

cÓNQUER



Pseudocódigo en *PseInt* (IV)

Módulo 2: IDE, Pseudocódigo y Python para principiantes



Contenidos

1. Actualización de nuestra “**caja de herramientas**” actual
 - a. Variables (entero, real, lógica, texto)
 - b. Instrucciones lectura/escritura
 - c. Estructura condicional SI-ENTONCES
 - d. Estructura selectiva SEGÚN
 - e. Funciones matemáticas
 - f. Estructuras iterativas MIENTRAS, REPETIR - HASTA QUE, PARA
2. Aprendiendo **nuevos conceptos**
 - a. Arrays: unidimensionales y bidimensionales
 - b. Cadenas de caracteres y funciones relacionadas
3. Algunos **ejemplos**
4. **Ejercicios** de la semana
5. Y en la **próxima clase...**

1. Nuestra “caja de herramientas” actual (i)

1. Variables

- Es un espacio en memoria que tiene asignado un identificador (*sumaTotal*) donde se guardan datos
- **Operaciones:** definición (identificador + tipo de dato), inicialización y modificación (+, -, *, /, MOD, ↑)

1. Nuestra “caja de herramientas” actual (i)

1. Variables

- Es un espacio en memoria que tiene asignado un identificador (*sumaTotal*) donde se guardan datos
- **Operaciones:** **definición (identificador + tipo de dato)**, inicialización y modificación (+, -, *, /, MOD, ↑)

Definir sumaTotal Como Entero

1. Nuestra “caja de herramientas” actual (i)

1. Variables

- Es un espacio en memoria que tiene asignado un identificador (*sumaTotal*) donde se guardan datos
- **Operaciones:** definición (identificador + tipo de dato), **inicialización** y modificación (+, -, *, /, MOD, ↑)

Definir sumaTotal Como Entero

sumaTotal = 12

1. Nuestra “caja de herramientas” actual (i)

1. Variables

- Es un espacio en memoria que tiene asignado un identificador (*sumaTotal*) donde se guardan datos
- **Operaciones:** definición (identificador + tipo de dato), inicialización y **modificación (+, -, *, /, MOD, ↑)**

```
Definir sumaTotal Como Entero
```

```
sumaTotal = 12
```

```
sumaTotal = sumaTotal + 4
```

1. Nuestra “caja de herramientas” actual (ii)

2. Instrucciones lectura/escritura

- **Lectura:** lee un dato introducido por consola y lo almacena en una variable
- **Escritura:** escribe en consola el valor de una variable o varias variables

Leer precio

1. Nuestra “caja de herramientas” actual (ii)

2. Instrucciones lectura/escritura

- Lectura: lee un dato introducido por consola y lo almacena en una variable
- Escritura: escribe en consola el valor de una variable o varias variables

Leer precio

Escribir "El precio es: ", precio

Escribir "El precio es: ", precio Sin Saltar

1. Nuestra “caja de herramientas” actual (iii)

3. Estructura condicional SI-ENTONCES

- Permite ejecutar un código diferente en función de una o varias expresiones lógicas

1. Nuestra “caja de herramientas” actual (iii)

3. Estructura condicional SI-ENTONCES

- Permite ejecutar un código diferente en función de una o varias expresiones lógicas

Sintaxis básica

```
Si expresion_logica Entonces
    ejecucion_codigo_1
    SiNo
        ejecucion_codigo_2
    FinSi
```

1. Nuestra “caja de herramientas” actual (iii)

3. Estructura condicional SI-ENTONCES

- Permite ejecutar un código diferente en función de una o varias expresiones lógicas

Sintaxis básica

```
Si expresion_logica Entonces  
| ejecucion_codigo_1  
SiNo  
| ejecucion_codigo_2  
FinSi
```

Sintaxis anidada

```
Si expresion_logica_1 Entonces  
| ejecucion_codigo_1  
SiNo  
| ejecucion_codigo_2  
FinSi  
SiNo  
| Si expresion_logica_3 Entonces  
| | ejecucion_codigo_3  
| SiNo  
| | ejecucion_codigo_4  
| FinSi  
FinSi
```

1. Nuestra “caja de herramientas” actual (iv)

4. Estructura selectiva SEGÚN

- Permite ejecutar un código diferente en función del valor de una variable

1. Nuestra “caja de herramientas” actual (iv)

4. Estructura selectiva SEGÚN

- Permite ejecutar un código diferente en función del valor de una variable

Sintaxis

```
Segun variable Hacer
  opcion_1:
    ejecucion_codigo_1
  opcion_2:
    ejecucion_codigo_2

  De otro Modo:
    ejecucion_codigo_otro_modo
FinSegun
```

1. Nuestra “caja de herramientas” actual (v)

5. Funciones matemáticas

- Permiten calcular el valor de una magnitud a partir de valor de otra
 - Valor absoluto
 - Valor truncado
 - Valor redondeado
 - Valor al azar

1. Nuestra “caja de herramientas” actual (v)

5. Funciones matemáticas

- Permiten calcular el valor de una magnitud a partir de valor de otra

- **Valor absoluto:** devuelve el número “sin signo”

```
resultado = abs(-3.6) // resultado = 3.6
```

- Valor truncado
 - Valor redondeado
 - Valor al azar

1. Nuestra “caja de herramientas” actual (v)

5. Funciones matemáticas

- Permiten calcular el valor de una magnitud a partir de valor de otra
 - Valor absoluto: devuelve el número “sin signo”

```
resultado = abs(-3.6) //resultado = 3.6
```

- **Valor truncado:** devuelve la “parte entera” de un número real

```
resultado = trunc(-3.6) //resultado = -3
```

- Valor redondeado
- Valor al azar

1. Nuestra “caja de herramientas” actual (v)

5. Funciones matemáticas

- Permiten calcular el valor de una magnitud a partir de valor de otra
 - Valor absoluto: devuelve el número “sin signo”

```
resultado = abs(-3.6) //resultado = 3.6
```

- Valor truncado: devuelve la “parte entera” de un número real

```
resultado = trunc(-3.6) //resultado = -3
```

- **Valor redondeado:** “aproxima” la parte entera del número

```
resultado = redon(-3.6) //resultado = -4
```

- Valor al azar

1. Nuestra “caja de herramientas” actual (v)

5. Funciones matemáticas

- Permiten calcular el valor de una magnitud a partir de valor de otra

- Valor absoluto: devuelve el número “sin signo”

```
resultado = abs(-3.6) //resultado = 3.6
```

- Valor truncado: devuelve la “parte entera” de un número real

```
resultado = trunc(-3.6) //resultado = -3
```

- Valor redondeado: “aproxima” la parte entera del número

```
resultado = redon(-3.6) //resultado = -4
```

- **Valor al azar:** devuelve un número aleatorio entre “0” y “numero-1”

```
resultado = azar(21) //resultado = [0 - 20]
```

1. Nuestra “caja de herramientas” actual (vi)

6. Estructuras iterativas (bucles)

- Es una secuencia de instrucciones que se ejecuta varias veces, mientras se cumpla cierta condición

MIENTRAS

REPETIR - HASTA QUE

PARA

1. Nuestra “caja de herramientas” actual (vi)

6. Estructuras iterativas (bucles)

- Es una secuencia de instrucciones que se ejecuta varias veces, mientras se cumpla cierta condición

MIENTRAS

- Se ejecuta “mientras” se cumpla cierta condición (exp/op lógica)

```
Mientras expr_logica Hacer
    ejecucion_codigo
FinMientras

Mientras (expr_logica_1) OP_LOG (expr_logica_2) Hacer
    ejecucion_codigo
FinMientras
```

NOTA1!!

Si la primera vez, la op/expr lógica es **FALSA**, el bucle **NO SE EJECUTARÁ NINGUNA VEZ**

NOTA2!!

