

# GOBSTONES

## Nuevos caminos en la enseñanza de programación



Pablo E. “Fidel” Martínez López  
Universidad Nacional de Quilmes  
octubre 2013



*“Podemos ver a un programa como lo que convierte una computadora de propósitos generales en un manipulador de símbolos de propósitos específicos, y lo hace sin necesidad de cambiar ni un solo cable. (...) Prefiero describirla justo de la manera opuesta: **un programa es un manipulador abstracto de símbolos, que puede transformarse en uno concreto suministrándole una computadora.** Después de todo, no es más el propósito de los programas dar instrucciones a nuestras máquinas; en estos días, es el propósito de las máquinas ejecutar nuestros programas.”*

*Edsger W. Dijkstra  
Sobre la crueldad de enseñar realmente ciencias  
computacionales, 1988*



*“...nunca se refieran a partes de programas o piezas de equipo con terminología antropomórfica, ni permitan que sus estudiantes lo hagan. Esta mejora lingüística es mucho más difícil de implementar de lo que podrían pensar...”*

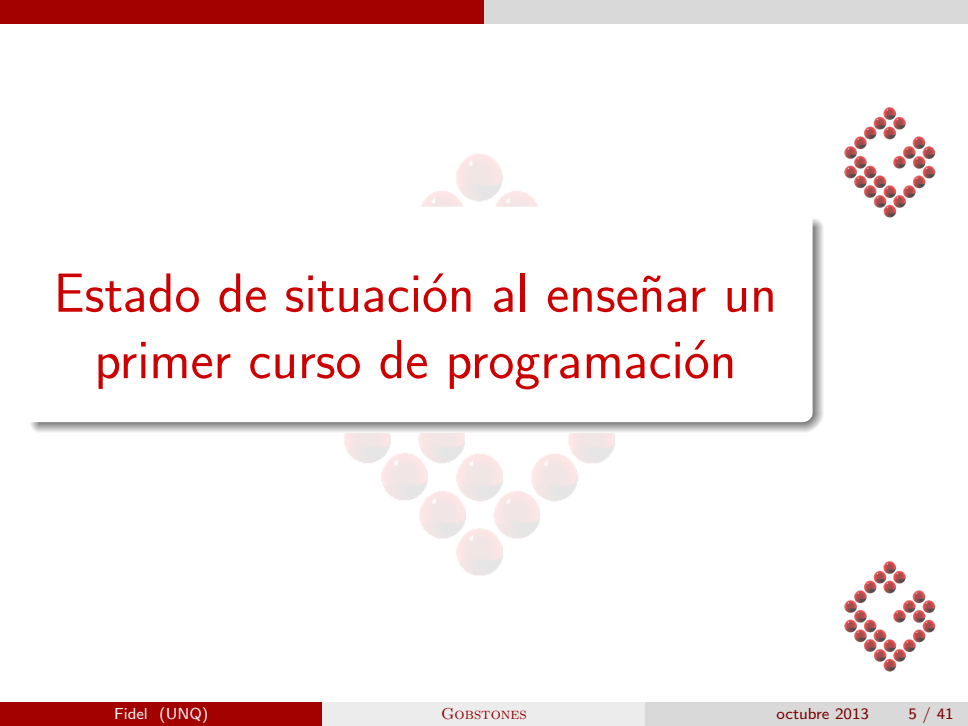
*Edsger W. Dijkstra  
Sobre la crueldad de enseñar realmente ciencias  
computacionales, 1988*



# Overview

- 1 Estado de situación al enseñar un primer curso de programación
- 2 Nuestra solución
  - Nuestra concepción de la programación
  - Secuencia didáctica innovadora
  - Selección de herramientas
- 3 GOBSTONES
- 4 Aplicación del enfoque
- 5 Conclusiones





# Estado de situación al enseñar un primer curso de programación

# Estudiante ideal vs. estudiante real

- Estudiante ideal
  - Buena base matemática
  - Entiende consignas y metáforas
  - Distingue lo fundamental de lo accesorio
  - Autónomo y proactivo
  - Capacidad de autoevaluación
- Estudiante real
  - Base matemática pobre
  - Poca capacidad de abstracción
  - Poca autonomía
  - Se pierde en las cuestiones accesorias



