Introducción a la Algorítmica y Programación (3300)

Prof. Ariel Ferreira Szpiniak - aferreira@exa.unrc.edu.ar
Departamento de Computación
Facultad de Cs. Exactas, Fco-Qcas y Naturales
Universidad Nacional de Río Cuarto

Teoría 6

Acciones y funciones anidadas,

Reglas de alcance.

Novedades UNRC

- El 1º de mayo la **UNRC** cumple 47 años, fue creada en **1971** dentro de un programa de adecuación de la enseñanza universitaria argentina a las necesidades del desarrollo y como respuesta a un fuerte movimiento social tanto local como regional que permitió la más grande conquista cultural de la región.
- Su creación fue un hito trascendente en el que participaron todos los sectores sociales de la comunidad local y regional con esfuerzo tenaz.
- Nuestra **Facultad (FEFQyN)** fue creada en **1975**, y las **carreras de Computación** en **1992**.

Novedades

Escudo

Simbolismo
El escudo que identifica a la UNRC fue seleccionado en un concurso nacional.

- La franja superior es una guarda indigenista que representa la tradición.
- Las iniciales de la Universidad sobre el cielo simbolizan la concreción de las aspiraciones de la juventud riocuartense, y de los hombres y mujeres que participaron en la obtención de una universidad nacional, pública y gratuita, para la ciudad y zona de influencia.
- Las montañas representan el empuje de la juventud, y la solidez y aplomo de los mayores que la guían.
- El río (franja celeste) el origen del nombre de la ciudad. El rojo, la pachamama, madre tierra.

Novedades

Escudo

Significación del lema

CREER... CREAR... CRECER...
Representa la vocación de la juventud en devenir hombres y mujeres que en libertad y dignidad aspiran al desarrollo nacional.

***CREER** que es posible transformar para **CREAR** nuevas opciones para que todos podamos **CRECER**.*

Son caminos a través de los cuales se manifiesta la UNRC, definiendo al mismo tiempo su relación con el hombre, con el mundo y con su época.

Abstracciones anidadas

- Las abstracciones anidadas se refieren a la posibilidad de declarar funciones y acciones dentro de otras funciones y acciones.
- El anidamiento también puede ser mixto, es decir, una acción puede tener definida una función dentro de ella (interna) y una función puede tener definida una acción dentro de ella (interna).
- La definición de las acciones o funciones internas deben aparecer en el léxico, es decir, junto a la declaración de las constantes y variables locales de la acción o función que las incluye.
- Es importante notar que las acciones y funciones internas son de uso ***exclusivo*** de la acción o función en que fueron declarados.

Abstracciones anidadas
Ejemplo

- Deseamos escribir algoritmo que contenga una acción **CalcSueldo**, que calcule el salario neto en pesos ($) de un empleado. La acción lee de la entrada estándar: (a) el salario bruto de un empleado, dado por un real con un signo ($, D o E), (b) el tipo de empleado, dado por los valores 1 o 2.
- Si el tipo de empleado es 1, el valor neto del salario se obtiene de aplicarle un único descuento al salario bruto, mientras que si el tipo es 2 se le aplican dos descuentos al salario bruto.
- Los porcentajes de descuento son leídos por el algoritmo, desde la entrada estándar, y pasados como parámetros a la acción **CalcSueldo**.

Abstracciones anidadas
Ejemplo (cont.)

Supongamos ya realizadas las etapas de análisis y diseño.

```
Algoritmo CalcularSueldo
Lexico
  TMoneda =(P,D,E) //tipo de moneda Pesos, Dolares o Euros
  desc1, desc2 ∈ R // descuentos
  tipoMoneda ∈ TMoneda //moneda en que cobra el sueldo
  salBruto, SalNeto ∈ R //salarios bruto y neto
  Acción ObtenerDescuentos(resultado d1, d2 ∈ R)
  Acción CalcSueldo(dato d1,d2 ∈ R, resultado bruto,neto ∈ R)
Inicio //CalcularSueldo
  ObtenerDescuentos(desc1, desc2)
  CalcSueldo(desc1, desc2, salBruto, salNeto)
Fin //CalcularSueldo
```


Abstracciones anidadas
Ejemplo (cont.)

