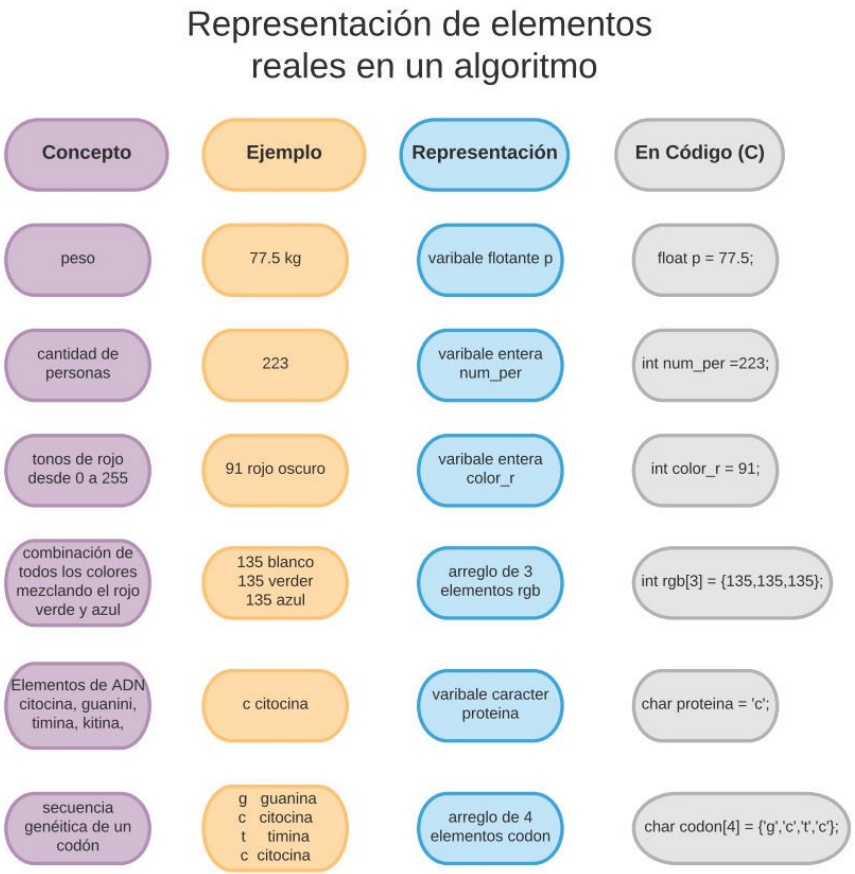
Sobre la Representación y la creación de Soluciones

Para poder generar soluciones descritas mediante algoritmos es necesario poder representar la realidad de una forma más simple, pero sin perder de vista los puntos que son relevantes para resolver el problema.

La representación comienza con el tipo de información que obtenemos del mundo real y como la planteamos en nuestro problema. En la siguiente figura mostramos ejemplos:



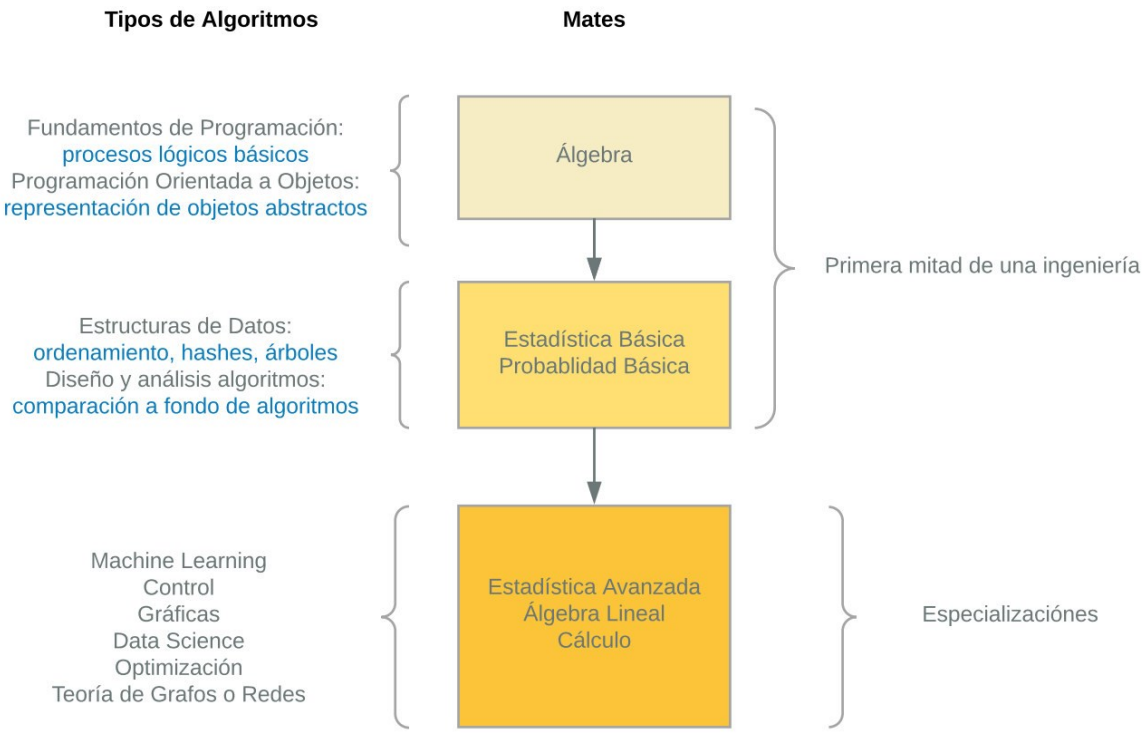
La representación también incluye como expresamos las relaciones entre los datos que representamos inicialmente. En la siguiente figura presentamos ejemplos de relaciones básicas:

