UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Unidad 2. La lógica de programación

Tema 4. Arreglos y estructuras de repetición

UNIVERSIDAD DE LOS ANDES

Arreglos y estructuras de repetición

- Contenido:
  - Introducción
  - Arreglos
    - Vectores
    - Matrices
    - De más de 2 dimensiones
  - Estructura de repetición
    - Repita hasta o hacer mientras
  - Representación algorítmica
  - Codificación

- Objetivo:

  Desarrollar habilidades en el uso de vectores, matrices y las estructuras de repetición

- Bibliografía:
  - Deitel y Deitel, cap. 4 y 6.
  - http://www.ing.ula.ve/~ibc/pr1
  - http://www.webdelprofesor.ula.ve/ingenieria/ibc
  - Navas y Besembel, tema V.
  - Joyanes, sec. 4.6, 4.8 y cap. 6.

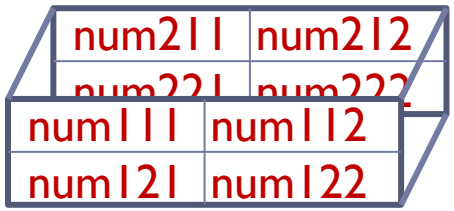
Tipos y estructura de un arreglo

- Tipos de arreglos:
  - Vector (arreglo unidimensional / 1D), un (1) subíndice
  - Matriz (arreglo bidimensional / 2D), dos (2) subíndices
  - Multidimensional (tres / 3D o más dimensiones), tres (3) o más subíndices
- Estructura de un arreglo:
  - Tipo de dato del arreglo
  - Nombre de referencia que indica la dirección base de memoria de inicio del arreglo
  - Número máximo (esperado o efectivo) de elementos que contendrá el arreglo

Tipo nombreDelArreglo[tamañoMáximo];

Ejemplo: Almacenar ocho (8) números enteros num

con un arreglo de 2x2x2 int num[2][2][2];



UNIVERSIDAD DE LOS ANDES

Operaciones sobre arreglos

- Leer el valor de un elemento del arreglo

  nombreDelArreglo[posiciónDelElemento]

- Escribir el valor de un elemento en el arreglo

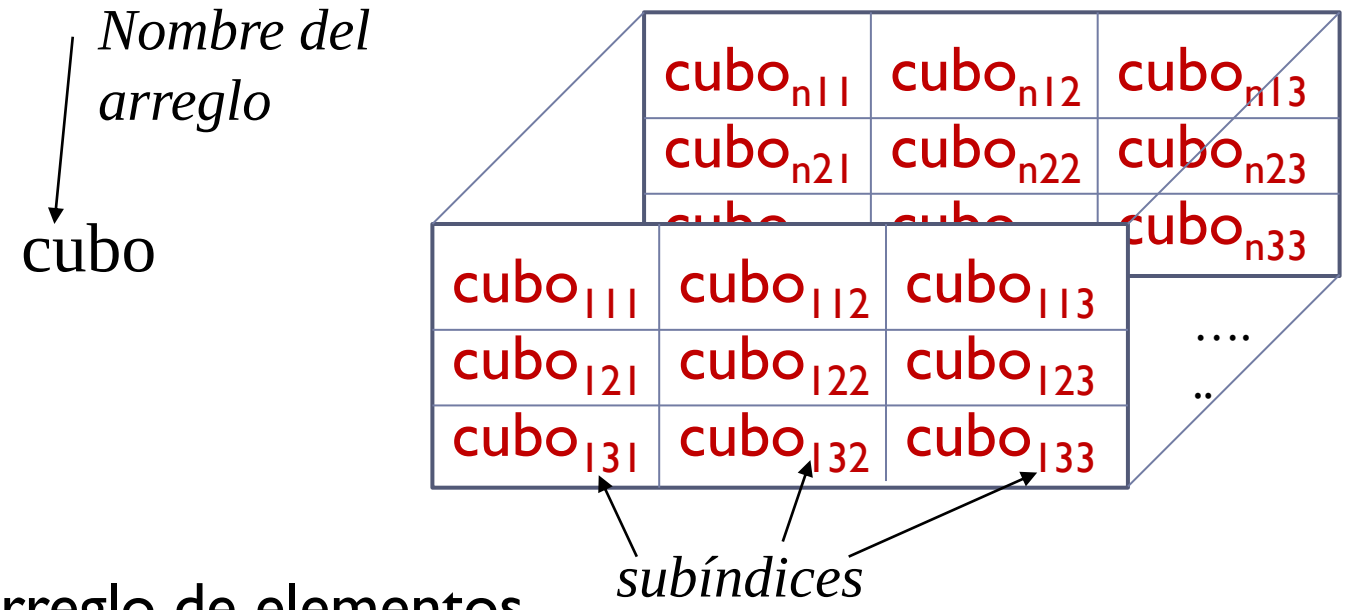
  nombreDelArreglo[posiciónDelElemento]=valor

