Inspira Crea Transforma

UNIVERSIDAD EAFIT

ESCUELA DE CIENCIAS APLICADAS E INGENIERÍA
INGENIERÍA AGRONÓMICA

ST0299 PENSAMIENTO COMPUTACIONAL II
Implementación de Sistemas para la Agricultura Digital y Pensamiento Algorítmico

Yomin Jaramillo M

Docente | Escuela de Ciencias Aplicadas e Ingeniería | Ingeniería Agronómica

Correo: yejaramilm@eafit.edu.co

Contenido

1. Simuladores
2. Lenguaje de Programación Arduino
3. Lenguaje Python
4. Pensamiento Algorítmico

1. Simuladores

Simuladores Arduino

Los simuladores Arduino son la herramienta que permiten probar diseños de circuitos complejos sin pérdidas de tiempo o daños en los componentes.

En la mayoría de estos simuladores para Arduino se puede realizar una depuración del código línea por línea, lo que permite identificar exactamente dónde está el error.

Hay diferentes tipos de simuladores que podemos utilizar, su uso dependerá del proyecto que se esté realizando y de los conocimientos.

Tipos de Simuladores

Simuladores de Arduino Online	• Simulador en línea (TinkerCad Circuits), permite diseñar proyectos com Arduino de forma simple, si bien es limitado al Arduino UNO se pueden diseñar y simular múltiples soluciones.
Simuladores de Arduino para PC	• Plataforma de automatización de diseño electrónico, simulación de circuitos y modelado de PCBs (Proteus Design Suite) • Software de simulación desarrollado por la empresa Autodesk (Autodesk Eagle)
Simulador de Circuitos Electrónicos	• Simuladores de protoboards y programas para dibujar esquemas eléctricos pero que no funcionan como simuladores Arduino. Algunos pueden incluir componentes para Arduino, pero no simulan el funcionamiento ni permiten compilar el código fuente.

Fuente: https://proyectosconarduino.com/curso/simuladores-arduino/

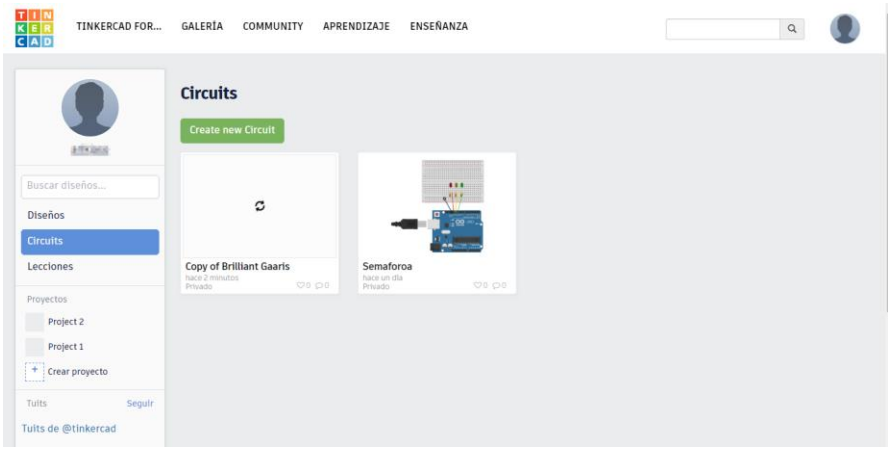
TinkerCAD - Simulador

Tinkercad es una herramienta online ofrecida por Autodesk.

Se utiliza de forma gratuita y sólo requiere crearse una cuenta de usuario.

Entre sus utilidades, probablemente la más conocida es la de diseñar piezas en 3D, sin embargo, ofrece también una posibilidad realmente interesante y es la de montar, programar y simular circuitos con Arduino.

Para ello, deberemos crearnos una cuenta de usuario y acceder. Seleccionando la opción “Circuits” podremos empezar a crear nuestros circuitos clicando sobre “Create new Circuit”:



Fuente: https://codigo21.educacion.navarra.es/recursos/tinkercad-simulador/

Ejercicio realizado

Usando TinkerCAD realice los siguientes simulaciones:

- Prender y apagar un LED de forma intermitente cada 2 segundos.
- Simular un semáforo en el que el rojo esté 2 segundos y el verde 3 segundos, para pasar de rojo a verde prende 1 segundo el amarillo y viceversa
- Simular una alarma con un circuito que tome la temperatura, mientras esta esté menor a 28 grados un LED en verde, si sube mas de estos grados un LED en rojo.

