

# TEMA 3:

Fundamentos para la construcción de código a partir del algoritmo

# TEMA 3:

## Arreglos unidimensionales estáticos numéricos

# Objetivo

- El alumno construirá programas utilizando el lenguaje de programación C a través de un análisis y modelado algorítmico previo.

# Variables

- Cada vez que declaramos una variable indicamos su tipo de dato y sabemos que únicamente reserva un espacio o cuadrito de memoria para guardar su valor.
- Si deben emplearse 10 datos declaramos 10 variables para reservar 10 espacios o cuadritos de memoria a los que se hace referencia por su nombre.
- La ubicación de los cuadritos es determinada por el sistema operativo y pueden encontrarse en diferentes direcciones de la memoria RAM.

# Utilidad de un Arreglo

- Qué pasa si el programa debe manejar 100 o 1000 datos del mismo tipo?
- Vamos a declarar y operar 1000 variables de nombres diferentes que deben ser además leídas, operadas e impresas?
- Qué pasa si el usuario debe indicar en tiempo de ejecución la cantidad de variables que desea operar y éstas son más o menos de las que declaramos en el programa?

# Utilidad de un Arreglo

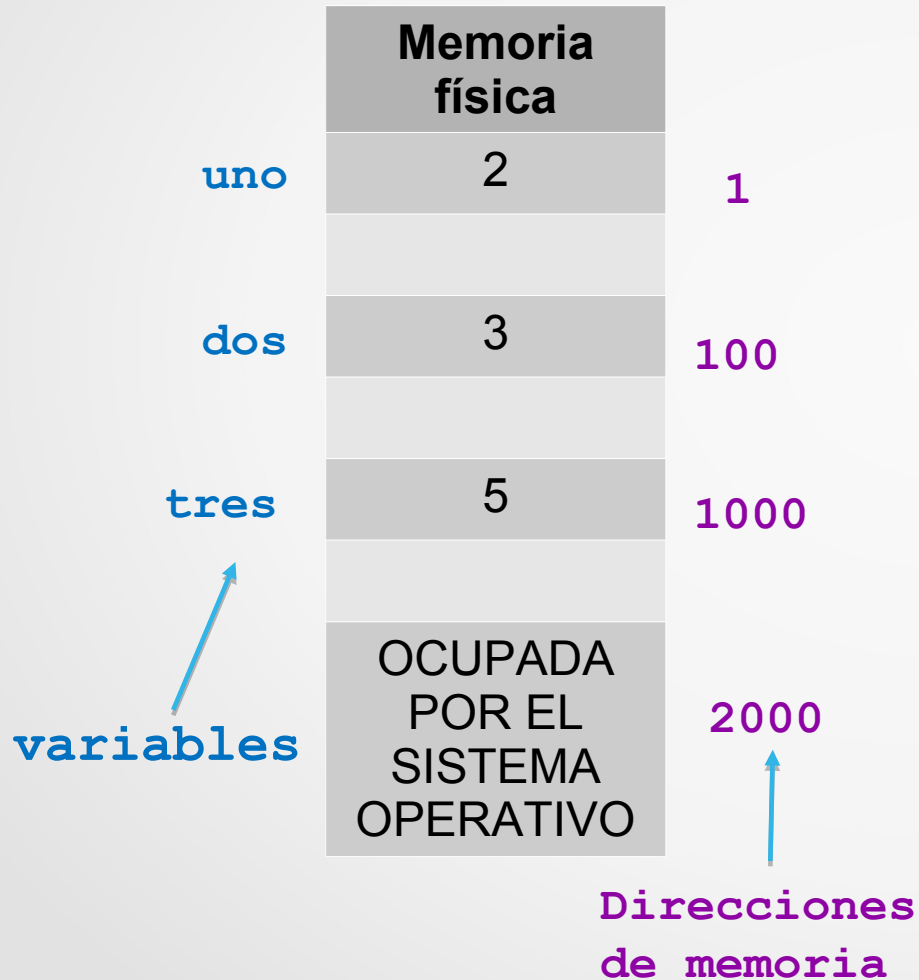
```
1  /*muestra de muchas variables
2   * hecho por huicho*/
3
4  #include <stdio.h> //para printf, scanf y getchar
5
6  int main(int argc, char* argv[])
7  {
8      int dato1=0;
9      int dato2=0;
10     int dato3=0;
11     int dato4=0;
12     .
13     .
14     .
15     int dato100=0;
16
17     scanf("%d", &dato1);
18     scanf("%d", &dato2);
19     .
20     .
21     .
22     scanf("%d", &dato100);
23
24     printf("%d %d %d ... %d", dato1, dato2, dato3, ..., dato100);
25 }
```

