Paradigmas Fundamentales de Programación

Procedimientos y Registros como valores básicos

Juan Francisco Díaz Frias

Maestría en Ingeniería, Énfasis en Ingeniería de Sistemas y Computación
Escuela de Ingeniería de Sistemas y Computación,
home page: http://eisc.univalle.edu.co
Universidad del Valle - Cali, Colombia

Modelos y Paradigmas de Programación

Universidad del Valle

Plan

Plan

Universidad del Valle

Plan

Plan

Lenguaje núcleo: sintaxis de las declaraciones ⟨d⟩

- Lenguaje núcleo sencillo: todos los programas en el modelo se pueden expresar en este lenguaje.
- Todas las declaraciones del lenguaje núcleo son declaraciones válidas del lenguaje completo.

⟨d⟩ ::=	
skip	Declaración vacía
⟨d⟩$_1$ ⟨d⟩$_2$	Declaración de secuencia
local ⟨x⟩ **in** ⟨d⟩ **end**	Creación de variable
⟨x⟩$_1$=⟨x⟩$_2$	Ligadura variable-variable
⟨x⟩=⟨v⟩	Creación de valor
if ⟨x⟩ **then** ⟨d⟩$_1$ **else** ⟨d⟩$_2$ **end**	Condicional
case ⟨x⟩ **of** ⟨patrón⟩ **then** ⟨d⟩$_1$ **else** ⟨d⟩$_2$ **end**	Reconocimiento de patrones
{ ⟨x⟩ ⟨y⟩$_1$ ... ⟨y⟩$_n$ }	Invocación de procedimiento

Plan

Lenguaje núcleo: sintaxis de los valores ⟨v⟩

⟨v⟩	::=	⟨número⟩ | ⟨registro⟩ | ⟨procedimiento⟩
⟨número⟩	::=	⟨ent⟩ | ⟨flot⟩
⟨registro⟩, ⟨patrón⟩	::=	⟨literal⟩
	|	⟨literal⟩(⟨campo⟩$_1$: ⟨x⟩$_1$ $\cdots$ ⟨campo⟩$_n$: ⟨x⟩$_n$)
⟨procedimiento⟩	::=	**proc** { $ ⟨x⟩$_1$ $\cdots$ ⟨x⟩$_n$ } ⟨d⟩ **end**
⟨literal⟩	::=	⟨átomo⟩ | ⟨bool⟩
⟨campo⟩	::=	⟨átomo⟩ | ⟨bool⟩ | ⟨ent⟩
⟨bool⟩	::=	**true** | **false**

Note que: en los registros y patrones, todos los argumentos ⟨x⟩$_1$, ..., ⟨x⟩$_n$ deben ser identificadores diferentes.

Plan

Lenguaje núcleo:Sintaxis de identificadores de variables

- ⟨x⟩ y ⟨y⟩ denotan identificadores de variable.
- Identificador de variable en Oz:
  - Una letra mayúscula seguida de cero o más caracteres alfanuméricos: `X`, `X1`, `Variable_Larga`.
  - Secuencia de caracteres entre ‘: `&esta es una 25\$\variable!&`.
- Todas las variables recién declaradas inician no-ligadas.
- Todos los identificadores de variables deben ser declarados explícitamente.

Lenguaje núcleo:Sintaxis de identificadores de variables

- ⟨x⟩ y ⟨y⟩ denotan identificadores de variable.
- Identificador de variable en Oz:
  - Una letra mayúscula seguida de cero o más caracteres alfanuméricos: `X`, `X1`, `Variable_Larga`.
  - Secuencia de caracteres entre ‘: `&esta es una 25\$\variable!&`.
- Todas las variables recién declaradas inician no-ligadas.
- Todos los identificadores de variables deben ser declarados explícitamente.

Lenguaje núcleo:Sintaxis de identificadores de variables

- ⟨x⟩ y ⟨y⟩ denotan identificadores de variable.
- Identificador de variable en Oz:
  - Una letra mayúscula seguida de cero o más caracteres alfanuméricos: `X`, `X1`, `Variable_Larga`.
  - Secuencia de caracteres entre ‘: `&esta es una 25\$\variable!&`.
- Todas las variables recién declaradas inician no-ligadas.
- Todos los identificadores de variables deben ser declarados explícitamente.

Lenguaje núcleo:Sintaxis de identificadores de variables

- ⟨x⟩ y ⟨y⟩ denotan identificadores de variable.
- Identificador de variable en Oz:
  - Una letra mayúscula seguida de cero o más caracteres alfanuméricos: `X`, `X1`, `Variable_Larga`.
  - Secuencia de caracteres entre ‘: `&esta es una 25\$\variable!&`.
- Todas las variables recién declaradas inician no-ligadas.
- Todos los identificadores de variables deben ser declarados explícitamente.

