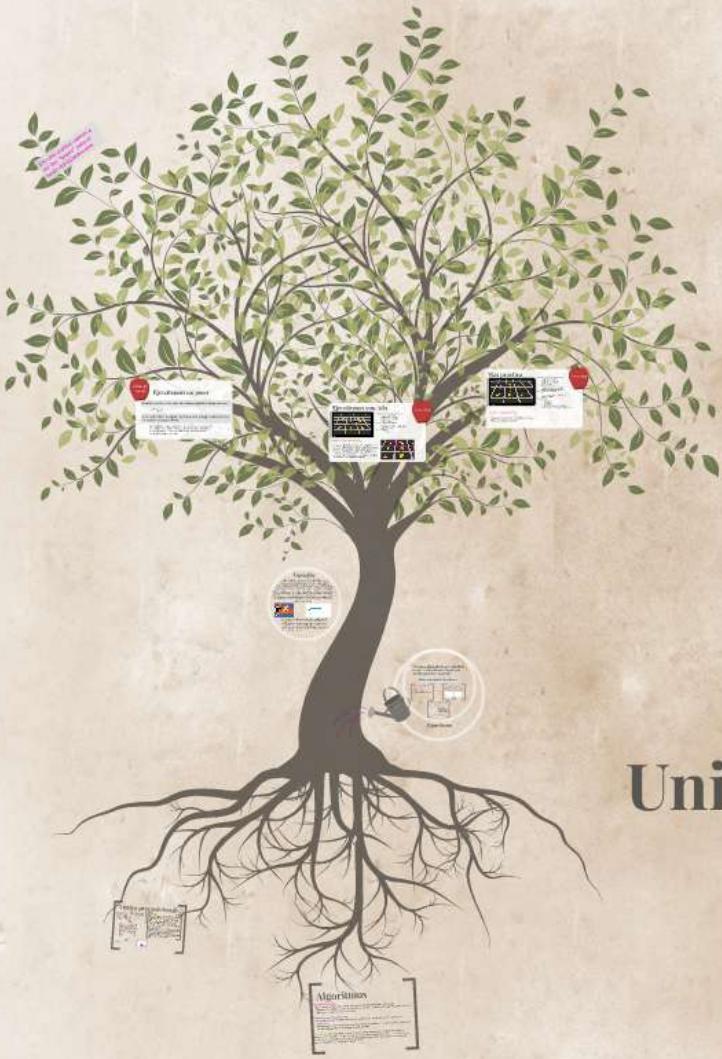
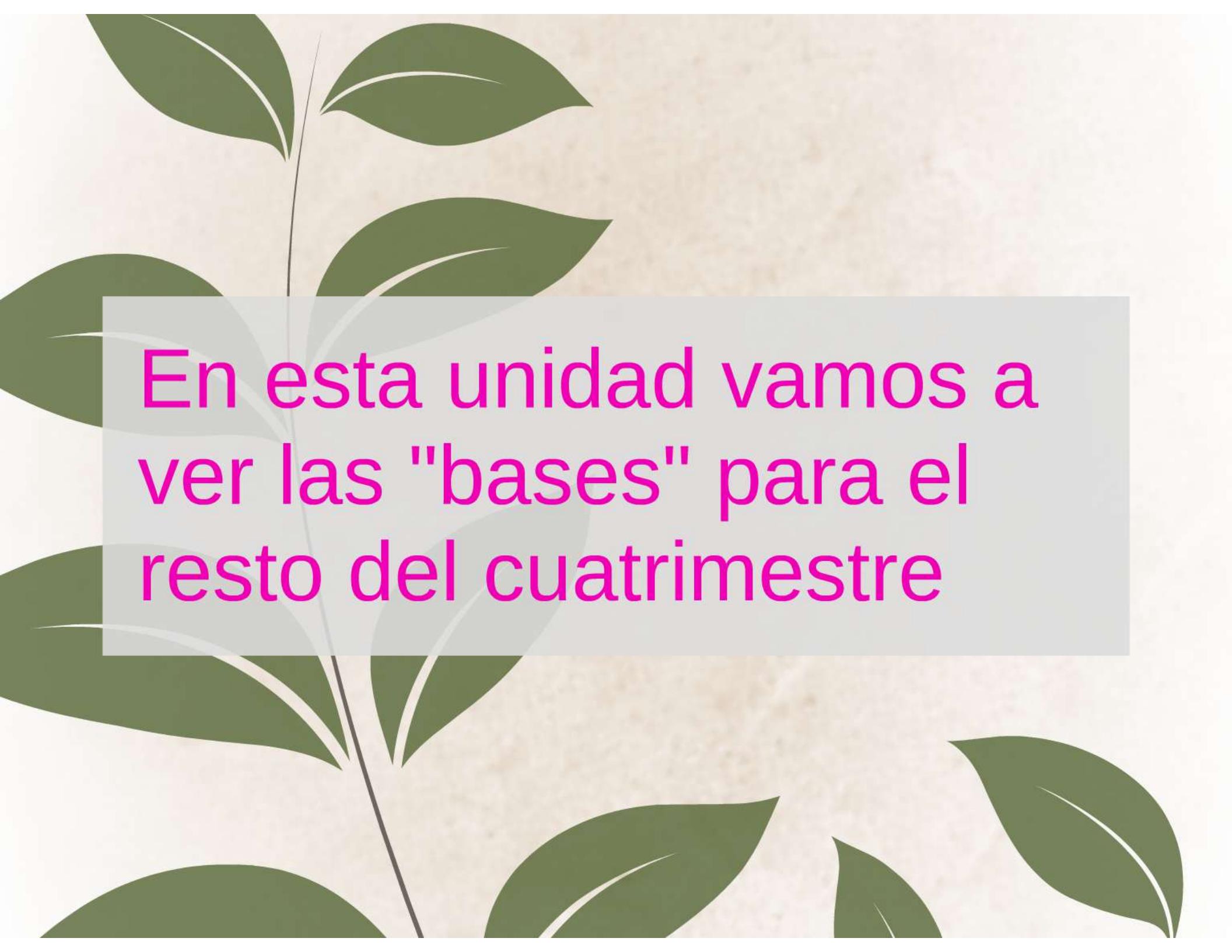


Unidad INTRODUCTORIA



Unidad INTRODUCTORIA



En esta unidad vamos a ver las "bases" para el resto del cuatrimestre

Lógica proposicional

¿Qué es una proposición?

