Computación I - CC1001

Clase 1

Victor Ramiro
vramiro@dcc.uchile.cl
Depto. Ciencias de la Computación

Agenda

- Introducción al curso
- Reglas del juego
- Mi primer programa

Introducción

Razonamiento algorítmico/lógico
Capacidad modelamiento/ abstracción
Habilidad gral resolver problemas

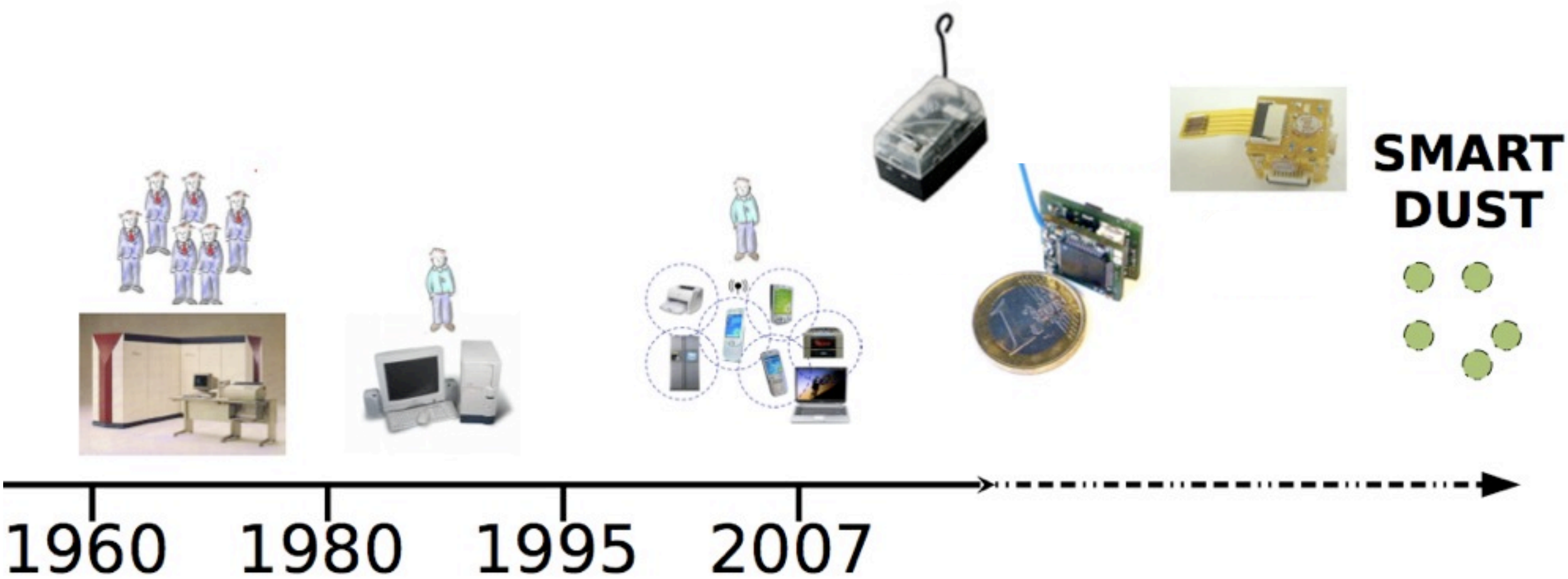