# ¿Cómo lo enfrentamos los docentes?

- El docente tradicional
  - Culpa al estudiante por sus faltas (de interés, de capacidad, etc.)
  - Insiste con métodos que funcionan a medias
  - Se refugia en frases exculpatorias (“algunos aprenden”, “no puedo hacer otra cosa”, “yo aprendí así”, etc.)
- Nosotros
  - Buscamos entender qué está mal
  - Conversamos con los estudiantes
  - Intentamos ponernos en su lugar



# ¿Qué encontramos en relación con la programación?

- Los enfoques tradicionales de enseñar un primer curso de programación
  - Demasiada información junta
  - Requerimientos altos en nivel de abstracción
  - Dependencia de conocimientos ausentes
  - Poca explicitación de conceptos fundamentales
  - Muchos elementos accesorios irrelevantes o complejos
- ¡Se imponía para nosotros rediseñar la secuencia de aprendizaje de programación!







# Nuestra solución



# ¿Qué enseñar en un primer curso y por qué?

- Preguntas que nos hicimos
  - ¿Qué transmitir?
  - ¿Qué enfoque utilizar?
  - ¿Qué elementos incluir?
  - ¿Cómo fundamentar estas decisiones?
- Todo esto teniendo en cuenta a los estudiantes reales



# ¿Qué enseñar en un primer curso y por qué?

- ¿Qué transmitir?
  - Formación de pensamiento abstracto
  - Ideas conceptuales fundamentales
  - Manejo de abstracciones básicas en materias posteriores
- ¿Qué enfoque utilizar?
  - Minimalista (solo ideas fundamentales)
  - De lo concreto a lo abstracto
  - Más detalles en un ratito...



# ¿Qué enseñar en un primer curso y por qué?

- ¿Qué elementos incluir?
  - Seleccionamos cuidadosamente los conceptos
  - Diseñamos un lenguaje conciso (GOBSTONES) basado en esa selección
  - Pero no olvidamos que el lenguaje es el *medio* y NO el objetivo
- ¿Cómo fundamentar estas decisiones?
  - Este es el punto más controversial
  - Tratamos de ofrecer ideas que trasciendan paradigmas y lenguajes particulares (conceptos transversales, generalizables)
  - Buscamos simplicidad sin sobresimplificación



# Concepción deseada de la programación

- Paradojas de la programación
  - La importancia del lenguaje
  - La naturaleza de los programas
- Conceptos fundamentales de programación
  - Elementos del lenguaje
  - Manejo del lenguaje
  - Herramientas abstractas



# Concepción deseada de la programación



## La paradoja del lenguaje

*“El lenguaje de programación que utilizamos  
no es importante,  
pero es extremadamente importante.”*

- Justificación
  - Es prioritario manejar y transmitir ideas
    - ¡el lenguaje es una herramienta!
  - Manipular las ideas es imprescindible
    - ¡el lenguaje es LA ÚNICA herramienta!



# Concepción deseada de la programación

- 
- 
- Es necesario que, en esta etapa del aprendizaje,
    - el lenguaje no se convierta en el verdadero objeto de estudio (minimizar elementos específicos del lenguaje)
    - las ideas sean comunes a todos los paradigmas y todos los lenguajes
    - nos concentremos en la *esencia* de las ideas
    - valoremos el pensamiento abstracto por sobre el pensamiento concreto

# Concepción deseada de la programación

## La paradoja de la naturaleza de los programas

*“Debemos entender a los programas olvidando que son entidades operacionales, pero sin olvidar que son entidades operacionales.”*

### • Justificación

- El modelo de ejecución siempre es operacional
  - aspecto operacional, de bajo nivel, *concreto*
- Los programas describen transformaciones de información, interacción con otros componentes y elementos abstractos
  - aspecto denotacional, de alto nivel, *abstracto*
- ¡Debemos tener en cuenta ambos, concentrándonos en el abstracto!





