Trabajo Práctico N° 05

Pasaje de Parámetros

Recomendaciones Generales e Información

Abreviaciones empleadas durante el práctico:

VL: Verificación de Límites, **VLE**: **VL** en Ejecución, **VLC**: **VL** en Compilación, **TPP**: Tipo de Pasaje de Parámetros

Abreviatura	**TPP**
PCV	Copia Valor
PCVR	Copia Valor Resultado
PR	Referencia
PN	Nombre

Los ejercicios y/o incisos marcados con **Plan2011** corresponden a la materia Lenguajes de Programación I (plan 2011)

Ejercicio 1

Suponga tener el siguiente programa:

```
program MAIN;
begin
  a,b:integer;
  procedure p (integer x, integer y, integer z)
  begin
      y:=y+1;
      z:=z+x;
  end
  a:=2;
  b:=3;
  p(a+b,a,a);
  printf("a = ",a);
end
```

a) Determinar, para los diferentes **TPP**, los resultados que se imprimirán para la variable **a**. Justifique realizando la pila de registros de activación
b) Compare los resultados entre **PR** con **PN** y **PCV** con **PN**. Explique, y en caso de que existan diferencias, indique los motivos por los cuales se producen.

Ejercicio 2

Suponga tener el siguiente programa:

```
program PARCIALITO;
a: array[1 ..3] of integer;
i: integer;
   procedure PARAMETROS (x,y: integer);
   begin
        i = i +1;
        x = x +1;
        printf("i = ",i,"x = ",x,"y = ",y);
   end;
begin
   i =1;
   a[1] =2; a[2] = 3; a[3] = 4;
   PARAMETROS (i,a[i]);
   printf("i= ",i,"a[",i,"] = ", a[i]);
end.
```

Deduzca el **TPP** utilizado, en cada inciso, si los resultados son:

	Resultado	**TPP**
a)	i =3 x = 3 y = 4 i =3 a[3] = 4	
b)	i =3 x = 3 y = 2 i =3 a[3] = 4	
c)	i =2 x = 2 y = 2 i =2 a[2] = 3	

Justifique la respuesta construyendo las pilas de ejecución

Ejercicio 3

Sea la siguiente función

```
int PARAM (a,b,c)
...
m = c;
h = 3 / c;
n = 2 * c;
```

y sea el siguiente llamado

```
u = param(d,e,ZZ(x));
```

a) ¿Cuántas veces se ejecuta la función ZZ si el **TPP** es por
   i. PR?
   ii. PCV?
   iii. PN?

b) Si el código fuese el siguiente

```
int PARAM (a,b,c)
...
m = c;
x = 1;
h = 3 / c;
x = 2;
n = 2 * c;
```

¿En cuál de los **TPP** el resultado se ve afectado?

Ejercicio 4

Dado el siguiente programa en un lenguaje tipo Algol:

```
program MAIN;
x,i: integer;
a: array[1..3] of integer;
procedure YZ(int y,int z);
begin
   y:=i+1;
   z:=a[i]+2;  [1]
end;
begin
   for x:=1 to 3 do a[x]:=x;
   i:=1;       [2]
   YZ(i,a[i]);
   YZ(i,a[i]);
end
```

a) Construya las pilas de registros de activación para cada semántica de pasaje de parámetros. Considere que el pasaje por referencia se implementa mediante alias, y nombre mediante reemplazo textual.

b) Ordene en forma ascendente las semánticas de pasaje de parámetros de acuerdo al tiempo que transcurre desde que comienza el programa hasta que se realiza la primera modificación de la variable i (sin tener en cuenta como modificación la inicialización de la misma por la instrucción `[2]`).

c) Analizar si las siguientes afirmaciones son verdaderas o falsas. Justificar todas las respuestas.

   i) La instrucción `[1]` produce la modificación del mismo elemento del arreglo todas las veces que se ejecuta (considerar todos los tipos de pasaje de parámetro).

   ii) Tanto para los **PCR** como por **PCVR**, el RA de **YZ** tendrá el mismo tamaño y, durante toda la ejecución del procedimiento, los valores de las variables contenidas en **YZ** serán los mismos para ambas semánticas.

   iii) Si el pasaje de parámetros fuera por **PR**, el llamado `YZ(3,i+a[i])` produciría un error de compilación de tipo sintáctico.

   iv) Tanto en **PR** como **PN**, los valores finales del arreglo **a** serán iguales.

