Paradigmas Fundamentales de Programación

El almacén de variables declarativas

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

Modelos y Paradigmas de Programación

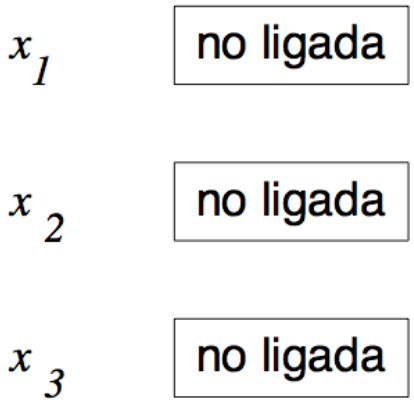
Universidad del Valle

Plan

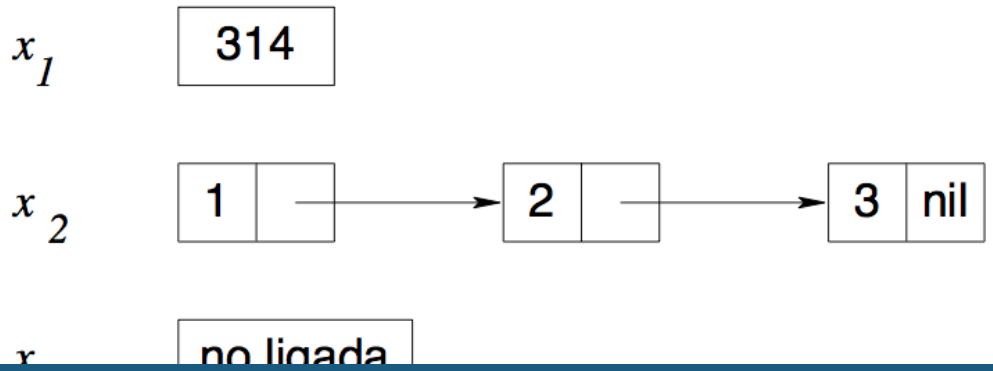
Plan

Almacén de asignación única

- Conjunto de variables que inicialmente no están ligadas y que pueden ser ligadas a un único valor.
- $\{x_1, x_2, x_3\}$:almacén con tres variables no-ligadas.

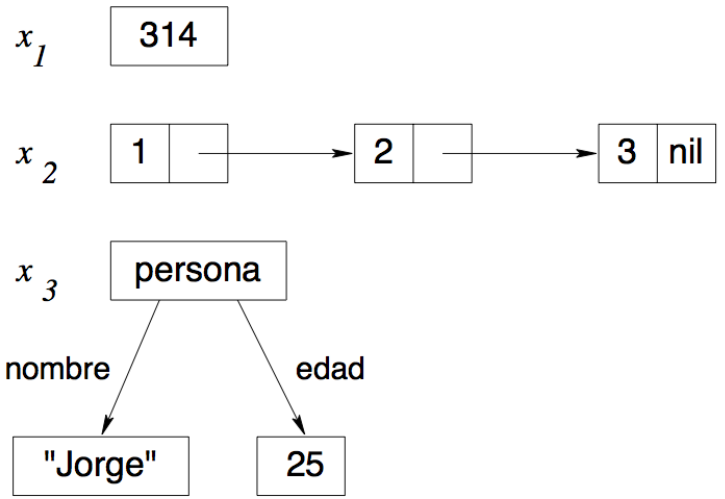


- $\{x_1 = 314, x_2 = \texttt{[1 2 3]}, x_3\}$: almacén con dos variables ligadas.



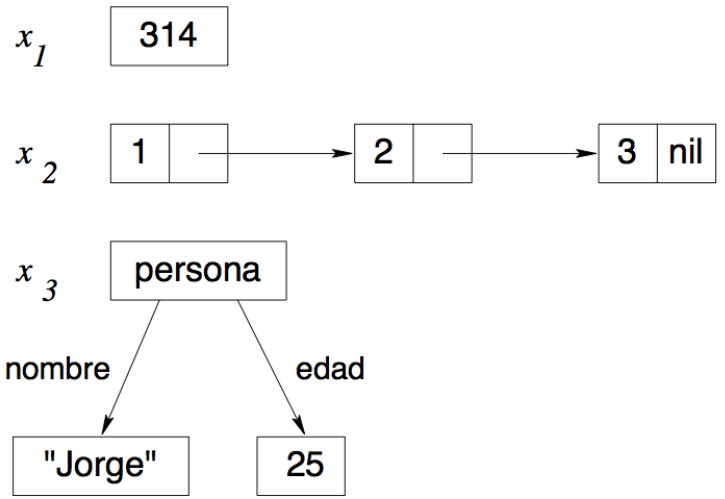
Variables declarativas

- Las variables en el almacén de asignación única se llaman variables declarativas.
- Una vez ligada, una variable declarativa permanece ligada a lo largo de toda la computación.
- Si todas las variables están ligadas a valores: almacén de valores.
- Un valor es una constante en el sentido matemático: 314, `[1 2 3]` y `persona(nombre:"Jorge" edad:25)` son valores.