Computación

Problemas

Soluciones

Computador

Evolución…

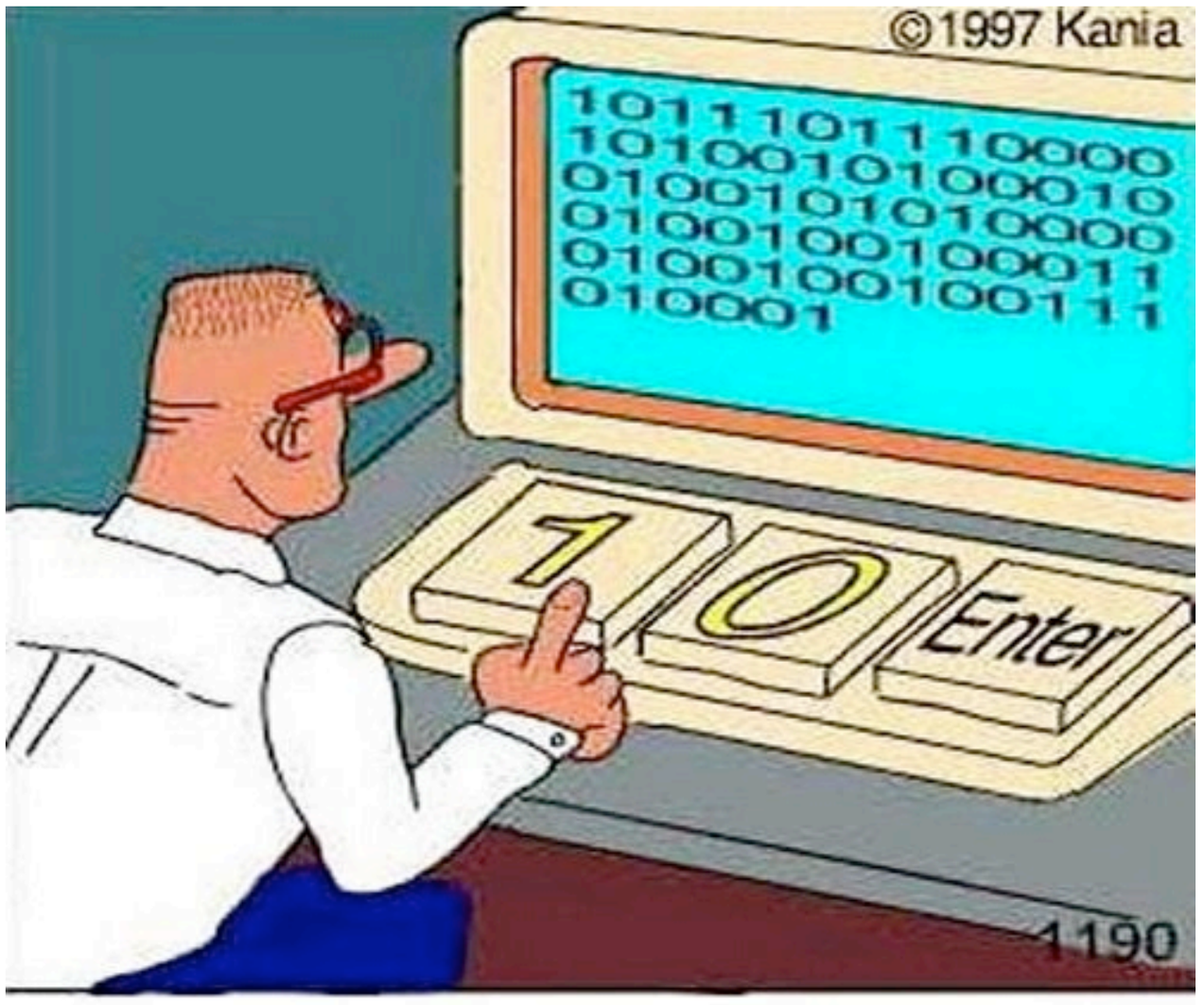


¿Por qué sirve un Lenguaje?

- Provee una forma para expresar algoritmos, de manera independiente de la máquina específica
- Provee una abstracción de alto nivel de una máquina compleja y difícil de programar