Plan 2011:

   v) Si la variable **i** fuera definida como **static i:integer**, se produciría un error al pasarla por referencia.

   vi) Los parámetros reales deben estar al alcance de la unidad llamada.

   vii) Con ninguno de los pasajes de parámetros se modifica el último elemento del arreglo **a**.

viii) En el caso de pasaje de arreglos como **PR** o **PN**, es posible que los parámetros reales estén almacenados por filas y los formales, por columnas.

Ejercicio 5

Sea la siguiente declaración de procedimiento, en un lenguaje tipo **Algol** como:

```
proc sub1 (a,b,c,d: int; e: function()
```

en la que se usa el mismo **TPP** para todos los parámetros,que posee valor por defecto para parámetros faltantes y dado el llamado:

```
sub1(x - y,7, z[i],  ,sub2)
```

Responder si se producen:

Se producen:	PR	PN	PCV	PCR	PCVR
¿Error durante la compilación?					
¿Error en ejecución?					
¿Resultados no esperados? (no son errores)					

a) Para cada caso, indicar el número del parámetro involucrado (1, 2, 3, 4 o 5).
Si son más de uno, indicarlos a todos.
Para todos los casos, justificar la respuesta para cada parámetro que ha señalado en cada columna,

b) ¿Cuáles de parámetros reales y formales existen en el registro de activación de **sub1** y cuáles en el registro de activación del llamador?
¿Cómo se modifican cada uno de éstos en el momento de finalizar la invocación en **PCVR, PN y PCV**?

c) Responder cuáles de las siguientes afirmaciones son verdaderas o falsas justificando todas sus respuestas:
   i. Si el primer parámetro es usado en **sub1** en tres lugares diferentes, la cantidad de veces que se evalúa la expresión es mayor, si el pasaje es por **PN** que por **PCV**.
   ii. El **PR** y **PCVR** producen los mismos errores en los parámetros 3, 4 y 5
   iii. **z[i]**, **x** e **y** deben ser variables locales a la función llamadora
   iv. Los parámetros formales de **sub1** están al alcance de la función llamadora

Plan 2011:

   v. Si el tipo de **z[i]** fuese distinto al del parámetro formal, el tipo de error que se produce tiene que ver con la sintaxis del lenguaje
   vi. El tamaño del registro de activación de **sub1** es igual para **PCV** que para **PN**.

Ejercicio 6 - Plan 2011

Dado el siguiente programa en un lenguaje tipo **Algol**, en el cual:

- Los enteros ocupan **4** bytes
- Las constantes no se almacenan como variables

- La pila crece hacia direcciones crecientes
- El registro de activación del main comienza en la dirección $\mathbf{9A54_h}$
- El registro de activación de la función parcial comienza en la dirección $\mathbf{9A84_h}$

```
static int tamReal=3;
main(){
   const tamFormal=3;
   int i=1;
   proc parcial(int a[1..tamFormal], int index){
      index++;
      print (&a[index]);
      print (&a[i]);
   }
   int c[1..tamReal];
   c[1]=1;
   c[2]=2;
   c[3]=3;
   parcial(c, i);
}
```

a) Clasificar todas las variables de acuerdo a su almacenamiento.

b) Determine la/s semántica/s de **TPP** que fue utilizada para generar cada una de las siguientes salidas, justificando con la pila de ejecución correspondiente:

Salida 1:	**Salida 2:**
9A9C 9A98	9A7C 9A7C

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

c.1) Es posible comprobar la compatibilidad absoluta de los parámetros reales con los formales en tiempo de compilación.

c.2) El momento en que se modifica la variable **i** es posterior en el caso de **PCVR** que en **PN**.

c.3) El programa no produciría ningún error en un lenguaje dinámico.

c.4) El tamaño de la pila de **RA** va a ser igual para la semántica nombre que para referencia usando punteros.

c.5) La variable **tamReal**, la variable **i** y la constante **tamFormal** tienen el mismo alcance y están almacenadas en el mismo **RA**.

c.6) Para todas las semánticas de **TPP** es posible dejar un parámetro faltante si el mismo tiene un valor por defecto.