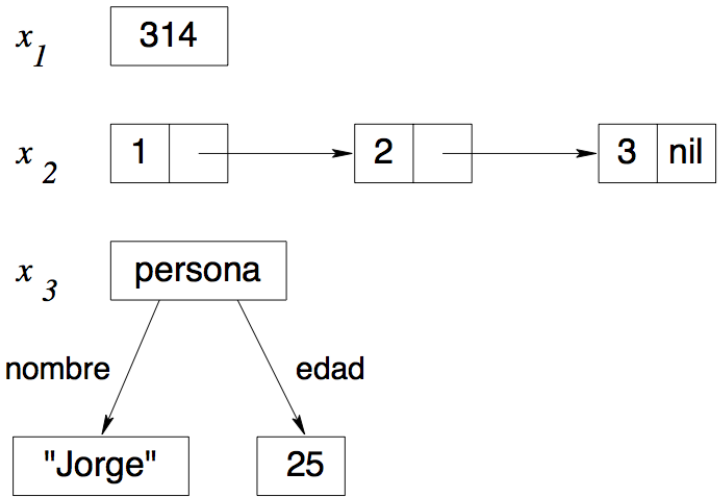
Variables declarativas

- Las variables en el almacén de asignación única se llaman variables declarativas.
- Una vez ligada, una variable declarativa permanece ligada a lo largo de toda la computación.
- Si todas las variables están ligadas a valores: almacén de valores.
- Un valor es una constante en el sentido matemático: 314, `[1 2 3]` y `persona(nombre:"Jorge" edad:25)` son valores.



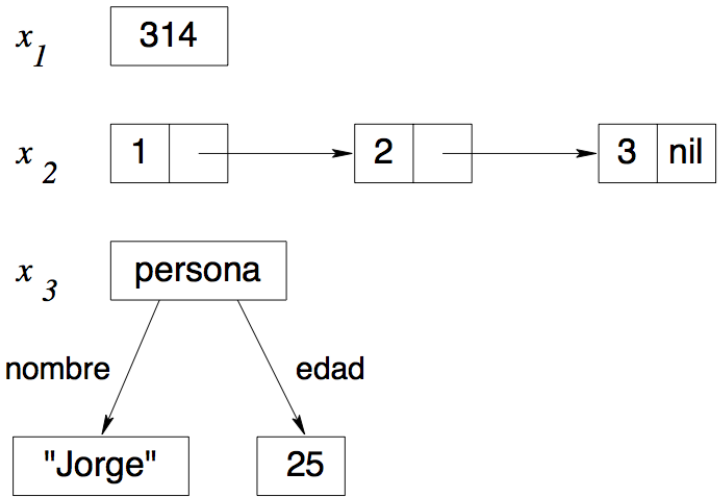
Variables declarativas

- Las variables en el almacén de asignación única se llaman variables declarativas.
- Una vez ligada, una variable declarativa permanece ligada a lo largo de toda la computación.
- Si todas las variables están ligadas a valores: almacén de valores.
- Un valor es una constante en el sentido matemático: 314, `[1 2 3]` y `persona(nombre:"Jorge" edad:25)` son valores.



Variables declarativas

- Las variables en el almacén de asignación única se llaman variables declarativas.
- Una vez ligada, una variable declarativa permanece ligada a lo largo de toda la computación.
- Si todas las variables están ligadas a valores: almacén de valores.
- Un valor es una constante en el sentido matemático: 314, `[1 2 3]` y `persona(nombre:"Jorge" edad:25)` son valores.



¿Para qué un almacén de asignación única?

Almacenes en los lenguajes más conocidos

- Standard ML, Haskell, y Scheme cuentan con un almacén de valores pues ellos calculan funciones sobre valores.
- Smalltalk, C++, y Java cuentan con un almacén de celdas, el cual consta de celdas cuyo contenido puede ser modificado.

Razones

- Calcular con valores parciales.
- La concurrencia declarativa.

¿Para qué un almacén de asignación única?

Almacenes en los lenguajes más conocidos

- Standard ML, Haskell, y Scheme cuentan con un almacén de valores pues ellos calculan funciones sobre valores.
- Smalltalk, C++, y Java cuentan con un almacén de celdas, el cual consta de celdas cuyo contenido puede ser modificado.

Razones

- Calcular con valores parciales.
- La concurrencia declarativa.
- La programación relacional (lógica) y la programación por restricciones.

¿Para qué un almacén de asignación única?

Almacenes en los lenguajes más conocidos

- Standard ML, Haskell, y Scheme cuentan con un almacén de valores pues ellos calculan funciones sobre valores.
- Smalltalk, C++, y Java cuentan con un almacén de celdas, el cual consta de celdas cuyo contenido puede ser modificado.

Razones

- Calcular con valores parciales.
- La concurrencia declarativa.
- La programación relacional (lógica) y la programación por restricciones.
- Eficiencia: recursión de cola y listas de diferencias.

¿Para qué un almacén de asignación única?

Almacenes en los lenguajes más conocidos

- Standard ML, Haskell, y Scheme cuentan con un almacén de valores pues ellos calculan funciones sobre valores.
- Smalltalk, C++, y Java cuentan con un almacén de celdas, el cual consta de celdas cuyo contenido puede ser modificado.

Razones

- Calcular con valores parciales.
- La concurrencia declarativa.
- La programación relacional (lógica) y la programación por restricciones.
- Eficiencia: recursión de cola y listas de diferencias.

¿Para qué un almacén de asignación única?

Almacenes en los lenguajes más conocidos

- Standard ML, Haskell, y Scheme cuentan con un almacén de valores pues ellos calculan funciones sobre valores.
- Smalltalk, C++, y Java cuentan con un almacén de celdas, el cual consta de celdas cuyo contenido puede ser modificado.

Razones

- Calcular con valores parciales.
- La concurrencia declarativa.
- La programación relacional (lógica) y la programación por restricciones.
- Eficiencia: recursión de cola y listas de diferencias.

¿Para qué un almacén de asignación única?