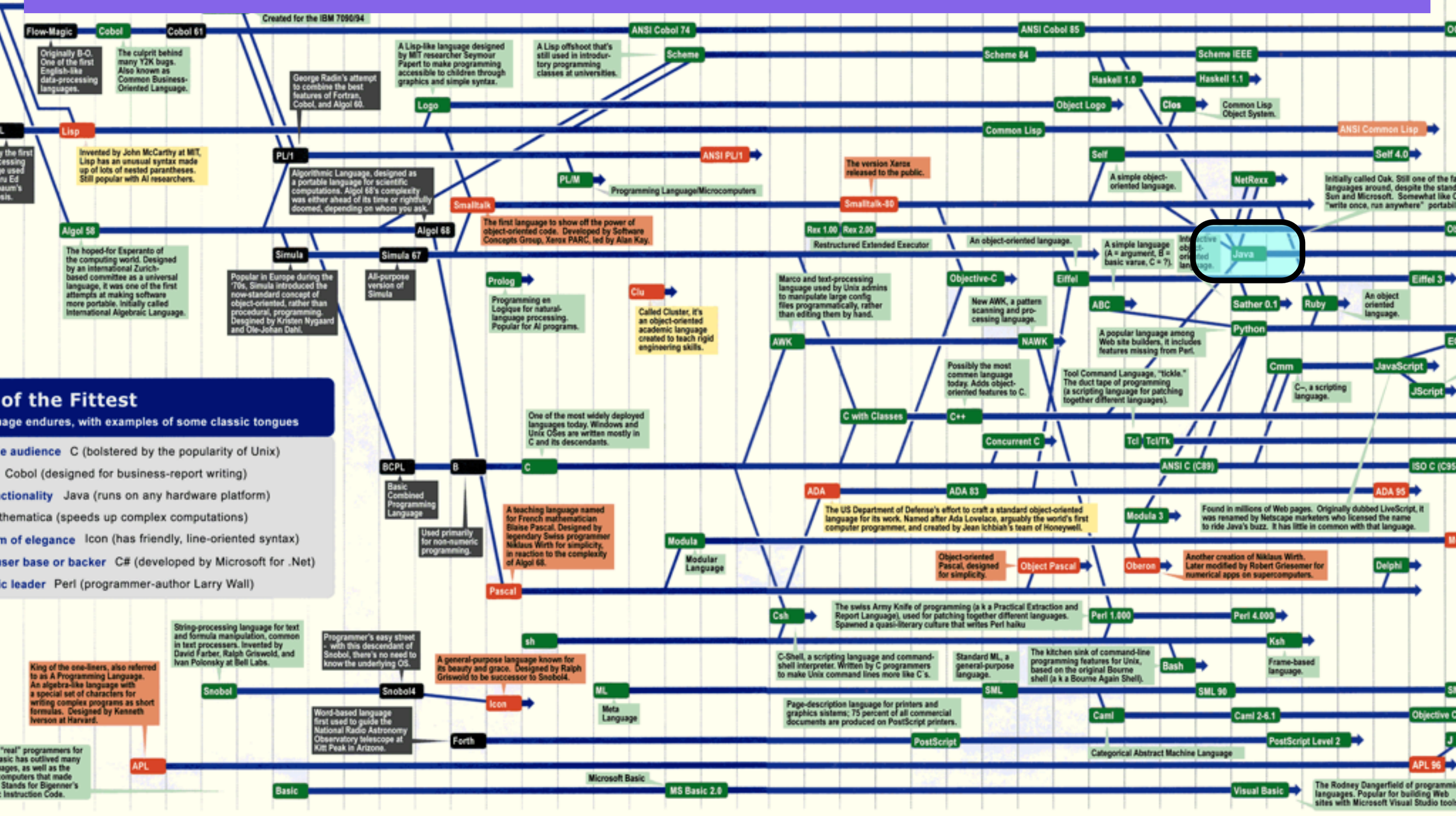
Programación

- Programación/ Codificación: Secuencia de instrucciones que especifica en forma precisa la manera de realizar una operación no trivial
- El programa debe estar escrito en un lenguaje que entienda el computador...
- ... y por el programador!