2. Lenguaje Programación Arduino

Lenguaje Nativo Arduino

Para programar un Arduino, el lenguaje estándar es C++, aunque es posible programarlo en otros lenguajes.

No es un C++ puro sino que es una adaptación que proveniente de avr-libc que provee de una librería de C de alta calidad para usar con GCC en los microcontroladores AVR de Atmel y muchas funciones específicas para los MCU AVR de Atmel.

El IDE para programar Arduino trae librerías de forma automática y no es necesario declararlas expresamente.

Características del lenguaje

El lenguaje de programación de Arduino se puede dividir en tres grandes partes:

- Estructura,
- Valores (variables y constantes),
- Funciones.

Estructuras	Valores	Funciones
• Sketc • Control • Operadores	• Constantes • Tipos de Datos • Conversión	• I/O Digitales • I/O Analógicas • Matemáticas

Referencia: https://www.arduino.cc/reference/es/

IDE Programación Arduino C++

El software de código abierto Arduino (IDE) facilita la escritura de código y la carga en la placa.

Este software se puede utilizar con cualquier placa Arduino.

https://www.arduino.cc/en/software

Ejercicio Resuelto

Valide que ha instalado el IDE de Arduino

Valide que los programas que realizó en TinkerCAD fueron bajados y llevados al editor que tiene instalado

Revise que ha probado los programas y funcionan

Realice nuevas versiones de estos en este IDE

3. Lenguaje Python

Lenguaje Python

Desde su introducción por Guido van Rossum en 1991, Python se ha convertido en uno de los lenguajes de programación de alto nivel de propósito general más utilizados, y es compatible con una de las mayores comunidades de desarrolladores de código abierto.

Python es un lenguaje de programación de código abierto que incluye muchas bibliotecas de soporte. Estas bibliotecas son la mejor característica de Python, lo que la convierte en una de las plataformas más extensibles.

Python es un lenguaje de programación dinámico, y utiliza un intérprete para ejecutar código en tiempo de ejecución en lugar de usar un compilador para compilar y crear códigos de bytes ejecutables.

Trabajar con Python para Arduino

Python lo podemos usar en varios escenarios:

- Usando la Biblioteca serial de Python llamada pySerial, con la cual debemos conectar con cable la tarjeta y el equipo PC a través del puerto serial.
- De forma desconectada utilizando una tarjeta adicional de Wi-Fi que permita la recepción de datos vía internet
- Usando interconexiones con antenas de corto, mediano y largo alcance

Iniciaremos usando la conexión de puerto serial y en los avances del curso usaremos otros mecanismos de integración como la conexión Wi-Fi y antenas

4. Pensamiento Algorítmico

¿Qué es el Pensamiento Computacional?

"*Proceso mental utilizado para **formular problemas** y sus **soluciones** de forma que las soluciones se **representan** en una forma que puede ser llevada a cabo por un agente de **procesamiento de información***".

