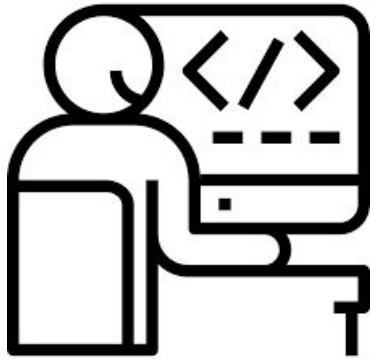


Conceptos de Algoritmos Datos y Programas

Conceptos de Algoritmos Datos y Programas

Lograr que el alumno cuando termine el curso, posea conocimientos, métodos y herramientas para resolver distintos problemas con la computadora logrando:

- Analizar problemas, poniendo énfasis en la **modelización**, **abstracción** y en la **modularización** de los mismos.
- Obtener una expresión sintética, precisa y **documentada** de los problemas y su solución.
- Analizar y expresar correctamente algoritmos, orientando los mismos a la **resolución de las partes** (módulos) en que se descomponen los problemas.
- Introducir las nociones de **estructuras de datos**, **tipos de datos** y **abstracción de datos**.

CADP – **TEMAS**

- Análisis de problemas
- Definiciones Fundamentales
- Modelos + Datos = programa

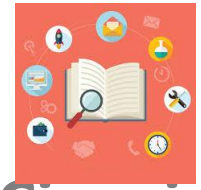
CADP – **DEFINICIONES**

Informática

Es la **ciencia** que estudia el análisis y **resolución de problemas** utilizando **computadoras**.

CADP – **DEFINICIONES**

Es la **ciencia** que estudia el análisis y **resolución de problemas** utilizando **computadoras**.



Ciencia

Se relaciona con una metodología fundamentada y racional para el estudio y resolución de los problemas. En este sentido la Informática se vincula especialmente con la Matemática y la Ingeniería

Resolución

Se puede utilizar las herramientas informáticas en aplicaciones de áreas muy diferentes tales como biología, comercio, control industrial, administración, robótica, educación, arquitectura, etc.

Computadora

Máquina digital y sincrónica, con cierta capacidad de cálculo numérico y lógico controlado por un programa almacenado y con probabilidad de comunicación con el mundo exterior. Ayuda al hombre a realizar tareas repetitivas en menor tiempo y con mayor exactitud. No razona ni crea soluciones, sino que ejecuta una serie de órdenes que le proporciona el ser humano

CADP – **DEFINICIONES**

Informática - Objetivo

Resolver problemas del mundo real utilizando una computadora (utilizando un software)

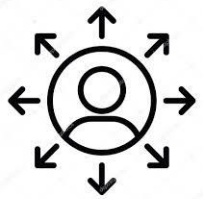
CADP – **PARADIGMAS DE PROGRAMACION**

Imperativo - procedural

En general, los lenguajes de programación pueden ser clasificados a partir del modelo que siguen para DEFINIR y OPERAR información. Este aspecto permite jerarquizarlos según el paradigma que siguen.

CADP – **COMO VAMOS A TRABAJAR**

Poseer un problema



Modelizar el problema

Modularizar la solución

Realizar el programa

Utilizar la computadora

CADP – **COMO VAMOS A TRABAJAR**

Poseer un problema

En el laboratorio se compraron dos robots lego y ahora se quiere que los robots implementen los algoritmos que los alumnos desarrollan con el entorno CMRE.

Cómo es la comunicación?

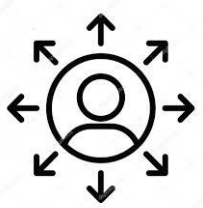
Cómo representamos la ciudad?

Qué consideraciones hay que tener?

ndo aparece la computadora?

Lenguaje?

CADP – **COMO VAMOS A TRABAJAR** → Modelar



El modelo define los mecanismos de interacción y sus condiciones. Establece el efecto sobre la máquina y el usuario. Indica los Informes necesarios.

Pensar que acciones se van a permitir y que implica cada acción permitida

Acciones permitidas para el robot.

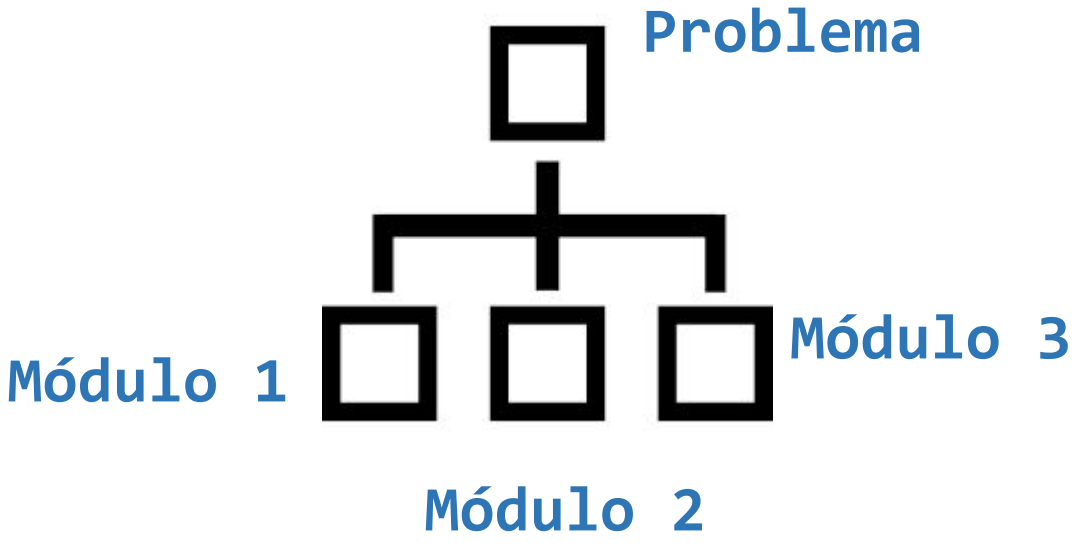
Condiciones para realizarlas.

Requerimientos de la máquina para cada acción.

Efecto de las acciones del robot en la máquina.

CADP – **COMO VAMOS A TRABAJAR** → Modularizar

A partir del modelo es necesario encontrar la forma de descomponer en partes (módulos) para obtener una solución.



La descomposición funcional de todas las acciones que propone el modelo nos ayudará a reducir la complejidad, a distribuir el trabajo y en el futuro a reutilizar los módulos.

CADP – **COMO VAMOS A TRABAJAR** → Realizar el programa

Una vez que se tiene la descomposición en funciones / procesos o módulos, debemos diseñar su implementación: esto requiere escribir el programa y elegir los datos a representar.

PROGRAMA = **Algoritmo** **+** **Datos**

Las instrucciones (que también se han denominado acciones) representan las operaciones que ejecutará la computadora al interpretar el programa. Un conjunto de instrucciones forma un algoritmo.

Los datos son los valores de información de los que se necesita disponer y en ocasiones transformar para ejecutar la función del programa.

CADP – **COMO VAMOS A TRABAJAR** → Realizar el programa

ALGORITMO

Especificación rigurosa de la secuencia de pasos (instrucciones) a realizar sobre un autómata para alcanzar un resultado deseado en un tiempo finito.

→ **Alcanzar el resultado en tiempo finito:** suponemos que un algoritmo comienza y termina. Está implícito que el número de instrucciones debe ser también finito.

→ **Especificación rigurosa:** que debemos expresar un algoritmo en forma clara y unívoca.

→ Si el **autómata** es una computadora, tendremos que escribir el algoritmo en un lenguaje “entendible” y ejecutable por la máquina.

CADP – **COMO VAMOS A TRABAJAR** → Realizar el programa

DATO

Es una representación de un objeto del mundo real mediante la cual podemos modelizar aspectos del problema que se quiere resolver con un programa sobre una computadora. Puede ser constante o variable.

→

Los pasos que realiza el robot en un recorrido

Las flores que hay en una esquina

Una imagen

El peso de una persona, el nombre, el dni, etc.

Qué características tiene el programa?

CADP – **COMO VAMOS A TRABAJAR**

Realizar el programa

Para el Desarrollador

- **Operatividad**: El programa debe realizar la función para la que fue concebido.
- **Legibilidad** : El código fuente de un programa debe ser fácil de leer y entender. Esto obliga a acompañar a las instrucciones con comentarios adecuados.
- **Organización**: El código de un programa debe estar descompuesto en módulos que cumplan las subfunciones del sistema.
- **Documentados:** Todo el proceso de análisis y diseño del problema y su solución debe estar documentado mediante texto y/o gráficos para favorecer la comprensión, la modificación y la adaptación a nuevas funciones.

Para la Computadora

- Debe contener instrucciones válidas.
- Deben terminar.
- No deben utilizar recursos inexistentes.

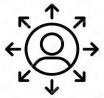
CADP – **COMO VAMOS A TRABAJAR** → Utilizar la computadora

COMPUTADORA

Máquina capaz de aceptar datos de entrada, ejecutar con ellos cálculos aritméticos y lógicos y dar información de salida (resultados), bajo control de un programa previamente almacenado en su memoria.

En cuál de todas las etapas apareció el lenguaje?

Poseer un problema

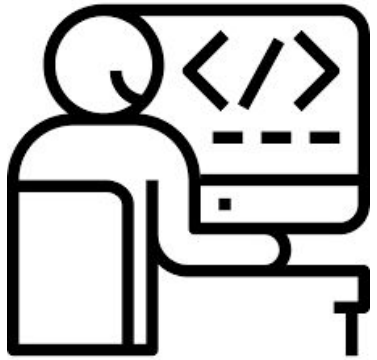


Modelizar el problema

Modularizar la solución

Realizar el programa

Utilizar la computadora



Conceptos de Algoritmos Datos y Programas

CADP – **TEMAS DE LA CLASE DE HOY**

- Tipos de datos
- Tipos de datos: entero real y lógico
- Operaciones

CADP – **TIPO DE DATOS**

DATO

Es una clase de objetos de datos ligados a un conjunto de operaciones para crearlos y manipularlos.

