Introduction to Computing

Immersive lab exploration of computing science and programming

Daniel Andrés Pinto Alvarado & Enzo Alda

Introducción a la Computación

Exploración inmersiva de laboratorio sobre ciencias de la computación y programación.

Daniel Andrés Pinto Alvarado & Enzo Alda

Overview

- Why are we here?
- Why this course matters?
- Requisites and Methodology
- Lab logistics
- Demo and Closing Remarks

Visión General

- ¿Por qué estamos aquí?
- ¿Por qué es importante este curso?
- Requisitos y Metodología
- Logística del Laboratorio
- Demo y Cierre Final

Why are you here?

- Because it is a graduation requirement?
- To learn to program and develop your careers?

- **Other reasons**
  - To acquire knowledge about computing science
  - To improve your ability to think and solve complex problems
  - To expand your understanding of the world

- Expanding your understanding of the world ➔
  - Identify current trends and anticipate future ones
  - Make something people want! Invent!
  - Improve people's lives
  - Create your legacy

¿Por qué están aquí?

- ¿Porque es un requisito de graduación?
- ¿Para aprender a programar y desarrollar sus carreras?

- **Oreas razones**
  - Para adquirir conocimientos sobre la ciencia de la computación
  - Para mejorar su capacidad de pensar y resolver problemas complejos
  - Para ampliar tu comprensión del mundo

- Ampliando tu comprensión del mundo ➔
  - Identificar las tendencias actuales y anticipar las futuras
  - ¡Crear algo que la gente quiere! ¡Inventar!
  - Mejorar la vida de las personas
  - Crear su legado

Why this course matters

- Because computing knowledge matters
  - Information technology is everywhere
  - The science and engineering of computing continues to evolve
  - Computing science is a key achievement of mankind

Por qué es importante este curso

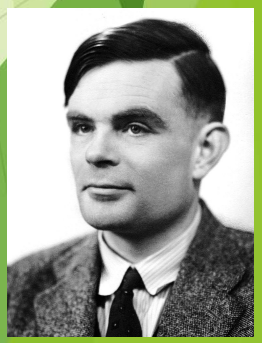
- Porque el conocimiento de la computación es importante
  - La tecnología de la información está en todas partes
  - La ciencia y la ingeniería de la computación continúan evolucionando
  - La ciencia de la computación es un logro clave de la humanidad.

Computing: A Human Quest for Knowledge

- Prof. Baralt will give you a deep dive into the concepts, ideas, discoveries ..
  - .. and the giants behind Computing Science
- Computing Science was born from Math, long before computers existed
  - More precisely, it was a product of the Math crisis in the early XX century
  - Computing science is a deductive science, closely tied to logic and math
  - In English it is usually (and improperly) known as “Computer Science”
  - But the science is about “computing” .. not “computers”!
- Alan Turing and Alonzo Church are considered the fathers of computing science ..
  - They stood on the shoulders of other giants, like Gödel .. Hilbert .. Euler .. Euclides (300 BC) !





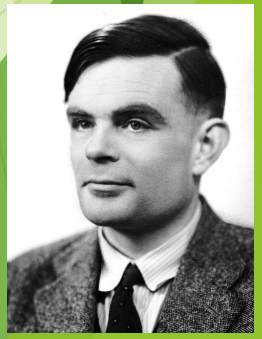



Computación: búsqueda humana del conocimiento

- El profesor Baralt les brindará una exploración de los conceptos, ideas, descubrimientos ..
  - ..y los gigantes detrás de la Ciencia de la Computación
- La computación nació de las matemáticas, antes de que existieran las computadoras
  - Más precisamente, fue un producto de la crisis de las matemáticas a principios del siglo XX
  - La computación es una ciencia deductiva, estrechamente ligada a la lógica y las matemáticas
  - En inglés se le conoce habitualmente (e incorrectamente) como “Computer Science” ..
  - .. pero La ciencia tiene que ver con la “computación” .. ¡No las “computadoras”!
- Alan Turing y Alonzo Church son considerados los padres de la ciencia de la computación ..
  - .. sobre los hombros de otros gigantes, como Gödel .. Hilbert .. Euler .. ¡Euclides (300 a. C.)!





Digital Literacy -> Computational Thinking -> Programming

- **Digital Literacy**: the ability to use information technology.

- **Computational Thinking**: the thought process involved in modeling problems and derive solutions that can be defined algorithmically, composing functional transformations and describing computational steps. When computational thinking is expressed in ways a computer can execute, it is called «Programming».