Si la op/expr lógica es siempre **VERDADERA**, será un bucle **INFINITO**

1. Nuestra “caja de herramientas” actual (vi)

6. Estructuras iterativas (bucles)

- Es una secuencia de instrucciones que se ejecuta varias veces, mientras se cumpla cierta condición

MIENTRAS

```
Definir temporizador Como Entero
temporizador = 0

Escribir "Introduce el número de segundos del temporizador"
Leer temporizador

Escribir "Comienza el temporizador..."
Mientras (temporizador > 0) Hacer
    Escribir "Quedan ", temporizador, " segundos"
    Esperar 1 Segundos
    temporizador = temporizador - 1
FinMientras
Escribir "El temporizador ha finalizado!!"
```

1. Nuestra “caja de herramientas” actual (vi)

6. Estructuras iterativas (bucles)

- Es una secuencia de instrucciones que se ejecuta varias veces, mientras se cumpla cierta condición

REPETIR - HASTA QUE

- Similar al bucle “mientras”, solo que la exp/op lógica se evalúa al final (ejec. 1^a vez)

```
Repetir
    ejecucion_codigo
Hasta Que expr_logica

Repetir
    ejecucion_codigo
Hasta Que (expr_logica_1) OP_LOG (expr_logica_2)
```

NOTA!!

Si la op/expr lógica es siempre **VERDADERA**, será un bucle **INFINITO**

1. Nuestra “caja de herramientas” actual (vi)

6. Estructuras iterativas (bucles)

- Es una secuencia de instrucciones que se ejecuta varias veces, mientras se cumpla cierta condición

REPETIR - HASTA QUE

```
Definir temporizador Como Entero
temporizador = 0

Escribir "Introduce el número de segundos del temporizador"
Leer temporizador

Escribir "Comienza el temporizador..."
Repetir
    Escribir "Quedan ", temporizador, " segundos"
    Esperar 1 Segundos
    temporizador = temporizador - 1
Hasta que (temporizador = 0)
Escribir "El temporizador ha finalizado!!"
```

1. Nuestra “caja de herramientas” actual (vi)

6. Estructuras iterativas (bucles)

- Es una secuencia de instrucciones que se ejecuta varias veces, mientras se cumpla cierta condición

PARA

- El nº de repeticiones depende del valor de una variable, que toma un valor inicial y se va incrementando/decrementando en cada repetición

```
Para i = valor_inicial Hasta valor_final Con Paso incr_decr Hacer  
    ejecucion_codigo  
FinPara
```

NOTA!!

La variable “i” del bucle **PARA** tiene que **DEFINIRSE**, como cualquier otra variable

1. Nuestra “caja de herramientas” actual (vi)

6. Estructuras iterativas (bucles)

- Es una secuencia de instrucciones que se ejecuta varias veces, mientras se cumpla cierta condición

PARA

```
Definir i, temporizador Como Entero
temporizador = 0
i = 0

Escribir "Introduce el número de segundos del temporizador"
Leer temporizador

Escribir "Comienza el temporizador..."
Para i = temporizador Hasta 1 Con paso -1 Hacer
    Escribir "Quedan ", i, " segundos"
    Esperar 1 Segundos
FinPara
Escribir "El temporizador ha finalizado!!"
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

¿Qué es un **array**?

- Es una estructura de datos, de **tamaño determinado**, en la que se almacenan valores de un mismo tipo de dato
- Además, un array tiene asignado un **identificador** (`edadesAlumnos`)

2. Aprendiendo nuevos conceptos (i)

ARRAYS

¿Qué es un **array**?

- Es una estructura de datos, de **tamaño determinado**, en la que se almacenan valores de un mismo tipo de dato
- Además, un array tiene asignado un **identificador** (`edadesAlumnos`)

Variable

Definir `edadAlumno` Como Entero



2. Aprendiendo nuevos conceptos (i)

ARRAYS

¿Qué es un **array**?

- Es una estructura de datos, de **tamaño determinado**, en la que se almacenan valores de un mismo tipo de dato
- Además, un array tiene asignado un **identificador** (`edadesAlumnos`)

Variable

Definir `edadAlumno` Como Entero

```
edadAlumno = 14
```



2. Aprendiendo nuevos conceptos (i)

ARRAYS

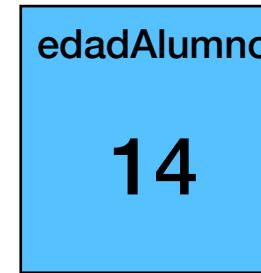
¿Qué es un **array**?

- Es una estructura de datos, de **tamaño determinado**, en la que se almacenan valores de un mismo tipo de dato
- Además, un array tiene asignado un **identificador** (`edadesAlumnos`)

Variable

Definir `edadAlumno` Como Entero

```
edadAlumno = 14
```



Pregunta

- ¿Qué pasa si queremos guardar las edades de 20 alumnos?
- ¿Definimos 20 variables?

CONQUERBLOCKS

2. Aprendiendo nuevos conceptos (i)

ARRAYS

¿Qué es un **array**?

- Es una estructura de datos, de **tamaño determinado**, en la que se almacenan valores de un mismo tipo de dato
- Además, un array tiene asignado un **identificador** (`edadesAlumnos`)

Variable

Definir `edadAlumno` Como Entero

```
edadAlumno = 14
```



Pregunta

¿Qué pasa si queremos guardar las edades de 20 alumnos?

¿Definimos 20 variables?

Array

Solución

Uso de un array de longitud “20” donde almacenar 20 datos de tipo entero



2. Aprendiendo nuevos conceptos (i)

ARRAYS

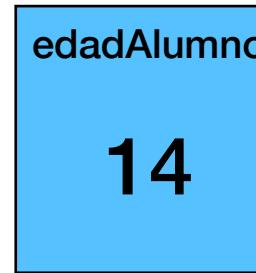
¿Qué es un **array**?

- Es una estructura de datos, de **tamaño determinado**, en la que se almacenan valores de un mismo tipo de dato
- Además, un array tiene asignado un **identificador** (`edadesAlumnos`)

Variable

Definir `edadAlumno` Como Entero

```
edadAlumno = 14
```



Pregunta

¿Qué pasa si queremos guardar las edades de 20 alumnos?

¿Definimos 20 variables?

Array

`edadesAlumnos`



Solución

Uso de un array de longitud “20” donde almacenar 20 datos de tipo entero

2. Aprendiendo nuevos conceptos (i)

ARRAYS

¿Qué es un **array**?

- Es una estructura de datos, de **tamaño determinado**, en la que se almacenan valores de un mismo tipo de dato
- Además, un array tiene asignado un **identificador** (`edadesAlumnos`)

Variable

Definir `edadAlumno` Como Entero

```
edadAlumno = 14
```



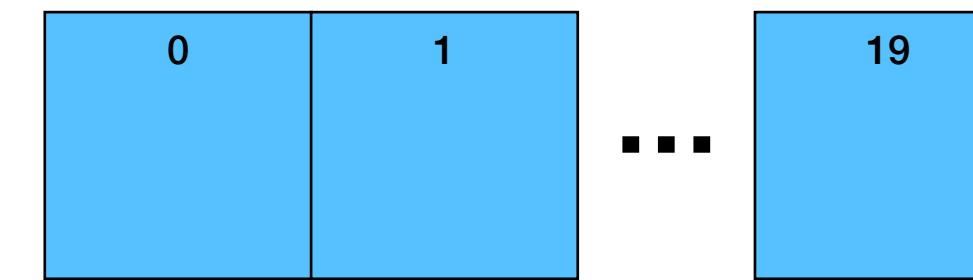
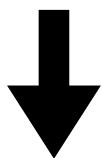
Pregunta

¿Qué pasa si queremos guardar las edades de 20 alumnos?

¿Definimos 20 variables?

Array

`edadesAlumnos`



Solución

Uso de un array de longitud “20” donde almacenar 20 datos de tipo entero



2. Aprendiendo nuevos conceptos (i)

ARRAYS

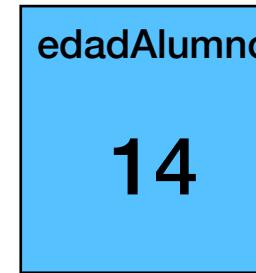
¿Qué es un **array**?

- Es una estructura de datos, de **tamaño determinado**, en la que se almacenan valores de un mismo tipo de dato
- Además, un array tiene asignado un **identificador** (`edadesAlumnos`)

Variable

Definir `edadAlumno` Como Entero