# Variables

vs.

# Arreglo

```
int uno=2, dos=3, tres=5;
```



```
int arre[3]={2,3,5};
```



# Arreglo unidimensional estático

- Son cuadritos o espacios de memoria RAM consecutivos del mismo tipo de dato.
- Todos los cuadritos tienen el mismo nombre.
- Cada cuadrilo se identifica por un índice de posición en el arreglo.
- Los índices van de 0 hasta un valor anterior a la cantidad reservada.
- Se declaran indicando su tipo de dato, nombre y tamaño que se cree necesitar entre corchetes.
- NO puede declararse sin un tamaño ni con una variable sin un valor inicial pero si definida previamente.



# Arreglo unidimensional estático

- El sistema operativo reserva el arreglo desde que se ejecuta el programa y la memoria es liberada cuando se finaliza.
- No puede cambiarse su dimensión en tiempo de ejecución.
- El arreglo no se opera al mismo tiempo, debe hacerse un cuadrito a la vez.
- Se emplea cualquier ciclo para recorrer cada elemento del arreglo.
- Pueden ser de varias dimensiones, emplearemos unidimensional (vector) y bidimensional (matriz).
- Si empleamos un cuadrito que no reservamos el sistema operativo puede sobrescribirlo o indicar violación de segmento.

# Cómo usar un arreglo?

- Declaración de un arreglo:

```
int array[3];    //3 espacios  
int arreglo[10]; //10 espacios
```

```
#define MAX 50 //define MAX con valor de 50  
int arreglo[MAX]; //50 espacios por valor de MAX
```

- Impresión de un cuadrado en específico del arreglo:

```
printf("Elemento 0: %d", arreglo[0]);
```

- Lectura de un cuadrado en específico del arreglo:

```
scanf("%d", &arreglo[0]);
```

# Cómo usar un arreglo?

- El primer elemento siempre es el índice cero:

```
int arre[2]; //reserva 2 espacios
```

```
arre[0]=1; //primer elemento es 0 y recibe valor
```

```
arre[1]=2; //segundo elemento es 1 y recibe valor
```

- Hay arreglos de los diferentes tipos de datos:

```
int arreglo[3]; //reserva 3 espacios enteros
```

```
float arr[100]; //reserva 100 espacios reales
```

```
unsigned char nombre[50]; //reserva 50 espacios
```

```
double datos[2000]; //reserva 2000 espacios
```

# Arreglos inicializados

- Los arreglos unidimensionales estáticos numéricos pueden inicializarse al igual que las variables normales pero colocando los valores entre llaves y separados por comas.

```
int arreglo[3]= {1,3,5};
```

- Los arreglos unidimensionales estáticos de caracteres pueden inicializarse indicando o no el tamaño a ocupar, pero siempre con un espacio adicional para almacenar el carácter de terminación nulo ('`\0`') que se asigna automáticamente al leer del teclado o manualmente al leer de un archivo o al inicializar carácter por carácter.

```
char mensaje1[]= "hola";
```

```
char mensaje2[5]= "hola";
```

```
char mensaje3[5]= {'h','o','l','a','\0'};
```

# Arreglo unidimensional en DFD, PSeInt y C

- En **DFD** se declara con el nombre de la variable y **entre paréntesis la cantidad de elementos** a reservar.
- En **PSeInt** se declara con el nombre de la variable y su tipo de dato, después la palabra reservada **Dimension** variable y **entre corchetes la cantidad de elementos** a reservar.
- En **lenguaje C** se declara con el tipo, nombre de la variable, **entre corchetes la cantidad de elementos** a reservar e inicializar si se desea.
- Puede imprimirse por separado indicando el índice del elemento.
- Puede recorrerse fácilmente empleando un ciclo **Para** o **for** porque se conocen los 3 elementos con un contador que indique el índice de posición y avance al siguiente.