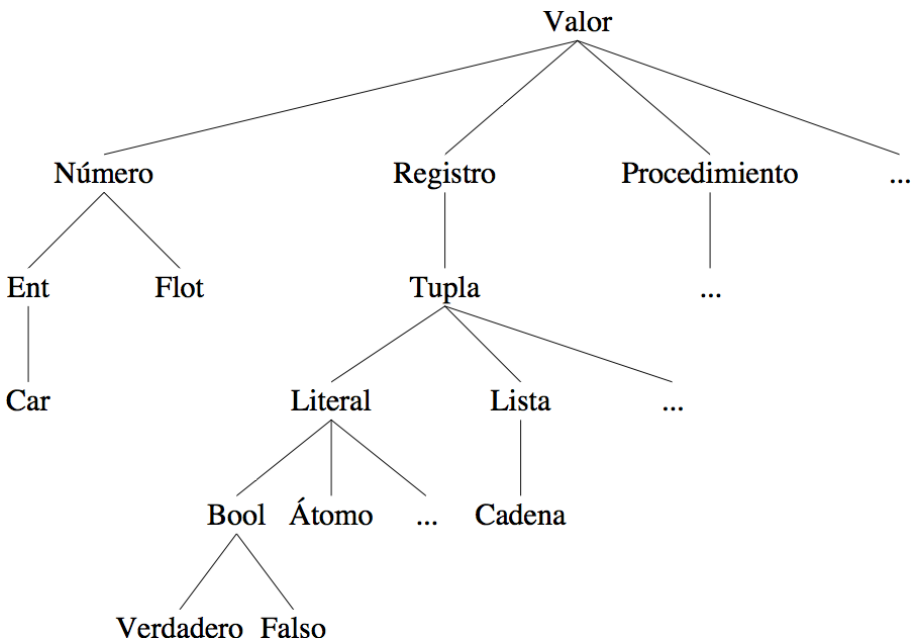
Lenguaje núcleo: Valores y tipos

- Tipo de datos = conjunto de valores + conjunto de operaciones.
- El modelo declarativo es tipado: cuenta con un conjunto de tipos básicos.
- Intentar usar una operación con valores del tipo errado es detectada por el sistema.
- Los programas pueden definir sus propios tipos: TAD.

Lenguaje núcleo: Tipos básicos

Jerarquía de tipos

Los tipos básicos del modelo declarativo son los números (enteros y flotantes), los registros (incuyendo átomos, booleanos, tuplas, listas, y cadenas), y los procedimientos.



Comentarios

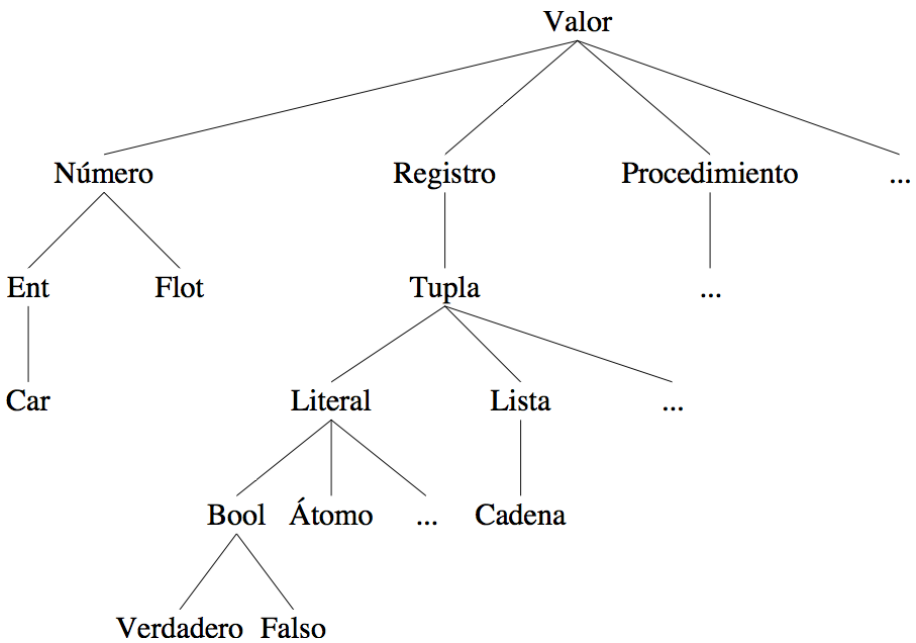
- Tipamiento dinámico: el tipo de una variable se conoce solo en el momento en que la variable se liga.
- La jerarquía está ordenada por inclusión de conjuntos.
- Las tuplas son registros y las listas son tuplas.

Universidad del Valle

Lenguaje núcleo: Tipos básicos

Jerarquía de tipos

Los tipos básicos del modelo declarativo son los números (enteros y flotantes), los registros (incuyendo átomos, booleanos, tuplas, listas, y cadenas), y los procedimientos.