```
edadAlumno = 14
```



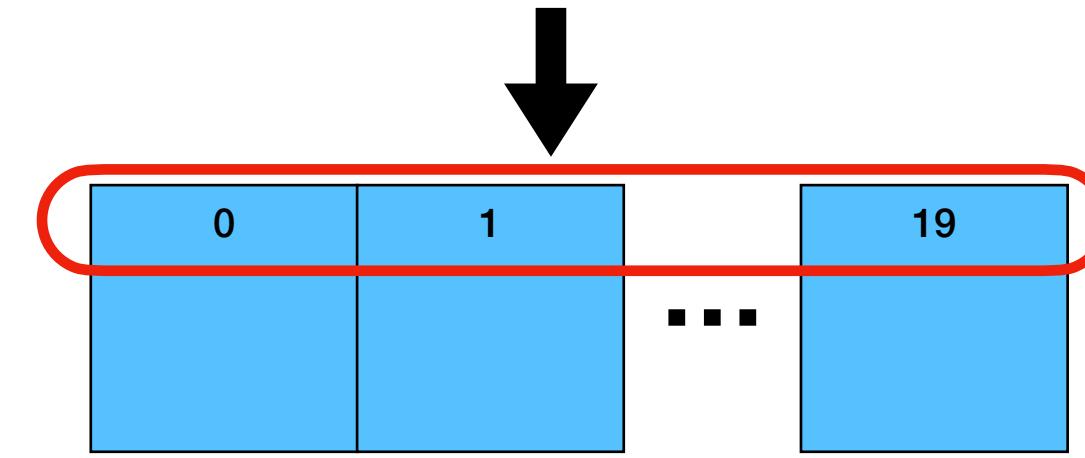
Pregunta

¿Qué pasa si queremos guardar las edades de 20 alumnos?

¿Definimos 20 variables?

Array

`edadesAlumnos`



Solución

Uso de un array de longitud “20” donde almacenar 20 datos de tipo entero



2. Aprendiendo nuevos conceptos (i)

ARRAYS

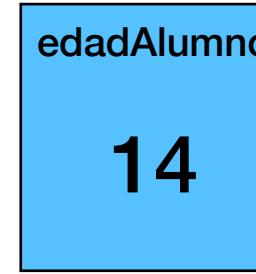
¿Qué es un **array**?

- Es una estructura de datos, de **tamaño determinado**, en la que se almacenan valores de un mismo tipo de dato
- Además, un array tiene asignado un **identificador** (`edadesAlumnos`)

Variable

Definir `edadAlumno` Como Entero

```
edadAlumno = 14
```



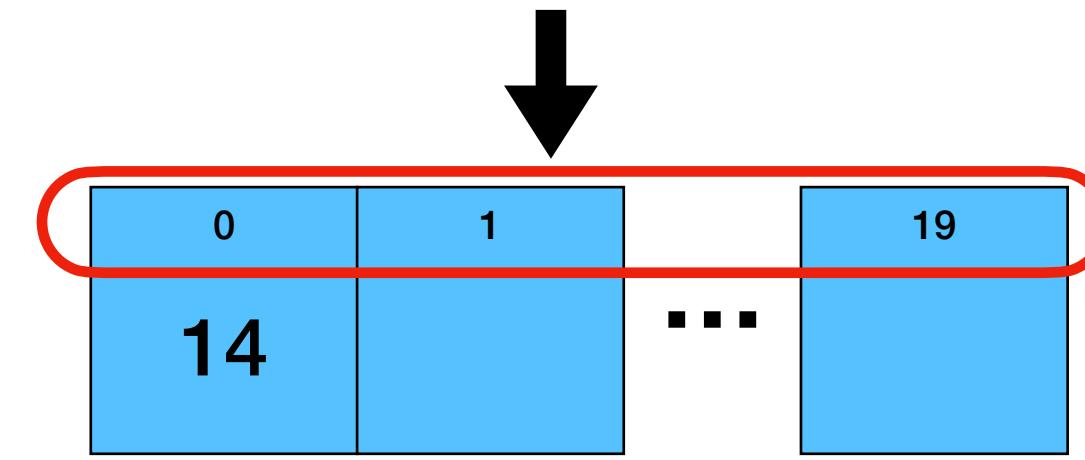
Pregunta

¿Qué pasa si queremos guardar las edades de 20 alumnos?

¿Definimos 20 variables?

Array

`edadesAlumnos`



Solución

Uso de un array de longitud “20” donde almacenar 20 datos de tipo entero



2. Aprendiendo nuevos conceptos (i)

ARRAYS

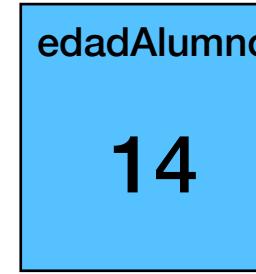
¿Qué es un **array**?

- Es una estructura de datos, de **tamaño determinado**, en la que se almacenan valores de un mismo tipo de dato
- Además, un array tiene asignado un **identificador** (`edadesAlumnos`)

Variable

Definir `edadAlumno` Como Entero

```
edadAlumno = 14
```



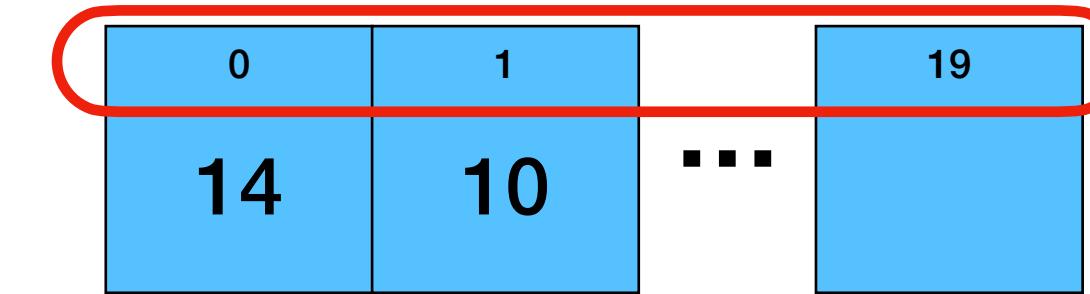
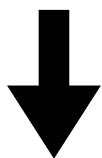
Pregunta

¿Qué pasa si queremos guardar las edades de 20 alumnos?

¿Definimos 20 variables?

Array

`edadesAlumnos`



Solución

Uso de un array de longitud “20” donde almacenar 20 datos de tipo entero



2. Aprendiendo nuevos conceptos (i)

ARRAYS

¿Qué es un **array**?

- Es una estructura de datos, de **tamaño determinado**, en la que se almacenan valores de un mismo tipo de dato
- Además, un array tiene asignado un **identificador** (`edadesAlumnos`)

Variable

Definir `edadAlumno` Como Entero

```
edadAlumno = 14
```



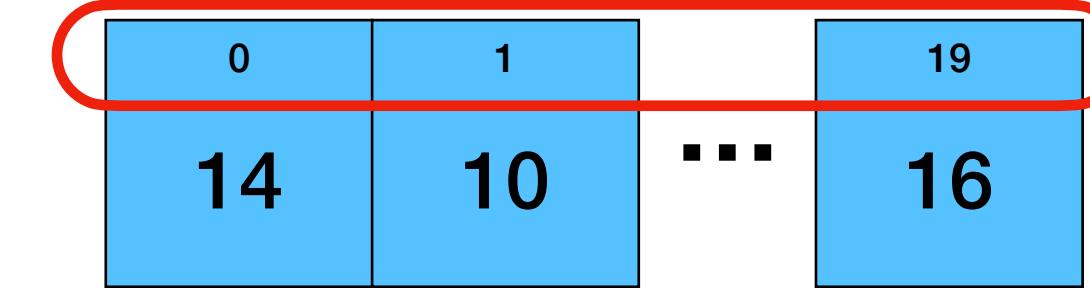
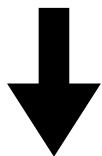
Pregunta

¿Qué pasa si queremos guardar las edades de 20 alumnos?

¿Definimos 20 variables?

Array

`edadesAlumnos`



Solución

Uso de un array de longitud “20” donde almacenar 20 datos de tipo entero



2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

1. Definición
2. Inicialización
3. Modificación
4. Lectura/escritura

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

1. Definición

```
Definir edadesAlumnos Como Entero  
Dimension edadesAlumnos[20]
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

