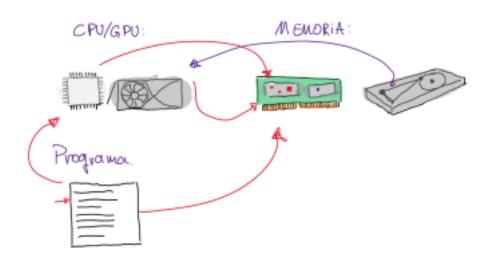
# **LENGUAJE IMP** (Lenguaje imperativo minimal)

### Intro

Un MODELO FUNCIONAL describe qué son (definiciones) las cosas (expresiones) y las reglas para transformarlas / simplificarlas

Un MODELO IMPERATIVO le indica a la computadora qué hacer (modificar memoria) Un PROGRAMA IMPERATIVO es una lista de instrucciones que modifican datos en una memoria (Java, C++, Python, ..)

Funcionamiento de las computadoras:



### MODELO DE PROGRAMACIÓN IMP

<u>Datos</u> => lo que queremos utilizar, guardar y leer <u>Memoria</u> => Lugar donde se guarda las cosas <u>Instrucciones</u> => Forma de interacción de los datos

**Datos** (no hay tipos)

=> Constructores (árboles)

**Memoria** => colección o lista de variables (datos fuertes) con datos asociados a las mismas

=> Operaciones: 1) leer variable y de memoria M: My => y

2) escribir/sobreescribir el valor de una variable: M <+ (y,z), agrega/cambia el valor de la variable y, por el valor de z

#### Instrucciones

=> Asignación (única instrucción que modifica la memoria)

y:=S[x] (la variable y, guarda el valor de S[x]

- 1) Buscamos el valor de x en memoria
- 2) Agregamos un sucesor a dicho valor
- 3) Guardamos el nuevo valor en y, en memoria
- => Estructuras de control (definen qué asignaciones vamos a ejecutar)

Asignación Múltiple/Simultánea => x1,..xN := e1,..,en

$$[(x,S[O])] => x,y := S[x],x => [(x,S[S[O]]), (y,S[O])]$$
 Memoria inicial Programa Memoria resultante

Intercambio de valores de variables:

$$[(x,1),(y,2)] => x,y := y,x => [(x,2),(y,1)]$$

Estructuras de control (definen el flujo del programas)

- 1) Secuencia p1; p2 (se ejecuta primero p1 y luego p2)
- 2) Selección case exp of ramas
- 3) Repetición while exp do ramas

## **Ejemplos**

```
Negación Booleana (b := !b)

Case b of {False \rightarrow b := True | True \rightarrow b := False}

Conjunción Booleana (b := b && c)

Case b of {False \rightarrow b := False | True \rightarrow b := c}

Disyuncion Booleana (b := b || c)

Case b of {True \rightarrow b := True | False \rightarrow b := c}

Par: (b := par(n))

b := true;

while n do {S[x] \rightarrow n:=x;
 (b:=!b)}
```

## **SINTAXIS**

```
Valores v := C vs
Expresiones e:= x | C es
Programas/Instrucciones:
                        Asignación p:= xs := es
                         Secuencia p1; p2
                         Selección case e of {rs}
                         Repetición while e do {rs}
rama := C xs \rightarrow p
p programa
type rama = (C, xs, p)
Memoria: lista de pares variable - valor [(variable, valor)]
Id variable
Operaciones: Actualizar ( \leftarrow ) :: Mem \rightarrow [(Id,valor)] \rightarrow Mem
                Buscar (@) :: Mem \rightarrow Id \rightarrow valor
Vacía: ε @ x = Null []
Última actualización: (m \leftarrow x vs) @ x = {
                                         v si(x,v) \subseteq x vs
                                         m @ x en otro caso
                                }
Evaluación de expresiones:
Variable: y := x;
Constructor: S[...] \rightarrow S[...]
Ejecución de un programa: Dado un programa y una memoria inicial, producir la memoria
resultante de ejecutar el programa
m > p > m'
Casos:
p := xs := es
        | p1 ; p2
        case e of rs
        while e do
rama := C xs \rightarrow p
```