Las acciones estarían definidas de la siguiente manera:

```
Acción ObtenerDescuentos(resultado d1, d2 ∈ R)
Inicio
  Entrada:d1 d2
Facción
Acción CalcSueldo(dato d1, d2 ∈ R, resultado bruto, neto ∈ R)
Lexico local
  Acción SignoMonetario(resultado signo ∈ TMoneda)
  Acción ObtenerSalBruto(dato signo1 ∈ TMoneda, resultado bruto1 ∈ R)
  Acción CalcNeto(dato d1, d2, bruto2 ∈ R, resultado neto1 ∈ R)
Inicio //CalcSueldo
  SignoMonetario(tipoMoneda)
  ObtenerSalBruto(tipoMoneda, bruto)
  CalcNeto(d1, d2, bruto, neto)
Facción //CalcSueldo
```

Abstracciones anidadas
Ejemplo (cont.)

```
Acción SignoMonetario(resultado signo ∈ TMoneda)
Inicio
 Entrada:signo // tipo de moneda del sueldo: P, D, E
Facción
Acción ObtenerSalBruto(dato signo1 ∈TMoneda,resultado bruto1∈R)
Lexico local
  bruaux ∈ R
  Función Convertir(dato b ∈ R, s ∈ Tmoneda) → R
  Inicio
    según
      s = P: ← b
      s = D: ← b * 20,60
      s = E: ← b * 23,90
    fsegún
  Ffunción
Inicio //ObtenerSalBruto
  Entrada:bruaux
  bruto1 ← Convertir(bruaux, signo1)
Facción //ObtenerSalBruto
```


Abstracciones anidadas
Ejemplo (cont.)

```
Acción CalcNeto(dato d1, d2, bruto2 ∈ R, resultado neto1 ∈ R)
Lexico local
  tipoEmp ∈ 1..2
Inicio
  // el tipo de empleado es 1 o 2
  Entrada:tipoEmp
  según
    tipoEmp = 1: neto1 ← bruto2 - bruto2 * d1 / 100
    tipoEmp = 2: neto1 ← bruto2 - bruto2 * (d1+d2) / 100
  fsegún
Facción
```


Identificadores locales

- Las acciones y las funciones pueden declarar identificadores de uso local.
- Estos identificadores son locales en el sentido de que son vistos solamente dentro de la acción o la función. O sea, fuera de ellos nadie sabe de su existencia y su utilización provocaría un fallo.

Reglas de alcance

- La existencia de identificadores globales y locales, así como la presencia de acciones y funciones anidadas, hace necesario definir con claridad las reglas de alcance de cada identificador.
- El **alcance** de un identificador es aquella porción del algoritmo en que se conoce al identificador.
- Veamos esto en el ejemplo anterior.

```
Acción CalcSueldo(dato d1, d2 ∈ R, resultado bruto, neto ∈ R)

    Acción SignoMonetario(resultado signo ∈ TMoneda)
    ...

    Acción ObtenerSalBruto(dato signo1 ∈TMoneda,resultado bruto1 ∈R)
    Lexico Local
      bruaux ∈ R
      Función Convertir(dato b ∈ R, s ∈ TMoneda) → R
    ...

    Acción CalcNeto(dato d1, d2, bruto2 ∈ R,resultado neto1 ∈ R)
    Lexico Local
      tipoEmp ∈ 1..2
    ...

Inicio //CalcSueldo
...
Facción //CalcSueldo
```


Reglas de alcance

¿Puedo usar **tipoEmp** en la acción **CalcularSueldo**?

- La variable **tipoEmp** es local a la acción **CalcNeto** y por lo tanto su alcance se reduce a esta acción. O sea, **tipoEmp** no se conoce ni en la acción **CalcSueldo** ni en el algoritmo principal **CalcularSueldo**.

¿Puedo utilizar **bruaux** y **Convertir** en otra acción que no sea **ObtenerSalBruto**?

- No, porque son locales a la acción **ObtenerSalBruto** y por lo tanto son conocidas únicamente por esta acción.

Reglas de alcance

- Los parámetros formales de **CalcSueldo** juegan el papel de identificadores:
  - locales a la acción, y
  - globales a las acciones anidadas.
- Por ejemplo: el parámetro formal **bruto** (que es de salida) es conocido en la acción **CalcNeto**. Por lo tanto podría usarse sin necesidad de pasarlo como parámetro.

Reglas de alcance

- La situación de los parámetros formales **d1** y **d2** de **CalcSueldo** es diferente en lo que respecta a **CalcNeto**.
- Dado que esta acción usa esos mismos nombres para sus parámetros formales, los identificadores **d1** y **d2** de **CalcSueldo** no son visibles dentro de **CalcNeto**.

