pr3(6,7)  … end ;
 function pr4: Tipo;
   var f : Tipo ;
   begin …  end ;
 begin  pr1(1,2)  end
```


Ejemplo (X)




```
program ra ;
 var  a, b : Tipo ;
 procedure pr1 (p1, p2 : Tipo);
    var  c, d: Tipo ;
   function pr2 (p3, p4,p5:Tipo):T;
     var e, c: Tipo ;
     begin … end ;
   procedure pr3 (p6, p7: Tipo);
     begin pr2(3,4,5)  end ;
   begin pr3(6,7)  … end ;
 function pr4: Tipo;
   var f : Tipo ;
   begin …  end ;
 begin  pr1(1,2)  end
```

Ejemplo de asignación de memoria

P → L_Decla

L_Decla → Decla | L_Decla Decla

Decla → DV

/* Declaración de variables ****/***

DV → T **id** ;

T → **int**

| **float**

| **bool**

| **struct** { C }

/** Miembros de estructuras ****/**

C → T **id**

| C_1 ; T **id**

3. Asignación de memoria

Ejemplo de asignación de memoria

P → L_Decla

L_Decla → Decla | L_Decla Decla

Decla → DV | DF

/* Declaración de variables ****/***

DV → T **id** ; | T **id** LI ;

T → **int**

| **float**

| **bool**

| **struct** { C }

/** Miembros de estructuras ****/**

C → T **id**

| C_1 ; T **id**

/* Índices de declaración de array ****/***

L_I → **[cte]** | L_I_1 **[cte]**

/* Declaración de función ****/***

DF → Cab_F Bloque

Cab_F → T **id** (P_F)

Bloque → **{** DVL L_Inst **}**

DVL → DVL DV | ε

/* Parámetros formales ****/***

P_F → L_PF | ε

L_PF → T **id** | T **id** , L_PF_1

Ejemplo de asignación de memoria

P → *{ NIVEL = 0 ; DESP = 0 ; }*

L_Decla

L_Decla → Decla | L_Decla Decla

Decla → DV | DF

/* Declaración de variables ****/***

DV → T **id** ; *{ InsertarTds (id.nom, "variable", T.tipo, NIVEL, DESP);*

DESP := DESP + T.talla ;}

DV → T **id** L_I ; *{ InsertarTds (id.nom, "variable", vector(T.tipo), NIVEL, DESP);*

*DESP := DESP + (L_I.talla * T.talla) ; }*

T → **int** *{T.tipo = Tentero; T.talla = TALLA_ENTERO; }*

| **float** *{ T.tipo = Treal; T.talla = TALLA_REAL; }*

| **bool** *{ T.tipo = Tlogico; T.talla = TALLA_LOGICO; }*

| **struct** { C } *{ T.tipo := testructura (C.tipo); T.talla = C.talla }*

Ejemplo de asignación de memoria

*/**** Índices de declaración de array ****/*

L_I → **[cte]** *{ LI.talla := cte.lexval;}*

| L_I_1 **[cte]** *{ L_I.talla := cte.lexval * L_I_1.talla ; }*

C → T **id** *{ C.tipo := (id.nom x T.tipo); C.talla = T.talla }*

| C_1 **;** T **id** *{ C.tipo := C1.tipo x (id.nom x T.tipo) ; C.talla = C1.talla + T.talla}*

Ejemplo de asignación de memoria

*/**** Declaración de función ****/*

DF → Cab_F Bloque *{ Descarga_nivel(NIVEL); NIVEL-- ; DESP:=OLD_DESP; }*

Cab_F → T **id** *{ NIVEL++ ; Carga_nivel(NIVEL) ;*
OLD_DESP := DESP; DESP := 0 ; DPAR := TALLA_EC+Dir_ret ; }

(P_F **)** *{ InsertarTds (id.nom,"funcion", P_F.tipo -> T.tipo, NIVEL-1, 0) ; }*

Bloque → **{** DVL L_Inst **}** DVL → DVL DV | ε

*/**** Parámetros formales ****/*

P_F → L_PF *{ P_F.tipo := L_PF.tipo }*

| ε *{ P_F.tipo := tvacio }*

L_PF → T **id** *{ DPAR := DPAR + T.talla ; L_PF := T.tipo ;*
InsertarTds (id.nom,"parametro",T.tipo, NIVEL, - DPAR) ; }

| T **id** **,** *{ DPAR := DPAR + T.talla ;*
InsertarTds (id.nom,"parametro",T.tipo, NIVEL, - DPAR) }

L_PF_1 *{ L_PF := L_PF_1 x T.tipo }*

Tema 8:
Representación de información en memoria

1. La tabla de símbolos (TDS)
   1. *Requisitos de la TDS*
   2. *Implementación de la TDS*
   3. *La TDS en un lenguaje con estructura de bloques*
2. Gestión de memoria en tiempo de ejecución
   1. *Conceptos básicos*
   2. *Asignación estática*
   3. *Gestión del montículo (heap)*
   4. *Gestión de la pila (stack) . Registros de activación*
3. Ejemplo de asignación de memoria