LOS VERDADEROS PROGRAMADORES PROGRAMAN EN BINARIO

Lenguajes


Created for the IBM 7090/94
Flow-Magic
Cobol
Cobol 61
ANSI Cobol 74
ANSI Cobol 85
Originally B-O. One of the first English-like data-processing languages.
The culprit behind many Y2K bugs. Also known as Common Business-Oriented Language.
A Lisp-like language designed by MIT researcher Seymour Papert to make programming accessible to children through graphics and simple syntax.
A Lisp offshoot that's still used in introdur-tory programming classes at universities.
Scheme
Scheme 84
Scheme IEEE
George Radin's attempt to combine the best features of Fortran, Cobol, and Algol 60.
Haskell 1.0
Haskell 1.1
Logo
Object Logo
Clos
Common Lisp Object System.
Lisp
Common Lisp
ANSI Common Lisp
Invented by John McCarthy at MIT, Lisp has an unusual syntax made up of lots of nested parantheses. Still popular with AI researchers.
PL/1
ANSI PL/1
Algorithmic Language, designed as a portable language for scientific computations. Algol 68's complexity was either ahead of its time or rightfully doomed, depending on whom you ask.
PL/M
Programming Language/Microcomputers
The version Xerox released to the public.
Self
Self 4.0
A simple object-oriented language.
NetRexx
Smalltalk
Smalltalk-80
The first language to show off the power of object-oriented code. Developed by Software Concepts Group, Xerox PARC, led by Alan Kay.
Algol 58
Algol 68
Rex 1.00
Rex 2.00
Restructured Extended Executor
An object-oriented language.
A simple language (A = argument, B = basic varue, C = ?).
Java
The hoped-for Esperanto of the computing world. Designed by an international Zurich-based committee as a universal language, it was one of the first attempts at making software more portable. Initially called International Algebraic Language.
Simula
Simula 67
Popular in Europe during the '70s, Simula introduced the now-standard concept of object-oriented, rather than procedural, programming. Desgined by Kristen Nygaard and Ole-Johan Dahl.
All-purpose version of Simula
Prolog
Programming en Logique for natural-language processing. Popular for AI programs.
Clu
Called Cluster, it's an object-oriented academic language created to teach rigid engineering skills.
Marco and text-processing language used by Unix admins to manipulate large config files programmatically, rather than editing them by hand.
Objective-C
Eiffel
Eiffel 3
New AWK, a pattern scanning and pro-cessing language.
ABC
Sather 0.1
Ruby
An object oriented language.
AWK
NAWK
Python
A popular language among Web site builders, it includes features missing from Perl.
Possibly the most commen language today. Adds object-oriented features to C.
Tool Command Language, "tickle." The duct tape of programming (a scripting language for patching together different languages).
Cmm
JavaScript
C--, a scripting language.
JScript
One of the most widely deployed languages today. Windows and Unix OSes are written mostly in C and its descendants.
C with Classes
C++
Concurrent C
Tcl
Tcl/Tk
BCPL
B
C
ANSI C (C89)
ISO C (C95
Basic Combined Programming Language
Used primarily for non-numeric programming.
A teaching language named for French mathematician Blaise Pascal. Designed by legendary Swiss programmer Niklaus Wirth for simplicity, in reaction to the complexity of Algol 68.
ADA
ADA 83
ADA 95
The US Department of Defense's effort to craft a standard object-oriented language for its work. Named after Ada Lovelace, arguably the world's first computer programmer, and created by Jean Ichbiah's team of Honeywell.
Modula 3
Found in millions of Web pages. Originally dubbed LiveScript, it was renamed by Netscape marketers who licensed the name to ride Java's buzz. It has little in common with that language.
Modula
Modular Language
Object-oriented Pascal, designed for simplicity.
Object Pascal
Oberon
Another creation of Niklaus Wirth. Later modified by Robert Griesemer for numerical apps on supercomputers.
Delphi
Pascal
Csh
The swiss Army Knife of programming (a k a Practical Extraction and Report Language), used for patching together different languages. Spawned a quasi-literary culture that writes Perl haiku
Perl 1.000
Perl 4.000
String-processing language for text and formula manipulation, common in text processers. Invented by David Farber, Ralph Griswold, and Ivan Polonsky at Bell Labs.
Programmer's easy street - with this descendant of Snobol, there's no need to know the underlying OS.
sh
Ksh
King of the one-liners, also referred to as A Programming Language. An algebra-like language with a special set of characters for writing complex programs as short formulas. Designed by Kenneth Iverson at Harvard.
A general-purpose language known for its beauty and grace. Designed by Ralph Griswold to be successor to Snobol4.
C-Shell, a scripting language and command-shell interpreter. Written by C programmers to make Unix command lines more like C's.
Standard ML, a general-purpose language.
The kitchen sink of command-line programming features for Unix, based on the original Bourne shell (a k a Bourne Again Shell).
Bash
Frame-based language.
Snobol
Snobol4
Icon
ML
Meta Language
SML
SML 90
Word-based language first used to guide the National Radio Astronomy Observatory telescope at Kitt Peak in Arizone.
Page-description language for printers and graphics sistems; 75 percent of all commercial documents are produced on PostScript printers.
Caml
Caml 2-6.1
Forth
PostScript
PostScript Level 2
Categorical Abstract Machine Language
APL
APL 96
Microsoft Basic
Basic
MS Basic 2.0
Visual Basic
of the Fittest
age endures, with examples of some classic tongues
e audience C (bolstered by the popularity of Cobol)
Cobol (designed for business-report writing)
ctionality Java (runs on any hardware platform)
thematica (speeds up complex computations)
n of elegance Icon (has friendly, line-oriented syntax)
ser base or backer C# (developed by Microsoft for .Net)
ic leader Perl (programmer-author Larry Wall)