- **Programming**: the process of expressing computational thinking in a language a computer can execute.

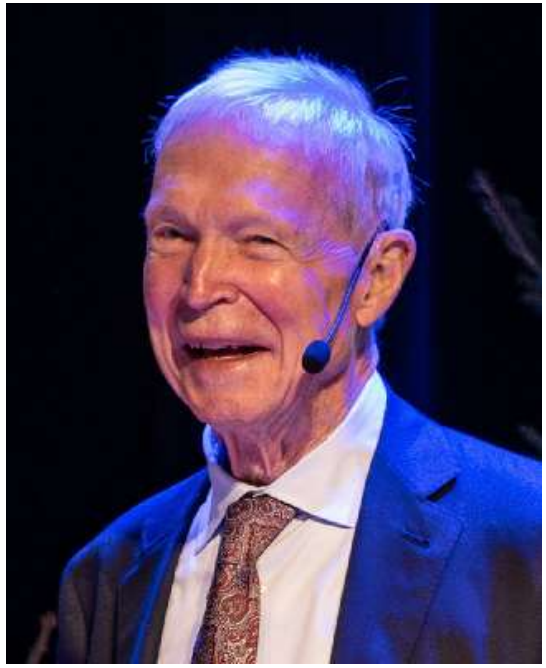
Alfabetización digital -> Pensamiento computacional -> Programación

- **Alfabetización digital**: la capacidad de utilizar la tecnología de la información.

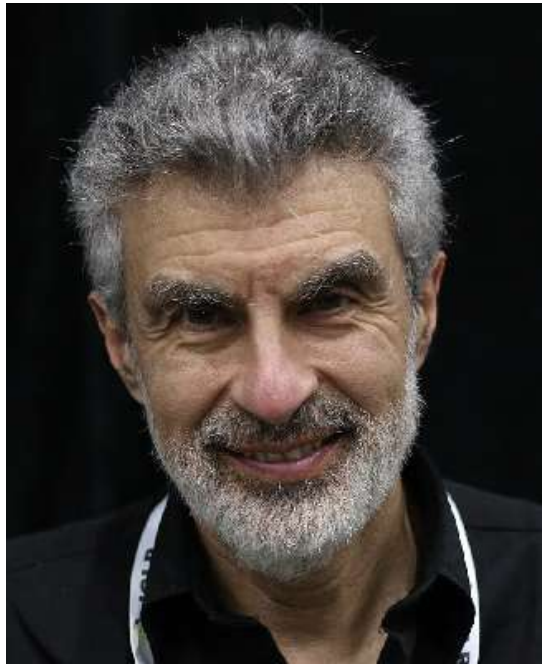
- **Pensamiento Computacional**: el proceso de pensamiento implicado en modelar problemas y derivar soluciones que pueden definirse algorítmicamente, componiendo transformaciones funcionales y describiendo los pasos computacionales. Cuando el pensamiento computacional se expresa de maneras que una computadora puede ejecutar, se denomina «programación».

- **Programación**: el proceso de expresar el pensamiento computacional en un lenguaje que una computadora puede ejecutar.

2024 Physics Nobel Prize
Premio Nobel de Física 2024



2018 Turing Award
Premio Turing 2018



nature

Explore content | About the journal | Publish with us | Subscribe

NEWS | 08 October 2024

Physics Nobel scooped by machine-learning pioneers

John Hopfield and Geoffrey Hinton pioneered computational methods that enabled the development of neural networks.

By Elizabeth Gibney & Davide Castelvecchi

nature portfolio

View all journals | Search | Log in

nature > collection

Collection | 08 October 2024

Nobel Prize in Physics 2024

The Nobel Prize in Physics 2024 has been awarded to John J. Hopfield and Geoffrey E. Hinton "for foundational discoveries and inventions that enable machine learning with artificial neural networks". In recognition of this award, Nature Portfolio presents a collection of research, review and opinion articles that celebrates the direct contributions by the awardees and the advances they have inspired.

Editors

Yann Sweeney

The Elephant in the Room

- AI in education: research and results
- Ignoring AI is not an option
- But using AI as a crutch will make you dumb
  - Don’t harm yourselves by cheating!
- Open problem:
  - Are we augmenting or replacing
- “Est modus in rebus”

El Elefante en la Habitación

- IA en educación: investigación y resultados
- Ignorar la IA es una mala idea
- Pero usar la IA como muleta los hará tontos
  - ¡No se hagan trampa a Uds. mismos!
- Problema abierto:
  - Estamos ampliando o sustituyendo
- “Est modus in rebus”
  - Es bueno el cilantro .. ¡Pero no tanto!