# Arreglo unidimensional de 3 elementos en DFD, PSeInt y C

- El contador **conde** se declara como **entero** o **entero corto** e inicializa de acuerdo a la herramienta y ciclo empleado.
- El contador **conde** **comienza a contar en 0** (cero) porque el primer elemento del arreglo es el índice 0.
- El contador **conde** **se detiene en el índice 2** porque el arreglo fue solicitado de 3 elementos.
- El contador **conde** **incrementa en 1** porque los índices son enteros consecutivos.

# Arreglo unidimensional en DFD

- Se recorre empleando un ciclo **Para**
- El contador **conde** se declara como entero e inicializa
- **conde** comienza en 0 (cero)
- **conde** se detiene en el índice 2
- **conde** incrementa en 1

● Salida por pantalla



Salida:

Elemento 0: 1

● Salida por pantalla



Salida:

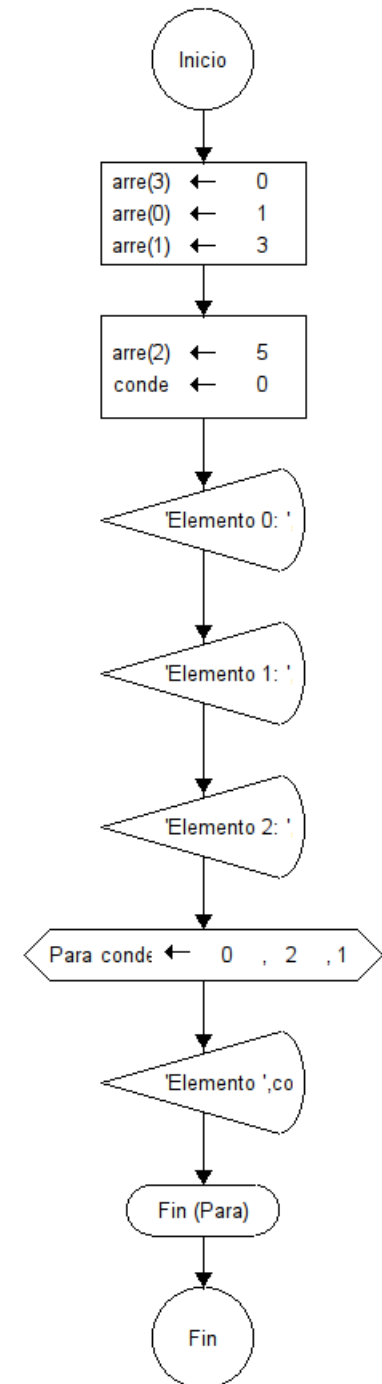
Elemento 1: 3

● Salida por pantalla



Salida:

Elemento 2: 5



# PSeInt y ejecución

```
1  Algoritmo arregloUniNum
2      Definir conde Como Entero
3      Definir arreglo como Entero
4      Dimension arreglo[3]
5
6      arreglo[0]= 1
7      arreglo[1]= 3
8      arreglo[2]= 5
9
10     Escribir 'Este algoritmo emplea un arreglo entero de 3'
11
12     Escribir 'Elemento 0: ',arreglo[0]
13     Escribir 'Elemento 1: ',arreglo[1]
14     Escribir 'Elemento 2: ',arreglo[2]
15
16     Escribir ''
17     Escribir 'El arreglo con ciclo es: '
18     Para conde<-0 Hasta 2 Con Paso 1 Hacer
19         Escribir 'Elemento ',conde,': ',arreglo[conde]
20     Fin Para
21
22 FinAlgoritmo
```