Tienen un rango de valores posibles

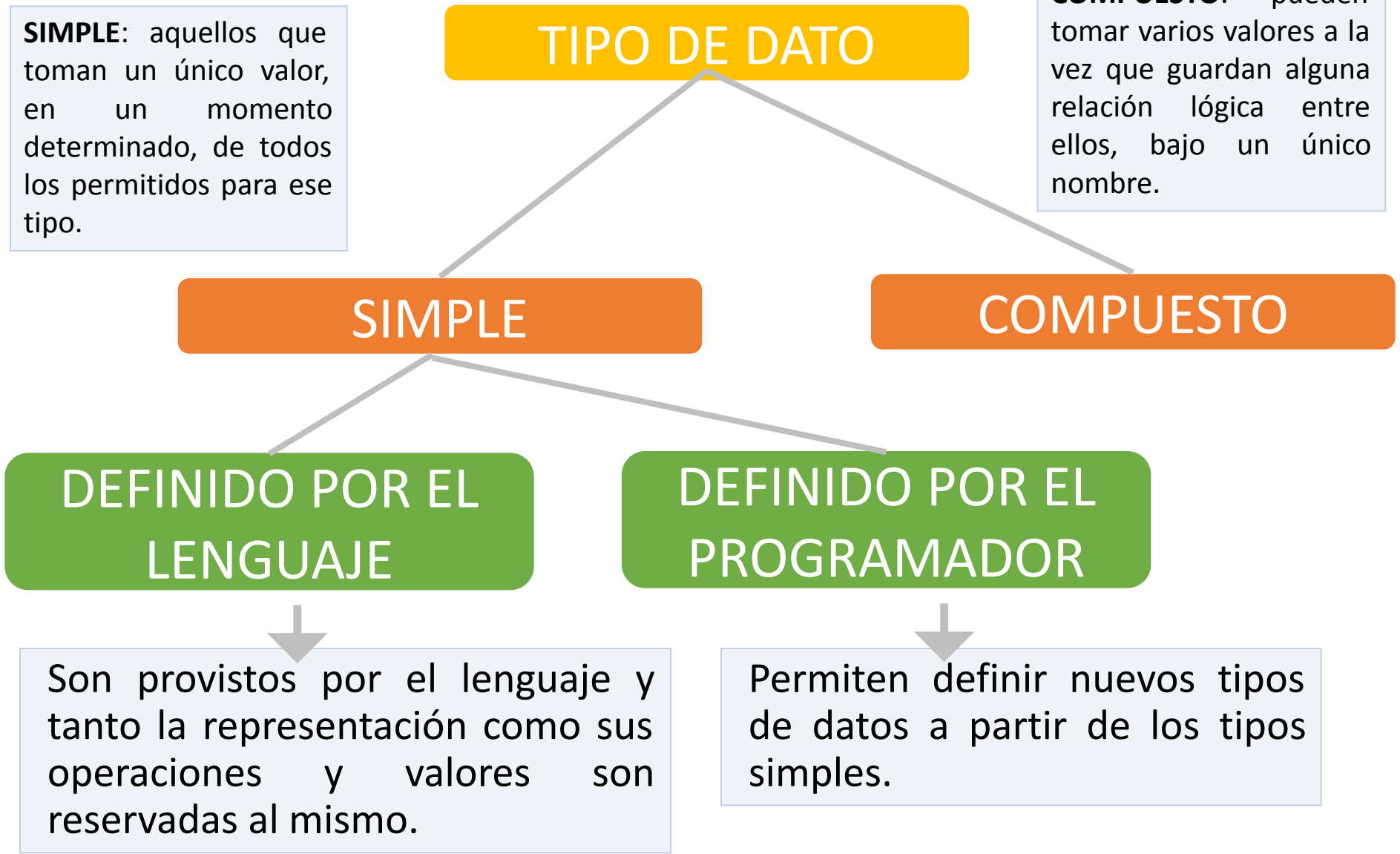
Tienen un conjunto de operaciones permitidas

Tienen una representación interna

CADP – **TIPO DE DATOS - Clasificación**

SIMPLE: aquellos que toman un único valor, en un momento determinado, de todos los permitidos para ese tipo.

COMPUESTO: pueden tomar varios valores a la vez que guardan alguna relación lógica entre ellos, bajo un único nombre.



CADP – **TIPO DE DATOS**

DATO NUMERICO

Representa el conjunto de números que se pueden necesitar.
Estos números pueden ser enteros o reales.

Tipo de datos
entero

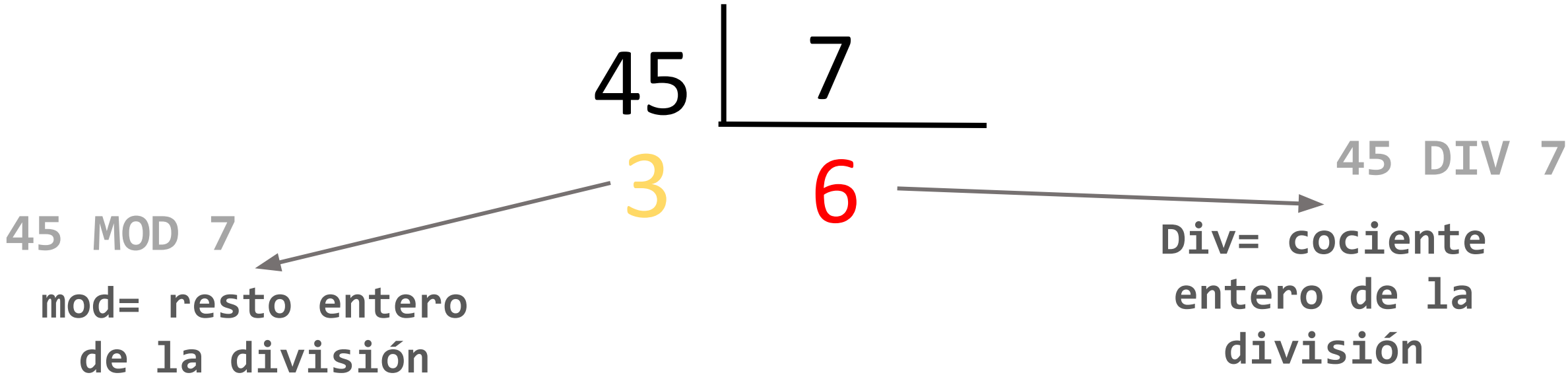
- Es un tipo de dato simple, ordinal
- Los valores son de la forma
  -10 , 200, -3000, 2560
- Al tener una representación interna, tienen un número mínimo y uno máximo que puede representar

Operaciones

Operadores Matemáticos	Operadores Lógicos	Operadores Enteros
• + • - • * • /	• < • > • = • <= • =>	• Mod • Div

Qué es div y mod?

CADP – **TIPO DE DATOS** DATO NUMERICO - ENTEROS



```
Supongamos a,b,c,d variables enteras
a:= 22
b:= 6
c:= a DIV b;   3 //no se tienen en cuenta los decimales
d:= a MOD c;   4
```

DATO NUMERICO

Representa el conjunto de números que se pueden necesitar.
Estos números pueden ser enteros o reales.

Tipo de datos
real

- Es un tipo de dato simple, permiten representar números con decimales
- Los valores son de la forma
  -10 , 200, -3000, 2560, 11.5, -22.89
- Al tener una representación interna, tienen un número mínimo y uno máximo

CADP – **TIPO DE DATOS** DATO NUMERICO - REAL

Operaciones

NO pueden utilizar DIV y MOD

Operadores Matemáticos

- +
- -
- *
- /

Operadores Lógicos

- <
- >
- =
- <=
- =>

CADP – **TIPO DE DATOS** EJERCICIO

Pensar tres casos en los cuales para representar los datos utilizaría un número real y no un entero.

Pensar un caso en el cual para representar el dato utilizaría un número entero y no un real.

CADP – **Tipos de Datos** DATO NUMERICO - PARENTESIS

Las expresiones que tienen dos o más operandos requieren reglas matemáticas que permitan determinar el orden de las operaciones.

El orden de precedencia para la resolución, ya conocido, es:

1. operadores ***, /, div y mod**
2. operadores **+, -**

En caso que el orden de precedencia natural deba ser alterado, es posible la utilización de paréntesis dentro de la expresión.

```
Supongamos a,b,c,d variables enteras
```

```
a:= 22  b:= 6
c := a DIV b +3 * 2;
d:= a DIV (b + 3 * 2);
```

CADP – **TIPOS DE DATOS**

DATO LOGICO

Permite representar datos que pueden tomar dos valores verdadero o falso.

Tipo de datos

lógico

Es un tipo de dato simple, ordinal

Los valores son de la forma

true = verdadero

false = falso

CADP – **Tipos de Datos** DATO LOGICO

Operaciones

Operadores Lógicos

- and (conjunción)
- or (disyunción)
- not (negación)

CADP – **Tipos de Datos** DATO LOGICO

Operaciones

V	V	V
F	F	F
V	F	F
F	V	F

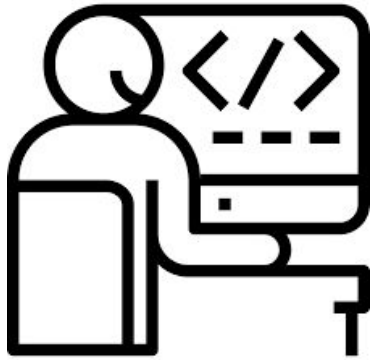
Conjunción

V	V	V
F	F	F
V	F	V
F	V	V

Disyunción

V	F
F	V

Negación



Conceptos de Algoritmos Datos y Programas

CADP – **TEMAS**

- Tipos de datos
- Tipo de dato char, string
- Operaciones
- Variables y constantes

CADP – **TIPOS DE DATOS**

DATO CARACTER

Representa un conjunto finito y ordenado de caracteres que la computadora reconoce. Un dato de tipo caracter contiene solo un caracter.

Tipo de datos

caracter

Es un tipo de dato simple, ordinal

Los valores son de la forma

a B ! $ L 4

CADP – **TIPOS DE DATOS** DATO CARACTER

Operaciones

Operadores Lógicos

- <, <=
- >, =>
- =
- <>

La Tabla ASCII contiene todos los caracteres y el orden entre los mismos. **http://ascii.cl/es/**

CADP – **TIPOS DE DATOS**

DATO STRING

Representa un conjunto finito de caracteres. Como máximo representa 256 caracteres. En general se utilizan para representar nombres.

Tipo de datos

string

- Es un tipo de dato compuesto
- Los valores son de la forma
  casa /erML

CADP – **TIPOS DE DATOS** DATO STRING

Operaciones

Operadores Lógicos

- <, <=
- >, =>
- =
- <>
- Existen otras pero no las veremos

CADP – **TIPOS DE DATOS** DATO STRING

Supongamos que se tiene x,y variables string

x:= ‘ABC’ y:=‘aBC’

x = “abc” ?

x = y ?