1. Definición

```
Definir edadesAlumnos Como Entero  
Dimension edadesAlumnos[20]
```

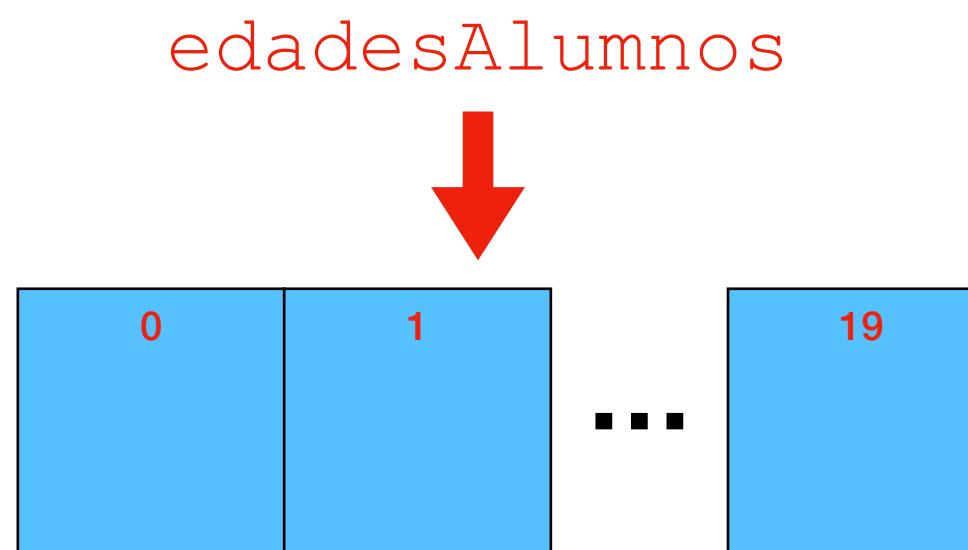
2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

1. Definición

```
Definir edadesAlumnos Como Entero  
Dimension edadesAlumnos[20]
```



2. Aprendiendo nuevos conceptos (i)

ARRAYS

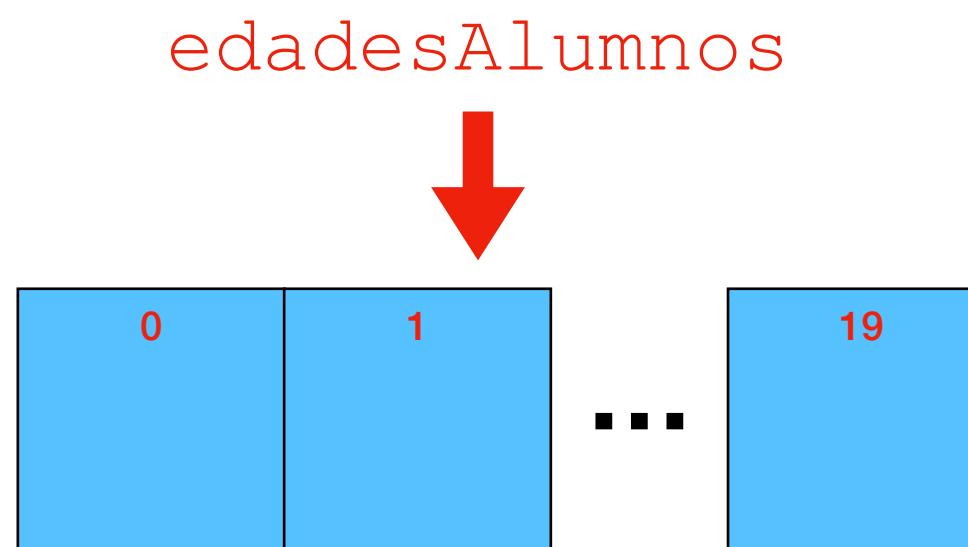
Operaciones

1. Definición

```
Definir edadesAlumnos Como Entero  
Dimension edadesAlumnos[20]
```

NOTA!!

Este tipo de **arrays** son
unidimensionales



2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

2. Inicialización

```
Para i = 0 Hasta 19 Con Paso 1 Hacer  
    edadesAlumnos[i] = 0 //Cada posición, se inicializa a 0  
FinPara
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

2. Inicialización

```
Para i = 0 Hasta 19 Con Paso 1 Hacer  
    edadesAlumnos[i] = 0 //Cada posición, se inicializa a 0  
FinPara
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

2. Inicialización

```
Para i = 0 Hasta 19 Con Paso 1 Hacer  
    edadesAlumnos[i] = 0 //Cada posición, se inicializa a 0  
FinPara
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

2. Inicialización

```
Para i = 0 Hasta 19 Con Paso 1 Hacer  
    edadesAlumnos[i] = 0 //Cada posición, se inicializa a 0  
FinPara
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

2. Inicialización

```
Para i = 0 Hasta 19 Con Paso 1 Hacer
    edadesAlumnos[i] = 0 //Cada posición, se inicializa a 0
FinPara
```

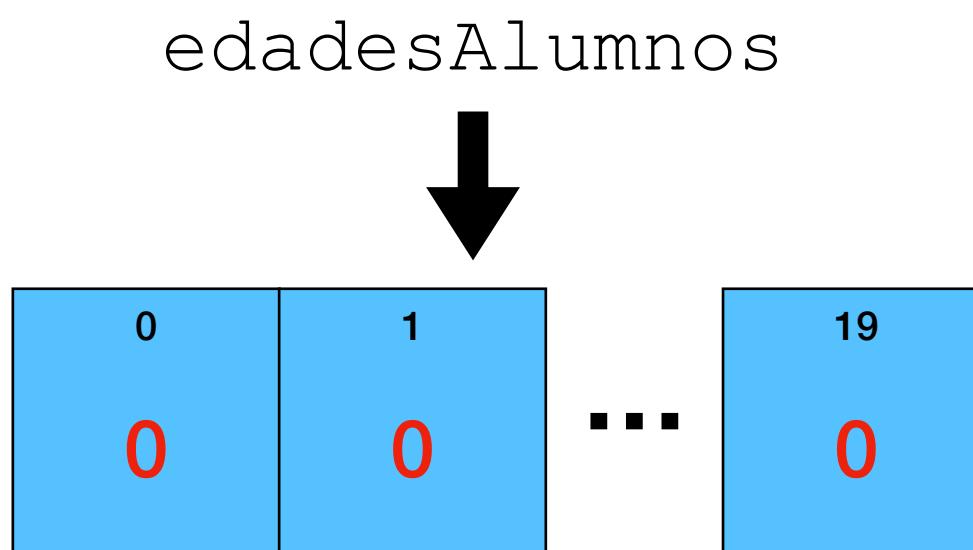
2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

2. Inicialización

```
Para i = 0 Hasta 19 Con Paso 1 Hacer
    edadesAlumnos[i] = 0 //Cada posición, se inicializa a 0
FinPara
```



2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

3. Modificación

```
//Solo se pueden guardar números enteros  
edadesAlumnos[1] = 16  
edadesAlumnos[9] = edadesAlumnos[1]
```

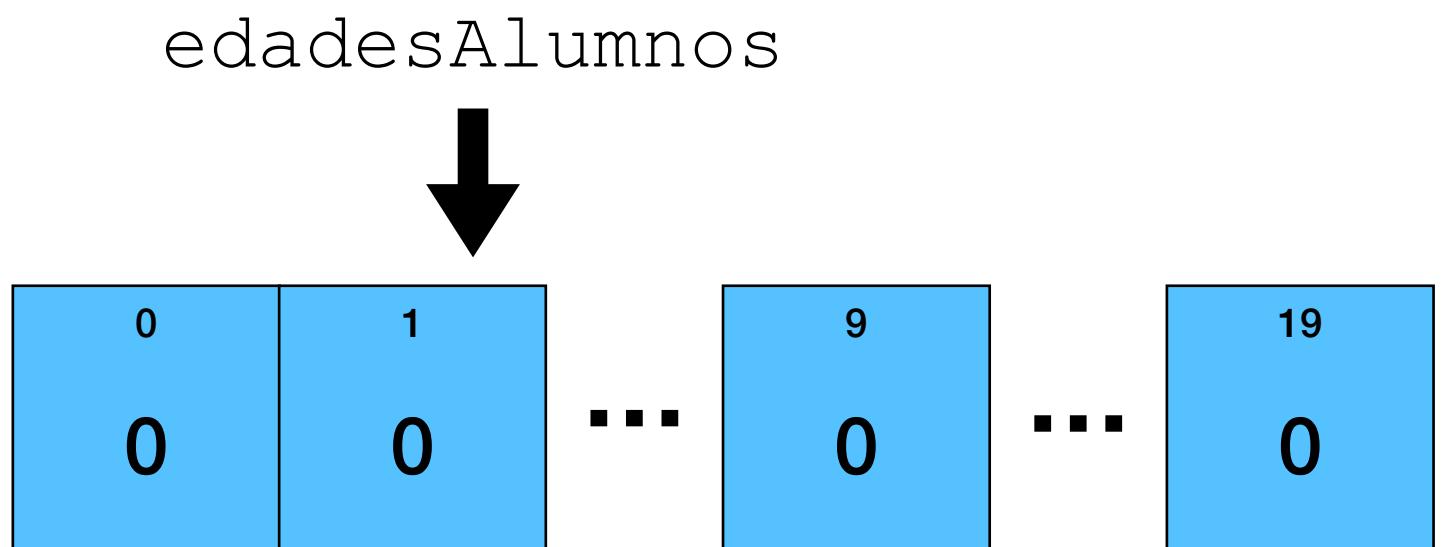
2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