Almacenes en los lenguajes más conocidos

- Standard ML, Haskell, y Scheme cuentan con un almacén de valores pues ellos calculan funciones sobre valores.
- Smalltalk, C++, y Java cuentan con un almacén de celdas, el cual consta de celdas cuyo contenido puede ser modificado.

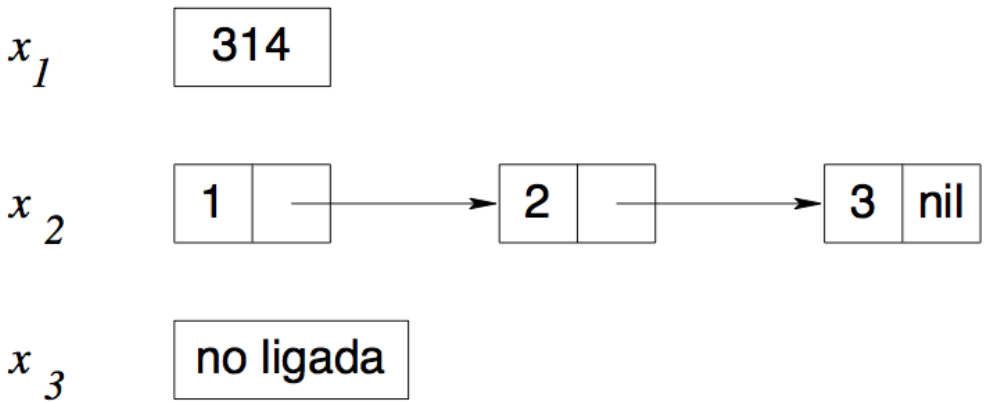
Razones

- Calcular con valores parciales.
- La concurrencia declarativa.
- La programación relacional (lógica) y la programación por restricciones.
- Eficiencia: recursión de cola y listas de diferencias.

Creación de valores

Ligadura de una variable con un valor

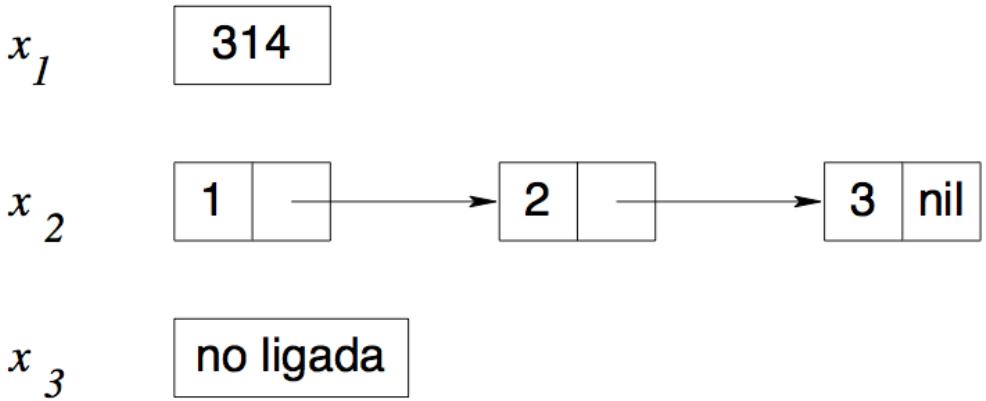
- x_i=*valor* construye el *valor* en el almacén y luego liga la variable x_i a ese valor.
- Si la variable ya está ligada, la operación verificará si los dos valores son compatibles. Si no lo son, se señala un error.
- x_1 = *314*
  x_2 = `[1 2 3]`
- El almacén resultante es:



Creación de valores

Ligadura de una variable con un valor

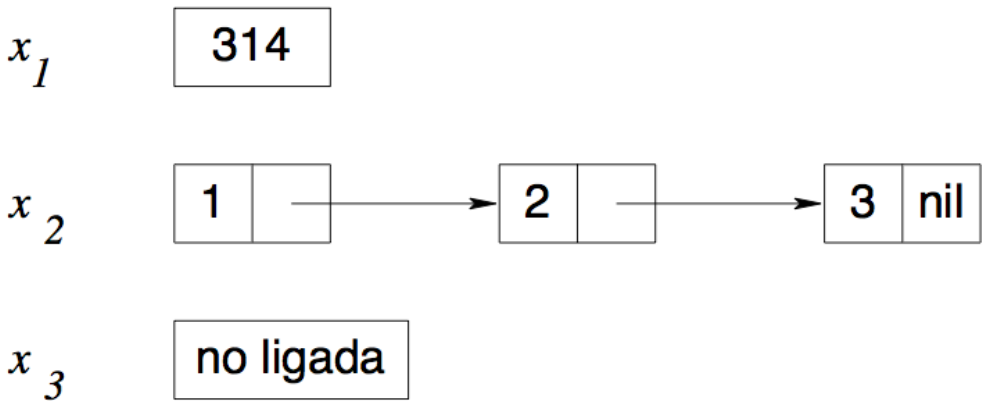
- x_i=*valor* construye el *valor* en el almacén y luego liga la variable x_i a ese valor.
- Si la variable ya está ligada, la operación verificará si los dos valores son compatibles. Si no lo son, se señala un error.
- x_1 = *314*
  x_2 = `[1 2 3]`
- El almacén resultante es:



Creación de valores

Ligadura de una variable con un valor

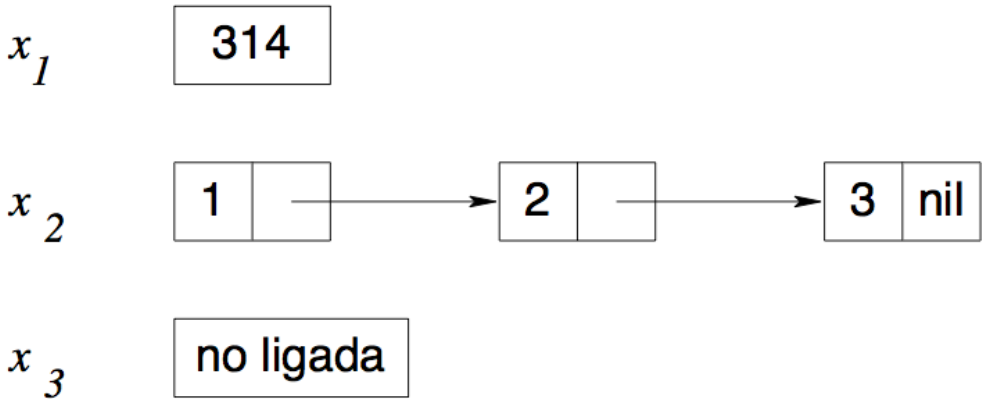
- x_i=*valor* construye el *valor* en el almacén y luego liga la variable x_i a ese valor.
- Si la variable ya está ligada, la operación verificará si los dos valores son compatibles. Si no lo son, se señala un error.
- x_1 = *314*
  x_2 = `[1 2 3]`
- El almacén resultante es:



Creación de valores

Ligadura de una variable con un valor

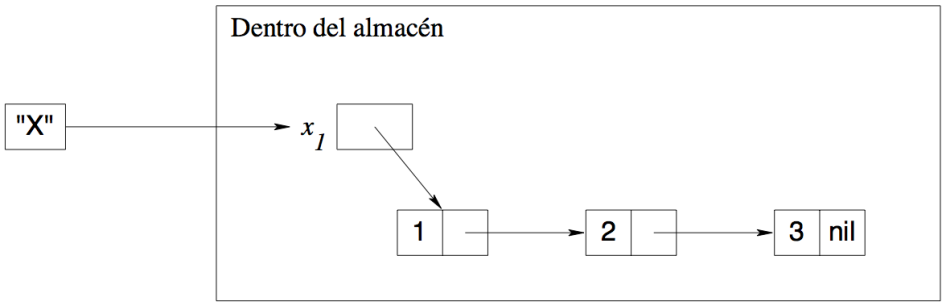
- x_i=*valor* construye el *valor* en el almacén y luego liga la variable x_i a ese valor.
- Si la variable ya está ligada, la operación verificará si los dos valores son compatibles. Si no lo son, se señala un error.
- x_1 = *314*
  x_2 = `[1 2 3]`
- El almacén resultante es:



Identificadores de variables

¿Cómo referenciar entidades del almacén?

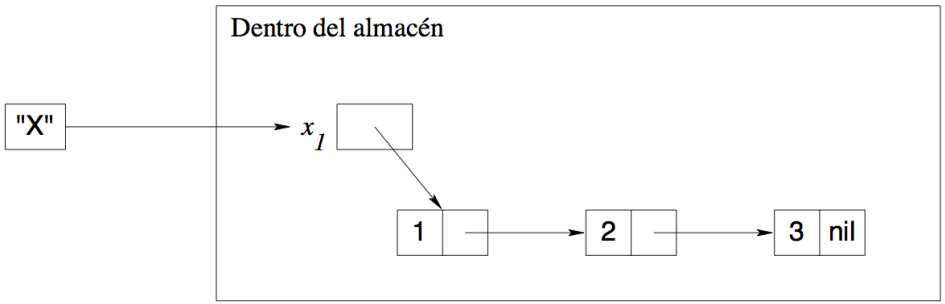
- Identificador de variable: nombre textual que referencia una entidad del almacén desde afuera de éste.
- Ambiente: Asociación de identificadores de variables con entidades del almacén.
- Ejemplo:



Identificadores de variables

¿Cómo referenciar entidades del almacén?

- Identificador de variable: nombre textual que referencia una entidad del almacén desde afuera de éste.
- Ambiente: Asociación de identificadores de variables con entidades del almacén.
- Ejemplo:



Identificador de variable y variable del almacén

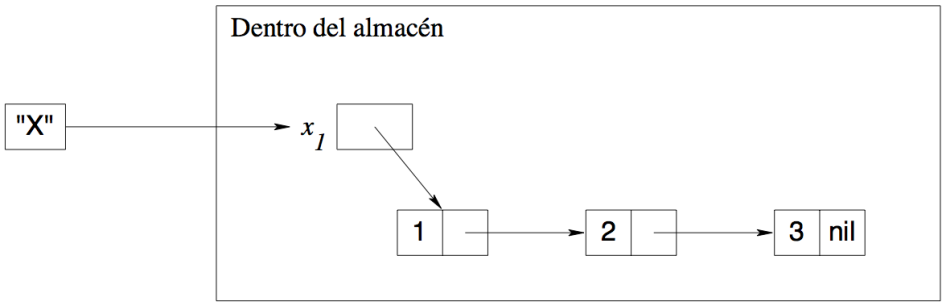
- Ambiente $\{X \rightarrow x_1\}$.
- Notación $\langle x \rangle$: Si $\langle x \rangle$ representa X, entonces el ambiente $\{\langle x \rangle \rightarrow x_1\}$ es el mismo que acabamos de mencionar.

Los identificadores de variables y sus correspondientes entidades del almacén se agregan al ambiente por medio de las declaraciones `local` y `declare`.

Identificadores de variables

¿Cómo referenciar entidades del almacén?