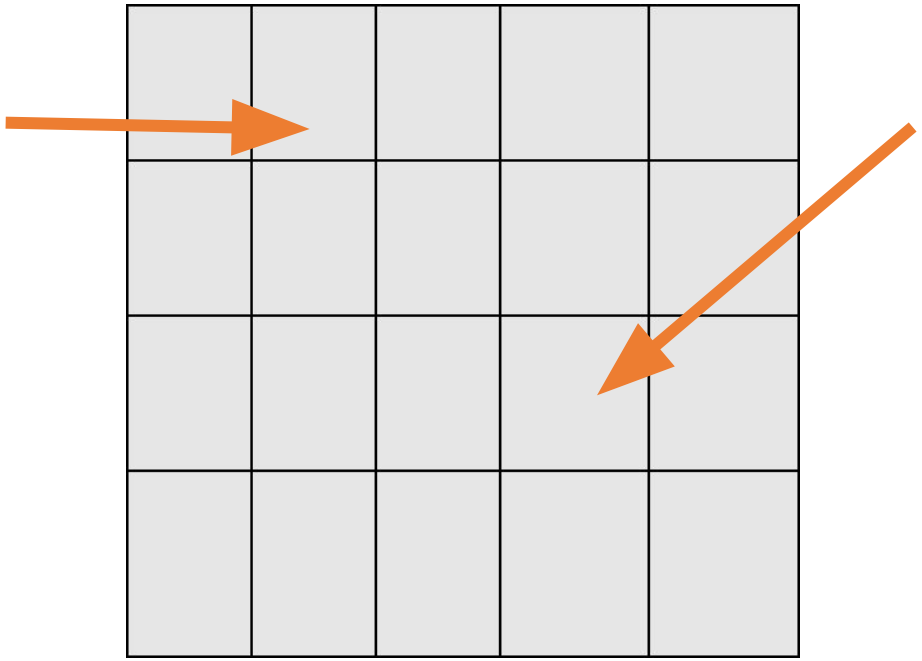
x > y ?

Qué resultado dan estas operaciones ?

CADP – **VARIABLES - CONSTANTES**

Variable
NOMBRE

Referencia una zona de memoria



Constante
NOMBRE

Referencia una zona de memoria

En qué se diferencian?

CADP – **VARIABLES - CONSTANTES**

Variables

Es una zona de memoria cuyo contenido va a ser alguno de los tipos mencionados anteriormente. La dirección inicial de esta zona se asocia con el nombre de la variable.
Puede cambiar su valor durante el programa.

Constantes

Es una zona de memoria cuyo contenido va a ser alguno de los tipos mencionados anteriormente. La dirección inicial de esta zona se asocia con el nombre de la variable.
NO puede cambiar su valor durante el programa.

CADP – **VARIABLES - CONSTANTES**

Qué información para resolver los ejercicios del curso hubiera sido útil declararla como constante?.

CADP – **TIPOS DE DATOS** RECORDAR

Los diferentes tipos de datos deben especificarse y a esta especificación dentro de un programa se la conoce como declaración.

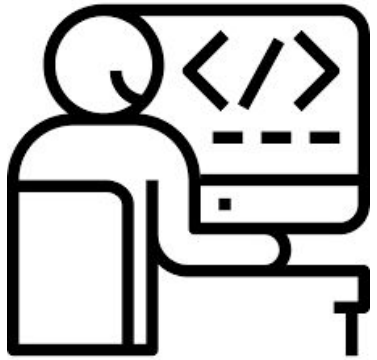
Una vez declarado un tipo podemos asociar al mismo variables, es decir nombres simbólicos que pueden tomar los valores característicos del tipo.

Algunos lenguajes exigen que se especifique a qué tipo pertenece cada una de las variables. Verifican que el tipo de los datos asignados a esa variable se correspondan con su definición. Esta clase de lenguajes se denomina fuertemente tipados (strongly typed).

Otra clase de lenguajes, que verifica el tipo de las variables según su nombre, se denomina auto tipados (self typed).

Otra clase de lenguajes, que verifica el tipo de las variables según su nombre, se denomina auto tipados (self typed).

Existe una tercera clase de lenguajes que permiten que una variable tome valores de distinto tipo durante la ejecución de un programa. Esta se denomina dinámicamente tipados (dinamically typed).



Conceptos de Algoritmos Datos y Programas

CADP – **TEMAS**

- Estructura de un programa
- Pre y post condiciones
- Operaciones de read y write

CADP – **HASTA AHORA LOS PROGRAMAS**

```
Programa nombre
areas
Procesos
  proceso nombre
   variables
   comenzar
   fin
variables
comenzar
fin
```

Y ahora cómo escribimos un programa?

CADP –**PROGRAMAS AHORA**

```
Program nombre;

Const
    .....

módulos {luego veremos como se declaran}

Var

begin
   ...
end.
```

Constantes del programa

Módulos del programa

Variables del programa

Cuerpo del programa

CADP –**PROGRAMAS AHORA**

Constantes del programa

Variables del programa

Cuerpo del programa

```
Program nombre;
Const
    N = 25;
    pi = 3.14;

módulos {luego veremos como se
  declaran}

var
edad: integer;
peso: real;
letra: char;
resultado: boolean;

begin
  edad:= 5;
  peso:= -63.5;
  edad:= edad + N;
  letra:= ‘A’;
  resultado:= letra = ‘a’;
end
```

CADP – **PRE y POST CONDICIONES**

PRE CONDICON

Es la información que se conoce como verdadera antes de iniciar el programa (ó módulo).

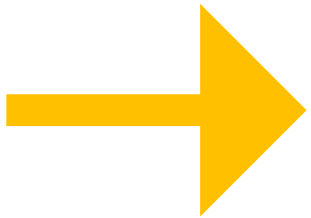
POST CONDICON

es la información que debería ser verdadera al concluir el programa (ó módulo), si se cumplen adecuadamente los pasos especificados.

CADP – **LECTURA / ESCRITURA** READ

HASTA AHORA

```
Program uno;

var
  edad: integer;
  valor: integer;

begin
  edad:=  5;

  valor:=  edad + 15;
end.
```



```
Program uno;

var
  edad: integer;
  valor: integer;
  suma:integer;
begin
  read (edad);
  valor:= 9;
  suma:= edad + valor;
end.
```

Cómo funciona el read?

READ

Es una operación que contienen la mayoría de los lenguajes de programación. Se usa para tomar datos desde un dispositivo de entrada (por defecto desde teclado) y asignarlos a las variables correspondientes.

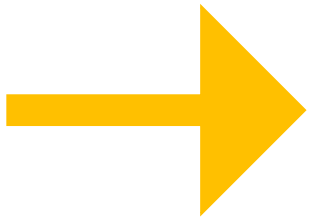
```
Program uno;
var
   cant: integer;
Begin
  read (cant);
End.
```

El usuario ingresa un valor, y ese valor se guarda en la variable asociada a la operación read.

CADP – **LECTURA / ESCRITURA** WRITE

HASTA AHORA

```
Program uno;

var
  edad: integer;
  valor: integer;

begin
  edad:=   5;

  Informar(edad);
end.
```



```
Program uno;

var
  edad: integer;
  valor: integer;

begin
  read (edad);
  valor:= edad + 15;
  write (valor);
end.
```

Cómo funciona el write?

WRITE

Es una operación que contienen la mayoría de los lenguajes de programación. Se usa para mostrar el contenido de una variable, por defecto en pantalla.

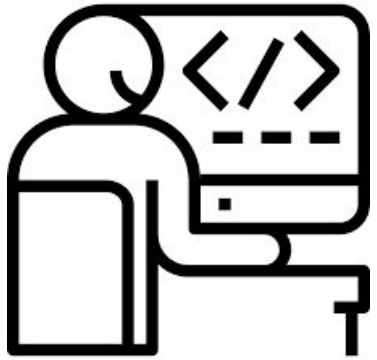
```
Program uno;
var
   cant: integer;
Begin
  read (cant);
  cant:= cant + 1;
  write (cant);
End.
```

Variantes del write?

El valor almacenado en la variable asociada a la operación write, se muestra en pantalla.

CADP – **LECTURA / ESCRITURA** WRITE

```
Program uno;
var
  ...
Begin
 ...
 write (“texto”);      Write (“Los valores ingresados son 0”)

 write (variable);  Write (num);

 write (“texto”,variable);   Write (“El resultado es:”,num);

 write (“texto”, resultado de una operación);  Write (“El resultado
                                               es:”,num+4);
End.
```

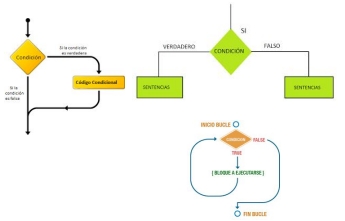


Conceptos de Algoritmos Datos y Programas

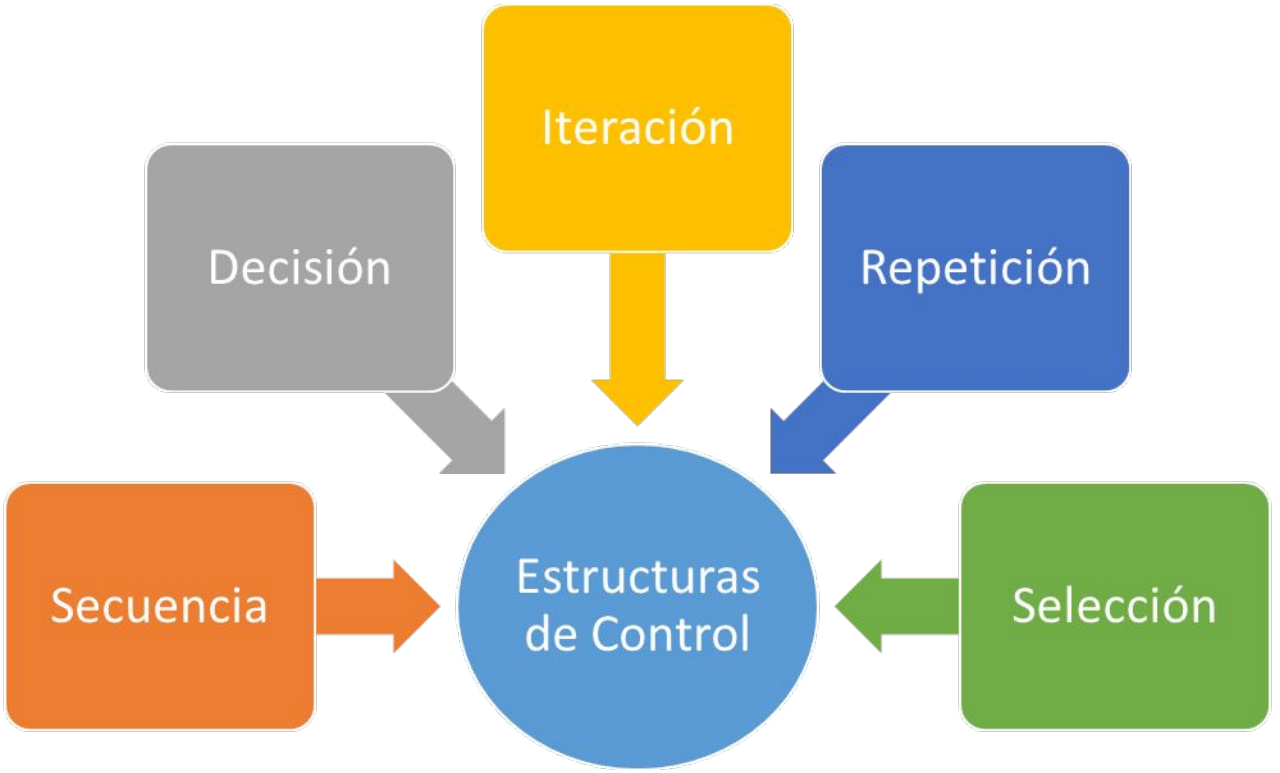
CADP – **TEMAS**

- Estructura de control
- Estructura de secuencia
- Estructura de control de decisión IF
- Estructura de control de selección CASE

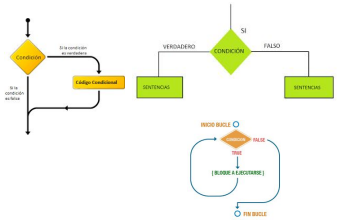
CADP – **ESTRUCTURAS DE CONTROL**



Todos los lenguajes de programación tienen un conjunto mínimo de instrucciones que permiten especificar el control del algoritmo que se quiere implementar. Como mínimo deben contener: secuencia, decisión e iteración.