Estrategias simples para representar relaciones

Tipo de Relación	Ejemplo de Relación	Forma de Calcularla
Diferencia entre 2 elementos numericos.	diferencia entre perros y gatos	abs(perros - gatos) o pow((perros - gatos), 2)
Relación entre dos variables. Porcentuales	Perros respecto a mascotas	regla de 3 perros/mascotas*100
Una variable directamente proporcional a otra	La ganacias obtenidas en pesos, son directamente proporcioanles al número de dólares y tipo de cambio.	Ganacias = Dolares*TipoDeCambio
Una variable inversamente proporcional a otra	presupuesto asignado a diversión vs alimentación	diversión/alimentación o alimentación/diversión
Decremento constante en el tiempo	tiempo restante para que se acabe el examen	minutos = minutos - 1 o minutos ++
Incremento constante en el tiempo	Contador de goles para un equipo de futbol soccer	marcador = marcador +1 o marcador ++
Incremento exponeciales	numeros de nodos en un arbol binario	marcador = marcador * marcado
Incremento que va decreciendo con el tiempo	Curva de aprendizaje para una nueva tecnología	log(incremento) o sqrt(incremento) > 0

Las representaciones tanto los tipos de dato como de las relaciones entre las datos son piezas más pequeñas de rompecabezas (algoritmos). Los algoritmos básicamente son estrategias creadas con estas piezas para solucionar problemas específicos. En un inicio aprenderán algoritmos sencillos e incluso escribirán los suyos para resolver problemas triviales. Entre más avancen en sus áreas de conocimiento aprenderán cada vez algoritmos más complejos. Los algoritmos son en sí el repositorio de estrategias que venimos generando de hace 100 años. Es decir, los algoritmos representan nuestro conocimiento acumulado sobre cómo resolver problemas de forma automatizada. Saber leer y escribir algoritmos nos abre las puertas a todas las soluciones que hemos creado hasta la fecha. El siguiente diagrama es una muy resumida y simplificada explicación de como se relacionan algunas de sus materias básicas con los

algoritmos que estarán viendo en su formación básica como ingenieros. Represe



Para solucionar cualquier problema, primero debemos:

Entender que es lo que nos están pidiendo.

Identificar que información o recursos nos están brindando.

Esto nos llevará a poder ver la situación de manera más clara. Usualmente a este nivel básico, se nos plantean 2 tipos de situaciones:

1) Situación 1: Automatizar un proceso que ya se conoce o que está explícito en los requerimientos. Esto es común cuando ya se cuenta con un algoritmo específico o una fórmula o un proceso clara y no ambiguo sobre como solucionar el problema. En esta situación, solo es necesario entonces implementar la solución.

   E.g. Crear un algoritmo que calcule el área de un circulo a partir de su radio con la fórmula: $area = PI * r^2$

2) Situación 2: Automatizar un proceso que no sabemos como ocurre, pero del cual tenemos ejemplos de entrada y de salida que esperamos. (De este tipo son los problemas de la programación competitiva) .

   E.g. Crear un algoritmo que dado 3 números encuentre el más pequeño.

En ambos casos una vez que entendemos el problema generamos ejemplos o casos de prueba. Esto es importante porque nos permite entender dada qué información esperamos qué resultado. Con estos casos probaremos nuestro algoritmos para saber si son correctos o están mal hechos. Siguiendo los ejemplos anteriores tenemos que:

E.g.	entrada	salida	
	Circulo_Area(3)	- >	28.27
	Circulo_Area(5)	- >	78.54
	Circulo_Area(-5)	- >	78.54

E.g.	entradas		salida
	núm más pequeño(10, 23,1)	- >	1
	núm más pequeño(200, -23,9)	- >	-23
	núm más pequeño(0, 0, 0)	- >	0

Una forma útil de comenzar con el proceso es resolver el problema primero a mano para ejemplos concretos, y escribir cada paso de forma detallada. Si un paso no se puede pasar mapear a una instrucción de programación, entonces el paso no está descrito de forma detallda y debe descomponerse en pasos más específicos. Los algoritmos para los ejemplos anteriores podrían ser los siguientes.

```
circulo_area(PI = 3.1415, r)     <- Estado Inicial
    area = PI*r*r
regresa(area)       <- Estado Final
```

```
circulo_area(PI = 3.1415, r)     <- Estado Inicial
    area = r*r
    area = PI*area
regresa(area)      <- Estado Final
```

```
numero_menor(a, b, c)     <- Estado Inicial
   if(a > b  > c )
        res = c
   esle if ( a < b  < c)
        res = a
   else
        res = b
regresa(res)       <- Estado Final
```

```
numero_menor(a, b, c)     <- Estado Inicial
   if(a > b  > c )
        regresa (c) y termina    <- Estado Final
   esle if ( a < b  < c)
        regresa (a) y termina   <- Estado Final
   else
        regresa (c) y termina   <- Estado Final
```

Como se puede ver hay varios algoritmos que pueden solucionar el mismo problema, es importarnte contemplar que aunque pueden haber varias soluciones, no todas son igual de buenas y es común sacrificar algunos aspectos (velocidad, complejidad, memoria, transparencia, extendibilidad) para mejorar otros.