Una proposición es una expresión de la cual tiene sentido decir si es verdadera o falsa, o sea es posible asignarle un valor de verdad (verdadero o falso, pero no ambos).

Atómica si no puede ser descompuesta en otras proposiciones. Cuando se unen varias proposiciones atómicas se forma una proposición **molecular o compuesta**.

Dicha unión se realiza mediante **conectivos lógicos** o términos de enlace.

Simbología

Se utilizan letras en minúsculas para simbolizar cada proposición atómica

p proposición atómica
p y q proposición molecular

Conectivos lógicos
Y como \wedge
O como \vee
No como \neg

TABLAS DE VERDAD

¿Qué son? Una representación gráfica de todas las combinaciones de los valores de verdad de una proposición lógica.

¿Cómo se construyen? Se comienza por distinguir las proposiciones atómicas y hacer una columna por cada una.

Respecto de la cantidad de filas se calcula con la fórmula:

2^n , es decir 2 elevado a la "n" siendo n la cantidad de proposiciones atómicas.

Luego se divide por dos la cantidad de filas y se coloca mitad con valor de verdad "verdadero" y mitad con valor de verdad "falso", y se sigue con ese mismo criterio con las otras filas.

A continuación se arman las proposiciones moleculares teniendo en cuenta el peso de los conectores lógicos (\vee , \wedge , \neg , \rightarrow , \leftrightarrow).

NO SON PROPOSICIONES:
"LA CASA"
"QUIERES VENIR A
SEÑAR?"

POR EJEMPLO:
"LA CASA ES BLANCA"
"ES DOMINGO"



¿Qué es una proposición?

Una proposición es una expresión de la cual tiene sentido decir si es verdadera o falsa, o sea es posible asignarle un valor de verdad (verdadero o falso, pero no ambos).

POR EJEMPLO:

"LA CASA ES BLANCA"

"ES DOMINGO"

SE PUEDEN
APLICAR
VALORES DE
VERDAD

NO SON PROPOSICIONES:

"LA CASA"

**"¿QUERÉS VENIR A
CENAR?"**

NO SE PUEDEN
APLICAR
VALORES DE
VERDAD

Atómica si no puede ser descompuesta en otras proposiciones.

Cuando se unen varias proposiciones atómicas se forma una proposición **molecular** o **compuesta**.

Dicha unión se realiza mediante **conectivos lógicos** o términos de enlace.

Simbología

Se utilizan letras en minúsculas para simbolizar cada proposición atómica

p proposición atómica

p y q proposición molecular

Conejtos lógicos

Y como \wedge

O como \vee

No como \neg

TABLAS DE VERDAD

¿Qué son? Una representación gráfica de todas las combinaciones de los valores de verdad de una proposición lógica.

¿Cómo se construyen? Se comienza por distinguir las proposiciones atómicas y hacer una columna por cada una. Respecto de la cantidad de filas se calcula con la fórmula: 2^n , es decir 2 elevado a la "n" siendo n la cantidad de proposiciones atómicas.

Luego se divide por dos la cantidad de filas y se coloca mitad con valor de verdad “verdadero” y mitad con valor de verdad “falso”, y se sigue con ese mismo criterio con las otras filas.

A continuación se arman las proposiciones moleculares teniendo en cuenta el peso de los conectores lógicos (\vee , \wedge , \neg , $\neg\vee$, $\neg\wedge$, \rightarrow).

DETALLES PARA
ANALIZAR
CUALQUIER
TABLA DE VERDAD

PRINCIPALES TABLAS DE VERDAD

p	q	$p \rightarrow q$
V	V	V
V	F	F
F	V	V
F	F	V

Tabla 4: Condicional

p	q	$p \leftrightarrow q$
V	V	V
V	F	F
F	V	F
F	F	V

Tabla 5: Bicondicional

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

Tabla 2: Conjunción

p	$\neg p$
V	F
F	V

Tabla 1: Negación

3 - Luego dividimos por dos la cantidad de filas y colocamos mitad con valor de verdad "verdadero" y mitad con valor de verdad "falso", y se sigue con ese mismo criterio con los otros filas.

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

Tabla 3: Disyunción

1- distinguimos los proposiciones atómicas y hacemos una columna por cada una.

2- Hacemos 4 filas dada la fórmula; 2ⁿ, siendo n la cantidad de proposiciones atómicas

4 - Por último armamos las proposiciones moleculares teniendo en cuenta el peso de los conectores lógicos (\wedge , \neg , \rightarrow , \leftrightarrow , \vee).



1 - distinguimos las proposiciones atómicas y hacemos una columna por cada una.

2- Hacemos 4 filas dada la fórmula: 2^n , siendo n la cantidad de proposiciones atómicas

V

V

F

F

V

F

V

F

V

V

V

F

					b
--	--	--	--	--	---

3 - Luego dividimos por dos la cantidad de filas y colocamos mitad con valor de verdad “verdadero” y mitad con valor de verdad “falso”, y se sigue con ese mismo criterio con las otras filas.