3. Modificación

```
//Solo se pueden guardar números enteros  
edadesAlumnos[1] = 16  
edadesAlumnos[9] = edadesAlumnos[1]
```



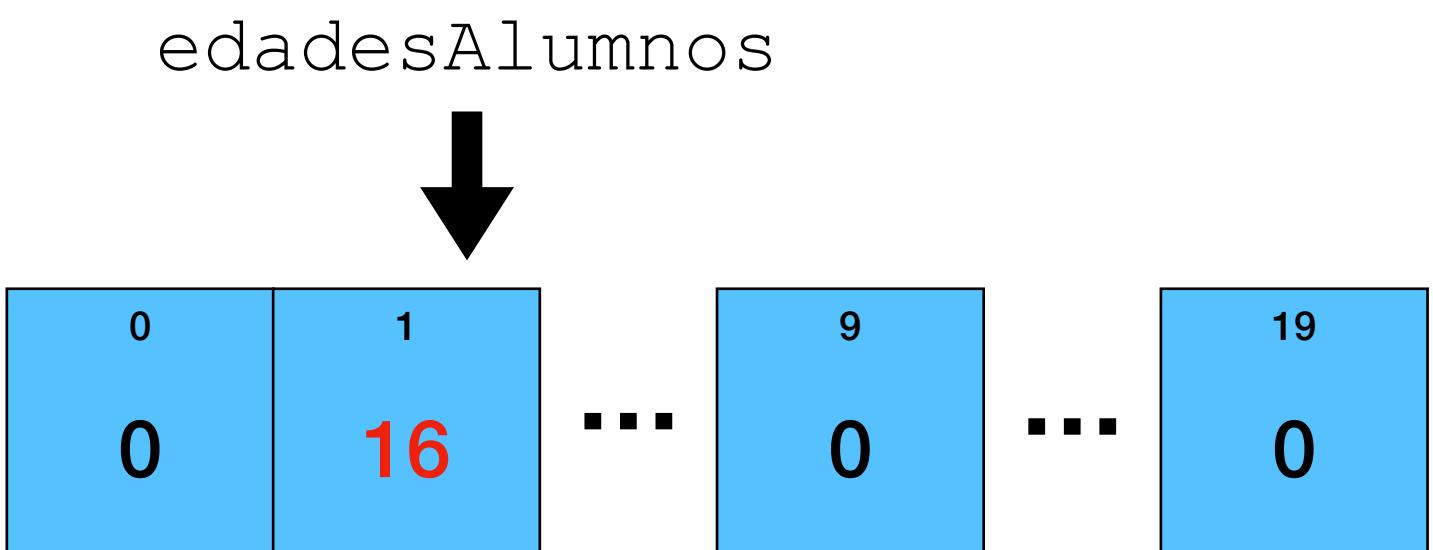
2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

3. Modificación

```
//Solo se pueden guardar números enteros  
edadesAlumnos[1] = 16  
edadesAlumnos[9] = edadesAlumnos[1]
```



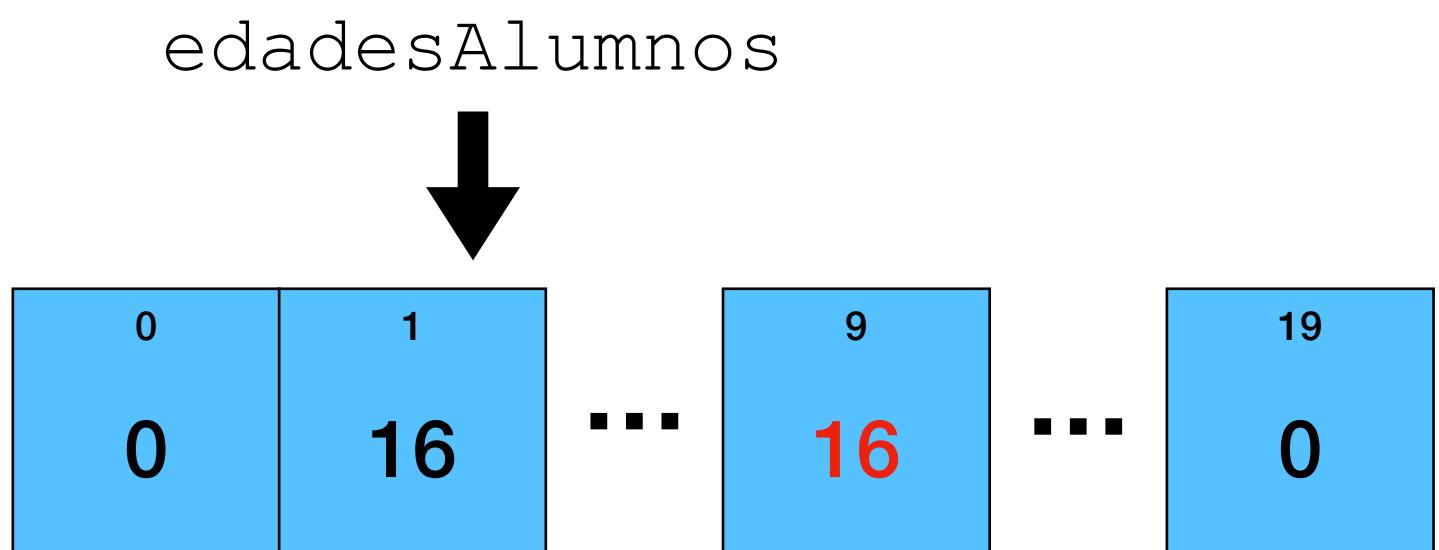
2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

3. Modificación

```
//Solo se pueden guardar números enteros  
edadesAlumnos[1] = 16  
edadesAlumnos[9] = edadesAlumnos[1]
```



2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

4. Lectura/Escritura

```
Leer edadesAlumnos[2]  
Escribir edadesAlumnos[2]
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

4. Lectura/Escritura

```
Leer edadesAlumnos[2]
Escribir edadesAlumnos[2]
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