- Identificador de variable: nombre textual que referencia una entidad del almacén desde afuera de éste.
- Ambiente: Asociación de identificadores de variables con entidades del almacén.
- Ejemplo:



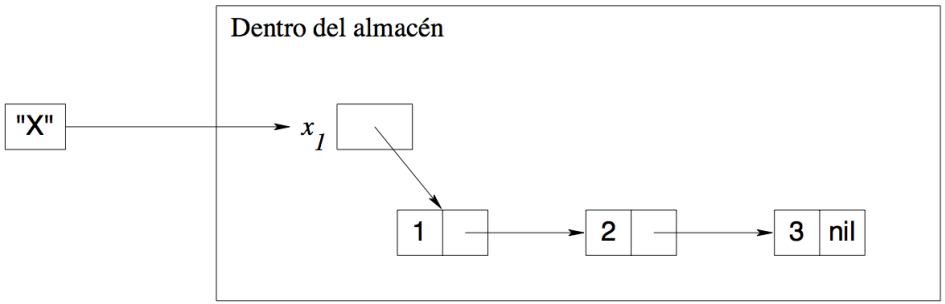
Identificador de variable y variable del almacén

- Ambiente $\{\mathrm{X} \rightarrow x_1\}$.
- Notación $\langle \mathrm{x} \rangle$: Si $\langle \mathrm{x} \rangle$ representa X, entonces el ambiente $\{\langle \mathrm{x} \rangle \rightarrow x_1\}$ es el mismo que acabamos de mencionar.
- Los identificadores de variables y sus correspondientes entidades del almacén se agregan al ambiente por medio de las declaraciones `local` y `declare`.

Identificadores de variables

¿Cómo referenciar entidades del almacén?

- Identificador de variable: nombre textual que referencia una entidad del almacén desde afuera de éste.
- Ambiente: Asociación de identificadores de variables con entidades del almacén.
- Ejemplo:



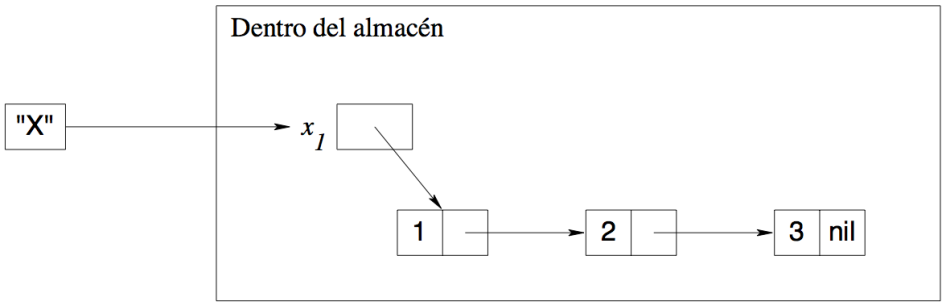
Identificador de variable y variable del almacén

- Ambiente $\{\mathrm{X} \rightarrow x_1\}$.
- Notación $\langle \mathrm{x} \rangle$: Si $\langle \mathrm{x} \rangle$ representa X, entonces el ambiente $\{\langle \mathrm{x} \rangle \rightarrow x_1\}$ es el mismo que acabamos de mencionar.
- Los identificadores de variables y sus correspondientes entidades del almacén se agregan al ambiente por medio de las declaraciones `local` y `declare`.

Identificadores de variables

¿Cómo referenciar entidades del almacén?

- Identificador de variable: nombre textual que referencia una entidad del almacén desde afuera de éste.
- Ambiente: Asociación de identificadores de variables con entidades del almacén.
- Ejemplo:

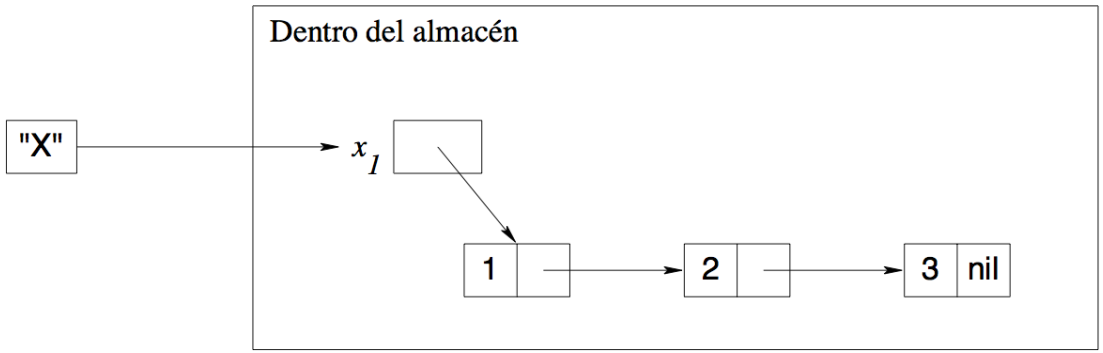


Identificador de variable y variable del almacén

- Ambiente $\{\mathrm{X} \rightarrow x_1\}$.
- Notación $\langle \mathrm{x} \rangle$: Si $\langle \mathrm{x} \rangle$ representa X, entonces el ambiente $\{\langle \mathrm{x} \rangle \rightarrow x_1\}$ es el mismo que acabamos de mencionar.
- Los identificadores de variables y sus correspondientes entidades del almacén se agregan al ambiente por medio de las declaraciones **`local`** y **`declare`**.

