Trabajo Práctico N° 02

Alcance, Registros de Activación

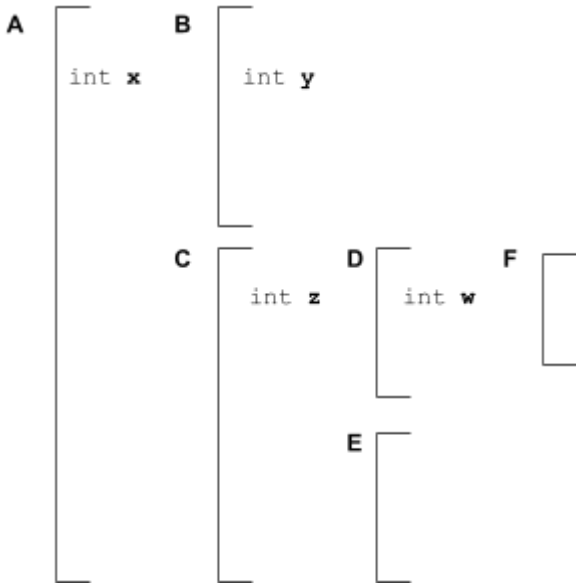
Recomendaciones Generales:

- **Alcance**
  - De una *Variable*: Qué funciones pueden usar la variable. Si se puede usar una variable o no.
  - De *nombre de una función*: El alcance de un procedimiento comprende el conjunto de instrucciones que lo pueden usar siguiendo la CE
- **Binding**: *Momento* preciso en que una *propiedad* de un *elemento* del lenguaje es *conocida*.
  - Elemento:
    - Variables ➔ Valor, Tipo, Almacenamiento, Alcance, Nombre, Tiempo de Vida (consecuencia de Almacenamiento y Alcance).
  - Momento: Tiempo en el que se conoce
    - Estático
      - En tiempo de Definición del Lenguaje
      - En Tiempo de Escritura del Programa
    - Dinámico
      - En Tiempo de Ejecución
- **Lenguajes Estáticos**: Almacenamiento Estático
- **Lenguajes Tipo Algol**:
  - Almacenamiento Dinámico
  - Tipos Estáticos
  - Alcance Estático
- **Lenguajes Dinámicos**
  - Almacenamiento Dinámico
  - Tipos Dinámicos o Alcance Dinámico (Posibles combinaciones)
- **Closures**: Unidad que se ejecuta en un ámbito distinto del cual fue definida
  - Binding de Ámbito
    - Estático: Python
    - Dinámico: Perl
- **Registros de Activación:** Contiene la información NUEVA necesaria para la ejecución de la Unidad Actual
  - Variables Locales, Cadena Estática, Cadena Dinámica, Entorno Global
- **Fórmula de cálculo de direcciones para Almacenamiento de Arreglos**

var z[t .. n, m .. u] of **integer;**		
Filas	**z[i,j]**	(i-t) (u-m+1) + (j-m) * tam elem
Columnas	**z[i,j]**	(j-m) (n-t+1) + (i-t) * tam elem

Ejercicio 1

Sea un programa cuyo anidamiento de bloques es el siguiente



a) Construir la matriz de llamados correspondiente.
b) ¿Cómo sería la pila de registros de activación para los siguientes llamados?

b.1)	A	→	C	→	D	→	E	→	B		
b.2)	A	→	C	→	C	→	B	→	D		
b.3)	A	→	B	→	C	→	E	→	A	→	D

c) Analizando si es posible realizar cada una de las siguientes asignaciones y en cuáles unidades se podrían realizar ¿qué registro de activación podrían estar en el tope de la pila en el momento de ejecutarse cada asignación?
   c.1) z := x + y
   c.2) w := x + z
   c.3) w := x + y
   c.4) x := z
   c.5) x := y

d) Explique detalladamente cómo se ejecutaría c.2) desde F. Es decir, cómo realiza la búsqueda de cada variable y qué parte de la asignación se realiza en compilación y qué parte en ejecución.

Plan 2011:

e) ¿Es correcto afirmar que *"El alcance de E es A, B, C, D, E, F"* o que *"El alcance de E es C, D, E, F"*?
f) Explicar cómo se construye, paso a paso, la cadena estática para los llamados válidos del inciso b).

Ejercicio 2

Sean A, B, C, D, E, y F los nombres de seis unidades de un programa que están anidadas unas en otras. Las llamadas permitidas entre estas unidades están indicadas en la siguiente tabla.

