

# Temario de Algoritmos Computacionales

M. en C. Diego Alberto Barceló Nieves  
Facultad de Ciencias  
Universidad Nacional Autónoma de México

El siguiente temario está basado en el [temario oficial del curso](#). Al elaborarlo, asumimos que quienes tomarán el curso no tienen experiencia previa con programación, pero sí tienen bases teóricas sólidas de álgebra lineal y cálculo diferencial e integral de una variable, así como nociones básicas de ecuaciones diferenciales ordinarias. Para los Módulos 1, 2 y 3 utilizaremos el lenguaje de programación [Julia](#). La duración aproximada de cada módulo se indica en paréntesis.

## 0. Introducción a la programación (3 semanas)

1. ¿Qué es un programa? (Paradigma imperativo de la programación)
2. ¿Qué ocurre cuando se ejecuta un programa? (Código de máquina y código fuente, lenguajes de programación, sintaxis y semántica, comentarios y mensajes de error)
3. Licencias: legalidad y ética. (*Software* de código cerrado y de código abierto, *software* privativo y *software* libre)
4. ¿Cómo escribo y ejecuto un programa? (Editor de texto y terminal virtual, REPLs e IDEs)
5. ¿Cómo aprendo a programar? (Manuales, documentación y foros de preguntas)
6. Herramientas útiles para hacer programación. ([Jupyter](#) y [Pluto](#), [Git](#) y [GitHub/GitLab](#))
7. Algoritmos, diagramas de flujo y pseudocódigo.

## 1. Estructura básica de la programación (4 semanas)

1. Operadores aritméticos y tipos de datos numéricos. (Precedencia y asociatividad)
2. Sistemas numéricos de punto flotante y error numérico. (Épsilon de máquina y propagación de error)
3. Tipos de datos de texto y arreglos. (Matrices, vectores, índices y subarreglos)
4. Variables, constantes y funciones. (Manejo de memoria y recursividad)
5. Operaciones lógicas y valores Booleanos. (Operadores lógicos y de comparación)
6. Condicionales y control de flujo. (Declaraciones `if`, `else` y `elseif`)
7. Ciclos. (Ciclos iterativos `while`, `for`, y ciclos recursivos)
8. Estructura de la programación modular. (Bibliotecas)

## 2. Representaciones visuales (3 semanas)

1. Gráfica de funciones y animaciones con [Plots](#).
2. Visualización de datos con [Plots](#).
3. Visualización de sistemas de ecuaciones diferenciales ordinarias con [Plots](#).
4. Manipulación de imágenes digitales con [JuliaImages](#).

### 3. Cómputo científico (4 semanas)

1. Métodos numéricos. (Estabilidad y convergencia).
2. Solución de sistemas lineales de ecuaciones algebraicas. (Método de eliminación Gaussiana)
3. Aproximación de raíces. (Método de Newton)
4. Solución de ecuaciones diferenciales ordinarias. (Método de Euler)
5. Caminante aleatorio.

### 4. Introducción a otros lenguajes de programación (2 semanas)

1. Introducción a Python. (Sintaxis, tipos de datos básicos y funciones con cantidades variables de parámetros)
2. Introducción a Programación Orientada a Objetos. (Clases y objetos, parámetros y atributos, herencia y polimorfismo)
3. Introducción a [Manim](#).

### Bibliografía recomendada para el curso

1. [Lauwens y Downey, \*Think Julia: How to Think Like a Computer Scientist\* \(2019\).](#)
2. Burden, Faires y Burden, *Numerical Analysis* (2016).
3. Cormen, Leiserson, Rivest y Stein, *Introduction to Algorithms* (2009).
4. Cairó, *Metodología de la programación: Algoritmos, diagramas de flujo y programas* (2003).

### Bibliografía complementaria

1. [Documentación de Julia](#).
2. [Documentación de Jupyter](#).
3. Material del curso “[Introduction to Computational Thinking](#)” del MIT.
4. Blum y Bresnahan, *Linux Command Line and Shell Scripting Bible* (2015).
5. [Stallman, \*Software libre para una sociedad libre\* \(2004\).](#)