Tabl



F	F	<	<	p	
---	---	---	---	---	--

Si dividimos por dos la cantidad de filas
nos mitad con valor de verdad
“dijo” y mitad con valor de verdad
y se sigue con ese mismo criterio con
esas filas.

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

4 - Por último armamos las proposiciones moleculares teniendo en cuenta el peso de los conectores lógicos (\vee , \wedge , \rightarrow , \leftrightarrow , \neg).

os las
icas y
lumna
o una.

dada
on la
o de
nicas

PRINCIPALES TABLAS DE VERDAD

DETERMINAR PARA
ABSTRACTAS
DIFERENTES
DADA PARA DE ESTAS
TABLAS DE VERDAD

p	q	$p \rightarrow q$
V	V	V
V	F	F
F	V	V
F	F	V

Tabla 4: Condicional

p	q	$p \leftrightarrow q$
V	V	V
V	F	F
F	V	F
F	F	V

Tabla 5: Bicondicional

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

Tabla 2: Conjunción

p	$\neg p$
V	F
F	V

Tabla 1: Negación

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

Tabla 3: Disyunción

3. Luego dividimos por dos la cantidad de filas y colocamos mitad con valor de verdad "verdadero" Y mitad con valor de falso "falso", y se sigue con ese mismo criterio con los otras filas.

1- distinguimos las proposiciones atómicas y hacemos una columna por cada una.

2- Hacemos 4 filas dado la fórmula: $p \wedge q$, siendo n la cantidad de proposiciones atómicas

4 - Por último armamos las proposiciones moleculares teniendo en cuenta el peso de los conectores lógicos (\wedge , \rightarrow , \neg , \vee , \leftrightarrow).

**DETENGÁMONOS PARA
ANALIZAR
CUIDADOSAMENTE
CADA UNA DE ESTAS
TABLAS DE VERDAD**

DETALLES PARA
ANALIZAR
CUALQUIER
TABLA DE VERDAD

PRINCIPALES TABLAS DE VERDAD

p	q	$p \rightarrow q$
V	V	V
V	F	F
F	V	V
F	F	V

Tabla 4: Condicional

p	q	$p \leftrightarrow q$
V	V	V
V	F	F
F	V	F
F	F	V

Tabla 5: Bicondicional

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

Tabla 2: Conjunción

p	$\neg p$
V	F
F	V

Tabla 1: Negación

3 - Luego dividimos por dos la cantidad de filas y colocamos mitad con valor de verdad "verdadero" y mitad con valor de verdad "falso", y se sigue con ese mismo criterio con los otros filas.

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

Tabla 3: Disyunción

1- distinguimos los proposiciones atómicas y hacemos una columna por cada una.

2- Hacemos 4 filas dada la fórmula; 2ⁿ, siendo n la cantidad de proposiciones atómicas

4 - Por último armamos las proposiciones moleculares teniendo en cuenta el peso de los conectores lógicos (\wedge , \neg , \rightarrow , \leftrightarrow , \vee).

Ejercitemos un poco

Realizar las tablas de cada una de las siguientes proposiciones:

- $(p \wedge q) \Rightarrow r$
- $\neg(p \wedge q) \vee r$

Ahora simbolizar la siguiente proposición y luego realizar la tabla de verdad correspondiente:

Juan partirá para Japón, si María se queda en Venecia. Rosa viajará a Luxemburgo o Juan no partirá para Japón. O María no se queda en Venecia o Rosa no viajará a Luxemburgo. Por consiguiente, María no se queda en Venecia.

Algoritmos

Algunas definiciones:

- Es un conjunto de instrucciones que nos sirven para resolver un problema.
- Conjunto finito de instrucciones para llevar a cabo una tarea. Constan de pasos finitos, no ambiguos y, de ser posible, eficientes.

Precondiciones y Poscondiciones

- **Precondición** es la información que se conoce como verdadera antes de iniciar el programa.
- **Poscondición** es la información que debería ser verdadera al concluir el programa, si se cumple adecuadamente el requerimiento pedido.

Comentarios: Además de encontrar datos e instrucciones en un programa podemos encontrar comentarios. Es texto aclaratorio para el programador o el usuario, que no es entendido ni ejecutado por la computadora. Este texto será muy importante cuando se intente modificar o corregir el programa

En otras palabras dijimos que "Algoritmo es una lista de órdenes a cumplir, que permite resolver un problema"

Ahora esto puede hacerse con:

Lenguaje coloquial

Veamos un algoritmo para Abrir una puerta que está cerrada con llave.

"Opción uno"

1. tomar la llave de la puerta
2. introducir la llave en la cerradura
3. girar la llave en el sentido de las agujas del reloj
4. girar la llave en el sentido de las agujas del reloj
5. abrir con el picaporte la puerta

(En el ejemplo anterior se han enumerado los pasos para un mejor entendimiento, sin embargo no es necesario)

Ahora, como una segunda opción, se podría plantear el hecho de verificar si la llave es la correcta. En ese caso se podrían elaborar una serie de pasos como la siguiente:

"Opción dos"

1. tomar la llave de la puerta
2. verificar que la llave es la correcta
 - a. introducir la llave en la cerradura
 - b. girar la llave en el sentido de las agujas del reloj
 - c. girar la llave en el sentido de las agujas del reloj
 - d. abrir con el picaporte la puerta

Lenguaje de programación

Un lenguaje es básicamente un medio de comunicación.

En programación un lenguaje formal es diseñado para realizar procesos que pueden ser llevados a cabo por máquinas como las computadoras.



Pero ahora nos concentraremos en Pseudocódigo

Pseudocódigo

Un pseudocódigo no es el lenguaje natural que utilizamos las personas, pero tampoco llega a ser un lenguaje de programación, por lo que puede verse como un "estadio intermedio" que resulta sencillo de interpretar para las personas a la hora de pensar en algoritmos.

