UNIVERSIDAD ADOLFO IBÁÑEZ

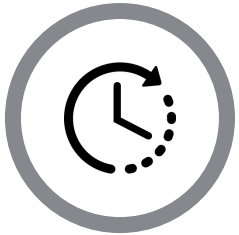
2022/Primer Semestre

Lenguajes y Paradigmas de Programación

Ma. Loreto Arriagada
loreto.arriagada.v@edu.uai.cl

INGENIERÍA

Acuerdos de la clase



RESPETO POR EL TIEMPO
DE INICIO Y TÉRMINO

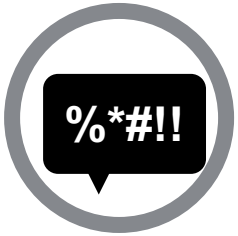
ALCEN LA MANO
PARA HABLAR

TRABAJEN SOLO
CON LO DE LA CLASE

EL FEEDBACK ES
BIENVENIDO

PARTICIPEN EN
LA CLASE

USE CONSCIENTEMENTE



EXPRESEN SUS IDEAS
LIBREMENTE

Introducción al Curso

- Programa del curso
- Metodología
- Evaluaciones

Programa del Curso

1. Paradigmas de Programación (8-9 semanas)
   a. Historia
   b. Paradigma procedural (C - Python)
   c. Paradigma orientado al objeto (Java - Python)
   d. Paradigma funcional (Clojure - Python)
2. Aplicaciones de los diferentes paradigmas
3. Proyecto

Programa del Curso

1. Paradigmas de Programación
2. Aplicaciones de los diferentes paradigmas (2 semanas)
   a. En cada clase (4) veremos un problema que se resuelve mejor con un paradigma
3. Proyecto (4 semanas)

Metodología

- 2 horas de cátedra semanales
  - Discusión en clase
  - Tareas para pensar todas las semanas
- Proyecto Semestral
  - Problema de la vida real
  - Deberán seleccionar el mejor paradigma para resolver cada una de las partes

Evaluaciones

- No hay pruebas formales → al final de cada paradigma haremos una aplicación (A1, A2, A3)
- Tareas y discusión en Clase (T)
- Proyecto (P)
- Examen Final

Evaluaciones

- Ponderación:

  NPE= a*(A1+A2+A3)/3+ (1-a) *(T+P)/2

  - a es el factor de importancia de pruebas
    - 0.5 si promedio de aplicaciones es mayor o igual a 4.0 (sin aproximación)
    - 0.9 en caso contrario

Evaluaciones

- Nota Final NF= NE*0.4+NPE*0.6
  - Sólo si NPE es mayor o igual a 3.0 (sin aproximación) se puede rendir examen
  - Examen es reprobatorio (si nota es menor a 3)
  - Se eximen si NPE es mayor o igual a 5.0 y todas las notas son sobre 4

¿Preguntas?

Necesidades básicas para el curso

Cuatro cosas mínimas

- Editor
- Compilador
- Control de Versiones
- Seguimiento de Errores

Michael Lopp (Autor de Being Geek: The Software Developer's Career Handbook)

Entonces, ¿qué necesita traer?

- Su laptop (mucho trabajo práctico)
- Como Editor pueden usar cualquiera que tenga soporte para múltiples lenguajes
- Como Compilador usaremos la **versión 3** de Python, la **versión 11** de Java
- Como controlador de versiones usaremos GIT con GITHub como servidor.

Entonces, ¿qué necesita traer?

Para el proyecto usaremos GraalVM

Pregunta…. ¿Qué es una VM?

GraalVM

GraalVM is a universal virtual machine for running applications written in JavaScript, Python, Ruby, R, JVM-based languages like Java, Scala, Groovy, Kotlin, Clojure, and LLVM-based languages such as C and C++.