# Concepción deseada de la programación

- Es necesario contar con una definición de programa que
  - abarque estos dos aspectos de manera equitativa
  - trascienda los paradigmas y lenguajes particulares
  - permita a un programador elegir en cada momento cuál de estos aspectos privilegiar



## Definición propuesta

*“Los programas son **descripciones ejecutables** de soluciones a problemas computacionales.”*

- Así podemos
  - ignorar a conveniencia el aspecto operacional, ¡sin dejarlo nunca de lado!
  - revalorizar el aspecto de descripción de nuestros programas (en cuanto a *texto*, y en cuanto a *qué se describe*)



# Selección de contenidos

- La elección de qué contenidos incluir en un primer curso
  - es de capital importancia
  - debe atender a la resolución de las paradojas enunciadas
  - debe acotarse al mínimo imprescindible



# Selección de contenidos

- Identificamos 3 categorías de contenidos

- (A) Elementos del lenguaje

- ① Categorías de elementos
    - ② Formas de combinación de elementos
    - ③ Herramientas para expresar abstracción

- (B) Manejo del lenguaje

- ① Manejo de sintaxis dura
    - ② Buenas prácticas

- (C) Herramientas abstractas

- ① División en subproblemas
    - ② Parametrización
    - ③ Parcialidad y precondiciones
    - ④ Esquemas de programas sencillos

» expandir



» expandir



» expandir



# Selección de contenidos

## ● Contenidos adicionales

- ¿Modelado de información?
- ¿Entrada/Salida?
- ¿Programación funcional?
- ¿Programación orientada a objetos?
- ¿Formalización de las ideas?
- ¿Estructuras de control más complejas/otras herramientas?
- ¿Arreglos?
- ¿Otras estructuras de datos?

» expandir



# Problemáticas posibles



- Errores posibles en la presentación

[▶ expandir](#)

- Exigir abstracción demasiado pronto
- Dilatar el uso de elementos concretos
- Abusar de metáforas y analogías
- Abusar de recursos gráficos
- Concentrarse demasiado en aspectos operacionales
- Incluir cuestiones de diseño
- Presentar ejemplos que requieran soluciones ad-hoc



# Selección del lenguaje de programación



- El lenguaje de programación es menos importante que las ideas
- Cualquier lenguaje podría servir. . .
- Sin embargo, no debería contener características o detalles irrelevantes
- Por eso diseñamos nuestro propio lenguaje: GOBSTONES





# Gobstones



# El lenguaje GOBSTONES

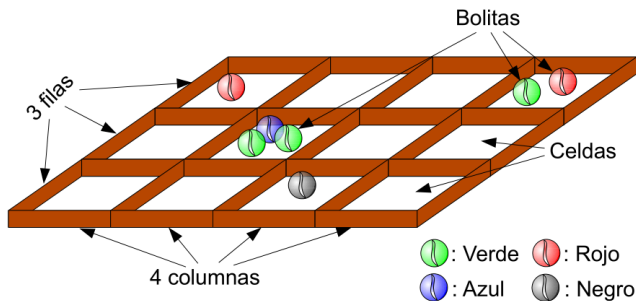
- No posee elementos indeseados (entrada/salida, estructuras de datos, etc.)
- Posee clara separación entre elementos con efectos y elementos puros
- Utiliza elementos concretos (tablero, bolitas) como universo de discurso
- No descansa sobre sus aspectos imperativos
- No tiene las limitaciones de otros lenguajes específicos (e.g. LOGO)
  - No está ligado a su universo de discurso
  - No orienta el pensamiento operacional
  - La transición a otros lenguajes mainstream es poco costosa (i.e. C, JAVA)
- Es libre





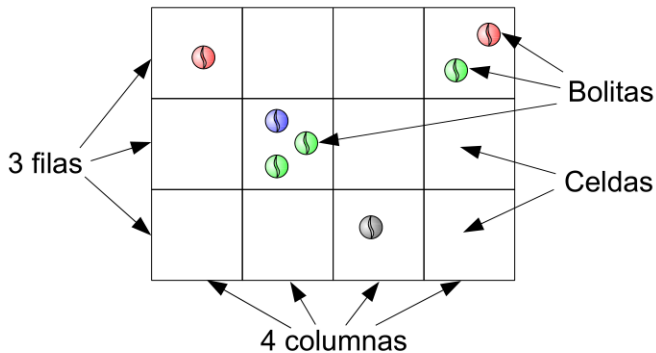
# Conociendo GOBSTONES

- Universo de discurso: Tablero y bolitas
  - elementos concretos
  - reemplazan a la memoria
  - permiten representaciones inicialmente visibles
  - no restringen las posibilidades computacionales