	A	**B**	**C**	**D**	**E**	**F**
A	R	L	X	X	X	X
B	G_2	R	L	X	X	X
C	G_3	G_2	R	L	L	L
D	G_4	G_3	G_2	R	G_1	G_1
E	G_4	G_3	G_2	G_1	R	G_1
F	G_4	G_3	G_2	G_1	G_1	R

a) ¿Puede haber más de un árbol de anidamiento para esta matriz de llamados?
Si su respuesta es afirmativa, dé 2 ejemplos.
Si su respuesta es negativa indique el motivo.

b) Si en C se define una variable int x, analice si es correcta la siguiente afirmación *"El alcance de x es B, D, E y F"*

Ejercicio 3

Dados los siguientes dos fragmentos de código Assembler referidos a la existencia de 3 variables:

1
```
MOV R1, $87B0
ADD R1, $432A
MOV  $9D86, R1
```

2
```
MOV R2, RP
MOV RP, RP[8]
MOV R, RP[40]
MOV RP, R2
MOV RP, RP[8]
MOV RP, RP[8]
ADD R, RP[20]
MOV RP, R2
MOV RP[20], R
```

a) Indicar qué realizan las instrucciones Assembler con las variables en cada caso .
b) Si bien, desde el punto de vista semántico se realiza lo mismo en ambos fragmentos con las variables, ¿Cuál es la diferencia esencial entre ambos? ¿Dónde se refleja esa diferencia en ambos códigos?

Plan 2011:

c) ¿Se puede deducir observando estos códigos el ámbito de cada variable?
Si su respuesta es afirmativa especifique el ámbito para cada variable.
Si es negativa indique por qué no lo deduce.

Ejercicio 4

Considere el siguiente programa, en un lenguaje tipo Algol sin shadowing con pasaje de parámetros por copia valor, que tiene dos errores de compilación. Se desea que el programa imprima 2 y 3.

```
void f(int k) {
   int k;
   static int m=k;
   for (int k=m;k<4;k++) {
            int k;
            print(k);
     {
      static const int k=4;
      int n;
      n:=m+k;
     }
   }
}
int main() {f(2);}
```

a) Identificar los dos errores indicando las instrucciones donde ocurren y explicando por qué se producen en compilación. Sin crear nuevas variables ni nuevas instrucciones, elimine los dos errores.

b) Una vez eliminados los errores, realizar la pila de ejecución del programa

c) ¿Cuántos registros de activación existen como máximo al mismo tiempo en ejecución? ¿Cuáles? Justifique.

d) Indique si las siguientes afirmaciones son verdaderas o falsas. Justifique brevemente sus respuestas

1. Para realizar la asignación **n:=m+k** se debe buscar a m siguiendo la cadena estática
2. Durante compilación se extrae el valor de m para realizar **m+k**
3. Si el llamado **f(2)** se realizase con **f(3.25)** se produce un error en tiempo de ejecución

Plan 2011: d)

4. Hay tres variables pertenecientes a una unidad que fueron declaradas fuera de los delimitadores sintácticos de las unidades correspondientes
5. Si se reemplaza **print(k)** por **print(n)** se produce un error de ejecución
6. La variable n es la de menor tiempo de vida y la constante **k** está al alcance de todas las unidades del programa
7. En la última iteración del for, durante la ejecución del programa, existen cuatro direcciones de memoria que corresponden a identificadores de un mismo nombre, al mismo tiempo en la pila de ejecución

Ejercicio 5

Dado el siguiente programa en un lenguaje tipo Algol con anidamiento de unidades y en el cual, las constantes son almacenadas como constantes literales .

```
static int x=1;
static int *y;
void main(){
   void f(int *p) { *p = x; }
   const  int z=2;
   int w[1..z];    [1]
   int *v;
   v=(int*) malloc(sizeof(int*));
   y=(int*) malloc(sizeof(int*));
   *v=z;
   f(y);
   w[1]=x;         [2]
}
```

a) Realizar un diagrama de la pila de ejecución
b) Indicar cuáles variables se crean durante la ejecución y su clasificación de acuerdo a su almacenamiento
c) Ordenar las variables, de menor a mayor, por su alcance y por su tiempo de vida justificando brevemente cada caso e indicando si algunas están al mismo nivel en el orden
d) Indicar en qué se modifica el programa, respecto de la clasificación de b), si [1] se reemplaza por int **w y [2] por w = &&x.
¿Qué efectos producen las modificaciones propuestas? Justifique sus respuestas
e) Indique si las siguientes afirmaciones son Verdaderas o Falsas. Justifique brevemente sus respuestas
   a. El tamaño de la pila de registros de activación puede variar en diferentes ejecuciones del programa
   b. La constante z está almacenada en el registro de activación de MAIN y su ámbito es igual que el de la variable w
   c. El binding de almacenamiento de todas las variables del programa se establece en compilación
   d. El binding de valor de todas las variables del programa se establece en ejecución