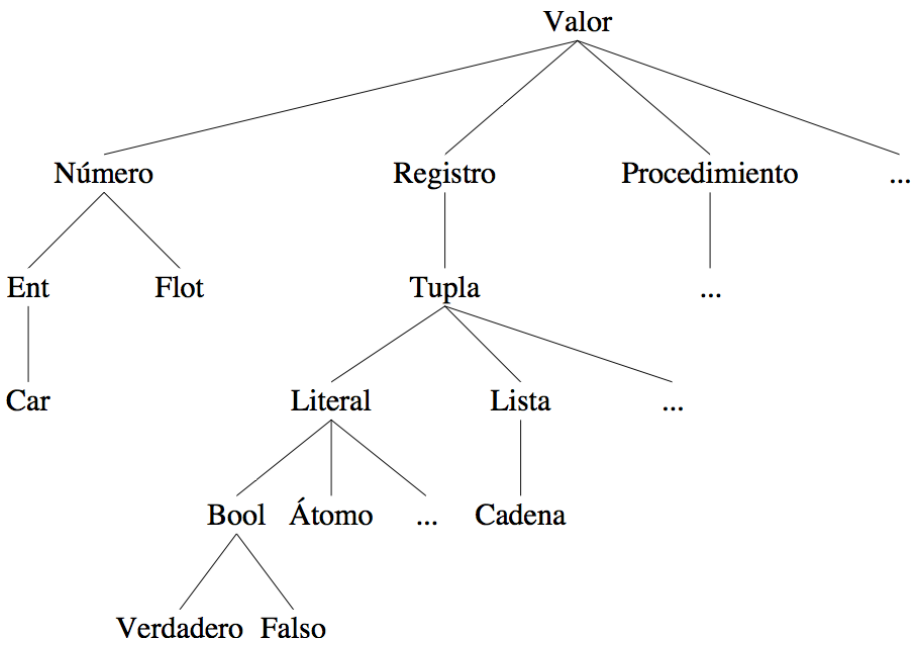
Comentarios

- Tipamiento dinámico: el tipo de una variable se conoce solo en el momento en que la variable se liga.
- La jerarquía está ordenada por inclusión de conjuntos.
- Las tuplas son registros y las listas son tuplas.

Lenguaje núcleo: Tipos básicos

Jerarquía de tipos

Los tipos básicos del modelo declarativo son los números (enteros y flotantes), los registros (incuyendo átomos, booleanos, tuplas, listas, y cadenas), y los procedimientos.



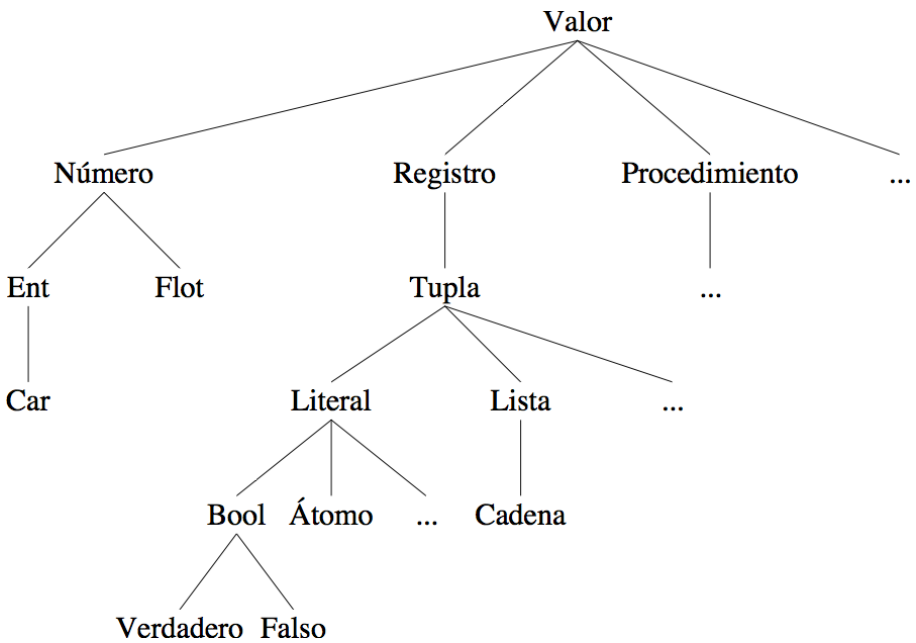
Comentarios

- Tipamiento dinámico: el tipo de una variable se conoce solo en el momento en que la variable se liga.
- La jerarquía está ordenada por inclusión de conjuntos.
- Las tuplas son registros y las listas son tuplas.

Lenguaje núcleo: Tipos básicos

Jerarquía de tipos

Los tipos básicos del modelo declarativo son los números (enteros y flotantes), los registros (incuyendo átomos, booleanos, tuplas, listas, y cadenas), y los procedimientos.