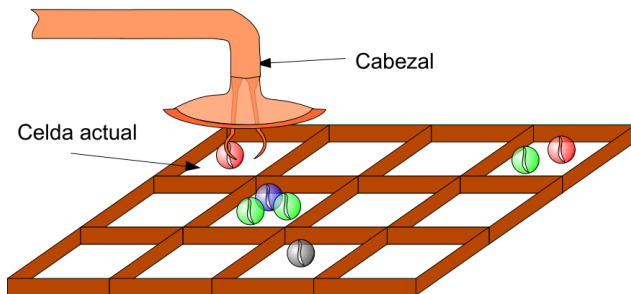
# Conociendo GOBSTONES

- Universo de discurso: Tablero y bolitas



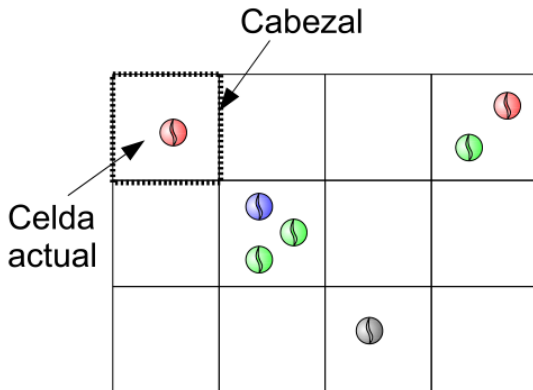
# Conociendo GOBSTONES

- Universo de discurso: Cabezal
  - provee un abanico de *acciones*
  - es comandado por los programas



# Conociendo GOBSTONES

- Universo de discurso: Cabezal 



# Conociendo GOBSTONES



- Elementos de programas GOBSTONES: “mundo” de las acciones
  - Programas (program)
  - Comandos
    - Comandos primitivos (Poner, Mover, etc.)
    - Estructuras de control básicas (secuencia, repeat, if-then-else, while, etc.)
    - Comandos definidos por el usuario
  - Procedimientos
    - ¡Abstracción de combinaciones de comandos!



# Conociendo GOBSTONES



- Elementos de programas GOBSTONES: “mundo” de los valores
  - Expresiones
    - Expresiones literales (números, colores, direcciones, etc.)
    - Operaciones sobre expresiones (suma, comparaciones, etc.)
    - Funciones primitivas (`hayBolitas`, `puedeMover`, etc.)
    - Funciones definidas por el usuario
    - Uso de *nombres* (parámetros, índices y variables)
  - Funciones
    - ¡Abstracción de combinaciones de expresiones!



# Conociendo GOBSTONES



- Características importantes
  - Pureza
    - Los comandos *solamente* producen efectos
    - Las expresiones *solamente* describen valores (las funciones NO PUEDEN alterar el tablero real)
  - Localidad en la comunicación
    - Solamente parámetros por valor y con alcance local a los procedimientos
    - ¡Los parámetros **no son** variables!
    - Las variables también son locales a los procedimientos (y su uso desalentado cuando no es imprescindible)



# Herramientas

- Existen varias herramientas que implementan GOBSTONES
  - los prototipos originales en HASKELL
    - interfaz mínima (en ASCII)
    - ninguna capacidad de editar tableros
    - baja performance
  - el PYGOBSTONES versión 0
    - interfaz de usuario básica
    - edición de tableros
    - mejoras notables de performance
  - el PYGOBSTONES versión 1 (HOY hace su DEBUT)
    - interfaz de usuario mejorada
    - edición de tableros mejorada
    - incorporación de características adicionales (interactividad, vestimentas, etc.)
- TODAS licenciadas bajo GPL





# Herramientas



- Conozcamos PYGOBSTONES v1.0
  - Herramienta desarrollada en PYTHON
  - Permite editar y ejecutar programas GOBSTONES
  - Maneja cuestiones de interfaz de usuario respecto a la ejecución
  - Veremos la herramienta en acción...





# Aplicación del enfoque

# Experiencias ya realizadas

- Se realizaron diversas experiencias con este enfoque
  - La materia *“Introducción a la Programación”* de la Tecnicatura en Programación Informática de la UNQ
    - ingresantes al sistema universitario (desde 2008)
  - Curso virtual de la carrera de Artes y Tecnología de la UNQ
    - universitarios formados en arte (en 2013)
  - Curso en la escuela Florentino Ameghino de Berazategui
    - 4to año de secundaria (desde 2012)
  - Proyecto PLATÓN con la DGE de la provincia de Bs.As.
    - 4to año de 16 escuelas secundarias técnicas (en 2013)
- En todos los casos los resultados han sido *excelentes*



# Material bibliográfico sobre GOBSTONES



- Escribimos un artículo sobre las ideas tras esta secuencia didáctica
  - “El nombre verdadero de la programación”  
[▶ Ver cita](#)
  - Se publicó en el SSI de las 41 JAIIO
- Estamos terminando de escribir un libro
  - **“Las bases conceptuales de la Programación.  
Una nueva forma de aprender a programar”**  
[▶ Ver cita](#)
  - Licenciado con Creative Commons
  - Veamos un adelanto del libro...