4. Lectura/Escritura

```
Leer edadesAlumnos[2]
Escribir edadesAlumnos[2]
```

2. Aprendiendo nuevos conceptos (i)

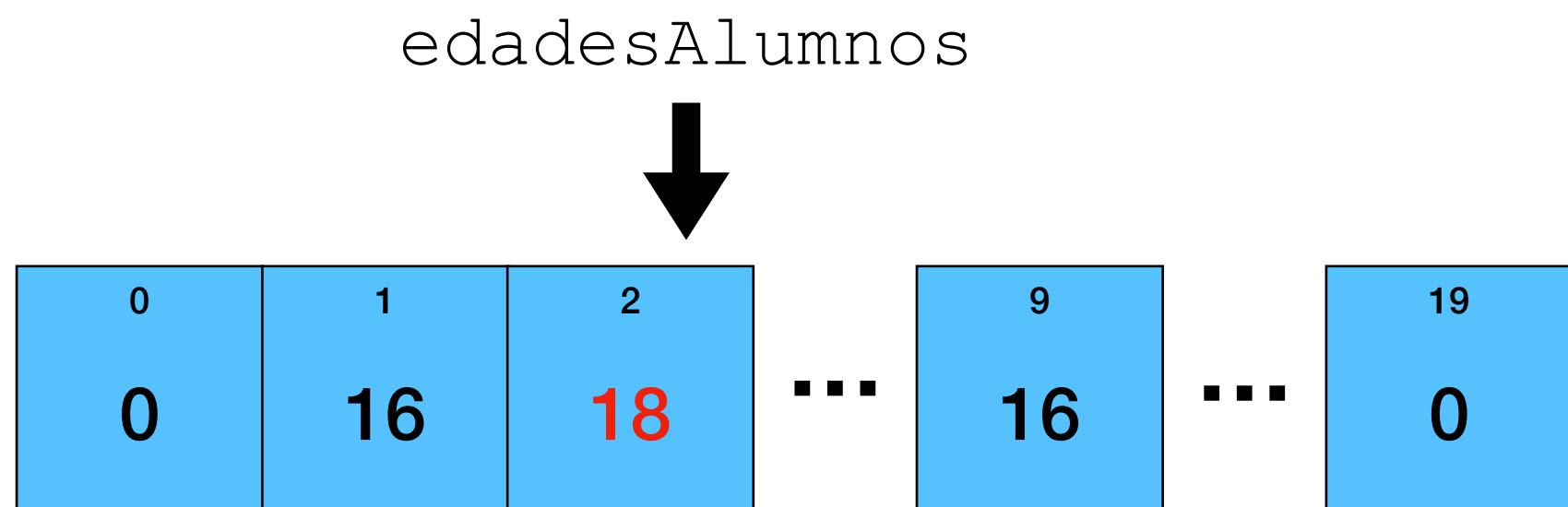
ARRAYS

Operaciones

4. Lectura/Escritura

```
Leer edadesAlumnos[2]  
Escribir edadesAlumnos[2]
```

```
*** Ejecución Iniciada. ***  
> 18
```



2. Aprendiendo nuevos conceptos (i)

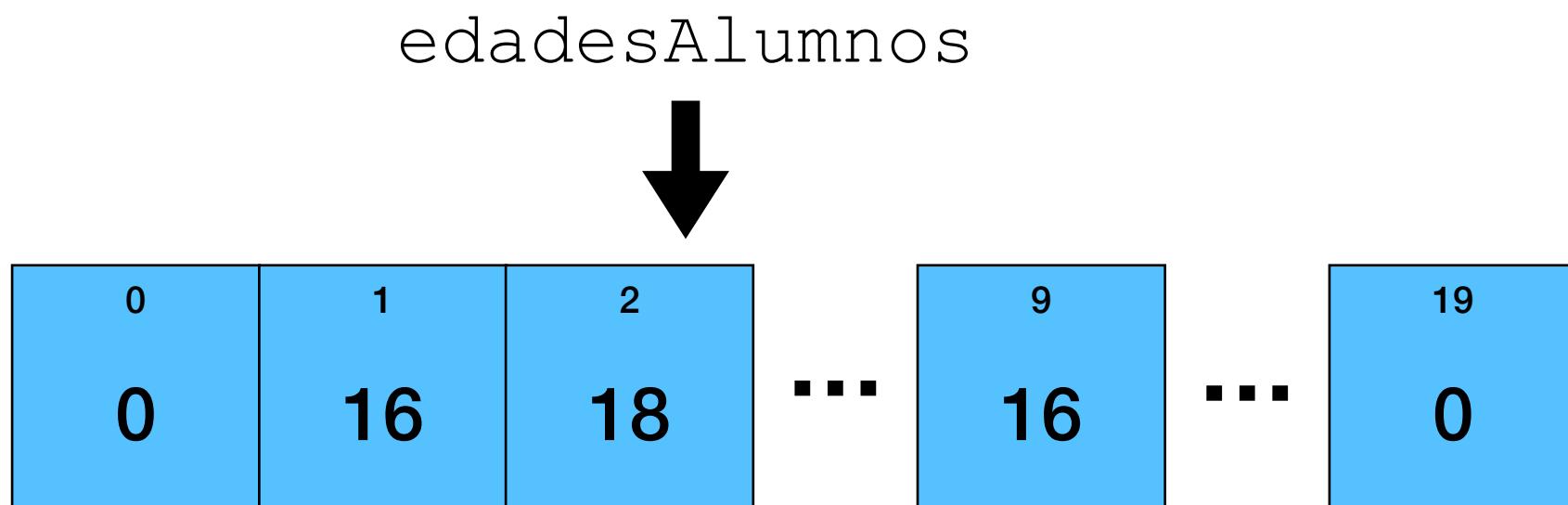
ARRAYS

Operaciones

4. Lectura/Escritura

```
Leer edadesAlumnos[2]  
Escribir edadesAlumnos[2]
```

```
*** Ejecución Iniciada. ***  
> 18  
18  
*** Ejecución Finalizada. ***
```



2. Aprendiendo nuevos conceptos (i)

ARRAYS

Ejemplo

```
//Definición
Definir numerosAzar, i, num Como Entero
Dimension numerosAzar[10]

//Inicialización
Para i = 0 Hasta 9 Con Paso 1 Hacer
    num = azar(20) //Números aleatorios del 0 al 19
    numerosAzar[i] = num //Cada posición, se inicializa con un número aleatorio
FinPara

//Ver todos los números del array
Para i = 0 Hasta 9 Con Paso 1 Hacer
    Escribir "Número [", i, "]: " numerosAzar[i]
FinPara
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Ejemplo

```
//Definición
Definir numerosAzar, i, num Como Entero
Dimension numerosAzar[10]

//Inicialización
Para i = 0 Hasta 9 Con Paso 1 Hacer
    num = azar(20) //Números aleatorios del 0 al 19
    numerosAzar[i] = num //Cada posición, se inicializa con un número aleatorio
FinPara

//Ver todos los números del array
Para i = 0 Hasta 9 Con Paso 1 Hacer
    Escribir "Número [", i, "]: " numerosAzar[i]
FinPara
```

```
*** Ejecución Iniciada. ***
Número [0]: 8
Número [1]: 19
Número [2]: 14
Número [3]: 14
Número [4]: 4
Número [5]: 12
Número [6]: 12
Número [7]: 10
Número [8]: 2
Número [9]: 9
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Arrays bidimensionales

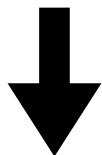
2. Aprendiendo nuevos conceptos (i)

ARRAYS

Arrays bidimensionales

Array unidimensional

arrayEnterosUnid



0	1	2	3
14	10	16	16

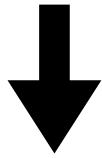
2. Aprendiendo nuevos conceptos (i)

ARRAYS

Arrays bidimensionales

Array unidimensional

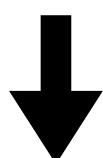
arrayEnterosUnid



0	1	2	3
14	10	16	16

Array bidimensional

arrayEnterosBid



0,0	0,1	0,2	0,3
14	10	16	16
1,0	1,1	1,2	1,3
11	76	34	0
2,0	2,1	2,2	2,3
1	4	67	23

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

1. Definición

Array unidimensional

```
Definir edadesAlumnos Como Entero  
Dimension edadesAlumnos[20]
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

1. Definición

Array unidimensional

```
Definir edadesAlumnos Como Entero  
Dimension edadesAlumnos[20]
```

Array bidimensional

```
Definir edadesAlumnosClase Como Entero  
Dimension edadesAlumnosClase[3,20]
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

1. Definición

Array unidimensional

```
Definir edadesAlumnos Como Entero  
Dimension edadesAlumnos[20]
```

Array bidimensional

```
Definir edadesAlumnosClase Como Entero  
Dimension edadesAlumnosClase[3, 20]
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

1. Definición

Array unidimensional

```
Definir edadesAlumnos Como Entero  
Dimension edadesAlumnos[20]
```

Array bidimensional

```
Definir edadesAlumnosClase Como Entero  
Dimension edadesAlumnosClase[3, 20]
```

0,0	0,1	0,2	...	0,19
1,0	1,1	1,2	...	1,19
2,0	2,1	2,2	...	2,19
...

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

2. Inicialización

Array unidimensional

```
Para i = 0 Hasta 19 Con Paso 1 Hacer  
    edadesAlumnos[i] = 0 //Cada posición, se inicializa a 0  
FinPara
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

2. Inicialización

Array unidimensional

```
Para i = 0 Hasta 19 Con Paso 1 Hacer
    edadesAlumnos[i] = 0 //Cada posición, se inicializa a 0
FinPara
```

Array bidimensional

```
Para filas = 0 Hasta 2 Con Paso 1 Hacer
    Para columnas = 0 Hasta 19 Con Paso 1 Hacer
        edadesAlumnosClase[filas,columnas] = 0
    FinPara
FinPara
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

2. Inicialización

Array unidimensional

```
Para i = 0 Hasta 19 Con Paso 1 Hacer  
    edadesAlumnos[i] = 0 //Cada posición, se inicializa a 0  
FinPara
```

Array bidimensional

```
Para filas = 0 Hasta 2 Con Paso 1 Hacer  
    Para columnas = 0 Hasta 19 Con Paso 1 Hacer  
        edadesAlumnosClase[filas, columnas] = 0  
    FinPara  
FinPara
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

2. Inicialización

Array unidimensional

```
Para i = 0 Hasta 19 Con Paso 1 Hacer  
    edadesAlumnos[i] = 0 //Cada posición, se inicializa a 0  
FinPara
```

filas			columnas
0,0	0,1	0,2	0,19
0	0	0	0
1,0	1,1	1,2	1,19
0	0	0	0
2,0	2,1	2,2	2,19
0	0	0	0
...