Ejercicio 7

Considerando un lenguaje con las siguientes características:

- Las constantes no se almacenan como variables
- Verifica límites en tiempo de ejecución
- El operador | | obtiene la **cantidad total de bytes** que ocupa una variable
- La palabra reservada **din** permite declarar arreglos como variables dinámicas con nombre

a) Analizar en el código de más abajo, para la invocación marcada con **//1**, si es posible realizarla con cada uno de los cinco parámetros reales del cuadro para las distintas semánticas de **PP**. Indicar en qué casos podrían producirse resultados inesperados.

	a : w	b : z	c : w+*z[1]	d : 2	e : fun1	Justificación
CV						
CVR						
CR						
Referencia						
Nombre						

```
def principal(): {
const int w = 3
def fun2(q=3):
{
	def fun1(n):
	{
		int s[1..w]
		int r[1..n]
		din t = [w,w,w,w]
		print(&r,&s, &t, &z)
		print(|r|,|s|,|t|,|z|)
	}
	int * z[1..w]                                  //2
	for (i=1; i++,i<=w): z[i] = nuevo int          //3
	fun1(w)
	fun3(w, z, w+*z[1], 2, fun1)           //1
}
def fun3(a, b, c, d, e):
{
	*b[d] = d++
	b[d] = &c
	e = fun2
	e(a)
}
fun2()
}
```

b) Indicar, para cada parámetro, qué modificadores *(in, out, in out)* podrían utilizarse en la llamada de la línea **//1** si este lenguaje replicara las semánticas de **PP** implementadas en ADA.

	a : w	b : z	c : w+*z[1]	d : 2	e : fun1	Justificación
in						
out						
in out						

c) Ubicar las variables presentes en el programa de acuerdo a su lugar de almacenamiento en el esquema de memoria, considerando pasaje de parámetros por copia.

	Estáticas	RA principal	RA fun1	RA fun2	RA fun3	Heap
Variables						

Plan 2011:

d) Indicar qué salidas serían posibles en la llamada a fun1(). Justifique su respuesta.

Salida 1	Salida 2	Salida 3	Salida 4

$136, $160, $4200, $116	$160, $172, $4200, $116	$4200, $4212, $4224, $4188	$136, $148, $4200, $116
12. 24, 28, 12	24, 24, 28, 12	24, 24, 28, 24	12, 12, 28, 12

e) Responda si las siguientes afirmaciones son verdaderas o falsas. Justifique las respuestas.
   i) Si se eliminase la línea **//3** y cambiase la declaración en la línea **//2** por **int z[1..q],** se utilizarían menos bytes de memoria para su almacenamiento.
   ii) Si se quitara el modificador ***const*** de la variable **w**, podría producirse fragmentación al usarla para declarar arreglos.

Ejercicio 8

Dado el siguiente programa en un lenguaje **XX** tipo **Algol** donde la forma de definir el **TPP** es similar a la de **Pascal**:

```
function sub1() {
  int x;
  int y;
  function sub2(int d,…) {
    printf (x);
    x:= d;
    printf (x);
  }
  function sub3(var int a, var int b, int c) {
   int x;
   x:= 3;
   a:= b+c;    [1]
   a:= b+c;    [2]
   sub4(sub2());
  }
  function sub4(function subx) {
     int x;
     x:= 4;
     subx(x);
 }
 x:= 1;
 y:= 2;
 sub3(x, x, x+y);
}
```

Construya la pila de ejecución. Luego, indique cuáles de las siguientes afirmaciones son verdaderas y cuáles falsas. Justifique TODAS sus respuestas.