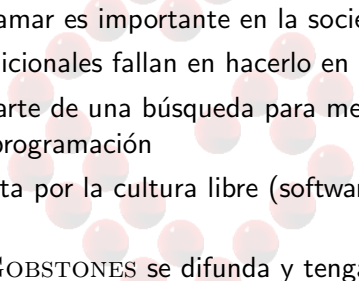





# Conclusiones



# Conclusiones

- 
- 
- Aprender a programar es importante en la sociedad moderna
  - Los métodos tradicionales fallan en hacerlo en escala
  - Este trabajo es parte de una búsqueda para mejorar el estado de la enseñanza de la programación
  - La libertad provista por la cultura libre (software libre, copyleft, etc.) es imprescindible
  - Esperamos que GOBSTONES se difunda y tengamos *feedback*



# Bibliografía



Pablo E. Martínez López, Eduardo A. Bonelli, and Federico A. Sawady O'Connor.  
El nombre verdadero de la programación. Una concepción de la enseñanza de la programación para la sociedad de la información.

In Gabriel Baum and Nora Sabelli, editors, *10mo Simposio sobre la Sociedad de la Información (SSI), dentro de las 41 Jornadas Argentinas de Informática (JAIIO)*, Facultad de Informática, UNLP, setiembre 2012.



Pablo E. Martínez López.

*Las bases conceptuales de la Programación. Una nueva forma de aprender a programar.*  
Publicación libre CC, octubre 2013.

URL: <http://www.gobstones.org/bibliografia/BasesConceptualesProg.pdf>.

# Bibliografía



Edsger W. Dijkstra.


On the cruelty of really teaching computing science (EWD-1036).

*E.W. Dijkstra Archive. Center for American History, 1989.*

URL: <http://www.cs.utexas.edu/users/EWD/ewd10xx/EWD1036.PDF>.







Ahora la demo...  
Y luego FIN

# Expansión de elementos del lenguaje



# Selección de contenidos

## (A) Elementos del lenguaje

- ① Categorías de elementos
  - Acciones, comandos, procedimientos
  - Valores, expresiones, funciones
- ② Formas combinación de elementos
  - secuencia (agregación)
  - alternativa
  - repetición
- ③ Herramientas para expresar abstracción – parametrización

◀ VOLVER



# Selección de contenidos

## (A) Elementos del lenguaje 1 – Categorías de elementos

- Por su naturaleza
  - Acciones** entidades que representan efectos
  - Valores** entidades que representan datos
- Por su forma de descripción
  - Comandos** descripciones de acciones
  - Expresiones** descripciones de valores
- Por su forma abstracta
  - Procedimientos** abstracciones de comandos
  - Funciones** abstracciones de expresiones

◀ VOLVER



# Selección de contenidos



## (A) Elementos del lenguaje 2 – Formas de combinación de elementos

|                                   | En comandos  | En valores                   |
|-----------------------------------|--|------------------------------|
| <b>Secuencia<br/>(agregación)</b> | secuenciación  | registros<br>{tuplas}        |
| <b>Alternativa</b>                | condicional (if-then-else)<br>indexada (switch)<br>{polimorfismo ad-hoc} | enumerativos<br>{variantes}  |
| <b>Repetición</b>                 | condicional (while)<br>indexada (repeat, foreach)<br>{recursión}         | listas<br>{tipos recursivos} |



# Selección de contenidos



- Para destacar...
  - Ambas categorías *describen* elementos
    - Las acciones aparecen como más concretas, por lo que empezamos por ellas
    - ¡Sin embargo, es importante remarcar su aspecto denotacional!
    - La falla en esto produce dificultades en la comprensión de los valores
  - Existe una dualidad entre valores y acciones, y sus formas
    - Las expresiones NO describen efectos
    - Los comandos NO describen valores
    - Por tanto, estas dos categorías DEBEN presentarse claramente separadas

