Diseño y Paradigmas de Lenguajes - Año 2014
Práctico Nro. 4
Control de secuencia y datos en subprogramas
Corresponde a Cap. VII Control de Subprogramas (Pratt en Español)

Ejercicio 1.

Considerando el siguiente código en lenguaje C, muestre gráficamente como van cambiando las estructuras de la activación de cada subprograma, en cada paso de la ejecución. Muestre los segmentos de códigos de los subprogramas y como se modifican la pila de ejecución y el heap. Tenga en cuenta que deben ser almacenados los valores correspondientes al *punto de regreso (retorno)*: (pi, pe) y los punteros CIP y CEP deben ser actualizados cuando corresponda.

```
int b=5;

int M(int d){
 static int c=1;
  c=c+d*b;
  return c;
}
void P(){
 int a=0;
 if (b!=0){
     b=0;
     P();
   }
  else a=1;
}
void N(int* u){
 int* v;
  (*u) = b;
  v = (int*)malloc(sizeof(int));
  v = u;
  P();
}
void main(){
 int* t = (int*)malloc(sizeof(int));
 b=M(5)
 N(t);
}
```

Ejercicio 2.

Especifique los ambientes de referenciación locales y no locales, de cada uno de los procedimientos y del programa principal del siguiente código en un lenguaje hipotético que soporta anidamientos.

```
Program Principal;
  var b,f: real;

  Procedure P(var e: real);
    Function Q(c: integer): integer;
      var b: integer;
      Procedure R(c: real, d: char);
         begin
           f:=2+e+c
         end;
      begin
       c:=3;
       b:=5;
       R(e,'d');
       Q:=5
      end;
    begin
     e:=7.0;
     f:=3.0;
     Q(4)
    end;

  Procedure U(e: real, c: real);
    var f: real;
    begin
     f:= e + b * 2.0 + c;
     P(f)
    end;
begin
 b:=5.5;
 f:=15.0;
 U(f,f*2)
end.
```

Ejercicio 3.

Dado el código tipo Pascal que se lista a continuación, se pide:

1. Considerando los ambientes de referenciación de cada subprograma, responda a las siguientes preguntas justificando claramente sus respuestas:
   a) En el procedimiento **P2** ¿Es posible acceder a la variable **a** de Principal?
   b) En el procedimiento **P4** ¿Es posible acceder a la variable **d** de **P1**?
   c) En el procedimiento **P4** ¿Es posible acceder al parámetro formal **b** de **P1** ?
2. ¿Existe algún punto del programa en el que dos o más variables se constituyan en *alias*? En caso afirmativo diga dónde y porqué.
3. Proponga al menos un alias, introduciendo modificaciones al código.

```
Program Principal;
  type arreglo=array[0..5]of integer;
  var a:integer;
      b: real;
      c: arreglo={2,4,6,8,10,12};

  Function P1(var a:integer, var b :integer): integer;
    var aa: real;
        c,d: integer;
    Procedure P2(d: integer);
      var bb, b: integer;
      begin
        b:=3;
        bb:= d + a + c;
      end;
    Procedure P3(a: real);
      begin
        a:=10.3;
        P2((trunc(a))
      end;
    begin
      P3(2.14);
      a:=2;
      b:=3
      aa:=15.0;
      P1:=1
    end;
  Function P4(var c:integer, var a:arreglo): integer;
    var aa:integer;
         d:integer;
    begin
      d:=5;
      aa:=P1(d,d);
      a[1]:=2;
      c:=c+2;
      P4:=c
    end;
   begin
     a:=1;
     b:=1.0;
     a:=P4(c[a],c);
   end.
```

Ejercicio 4.