a) El lenguaje **XX** no admite en su sintaxis parámetros anónimos variables.
b) Todas las variables que corresponden a los parámetros formales de la función **sub3** se crean en el **RA** de **sub3** al ejecutarse el programa.
c) La variable **d** es una variable global a la función **sub2**.
d) Las dos instrucciones `printf(x)` dan como resultado el mismo valor.
e) Durante la ejecución de **sub3** ocurre que `[1]` y `[2]` arrojan resultados diferentes de **a**.
f) Si se cambiase la definición del prototipo de function **sub3** por `function sub3(int a,int b,int c)`, los resultados del programa no variarán.
g) Si el lenguaje **XX** en su sintaxis de **TPP** usase el prefijo **var** para la semántica de **PN**, los resultados de `[1]` y `[2]` serían los mismos que cuando el prefijo var se usó para **PR**.
h) Si el lenguaje **XX** en su sintaxis de pasaje de parámetros usase el prefijo **var** para la semántica de pasaje **PCVR**, los resultados de `[1]` y `[2]` serían idénticos entre sí y diferentes a los que dieron cuando el prefijo var se usó para **PR**.

i) Si se reemplaza el prototipo de definición para la función **sub3** por `function sub3(var int a, var int b, var int c)` se podrían producir errores de ejecución.

Ejercicio 9

Se tiene el siguiente programa en un lenguaje tipo **Algol**:

```
func z() {
  int x=0;          //*0
  func a() {
      int x=1;      //*1
      func b() {
          print x;
       }
       e(b);
  }
  func c() {
      int x=2;      //*2
      func d() {
          print x;
       }
       e(d);
   }
   func e(f) {
          int x=3; //*3
          f();
   }
   a();
   c();
}
```

a) Indicar qué y por qué imprimirá el programa si el lenguaje tiene pasaje de parámetros de ámbito profundo
b) Realizar un diagrama mostrando la pila de ejecución y la ubicación de las variables
c) ¿Cómo cambian las impresiones si se elimina la instrucción `*3`? Justifique
d) ¿Cómo cambian las impresiones si se elimina la instrucción `*2`? Justifique
e) ¿Qué sucede si se eliminan las instrucciones `*1` y `*0`? Justifique

Ejercicio 10

Dado el siguiente fragmento de código:

a) Para cada una de las salidas, indique y **justifique brevemente**:
   i) ¿Qué **tipo de alcance** tiene el lenguaje?
   ii) ¿Cuál **semántica de pasaje de parámetros** usa para variables de tipo entero?

Aclaración: *Todos los lenguajes capturan, en caso de closures, las variables declaradas en la función que las devuelve.*

Código

Salida 1	**Salida 2**

```
1   int x = 1
2
3   func PP(int a, func c) {
4     a++
5     print "llegó $a y leo $x"
6     c()
7     c()
8   }
9
10  func devuelveclosure() {
11    int y = x \\1
12    print "solo devuelvo una closure"
13    return closure () {
14        \\2
15        print "y es $y, qué $x usé?"
16        y++
17      }
18  }
19
20  func main() {
21    int x = 4
22    z = devuelveclosure()
23    PP(x, z)
24    print "fin $x"
25  }
```

"solo devuelvo una closure" "llegó 5 y leo 1" "y es 1, qué 1 usé?" "y es 2, qué 1 usé?" "fin 4"	"solo devuelvo una closure" "llegó 5 y leo 5" "y es 4, qué 5 usé?" "y es 5, qué 5 usé?" "fin 5"
Salida 3	**Salida 4**
"solo devuelvo una closure" "llegó 5 y leo 1" "y es 1, qué 1 usé?" "y es 2, qué 1 usé?" "fin 5"	"solo devuelvo una closure" "llegó 5 y leo 4" "y es 4, qué 4 usé?" "y es 5, qué 4 usé?" "fin 4"

b) Realizar la tabla de símbolos o el diagrama de pila (según corresponda) que refleje el comportamiento del fragmento de programa para:
   i) Salida 1
   ii) Salida 2

c) Para cada afirmación, indicar si es **verdadera o falsa**, **justificando todas sus respuestas**.
   i) De acuerdo a la Salida 1, es posible afirmar que en este lenguaje las closures no tienen sensibilidad a la historia.
   ii) El pasaje de parámetros permite que las unidades puedan leer valores de variables que no tienen al alcance.

d) Si se mueve la línea comentada con **//1** a la línea comentada con **//2**, indicar qué imprimiría el programa para:
   i) Lenguaje de alcance estático, pasaje de parámetros de enteros por copia.
   ii) Lenguaje de alcance dinámico, pasaje de parámetros de enteros por copia.