Objetivos

- General
  - Resolver problemas
- Específico
  - Escribir programas que resuelvan problemas
- Largo plazo (propósitos)
  - Razonamientos algorítmico y lógico
  - Capacidades de abstracción y modelamiento
  - Habilidad general para resolver problemas

Contenidos

1. Fundamentos de programación
2. Programación orientada a objetos
3. Listas y tablas de valores
4. Computación numérica
5. Búsqueda y ordenamiento de información

Evaluaciones

- NF (nota final): 70%NC + 30%NT (NC,NT ≥ 4)
- NT (nota tareas): promedio 5 tareas (1 por cap) + tarea recuperativa
- NC (nota control): prom ponderado controles

control	contenidos	Semana	ponderación
1	1	4	20 %
2	2	8	20 %
3	3 y 4	12	20 %
examen	1 a 5	?	40 %

Ejercicios

- Se evalúan entre 1 y 4 (%=0,≤33, ≤66, ≤100)
- Se considera promedio de los 20 mejores (de 27)
- Si tiene menos de 20, NF=I + ejercicio recuperativo presencial
- Sirven para eximirse del examen de acuerdo a la sgte. tabla:

promedio c1,c2,c3	promedio ejercicios
5,5	cualquiera
5,4	≥ 1.5
5,3	≥ 2.0
5,2	≥ 2.5
5,1	≥ 3.0
5,0	≥ 3.5

Reglas del Juego

Reglas del Juego: Clases

Respeto

- Celulares apagados, mantener silencio, respeto a sus compañeros.

Estar informado

- La información dada en clases o publicada en u-cursos es oficial y es su responsabilidad saberla.
- Recuerde ingresar su e-mail en u-cursos

Plazo de reclamos

- Errores, problemas con sus notas se podrán reclamar hasta 20 días después de la publicación de la nota. Después no se aceptarán reclamos.

No copiar

- El trabajo en tareas, controles y ejercicios es individual y no se tolerará la copia.

Reglas del Juego: Tareas

- Trabajos con tiempo de entrega de entre 1 y 3 semanas
- Se avisará su publicación mediante u-cursos (sección tareas)
- Son individuales
- El trabajo se realiza fuera del horario de clases
- 10 UD = 10 horas/semana= 3 horas de cátedra + 2 horas de auxiliar + 5 horas semanales de trabajo
- Entrega es por u-cursos (sección tareas)