Array bidimensional

```
Para filas = 0 Hasta 2 Con Paso 1 Hacer  
    Para columnas = 0 Hasta 19 Con Paso 1 Hacer  
        edadesAlumnosClase[filas, columnas] = 0  
    FinPara  
FinPara
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

3. Modificación

Array unidimensional

```
//Solo se pueden guardar números enteros  
edadesAlumnos[1] = 16  
edadesAlumnos[9] = edadesAlumnos[1]
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

3. Modificación

Array unidimensional

```
//Solo se pueden guardar números enteros  
edadesAlumnos[1] = 16  
edadesAlumnos[9] = edadesAlumnos[1]
```

Array bidimensional

```
edadesAlumnosClase[2,19] = 14  
edadesAlumnosClase[1,2] = edadesAlumnosClase[2,19]
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

3. Modificación

Array unidimensional

```
//Solo se pueden guardar números enteros  
edadesAlumnos[1] = 16  
edadesAlumnos[9] = edadesAlumnos[1]
```

columnas			filas
0,0	0,1	0,2					
0	0	0	0,0	...	0,19	0	0,19
0	0	14	1,0	...	1,19	0	1,19
0	2,1	2,2	2,0	...	2,19	14	2,19
0	0	0	2,1	...	2,19	14	2,19

Array bidimensional

```
edadesAlumnosClase[2,19] = 14  
edadesAlumnosClase[1,2] = edadesAlumnosClase[2,19]
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

4. Lectura/Escritura

Array unidimensional

```
Leer edadesAlumnos[2]  
Escribir edadesAlumnos[2]
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

4. Lectura/Escritura

Array unidimensional

```
Leer edadesAlumnos[2]
Escribir edadesAlumnos[2]
```

Array bidimensional

```
Leer edadesAlumnosClase[2,19]
Escribir edadesAlumnosClase[2,19]
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

4. Lectura/Escritura

Array unidimensional

```
Leer edadesAlumnos[2]
Escribir edadesAlumnos[2]
```

Array bidimensional

```
Leer edadesAlumnosClase[2,19]
Escribir edadesAlumnosClase[2,19]
```

filas			columnas
0,0	0,1	0,2	0,19
0	0	0	0
1,0	1,1	1,2	1,19
0	0	14	0
2,0	2,1	2,2	2,19
0	0	0	18

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Operaciones

4. Lectura/Escritura

Array unidimensional

```
Leer edadesAlumnos[2]
Escribir edadesAlumnos[2]
```

Array bidimensional

```
Leer edadesAlumnosClase[2,19]
Escribir edadesAlumnosClase[2,19]
```

columnas			filas
0,0	0,1	0,2	
0	0	0	...
1,0	1,1	1,2	...
0	0	14	...
2,0	2,1	2,2	...
0	0	0	...
			0,19
			0
			1,19
			0
			2,19
			18

```
*** Ejecución Iniciada. ***
> 18
18
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Ejemplo

```
//Definición
Definir numerosAzar, filas, columnas, num Como Entero
Dimension numerosAzar[3,10]

//Inicialización
Para filas = 0 Hasta 2 Con Paso 1 Hacer
    Para columnas = 0 Hasta 9 Con Paso 1 Hacer
        num = azar(10) //Números aleatorios del 0 al 9
        numerosAzar[filas,columnas] = num
    FinPara
FinPara

//Ver todos los números del array bidimensional
Para filas = 0 Hasta 2 Con Paso 1 Hacer
    Para columnas = 0 Hasta 9 Con Paso 1 Hacer
        Escribir numerosAzar[filas,columnas], " " sin saltar
    FinPara
    Escribir ""
FinPara
```

2. Aprendiendo nuevos conceptos (i)

ARRAYS

Ejemplo

```
//Definición
Definir numerosAzar, filas, columnas, num Como Entero
Dimension numerosAzar[3,10]

//Inicialización
Para filas = 0 Hasta 2 Con Paso 1 Hacer
    Para columnas = 0 Hasta 9 Con Paso 1 Hacer
        num = azar(10) //Números aleatorios del 0 al 9
        numerosAzar[filas,columnas] = num
    FinPara
FinPara

//Ver todos los números del array bidimensional
Para filas = 0 Hasta 2 Con Paso 1 Hacer
    Para columnas = 0 Hasta 9 Con Paso 1 Hacer
        Escribir numerosAzar[filas,columnas], " " sin saltar
    FinPara
    Escribir ""
FinPara
```

```
*** Ejecución Iniciada. ***
7 8 1 6 3 3 3 4 4 6
8 2 6 3 1 4 0 2 5 1
0 4 2 3 3 6 5 6 8 8
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Variables de tipo *Texto*

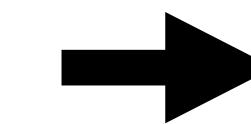
```
Definir nota Como Texto  
nota = "Nota de texto"
```

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Variables de tipo *Texto*

```
Definir nota Como Texto  
nota = "Nota de texto"
```



Cadenas de caracteres

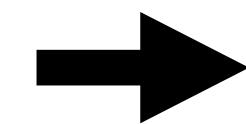
N-o-t-a- -d-e- -t-e-x-t-o
13 caracteres

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Variables de tipo *Texto*

```
Definir nota Como Texto  
nota = "Nota de texto"
```



Cadenas de caracteres

N-o-t-a- -d-e- -t-e-x-t-o
13 caracteres

Array de caracteres

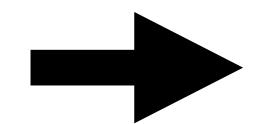
0	1	2	3	4	5	6	7	8	9	10	11	12
N	o	t	a		d	e		t	e	x	t	o

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Variables de tipo *Texto*

```
Definir nota Como Texto  
nota = "Nota de texto"
```



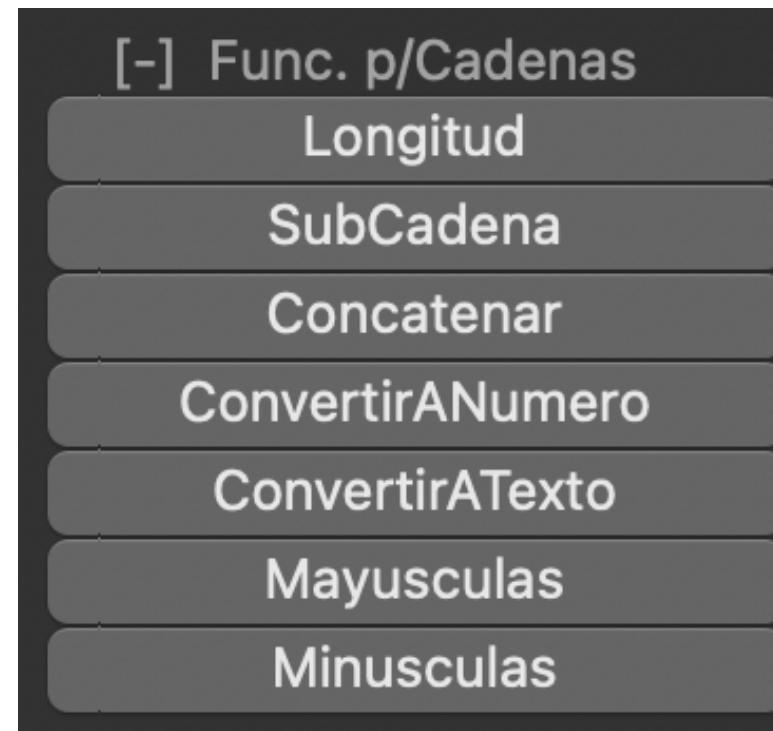
Cadenas de caracteres

N-o-t-a- -d-e- -t-e-x-t-o
13 caracteres

Array de caracteres

0	1	2	3	4	5	6	7	8	9	10	11	12
N	o	t	a		d	e		t	e	x	t	o

Funciones con cadenas/arrays de caracteres en *PseInt*



2. Aprendiendo nuevos conceptos (ii)

CADERAS DE CARACTERES

Funciones con cadenas

1. Longitud (cadena) : devuelve la cantidad de caracteres de la cadena

2. Aprendiendo nuevos conceptos (ii)

CADERAS DE CARACTERES

Funciones con cadenas

1. Longitud (cadena) : devuelve la cantidad de caracteres de la cadena

```
Definir nota Como Texto
Definir long Como Entero
nota = "Nota de texto"

long = Longitud(nota)

Escribir "La longitud de la cadena es: ", long
```