Creación de valores con identificadores

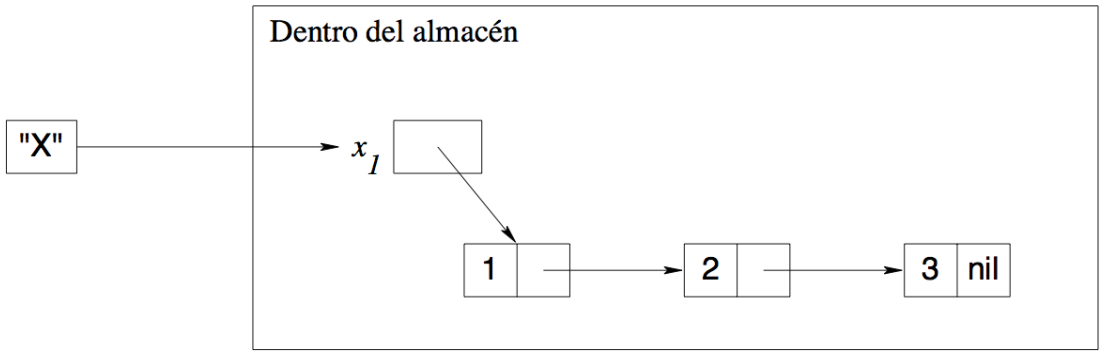
- "=": operación de ligadura.
- `X=[1 2 3]`: el identificador "X" aún referencia a x_1, pero éste está ligado ahora a `[1 2 3]`.



- Desreferenciación:conseguir el valor asociado a un identificador.

Creación de valores con identificadores

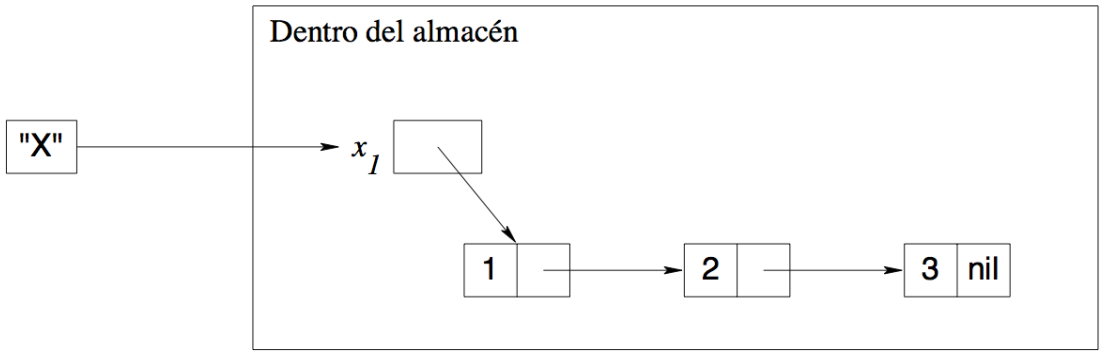
- "=": operación de ligadura.
- `X=[1 2 3]`: el identificador "`X`" aún referencia a x_1, pero éste está ligado ahora a `[1 2 3]`.



- Desreferenciación:conseguir el valor asociado a un identificador.

Creación de valores con identificadores

- "=": operación de ligadura.
- `X=[1 2 3]`: el identificador "`X`" aún referencia a x_1, pero éste está ligado ahora a `[1 2 3]`.

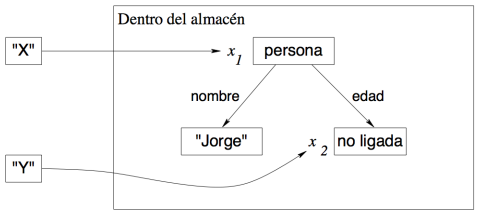


- Desreferenciación:conseguir el valor asociado a un identificador.

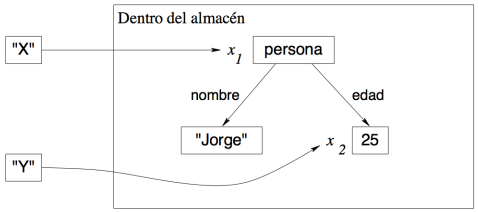
Valores parciales

Valor parcial o completo

- Valor parcial: estructura de datos que puede contener variables no-ligadas.
- Ejemplo:



- Después de `Y=25`:



- x_1 es un valor completo

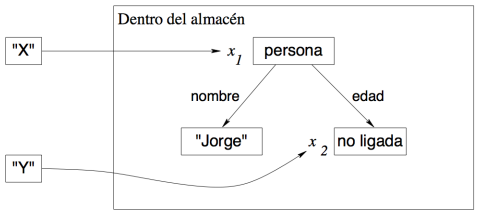
Observaciones

- Una variable declarativa puede ser ligada a varios valores parciales siempre y cuando estos sean compatibles dos a dos.
- Decimos que un conjunto de valores parciales es compatible si las variables no ligadas que se encuentran en ellos se pueden ligar de manera que todos los valores sean iguales.
- `persona(edad:25)` y `persona(edad:x)` son compatibles
- `persona(edad:25)` y `persona(edad:26)` no lo son.