CADP – **ESTRUCTURAS DE CONTROL**

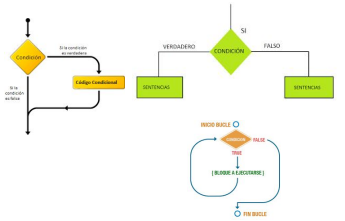


SECUENCIA

La estructura de control más simple, está representada por una sucesión de operaciones (por ej. asignaciones), en la que el orden de ejecución coincide con el orden físico de aparición de las instrucciones.

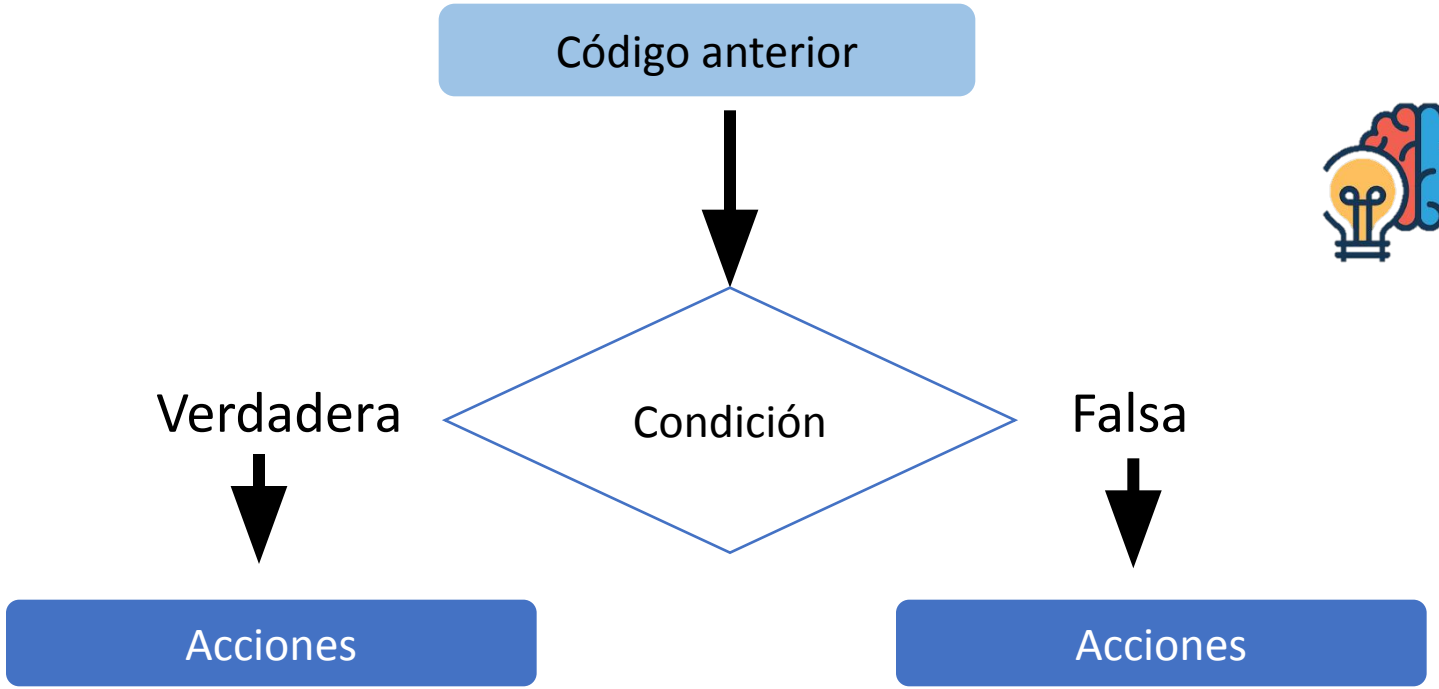
```
Program uno;
...
var
   num:integer;
begin
   read (num);
   write (num);
end.
```

CADP – **ESTRUCTURAS DE CONTROL**



DECISION

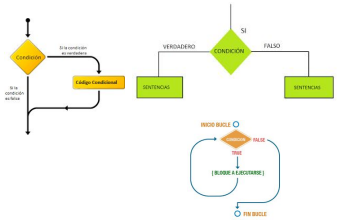
En un algoritmo representativo de un problema real es necesario tomar decisiones en función de los datos del problema. La estructura básica de decisión entre dos alternativas es la que se representa simbólicamente:



A qué estructura de control vista en el entorno del robot se parece?.

Cómo es la sintaxis?

CADP – **ESTRUCTURAS DE CONTROL** DECISION



```
if (condición) then
  accion;
```

más de
una acción

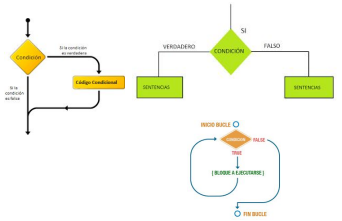
```
if (condición) then
 begin
  acción 1;
  acción 2;
 end;
```

```
if (condición) then
  acción 1
else
  acción 2;
```

```
if (condición) then
 begin
  acción 1;
  acción 2;
 end
else
  acción 3;
```

```
if (condición) then
 begin
  acción 1;
  acción 2;
 end
else
  begin
   acción 3;
   acción 4;
  end;
```

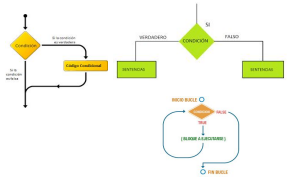
CADP – **ESTRUCTURAS DE CONTROL** DECISION



Realice un programa que lea dos números enteros e informe si la suma de los mismos es mayor a 20.

- Cómo leo un número
- Cómo veo si la suma es > 20
- Cómo muestro el resultado

CADP – **ESTRUCTURAS DE CONTROL** DECISION

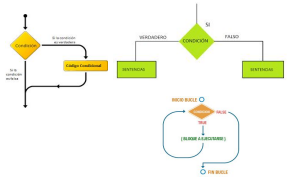


Realice un programa que lea dos números enteros e informe si la suma de los mismos es mayor a 20.

```
Program uno;
var
   num1,num2,suma:integer;

begin
   read (num1);
   read (num2);
   suma:= num1 + num2;
   if (suma > 20)
   then
      write (“La suma supera 20”)
   else
      write (“La suma NO supera 20”);
end.
```

Otra forma?

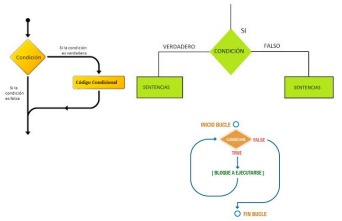
CADP – **ESTRUCTURAS DE CONTROL** DECISION



Realice un programa que lea dos números enteros e informe si la suma de los mismos es mayor a 20.

```
Program uno;
var
   num1,num2:integer;

begin
   read (num1);
   read (num2);
   if ((num1+num2) > 20)
   then
      write (“La suma supera 20”)
   else
      write (“La suma NO supera 20”);
end.
```

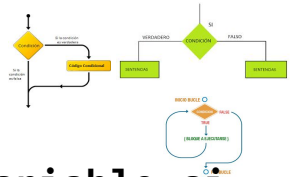
CADP – **ESTRUCTURAS DE CONTROL** DECISION



Realice un programa que lea un número (suponga > 0) y asigne el valor 10 a una variable si el número es menor a 10; asigne 50 a la misma variable si el número es mayor a 10 pero menor que 50; y 100 si el número es mayor a 50.

- Cómo leo un número
- Cómo verifico en que rango está
- Cómo muestro el resultado

CADP – **ESTRUCTURAS DE CONTROL** DECISION

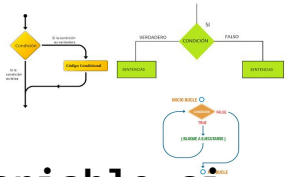


Realice un programa que lea un número (suponga > 0) y asigne el valor 10 a una variable si el número es menor a 10; asigne 50 a la misma variable si el número es mayor a 10 pero menor que 50; y 100 si el número es mayor a 50.

```
Program uno;
var
   num1,resultado:integer;

begin
   read (num1);
   if (num1 <= 10) then resultado:= 10;

   if (num1 > 10) and (num1 <= 50) then resultado:= 50;

   if (num1 > 50) then resultado:= 100;
   write (resultado);
end.
```

Es la mejor solución?