Dicho esto creamos nuestro propio pseudocódigo.

Veamos la ciudad de Ada en la imagen que se muestra a continuación:



Lenguaje coloquial

Veamos un algoritmo para **Abrir una puerta que está cerrada con llave.**

“Opción uno”

1. tomar la llave de la puerta
2. introducir la llave en la cerradura
3. girar la llave en el sentido de las agujas del reloj
4. girar la llave en el sentido de las agujas del reloj
5. abrir con el picaporte la puerta

(En el ejemplo anterior se han enumerado los pasos para un mejor entendimiento, sin embargo no es necesario)

Ahora, como una segunda opción, se podría plantear el hecho de verificar si la llave es la correcta. En ese caso se podría elaborar una serie de pasos como la siguiente:

“Opción dos”

1. tomar la llave de la puerta
2. verificar que la llave es la correcta
 - a. introducir la llave en la cerradura
 - b. girar la llave en el sentido de las agujas del reloj
 - c. girar la llave en el sentido de las agujas del reloj
 - d. abrir con el picaporte la puerta

Lenguaje de programación

Un lenguaje es básicamente un medio de comunicación.

En programación un lenguaje formal es diseñado para realizar procesos que pueden ser llevados a cabo por máquinas como las computadoras.

En IPI trabajaremos con el lenguaje Python



```
def add5(x):
    return x+5

def dotwrite(ast):
    nodename = getNodeName()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print '  ts [%s=%s' % (nodename, label),
    if isinstance(ast[1], str):
        if ast[1].strip():
            print '= %s]' % ast[1]
        else:
            print ']'
    else:
        print ']';
    children = []
    for n, childenumerate(ast[1:]):
        children.append(dotwrite(child))
    print ', ts-> {' % nodename
    for n in children:
        print '%s' % name,
```

Pero ahora nos
concentraremos en
Pseudocódigo

Pseudocódigo

Un pseudocódigo no es el lenguaje natural que utilizamos las personas, pero tampoco llega a ser un lenguaje de programación, por lo que puede verse como un “estado intermedio” que resulta sencillo de interpretar para las personas a la hora de pensar en algoritmos.

Dicho esto creemos nuestro propio pseudocódigo.

Veamos la ciudad de **Ada** en la imagen que se muestra a continuación:



- Se trata de una ciudad que tiene **nueve (9) manzanas**, las llamaremos: M1, M2, y así hasta M9.
- También hay **ocho (8) calles**, las llamaremos: Calle1, Calle 2, y así hasta Calle 8.
- Observemos que cada calle tiene su **mano** indicada por una flecha (no hay doble mano en esta ciudad).
- Luego cada esquina tiene un **semáforo** que indica el “paso” con el color verde, “precaución” con color amarillo y “prohibido” pasar con color rojo.
- En las veredas puede haber **residuos** y también **cestos** para esos residuos.
- Por último supongamos que esa ciudad la recorrerá nuestra amiga **Ada**, quién es bastante distraída y no conoce demasiado cómo desenvolverse en la ciudad. Por eso debemos indicarle correctamente qué debe hacer y de qué manera.

Sentencias de nuestro pseudocódigo

Adrián necesita ir a la tienda que está en la esquina de la Calle 4 y la Calle 2. ¿Qué sentencias de pseudocódigo necesitará para que Ada lo logre?

Algoritmo (ejemplo 0):
Comenzar a caminar
Caminar hacia la calle 4
Caminar hacia la calle 2
Comenzar una esquina

Secuencia: en el ejemplo las instrucciones se ejecutan una a continuación de la otra.

Estructuras de control

Si analizamos el algoritmo anterior, ¿qué sucedería si el semáforo no está en verde? nunca cruzaríamos ni cometeríamos la ciudad que fija. Entonces la estructura de control Decisión para este caso no es adecuada.

Podríamos agregar una nueva sentencia como: “Observar el estado del semáforo” y decidir a partir de ello.

Determinaríamos poder preguntar: ¿el semáforo está en verde? Caso contrario, nos quedamos en la calle y volvemos a preguntarnos en qué quita la estructura de control Decisión sería conveniente.

Algoritmo (ejemplo 1):
Comenzar a caminar
Caminar hacia la calle 4
Caminar hacia la calle 2
Comenzar una esquina

Decisión: es una estructura también llamada bifurcación, que trae con una condición y “desde” a partir de dónde.

Algoritmo (ejemplo 2):
Comenzar a caminar
Caminar hacia la calle 4
Caminar hacia la calle 2
Comenzar una esquina

Estructuras de control

Continuando con el algoritmo anterior, ¿qué sucedería si el semáforo no está en verde? nunca cruzaríamos ni cometeríamos la ciudad que fija. Entonces la estructura de control Decisión para este caso no es adecuada.

Podríamos con una repetición y como tenemos la pregunta ¿el semáforo está en verde?, utilizamos un Mientras queja que la pregunta sea negativa.

Volvamos a escribir el algoritmo:

Algoritmo (ejemplo 3):
Comenzar a caminar
Caminar hacia la calle 4
Caminar hacia la calle 2
Comenzar una esquina

Algoritmo (ejemplo 4):
Comenzar a caminar
Caminar hacia la calle 4
Caminar hacia la calle 2
Comenzar una esquina

Estructura de control

En los ejemplos anteriores vimos la estructura “seguir avanzar” y a partir de una variación. Mientras (sabiendo una condición) hacer... Sin embargo, estos repeticiones pueden darse siempre cuando queremos repetir una cierta cantidad de veces (sabiendo de antemano el número de veces). Al repetir se dice bucle.