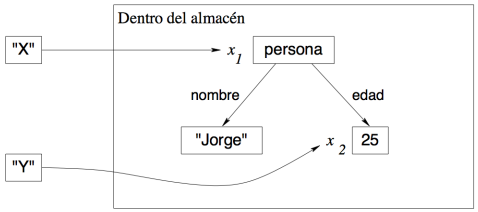
Valores parciales

Valor parcial o completo

- Valor parcial: estructura de datos que puede contener variables no-ligadas.
- Ejemplo:



- Después de `Y=25`:



- x_1 es un valor completo

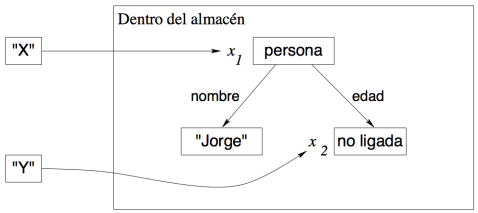
Observaciones

- Una variable declarativa puede ser ligada a varios valores parciales siempre y cuando estos sean compatibles dos a dos.
- Decimos que un conjunto de valores parciales es compatible si las variables no ligadas que se encuentran en ellos se pueden ligar de manera que todos los valores sean iguales.
- `persona(edad:25)` y `persona(edad:X)` son compatibles
- `persona(edad:25)` y `persona(edad:26)` no lo son.

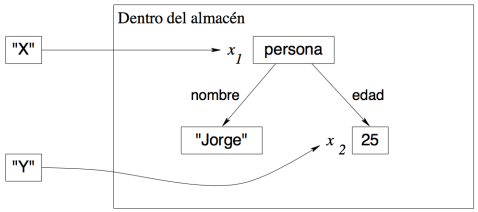
Valores parciales

Valor parcial o completo

- Valor parcial: estructura de datos que puede contener variables no-ligadas.
- Ejemplo:



- Después de `Y=25`:



- x_1 es un valor completo

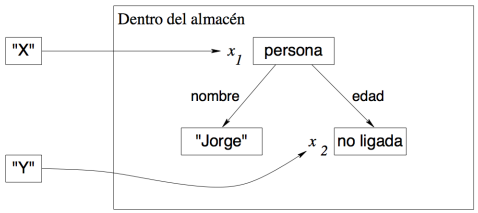
Observaciones

- Una variable declarativa puede ser ligada a varios valores parciales siempre y cuando estos sean compatibles dos a dos.
- Decimos que un conjunto de valores parciales es compatible si las variables no ligadas que se encuentran en ellos se pueden ligar de manera que todos los valores sean iguales.
- `persona(edad:25)` y `persona(edad:x)` son compatibles
- `persona(edad:25)` y `persona(edad:26)` no lo son.

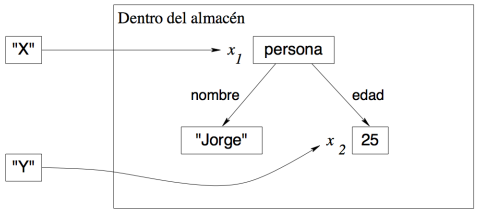
Valores parciales

Valor parcial o completo

- Valor parcial: estructura de datos que puede contener variables no-ligadas.
- Ejemplo:



- Después de `Y=25`:



- x_1 es un valor completo

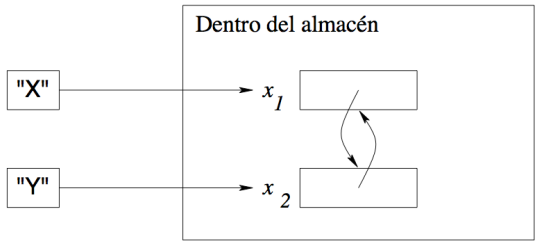
Observaciones

- Una variable declarativa puede ser ligada a varios valores parciales siempre y cuando estos sean compatibles dos a dos.
- Decimos que un conjunto de valores parciales es compatible si las variables no ligadas que se encuentran en ellos se pueden ligar de manera que todos los valores sean iguales.
- `persona(edad:25)` y `persona(edad:x)` son compatibles
- `persona(edad:25)` y `persona(edad:26)` no lo son.

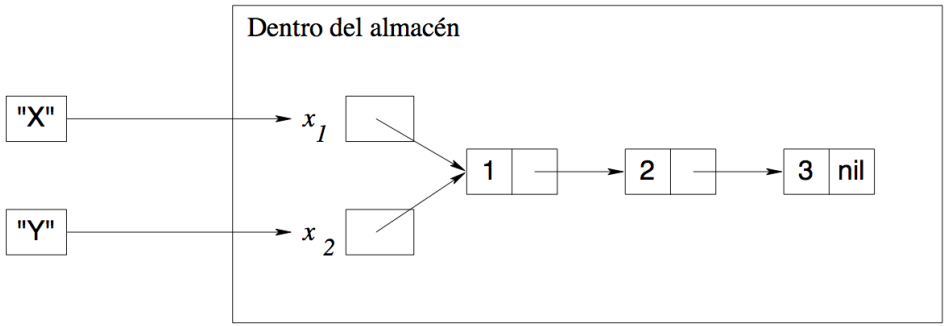
Ligadura variable-variable

En el ambiente $\{\texttt{X} \longrightarrow x_1, \texttt{Y} \longrightarrow x_2\}$

Después de `X=Y`: x_1 y x_2 son iguales entre sí.



Después de `X=[1 2 3]`



Variables de flujo de datos

Creación y ligadura de variables

- Al mismo momento: lengs. funcs.
- Primero uso; luego creación: error.
- Primero creación; luego uso: error de uso.

¿Qué hacer ante un error de uso?

- Ejecución continúa; no mensaje de error. El contenido de la variable es indefinido, i.e., es "basura": C++.
- Ejecución continúa; no mensaje de error. La variable toicia en un valor por defecto: Java.
- Ejecución se detiene con un mensaje de error: Prolog.
- Ejecución se suspende y espera hasta que la variable sea ligada y luego continúa: Oz.

Ventajas de la suspensión

Variables de flujo de datos

Creación y ligadura de variables

- Al mismo momento: lengs. funcs.
- Primero uso; luego creación: error.
- Primero creación; luego uso: error de uso.