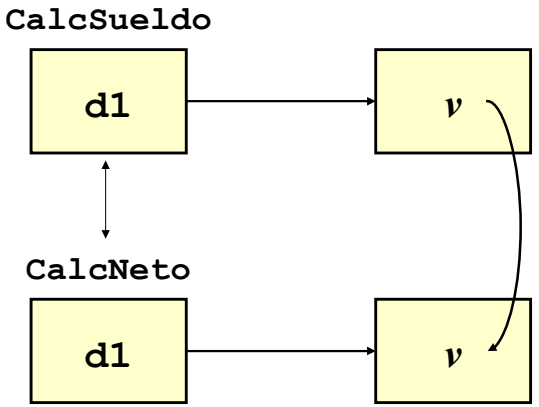
Reglas de alcance

- Por ejemplo, las ocurrencias de **d1** en **CalcNeto**, se refieren al parámetro formal **d1** del mismo y **no** a la variable global **d1** (parámetro formal de **CalcSueldo**).
- A pesar de tener nombres iguales, y que la invocación a **CalcNeto** en el algoritmo principal de **CalcSueldo** tiene a estos mismos nombres como argumentos, **d1** y **d2** en una y otra acción denotan variables **diferentes**.

Reglas de alcance

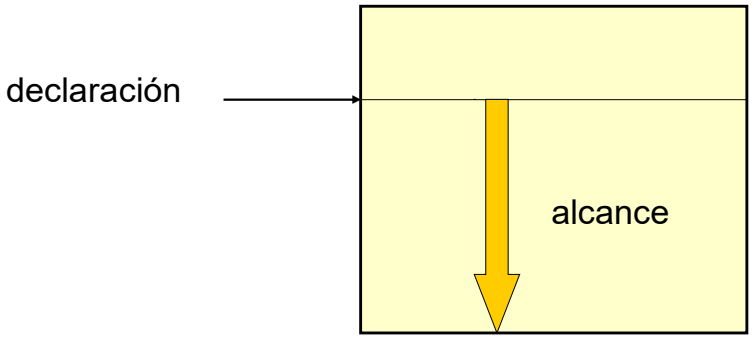
Llamada desde **CalcSueldo** a

CalcNeto(d1,d2,bruto,neto)



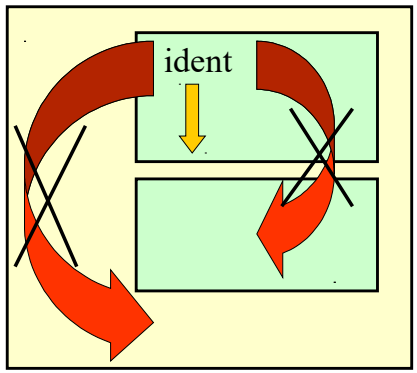
Reglas de alcance Síntesis

- El alcance de todo identificador es la porción de texto (declaraciones y sentencias) del algoritmo, acción o función que **sigue** a la declaración del identificador; y finaliza donde termina el algoritmo, acción o función que lo contenga.



Reglas de alcance Síntesis

- Esto significa que todo identificador es sólo conocido en el bloque en que es declarado (y en los bloques internos a este), pero **no** lo es ni en los bloques hermanos ni en ningún bloque externo.



Reglas de alcance Síntesis

- Los parámetros formales de una acción o función son tratados como variables locales del mismo. Por lo tanto, su alcance está acotado a se bloque.
- Los identificadores locales que tienen el mismo nombre que identificadores globales tienen prioridad sobre los globales. Por lo tanto, para hacer referencia a una variable global en tal condición, ésta debe ser pasada como parámetro.

En anidamiento no es muy utilizado porque dificulta el reuso.

Reglas de alcance en C

- Para variables, constantes, tipos, y parámetros son similares a las vistas hasta ahora.
- **Functions anidadas**: **C no permite anidar functions** (otros lenguajes como Pascal si lo permiten).
- Sin embargo, dentro de una ***function*** puede existir la llamada a una o más ***functions*** declaradas previamente.
- Además, en C está permitido hacer declaraciones en cualquier lugar, pero es una práctica que por el momento no utilizaremos, a los fines de ser ordenados en la traducción de los algoritmos a programas C.

Bibliografía

– Scholl, P. y Peyrin, J.-P. “Esquemas Algorítmicos Fundamentales: Secuencias e iteración”. (pags. 1 - 34)
– Biondi, J. y Clavel, G. “Introducción a la Programación. Tomo 1: Algorítmica y Lenguajes”. (pags. 1 – 12, 13 – 34, 127 - 140)
– Wirth, N. “Algoritmos + Estruturas de Datos = Programas”. (pags. 1 – 12).
– Watt, David: Programming Language Concepts and Paradigms, Prentice-Hall International Series in Computer Science (1990). Cap 4.
– Quetglás, Toledo, Cerverón. "Fundamentos de Informática y Programación“. Capítulo 3. http://robotica.uv.es/Libro/Indice.html
• Programación Modular (pags 110 – 125)