Arreglo tridimensional / 3D y arreglos multidimensionales

- Grupo de localidades consecutivas de memoria relacionadas por su nombre de referencia y su tipo de dato
- Cada localidad o grupo de localidades almacena un elemento del arreglo
- Se necesita un subíndice para cada dimensión del mismo

Notación algorítmica del arreglo dD

- Cada elemento del arreglo es accedido mediante el nombre del mismo y un subíndice que representan la posición numérica (entero no negativo) de dicho elemento dentro del arreglo



- Crear un arreglo de elementos

TipoDeDato nombreArreglo[num1D][num2D]…[numdD]

Real cubo[3][3][7]

Cubo de 63 elementos

Declaración de un arreglo en C/C++

tipoDeDato nombreArreglo[num1D][num2D]…[numdD];

- Ejemplos:

```
float   x[6][4][5];      // Arreglo real de 120 de elementos
char    w[4][10][2];     // Arreglo de caracteres de 80 elementos
int     w[10][3][2][4];  // Arreglo de 240 elementos enteros
```

Codificación C/C++

```
#define  NUMDIAS  31
#define  NUMHORAS  24
#define  NUMMESES  12

int    anio[NUMHORAS][NUMDIAS][NUMMESES];
```

Dibuje el arreglo anio

Acceso a los elementos de un arreglo

Cada elemento de un arreglo puede usarse como una variable cualquiera

Ejemplos:

```
X[i][j][k]=4;
printf("valor= %i", nx[2][2][3]);
cin>>d[2][1][1][2];
cout<<"Valor de P = "<<P[2][1][1]<<endl;
```

Iniciación de los elementos de un arreglo

Todos los elementos del arreglo tienen asignado un valor inicial

```
int     max[7][4][2][2] = {0};
    max[0][0][0][0]=max[0][0][0][1]=max[0][0][1][0]= … =matriz[6][3][1][1]=0
float   x[5][2][2]={0.34, 0.45, 0.67, 0.89};
    x[0][0][0]=0.34        x[0][0][1]=0.45    x[0][1][0]=0.67  x[0][1][1]=0.89
```

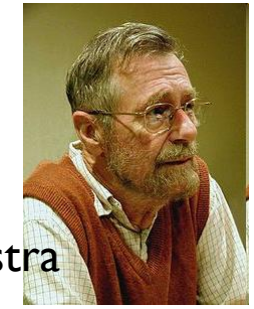
No todos los elementos tienen valor inicial

Programación estructurada

- Enfoque disciplinado que permite escribir programas estructurados, utilizando las siguientes tres estructuras de control:
  - Secuencial (asignación, lectura, escritura)
  - Decisión o selección (simple, doble, múltiple)
  - Repetición
    - Repita para
    - Repita mientras
    - **Repita hasta**

Dahl, Dijkstra y Hoare. Structured Programming. Academic Press, 1972

Dahl 

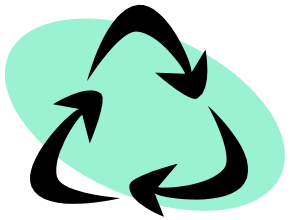
Dijkstra 

Hoare 

UNIVERSIDAD DE LOS ANDES

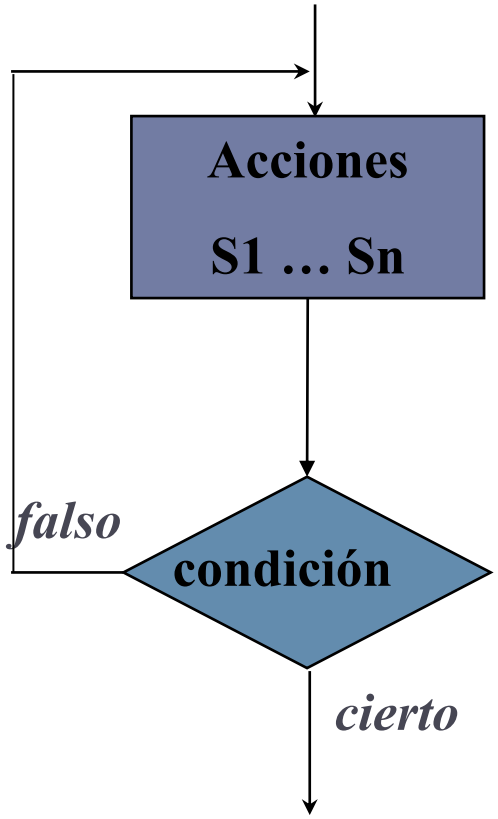
Estructuras de repetición

- Permiten que un conjunto de sentencias (una o varias) de un programa sea ejecutado por la computadora en forma repetida
- Tipos de estructuras de repetición
  - Repetición controlada por un contador:
    - Repita para
  - **Repetición condicional o controlada por un centinela:**
    - Repita mientras (verificación al inicio del bloque)
    - **Repita hasta (verificación al final del bloque)**



Repita hasta

Diagrama de flujo



Pseudocódigo en español

Repita

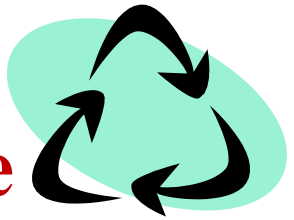
S_1

….