Wing

Habilidades

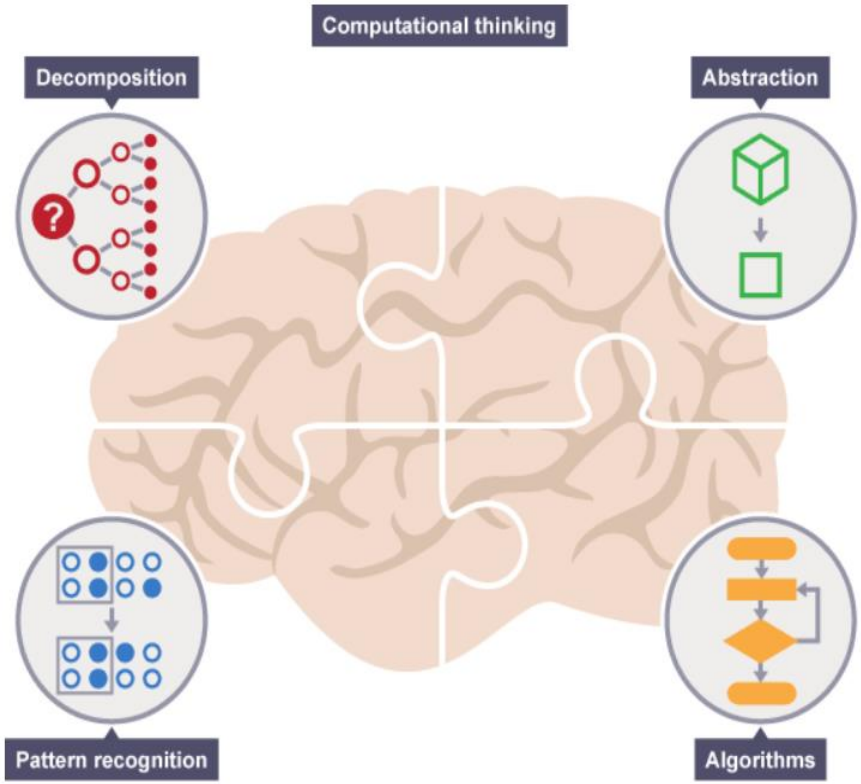
Reformular problemas	Descomponer problemas	Recopilar datos	Analizar datos
Representar datos	Abstraer	Codificar	Depurar
Analizar posibles soluciones	Simular	Generalizar	Automatizar

¿Para qué estudiar pensamiento computacional?

Formular problemas de forma que sus soluciones pueden ser representadas como secuencias de **instrucciones y algoritmos.**

Reconocer aspectos de la informática en el mundo que nos rodea, y aplicar herramientas y técnicas de la informática para comprender y razonar sobre los **sistemas y procesos** tanto naturales como artificiales.

Pilares del Pensamiento Computacional



Nombre: Introduction to Computational Thinking
Autor: BBC
Fuente: https://www.bbc.co.uk/education/guides/zp92mp3/revision

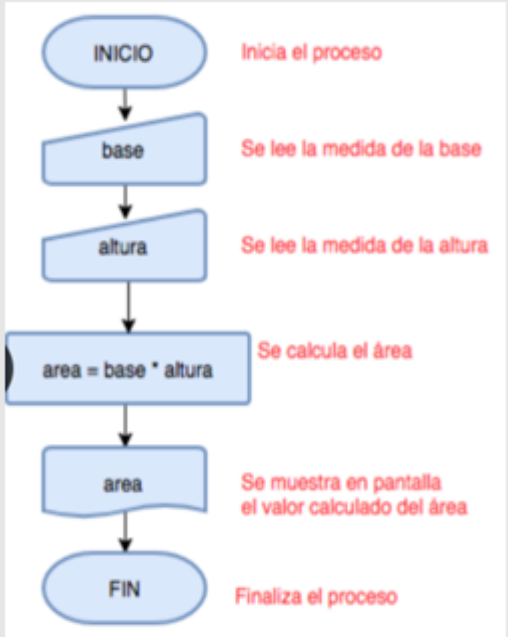
Tomar un problema complejo y descomponerlo en pequeños problemas más manejables.

Cada uno de estos problemas más pequeños se puede analizar individualmente, teniendo en cuenta cómo se han resuelto problemas similares anteriormente y centrándose solo en **los detalles importantes**, ignorando la información irrelevante.

Después se pueden diseñar pasos simples para resolver cada uno de los problemas más pequeños.

Problema - Área

Calcula el área de un rectángulo sabiendo la medida de su base y de su altura.