Ejercicio 6

Considere el siguiente fragmento de programa

```
main()
begin
     int m;
     function zz( )
     begin
       int costo[m..20,10..20];
       m:=17;
       costo[15,15]:=0;
     end
     m:=5;
     zz( );
end
```

Suponga que el lenguaje verifica límites en el tiempo de ejecución. Considere especialmente la instrucción destacada **costo**[15,15]:=0.

a) ¿Cuáles de las siguientes conductas describen lo que ocurre al ejecutarse esta instrucción? Justifique.
   a.1) Se produce un error siempre que se arriba a esta instrucción

a.2) Se produce un error en algunos casos.

a.3) Se produce un error en la primera ejecución pero no en las siguientes.

a.4) No se produce nunca un error

b) ¿Qué ocurriría si **zz()** se llamase nuevamente (o sea, hay dos llamados consecutivos de **zz**)?

Plan 2011:

c) ¿Qué ocurriría si en **zz()** hubiese un loop?

Ejercicio 7

Se tiene el siguiente fragmento de código en un lenguaje tipo **Algol**, simil C, las constantes simbólicas no son almacenadas como variables.

Responda cada uno de los incisos. Justifique todas sus respuestas.

```
static int c=1;
void f() {
  int b[c];
  if (c<2){
     c++;
     f();
     {
      int c=3;
      while (++c < 5){
         const int c=5;
         int b[c];
         cout << c;
      }
      cout << c;
     }
 }
  cout << c;
}
int main() {
     int c=6;
     f();
     return 0;
}
```

a) Construya la pila de ejecución e indique los resultados de cada salida justificando los mismos.

b) ¿Cuántas variables **c** hay al mismo tiempo en memoria como máximo en el programa?
¿En qué momento de la ejecución se produce? Indicar además la sentencia en la que sucede.

c) ¿Qué cambios se producen en las salidas si se declaran todas las variables **c** como **static**? ¿Cómo cambiaría el alcance de las mismas? ¿A cuáles les cambiaría el tiempo de vida?

d) ¿Cuántas variables **b** hay, al mismo tiempo, en memoria?
¿Qué diferencias sintácticas y semánticas hay entre éstas?

e) ¿Qué diferencia, en cuanto a su alcance, se produce si la primera unidad anónima definida en la función **f** tuviese nombre?

Ejercicio 8

Dado el siguiente programa en un lenguaje tipo Algol, en el que las constantes no son almacenadas, se admite **shadowing o reutilización de nombre de variables**, y **sólo** se pueden realizar operaciones entre **operandos de exactamente el mismo tipo,** por lo que el compilador buscará la variable que coincida en tipo entre las disponibles:

```
int multiplica(int n1,int n2){
      int*suma=malloc(sizeof(int));
      *suma=0;
      int arr[n2];
      for (int i=0;i<n2;i++){
            arr[i]=n1;
      }
      for (int i=0;i<n2;i++){
            *suma+=arr[i];
      }
      return *suma;
}

int triple1(int n){
      static float numero(1)=6.0; //1
      return n*numero;            //2
}

int triple2(int n){
      return n*numero;            //3
}

int numero(4)=3;                  //4
main(){
      int numero(5)=8;            //5
      const int uno=1;
      print(multiplica(3,2));
      print(triple1(uno));
      numero=4;
      print(triple1(uno));
      print(triple2(uno));
}
```

a) Complete la siguiente tabla:
Aclaraciones: *para este inciso, los siguientes y en el código, el número entre paréntesis luego de la variable* ***numero*** *indica la línea en la que fue declarada.* ***Instancias en memoria*** *son las instancias de la variable que podrían existir en la ejecución del programa.*

Variable	Tipo de almacenamiento	Lugar donde se almacena	Instancias en memoria
suma			
malloc(sizeof(int))			
arr[n2]			
numero(1)			
numero(4)			
numero(5)			
uno			

b) ¿Qué valores imprime el programa, por qué?

