Principios de Lenguajes de Programación

Control de datos: Bloque y Alcance

Facultad de Informática
Universidad Nacional del Comahue

Primer Cuatrimestre

Índice

- Bloque
- Alcance
- Tiempo de vida
- Entorno de referencia

Bibliografia:

- Pratt cap. 7

Atributos del Control de Datos

- Forma del control de datos determinar la **accesibilidad de los datos** en diferentes puntos durante la ejecución del programa
- Problema Central:
  - El ***significado* de los *nombres (identificadores)***, la correspondencia entre los **nombres y los objetos** (de datos, de control, etc.) y **su locación** en ejecución

Nombres y Entorno de Referencia

- Acceso a Objetos de Datos:
  - un dato se hace disponible para un operador (como operando), dos formas:
    - Transmisión Directa
    - Referencia a través de un nombre (identificador)

Nombres y Entorno de Referencia

- Transmisión Directa
  - Un objeto de datos que se calcula en un punto como resultado de una operación, se "pasa" directamente como operador en otra operación
  - Ejemplo

```
x = y + 2*z;
```

    - El resultado del producto se transmite directamente como sumando a la adición

Nombres y Entorno de Referencia

- **Referencia por identificador/nombre de OD**
  - Un OD tiene (puede tener) un nombre cuando se crea y ese nombre (identificador) se utiliza para designarlo como operando en una expresión de operadores
- **Nombres (identificadores) en un programa**
  - Variables, parámetros formales, subprogramas
  - Tipos y Constantes predefinidos
  - Etiquetas de enunciados
  - Nombres de Excepciones
  - Operaciones primitivas
  - Constantes literales

Entorno de Referencia: asociaciones

- **Asociaciones:** ligaduras de identificadores a objetos de datos u objetos de control (subprogramas) en particular. Cada ligadura puede estar representada como un par: identificador y su asociación al OD o subprograma

  A:= B + FN(C)

- **Entorno de referencia:** conjunto de asociaciones para un entorno (subprograma dado)
- **Operaciones de Referencias:**
  - Durante la ejecución de los programas, determinar el objeto de dato particular (o subprograma) asociado con el identificador

  ref_op: *id* x *ent_ref* → *OD o subprograma*

Entornos de Referencia: clasificación

- Entorno Local:
  - Conjunto de asociaciones creadas en un subprograma (usualmente en su inicio/resolución)
    - Parámetros formales,
    - Variables locales
    - Subprogramas internos definidos (imbricados)
- Entorno No Local
  - El conjunto de asociaciones para identificadores utilizados dentro del programa pero que no fueron creados en él