● ● ● PSeInt - Ejecutando proceso ARREGLOUNINUM

```
*** Ejecución Iniciada. ***
Este algoritmo emplea un arreglo entero de 3
Elemento 0: 1
Elemento 1: 3
Elemento 2: 5

El arreglo con ciclo es:
Elemento 0: 1
Elemento 1: 3
Elemento 2: 5
*** Ejecución Finalizada. ***
```



# Arreglo en C y ejecución

- Se recorre empleando un ciclo **for**
- El contador **conde** se declara como **short** o entero corto
- **conde** comienza en 0 (cero)
- **conde** se detiene en el índice 2
- **conde** incrementa en 1

```
1  /*programa que muestra arreglo por separado y con ciclo for
2  * hecho por Huicho*/
3
4  #include <stdio.h> //para printf y getchar
5
6  int main(int argc, char* argv[])
7  {
8      //declaración de arreglo de tamaño 3 con valores iniciales
9      int arreglo[3]= {1, 3, 5};
10     short conde; //contador para recorrer elementos del arreglo
11
12     printf("Este programa emplea un arreglo entero de 3 \n");
13
14     printf("\n Elemento 0: %d", arreglo[0]);
15     printf("\n Elemento 1: %d", arreglo[1]);
16     printf("\n Elemento 2: %d", arreglo[2]);
17
18     printf("\n\n El arreglo con ciclo es: \n");
19
20     //el ciclo debe recorrer los índices 0, 1 y 2
21     for (conde=0; conde<=2; conde++)
22     {
23         printf("\n Elemento %hd: %d", conde, arreglo[conde]);
24     }
25
26     getchar();
27     return 0;
28 }
```



```
/bin/bash
Este programa emplea un arreglo entero de 3

Elemento 0: 1
Elemento 1: 3
Elemento 2: 5

El arreglo con ciclo es:

Elemento 0: 1
Elemento 1: 3
Elemento 2: 5
```

Hecho por Huicho :)

# Cómo leer un arreglo?

- Bloque de código para leer incluyendo ciclo, petición y lectura.
- El tope del ciclo es 2 por ser declarado de 3 elementos o puede ser una variable con la cantidad dada por el usuario.

```
float a[3]; //los 3 datos son reales
```

```
short conde; //el contador es entero corto
```

```
for (conde=0; conde<=2; conde++)  
{  
    printf("Ingrese elemento[%hd]: ", conde);  
    scanf("%f", &a[conde]);  
}
```

# Cómo imprimir un arreglo?

- Bloque de código para imprimir incluyendo ciclo.
- El tope del ciclo es 2 por ser declarado de 3 elementos o puede ser una variable con la cantidad dada por el usuario.

```
float a[3]; //los 3 datos son reales
```

```
short conde; //el contador es entero corto
```

```
for(conde=0; conde<=2; conde++)  
{  
    printf("\n\t Elemento[%hd] : %f", conde, a[conde]);  
}
```

# Suma de vectores

```
1  /*programa que muestra suma de 2 vectores de 3 elementos con ciclos for
2   * hecho por huicho*/
3
4  #include <stdio.h> //para printf, scanf y getchar
5
6  int main(int argc, char* argv[])
7  {
8      float a[3], b[3], c[3];
9      short conde;
10
11     printf("Programa que suma 2 vectores de 3 elementos \n\n");
12
13     printf("Primer vector a: \n");
14     for(conde=0; conde<=2; conde++) //ciclo para leer a
15     {
16         printf("Ingrese el valor de a[%hd]: ", conde);
17         scanf("%f", &a[conde]);
18     }
19
20     printf("\nSegundo vector b: \n");
21     for(conde=0; conde<=2; conde++) //ciclo para leer b
22     {
23         printf("Ingrese el valor de b[%hd]: ", conde);
24         scanf("%f", &b[conde]);
25     }
26
27     for(conde=0; conde<=2; conde++) //ciclo para sumar a y b
28     {
29         c[conde]= a[conde] + b[conde];
30         printf("\n c[%hd]= %f + %f = %f", conde, a[conde], b[conde], c[conde]);
31     }
32
33     printf("\n\n a(%g, %g, %g)", a[0], a[1], a[2]);
34     printf(" + b(%g, %g, %g)", b[0], b[1], b[2]);
35     printf(" = c(%g, %g, %g)", c[0], c[1], c[2]);
36
37     getchar();
38     getchar();
39     return 0;
40 }
```