Requisites and Methodology

- **Requisites**
  - Elementary school arithmetic knowledge is required
  - Previous knowledge of programing and computing theory is ***not*** required
  - **Taking the course seriously!**

- **Methodology**
  - Cognitive theory: learn by doing!
  - Multiple notations – syntax ⇔ semantics, C/C++, Zilly/Lilly, ..
  - Self-taught but not self-paced:
    - facilitators provide lab assistance as described next

Requisitos y Metodología

- **Requisitos**
  - Se requieren conocimientos de aritmética a nivel de la escuela primaria
  - ***No*** se requieren conocimientos previos de programación y teoría de la computación
  - **¡Tomarse el curso en serio!**

- **Metodología**
  - Teoría cognitiva: ¡aprenden haciendo!
  - Notaciones múltiples: sintaxis ⇔ semántica, C/C++, Zilly/Lilly, ..
  - Autodidacta pero no a su propio ritmo:
    - los facilitadores brindan asistencia de laboratorio como se describe a continuación

Lab Logistics

- Presented by Daniel Andrés Pinto Alvarado

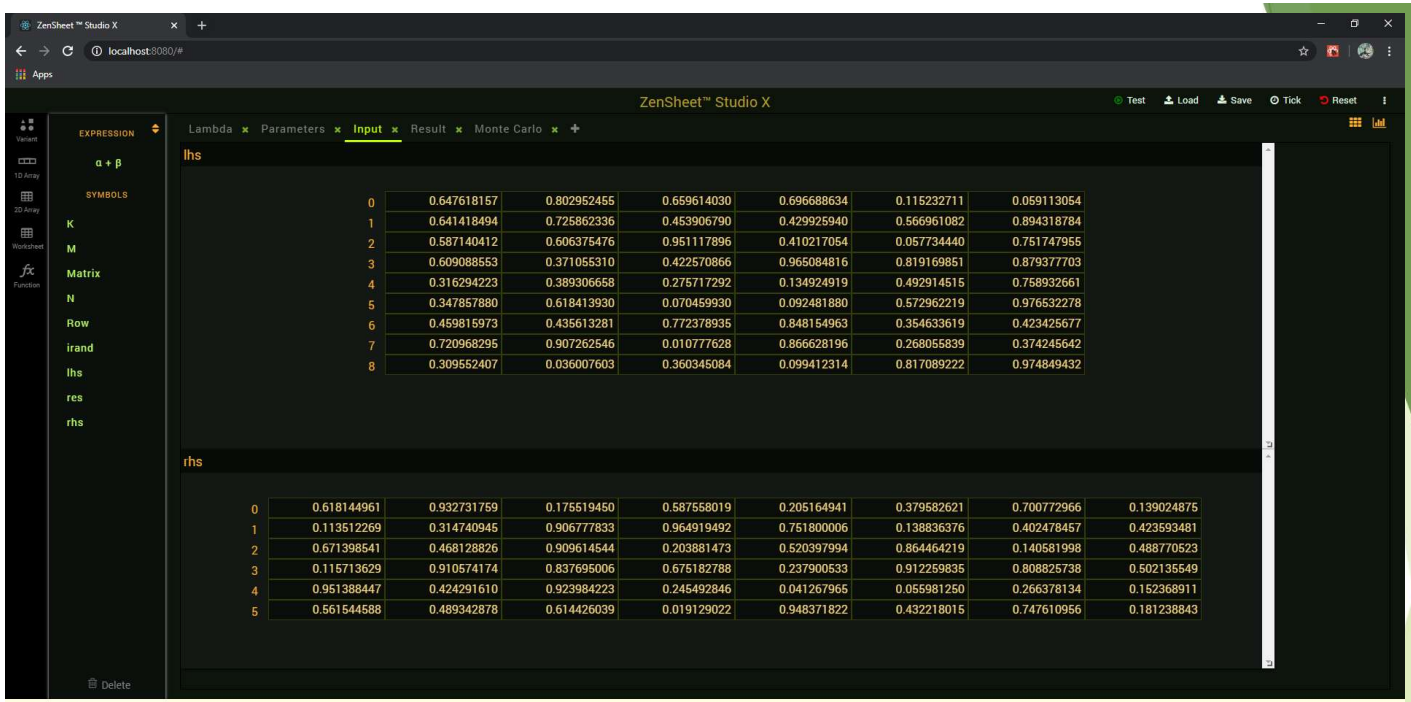
Logística del Laboratorio

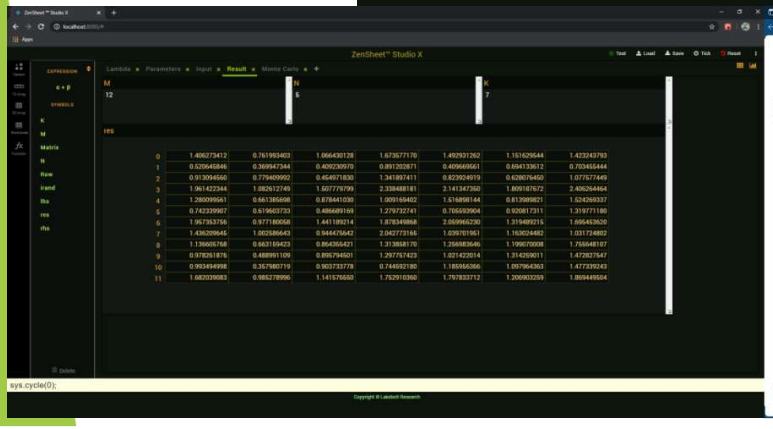
- Presentada por Daniel Andrés Pinto Alvarado

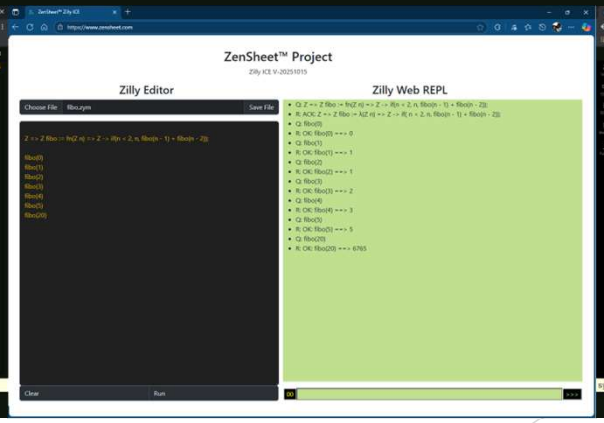
Logística del Laboratorio

- Los estudiantes deben configurar su infraestructura del laboratorio
  - ¡No se retrasen en hacerlo! Sigan las instrucciones y recomendaciones
  - Hicimos lo posible para simplificarla: ¡aun más que en los trimestres pasados!
    - Actitud: "No soy ingeniero de soporte: ¡Por ti seré, Venezuela!"
  - El profesor Pinto Alvarado les va a enviar tutoriales (videos) para ayudarlos
- El material del curso es autocontenido en gran medida
  - Con referencias a fuentes de la Web, donde se más conveniente o necesario
  - Esto refuerza el espíritu de aprender por su cuenta lo más posible
  - Pero no van a sentirse "abandonados"