Supongamos que queremos recorrer la manzana 3 (M3), comenzando por la vereda de la Calle 1, luego la Calle 4, la Calle 3 y por último la Calle 2 regresando nuevamente al lugar de origen.

Algoritmo (ejemplo 5):
Comenzar a caminar
Caminar hacia la calle 1
Caminar hacia la calle 4
Caminar hacia la calle 3
Caminar hacia la calle 2
Comenzar una esquina

Algoritmo (ejemplo 6):
Comenzar a caminar
Caminar hacia la calle 1
Caminar hacia la calle 4
Caminar hacia la calle 3
Caminar hacia la calle 2
Comenzar una esquina

Este ejercicio permitirá ver que lo que hacemos es repetir cuatro 4x dos instrucciones. La idea es que conociendo la cantidad de veces, podemos la estructura de control repetir.

Sentencias de nuestro pseudogódigo



Ahora indiquemosle a Ada que camine dos cuadras, es decir que vaya hasta la esquina de la Calle 1 y Calle 6.

ALGORITMO (Ejemplo 1):

Caminar una cuadra
Cruzar la calle
Caminar una cuadra

Ada puede entender las siguientes órdenes:

- Caminar una cuadra
- Doblar hacia la derecha
- Doblar hacia la izquierda
- Recoger un residuo
- Colocar en el cesto de residuos
- Cruzar la calle

Por último debemos saber que Ada comenzará cada algoritmo ubicada en la esquina de la Calle 1 y Calle 2, y siempre mirando hacia la Calle 4



Secuencia: en el ejemplo las instrucciones se ejecutan una a continuación de la otra.

'entencias de nuestro pseudogódigo'



Indiquémosle a Ada que camine dos cuadras, es decir que recorra la esquina de la Calle 1 y

Ada puede entender las siguientes órdenes:

- Caminar una cuadra
- Doblar hacia la derecha
- Doblar hacia la izquierda
- Recoger un residuo
- Colocar en el cesto de residuos
- Cruzar la calle

Por último debemos saber que Ada comenzará cada algoritmo ubicada en la esquina de la Calle 1 y Calle 2, y siempre mirando hacia la Calle 4





- Recoger un residuo
- Colocar en el cesto de residuos
- Cruzar la calle

Por último debemos saber que Ada comenzará cada algoritmo ubicada en la esquina de la Calle 1 y Calle 2, y siempre mirando hacia la Calle 4

Ahora indiquémosle a Ada que camine dos cuadras, es decir que vaya hasta la esquina de la Calle 1 y Calle 6.

ALGORITMO (Ejemplo 1):

Caminar una cuadra

Cruzar la calle

Caminar una cuadra



Secuencia: en el ejemplo las instrucciones se ejecutan una a continuación de la otra.

Estructuras de control

Si analizamos el algoritmo vemos que efectivamente Ada caminaría dos cuadras, pero ¿qué sucede al cruzar la calle?

Lo hace sin observar el estado del semáforo.

Podríamos agregar una nueva sentencia como “Observar el estado del semáforo” y decidir a partir de ello.

Deberíamos también poder preguntar ¿el semáforo está en verde?

Con estos cambios reescribamos el algoritmo anterior pensando en que quizá la estructura de control *Decisión* sería conveniente.

ALGORITMO (Ejemplo 1):

Caminar una cuadra

Observar el estado del semáforo

Si (el semáforo está en verde)

Cruzar la calle

Caminar una cuadra

Decisión: es una estructura también llamada bifurcación, que trabaja con una condición y "decide" a partir de ésta.

Estructuras de control

Continuando con el algoritmo anterior ¿qué sucedería si el semáforo no está en verde? nunca cruzaríamos ni caminaríamos la cuadra que falta. Entonces la estructura de control *Decisión* para este caso no es adecuada.

Probemos con una *repetición* y como tenemos la pregunta ¿el semáforo está en verde?, utilicemos un *Mientras* (para que la pregunta sea repetitiva).

Volvamos a escribir el algoritmo.

ALGORITMO (Ejemplo 1):

Caminar una cuadra

Observar el estado del semáforo

Mientras no sea cierto que (el semáforo está en verde)

 Observar el estado del semáforo

 Cruzar la calle

 Caminar una cuadra

Podríamos escribirlo de la siguiente forma:

ALGORITMO (Ejemplo 1):

Caminar una cuadra

Observar el estado del semáforo

Mientras (el semáforo está en rojo) o (el semáforo está en amarillo)

 Observar el estado del semáforo

 Cruzar la calle

 Caminar una cuadra

Observemos que a la condición que “el semáforo está en verde” la negamos, es decir, preguntamos si no era cierto. Esto es posible ya que observaremos el semáforo mientras no esté en color verde, ya cuando finalmente el semáforo habilite a Ada para cruzar la calle lo hará y caminará una cuadra. Sin embargo hubiéramos podido también realizar otras preguntas. Por ejemplo hubiéramos podido preguntar ¿el semáforo está en rojo? Y también ¿el semáforo está en amarillo?

¿SE PODRÍA PREGUNTAR LO SIGUIENTE?

MIENTRAS (EL SEMÁFORO ESTÁ EN ROJO) Y (EL SEMÁFORO ESTÁ EN AMARILLO)

semáforo está en verde?, utilizemos un bucle que sea repetitiva).

Volvamos a escribir el algoritmo.