Comentarios

- Tipamiento dinámico: el tipo de una variable se conoce solo en el momento en que la variable se liga.
- La jerarquía está ordenada por inclusión de conjuntos.
- Las tuplas son registros y las listas son tuplas.

Plan

Lenguaje núcleo: sintaxis para construir valores

⟨v⟩	::=	⟨número⟩ | ⟨registro⟩ | ⟨procedimiento⟩
⟨número⟩	::=	⟨ent⟩ | ⟨flot⟩
⟨registro⟩, ⟨patrón⟩	::=	⟨literal⟩
	|	⟨literal⟩(⟨campo⟩$_1$: ⟨x⟩$_1$ $\cdots$ ⟨campo⟩$_n$: ⟨x⟩$_n$)
⟨procedimiento⟩	::=	**proc** { $ ⟨x⟩$_1$ $\cdots$ ⟨x⟩$_n$ } ⟨d⟩ **end**
⟨literal⟩	::=	⟨átomo⟩ | ⟨bool⟩
⟨campo⟩	::=	⟨átomo⟩ | ⟨bool⟩ | ⟨ent⟩
⟨bool⟩	::=	**true** | **false**

Plan

Tipos básicos: Números, átomos y booleanos

Números

- Enteros o de punto flotante.
- Ejemplos de enteros: `314`, `0`, y `~10`
- Ejemplos de punto flotante: `1.0`, `3.4`, `2.0e2`,y `~2.0E~2`.

Booleanos

Átomos `true` y `false`

Átomos

- Átomo:constante simbólica que puede utilizarse como un elemento indivisible (atómico) en los cálculos.
- Secuencia de caracteres iniciando con una letra minúscula y seguido de cualquier número de caracteres alfanuméricos: `una_persona`, `donkeyKong3`.
- Secuencia de caracteres imprimibles encerrados entre comillas sencillas: `'#### hola ####'`.

Tipos básicos: Números, átomos y booleanos

Números

- Enteros o de punto flotante.
- Ejemplos de enteros: `314`, `0`, y `~10`
- Ejemplos de punto flotante: `1.0`, `3.4`, `2.0e2`,y `~2.0E~2`.

Booleanos

Átomos `true` y `false`

Átomos

- Átomo:constante simbólica que puede utilizarse como un elemento indivisible (atómico) en los cálculos.
- Secuencia de caracteres iniciando con una letra minúscula y seguido de cualquier número de caracteres alfanuméricos: `una_persona`, `donkeyKong3`.
- Secuencia de caracteres imprimibles encerrados entre comillas sencillas: `'#### hola ####'`.

Universidad del Valle

Tipos básicos: Números, átomos y booleanos

Números

- Enteros o de punto flotante.
- Ejemplos de enteros: `314`, `0`, y `~10`
- Ejemplos de punto flotante: `1.0`, `3.4`, `2.0e2`,y `~2.0E~2`.

Booleanos

Átomos **`true`** y **`false`**

Átomos

- Átomo:constante simbólica que puede utilizarse como un elemento indivisible (atómico) en los cálculos.
- Secuencia de caracteres iniciando con una letra minúscula y seguido de cualquier número de caracteres alfanuméricos: `una_persona`, `donkeyKong3`.
- Secuencia de caracteres imprimibles encerrados entre comillas sencillas: `'#### hola ####'`.

Plan

Tipos básicos: Registros y tuplas

Registros

- Estructura de datos compuesta:
  - $\langle l \rangle(\langle c \rangle_1 : \langle x \rangle_1 \cdots \langle c \rangle_n : \langle x \rangle_n)$
  - $\langle l \rangle$ es una etiqueta.
  - Los campos pueden ser átomos, enteros, o booleanos.
- Ejemplos:
  - `persona(edad:X1 nombre:X2)`
  - `persona(1:X1 2:X2)`,
  - `'|'(1:H 2:T)`,
  - `'#'(1:H 2:T)`,
  - `nil`, y `persona`.
- Un átomo es un registro sin campos.

Tuplas (Azúcar)

- Una tupla es un registro cuyos campos son enteros consecutivos, comenzando desde el 1.
- `persona(1:X1 2:X2)` y `persona(X1 X2)`

Listas (Azúcar)

- `nil` o `'|'(H T)`
- `H|T` ≡ `'|'(H T)`
- `1|2|3|nil` ≡ `1|(2|(3|nil))`.
- `[1 2 3]` ≡ `1|2|3|nil`.

Tipos básicos: Registros y tuplas

Registros

- Estructura de datos compuesta:
  - $\langle l \rangle(\langle c \rangle_1 : \langle x \rangle_1 \cdots \langle c \rangle_n : \langle x \rangle_n)$
  - $\langle l \rangle$ es una etiqueta.
  - Los campos pueden ser átomos, enteros, o booleanos.
- Ejemplos:
  - `persona(edad:X1 nombre:X2)`
  - `persona(1:X1 2:X2)`,
  - `'|'(1:H 2:T)`,
  - `'#'(1:H 2:T)`,
  - `nil`, y `persona`.
- Un átomo es un registro sin campos.