Tarea 1

- Instalar Python 3
- Instalar Java 11 (JDK, no JRE)
- Instalar GraalVM (https://www.graalvm.org)
- Instalar GIT
- Crear una cuenta en GITHub
- Instalar client GITHub
- Instalar IDE de su preferencia (Visual Studio Code es bueno si no tienen alguno)

Control de Versiones

NOTA:

- No enseñaré control de versiones en esta clase (eso lo verán en programación profesional), pero les haré llegar le haré llegar los comandos básicos.

¿Preguntas?

Estamos listos para comenzar

Si no sabes de donde vienes no sabes adonde vas

Historia de la Programación y Paradigmas

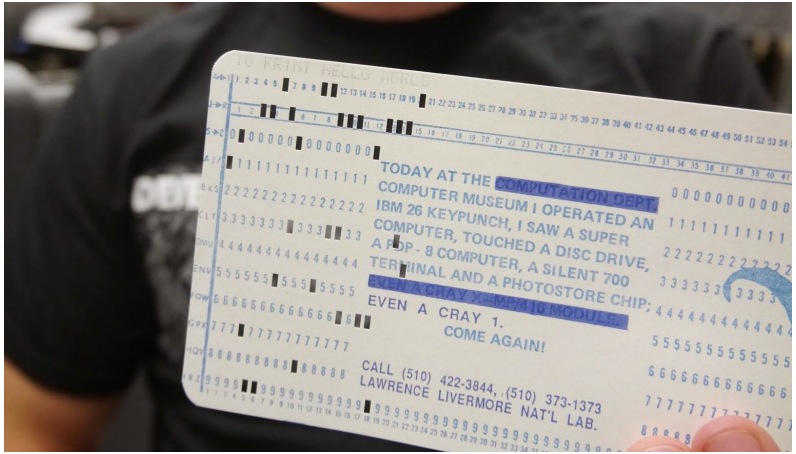
Instrucciones

Forma de decirle al computador lo que queremos hacer

- Noten que las instrucciones es la especificación de lo que queremos → diseño
- Una vez ejecutado el control lo dejamos a la máquina

¿Cómo se dan las instrucciones?

- Punch cards
- Assembly



https://www.flickr.com/photos/mhawksey/15874024102

```
section     .text
global      _start              ;must be declared for linker (ld)

_start:                         ;tell linker entry point

    mov     edx,len             ;message length
    mov     ecx,msg             ;message to write
    mov     ebx,1               ;file descriptor (stdout)
    mov     eax,4               ;system call number (sys_write)
    int     0x80                ;call kernel

    mov     eax,1               ;system call number (sys_exit)
    int     0x80                ;call kernel

section     .data

    msg     db  'Hello, world!',0xa         ;our dear string
    len     equ $ - msg                     ;length of our dear
string
```

¿Qué opinan de esa forma?

Lenguajes de Programación (cómo los conocemos hoy)

FORTRAN

Algunos de los creadores de Fortran en 1982. Fuente: IBM

```
program hello
print *, 'Hola Mundo'
end program hello
```

Imperative Programming

Primer Paradigma

```
program hello
character::name*25
print *, 'Ingresa tu nombre:'
read *, name
print *, 'Hola', name
end program hello
```

```
program hello
character::name*25,apellido*25
integer largo
print *, 'Ingresa tu nombre y apellido:'
read *, name, apellido
print *, 'Hola ', name, apellido
largo = len(trim(name))
print *, largo
end program hello
```

¿Y si queremos preguntar el nombre a 5 personas?