ALGORITMO (Ejemplo 1):

Caminar una cuadra

Observar el estado del semáforo

Mientras no sea cierto que (el semáforo está en verde)

 Observar el estado del semáforo

 Cruzar la calle

 Caminar una cuadra

Podríamos escribirlo de la siguiente forma:

ALGORITMO (Ejemplo 1):

Caminar una cuadra

Observar el estado del semáforo

Mientras (el semáforo está en rojo) o (el semáforo está en amarillo)

Como tenemos la pregunta ¿el semáforo es un *Mientras* (para que la pregunta

Observemos que a la condición que “el semáforo está en verde” la negamos, es decir, preguntamos si no era cierto. Esto es posible ya que observaremos el semáforo mientras no esté en color verde, ya cuando finalmente el semáforo habilite a Ada para cruzar la calle lo hará y caminará una cuadra. Sin embargo hubiéramos podido también realizar otras preguntas. Por ejemplo hubiéramos podido preguntar ¿el semáforo está en rojo? Y también ¿el semáforo está en amarillo?

en

10)

Cruzar la calle
Caminar una cuadra

Podríamos escribirlo de la siguiente forma:

ALGORITMO (Ejemplo 1):

Caminar una cuadra

Observar el estado del semáforo

Mientras (el semáforo está en rojo) o (el semáforo está en amarillo)

 Observar el estado del semáforo

 Cruzar la calle

 Caminar una cuadra

¿SE PODRÍA PREGUNTAR LO SIGUIENTE?

MIENTRAS (EL SEMÁFORO ESTÁ EN ROJO) Y (EL SEMÁFORO ESTÁ EN AMARILLO)

L):

el semáforo

(el semáforo está en rojo) o (el semáforo está en

o del semáforo

¿SE PODRÍA PREGUNTAR LO SIGUIENTE?

MIENTRAS (EL SEMÁFORO ESTÁ EN ROJO) Y (EL SEMÁFORO ESTÁ EN AMARILLO)

Estructura de control

En los ejemplos anteriores vimos la repetición con un "Mientras" y a partir de una condición. **Mientras (sucede una condición) hacer...**

Sin embargo, estas repeticiones pueden darse también cuando queremos repetir una **cierta cantidad de veces** (sabiendo de antemano el número de veces a repetir).

Supongamos que Ada desea recorrer la manzana 1 (M1), comenzando por la vereda de la Calle 1, luego la Calle 4, la Calle 3 y por último la Calle 2 llegando nuevamente al lugar de origen.



ALGORITMO (Ejemplo 2):

Repetir 4 veces

Caminar una cuadra

Doblar hacia la derecha

ALGORITMO (Ejemplo 2):

Caminar una cuadra

Doblar hacia la derecha

Esa solución podemos ver que lo que hacemos es repetir cuatro (4) dos sentencias, en este caso podemos utilizar la estructura de control *repetición*.

Estructura de control

En los ejemplos anteriores vimos la repetición con un "Mientras" y a partir de una condición. **Mientras (sucede una condición) hacer...**

Sin embargo, estas repeticiones pueden darse también cuando queremos repetir una **cierta cantidad de veces** (sabiendo de antemano el número de veces a repetir).

Supongamos que Ada desea recorrer la manzana 1 (M1), comenzando por la vereda de la Calle 1, luego la Calle 4, la Calle 3 y por último la Calle 2 llegando nuevamente al lugar de origen.



ALGORITMO (Ejemplo 2):

Repetir 4 veces

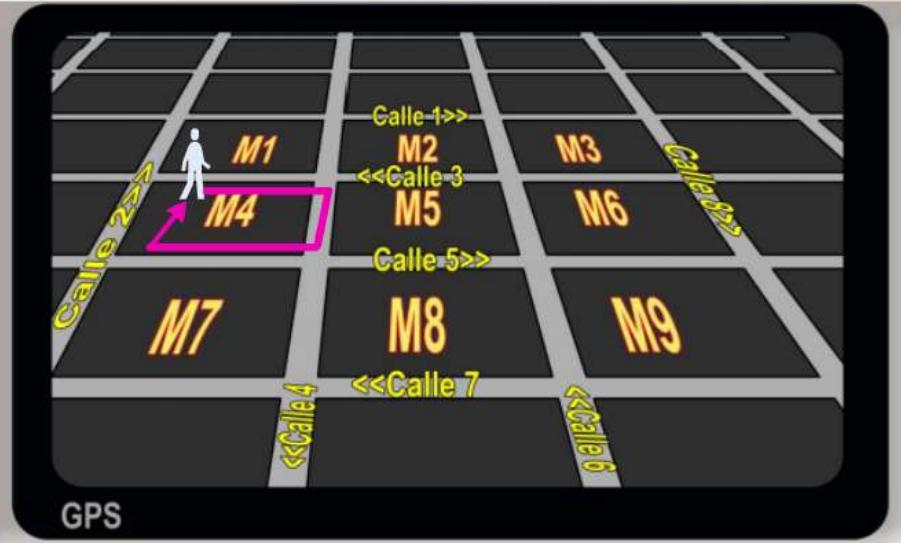
Caminar una cuadra
Doblar hacia la derecha

ALGORITMO (Ejemplo 2):

Caminar una cuadra
Doblar hacia la derecha
Caminar una cuadra
Doblar hacia la derecha
Caminar una cuadra
Doblar hacia la derecha
Caminar una cuadra
Doblar hacia la derecha

Esa solución podemos ver que lo que hacemos es repetir cuatro (4) dos sentencias, en este caso podemos utilizar la estructura de control *repetición*.

por la vereda de la Calle 1, luego la Calle 4, la Calle 3 y por último la Calle 2 llegando nuevamente al lugar de origen.