Reglas del Juego: Tareas

- **Respete las horas: Entrega hasta las XX:YY:ZZ hrs**
  - Entrega a las XX:YY:ZZ+1 segundo => nota 1.0
  - Note que la hora de u-cursos puede estar desfasada con respecto a la hora de su reloj, computador, etc
- **Entrega por u-cursos, sección tareas**
  - Entrega por mail u otra sección de u-cursos => nota 1.0
- **Entregar archivos requeridos**
  - Si se solicitan archivos .java, y se entregan los .class => nota 1.0
- **Entregar tarea que compile**
  - Si no compila => nota 1.0
- **Tarea individual**
  - Copia total o parcial de código, para ambos involucrados nota 1.0 + sanción
  - Proteja su tarea

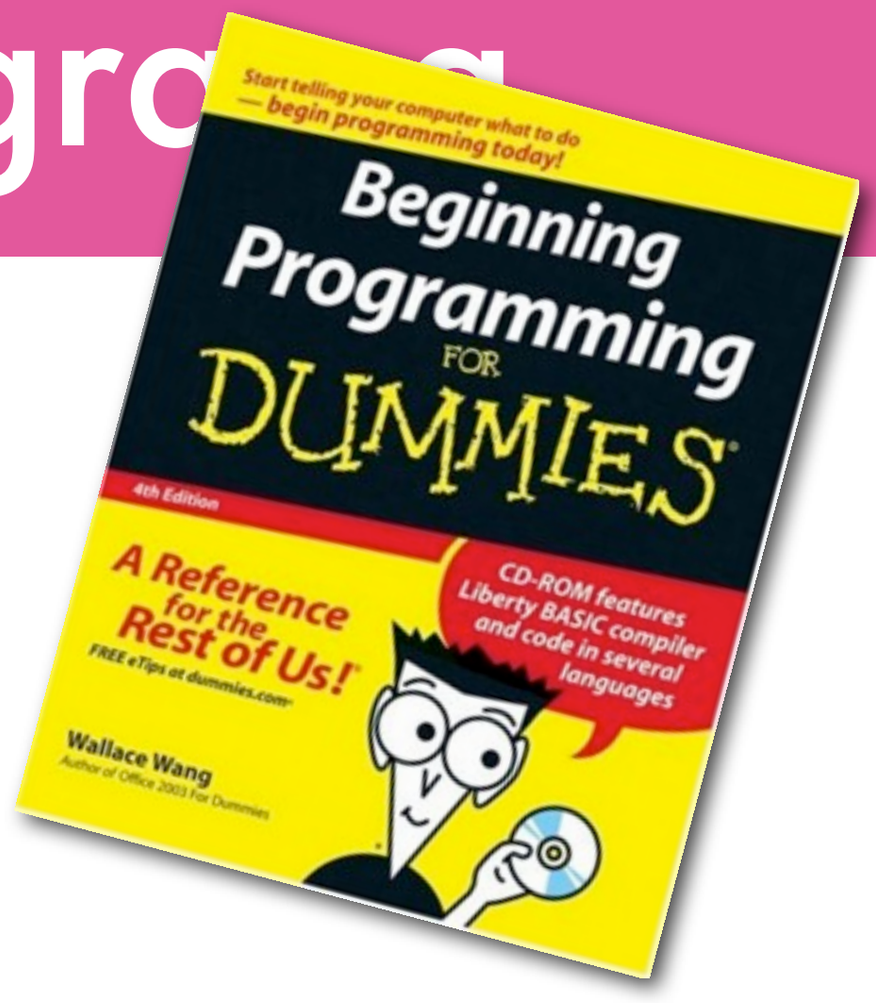
Frases Célebres

- "...Se me cortó la luz/internet y no pude enviar mi tarea..."
- "Me enfermé dos días antes del plazo y no pude hacer la tarea..."
- "Tuve mucho trabajo en cálculo/ álgebra/ física..."
- "Pero sólo le faltaba ; para que funcionara!"