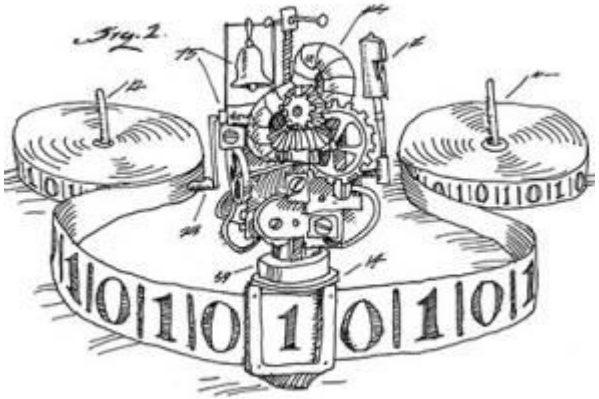
```
program hello
character::name*25
print *, 'Ingresa tu nombre:'
read *, name
print *, 'Hola', name
print *, 'Ingresa tu nombre:'
read *, name
print *, 'Hola', name
print *, 'Ingresa tu nombre:'
read *, name
print *, 'Hola', name
print *, 'Ingresa tu nombre:'
read *, name
print *, 'Hola', name
print *, 'Ingresa tu nombre:'
read *, name
print *, 'Hola', name
end program hello
```

```
program hello
integer i
character::name*25
i=0
10 if i.LT.5 Then
print *, 'Ingresa tu nombre:'
  read *, name
  print *, 'Hola', name
  i = i + 1
  GO TO 10
  end if
  end program hello
```

```
program hello
integer i
character::name*25
i=0
do while (i.LT.5)
   print *, 'Ingresa tu nombre:'
   read *, name
   print *, 'Hola ', name
   i = i + 1
end do
end program
```

¿GO TO? ¿Por qué?

Máquina de Turing



Tarea 2

Edsger W. Dijkstra publicó una carta titulada "Go To Statement considered Harmful" en 1968

Leer la versión disponible en http://david.tribble.com/text/goto.html y extraer conceptos de interés para discutir la próxima clase

Debemos preferir …

… otras estructuras de control de flujo

- Condiciones
- Ciclos
- Subrutinas
- …

Structured Programming

Cómo lo vemos en programación I

C: la madre de todos los lenguajes estructurados modernos

```
#include <stdio.h>

int main(int argc, char** argv) {
    printf("Hola Mundo\n");
    return 0;
}
```

Lenguaje de Programación más utilizado en la actualidad

Python
(1991)