ALGORITMO (Ejemplo 2): Repetir 4 veces

Caminar una cuadra
Doblar hacia la derecha

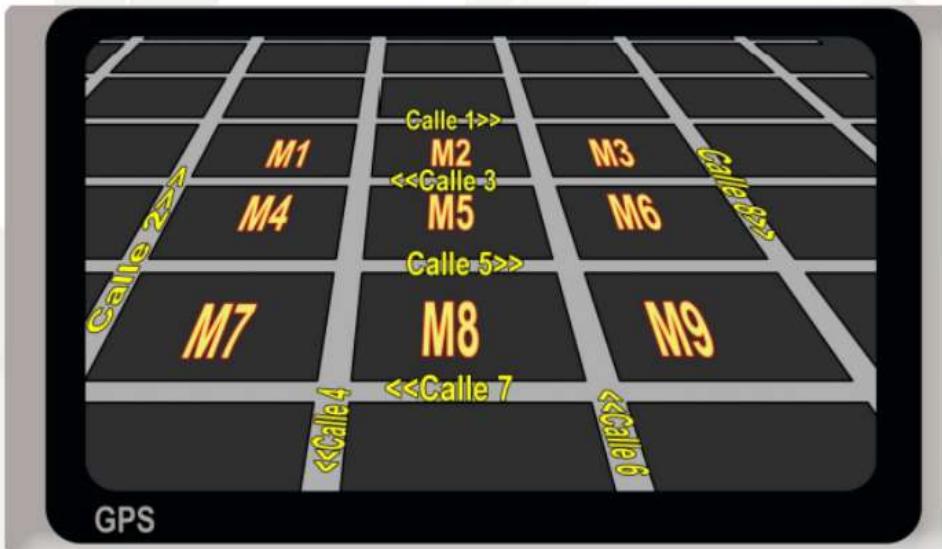
ALGORITMO (Ejemplo 2):

Caminar una cuadra
Doblar hacia la derecha
Caminar una cuadra
Doblar hacia la derecha
Caminar una cuadra
Doblar hacia la derecha
Caminar una cuadra
Doblar hacia la derecha

Esa solución podemos ver que lo que hacemos es repetir cuatro (4) dos sentencias, en este caso podemos utilizar la estructura de control *repetición*.

Ejercitemos con Ada

Pseudocód



Escribir los siguientes algoritmos:

- Ada debe caminar por la calle 1 hasta la esquina de las calles 1 y 8, y luego regresar hasta la esquina de las calles 1 y 6.
- Ada debe caminar por la calle 2 hasta la calle 5, luego caminar 3 cuadras por la calle 5 hasta la esquina de las calles 5 y 8.
- Ada debe caminar por la calle 1 hasta la esquina de las calles 1 y 4, y luego recorrer el perímetro de la manzana 2.
- Ada debe caminar "en escalera". Lo hará por la calle 1, calle 4, calle 3 y calle 6 (ver la imagen a la derecha)

Teniendo en cuenta las sentencias:

Caminar una cuadra

Doblar hacia la derecha

Doblar hacia la izquierda

Cruzar la calle

Las estructuras de control:

Si

Mientras (condición o condiciones)

Repetir X-veces.

Y la condición:

o Pasan autos,



Variable

Constantemente estamos en contacto con valores que cambian en nuestra vida cotidiana, lo cual tiene que ver con los datos que forman parte de la información que se puede manejar.

Por ejemplo la temperatura de un día puede verse como dato que varía, el peso de una persona, la cantidad de amigos en Facebook, y muchos más.

Lo mismo sucede si miramos un partido de basquet. Al comienzo cada equipo tendrá una cantidad de 0 (cero) puntos. A medida que el partido avance los puntos de cada equipo pueden variar (incrementándose cada vez que se convierta un simple, doble o triple). Si llamáramos Puntos del equipo A (a los puntos convertidos por uno de los equipos) y Puntos del equipo B (a los puntos convertidos por el equipo restante) estas serían las variables que se incrementen cada vez que un equipo convierta puntos.



Supongamos que el equipo A convirtiera en el primer cuarto: siete simples (cada simple vale un punto), cuatro dobles (cada doble vale dos puntos) y cinco triples (cada uno valiendo tres puntos). Hagamos la cuenta:

7 simples →
4 dobles →
5 triples →
7 puntos
8 puntos ($4 * 2$)
15 ($5 * 3$)

La suma del total de todos los puntos obtenidos por el equipo A es de 30 para el primer cuarto. Por lo que la variable **Puntos del equipo A** tendría el valor 30.

Luego en el segundo cuarto probablemente ese mismo equipo convertirá más puntos, en ese caso se le sumarán los puntos correspondientes.

Algo similar ocurrirá con la variable **Puntos del equipo B**, cada vez que el equipo B convierta algún punto se incrementará variando su puntaje. De allí el concepto de variable.

Variable

Constantemente estamos en contacto con valores que cambian en nuestra vida cotidiana, lo cual tiene que ver con los datos que forman parte de la información que se puede manejar.

Por ejemplo la temperatura de un día puede verse como dato que varía, el peso de una persona, la cantidad de amigos en Facebook, y muchos más.

Lo mismo sucede si miramos un partido de basquet. Al comienzo cada equipo tendrá una cantidad de 0 (cero) puntos. A medida que el partido avance los puntos de cada equipo pueden variar (incrementándose cada vez que se convierta un simple, doble o triple). Si llamáramos Puntos del equipo A (a los puntos convertidos por uno de los equipos) y Puntos del equipo B (a los puntos convertidos por el equipo restante) estas serían las variables que se incrementen cada vez que un equipo convierta puntos.



Supongamos que el equipo A convirtiera en el primer cuarto: siete simples (cada simple vale un punto), cuatro dobles (cada doble vale dos puntos) y cinco triples (cada uno valiendo tres puntos). Hagamos la cuenta:

- 7 simples →
- 4 dobles →
- 5 triples →
- 7 puntos
- 8 puntos ($4 * 2$)
- 15 ($5 * 3$)

La suma del total de todos los puntos obtenidos por el equipo A es de 30 para el primer cuarto. Por lo que la variable **Puntos del equipo A** tendría el valor 30.

