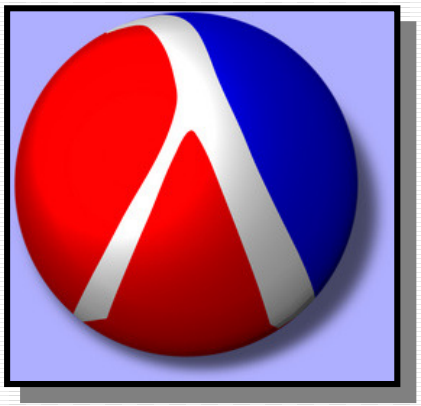
Fundamentos de Programación

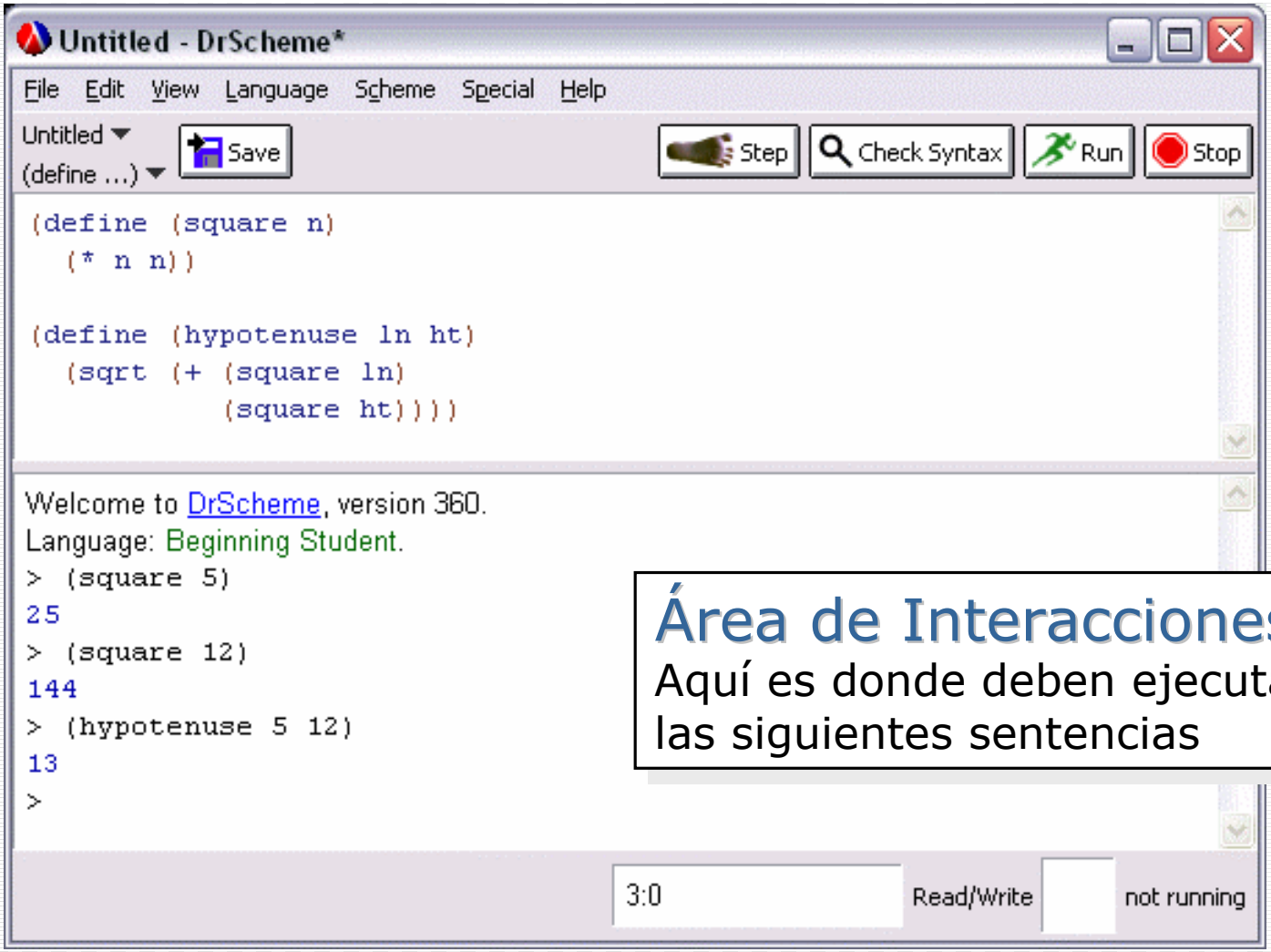
Construcciones fundamentales en Scheme



Profesor: Daniel Wilches Maradei
Diapositivas Originales: Jesús A. Aranda

Universidad del Valle

Explicación previa



Área de Interacciones

Aquí es donde deben ejecutarse las siguientes sentencias

Operaciones aritméticas

- (+ 2 2) ; Devolvería 4
  En Scheme las operaciones son prefijas, es decir, que el operador se encuentra antes de todos sus operandos

- En Scheme todas las operaciones van encerradas entre paréntesis. Esto hace que la sintaxis sea más sencilla para el intérprete/compilador, no hay lugar a ambigüedades.