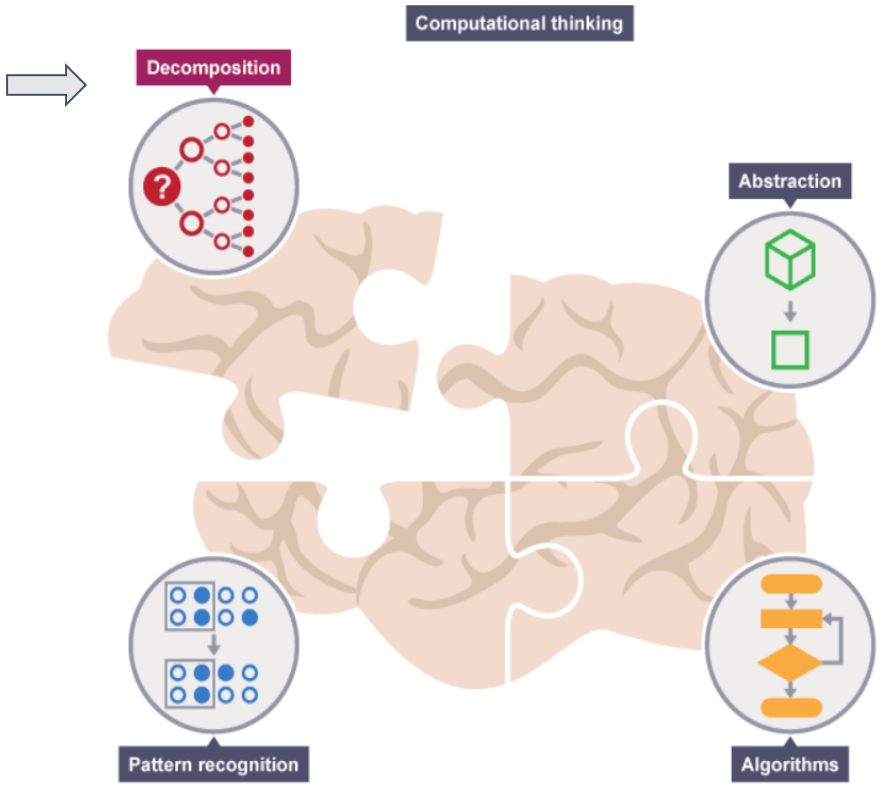


Descomposición

Dividir un sistema en partes pequeñas más fáciles de entender, programar y mantener.

Examinar y resolver, o diseñar individualmente las partes pequeñas.

Si un problema no se descompone, es más difícil de resolver.

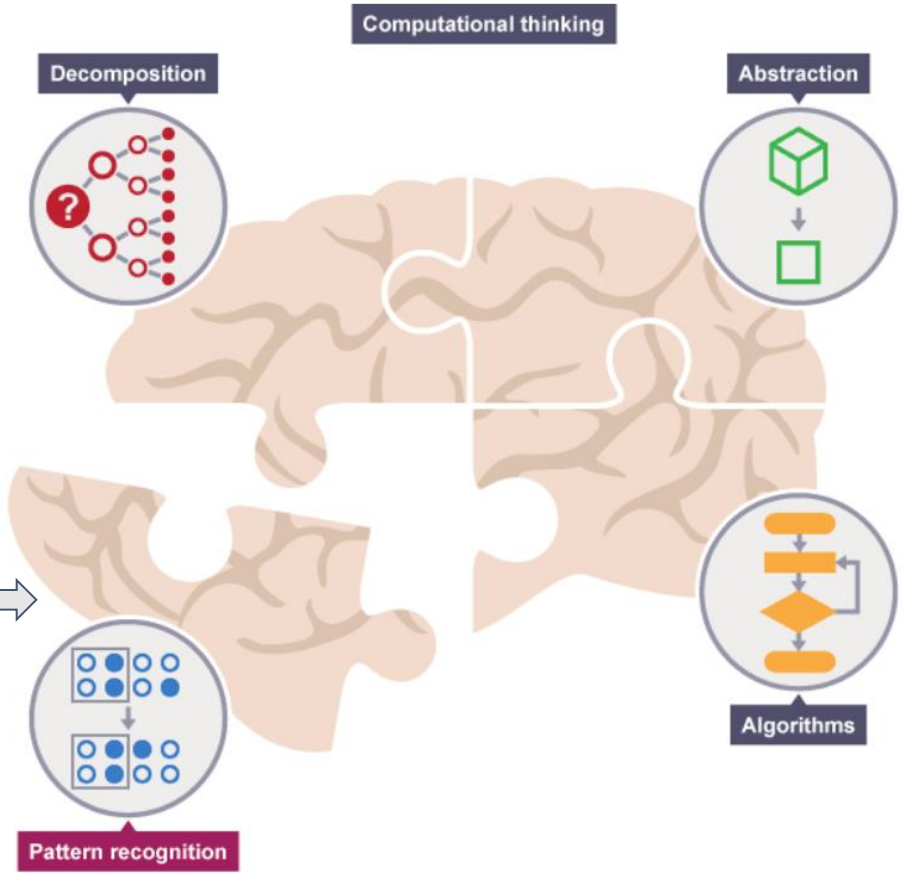


Nombre: Introduction to Computational Thinking
Autor: BBC
Fuente: https://www.bbc.co.uk/education/guides/zp92mp3/revision