c) Para cada etapa de la ejecución del programa, indique con SI/NO en cada celda, el alcance y tiempo de vida de las siguientes variables:

	inicio de ejecución main->	ejecución de multiplica->	ejecución de triple1->	ejecución de triple1 ->	ejecución de triple2 ->	fin de ejecución de main
numero(1) está al alcance						
numero(1) existe en memoria						
suma está al alcance						
suma existe en memoria						
malloc está al alcance						
malloc existe en memoria						

d) Indicar si las siguientes afirmaciones son verdaderas o falsas justificando cada una de las respuestas:

i. En **//3** se usa la variable numero(1) porque está al alcance.
ii. En **//2** se usa la variable numero(5) porque está al alcance y su tipo es compatible en la operación.
iii. Durante la ejecución del programa, el nombre de las variables se almacena porque se necesita para buscar las mismas.
iv. Si el lenguaje tuviese alcance dinámico, la variable usada en **//3** sería la variable numero(5) porque es la última declarada con ese nombre.
v. Durante la ejecución del programa, la variable arr va a ocupar en memoria menos que el tamaño de sus datos
vi. El tamaño de la variable arr es conocido en ejecución puede cambiar de tamaño, por lo que se debe almacenar en el heap

Ejercicio 9

En un lenguaje que verifica límites en tiempo de ejecución, se tiene el siguiente arreglo en el que sólo dos de los cuatros límites son variables:

```
var z[10 .. n, m .. 15] of integer;
```

Analice el almacenamiento de los límites.
Indique cuáles de las siguientes afirmaciones son VERDADERAS y cuáles FALSAS, justificando todas sus respuestas.

a) Resulta más eficiente ubicar todos los límites en el descriptor del registro de activación que contiene la variable **z.**
b) Resulta menos costoso ubicar todos los límites en el código ejecutable
c) Resulta más eficiente ubicar las constantes en el código ejecutable y los valores de **n** y **m** en el registro de activación que contiene la variable **z**
d) **n** debe almacenarse en el registro de activación si NO se verifican límites pero el arreglo está almacenado por filas.

Ejercicio 10

Dados los diferentes programas en un lenguaje con alcance estático:

a) Asociar cada programa con alguna/s de las siguientes salidas. Justifique su respuesta.

S1	**S2**	**S3**	**S4**
1	2	3	Error de ejecución

P1	P2	P3	P4

fun main() {
var x = 1
fun a() {
  var x = 2
  b(c)
}
fun b(f) {
  var x = 3
  f()
  }
fun c() {
  print x
  }
a()
}
	fun main() {
	var x = 1
	fun a() {
	  var x = 2
	  fun c() {
	  print x
	  }
	  b(c)
	}
	fun b(f) {
	  var x = 3
	  f()
	  }
	a()
	}
		fun main() {
		var x = 1
		fun a() {
		  var x = 2
		  fun b(f) {
		  var x = 3
		  f()
		  }
		  b(c)
		}
		fun c() {
		  print x
		  }
		a()
		}
			fun main() {
			var x = 1
			fun a() {
			  var x = 2
			  fun b(f) {
			  var x = 3
			  f()
			  }
			fun c() {
			  print x
			  }
			b(c)
			}
			a()
			}

b) Asociar cada programa con un árbol de anidamiento correspondiente

Opción	Árbol (“ x[y]” significa “x anida a y”)	Programa correspondiente
A1	main [a b c]	
A2	main [a [c] b]	
A3	main [a [b] [c]]	
A4	main [a [b c]]	

c) ¿Cuáles de las siguientes representaciones en memoria es la correcta?

Representación de memoria	dirección de memoria, descripción del contenido de dirección de memoria, [contenido de la dirección de memoria]	Programa correspondiente
M1	$100, cadena dinámica de main, [-] $104, cadena estática de main,[-] $108, x de main, 1 $112, cadena dinámica de a, [$100] $116, cadena estática de a, [$100] $120, x de a, 2 $124, cadena dinámica de b, [$112] $128, cadena estática de b, [$100] $132, código de c, [-$80000] $132, puntero a cadena estática de c, [$100] $136, x de b, 3 $140, cadena dinámica de f, [$124] $144, cadena estática de f, [$100]	
M2	$100, cadena dinámica de main, [-] $104, cadena estática de main,[-]	