Reglas del Juego: Tareas

- Hacer la tarea es una excelente forma de estudiar para los controles
- Cuerpo docente estará disponible para responder consultas sobre la tarea, vía:
  - Foro de u-cursos, E-mail, Hora de consultas
- Recomendaciones importantes:
  - Comenzar tareas con tiempo!
  - Entregar tareas con tiempo!
    - Planificar la entrega para 1 día antes
    - Se puede entregar cuantas veces quieran antes que termine el plazo, sólo contará la última entrega
    - Revisar que lo entregado sea realmente lo que quería entregar

Primer Progra



Problema

- Escribir un programa para que el computador calcule los porcentajes de hombres y mujeres de acuerdo al siguiente diálogo con una persona (usuario):

```
Número de Hombres: 649
Número de Mujeres: 125

% de Hombres=83.8
% de Mujeres=16.2
```

Algoritmo

- **Algoritmo:** Etapas (pasos, instrucciones) que debe realizar (ejecutar) el computador para resolver el problema. Ejemplo:

1. Escribir (mostrar) en la pantalla la frase ***"nº de hombres? "***
2. Leer (obtener) el n° ingresado por la persona usando el teclado
3. Escribir (mostrar) en la pantalla *"n° de mujeres? "*
4. Leer el n° ingresado por la persona
5. Calcular el % (p) de hombres c/r a suma de hombres (h) y mujeres (m) como: $$\frac{h}{h+m} \times 100$$
6. Escribir una línea con la frase *"% de hombres="* y el n° p
7. Escribir una línea con *"% de mujeres="* y el n° 100-p

Programa

- **Programa**: Algoritmo expresado en un lenguaje de programación
- Programa en lenguaje Java (instrucciones principales)

```
(1)U.print("nº de hombres? ");
(2)int h = U.readInt();
(3)U.print("nº de mujeres? ");
(4)int m = U.readInt();
(5)double p = 100.0*h/(h+m);
(6)U.print("% de hombres="); U.println(p);
(7)U.print("% de mujeres="); U.println(100-p);
```

Lenguaje Java: características generales

- En idioma inglés
  - palabras **`int, print, read,...`**
- Uso de notación matemática
  - expresiones: **`100.0*h/(h+m), p-100`**
  - funciones: **`print(...), readInt()`**

- Sintaxis (gramática, escritura) reglamentada
  - instrucciones terminan con `;`
  - palabras con significados especiales (**`int, double, print`**)
  - uso de minúsculas (salvo casos especiales)
- Semántica (interpretación) precisa (sin ambigüedades)
  - instrucciones con significado/efecto bien definido

Instrucción (1)

```
U.print("n° de hombres? ");
```

- Escribir (mostrar) en la pantalla frase
  "n° de hombres? "

- **Sintaxis:**

  ```
  U.print("...");
  ```

- **Semántica**: Escribe parámetro (argumento) de la función U.print en la pantalla del computador

Instrucción (2)

```
int h = U.readInt();
```

- Lee un número entero desde el teclado y lo guarda en la variable h
- **Equivalencia**

```
int h; //declaración
h = U.readInt(); //lectura
```

- **Nota:**

```
//frase
```

- Es un comentario que contiene observaciones de utilidad para el programador, pero que es ignorado por el computador al ejecutar las instrucciones del programa.

Instrucción (2.1)

```
int h;
```

Semántica: declara (define) h como una variable entera

- ¿variable?
  - representación simbólica de un valor (número)
  - representa una ubicación (celda) en la memoria del computador
  - capacidad: **un** valor/número (del tipo indicado)
  - puede variar (cambiar) su valor

Sintaxis: tipo nombre;

- tipo int: nº entero de hasta 9 dígitos. Ejs:123, -4, 2147483647 (mayor)
- tipo double: nº real de hasta 15 dígitos de precisión. Ejs: 100.0, 0.0001
- nombre (identificador): letra seguida de letras, dígitos o _

Instrucción (2.2)

```
h = U.readInt();
```

Lee un nº entero y lo guarda en la variable h

Sintaxis: `variable = U.readTipo();`

`readTipo(): readInt() o readDouble()`

Semántica:

1. método (función) `U.readInt()`
   1.1. espera que el usuario ingrese un número (dígitos y enter)
   1.2. lee (obtiene) el número
   1.3. entrega el número como resultado
2. número se guarda en (asigna a) la variable h

Instrucción (5)