S_n

Hasta (<condición>)

Código en C++

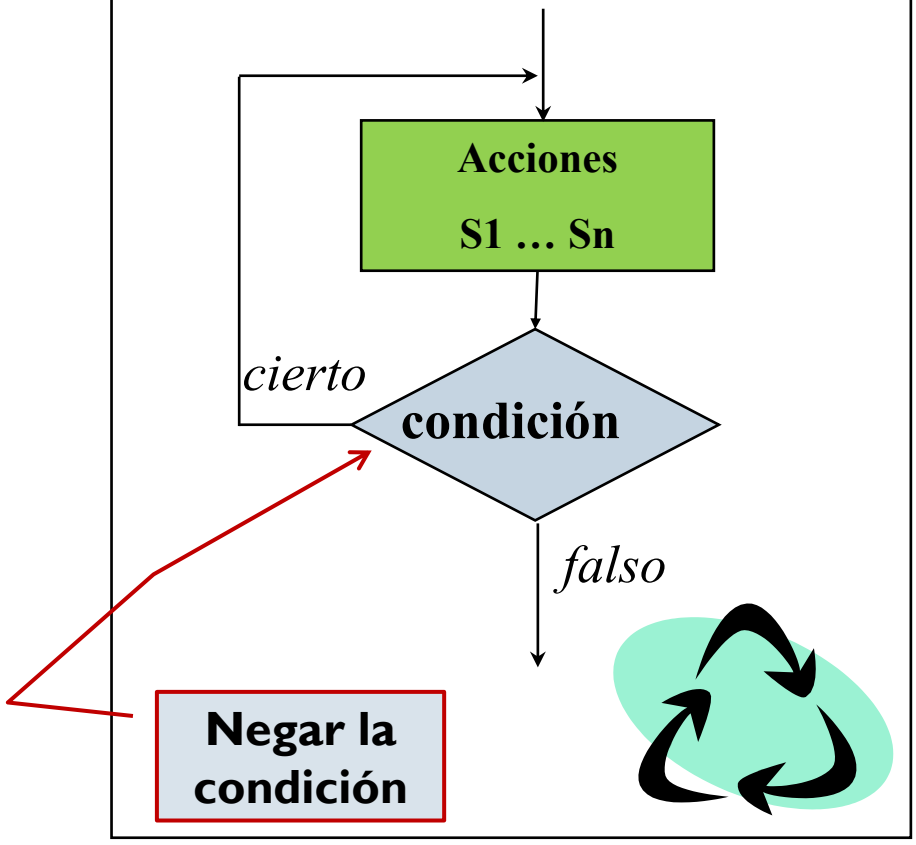


no existe

UNIVERSIDAD DE LOS ANDES

Hacer mientras

Diagrama de flujo



Pseudocódigo en español

```
Hacer
        S1
        ....
        Sn
mientras (<condición>)
```

Código en C/C++

```
do
{
        S1
        ....
        Sn
} while (<condición>);
```

TDSO [X1,...,Xn = E1,...,En] (C)

UNIVERSIDAD DE LOS ANDES

Hacer mientras

- La estructura "Hacer-Mientras" es una sentencia "Repita mientras" con la diferencia que evalúa la condición al final del lazo y no al principio
- Las sentencias (una o más) del cuerpo del lazo se ejecutan mientras que la condición (expresión lógica) es **cierta**
- Como se pregunta al final por la condición, el lazo se ejecuta una o más veces
- Si la condición nunca se hace falsa, el programa entra en un lazo infinito, es decir, las sentencias del cuerpo del lazo se ejecutarán indefinidamente

HACER MIENTRAS C SE CUMPLA

[X1,...,Xn = E1,...,En] (C)

UNIVERSIDAD DE LOS ANDES

Ejemplo 1

Pseudocódigo en Español	Código en C/C++
[Desplegar " a es mayor"](a ≥ b)	do cout << "a es mayor\n"; while(a >= b);

conta=0
[conta=conta+1
nota = valor suministrado
Desplegar conta, nota
](conta ≤ 15)
	conta=0;
	do
	{ conta++;
	cin >> nota;
	cout << conta << nota <<
	endl;
	}
	while(nota <= 15);