Patrones

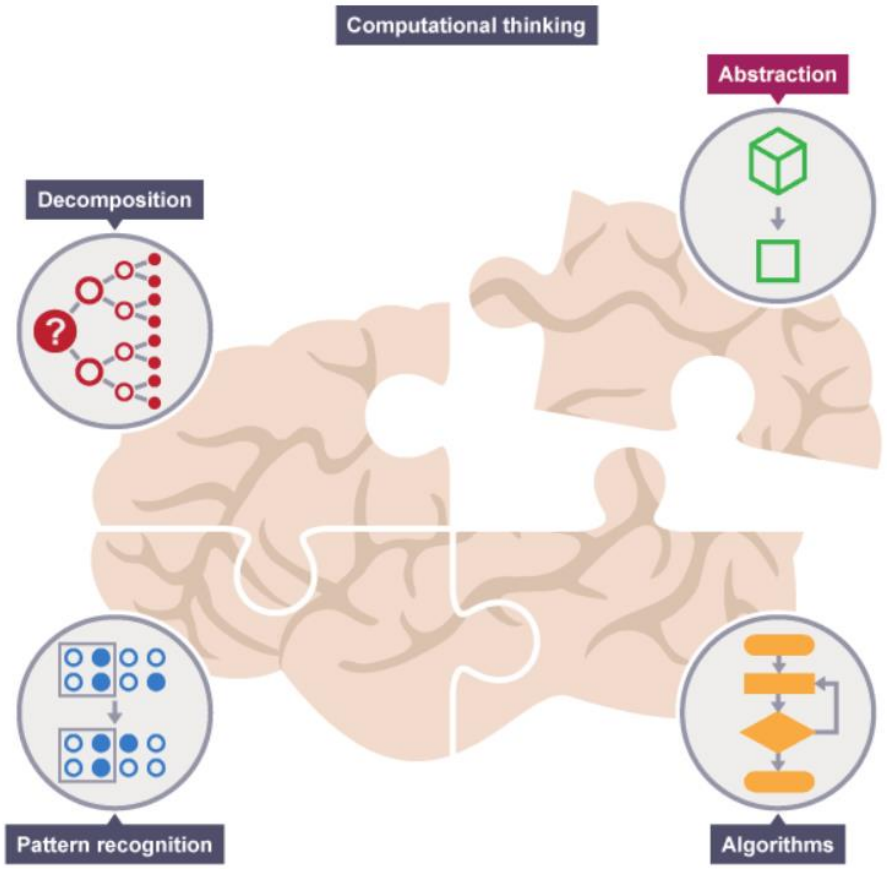
Cuando descomponemos un problema complejo, encontramos patrones entre los problemas más pequeños que creamos.

Encontrar similitudes entre problemas pequeños y descompuestos que pueden ayudarnos a resolver problemas más complejos de manera más eficiente.



Nombre: Introduction to Computational Thinking
Autor: BBC
Fuente: https://www.bbc.co.uk/education/guides/zp92mp3/revision