Luego en el segundo cuarto probablemente ese mismo equipo convertirá más puntos, en ese caso se le sumarán los puntos correspondientes. Algo similar ocurrirá con la variable **Puntos del equipo B**, cada vez que el equipo B convierta algún punto se incrementará variando su puntaje. De allí el concepto de variable.

mpie, doble o triple). Si llamaramos Puntos del equipo A (a los puntos convertidos por uno de los equipos) y Puntos del equipo B (a los puntos convertidos por el equipo restante) estas serían las variables que se incrementen cada vez que un equipo convierta puntos.



Supongamos que el equipo A convirtiera en el primer cuarto: siete simples (cada simple vale un punto), cuatro dobles (cada doble vale dos puntos) y cinco triples (cada uno valiendo tres puntos). Hagamos la cuenta:

7 simples →
4 dobles →
5 triples →
7 puntos
8 puntos ($4 * 2$)
15 ($5 * 3$)

La suma del total de todos los puntos obtenidos por el equipo A es de 30 para el primer cuarto. Por lo que la variable **Puntos del equipo A** tendría el valor 30.

Luego en el segundo cuarto probablemente ese mismo equipo convertirá más puntos, en ese caso se le sumarán los puntos correspondientes.

Algo similar ocurrirá con la variable **Puntos del equipo B**, cada vez que el equipo B convierta algún punto se incrementará variando su puntaje.

De allí el concepto de variable.

Más práctica



Escribir el siguiente algoritmo:

- Ada debe recorrer la manzana 1 (M1) por las calles: 1, 4, 3 y 2; contando la cantidad de residuos encontrados e informando al final cuántos residuos se encontraron.

Teniendo en cuenta las sentencias:

- Caminar una cuadra
- Doblar hacia la derecha
- Doblar hacia la izquierda
- Cruzar la calle
- Recoger un residuo
- Colocar en el cesto de residuos
- Observar el estado del semáforo

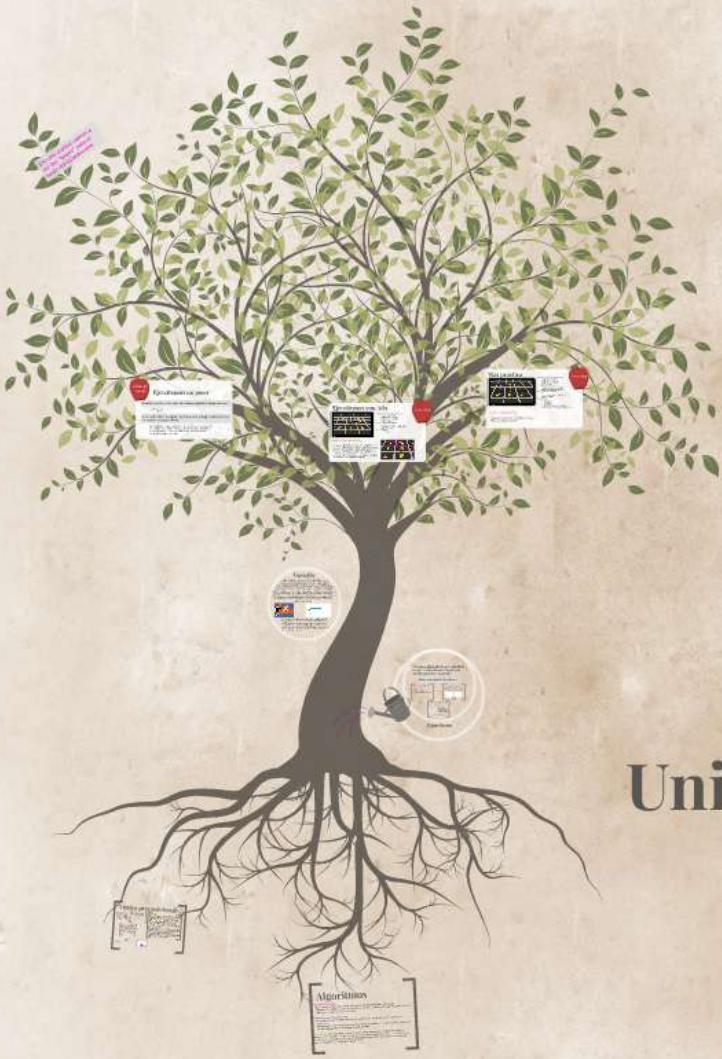
Las estructuras de control:

- Si
- Mientras (condición o condiciones)
- Repetir X-veces.

Y la condición:

- o Pasan autos,
- o Hay árboles,
- o Hay residuos,
- o Hay un local de venta de repuestos,
- o Hay un local de venta de electrodomésticos

Pseudocód



Unidad INTRODUCTORIA

Esta presentación fue diseñada por el siguiente equipo docente:

Mg. Claudia Russo

Lic. Paula Lencina

AC María Lanzillotto

Lic. Leticia Galante

Prog. Trinidad Picco

AC. Patricia Miguel

AC M. del Carmen Muller

Lic. Mariana Adó

Lic. Cecilia Rastelli



Atribución – No Comercial – Compartir Igual (by-nc-sa):

No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original. Esta licencia no es una licencia libre.



Atribución (Attribution): En cualquier explotación de la obra autorizada por la licencia será necesario reconocer la autoría (obligatoria en todos los casos).



No Comercial (Non commercial): La explotación de la obra queda limitada a usos no comerciales.



Compartir Igual (Share alike): La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Por lo que las autoras agradecen el reconocimiento de la autoría correspondiente.
Para mayor información se recomienda acceder a :

<https://creativecommons.org/licenses/by-nc-sa/4.0/>