Operaciones aritméticas
Más ejemplos

- (/ 2 7) ;División 2/7
- (* 3 4) ;Multiplicación 3x4
- (+ 7 (/ 20 5)) ;7+20/5
- (+ (/ 2 5) (/ 1 5) (/ 2 5))
  ;2/5 + 1/5 + 2/5
- (/ 5 0) ;Error, división por 0

Acerca de la ambigüedad

- La ambigüedad de la que hablábamos antes es la del orden de evaluación de las primitivas (* / + -)
- Por ejemplo:

  3 + 4 * 5
- Para solucionar acordamos un orden de precedencia de los operadores. En Scheme, debido a que los paréntesis son obligatorios, esta ambigüedad no se da:

  (+ 3 (* 4 5))

Ejercicio en clase

- Escriba en notación prefija (la usada por Scheme) las siguientes operaciones (no se vale simplificar !!):
  - 5/10 + 8/9 – 14/5
  - (7 + 5) / 6 - (9 * 5) - 6
  - $3^2 + 4^2$
  - tan (90)

Algunos elementos del lenguaje

- Números enteros:

  4 −6 0

- Números reales:

  3.1415 2.72

- Fraccionarios:

  1/2 4/3

- Símbolos:

  var1 sueldo

Ejemplo de operaciones

- La función sqrt retorna la raíz cuadrada de un número:

  sqrt(4) => (sqrt 4)

- Si escribimos ese ejemplo en el área de interacción de Scheme nos devolvería 2.
- Pero si ejecutamos una operación cuyo resultado no sea entero:

  (sqrt (sqrt 4)) => #i1.4142135623730951

Ejemplo de operaciones

- El valor devuelto es el valor correcto de la evaluación de (sqrt(sqrt 4)). El prefijo #i significa que el número mostrado es aproximado.

- La notación #i es necesaria ya que hay operaciones cuyo resultado no puede ser representado de manera finita

Más operaciones

(sqrt A)	Raíz cuadrada de A
(expt A B)	A elevado a la B
(remainder A B)	Residuo de A/B
(log A)	Logaritmo natural de A
(sin A)	Seno de A

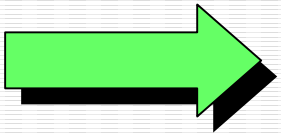
Creación de funciones

- Las funciones anteriormente vistas (sqrt, expt, ...) están predefinidas en el lenguaje Scheme. Pero este nos da la posibilidad de crear nuestras propias funciones
- Para crear funciones hacemos uso de la palabra reservada define

Creación de funciones

- Por ejemplo, creemos una función que calcule el cuadrado de un número, y llamémosla cuadrado

```
(define (cuadrado x)
        (* x x))
```