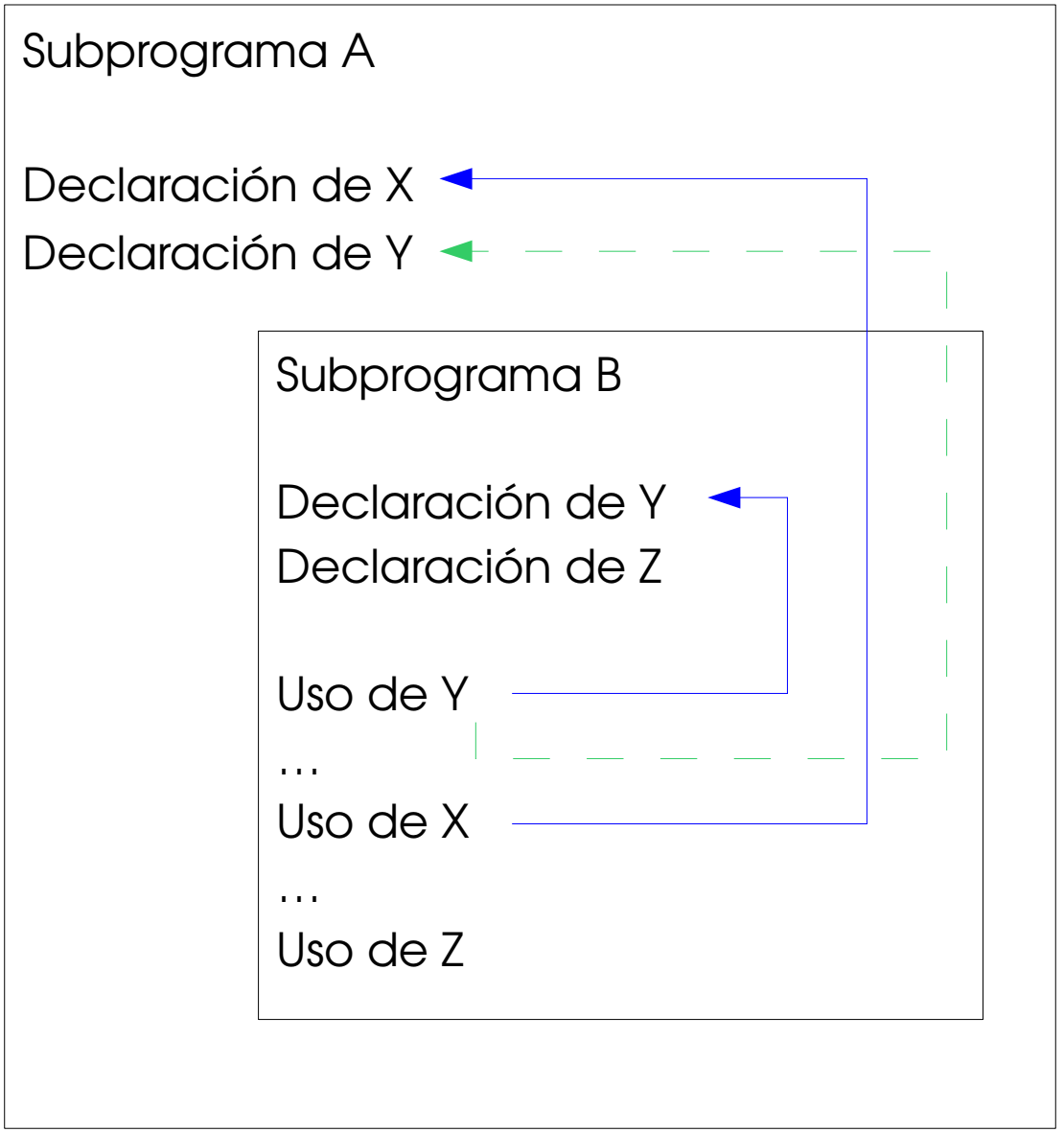
Entornos de Referencia: clasificación

- **Entorno No Local**
  - Entorno de Referencia Global:
    - Asociaciones creadas al inicio de la ejecución de un programa principal (a veces antes) disponible para ser utilizados por un subprograma
  - Entorno de Referencia Predefinido
    - Asociaciones predefinidas, usualmente ligadas en tiempo de definición/diseño del lenguaje

Entorno de Referencia: visibilidad

- **Visibilidad de las Asociaciones**:
  - Las asociaciones son visibles si forman parte de su entorno de referencia
  - Asociación existe pero no del entorno, son "ocultas"
- **Visibilidad**
  - Organización estática (textual - espacial)
  - Organización dinámica (temporal)
- **Aliasing**
  - Visibilidad con varios identificadores a un mismo objeto de datos (u otro)