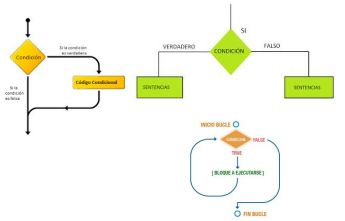
CADP – **ESTRUCTURAS DE CONTROL** DECISION



Realice un programa que lea un número (suponga > 0) y asigne el valor 10 a una variable si el número es menor a 10; asigne 50 a la misma variable si el número es mayor a 10 pero menor que 50; y 100 si el número es mayor a 50.

```
Program uno;
var
   num1,resultado:integer;

begin
   read (num1);
   if (num1 <= 10) then resultado:= 10
   else
     if (num1 > 10) and (num1 <= 50) then resultado:= 50
     else
       resultado:= 100;
   write (resultado);
end.
```

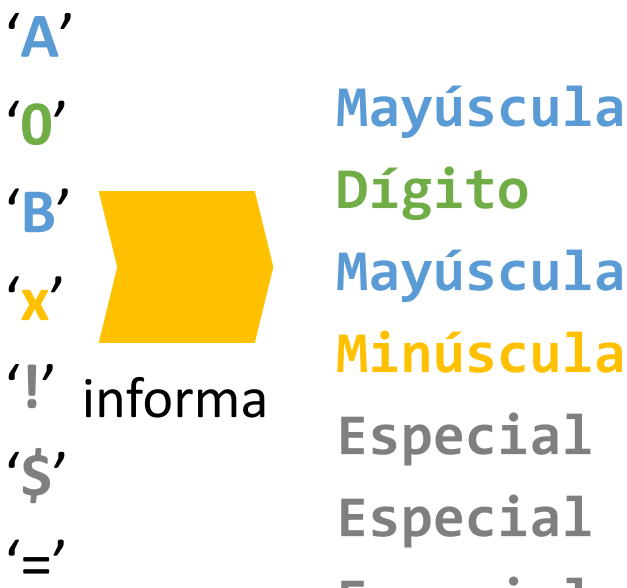
CADP – **ESTRUCTURAS DE CONTROL** SELECCION



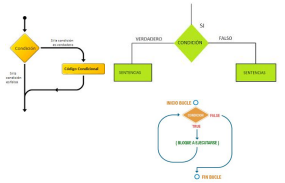
Realizar un programa que lea un caracter y al finalizar informe si se leyó un caracter mayúscula, minúscula, dígito, y ó especiales ha leído.

- Qué tipo de datos leo?

- Cómo identifico que carácter es?



CADP – **ESTRUCTURAS DE CONTROL** SELECCION

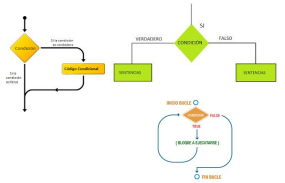


```
Program uno;
var
   car:char;

begin
   read (car);

  if (car = ‘a’)  or (car = ‘b’)… or (car = ‘z’) then
     write (“minúscula”)
  if (car = ‘A’)  or (car = ‘B’)… or (car = ‘Z’) then
       write (“mayúscula”);
  if (car = ‘0’)  or (car = ‘1’)… or (car = ‘9’) then
       write (“numero”);
  if (car = ‘@’)  or (car = ‘!’)… or (car = ‘*’) then
       write (“especial”);

End.
```

Se puede mejorar?

Si sólo existe el if

CADP – **ESTRUCTURAS DE CONTROL** SELECCION



```
Program uno;
var
   car:char;
begin
   read (car);
   if (car = ‘a’)  or (car = ‘b’)... or (car = ‘z’) then
       write (“minúscula”);
   else
   if (car = ‘A’)  or (car = ‘B’)... or (car = ‘Z’) then
       write (“mayúscula”);
   else
   if (car = ‘0’)  or (car = ‘1’)... or (car = ‘9’) then
       write (“digito”);
   else  if (car = ‘@’)  or (car = ‘!’)... or (car = ‘*’)
then
        write (“especial”);
End.
```

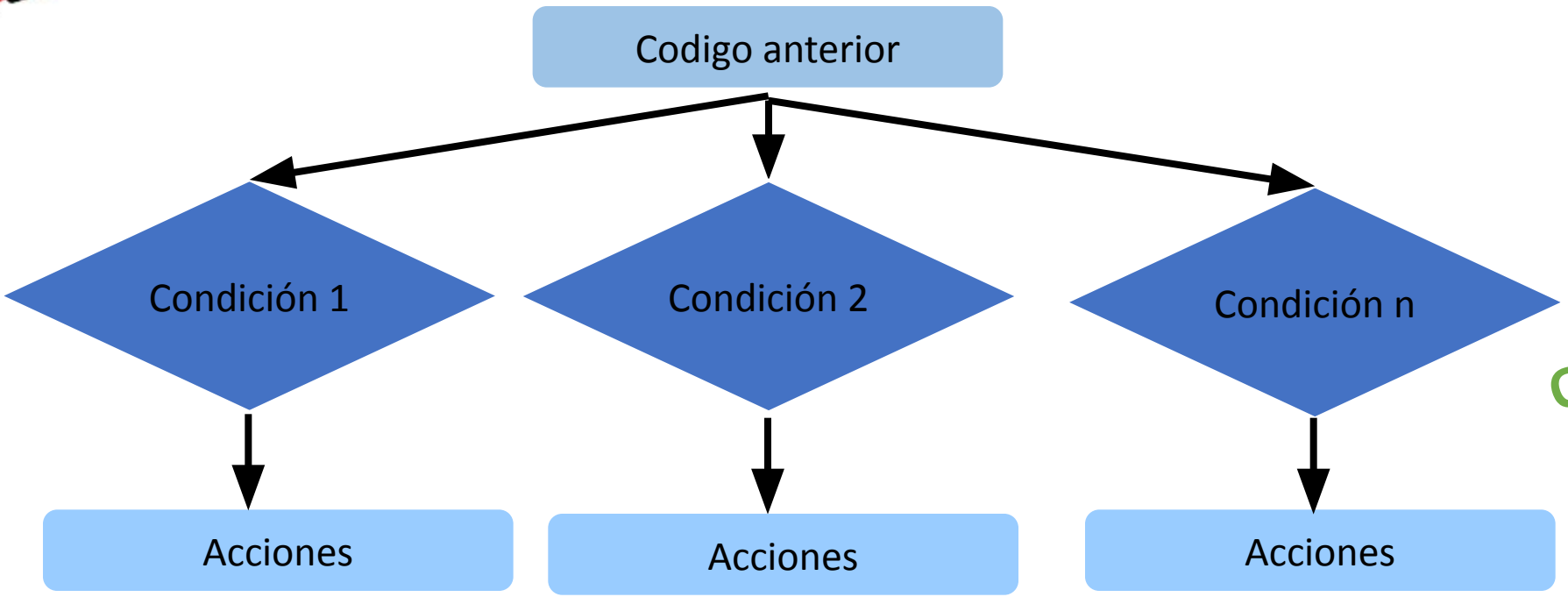
Se puede mejorar?

Si sólo existe el if

CADP – **ESTRUCTURAS DE CONTROL**

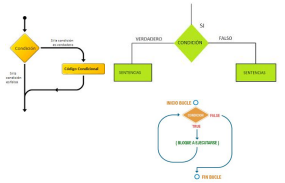
SELECCION

Permite realizar distintas acciones dependiendo del valor de una variable de tipo ordinal.



Cómo es la sintaxis?

CADP – **ESTRUCTURAS DE CONTROL** SELECCION

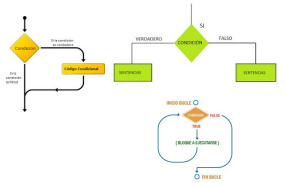


```
case (variable) of
  condicion1:  accion1;
  condición 2: acción2;
  …..
  condición n: acción n;
end;
```

más de una acción

```
case (variable) of
  condicion1:   accion1;
  condición 2:  begin
                  acción2;
                  accion3;
                end;
  …..
  condición n:    accion;
end;
```

CADP – **ESTRUCTURAS DE CONTROL** SELECCION



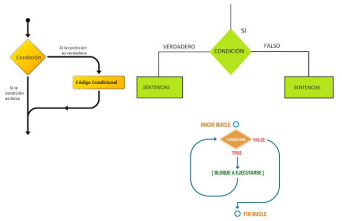
```
Program uno;
var
   car:char;
begin
   read (car);
   case car of
     car = ‘a’: write(“minúscula”);
       …
     car = ‘z’: write(“minúscula”);

     car = ‘A’: write(“mayúscula”);
          …
     car = ‘Z’: write(“mayúscula”);

      car = ‘0’: write(“dígito”);
      …
      car = ‘9’: write(“dígito”);
      else  write(“especial”);
    end;
End.
```

Se puede mejorar?

CADP – **ESTRUCTURAS DE CONTROL** SELECCION

```
Program uno;
var
   car:char;
begin
   read (car);
   case car of
         ‘a’.. ‘z’: write (“minúscula”);
         ‘A’.. ‘Z’: (“mayúscula”);
         ‘0’.. ‘9’: (“dígito”);
          else (“especial”);
      end;
End.
```

CADP – **ESTRUCTURAS DE CONTROL** SELECCION



- La variable del case debe ser de tipo ordinal
- Las opciones deben ser disjuntas

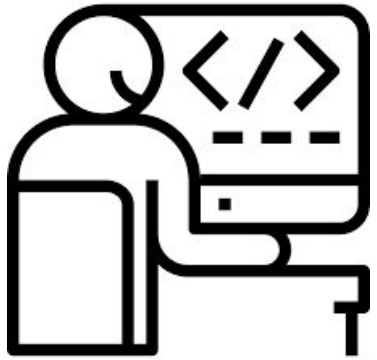
CADP – **Estructuras de control**

Problema: se leen valores de alturas de personas, hasta leer la altura 1.59. Informar la cantidad de personas que miden entre 1.00 y 1.30; la cantidad de personas que miden entre 1.31 y 1.50; la cantidad de personas que miden entre 1.51 y 1.89 y las que miden más de 1.89

El alumno 1: utiliza una variable real para leer las alturas y cuatro contadores para contar la cantidad de personas en cada rango.
Además utiliza un while como estructura de control principal y adentro utiliza un case que incluye los rangos de alturas para saber cual contador sumar. Al final informa los valores de los contadores.

El alumno 2: utiliza una variable real para leer las alturas y cuatro contadores para contar la cantidad de personas en cada rango.
Además utiliza un while como estructura de control principal y adentro utiliza un if con else para saber cual contador sumar. Al final informa los valores de los contadores.

Ambas soluciones son correctas?

