

Fundamentos del Diseño de Lenguajes de Programación - 2024
Práctico Nro. 5
Control de secuencia y datos en subprogramas
Corresponde a Cap. IX Control de Subprogramas

Ejercicio 1.

Considerando el siguiente código en lenguaje C, muestre gráficamente como va cambiando la estructura de la activación del subprograma, en cada paso de la ejecución. Muestre los segmentos de códigos y como se modifica la pila de ejecución. 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, teniendo en cuenta que I1, I2, etc. indican el número de instrucción. Utilice la herramienta *pythontutor* para verificar la ejecución.

```
int mod(int n, int m){
I3 int resu;
I4 if (n < m)
    I5 return n;
    else {
        I6 resu = mod(n - m,m);
        I7 return(resu);
    I8 }
I9 }

void main(){
    I1 int i = mod(5,2);
I2 }
```

Ejercicio 2.

A partir del siguiente código en un lenguaje hipotético que soporta anidamientos, teniendo en cuenta que I1, I2, etc. indican el número de instrucción:

```
void Main()
{ // inicio de Main
    I1 int a = 7;
    I2 int b = 2;
    void SubB(int d)
    { // inicio de SubB
        I3 char b = 'a';
        I4 int c = 4;
        void SubA(int c, float b)
        { // inicio de SubA
            I5 char e = 'f';
            void SubC(int a, int b)
            { // inicio de SubC
                I6 a = c + 2;
                I7 } // fin de SubC
            I8 SubC(c,7);
        I9 } // fin de SubA
        I10 c += 2;
        I11 SubA(a, c + 3.0);
    I12 } // fin de SubB
I13 SubB(4);
I14} // fin de Main
```

Se pide:

1. Considerando los ambientes de referenciación de cada subprograma, responda a las siguientes preguntas justificando claramente sus respuestas:
 - a) En el subprograma **SubC** ¿Es posible acceder a la variable local **c** de **SubB**?
 - b) En el subprograma **SubB** ¿Es posible acceder a la variable local **e** de **SubA**?
 - c) En el **Main** ¿Es posible invocar al subprograma **SubA**?
2. Complete la siguiente tabla con los ambientes de referenciación locales y no locales, de cada uno de los subprogramas.

	local	no local
<i>SubC</i>		
<i>SubB</i>		
<i>SubA</i>		
<i>Main</i>		

Ejercicio 3.

Diga que variables son visibles en los bloques del siguiente código con la sintaxis inspirada en C++:

```
char r = 'a'; char k = '2';
void main() { # Inicio bloque #
int r = 2; int j = 5;
    { # Inicio bloque #
        int j = 0; int i = 0;
        while (i <= 5){ # Inicio bloque #
            int j = 1;
            i = i + r;
        };
        r = k;
    };
}
```

Ejercicio 4.

Para el código en C dado a continuación:

```
int b = 4; int* q;
void F1(int* a1, int* a2){
    int* s;
    (*a1)--;
    (*q) = (*a1) + 2;
    s = q;
    (*a2) = b + 5;
    b = (*s);
}
void main(){
    q = (int*)malloc(sizeof(int));
    F1(&b,q);
}
```

¿Existe algún punto del código en el que dos o más variables se constituyan en *alias*? En caso afirmativo diga dónde y porqué.

Ejercicio 5.

Para el siguiente código en un lenguaje que soporta anidamientos y asumiendo que se utiliza una regla de alcance estático, teniendo en cuenta que I0, I1, etc. indican el número de instrucción:

```
void Main()
{ // inicio de Main

I0 char a = 'p';
I1 int  c = 3;
I2 int  f = 4;
void SubA(int d)
    { // inicio de SubA
I3 float c = 7.2;
I4 int g = 2;
    void SubB(char h, int a, int g)
        { // inicio de SubB
            int SubC(int a, int c)
                { // inicio de SubC
I5 char i = 'D';
I6 return (c + g);
I7 } // fin de SubC

            void SubD()
                { // inicio de SubD
I8 float g = 6.5;
I9 int f = 4;
I10 f = SubC(d,9);
I11 } // fin de SubD
I12 SubD();
I13 } // fin de SubB
I14 g += 2;
I15 SubB('s',g,7);
I16 } // fin de SubA
I17 SubA(c + 3);
I18} // fin de Main
```

- Determine los ambientes de referenciación locales y no locales para los subprogramas SubB y SubC.
- Muestre cómo queda la pila de ejecución en el caso en que la regla de alcance estático se implemente utilizando la cadena estática hasta la ejecución de subprograma SubD inclusive.
- Muestre cómo queda la pila de ejecución en el caso en que la regla de alcance estático se implemente utilizando el visualizador hasta la ejecución de subprograma SubC inclusive.
- En el subprograma SubC se accede a la variable g, 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 g? ¿Cuál de las dos implementaciones del alcance estático es más eficiente para el acceso de referencias no locales?

Ejercicio 6.

Dado el siguiente código de programa, tipo JavaScript:

```
// programa principal
var x = 2, y = 3, z = 5;
function fun1() {
  var a = 4, y = 9, z = 1;
  . . .
}
function fun2() {
  var a = 6, b = 7, z = 3;
  . . .
}
function fun3() {
  var a = 2, x = 7, w = 6;
  a = y;
}
```

Considere la siguiente secuencia de llamadas a subprogramas, y asuma que se utiliza Alcance Dinámico:

main invoca a fun1; fun1 invoca a fun2; fun2 invoca a fun3.

Se pide para cada secuencias de llamadas:

- a) ¿Qué variables son visibles durante la ejecución de la última función invocada?. Especifique (con cada variable) el nombre de la función en la que fue definida.
- b) Implemente la regla de alcance dinámico utilizando la búsqueda directa en la pila de ejecución.
- c) Muestre paso a paso la implementación de la regla de alcance dinámico que utiliza la Tabla Central y Pila Oculta hasta el retorno de la invocación de *fun2* .
- d) ¿Cómo se resuelve la referencia a la variable *y* en el subprograma *fun3*, cuando se utiliza la búsqueda directa? y ¿cuándo se utiliza la Tabla Central y Pila Oculta?

Ejercicio 7.

Para el siguiente código en un lenguaje hipotético con sintaxis similar a C, teniendo en cuenta que I0, I1, etc. indican el número de instrucción:

```
void Main()
{ // inicio de Main
I0 int    r = 7;
I1 float  t = 8.3;
I2 int    k = 5;
void SubZ(char r, int u)
    { // inicio de SubZ
        I3 float s = 4.0;
        I4 s = t + 1;
    I5 } // fin de SubZ
void SubT(float i, int j, int t)
    { // inicio de SubT
        void SubU(float k)
            { // inicio de SubU
                float SubV()
                    { // inicio de SubV
                        I6 char r = 'm';
                        I7 float k = 5.2;
                        I8 k += t;
                        I9 SubZ('d',2);
                        I10 return (k);
                    I11 } // fin de SubV
                I12 k = SubV();
            I13 } // fin de SubU
        I14 j++;
        I15 SubU(i);
    I16 } // fin de SubT
I17 SubT(6.4, r + 5, 3);
I18} // fin de Main
```

Se pide:

- Muestre paso a paso la implementación de alcance dinámico que utiliza la Tabla Central y Pila Oculta, hasta la sentencia marcada con *I6*.
- Ahora suponga que se utiliza una regla de alcance estático implementada con la cadena estática, en el *SubV* se accede a la variable *t*, dé la fórmula de acceso a esta variable.