Abstracción



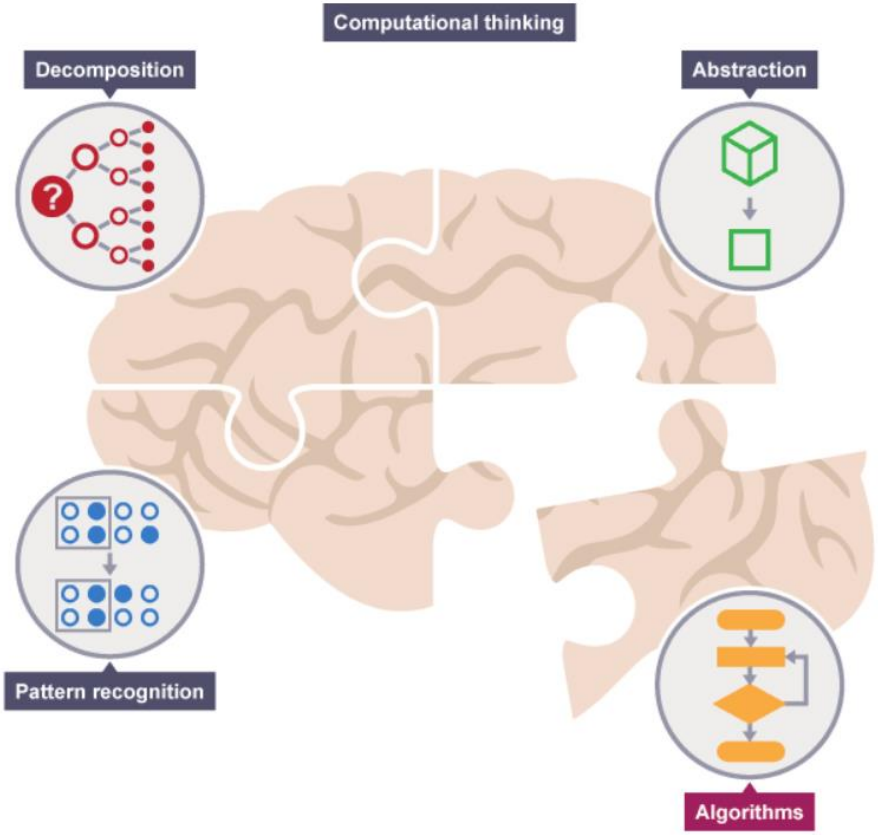
Luego de descomponer los problemas, buscamos patrones entre y dentro de los problemas más pequeños.

Después filtramos, ignorando, las características de los patrones que no necesitamos para concentrarnos en las que sí necesitamos.

Creamos una representación (idea) de lo que estamos tratando de resolver.

Nombre: Introduction to Computational Thinking
Autor: BBC
Fuente: https://www.bbc.co.uk/education/guides/zp92mp3/revision

Algoritmia



Nombre: Introduction to Computational Thinking
Autor: BBC
Fuente: https://www.bbc.co.uk/education/guides/zp92mp3/revision

Podemos descomponer el problema en partes más pequeñas y luego planear cómo encajan nuevamente en un orden adecuado para resolver el problema.

Un **algoritmo** es un plan, un conjunto de instrucciones paso a paso para resolver un problema. Características:

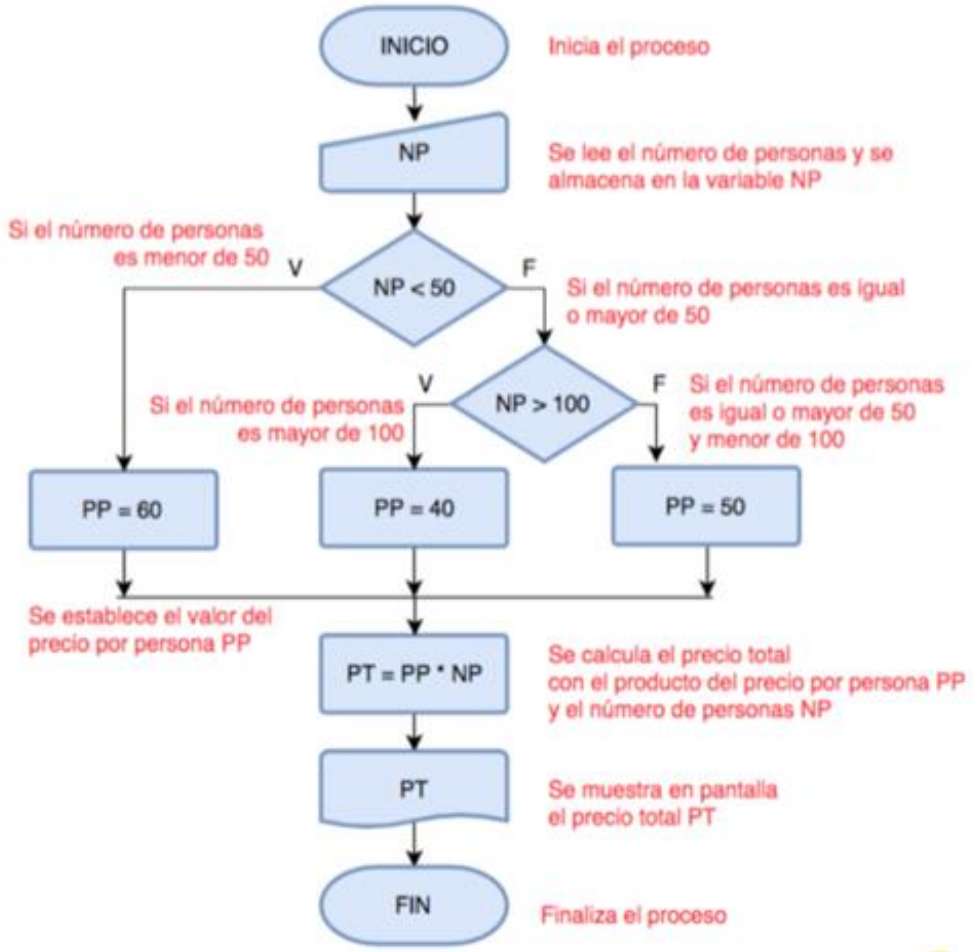
- Ser claro
- Tener un punto de partida
- Tener un punto de finalización
- Tener instrucciones claras