Tuplas (Azúcar)

- Una tupla es un registro cuyos campos son enteros consecutivos, comenzando desde el 1.
- `persona(1:X1 2:X2)` y `persona(X1 X2)`

Listas (Azúcar)

- `nil` `'|'(H T)`
- `H|T` == `'|'(H T)`
- `1|2|3|nil` == `1|(2|(3|nil))`.
- `[1 2 3]` == `1|2|3|nil`.

Tipos básicos: Registros y tuplas

Registros

- Estructura de datos compuesta:
  - $\langle l \rangle(\langle c \rangle_1 : \langle x \rangle_1 \cdots \langle c \rangle_n : \langle x \rangle_n)$
  - $\langle l \rangle$ es una etiqueta.
  - Los campos pueden ser átomos, enteros, o booleanos.
- Ejemplos:
  - `persona(edad:X1 nombre:X2)`
  - `persona(1:X1 2:X2)`,
  - `'|'(1:H 2:T)`,
  - `'#'(1:H 2:T)`,
  - `nil`, y `persona`.
- Un átomo es un registro sin campos.

Tuplas (Azúcar)

- Una tupla es un registro cuyos campos son enteros consecutivos, comenzando desde el 1.
- `persona(1:X1 2:X2)` y `persona(X1 X2)`

Listas (Azúcar)

- `nil o '|'(H T)`
- `H|T ≡ '|'(H T)`
- `1|2|3|nil ≡ 1|(2|(3|nil))`.
- `[1 2 3] ≡ 1|2|3|nil`.

Tipos básicos: Registros y tuplas

Registros

- Estructura de datos compuesta:
  - $\langle l \rangle(\langle c \rangle_1 : \langle x \rangle_1 \cdots \langle c \rangle_n : \langle x \rangle_n)$
  - $\langle l \rangle$ es una etiqueta.
  - Los campos pueden ser átomos, enteros, o booleanos.
- Ejemplos:
  - `persona(edad:X1 nombre:X2)`
  - `persona(1:X1 2:X2)`,
  - `'|'(1:H 2:T)`,
  - `'#'(1:H 2:T)`,
  - `nil`, y `persona`.
- Un átomo es un registro sin campos.

Tuplas (Azúcar)

- Una tupla es un registro cuyos campos son enteros consecutivos, comenzando desde el 1.
- `persona(1:X1 2:X2)` y `persona(X1 X2)`

Listas (Azúcar)

- `nil` o `'|'(H T)`
- `H|T` ≡ `'|'(H T)`
- `1|2|3|nil` ≡ `1|(2|(3|nil))`.
- `[1 2 3]` ≡ `1|2|3|nil`.
- cadena: lista de cod. de caracs. `"E=mc^2"` ≡ `[69 61 109 99 94 50]`

Tipos básicos: Registros y tuplas

Registros

- Estructura de datos compuesta:
  - $\langle l \rangle(\langle c \rangle_1 : \langle x \rangle_1 \cdots \langle c \rangle_n : \langle x \rangle_n)$
  - $\langle l \rangle$ es una etiqueta.
  - Los campos pueden ser átomos, enteros, o booleanos.
- Ejemplos:
  - `persona(edad:X1 nombre:X2)`
  - `persona(1:X1 2:X2)`,
  - `'|'(1:H 2:T)`,
  - `'#'(1:H 2:T)`,
  - `nil`, y `persona`.
- Un átomo es un registro sin campos.

Tuplas (Azúcar)

- Una tupla es un registro cuyos campos son enteros consecutivos, comenzando desde el 1.
- `persona(1:X1 2:X2)` y `persona(X1 X2)`

Listas (Azúcar)

- `nil` o `'|'(H T)`
- `H|T` ≡ `'|'(H T)`
- `1|2|3|nil` ≡ `1|(2|(3|nil))`.
- `[1 2 3]` ≡ `1|2|3|nil`.
- cadena: lista de cod. de caracs. `"E=mc^2"` ≡ `[69 61 109 99 94 50]`.

Tipos básicos: Registros y tuplas

Registros

- Estructura de datos compuesta:
  - $\langle l \rangle(\langle c \rangle_1 : \langle x \rangle_1 \cdots \langle c \rangle_n : \langle x \rangle_n)$
  - $\langle l \rangle$ es una etiqueta.
  - Los campos pueden ser átomos, enteros, o booleanos.
- Ejemplos:
  - `persona(edad:X1 nombre:X2)`
  - `persona(1:X1 2:X2)`,
  - `'|'(1:H 2:T)`,
  - `'#'(1:H 2:T)`,
  - `nil`, y `persona`.
- Un átomo es un registro sin campos.