Visibilidad: Ejemplo



Reglas de Alcance Estático para los programas estructurados en bloques

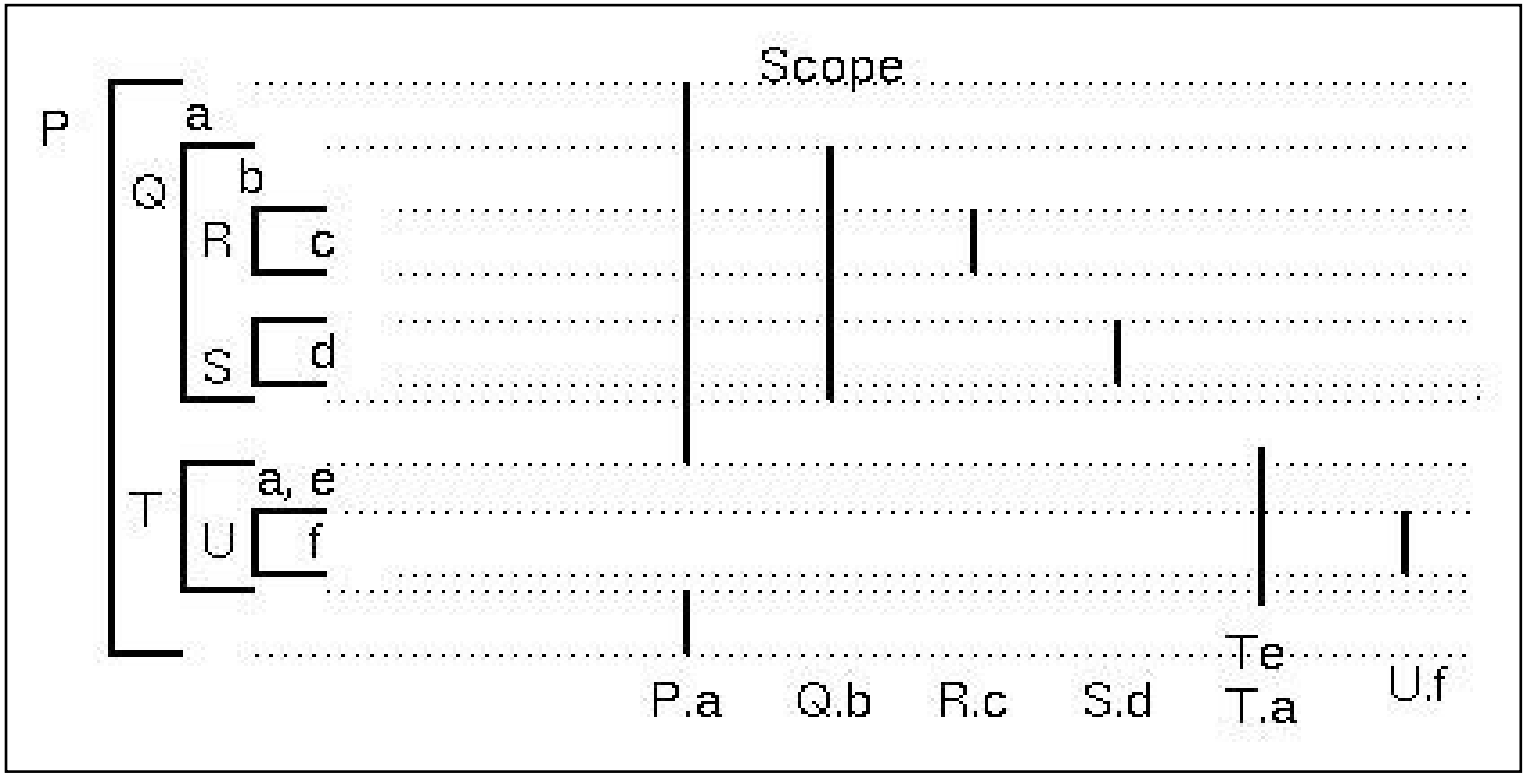
Atributos de Variables: Alcance

- El ***alcance de una declaración*** es la región del programa sobre el cual se establecen las ligaduras a partir de una declaración
- El ***alcance*** de una variable es el conjunto de instrucciones en los cuales es “visible”/”alcanzable”
- Una variable es ***visible*** en una instrucción si puede ser referenciada (nuevas declaraciones generan invisibilidad)
- Ejemplo
  - En una lenguaje estructurado en bloques, el alcance es el código entre la declaración y el final del bloque (por ejemplo {...} en C y Java).

Atributos de Variables: Alcance

- Las ***variables no locales*** a una unidad de programa son aquellas visibles pero no declaradas allí
- Las ***reglas de alcance*** del lenguaje determinan cómo se referencian los nombres asociados a las variables (o entidades)
- El ***alcance*** puede extenderse hacia atrás al comienzo de un bloque (por ejemplo las declaraciones de clases en Java y C++)

Alcance



- Q y T son declaraciones de procedimientos internos a P, por lo que el alcance de los nombres Q y T es el mismo que para a.
- R y S son declaraciones del procedimiento Q.
- U es una declaración de un procedimiento de T.

Alcance estático

- Basado en el código del programa, también llamado léxico, porque sigue las "jerarquías del código"
- Para "conectar" el nombre referenciando a una entidad (variable), debe "hallar" la declaración
- Proceso de búsqueda:
  - Buscar las declaraciones (primero las locales), luego en los alcances que contienen del actual, hasta que alguno tenga el identificador (nombre)
  - Los alcances estáticos que contienen (a un alcance específico) se denominan los ancestros estáticos
    - el ancestro estático más próximo es el padre estático