    - Los profesores del laboratorio son "***facilitadores***": van a facilitar su aprendizaje
    - Estrenando las "Happy Hours": una o dos horas a la semana! La participación es opcional.
- La evaluación (40% de la nota) va a consistir de varias modalidades
  - quizzes, exámenes, y quizás otras
- Envíen preguntas y quejas (por ejemplo: reportes de bugs) por email a
  - daniel.andres.pinto@gmail.com y enzo@ieee.org

Demo


ZenSheet™ Studio X
localhost:8080/#
Apps
ZenSheet™ Studio X
Test
Load
Save
Tick
Reset
EXPRESSION
α + β
SYMBOLS
K
M
Matrix
N
Row
irand
lhs
res
rhs
Delete
Lambda
Parameters
Input
Result
Monte Carlo
lhs
0 0.647618157 0.802952455 0.659614030 0.696688634 0.115232711 0.059113054
1 0.641418494 0.725862336 0.453906790 0.429925940 0.566961082 0.894318784
2 0.587140412 0.606375476 0.951117896 0.410217054 0.057734440 0.751747955
3 0.609088553 0.371055310 0.422570866 0.965084816 0.819169851 0.879377703
4 0.316294223 0.389306658 0.275717292 0.134924919 0.492914515 0.758932661
5 0.347857880 0.618413930 0.070459930 0.092481880 0.572962219 0.976532278
6 0.459815973 0.435613281 0.772378935 0.848154963 0.354633619 0.423425677
7 0.720968295 0.907262546 0.010777628 0.866628196 0.268055839 0.374245642
8 0.309552407 0.036007603 0.360345084 0.099412314 0.817089222 0.974849432
rhs
0 0.618144961 0.932731759 0.175519450 0.587558019 0.205164941 0.379582621 0.700772966 0.139024875
1 0.113512269 0.314740945 0.906777833 0.964919492 0.751800006 0.138836376 0.402478457 0.423593481
2 0.671398541 0.468128826 0.909614544 0.203881473 0.520397994 0.864464219 0.140581998 0.488770523
3 0.115713629 0.910574174 0.837695006 0.675182788 0.237900533 0.912259835 0.808825738 0.502135549
4 0.951388447 0.424291610 0.923984223 0.245492846 0.041267965 0.055981250 0.266378134 0.152368911
5 0.561544588 0.489342878 0.614426039 0.019129022 0.948371822 0.432218015 0.747610956 0.181238843
sys.cycle(0);
Copyright © Lakebolt Research