Tuplas (Azúcar)

- Una tupla es un registro cuyos campos son enteros consecutivos, comenzando desde el 1.
- `persona(1:X1 2:X2)` y `persona(X1 X2)`

Listas (Azúcar)

- `nil` o `'|'(H T)`
- `H|T` ≡ `'|'(H T)`
- `1|2|3|nil` ≡ `1|(2|(3|nil))`.
- `[1 2 3]` ≡ `1|2|3|nil`.
- cadena: lista de cod. de caracs. `"E=mc^2"` ≡ `[69 61 109 99 94 50]`.

Plan

Tipos básicos: procedimientos

- Un procedimiento es un valor de tipo procedimiento.
- `proc {` $\$ \langle y \rangle_1 \cdots \langle y \rangle_n$ `}` $\langle d \rangle$ `end` crea un valor nuevo de tipo procedimiento.
- La declaración

  `proc {` $\langle x \rangle \langle y \rangle_1 \cdots \langle y \rangle_n$ `}` $\langle d \rangle$ `end`

  es lo mismo que

  $\langle x \rangle$ = `proc {` $\$ \langle y \rangle_1 \cdots \langle y \rangle_n$ `}` $\langle d \rangle$ `end`

  Sin embargo, oculta la distinción entre crear un valor y ligarlo a un identificador.

Tipos básicos: procedimientos

- Un procedimiento es un valor de tipo procedimiento.
- **proc** { $ $\langle y \rangle_1 \cdots \langle y \rangle_n$ } $\langle d \rangle$ **end** crea un valor nuevo de tipo procedimiento.
- La declaración

$$\texttt{proc } \{ \langle x \rangle \ \langle y \rangle_1 \cdots \langle y \rangle_n \} \ \langle d \rangle \ \texttt{end}$$

es lo mismo que

$$\langle x \rangle = \texttt{proc } \{ \$ \ \langle y \rangle_1 \cdots \langle y \rangle_n \} \ \langle d \rangle \ \texttt{end}$$

Sin embargo, oculta la distinción entre crear un valor y ligarlo a un identificador.

Tipos básicos: procedimientos

- Un procedimiento es un valor de tipo procedimiento.
- `proc { $` $\langle y \rangle_1 \cdots \langle y \rangle_n$ `}` $\langle d \rangle$ `end` crea un valor nuevo de tipo procedimiento.
- La declaración

$$\texttt{proc } \{ \langle x \rangle \ \langle y \rangle_1 \cdots \langle y \rangle_n \} \ \langle d \rangle \texttt{ end}$$

es lo mismo que

$$\langle x \rangle = \texttt{proc } \{ \$ \ \langle y \rangle_1 \cdots \langle y \rangle_n \} \ \langle d \rangle \texttt{ end}$$

Sin embargo, oculta la distinción entre crear un valor y ligarlo a un identificador.

Operaciones básicas

Operación	Descripción	Tipo del argumento
`A==B`	Comparación de igualdad	Valor
`A\=B`	Comparación de desigualdad	Valor
`{IsProcedure P}`	Prueba si es procedimiento	Valor
`A=<B`	Comparación menor o igual	Número o Átomo
`A<B`	Comparación menor	Número o Átomo
`A>=B`	Comparación mayor o igual	Número o Átomo
`A>B`	Comparación mayor	Número o Átomo
`A+B`	Suma	Número
`A-B`	Resta	Número
`A*B`	Multiplicación	Número
`A` **`div`** `B`	División	Ent
`A` **`mod`** `B`	Módulo	Ent
`A/B`	División	Flot
`{Arity R}`	Aridad	Registro
`{Label R}`	Etiqueta	Registro
`R.F`	Selección de campo	Registro

¿Por qué registros y procedimientos como tipos básicos?

El poder de los registros

- Cimientos de la mayoría de las estructuras de datos incluyendo listas, árboles, colas, grafos, etc.
- El poder adicional aparece en mayor o menor grado, dependiendo de cuán bien o cuán pobremente se soporten los registros en el lenguaje.
- Para máximo poder, el lenguaje debería facilitar su creación, su exploración, y su manipulación. Los lenguajes que proveen este nivel de soporte para los registros se llaman lenguajes simbólicos.
- Incrementan la efectividad de muchas otras técnicas: programación orientada a objetos, diseño de interfaces gráficas de usuario (GUI, por su sigla en inglés), y programación basada en componentes.

¿Por qué registros y procedimientos como tipos básicos?

El poder de los registros

- Cimientos de la mayoría de las estructuras de datos incluyendo listas, árboles, colas, grafos, etc.
- El poder adicional aparece en mayor o menor grado, dependiendo de cuán bien o cuán pobremente se soporten los registros en el lenguaje.
- Para máximo poder, el lenguaje debería facilitar su creación, su exploración, y su manipulación. Los lenguajes que proveen este nivel de soporte para los registros se llaman lenguajes simbólicos.
- Incrementan la efectividad de muchas otras técnicas: programación orientada a objetos, diseño de interfaces gráficas de usuario (GUI, por su sigla en inglés), y programación basada en componentes.