	$108, x de main, 1 $112, cadena dinámica de a, [$100] $116, cadena estática de a, [$100] $120, x de a, 2 $124, cadena dinámica de b, [$112] $128, cadena estática de b, [$100] $132, puntero a cadena estática de c, [$112] $136, x de b, 3 $140, cadena dinámica de f, [$124] $144, cadena estática de f, [$112]	
M3	$100, cadena dinámica de main, [-] $104, cadena estática de main,[-] $108, x de main, 1 $112, cadena dinámica de a, [$100] $116, cadena estática de a, [$100] $120, x de a, 2 $124, cadena dinámica de b, [$112] $128, cadena estática de b, [$112] $132, puntero a cadena estática de c, [$100] $136, x de b, 3 $140, cadena dinámica de f, [$124] $144, cadena estática de f, [$100]	
M4	$100, cadena dinámica de main, [-] $104, cadena estática de main,[-] $108, x de main, 1 $112, cadena dinámica de a, [$100] $116, cadena estática de a, [$100] $120, x de a, 2 $124, cadena dinámica de b, [$112] $128, cadena estática de b, [$112] $132, puntero a cadena estática de c, [$112] $136, x de b, 3 $140, cadena dinámica de f, [$124] $144, cadena estática de f, [$112]	

Plan 2011:

d) Sin considerar pasaje de funciones como parámetro, ¿en cuáles casos la cadena estática de la función **f** no se corresponde con la construida por el método de construcción de cadena estática?

e) Si se cambia el **Programa 4** y se anida la función **c** dentro de **b** ¿qué sucede con el programa? Justifique

Opción	**Respuesta**
1	Imprime 3
2	Error de compilación
3	Imprime 2
4	Imprime 1

Ejercicio 11

Dado el siguiente programa, responder los siguientes incisos:

```
func f() {
}

func main() {
var x = 1
func() {print x}()
var a = func() {print x}
...
a()
}
```

a) Completar la matriz de llamados para la primera expresión lambda
b) Completar la matriz de llamados para la segunda expresión lambda
c) Completar la matriz de llamados para la segunda expresión lambda si la variable a es global

Ejercicio 12

Considere que se ejecuta, en un lenguaje tipo Algol, la siguiente sentencia: x:= y + z;
Ordene los siguientes casos según su eficiencia (rapidez) en tiempo de ejecución:
i)

a) Todas las variables son locales y no estáticas
b) Todas las variables son globales y no estáticas
c) Todas las variables son estáticas
d) Todas las variables son locales y estáticas
e) Todas las variables son globales y estáticas

ii) ¿En qué cambia el ordenamiento si se analiza en tiempo de compilación?
Justifique todas sus respuestas.

Ejercicio 13

Analice similitudes y diferencias de los descriptores de variables dinámicas con nombre con los descriptores de variables semi dinámicas en lenguaje **Algol**.
Indique si son verdaderas o falsas las afirmaciones. Justificar todas las respuestas.

a) Ambos descriptores contienen los mismos campos de información
b) Si los valores que establecen los límites de ambas variables cambian durante el tiempo de vida del registro de activación que las contiene, entonces el tamaño de las variables cambian durante la ejecución del mismo
c) Las instrucciones para ubicar los descriptores de las variables semi dinámicas y las variables dinámicas con nombre se generan en compilación
d) El tiempo de vida de ambos descriptores es el mismo mientras que el tiempo de vida de los datos apuntados no coincide
e) La dirección efectiva donde se encuentran los datos de ambas variables se obtiene durante compilación

Ejercicio 14

Considere un lenguaje de tipo **Algol.**

I. Dadas las siguientes variables, ordenarlas de mayor a menor según su alcance, considerando el contexto de la unidad local determinada. Si una variable no estuviese al alcance, indicarlo.
    a) Variable estática local
    b) Variable global a todas las unidades
    c) Variable semiestática local

d) Variable estática declarada en el abuelo de la unidad
e) Variable dinámica con nombre local
f) Variable semi dinámica declarada en el padre de la unidad
g) Variable anónima apuntada por una variable local
h) Variable estática declarada en el padre de la unidad
i) Variable estática declarada en una unidad hija

II. ¿En qué cambia el ordenamiento en un lenguaje dinámico con alcance dinámico?
III. ¿En qué cambia el ordenamiento en un lenguaje dinámico con alcance estático?

Justifique sus respuestas