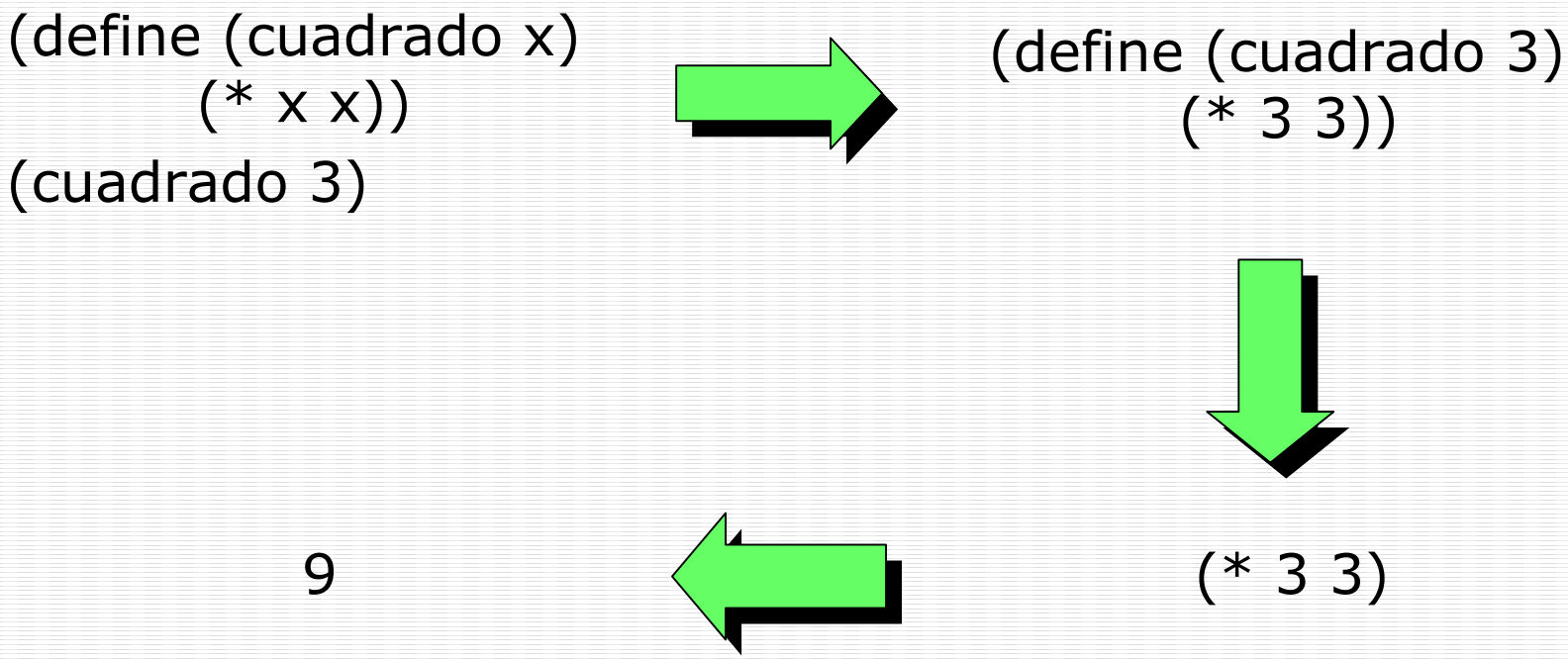


$cuadrado(x) = x^2$

Creación de funciones

- Cuando invoquemos (cuadrado 3) Scheme entenderá llamará a nuestra función cuadrado, es decir, ejecutará el cuerpo de esta función reemplazando x por 3:

Creación de funciones



Creación de funciones

- Una de las ventajas de Scheme es que nos permite trabajar con números muy, muy grandes:

  (cuadrado 863485395619280478866760904715026010137 6)

  Retornaría:

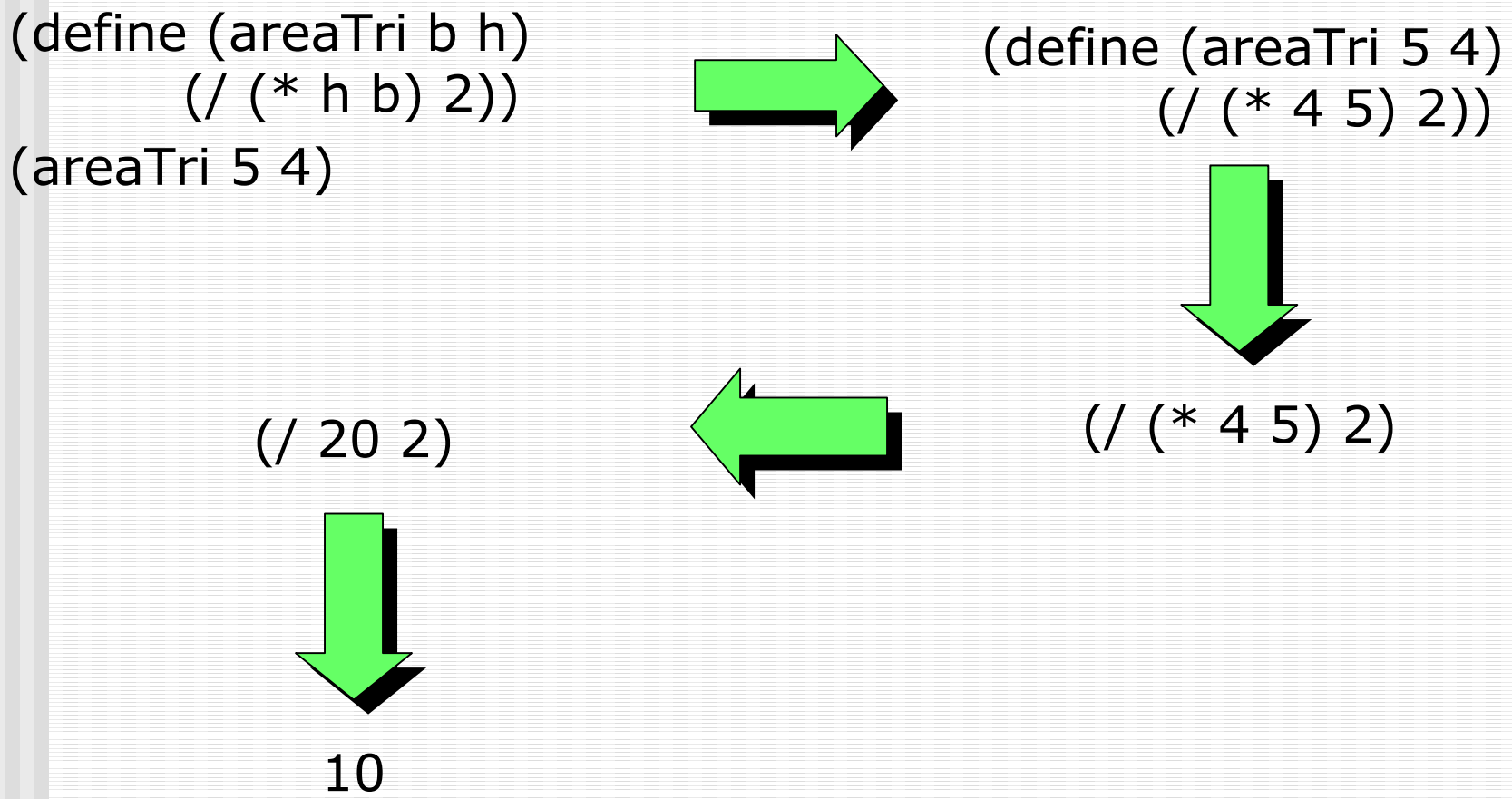
  74560702844778532320361669586226789311642106932730194591762087595409525797093376