¿Por qué procedimientos?

¿Por qué registros y procedimientos como tipos básicos?

El poder de los registros

- Cimientos de la mayoría de las estructuras de datos incluyendo listas, árboles, colas, grafos, etc.
- El poder adicional aparece en mayor o menor grado, dependiendo de cuán bien o cuán pobremente se soporten los registros en el lenguaje.
- Para máximo poder, el lenguaje debería facilitar su creación, su exploración, y su manipulación. Los lenguajes que proveen este nivel de soporte para los registros se llaman lenguajes simbólicos.
- Incrementan la efectividad de muchas otras técnicas: programación orientada a objetos, diseño de interfaces gráficas de usuario (GUI, por su sigla en inglés), y programación basada en componentes.

¿Por qué procedimientos?

- ¿Por qué no usar objetos? (POO) ¿o funciones? (PF)
- Los procedimientos son más apropiados que los objetos porque son más simples.

¿Por qué registros y procedimientos como tipos básicos?

El poder de los registros

- Cimientos de la mayoría de las estructuras de datos incluyendo listas, árboles, colas, grafos, etc.
- El poder adicional aparece en mayor o menor grado, dependiendo de cuán bien o cuán pobremente se soporten los registros en el lenguaje.
- Para máximo poder, el lenguaje debería facilitar su creación, su exploración, y su manipulación. Los lenguajes que proveen este nivel de soporte para los registros se llaman lenguajes simbólicos.
- Incrementan la efectividad de muchas otras técnicas: programación orientada a objetos, diseño de interfaces gráficas de usuario (GUI, por su sigla en inglés), y programación basada en componentes.

¿Por qué procedimientos?

- ¿Por qué no usar objetos? (POO) ¿o funciones? (PF)
- Los procedimientos son más apropiados que los objetos porque son más simples.
- Los procedimientos son más adecuados que las funciones porque no definen necesariamente entidades que se comportan como funciones matemáticas.

¿Por qué registros y procedimientos como tipos básicos?

El poder de los registros

- Cimientos de la mayoría de las estructuras de datos incluyendo listas, árboles, colas, grafos, etc.
- El poder adicional aparece en mayor o menor grado, dependiendo de cuán bien o cuán pobremente se soporten los registros en el lenguaje.
- Para máximo poder, el lenguaje debería facilitar su creación, su exploración, y su manipulación. Los lenguajes que proveen este nivel de soporte para los registros se llaman lenguajes simbólicos.
- Incrementan la efectividad de muchas otras técnicas: programación orientada a objetos, diseño de interfaces gráficas de usuario (GUI, por su sigla en inglés), y programación basada en componentes.

¿Por qué procedimientos?

- ¿Por qué no usar objetos? (POO) ¿o funciones? (PF)
- Los procedimientos son más apropiados que los objetos porque son más simples.
- Los procedimientos son más adecuados que las funciones porque no definen necesariamente entidades que se comportan como funciones matemáticas.
- Podemos definir componentes y objetos como abstracciones basadas en procedimientos.
- Los procedimientos son flexibles puesto que no hacen ninguna suposición sobre el número de entradas y salidas.
- Los procedimientos son procesos, por lo tanto adecuados para la concurrencia.

¿Por qué registros y procedimientos como tipos básicos?

El poder de los registros

- Cimientos de la mayoría de las estructuras de datos incluyendo listas, árboles, colas, grafos, etc.
- El poder adicional aparece en mayor o menor grado, dependiendo de cuán bien o cuán pobremente se soporten los registros en el lenguaje.
- Para máximo poder, el lenguaje debería facilitar su creación, su exploración, y su manipulación. Los lenguajes que proveen este nivel de soporte para los registros se llaman lenguajes simbólicos.
- Incrementan la efectividad de muchas otras técnicas: programación orientada a objetos, diseño de interfaces gráficas de usuario (GUI, por su sigla en inglés), y programación basada en componentes.

¿Por qué procedimientos?

- ¿Por qué no usar objetos? (POO) ¿o funciones? (PF)
- Los procedimientos son más apropiados que los objetos porque son más simples.
- Los procedimientos son más adecuados que las funciones porque no definen necesariamente entidades que se comportan como funciones matemáticas.
- Podemos definir componentes y objetos como abstracciones basadas en procedimientos.
- Los procedimientos son flexibles puesto que no hacen ninguna suposición sobre el número de entradas y salidas.
- Los procedimientos son procesos, por lo tanto adecuados para la concurrencia.

¿Por qué registros y procedimientos como tipos básicos?