Alcance estático

- Algunos lenguajes permiten definiciones de subprogramas anidadas, que crean alcances estáticos anidados/imbricados
  - pe. Ada, JavaScript, Fortran 2003, PHP, etc.
- Los nombres pueden "esconderse" en una unidad de programa al tener una entidad (variable) más cercana con el mismo nombre
  - Acceso a estas variables "escondidas" / invisibles
    - Ada, *unidad.nombre*
    - C++, con operador de ámbito/alcance (::)

Acceso a variables escondidas: C++

```
int x; // Variable global


int main()
{
   int x;      // Variable local que "enmascara"
               // a la variable global

   x = 10;     // Accedemos a la variable local
   ::x = 100; // Mediante el operador de alcance
               // accedemos a la global
   return 0;
```

Bloques

- Un método para crear alcances estáticos dentro de las unidades de programas (Algol 60 en adelante)
- Por ejemplo:

```
void sub() {
    int count;
    while (...) {
        int count;
        count++;
        ...
    }
    …
}
```

Declaraciones: Orden

- C99, C++, Java, y C#
  - Permiten la declaración de una variable como una sentencia en cualquier lugar del programa
- En C99, C++, y Java, el *alcance* de todas las variables locales es desde la *declaración* hasta fin de bloque
- En C#, el *alcance* de todas las variables es el bloque en el que está declarado, no importa el lugar de la declaración
  - Sin embargo, la variable debe declararse antes de poder ser usada

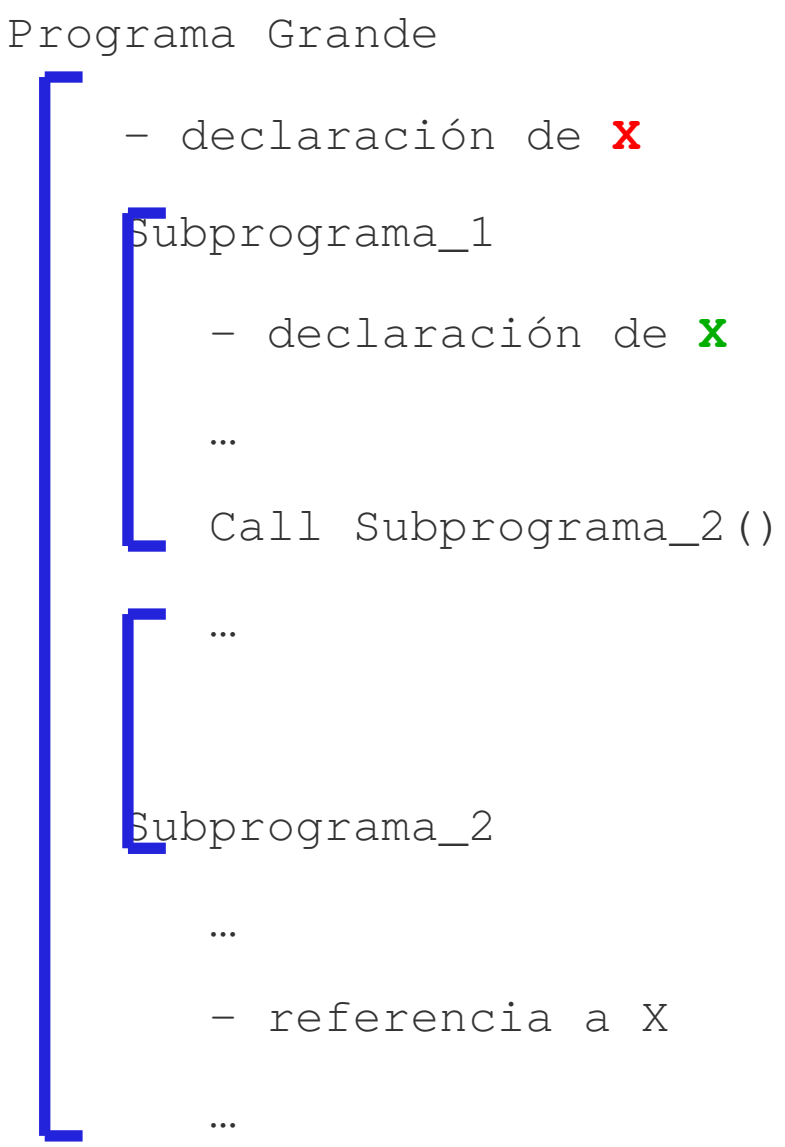
Alcance Global

- C, C++, PHP, y Python permiten una estructura de programa que defina una secuencia de funciones en un archivo
  - Permiten declaración de variables fuera de las definiciones de las funciones
- C y C++ permiten las declaraciones (atributos) y definiciones (atributos y almacenamiento)
  - Una declaración fuera de una función especifica que se define en otro archivo
  - *auto, register, static, extern*