◀ VOLVER



# Selección de contenidos



## (A) Elementos del lenguaje 3 – Herramientas para expresar abstracción

- Parametrización
  - mecanismo para capturar similitudes en el código, abstrayendo diferencias
  - herramienta abstracta por excelencia
  - consecuentemente, es difícil de transmitir en este nivel
  - sólo en su forma más básica: pasaje de parámetros por valor

◀ VOLVER





# Expansión de manejo del lenguaje





# Selección de contenidos

## (B) Manejo del lenguaje

### ① Manejo de sintaxis dura

- debemos inculcar la noción de las reglas estrictas de sintaxis
- notoriamente dificultoso en alumnos con poca formación matemática
- la sintaxis debe ser simple, con pocos elementos

### ② Buenas prácticas (cuestiones de estilo)

- buen uso de nombres de identificadores
- comentarios
- indentación

◀ VOLVER

# Selección de contenidos



## (B) Manejo del lenguaje 1 – Buenas prácticas

- Buen uso de nombres de identificadores
  - El nombre de un elemento (parámetro, variable, procedimiento) no es un detalle esencial en la ejecución
  - Sin embargo, un nombre bien elegido contribuye a inducir una visión denotacional adecuada
  - Desarrollar la capacidad de reconocer buenos nombres es fundamental

◀ VOLVER



# Selección de contenidos



## (B) Manejo del lenguaje 2 – Buenas prácticas

- Comentarios
  - Los comentarios no son tenidos en cuenta en la ejecución
  - Sin embargo, un comentario bien elegido facilita el reconocimiento de ideas abstractas, y evidencia su manejo
  - Desarrollar la capacidad de realizar comentarios adecuados es fundamental

◀ VOLVER



# Selección de contenidos



## (B) Manejo del lenguaje 3 – Buenas prácticas

- Indentación
  - La indentación no es tomada en cuenta en la ejecución
  - Sin embargo, una indentación adecuada destaca la estructura del código, y evidencia su comprensión
  - Desarrollar la capacidad de indentar adecuadamente es fundamental

◀ VOLVER





# Expansión de herramientas abstractas



# Selección de contenidos

## (C) Herramientas abstractas

- 1 División en subproblemas
- 2 Parametrización
- 3 Parcialidad y precondiciones
- 4 Esquemas de programas sencillos

◀ VOLVER



# Selección de contenidos



## (C) Herramientas abstractas 1 - División en subproblemas

- Descomposición de un problema en partes más sencillas
- Composición de las soluciones obtenidas
- Idea de *delegación* (sin explicitarla)
- No interesa la eficiencia, sino la visión abstracta de composicionalidad
- Es complicado detectarla al principio, porque los ejemplos son muy simples

◀ VOLVER



# Selección de contenidos



## (C) Herramientas abstractas 2 - Parametrización

- Uso de parámetros para lograr generalidad y disminuir la complejidad del código
- Es notorio el manejo de abstracción requerido para hacerlo bien
- Se relaciona íntimamente con la división en subtarear, ya que disminuye la cantidad de subtarear necesarias
- Si bien no trabajamos explícitamente en este aspecto, está presente

◀ VOLVER





# Selección de contenidos



## (C) Herramientas abstractas 3 - Parcialidad y precondiciones

- Parcialidad: Fallos en ejecución
- Precondiciones: Requisitos para evitar dichos fallos
- Ideas
  - Lenguaje de propósitos generales  $\Rightarrow$  situaciones anómalas
  - Se necesita una adecuada conceptualización
  - Se necesitan herramientas para manejar estas situaciones
  - Precondiciones vs. debugging



# Selección de contenidos



## (C) Herramientas abstractas 3 - Parcialidad y precondiciones (cont.)

- Precondiciones vs. debugging
- Debugging
  - Herramienta básicamente operacional
  - Induce al ensayo por prueba y error
- Precondiciones
  - Herramienta básicamente abstracta
  - Permite el razonamiento del programa en alto nivel
- El debugging es una mala elección pedagógica

◀ VOLVER



# Selección de contenidos



## (C) Herramientas abstractas 4 - Esquemas de programas sencillos

- Noción fundamental en programación
- Un esquema abstrae la estructura de muchos programas
- Esquema de *recorrido*
  - Esquema de procesamiento de secuencias de elementos
  - Basada en las ideas de invariantes de ciclo, folds (catamorfismos) y las “máquinas” de Scholl&Peyrin
  - Permite identificar los elementos necesarios para el correcto tratamiento de la secuencia