¿Qué hacer ante un error de uso?

- Ejecución continúa; no mensaje de error. El contenido de la variable es indefinido, i.e., es “basura”: C++.
- Ejecución continúa; no mensaje de error. La variable inicia en un valor por defecto: Java.
- Ejecución se detiene con un mensaje de error: Prolog.
- Ejecución se suspende y espera hasta que la variable sea ligada y luego continúa: Oz.

Ventajas de la suspensión

Variables de flujo de datos

Creación y ligadura de variables

- Al mismo momento: lengs. funcs.
- Primero uso; luego creación: error.
- Primero creación; luego uso: error de uso.

¿Qué hacer ante un error de uso?

- Ejecución continúa; no mensaje de error. El contenido de la variable es indefinido, i.e., es “basura”: C++.
- Ejecución continúa; no mensaje de error. La variable inicia en un valor por defecto: Java.
- Ejecución se detiene con un mensaje de error: Prolog.
- Ejecución se suspende y espera hasta que la variable sea ligada y luego continúa: Oz.

Ventajas de la suspensión

Variables de flujo de datos

Creación y ligadura de variables

- Al mismo momento: lengs. funcs.
- Primero uso; luego creación: error.
- Primero creación; luego uso: error de uso.

¿Qué hacer ante un error de uso?

- Ejecución continúa; no mensaje de error. El contenido de la variable es indefinido, i.e., es “basura”: C++.
- Ejecución continúa; no mensaje de error. La variable inicia en un valor por defecto: Java.
- Ejecución se detiene con un mensaje de error: Prolog.
- Ejecución se suspende y espera hasta que la variable sea ligada y luego continúa: Oz.

Ventajas de la suspensión

- variables de flujo de datos: variables declarativas que hacen que un programa se suspenda hasta que sean ligadas.
- Disponible en algunos lenguajes de programación.

Variables de flujo de datos

Creación y ligadura de variables

- Al mismo momento: lengs. funcs.
- Primero uso; luego creación: error.
- Primero creación; luego uso: error de uso.

¿Qué hacer ante un error de uso?

- Ejecución continúa; no mensaje de error. El contenido de la variable es indefinido, i.e., es “basura”: C++.
- Ejecución continúa; no mensaje de error. La variable inicia en un valor por defecto: Java.
- Ejecución se detiene con un mensaje de error: Prolog.
- Ejecución se suspende y espera hasta que la variable sea ligada y luego continúa: Oz.

Ventajas de la suspensión

- variables de flujo de datos: variables declarativas que hacen que un programa se suspenda hasta que sean ligadas.
- Supremamente útiles en programación concurrente.
- Si $Z = X + Y$ y $Z = 3$ y $Y = 1$ se ejecuta correctamente porque se calculará X correctamente (propiedad de independencia del orden).

Variables de flujo de datos

Creación y ligadura de variables

- Al mismo momento: lengs. funcs.
- Primero uso; luego creación: error.
- Primero creación; luego uso: error de uso.

¿Qué hacer ante un error de uso?

- Ejecución continúa; no mensaje de error. El contenido de la variable es indefinido, i.e., es “basura”: C++.
- Ejecución continúa; no mensaje de error. La variable inicia en un valor por defecto: Java.
- Ejecución se detiene con un mensaje de error: Prolog.
- Ejecución se suspende y espera hasta que la variable sea ligada y luego continúa: Oz.

Ventajas de la suspensión

- variables de flujo de datos: variables declarativas que hacen que un programa se suspenda hasta que sean ligadas.
- Supremamente útiles en programación concurrente.
- Si `A=23` y `B=A+1` se ejecutan concurrentemente, siempre se calculará `B` correctamente: propiedad de independencia del orden.

Variables de flujo de datos

Creación y ligadura de variables

- Al mismo momento: lengs. funcs.
- Primero uso; luego creación: error.
- Primero creación; luego uso: error de uso.

¿Qué hacer ante un error de uso?

- Ejecución continúa; no mensaje de error. El contenido de la variable es indefinido, i.e., es “basura”: C++.
- Ejecución continúa; no mensaje de error. La variable inicia en un valor por defecto: Java.
- Ejecución se detiene con un mensaje de error: Prolog.
- Ejecución se suspende y espera hasta que la variable sea ligada y luego continúa: Oz.

Ventajas de la suspensión

- variables de flujo de datos: variables declarativas que hacen que un programa se suspenda hasta que sean ligadas.
- Supremamente útiles en programación concurrente.
- Si `A=23` y `B=A+1` se ejecutan concurrentemente, siempre se calculará `B` correctamente: propiedad de independencia del orden.

Variables de flujo de datos

Creación y ligadura de variables

- Al mismo momento: lengs. funcs.
- Primero uso; luego creación: error.
- Primero creación; luego uso: error de uso.

¿Qué hacer ante un error de uso?

- Ejecución continúa; no mensaje de error. El contenido de la variable es indefinido, i.e., es “basura”: C++.
- Ejecución continúa; no mensaje de error. La variable inicia en un valor por defecto: Java.
- Ejecución se detiene con un mensaje de error: Prolog.
- Ejecución se suspende y espera hasta que la variable sea ligada y luego continúa: Oz.

Ventajas de la suspensión

- variables de flujo de datos: variables declarativas que hacen que un programa se suspenda hasta que sean ligadas.
- Supremamente útiles en programación concurrente.
- Si `A=23` y `B=A+1` se ejecutan concurrentemente, siempre se calculará `B` correctamente: propiedad de independencia del orden.