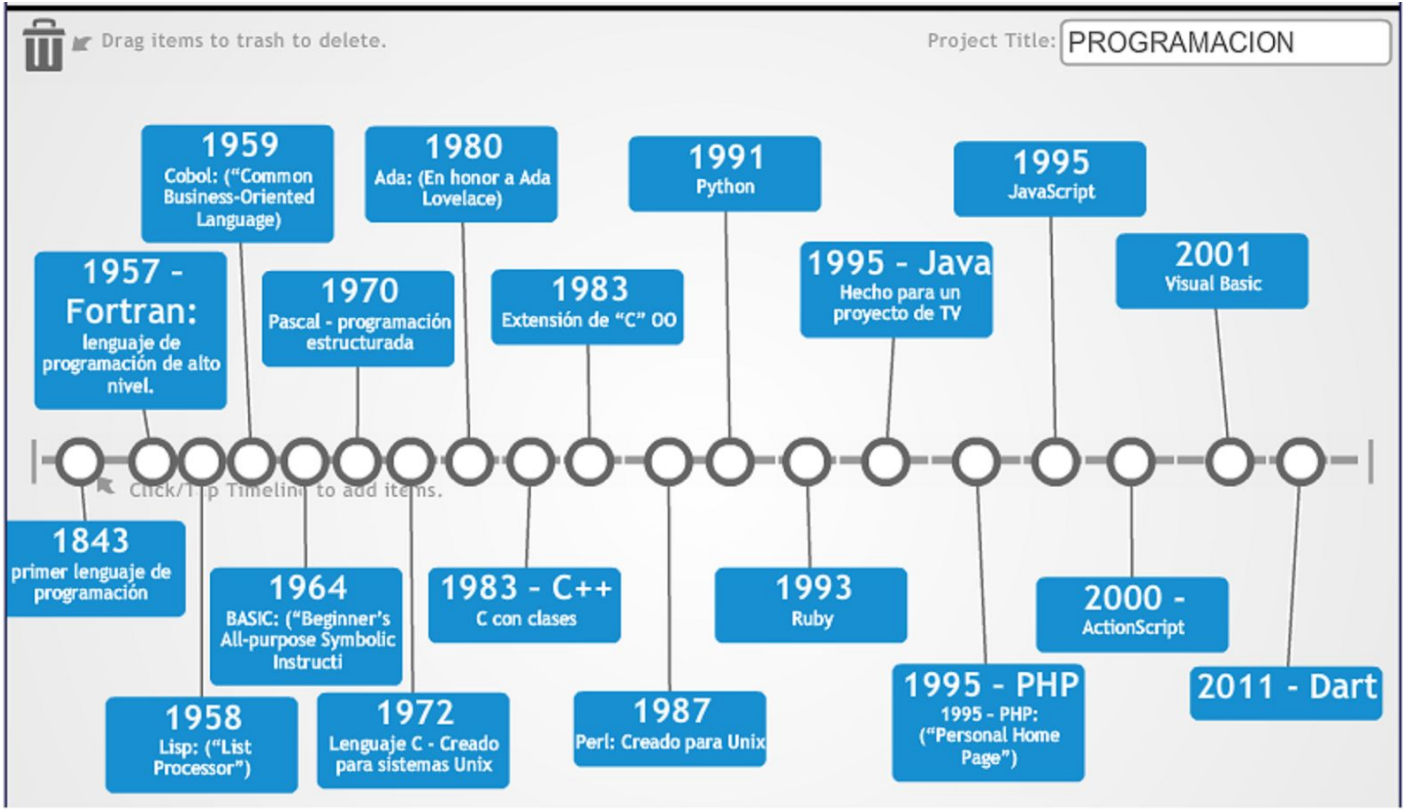
```
print("Hola Mundo\n")
```

Esto no se ve como programación estructurada, sino que imperativa pura… cambiemoslo :)

```
def main():
    print("Hola Mundo\n")

if __name__ == "__main__":
    main()
```

Esto le dice a Python que si el programa se ejecuta desde este archivo, entonces llame a la función main()



http://pcmasnet.blogspot.com/2016/04/linea-de-tiempo-de-la-programacion-de.html

Todos somos listas

LISP
(1958)

```
(write-line "Hola Mundo")
```

Functional Programming

Resuelva

(/ (+ (* 30 2) 5) 4)

```
(print
  (reduce #'+
    (subseq
      (sort (list 17 4 23 9) #'<)
    0 3)
  )
)
```

¿Python?

El ejemplo anterior no cambia en Python, pero la verdad es que Python si soporta programación funcional

```
from functools import reduce

palabras = ["Hola", " ", "Mundo","\n"]

oracion = reduce(lambda a, b: a + b,
palabras)
print(oracion)
```

Orientación al Objeto

SIMULA67

SmallTalk

'Hello World' printNl

Object Oriented Programming

Pero…

… el ejemplo anterior no se ve cómo orientado al objeto, sino como programación estructurada simple

¿Y esto?

```
1 to:10 do:[:i |
  Transcript show: (i asString).
].
```

¿Python?

```
class Main(object):
    def helloWorld():
        print("Hello World\n")

Main.helloWorld()
```

Respetar las reglas

ProLog

```
main :- write('Hola Mundo'), nl, halt.
```

Logic Programming

```
male(james1).
male(charles1).
male(charles2).
male(james2).
male(george1).
female(catherine).
female(elizabeth).
female(sophia).
```

```
parent(charles1, james1).
parent(elizabeth, james1).
parent(charles2, charles1).
parent(catherine, charles1).
parent(james2, charles1).
parent(sophia, elizabeth).
parent(george1, sophia).
```

```
?- parent(charles1, george1).
?- parent(james2, X)
```

¿Qué vimos?

- Programa del Curso
- Necesidades básicas
- Un poco de historia y paradigmas de programación

La próxima semana

- Comenzaremos con un repaso de programación procedural/estructurada
  - C es el ejemplo de este paradigma
  - Como vimos, python es ejemplo de todo :)

¿Preguntas?

Recuerden la tarea 1