Alcance Dinámico

- Basado en la secuencia de llamadas de los subprogramas en una programa, no a su estructura de texto
  - Temporal vs espacial
- Referencias a las variables:
  - Se conectan a las declaraciones buscando hacia atrás en la cadena de llamadas a los subprogramas en ejecución hasta ese punto

Alcance: ejemplo



- Secuencia de llamadas:
  - Grande llama a Subprograma_1
  - Subprograma_1 llama a Subprograma_2
  - Subprograma_2 referencia a X
- Grande → Sub1 → Sub2

Alcance: ejemplo

- Alcance Estático:
  - Referencia de **X es de Programa Grande**
- Alcance Dinámico
  - Referencia de **X es de Subprograma 1**

Reglas de Alcance: Implementación

- Reglas de Alcance Estático
  - Por medio de una tabla de las declaraciones locales
- Reglas de Alcance Dinámico
  - Retención: asociaciones y valores ligados se retienen después de la ejecución
  - Borrado: asociaciones se borran luego de la ejecución (se pierden los valores)

Reglas de Alcance: Implementación

```
program main;
var A, B, C: real;
procedure Sub1(A: real);
    var D: real;
    procedure Sub2 (C:real);
        var D: real;
        begin
        – Statements
        C := C+B;
        – Statements
        end;
    begin
    – Statements
    Sub2(B);
    – Statements
    end;
begin
– Statements
Sub1(A);
– Statements
end.
```

Referencing environment for Sub2
Local C, D
Nonlocal A, Sub2 in Sub1
B, Sub1 in main

Referencing environment for Sub1
Local A, D, Sub2
Nonlocal B, C, Sub1 in main

Referencing environment for main
Local: A, B, C, Sub1

Figure 7.3. Referencing environments in a Pascal program

Evaluación de Alcance Dinámico

- **Ventajas**:
  - Conveniencia
  - Relacionado con las asociaciones de los identificadores
- **Desventajas**:
  - Mientras un subprograma está en ejecución, se hacen visibles todas sus variables a los subprogramas llamados
  - Imposible realizar chequeo/control estático
  - No es posible determinar estáticamente el tipo de una variable

Alcance vs. Tiempo de Vida

- Alcance: visibilidad
- Tiempo de Vida: ligaduras en vigencia

Entorno de Referencia

- Para una sentencia es el Conjunto de nombres visibles en esa sentencia
- En los lenguajes de alcance estático:
  - Variables locales + variables no locales visibles para cada subprograma englobador
- En los lenguajes de alcance dinámico:
  - Variables locales + variables no locales visibles para cada subprograma activo
  - Un subprograma está activo si su ejecución ha comenzado y no terminado

Ejemplificación

```
program Principal;
 var a, b;
 function F1;
  var b, d;
  begin
    b := 1; a := b + 3;
    d := a + b;
    write (a, b, d);
    F2;
    return ( a + b );
  end;
```

```
procedure F2;
 var a;
 begin
   a := b + 5; b := 3;
   write (a, b);
 End;


begin
  a:= 1; b:= 2;
  write ( F1 );
end.
```

Ejemplificación

Programa	Eltos Locales	Eltos no Locales	
		Visibles	No Visibles

Tarea

- Analizar ejemplo:

  Para el siguiente programa, indicar todas las componentes de los ambientes de referencia locales y no locales para **Main**, **P**, **Q** y **R** y las salidas respectivas, justificando oportunamente, para el programa escrito en un lenguaje con reglas:

  - de alcance estático
  - de alcance dinámico

Ejemplificación

```
program Main;

var a, b, c;

    function Q;

    var b, u, x;

    begin

        b:= 1; u:= 1;

        a := 2*b + u;

        x := c + a;

        write (a, b, x);

        return ( a + b );

    end;
```

```
    procedure R;

    var z, y, a;

       procedure P;

       var x, y;

        begin

           write ( Q + b );

        end;

     Begin  (* R *)

       a := b + 5;

       y := c + a;

       write (a, b, y, c);

       P;

     end;

Begin (* Main *)

  a:= 1; b:= 2; c:= 3;

  write ( Q );

  R;

end.
```