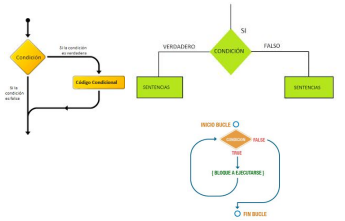


Conceptos de Algoritmos Datos y Programas

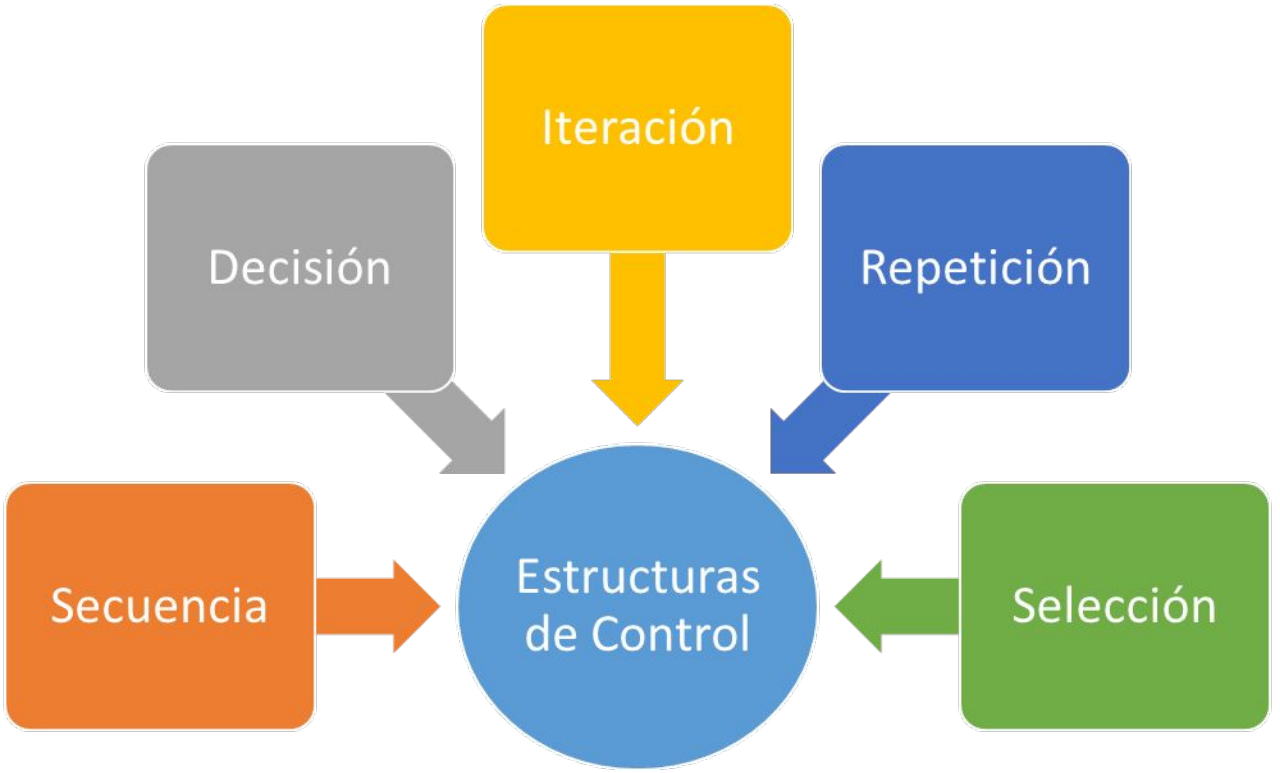
CADP – **TEMAS**

- Estructura de control
- Estructura de iteración
- Estructura de control WHILE y REPEAT UNTIL

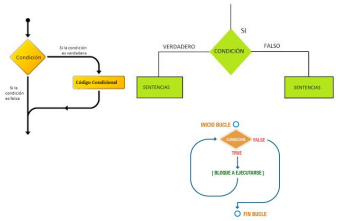
CADP – **ESTRUCTURAS DE CONTROL**



Todos los lenguajes de programación tienen un conjunto mínimo de instrucciones que permiten especificar el control del algoritmo que se quiere implementar. Como mínimo deben contener: secuencia, decisión e iteración.



CADP – **ESTRUCTURAS DE CONTROL**



ITERACION

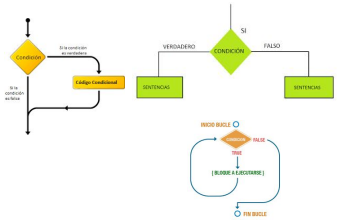
Puede ocurrir que se desee ejecutar un bloque de instrucciones desconociendo el número exacto de veces que se ejecutan.

Para estos casos existen en la mayoría de los lenguajes de programación estructurada las estructuras de control iterativas condicionales.

Como su nombre lo indica las acciones se ejecutan dependiendo de la evaluación de la condición.

Estas estructuras se clasifican en pre-condicionales y post-condicionales.

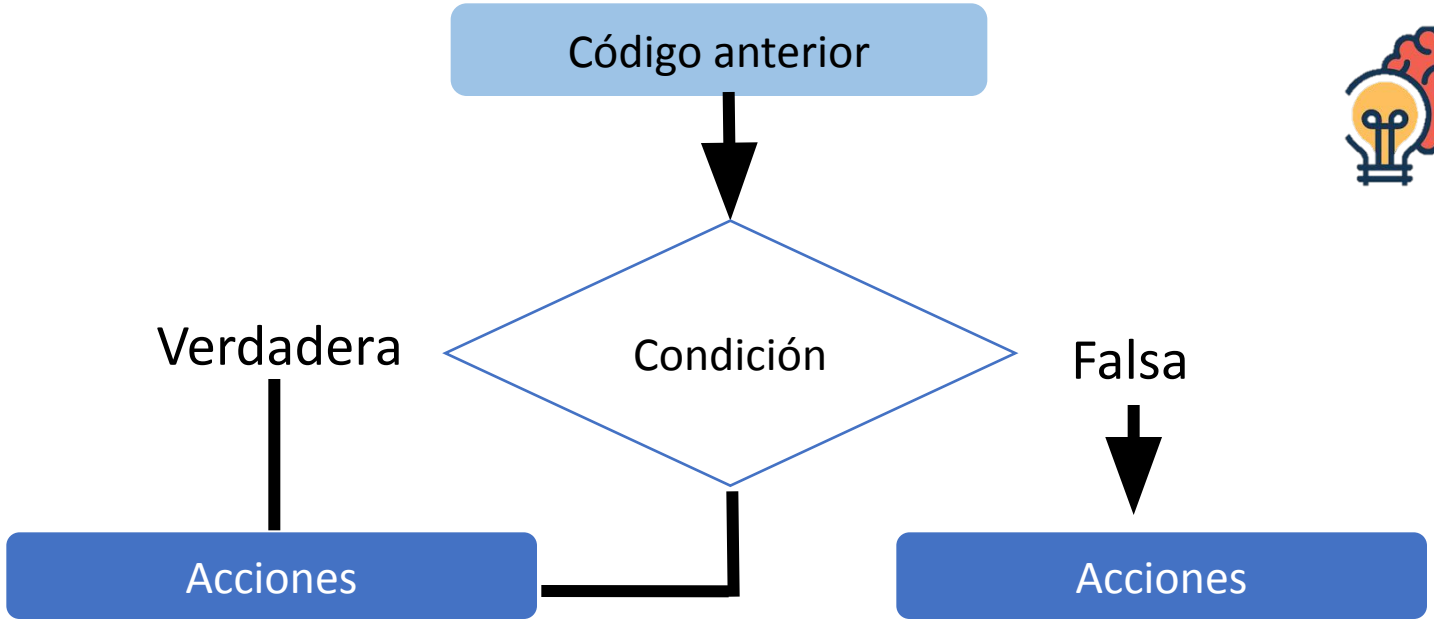
CADP – **ESTRUCTURAS DE CONTROL**



ITERACION - PRECONDICIONAL

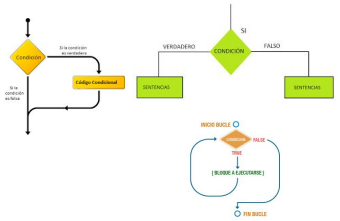
Evalúan la condición y si es verdadera se ejecuta el bloque de acciones. Dicho bloque se pueda ejecutar 0, 1 ó más veces.

Importante: el valor inicial de la condición debe ser conocido o evaluable antes de la evaluación de la condición.



A qué estructura de control vista en el entorno del robot se parece?.

Cómo es la sintaxis?



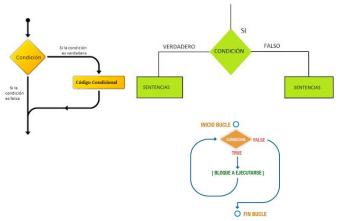
ITERACION - PRECONDICIONAL

```
while (condición) do
   accion;
```

más de una acción

```
while(condición) do
 begin
  acción 1;
  acción 2;
 end;
```

CADP – **ESTRUCTURAS DE CONTROL** ITERACION



Realizar un programa que lea códigos de productos hasta leer un código igual a 30. Al finalizar informe la cantidad de productos con código par.

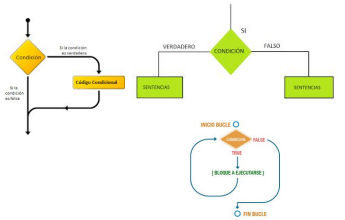
- Cómo leo un código
- Cómo veo si es par
- Cuál es la condición de fin
- Cómo muestro el resultado

35
70
32
5
30

informa

2

CADP – **ESTRUCTURAS DE CONTROL** ITERACION



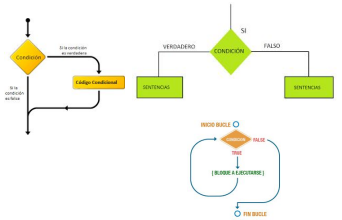
```
Program uno;
var
  resto,prod:integer;
  total:integer;


begin
   total:=0;
   while (prod <> 30)do
    begin
      read(prod);
      resto:= prod MOD 2;
      if (resto = 0)then
        total:= total + 1;
    end;
   write (total);
end.
```

Cuál es el error?

```
Program dos;
var
  prod,resto:integer;
  total:integer;

begin
   total:=0;
   read (prod);
   while (prod <> 30)do
    begin
      resto:= prod MOD 2;
      if (resto = 0)then
        total:= total + 1;
      read (prod);
    end;
   write (total);
end.
```

Otra alternativa?