# Selección de contenidos



## (C) Herramientas abstractas 4 - Esquemas de programas sencillos (cont.)

- Un *recorrido* consta de
  - inicialización
  - condición de corte
  - procesamiento del elemento actual
  - paso al siguiente elemento
  - finalización



# Selección de contenidos

## (C) Herramientas abstractas 4 - Esquemas de programas sencillos (cont.)

```
procedure PintarTablero()
{
  // OBJETIVO: "Pinta" el tablero de rojo
  //           (colocando una bolita roja en cada celda)
  // PRECONDICION: ninguna, dado que es una operacion total
  // OBSERVACIONES: se estructura como recorrido sobre columnas
  IrALaPrimeraColumna()      // inicialización
  while(hayOtraColumna())    // condición de corte
  {
    ProcesarColumnaActual()   // procesamiento de un elemento
    PasarASiguienteColumna() // paso al siguiente
  }
  ProcesarColumnaActual()     // finalización
}
```



# Expansión de contenidos adicionales



# Selección de contenidos

- Contenidos adicionales

- ¿Modelado de información?

Lo mínimo imprescindible

- De hecho, se usa desde el principio
    - Pero se explicita solo al final y no se pone énfasis en ella
    - Los estudiantes prácticamente no modelan por sí mismos

- ¿Entrada/Salida?

No, puesto que la entrada/salida

- es eminentemente operacional
    - desvía el foco de atención de los aspectos denotacionales
    - requiere herramientas de abstracción complejas

◀ VOLVER



# Selección de contenidos

- Contenidos adicionales (cont.)

- ¿Programación funcional?

No, puesto que la programación funcional

- es de índole fundamentalmente abstracta
    - requiere herramientas muy complejas para reemplazar los efectos
    - precisa como base los elementos impartidos

- ¿Programación orientada a objetos?

No, puesto que la programación orientada a objetos

- tiene muchos más conceptos que transmitir
    - varios de esos conceptos no son transversales
    - precisa como base los elementos impartidos

◀ VOLVER





# Selección de contenidos

- Contenidos adicionales (cont.)

- ¿Formalización de las ideas?

No, puesto que la formalización de las ideas

- requiere niveles de madurez matemática y de abstracción superiores
    - no es imprescindible en un primer curso
    - solo es necesaria en una formación de índole específica

- ¿Estructuras de control más complejas/otras herramientas?  
(excepciones, unit testing, etc.)

No, puesto que estos conceptos

- no aportan a la comprensión de la programación básica
    - resuelven problemas avanzados que no son para un primer curso
    - requieren como base muchos de los conceptos impartidos

◀ VOLVER



# Selección de contenidos

- Contenidos adicionales (cont.)

- ¿Arreglos?

No, puesto que los arreglos

- inducen programas menos estructurales
    - son más concretos, más dependientes de un modelo de memoria
    - precisan la idea adicional de índice y su manejo
    - poseen tamaño fijo
    - orientan a la modificación *in place*, destructiva

- ¿Otras estructuras de datos?

No, puesto que las estructuras de datos

- requieren mayor nivel de abstracción
    - requieren como base los elementos impartidos

◀ VOLVER





# Expansión de problemáticas asociadas



# Problemáticas asociadas



- Errores posibles – Abuso de metáforas y analogías
  - Deben usarse de manera pertinente y crítica
  - No deben generalizarse ni extenderse más allá de su contexto
  - No deben reemplazar a la idea a ilustrar
- Errores posibles – Abuso de recursos gráficos
  - Limitan a futuro la manipulación de elementos abstractos
  - Desfavorecen la manipulación simbólica
  - Muchos elementos son difíciles de representar de manera gráfica
  - Resultan en una reducción del objeto de estudio



# Problemáticas asociadas

- Errores posibles – Demasiados aspectos operacionales
  - Limitan el pensamiento denotacional, abstracto
  - Inducen al uso de mecanismos de “prueba y error”
  - La sencillez inicial que permiten es una trampa
- Los atendemos de varias maneras
  - No ofrecemos mecanismos de debugging
  - Sólo se presenta el tablero final (nunca los intermedios)
  - Inducimos a pensar en subtareas desde el comienzo

[◀ VOLVER](#)