Problema – Cuenta restaurante

En un restaurante que organiza banquetes colectivos cobran 60 €/ persona si el número de comensales es inferior a 50; 50 €/persona si está entre 50 y 100 y 40 €/persona si es superior a 100 invitados.

Calcula el presupuesto total en función del número de personas.

Problema – Cuenta restaurante



Problema Computacional

Problema **abstracto** que permite establecer formalmente la relación deseada entre la entrada de un **algoritmo** (modelo matemático) y su salida.

Una **solución algorítmica** a un problema abstracto consiste de un algoritmo que por cada instancia del problema calcula al menos una solución correspondiente –en caso de haberla– o expide un certificado de que no existe solución alguna.

Un **problema abstracto** se convierte en un **problema concreto** cuando las instancias y soluciones están codificadas en forma de lenguajes formales.

Elementos de un problema computacional

Problema abstracto: Ordenar de manera ascendente un conjunto de números enteros

Instancias:

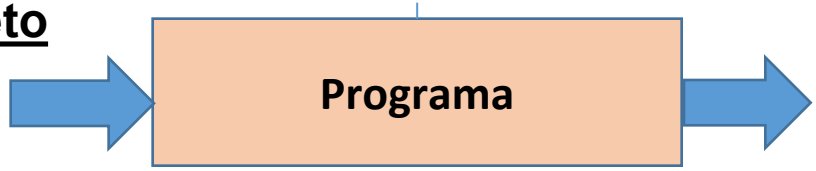
Sucesión finita de número enteros:
(a2, a3, … an)

Solución:

Permutación de la sucesión de entrada
(a1´, a2´… an´) tal que :
(a1´ ≤ a2´ ≤ … ≤ an)

Consistente con

Problema concreto



Instancias:
(8, 5, 20, 1, 4)
(7, 3, 1, 9, -1, 7, 4)
(6, 3, 5.8, 4, 6.2)

¿Cuál es la solución esperada en cada caso?

Clasificación de los problemas computacionales

Según el **tipo de problema** computacional

- Decisión
- Ordenamiento y búsqueda
- Optimización

Según la **capacidad de terminación** del algoritmo

Según el **modelo algorítmico** que lo resuelve

Según el tipo de problema computacional (1)

Problemas de decisión:

- Problema en donde las respuestas posibles son «sí» o «no».

Ejemplos:

- Determinar si un número entero es primo o no.
- Determinar si el libro está prestado o no.