Ejercicio 11

En un lenguaje tipo Algol con verificación de límites de arreglos, considere los siguientes casos de declaración de una función f que recibe un arreglo como parámetro. **El lenguaje almacena los arreglos por filas.** La llamada a **random(1,10**) devuelve un número entero al azar entre 1 y 10.

A.	`void f(int x[1..4][1..6]) {`
	`  x[3][4] = 5;`
	`}`
		B.	`void f(int x[1..4][]) {`
			`  x[3][4] = 5;`
			`}`

C.	`void f(int x[1..4][1..6]) {`
	`  x[6][8] = 5;`
	`}`
		D.	`void f(int x[1..4][1..6]) {`
			`  x[random(1,10)][4] = 5;`
			`}`

E.	`void f(int x[][]) {`
	`  x[3][4] = 5;`
	`}`
		F.	`void f(int x[][1..6]) {`
			`  x[3][4] = 5;`
			`}`

y la función principal del programa:

```
int main() {
          int a[1..4][1..6];
          f(a);
}
```

Indicar para cada caso si:

- **i.** Se produce un error de compilación
- **ii.** Se puede producir un error en tiempo de ejecución
- **iii.** Se puede producir un resultado inesperado
- **iv.** No se produce ningún problema

Fundamentar las respuestas explicando **por qué** se producen tanto los errores de compilación o de ejecución, como resultados inesperados.

Ejercicio 12

Considerando el siguiente código y la salida correspondiente en un lenguaje tipo Algol, en el cual las variables estáticas se ubican en direcciones entre $10 y $90, los registros de activación se ubican en direcciones a partir de $100 y los enteros ocupan 4 bytes *(No se considera el espacio que ocupan las cadenas estática y dinámica)*. El lenguaje no almacena las direcciones del código ejecutable de la unidad del closure (llamada estática). Los registros de activación de cada unidad se inician en múltiplos de 100.

Responder los incisos:

- **a)** Realizar los diagramas de las pilas de ejecución para los momentos donde se terminan de ejecutar cada una de las siguientes líneas de código:
  - 20
  - 7 *(durante la llamada en 21)*
  - 10 (*durante la llamada en 21)*
  - 7 *(durante la llamada en 22)*
  - 10 *(durante la llamada en 22)*
  - 23
  - 7 *(durante la llamada en 24)*
  - 10 *(durante la llamada en 24)*
  - 7 *(durante la llamada en 16)*
  - 10 *(durante la llamada en 16)*

Código		**Salida 1**
1	`static int w = 1`	`5`
2	`func f1() {`	`$10, $104, $200`
3	`  int x = 2`	`8`
4	`  return () {`	`$10, $104, $200`
5	`      int z = x`	`7`
6	`      print w+x+z`	`$10, $108, $200`
7	`      print &w, &x, &z`	`$200`
8	`      w++`	`10`
9	`      x++`	`$10, $108, $300`
10	`      z++`	`$100, $104, $108`
11	`    }`	

```
}

func f2(func c) {
  int s = 1
  print &s
  c()
}

func main() {
   int y=0
   c1 = f1()
   c1()
   c1()
   c2 = f1()
   c2()
   f2(c2)
   print &y, &c1, &c2
}
```

b) Responder si las siguientes afirmaciones son verdaderas o falsas, fundamentando las respuestas

i) Cuando se construye un closure, el lenguaje captura las variables no locales (al closure) por referencia

ii) Cuando se construye un closure, el lenguaje captura las variables globales por copia

iii) Cuando se pasa un closure como parámetro, se lo hace por copia valor.

iv) Los closures de este lenguaje tienen sensibilidad a la historia. Si la afirmación es verdadera, indicar cómo el lenguaje podría evitar la sensibilidad, si es falsa indicar por qué lo es.

v) Los closures de este lenguaje pueden producir efectos laterales. Si la afirmación es verdadera, indicar cómo se podría evitar los efectos laterales, si es falsa indicar por qué lo es.