El poder de los registros

- Cimientos de la mayoría de las estructuras de datos incluyendo listas, árboles, colas, grafos, etc.
- El poder adicional aparece en mayor o menor grado, dependiendo de cuán bien o cuán pobremente se soporten los registros en el lenguaje.
- Para máximo poder, el lenguaje debería facilitar su creación, su exploración, y su manipulación. Los lenguajes que proveen este nivel de soporte para los registros se llaman lenguajes simbólicos.
- Incrementan la efectividad de muchas otras técnicas: programación orientada a objetos, diseño de interfaces gráficas de usuario (GUI, por su sigla en inglés), y programación basada en componentes.

¿Por qué procedimientos?

- ¿Por qué no usar objetos? (POO) ¿o funciones? (PF)
- Los procedimientos son más apropiados que los objetos porque son más simples.
- Los procedimientos son más adecuados que las funciones porque no definen necesariamente entidades que se comportan como funciones matemáticas.
- Podemos definir componentes y objetos como abstracciones basadas en procedimientos.
- Los procedimientos son flexibles puesto que no hacen ninguna suposición sobre el número de entradas y salidas.
- Los procedimientos son procesos, por lo tanto adecuados para la concurrencia.

¿Por qué registros y procedimientos como tipos básicos?

El poder de los registros

- Cimientos de la mayoría de las estructuras de datos incluyendo listas, árboles, colas, grafos, etc.
- El poder adicional aparece en mayor o menor grado, dependiendo de cuán bien o cuán pobremente se soporten los registros en el lenguaje.
- Para máximo poder, el lenguaje debería facilitar su creación, su exploración, y su manipulación. Los lenguajes que proveen este nivel de soporte para los registros se llaman lenguajes simbólicos.
- Incrementan la efectividad de muchas otras técnicas: programación orientada a objetos, diseño de interfaces gráficas de usuario (GUI, por su sigla en inglés), y programación basada en componentes.

¿Por qué procedimientos?

- ¿Por qué no usar objetos? (POO) ¿o funciones? (PF)
- Los procedimientos son más apropiados que los objetos porque son más simples.
- Los procedimientos son más adecuados que las funciones porque no definen necesariamente entidades que se comportan como funciones matemáticas.
- Podemos definir componentes y objetos como abstracciones basadas en procedimientos.
- Los procedimientos son flexibles puesto que no hacen ninguna suposición sobre el número de entradas y salidas.
- Los procedimientos son procesos, por lo tanto adecuados para la concurrencia.

¿Por qué registros y procedimientos como tipos básicos?

El poder de los registros

- Cimientos de la mayoría de las estructuras de datos incluyendo listas, árboles, colas, grafos, etc.
- El poder adicional aparece en mayor o menor grado, dependiendo de cuán bien o cuán pobremente se soporten los registros en el lenguaje.
- Para máximo poder, el lenguaje debería facilitar su creación, su exploración, y su manipulación. Los lenguajes que proveen este nivel de soporte para los registros se llaman lenguajes simbólicos.
- Incrementan la efectividad de muchas otras técnicas: programación orientada a objetos, diseño de interfaces gráficas de usuario (GUI, por su sigla en inglés), y programación basada en componentes.

¿Por qué procedimientos?

- ¿Por qué no usar objetos? (POO) ¿o funciones? (PF)
- Los procedimientos son más apropiados que los objetos porque son más simples.
- Los procedimientos son más adecuados que las funciones porque no definen necesariamente entidades que se comportan como funciones matemáticas.
- Podemos definir componentes y objetos como abstracciones basadas en procedimientos.
- Los procedimientos son flexibles puesto que no hacen ninguna suposición sobre el número de entradas y salidas.
- Los procedimientos son procesos, por lo tanto adecuados para la concurrencia.

¿Por qué registros y procedimientos como tipos básicos?

El poder de los registros

- Cimientos de la mayoría de las estructuras de datos incluyendo listas, árboles, colas, grafos, etc.
- El poder adicional aparece en mayor o menor grado, dependiendo de cuán bien o cuán pobremente se soporten los registros en el lenguaje.
- Para máximo poder, el lenguaje debería facilitar su creación, su exploración, y su manipulación. Los lenguajes que proveen este nivel de soporte para los registros se llaman lenguajes simbólicos.
- Incrementan la efectividad de muchas otras técnicas: programación orientada a objetos, diseño de interfaces gráficas de usuario (GUI, por su sigla en inglés), y programación basada en componentes.

¿Por qué procedimientos?

- ¿Por qué no usar objetos? (POO) ¿o funciones? (PF)
- Los procedimientos son más apropiados que los objetos porque son más simples.
- Los procedimientos son más adecuados que las funciones porque no definen necesariamente entidades que se comportan como funciones matemáticas.
- Podemos definir componentes y objetos como abstracciones basadas en procedimientos.
- Los procedimientos son flexibles puesto que no hacen ninguna suposición sobre el número de entradas y salidas.
- Los procedimientos son procesos, por lo tanto adecuados para la concurrencia.