SI
Verdadero

NO
Falso

Fuente: https://www.flickr.com/photos/73119057@N00/3026580124

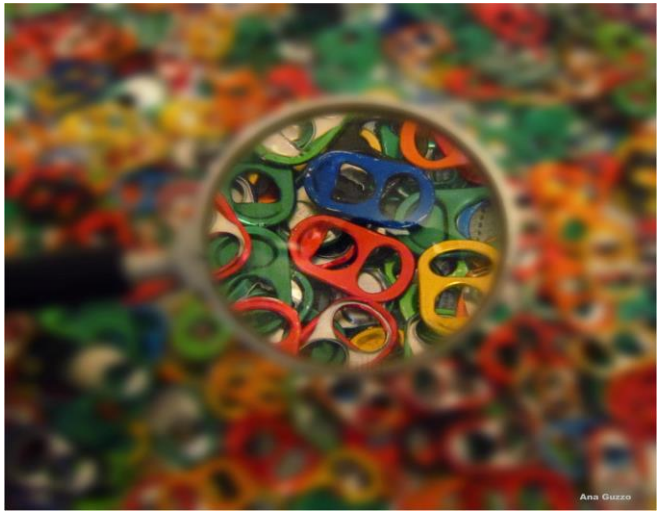
Según el tipo de problema computacional (2)

Problemas de ordenamiento y búsqueda:

- **Búsqueda**: Encontrar un elemento que satisface una propiedad.
- **Ordenamiento**: establecer una **relación de orden** entre los elementos. Es importante hacer un ordenamiento antes de la búsqueda.

Ejemplos:

- Dado un conjunto de números enteros
  - Encontrar **un** número que sea primo.
- Dado una lista de nombres:
  - Buscar un nombre determinado.



Fuente:
https://www.flickr.com/photos/allg/8288766013

Según el tipo de problema computacional (3)

Problemas de optimización:

- No solo se busca una solución, sino que se busca "***la mejor***" de todas. Cada problema de optimización es un problema de búsqueda y una ***función objetivo***, que determina la calidad de las soluciones.

Ejemplo:

- Encontrar la ruta más corta para ir de un punto p1 a un punto p2.

Según la capacidad de terminación del algoritmo

Problemas no computables:

- No existe un algoritmo.
- Ejemplo: Un programa que lea otro programa para ver si terminó su ejecución.

Problemas no tratables:

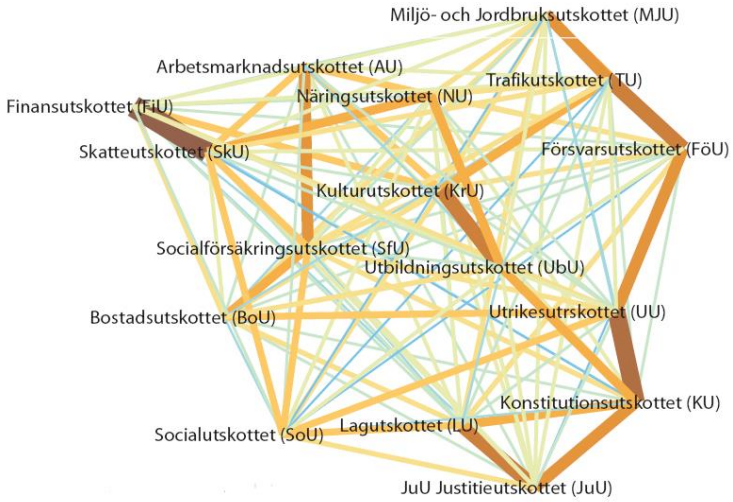
- Existe un algoritmo pero se requieren tantos recursos que no es útil en la práctica.
- Ejemplo: Generar todas las combinatorias posibles de las letras del alfabeto.
- En estos casos se utilizan **heurísticas** para buscar una solución aproximada.

Problemas tratables:

- Existe un algoritmo que busca la mejor solución.

Según el modelo algorítmico

- Estructuras de datos básicas (listas unidimensionales, listas multidimensionales, pilas, colas, etc.).
- Algoritmos para manejo de cadenas de caracteres.
- Algoritmos de ordenamiento.
- Aritméticos y álgebra computacional.
- Algoritmos combinatorios.
- Teoría de números.
- Algoritmos de grafos.



Referencia: Programming Challenges.
Autores: Steven Skiena, Miguel Revilla

FIN DE TEMA

Implementación de Sistemas para la Agricultura Digital y Pensamiento Algorítmico