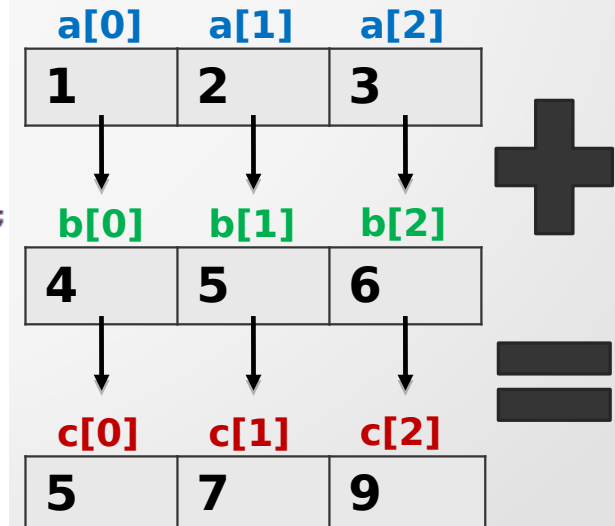
```
/bin/bash
Programa que suma 2 vectores de 3 elementos

Primer vector a:
Ingrese el valor de a[0]: 1
Ingrese el valor de a[1]: 2
Ingrese el valor de a[2]: 3

Segundo vector b:
Ingrese el valor de b[0]: 4
Ingrese el valor de b[1]: 5
Ingrese el valor de b[2]: 6

c[0]= 1.000000 + 4.000000 = 5.000000
c[1]= 2.000000 + 5.000000 = 7.000000
c[2]= 3.000000 + 6.000000 = 9.000000

a(1, 2, 3) + b(4, 5, 6) = c(5, 7, 9)
```



Hecho por Huicho :)

# Preguntar al usuario el tamaño

- Un arreglo estático no puede reservar el tamaño exacto dado en tiempo de ejecución.
- Puede definirse en el código un tamaño suficientemente grande y preguntar al usuario el número de elementos que desea dentro del rango previamente reservado.

```
#define MAX 50 //define MAX con valor de 50  
int arreglo[MAX]; //50 espacios por valor de MAX
```

- Mientras el programa esté en ejecución el sistema operativo no puede reasignar el espacio solicitado para el arreglo se encuentre o no ocupado en su totalidad.



```

1  /*programa que lee y muestra arreglo de n elementos con ciclos for
2  * hecho por huicho*/
3
4  #include <stdio.h> //para printf, scanf y getchar
5
6  #define MAX 100
7
8  int main(int argc, char* argv[])
9  {
10     float arreglo[MAX]; //arreglo para reales con tamaño máximo de 100
11     short conde; //contador que recorre cada elemento del arreglo
12     short cuantos; //guarda la cantidad de elementos del usuario
13
14     printf("Programa que lee e imprime arreglo de n elementos \n\n");
15     printf("Ingresa la cantidad de elementos: ");
16     scanf("%hd", &cuantos);
17
18     printf("\n Arreglo a: \n\n");
19     for(conde=0; conde<=cuantos-1; conde++) //también conde<cuantos
20     {
21         printf("Ingrese el valor del elemento[%hd]: ", conde);
22         scanf("%f", &arreglo[conde]);
23     }
24
25     printf("\n Arreglo a: \n");
26     for(conde=0; conde<=cuantos-1; conde++) //también conde<cuantos
27     {
28         printf("\n\t Elemento[%hd]: %g", conde, arreglo[conde]);
29     }
30
31     getchar();
32     getchar();
33     return 0;
34 }