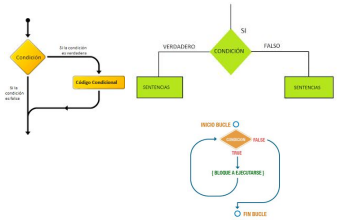
CADP – **ESTRUCTURAS DE CONTROL** ITERACION



```
Program dos;
var
  prod:integer;
  total:integer;

begin
   total:=0;
   read (prod);
   while (prod <> 30)do
    begin
     if (prod MOD 2 = 0)then
       total:= total + 1;
     read (prod);
    end;
   write (total);
end.
```

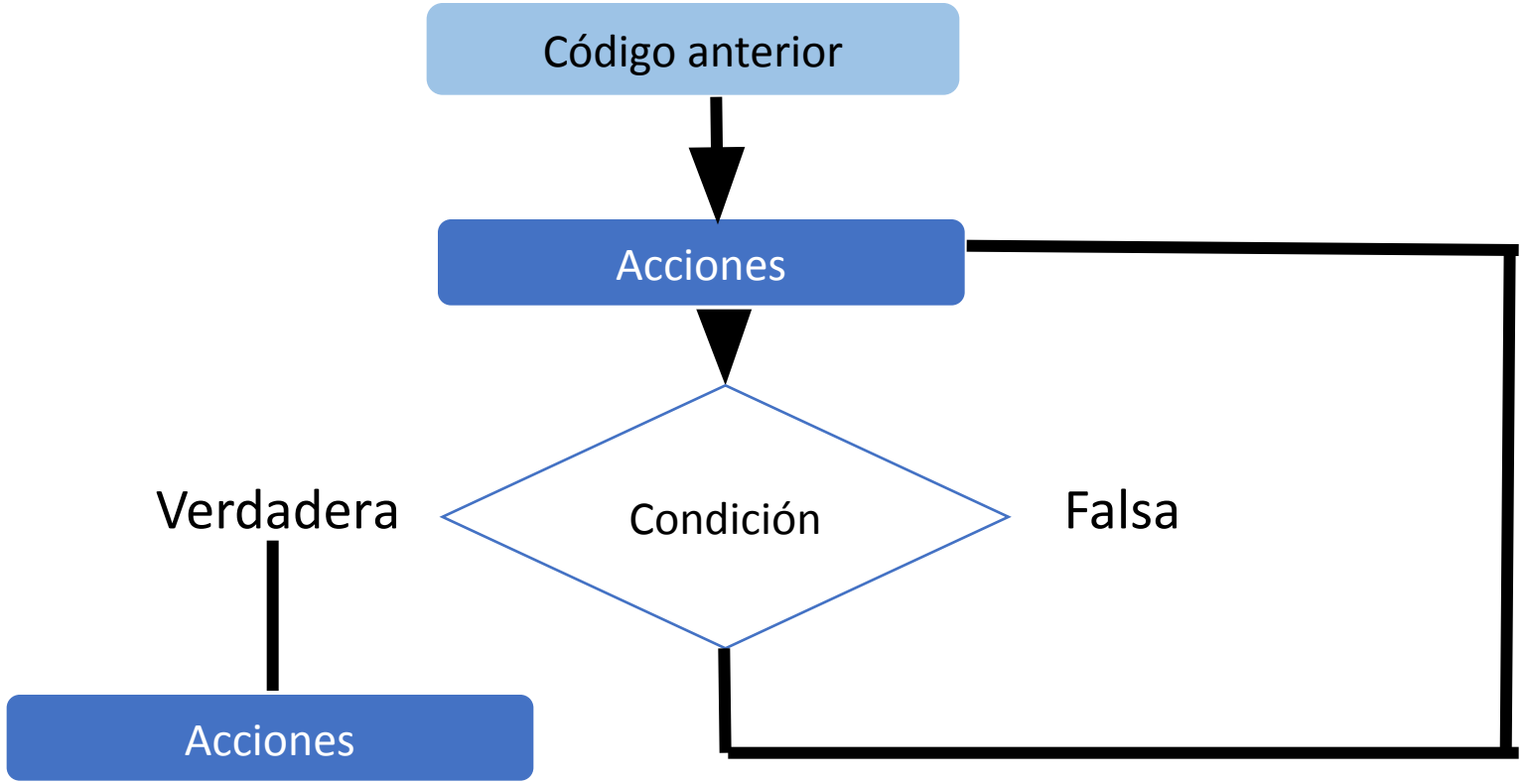
No se utiliza la variable resto

CADP – **ESTRUCTURAS DE CONTROL**



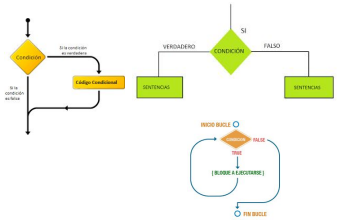
ITERACION - POSTCONDICIONAL

Ejecutan las acciones luego evalúan la condición y ejecutan las acciones mientras la condición es falsa. Dicho bloque se pueda ejecutar 1 ó más veces.



A qué estructura de control vista en el entorno del robot se parece?.

Cómo es la sintaxis?



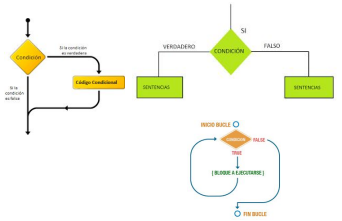
ITERACION - POSTCONDICIONAL

```
repeat
    accion;
until (condición);
```

más de una acción

```
repeat
   acción 1;
   acción 2;
until (condicion);
```

CADP – **ESTRUCTURAS DE CONTROL** ITERACION



Realizar un programa que lea códigos de productos hasta leer un código igual a 30. Al finalizar informe la cantidad de productos con código par. El último producto debe procesarse.

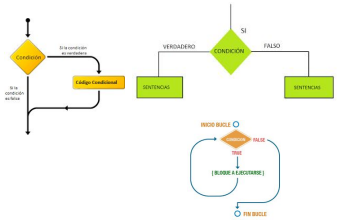
- Cómo leo un producto
- Cómo veo si es par
- Cuál es la condición de fin
- Cómo muestro el resultado

35
70
32
5
30

informa

3

CADP – **ESTRUCTURAS DE CONTROL** ITERACION



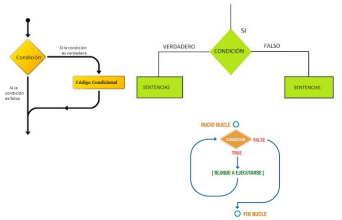
Sino existiera el repeat until

```
Program uno;
var
  prod:integer;
  total:integer;
 begin
   total:=0;
   read(prod);
   while (prod <> 30)do
    begin
      if (prod MOD 2 = 0)then
       total:= total + 1;
      read(prod);
    end;
   if (prod MOD 2 = 0)then
       total:= total + 1;
   write (total);
end.
```

Cuál es el problema de este tipo de soluciones?

Todo el procesamiento sobre la variable **prod** se debe repetir dentro y fuera del while

CADP – **ESTRUCTURAS DE CONTROL** ITERACION

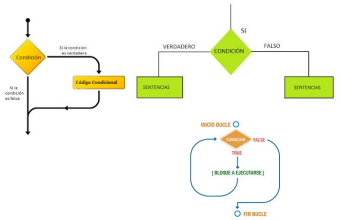


```
Program correcto;
var
   prod:integer;
   total:integer;

begin
    total:=0;
    repeat
      read (prod);
      if (prod MOD 2 = 0)then
        total:= total + 1;
    until (prod = 30)
    write (total);
end.
```

Se ejecuta cuando la condición es falsa

CADP – **Estructuras de control**

ITERACION



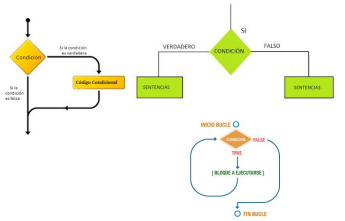
PRE CONDICIONALES

Evalúa la condición y en caso de ser verdadera, ejecuta las acciones.
Se repite mientras la condición es verdadera.
Puede ejecutarse 0, 1 o más veces.

POST CONDICIONALES

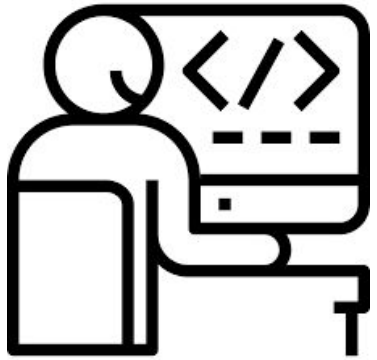
Ejecuta las acciones y luego evalúa la condición.
Se repite mientras la condición es falsa.
Puede ejecutarse 1 o más veces.

CADP – **ESTRUCTURAS DE CONTROL**



Mirando estos enunciados que estructuras de control usarías?

- Realizar un programa que lea un número e informe si el número es par o impar
- Realizar un programa que lea un letras hasta leer la letra “@” la cual debe procesarse e informe la cantidad de letras ‘á’ leídas.
- Realizar un programa que lea un letras hasta leer la letra “@” e informe la cantidad de letras ‘á’ leídas.

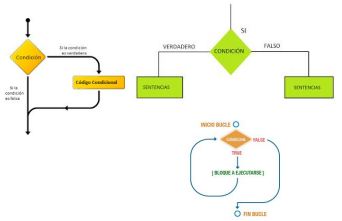


Conceptos de Algoritmos Datos y Programas

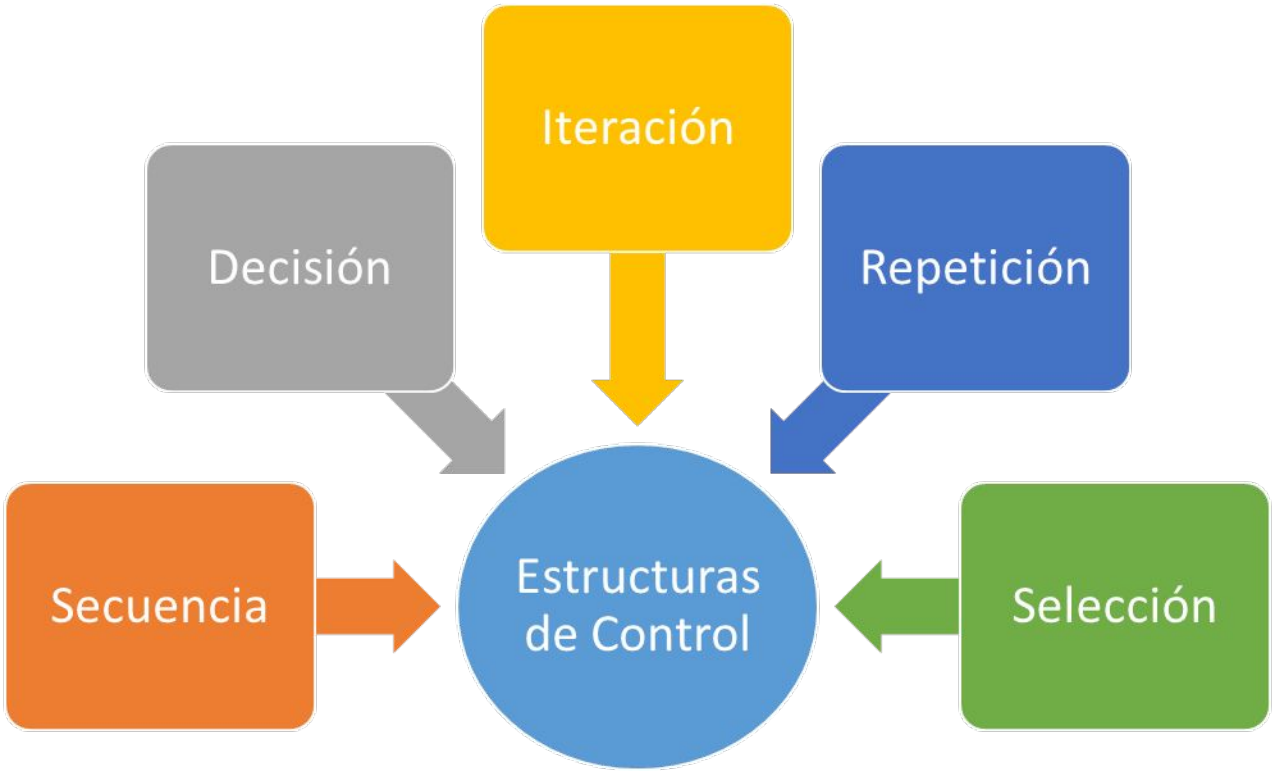
CADP – **TEMAS**

- Estructura de control
- Estructuras de control repetitivas
- Estructura de control FOR

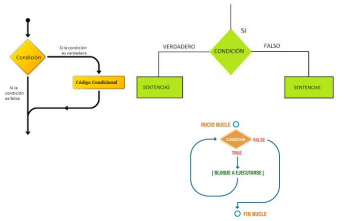
CADP – **ESTRUCTURAS DE CONTROL**



Todos los lenguajes de programación tienen un conjunto mínimo de instrucciones que permiten especificar el control del algoritmo que se quiere implementar. Como mínimo deben contener: secuencia, decisión e iteración.