Considere el siguiente programa en lenguaje C:

```
int a[3],k;

 void operacion(int r, int s) {
  int t;

  k++;
  r=2;
  printf("%d, %d\n", r, a[0]);
  a[0] = a[1] * 3;
  s = r + 1;
  printf("%d, %d\n", s, a[1]);
  t = s-1;
  a[t] = a[t] + k;
 }
```

```
int main() {
 int i;

  for(i=0; i < 4;i++) a[i] = i;
  k=1;
  operacion(a[0], a[k]);
  return 0;

}
```

Muestre el valor de cada una de las variables al finalizar la ejecución del programa y los carteles que muestra. Considere que todos los parámetros son pasados:

1. Por referencia.
2. Por valor.
3. Por valor-resultado.
4. Por nombre.

Nota: Asuma que el lenguaje provee estos métodos de pasaje de parámetros.

Ejercicio 5.

Dado el siguiente fragmento de código en el lenguaje C:

```
int i=2, j=3, k[3][4];

k[0][0]=9; k[0][1]=8; k[0][2]=7; k[0][3]=6;  //fila 1
k[1][0]=5; k[1][1]=4; k[1][2]=3; k[1][3]=2;  //fila 2
k[2][0]=1; k[2][1]=2; k[2][2]= -1; k[2][3]= -2;  //fila 3

void p (int a, int b, int c) {
a=1; b=0; print(c);
while(b < 4){
   c=c*5;
   b++;
 }
}
int main() {
 p(i,j,k[i][j]);
}
```

a) Ejecute el programa, mostrando paso a paso la ejecución. Considere que C implementa los pasajes de parámetros por valor, referencia y nombre, y que a esta pasado por valor, b por referencia y c por nombre.

b) Ejecute el programa, mostrando paso a paso la ejecución. Considere que todos los parámetros son pasados por referencia.

Ejercicio 6.

Para el código que se muestra a continuación en un lenguaje que soporta anidamientos y asumiendo que se utiliza una regla de alcance estático, se pide:

1. Muestre cómo queda la pila de ejecución en el caso en que la regla de alcance estático se implemente utilizando la cadena estática.
2. Muestre cómo queda la pila de ejecución en el caso en que la regla de alcance estático se implemente utilizando el visualizador.
3. ¿Qué mecanismo (cadena estática o visualizador) es más eficiente? ¿Por qué?
4. En el subprograma `Proc2` se accede a la variable `a`, calcule cómo se accedería a esta variable cuando se usa la cadena estática. Ahora suponga que se utiliza una regla de alcance estático implementada con el visualizador, ¿Cómo se accedería a la variable `a`?

```
Program Principal
  var a,b: integer;
        l: boolean;
  Procedure Pro1 (c:integer)
    var d,e:integer;
    Function Pro2 (f:integer): integer;
      var e:char
      begin
        ...
        a := 5;
      end;
    Procedure Pro3 (h:integer)
       var d:real;
       begin
         ...
         h := Pro2(h);
       end
    begin
      ...
      Pro3(d);
      ...
    end
  Procedure Pro4 (a:boolean)
    var c,e:integer;
    Procedure Pro5 (var l:integer)
      var m,a:real;
      begin
        ...
      end
    begin
      ...
      Pro1(c);
      ...
    end
begin
  ...
  Pro4(l);
  ...
end.
```

Ejercicio 7.

Dado el siguiente código de programa, tipo C:

```
void Main() {
 int a,b,c;
 Fun2()
}

void Fun1(int c){
 int d;
 b=10;
 ....
}

void Fun2(){
 int b,e;
 Fun3()
}

void Fun3(){
 int c,a;
 Fun1(a)
}
```

1. Muestre como queda la pila de ejecución cuando se está ejecutando la última función invocada.
2. Implemente la regla de alcance dinámico utilizando la búsqueda directa en la pila de ejecución.
3. Muestre paso a paso la implementación de la regla de alcance dinámico que utiliza la Tabla Central y Pila Oculta de toda la ejecución del programa.
4. ¿Cómo se resuelve la referencia a la variable `b` en el procedimiento `Fun1`, cuando se utiliza la búsqueda directa? y ¿cuándo se utiliza la tabla central y pila oculta?