```

- El tope del ciclo puede ser `conde<=cuantos-1` o `conde<cuantos`
- El tamaño máximo del arreglo es de 100 elementos, desde el índice 0 hasta el 99
- **Es muy importante avisar al usuario la cantidad máxima de elementos y validar que la cantidad ingresada se encuentre en el rango permitido.**

```

/bin/bash
Programa que lee e imprime arreglo de n elementos
Ingresa la cantidad de elementos: 5

Arreglo a:

Ingrese el valor del elemento[0]: 1
Ingrese el valor del elemento[1]: 11
Ingrese el valor del elemento[2]: 111
Ingrese el valor del elemento[3]: 1111
Ingrese el valor del elemento[4]: 11111

Arreglo a:

Elemento[0]: 1
Elemento[1]: 11
Elemento[2]: 111
Elemento[3]: 1111
Elemento[4]: 11111

```

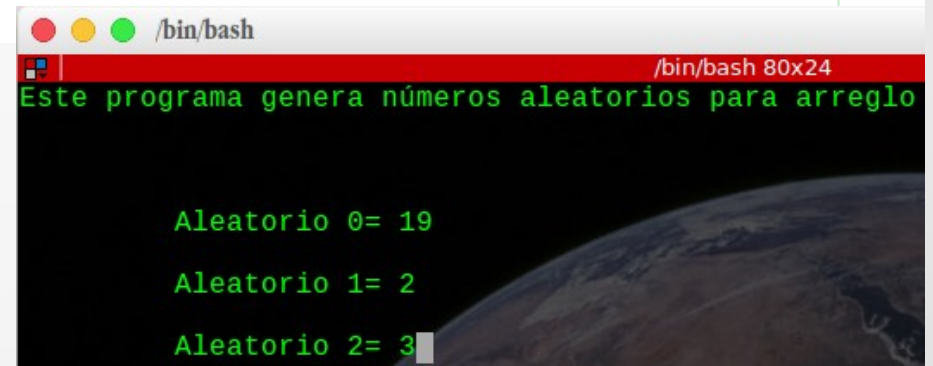
# Operaciones sobre arreglos

OPERACIÓN	VARIABLE	ARREGLO UNIDIMENSIONAL
Asignación de valor	<code>x=0 ;</code>	<code>arreglo[0]= 0 ;</code>
Comparación	<code>if (x==0)</code>	<code>if (arreglo[0]==0)     neutros++; else if (arreglo[0]&gt;0)     positivos++; else if (arreglo[0]&lt;0)     negativos++;</code>
Acumulador para multiplicar	<code>x= x * 5 ;</code>	<code>arreglo[0]= arreglo[0]*5;</code>
Acumulador para sumar	<code>x= x + conde ;</code>	<code>arre[0]= arre[0] + conde;</code>

# Números aleatorios

- Incluir `time.h`
- Incluir `stdlib.h`
- `srand(time(NULL))` ;  
genera el número semilla que cambia con el tiempo y a partir del cual se obtiene una serie de pseudoaleatorios.
- `rand() % limite`  
genera números enteros en el rango de 0 hasta el valor de *limite-1*

```
1  /*programa para asignar pseudoaleatorios a un arreglo
2  * hecho por huicho*/
3
4  #include <stdio.h> //para printf, scanf y getchar
5  #include <time.h> //para time
6  #include <stdlib.h> //para srand, rand
7
8  int main(int argc, char* argv[])
9  {
10     int aleatorio[3]={0,0,0};
11     short conde=0;
12
13     srand(time(NULL)); //genera número semilla a partir del tiempo
14
15     printf("Este programa genera n%cmoros aleatorios para arreglo \n\n", 163);
16     for(conde=0; conde<=2; conde++)
17     {
18         aleatorio[conde]= rand() % 50; //genera un aleatorio desde 0 hasta 49
19         printf("\n\n\t Aleatorio %hd= %d", conde, aleatorio[conde]);
20     }
21
22     getchar();
23     return 0;
24 }
```



The screenshot shows a terminal window titled `/bin/bash` with a red header bar. The output of the program is displayed in green text on a black background. The first line of output is `Este programa genera números aleatorios para arreglo`. The subsequent lines show the generated random numbers for each index of the array: `Aleatorio 0= 19`, `Aleatorio 1= 2`, and `Aleatorio 2= 3`. A cursor is visible at the end of the last line.