`double p=100.0*h/(h+m);`

calcula 100.0*h/(h+m) y guarda el resultado en la variable p

equivalencia:

```
double p; //declaración
p=100.0*h/(h+m); //asignación
```

Instrucción de asignación:

Sintaxis: `variable=expresión;`

Semántica:

1. evaluar expresión (calcular resultado)
2. asignar (guardar) resultado a (en) la variable

Expresion

- **Sintaxis**
  - se escribe en una línea (hacia el lado) y no en varios niveles
  - componentes: variables, constantes, operadores, funciones
- **Semántica**
  - se calcula el resultado de la expresión de acuerdo a la siguiente proridad de los operadores:

1. operadores unarios: +,-
2. operadores “multiplicativos”: *,/
3. operadores “aditivos”: +,-

Ej: `-a+b*c` es `(-a)+(b*c)`, similar a álgebra

Expresión: Semántica

- a **igual prioridad**, se evalúa de izq. a der.

  Ej: `a*b/c es (a*b)/c`

- **paréntesis** modifican o confirman orden de evaluación

  Ej: `(a+b)*c` modifica, `a+(b*c)` confirma

- **¿tipo del resultado?**
  - tipo común. Ej: `1/2` es `0` (entero), `1.0/2.0` es `0.5` (real)
  - real si operandos son de tipos distintos. Ej: `1.0/2` es `0.5`
  - **Nota**: ¿resultado de `h/(h+m)*100.0`? ¿por qué?

Imprimir una línea

`U.println(p);`

escribe el valor de la variable p y "salta" a la sgte línea

Sintaxis: `U.println(expresión)`

Semántica:

1. evaluar expresión (ejs: p, 100-p)
2. escribir resultado
3. posicionar el cursor al comienzo de la sgte línea en la pantalla

Notas:

Expresión puede ser una frase. Ej: `U.println("chao");`

U.print también admite expresiones. Ej: `U.print(100-p);`

Ejercicio

- Escribir las instrucciones que establezcan el diálogo indicado en el siguiente ejemplo:

Calcular velocidad de un móvil

distancia (metros)? **100.0**

tiempo (segundos)? **9.78**

velocidad=36.81 km/hora

Solución

```
U.println("Calcular velocidad de un móvil");

U.print("distancia(metros)?");
double d=U.readDouble();

U.print("tiempo(segundos)?");
double t=U.readDouble();

U.print("velocidad=");
//alternativa: 3.6*d/t
U.print((d/1000)/(t/3600));
U.println(" km/hora");
```

Solución 2. Usando abreviaturas

```
U.println("Calcular velocidad de un móvil");

double d = U.readDouble("distancia(metros)?");

double t=U.readDouble("tiempo(segundos)?");

U.println("velocidad=" + 3.6*d/t +" km/hora");
```

Explicaciones…

`double d=U.readDouble("distancia(metros)?");`

es equivalente a:

```
U.print("distancia(metros)?");
double d=U.readDouble();
```

`U.println("velocidad=" + 3.6*d/t + " km/hora");`

es equivalente a:

```
U.print("velocidad=");
U.print(3.6*d/t);
U.println(" km/hora");
```

+: operador de concatenación o añadidura

Solución 3. Usando

```
U.println("Calcular velocidad de un móvil");

double d=U.readDouble("distancia(metros)?"),
       t=U.readDouble("tiempo(segundos)?");

U.println("velocidad=" + 3.6*d/t +" km/hora");
```

Explicaciones…

Sintaxis:

tipo variable1=exp1, variable2=exp2, …;

Semántica:

tipo variable1=exp1;

tipo variable2=exp2;

…

Programa Completo

```
import java.io.*;
class Programa{
  static public void main(String[]args)
  throws IOException{
    … //instrucciones
  }
}
```

Notas

- Todo programa debe estar contenido en una clase: `class Nombre{…}` y grabarse en archivo `Nombre.java`
- Burocracia necesaria porque Java no se diseñó para principiantes
- Lectura y escritura es complicada, pero se simplifica usando la clase U (clase de la U con funciones para leer y escribir frases y números). Disponible en u-cursos.