ZenSheet™ Project
Zilly Editor
Zilly Web REPL

Evaluating expressions

- As you already know, Zilly computes with integer numbers
- There are two important functions we can use in expressions: ***lt*** and ***sub***
- Their semantics (i.e., their meaning, what they compute) is as follows:

  lt(n)(k) ➔ ***k < n*** (specifically, 1 if *k < n* and 0 otherwise)

  sub(n)(k) ➔ ***k - n*** (i.e., subtract n from k)
- Turns out, those two operations are all you need to implement integer arithmetic!

- There are a few other predefined functions: ***random*** is useful for testing:

  random*(n)* ➔ (0 if *n* = 0, otherwise, a random integer between *0* and *n-1 included*)

- Some predefined functions are system functions, prefixed by *sys*, like sys.reset

  Executing sys.reset() erases all user-defined variables.

Evaluando expresiones

- Zilly realiza cálculos con números enteros.
- Hay dos funciones importantes que podemos usar en expresiones: ***lt*** y ***sub***
- Su semántica (es decir, su significado, lo que calculan) es la siguiente:

  lt(n)(k) ➔ ***k < n*** (específicamente, 1 si *k < n* y 0 en caso contrario).

  sub(n)(k) ➔ ***k - n*** (es decir, resta n de k).
- Esas dos operaciones son todo lo que necesitan para implementar la aritmética de enteros

- Hay otras funciones predefinidas: ***random*** es útil para realizar pruebas:

  random(n) ➔ (0 si n = 0; de lo contrario, un entero aleatorio entre 0 y n-1 incluido)

- Algunas funciones predefinidas son funciones de sistema, prefijadas con sys, como sys.reset

  Al ejecutar sys.reset() se borran todas las variables definidas por el usuario

Conditional Expressions

- As is the case in life, complex computations require making decisions

- For instance, if we need to choose the smaller of two numbers, x and y, then …
  - In C, C++, Java, C#, JavaScript, and other C-inspired languages we write: (x < y ? x : y)
  - In Python we write: (x if x < y else y)
  - In LISP we write: if ((< x y) x y)
  - In MS-Excel, Google Sheets, and most spreadsheets we write: if(x < y, x, y)

- **All** the syntactic "sugars" (**forms**) above require exactly three expressions:
  - A Boolean expression, known as the *condition*, which is evaluated first
  - An expression (known as *consequent*), evaluated if and only if the *condition* is true
  - An expression (known as *alternative*), evaluated if and only if the *condition* is false
  - (different sugars, same semantics … get used to focus on semantics)

Expresiones Condicionales

- Al igual que en la vida real, los cálculos complejos requieren tomar decisiones

- Por ejemplo, si necesitamos escoger el menor de dos números, x e y, entonces …
  - En C, C++, Java, C#, JavaScript y otros lenguajes inspirados en C, escribimos: (x < y ? x : y)
  - En Python escribimos: (x if x < y else y)
  - En LISP escribimos: if ((< x y) x y)
  - En MS-Excel, Google Sheets, y hoja de cálculo, escribimos: if(x < y, x, y)

- **Todos** los "azúcares" sintácticos (**formas**) anteriores requieren exactamente tres expresiones:
  - Una expresión booleana, conocida como *condición*, que se evalúa primero
  - Una expresión (*consecuente*), que se evalúa si y solo si la *condición* es verdadera
  - Una expresión (*alternativa*), que se evalúa si y solo si la *condición* es falsa

Closing Remarks

Cierre Final