Ejercicio 13 Plan 2011

Dada la siguiente gramática en la que las reglas comienzan con **'?'** seguidas del no terminal de definición, seguido del símbolo **':'** (se define como) y a continuación una secuencia de no terminales en minúscula y terminales entre comillas dobles.
Suponer definiciones convencionales para los no terminales: **id**, **iteración**, **seleccion** y **asignación**, por ejemplo las del lenguaje **Java**.

1. ?programa: declaraciones instruccionesejecutables
2. ?declaraciones: declaracionesvariables declaracionesfunciones
3. ?declaracionesvariables: tipo id | declaracionesvariables tipo id
4. ?declaracionesfunciones: definicionfuncion | declaracionesfunciones definicionfuncion
5. ?definicionfuncion: tipo id "(" parametrosformales ")" cuerpo
6. ?cuerpo: "{" declaraciones instruccionesejecutables retorno "}"
7. ?retorno: "return" id
8. ?parametrosreales: parametroreal | parametrosreales ";" parametroreal
9. ?parametroreal: id | invocacionfuncion
10. ?invocacionfuncion: id"("parametrosreales")"
11. ?tipo: "int" | "float" | "auto"

12. ?parametrosformales: parametroformal | parametrosformales ";" parametroformal
13. ?parametroformal: parametroformalvariable | parametroformalfuncion
14. ?parametroformalvariable: modificadorsemantica tipo id
15. ?parametroformalfuncion: encabezadofuncion
16. ?encabezadofuncion: tipo id "(" parametrosformales ")"
17. ?modificadorsemantica: "in" | "out" | "inout" | "ref" | "name"
18. ?instruccionesejecutables: iteracion | seleccion | asignacion

Responder los siguientes incisos, indicando el conjunto de reglas que lo permiten.
En caso contrario, modificar la gramática y/o proponer reglas nuevas para que el lenguaje lo permita:

i)

a) El lenguaje admite que una función no retorna un resultado
b) El lenguaje posee diferentes semánticas de pasaje de parámetros para variables
c) El lenguaje posee diferentes semánticas de pasaje de parámetros para funciones
d) El lenguaje permite recibir una función como parámetro
e) El lenguaje permite retornar el valor resultante de ejecutar una función
f) El lenguaje permite intercalar declaraciones de funciones y variables en cualquier lugar del programa

ii) Para los siguientes programas indicar si son aceptados o no con la gramática original, fundamentando por qué. En caso de no ser aceptado proponer modificaciones a las reglas para que se acepte.

a) `int a(in int b;int a(out int b)) {int a x=3 return a(a)} x=3`

b) `int b(int a(int b)) {auto a x=3 return b} x=3`

Ejercicio 14

Considere un lenguaje que que implementa las distintas semánticas de **TPP** donde sólo es posible uno de los siguientes **TPP** a la vez para un llamado:

- **PCV**
- **PCR**
- **PCVR**
- **PR** implementado con alias
- **PR** implementado con puntero al parámetro real
- **PN** implementado con reemplazo textual del parámetro formal por el real y re-compilación

a) Ordenar en forma decreciente las semánticas de acuerdo al tamaño que ocupa el **RA** de la función llamada.
b) ¿En cuáles casos se necesita de la cadena estática para ubicar el parámetro real?
c) ¿En cuáles casos se dan resultados inesperados si se quiere pasar un elemento de un arreglo?
d) ¿En cuál semántica el parámetro real se actualiza instantáneamente luego de su uso en una instrucción?
e) ¿En cuáles casos se produce un error en tiempo de ejecución si se pasa una constante?
f) Para cada caso ¿qué ocurre si se pasa una variable estática global como parámetro?
g) En el inciso a) qué ocurre si se pasa un parámetro cuyo tipo es más pequeño que un puntero.
h) ¿En cuáles semánticas puede haber errores de compilación debido a problemas de alcance? Ejemplificar Proporcionar un ejemplo de un caso donde se produzca un error en tiempo de ejecución durante el llamado.