- Con la palabra define podemos también definir variables:

  (define PI 3.1415)

  (define nombre "juanita")

Funciones más complejas



Usando el Stepper

- DrScheme incluye una herramienta que presenta paso a paso la evaluación de una expresión
- El Stepper puede mostrarnos el orden de evaluación de la siguiente expresión por ejemplo:

```
(/ (* 2 4) (/ (* 6 4) (* 12 4)))
```

Ejercicio

- Cómo escribirían una función que calcule la longitud de la hipotenusa de un triángulo rectángulo (usando el Teorema de Pitágoras) ?

$$c^2 = a^2 + b^2$$

$$c = \text{sqrt}(a^2 + b^2)$$

Solución al ejercicio

```
(define (hipotenusa ladoA ladoB)
        (sqrt (+ (* ladoA ladoA)
                 (* ladoB ladoB))))
```

O usando la función "cuadrado" definida anteriormente:

```
(define (hipotenusa ladoA ladoB)
        (sqrt (+ (cuadrado ladoA)
                 (cuadrado ladoB))))
```

Estilo en la programación

- Cuando realicen un programa es importante mantenerlo “bonito”, es decir, bien indentado, bien comentado, usar nombre explicativos y adecuados para funciones y variables
- Esto hace que su código sea más fácil de leer y mantener. El código fuente es leído muchas más veces de la que es escrito.

Definición de variables

- Las variables, al igual que las funciones, se definen usando la construcción (define ....)
- La sintaxis varía un poco:

```
(define RADIO 10)
(define RADIO2 (cuadrado 5))
```

Tipos de errores en Scheme

- Errores de Sintaxis

  Variables no definidas

  Construcción incorrecta

  ```
  ((+ 3 4) * (/ 8 9))
  ```

- Errores en tiempo de ejecución

  Dividir por cero

  Lamar a una función con los parámetros incorrectos

  ```
  (/ 5 0)
  ```

- Errores lógicos

  Nos equivocamos al decirle al computador lo que queremos que haga

  ```
  (define (cuadrado x)
  (+ x x))
  ```

Scheme es interpretado

- El Scheme, tal como lo usaremos, será un lenguaje interpretado. Es decir, no crearemos ejecutables a partir de nuestro código. Aunque si hay manera de hacerlo.

Compilados	Híbridos	Interpretados
C++	Java	PLSQL
Pascal	C#	PHP

Tipos en Scheme

- En Scheme no se le asignan tipos a las variables en cuanto se declaran, solo cuando se usan:

```
(define NOMBRE "Scheme")
```

- En Scheme los errores de tipos son errores en tiempo de ejecución:

```
(+ NOMBRE 2)
```