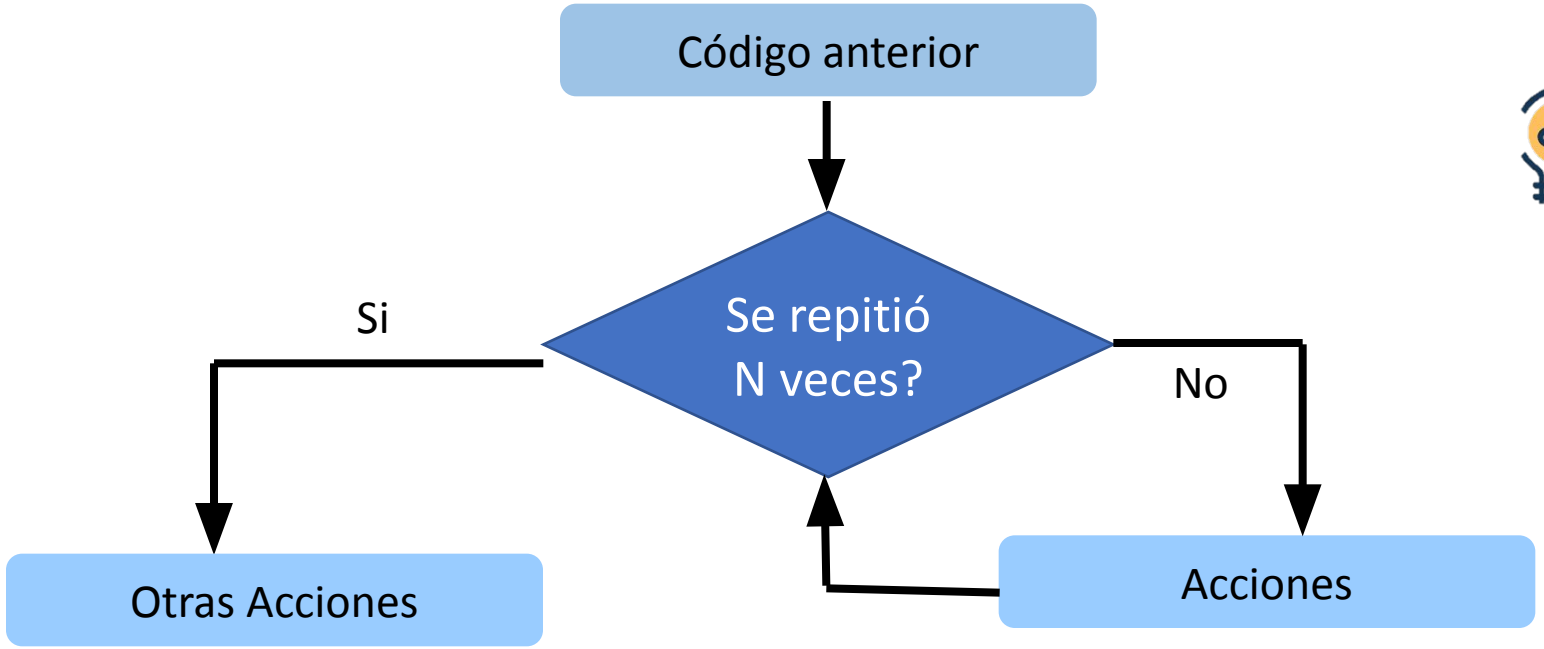


CADP – **ESTRUCTURAS DE CONTROL**



REPETICION

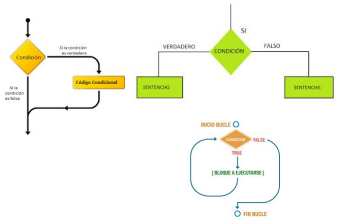
Es una extensión natural de la secuencia. Consiste en repetir N veces un bloque de acciones.
Este número de veces que se deben ejecutar las acciones es fijo y conocido de antemano



A qué estructura de control vista en el entorno del robot se parece?.

Cómo es la sintaxis?

CADP – **ESTRUCTURAS DE CONTROL** REPETICION



```
for indice :=  valor_inicial to valor_final do
     accion 1;
```

más de una acción

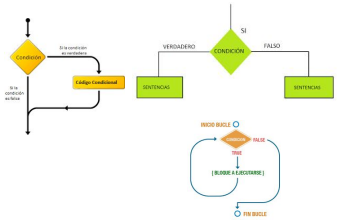
```
for indice :=  valor_inicial to valor_final do
  begin
     accion 1;
     accion 2;
  end;
```

Qué es el índice?

Dónde se declara?

Qué son valor final e inicial?

CADP – **ESTRUCTURAS DE CONTROL** REPETICION



Ejemplo 0:

```
For i := 1 to  10 do
      accion;
```

¿De qué tipo es el índice i?
¿qué valores toma i?

i:integer
i = 1,2,3,4,5,6,7,8,9,10

Ejemplo 1:

```
For i := 'A'  to  'H' do
      accion;
```

¿De qué tipo es el índice i?
¿qué valores toma i?

i:char
i = 'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H'

Ejemplo 2:

```
For i:= False  to  True do
      accion;
```

¿De qué tipo es el índice i?
¿qué valores toma i?

i:boolean
i = False True

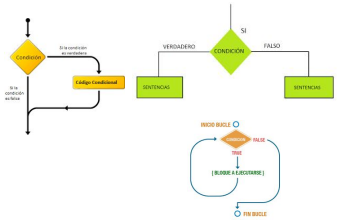
Ejemplo 3:

```
For i := 20 to 18 do
    accion;
```

```
For i := 20 downto 18 do
      begin
         accion;
         accion;
      end;
```

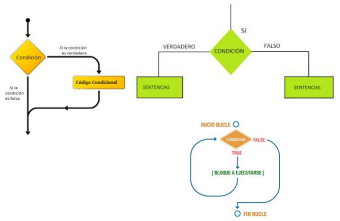
i:integer
i = 20 19 18

CADP – **ESTRUCTURAS DE CONTROL** REPETICION



- La variable índice debe ser de tipo ordinal
- La variable índice no puede modificarse dentro del lazo
- La variable índice se incrementa y decrementa automáticamente
- Cuando el for termina la variable índice no tiene valor definido.

CADP – **ESTRUCTURAS DE CONTROL** REPETICION



Realizar un programa que lea precios de 10 productos que vende un almacén. Al finalizar informe la suma de todos los precios leídos.

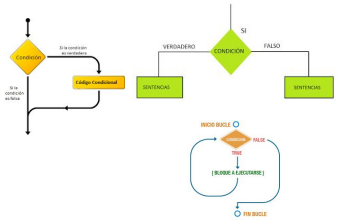
- Qué valor es el precio?
- Cuál es la condición de fin?
- Cómo calculo la suma

100,5
56,5
15
10
12,5
14
7,5
150,00
25,40
78,50

informa

474,4

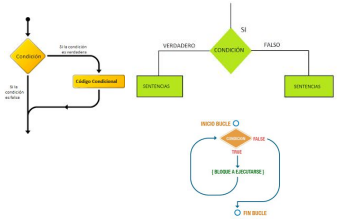
CADP – **ESTRUCTURAS DE CONTROL** REPETICION



```
Program uno;
var
   precio,total:real;
   i:integer;
begin
   total := 0;
   for i:= 1 to 10 do
     begin
         read (precio);
         total:= total + precio;
     end;
   write (“La suma de los precios de los
            productos del almacén son: ”,total);
end.
```

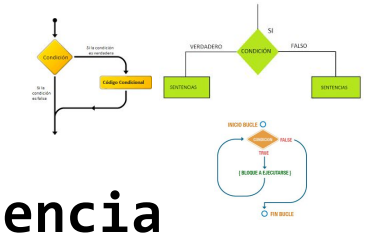
Qué modificaría si quiere informar al final, también el precio del 5to producto?

CADP – **ESTRUCTURAS DE CONTROL** REPETICION



```
Program uno;
var
   quinto,precio,total:real;
   i:integer;
begin
   total := 0;
   for i:= 1 to 10 do
     begin
         read (precio);
         if (i=5) then
             quinto:= precio;
         total:= total + precio;
     end;
   write (“La suma de los precios de los
            productos del almacén son: ”,total);
   write (“El precio del quinto producto es: ”,quinto);
end.
```

CADP – **ESTRUCTURAS DE CONTROL**



Qué crees que imprime el programa, si se leyera esta secuencia de números:

```
Program uno;
var
   i,num1,num2:integer;
Begin
  num2:= 0;
  for i:= 1 to 5 do
     begin
         read (num1);
         while (num1 mod 2 = 0) do
           begin
             num2:= num2 + 1;
             read (num1);
           end;
     end;
   write (num2);
end.
```

```
4
8
126
5
3
6
1
1568
6
10
7
19
22
24
3
```

CADP – **ESTRUCTURAS DE CONTROL**

Qué crees que imprime el programa, si se leyera esta secuencia de números:

```
Program uno;
var
   i,j,num1,num2:integer;
Begin
  num2:= 0;
  for i:= 1 to 3 do
     begin
         read (num1);
         for j:= 1 to 2 do
          begin
            if (num1 mod 2 = 1) then
              num2:= num2+1;
            read (num1);
         end;
         read (num1);
     end;
   write (num2);
end.
```

```
4
7
126
5
3
6
1
1568
6
10
7
19
22
24
3
```