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Funciones con cadenas

1. Longitud (cadena) : devuelve la cantidad de caracteres de la cadena

```
Definir nota Como Texto  
Definir long Como Entero  
nota = "Nota de texto"  
  
long = Longitud(nota)  
  
Escribir "La longitud de la cadena es: ", long
```

```
*** Ejecución Iniciada. ***  
La longitud de la cadena es: 13  
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (ii)

CADERAS DE CARACTERES

Funciones con cadenas

2. SubCadena (cadena, desde, hasta) : devuelve una parte de la cadena, empezando en la posición “desde” (entero) y terminando en la posición “hasta” (entero)

2. Aprendiendo nuevos conceptos (ii)

CADERAS DE CARACTERES

Funciones con cadenas

2. SubCadena (cadena, desde, hasta) : devuelve una parte de la cadena, empezando en la posición “desde” (entero) y terminando en la posición “hasta” (entero)

```
Definir nota, subcad Como Texto
Definir inicio, long Como Entero
nota = "Nota de texto"
inicio = 4
subcad = SubCadena(nota, inicio, 13)
long = Longitud(subcad)

Escribir "La subcadena es:", subcad
Escribir "Longitud subcadena: ", long
```

2. Aprendiendo nuevos conceptos (ii)

CADERAS DE CARACTERES

Funciones con cadenas

2. SubCadena (cadena, desde, hasta) : devuelve una parte de la cadena, empezando en la posición “desde” (entero) y terminando en la posición “hasta” (entero)

```
Definir nota, subcad Como Texto  
Definir inicio, long Como Entero  
nota = "Nota de texto"  
inicio = 4  
subcad = SubCadena(nota, inicio, 13)  
long = Longitud(subcad)
```

```
Escribir "La subcadena es:", subcad  
Escribir "Longitud subcadena: ", long
```

```
*** Ejecución Iniciada. ***  
La subcadena es: de texto  
Longitud subcadena: 9  
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (ii)

CADERAS DE CARACTERES

Funciones con cadenas

2. SubCadena (cadena, desde, hasta) : devuelve una parte de la cadena, empezando en la posición “desde” (entero) y terminando en la posición “hasta” (entero)

```
Definir nota, subcad Como Texto  
Definir inicio, long Como Entero  
nota = "Nota de texto"  
inicio = 4  
subcad = SubCadena(nota, inicio, 13)  
long = Longitud(subcad)  
  
Escribir "La subcadena es:", subcad  
Escribir "Longitud subcadena: ", long
```

```
*** Ejecución Iniciada. ***  
La subcadena es: de texto  
Longitud subcadena: 9  
*** Ejecución Finalizada. ***
```

Nota!!

Se tienen en cuenta los espacios!!!

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Funciones con cadenas

3. Concatenar (cadena, cadena) : devuelve la unión de dos cadenas

2. Aprendiendo nuevos conceptos (ii)

CADERAS DE CARACTERES

Funciones con cadenas

3. Concatenar(cadena, cadena) : devuelve la unión de dos cadenas

```
Definir cadena1, cadena2, cadenaResultado Como Texto
cadena1 = "Ejemplo de"
cadena2 = " concatenación de cadenas"
cadenaResultado = ""

cadenaResultado = Concatenar(cadena1, cadena2)

Escribir "La concatenación es: ", cadenaResultado
```

2. Aprendiendo nuevos conceptos (ii)

CADERAS DE CARACTERES

Funciones con cadenas

3. Concatenar(cadena, cadena) : devuelve la unión de dos cadenas

```
Definir cadena1, cadena2, cadenaResultado Como Texto  
cadena1 = "Ejemplo de"  
cadena2 = " concatenación de cadenas"  
cadenaResultado = ""  
  
cadenaResultado = Concatenar(cadena1, cadena2)  
  
Escribir "La concatenación es: ", cadenaResultado
```

```
*** Ejecución Iniciada. ***  
La concatenación es: Ejemplo de concatenación de cadenas  
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Funciones con cadenas

4. ConvertirANumero (cadena) : convierte un cadena en un número

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Funciones con cadenas

4. ConvertirANumero(cadena): convierte un cadena en un número

```
Definir nota Como Texto  
Definir num Como Entero  
  
nota = "23"  
num = ConvertirANumero(nota)  
  
Escribir "El número es: ", num
```

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Funciones con cadenas

4. ConvertirANumero(cadena): convierte un cadena en un número

```
Definir nota Como Texto  
Definir num Como Entero  
  
nota = "23"  
num = ConvertirANumero(nota)  
  
Escribir "El número es: ", num
```

```
*** Ejecución Iniciada. ***  
El número es: 23  
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Funciones con cadenas

5. ConvertirATexto (numero) : convierte un número en una cadena de caracteres

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Funciones con cadenas

5. ConvertirATexto (numero) : convierte un número en una cadena de caracteres

```
Definir nota Como Texto
Definir num Como Real

num = 34
nota = ConvertirATexto(num)

Escribir "La cadena es: ", nota
```

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Funciones con cadenas

5. ConvertirATexto (numero) : convierte un número en una cadena de caracteres

```
Definir nota Como Texto
Definir num Como Real

num = 34
nota = ConvertirATexto(num)

Escribir "La cadena es: ", nota
```

```
*** Ejecución Iniciada. ***
La cadena es: 34
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Funciones con cadenas

6. Mayusculas (cadena) : devuelve la cadena en mayúsculas

2. Aprendiendo nuevos conceptos (ii)

CADERAS DE CARACTERES

Funciones con cadenas

6. Mayusculas (cadena) : devuelve la cadena en mayúsculas

```
Definir cadena1, cadena2 Como Texto
cadena1 = "Esto es un EJEMPLO"

cadena2 = Mayusculas(cadena1)

Escribir "La cadena en mayúsculas es: ", cadena2
```

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Funciones con cadenas

6. Mayusculas (cadena) : devuelve la cadena en mayúsculas

```
Definir cadena1, cadena2 Como Texto  
cadena1 = "Esto es un EJEMPLO"
```

```
cadena2 = Mayusculas(cadena1)
```

```
Escribir "La cadena en mayúsculas es: ", cadena2
```

```
*** Ejecución Iniciada. ***  
La cadena en mayúsculas es: ESTO ES UN EJEMPLO  
*** Ejecución Finalizada. ***
```

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Funciones con cadenas

7. Minusculas(cadena): devuelve la cadena en minúsculas

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Funciones con cadenas

7. Minusculas(cadena): devuelve la cadena en minúsculas

```
Definir cadena1, cadena2 Como Texto
cadena1 = "Esto es un EJEMPLO"

cadena2 = Minusculas(cadena1)

Escribir "La cadena en minúsculas es: ", cadena2
```

2. Aprendiendo nuevos conceptos (ii)

CADENAS DE CARACTERES

Funciones con cadenas

7. Minusculas(cadena): devuelve la cadena en minúsculas

```
Definir cadena1, cadena2 Como Texto  
cadena1 = "Esto es un EJEMPLO"  
  
cadena2 = Minusculas(cadena1)
```

```
Escribir "La cadena en minúsculas es: ", cadena2
```

```
*** Ejecución Iniciada. ***  
La cadena en minúsculas es: esto es un ejemplo  
*** Ejecución Finalizada. ***
```

**Y ahora...
vamos a ver unos ejemplo sobre lo que
hemos visto**

:-)