Nota: Si hay mas de 1 sentencia en el bloque, se escribe { } en C/C++

UNIVERSIDAD DE LOS ANDES

Corrida en frío

```
#include <iostream>
using namespace std;

void main ()
{   int     i = 1;
    do
    {     cout <<  i << endl;
          i++;
    } while ( i <= 3 );
    cout << ”Escribe los números 1, 2 y 3” << endl;
}
```

Corrida en frío

iteración	i
(0)	1
(1)	2
(2)	3
(3)	4

UNIVERSIDAD DE LOS ANDES

Ejercicio resuelto 1

- Enunciado del problema

  Sumar todos los enteros pares desde 2 hasta 100

- Análisis E-P-S

Entrada	Variable	Descripción	Tipo de dato	Rango válido
	N/A			
Proceso	para cada numero par generado entre 2 y 100 acumular su valor en el acumulador de pares (ap)			
Salida	Mensaje indicando ap	Acumulador de pares	Entero	>0

Ejercicio resuelto 1. Diseño

	sumaParesRM	
	{pre: }	{pos: AP > 0}
1 2 3	AP, num=0, 2 [AP, num = AP + num, num + 2] (num < 100) Desplegar "Suma de los números pares entre 2 y 100 = ", AP	•**AP**: Entero+. Acumulador de números pares •**num:** Entero+. Contador de números pares
1	-> Suma de los números pares entre 2 y 100 = 2550	Caso exitoso

Completar la tabla

num	AP	num	AP	num	AP	num	AP	num	AP	num	AP	num	AP	num	AP
	0	14	56	28	210	42	462	56		70		84		98	
2	2	16	72	30	240	44	506	58		72		86		100	2250
4	6	18	90	32	272	46	552	60		74		88			
6	12	20	110	34	306	48	600	62		76		90			
8	20	22	132	36	342	50	650	64		78		92			
10	30	24	156	38	380	52	702	66		80		94			
12	42	26	182	40	420	54	756	68		82		96			

Ejercicio resuelto 1. Implementación

```
#include <stdio.h>
int main()
{
   unsigned int          AP = 0, num=2;
   do
   {      AP += num;
          num +=2;
   } while ( num <= 100);
   printf("Suma de numeros pares entre 2 y 100 = %d\n", AP);
   return 0;
}
```

UNIVERSIDAD DE LOS ANDES

Resumen

Tipo nombreDelArreglo[tamañoMáximo]= {listaValoresIniciales};

nombreDelArreglo[posiciónDelElemento]

nombreDelArreglo[posiciónDelElemento]=valor

Código en C/C++

```
do
{
        S1
        ....
        Sn
} while (<condición>);
```

"Hacer mientras" se utiliza mayormente para validar datos de entrada

HACER MIENTRAS C SE CUMPLA

[X1,...,Xn = E1,...,En] (C)

UNIVERSIDAD DE LOS ANDES

Ejercicio resuelto 2

- Enunciado del problema

  Dadas las notas de n estudiantes de las materias de Programación 1, Física11 y Cálculo 10. Calcular el promedio de notas de cada estudiante

- Análisis E-P-S

Entrada	Variable	Descripción	Tipo de dato	Rango válido
	n notas[n][4]	Numero de estudiantes Notas de Programación 1, Física11, Cálculo 10 y promedio	Entero+ MatrizDeReal	≥0 notas[i][j]∈[0, 20]
Proceso	Para cada estudiante hacer: Calcular notas[i][4] = $\frac{notas[i][1]+notas[i][2]+notas[i][3]}{3}$			
Salida	Mensaje indicando el valor de promedio para cada estudiante	Promedio de las tres notas dadas para cada estudiante	VectorDeReal	[0, 20]

Ejercicio resuelto 2. Diseño

promedio3notas

{pre: $n \in \aleph$} {pos: $n \in \aleph$, $\forall i$, $PR1_i$, $FI11_i$, $CA10_i$, $promedio_i \in \Re$ }

1 2	[Desplegar "Introduzca el numero de estudiantes ≥ 1" n = valor suministrado] (n <0) [Desplegar "Para el estudiante ", k [Desplegar "Introduzca la nota de PR1 entre 0 y 20" notas[k][1] = valor suministrado] (notas[k][1] <0 ∨ notas[k][1] > 20) [Desplegar "Introduzca la nota de Fisica1 entre 0 y 20" notas[k][2] = valor suministrado] (notas[k][2] < 0 ∨ notas[k][2] > 20) [Desplegar "Introduzca la nota de Cálculo 1 entre 0 y 20" notas[k][3] = valor suministrado] (notas[k][3] < 0 ∨ notas[k][3] > 20) Desplegar "Promedio de notas del estudiante ", k, " = ", notas[k][4]= (notas[k][1]+notas[k][2]+notas[k][3])/3] k = 1, n, 1	● **n:** Natural. Número total de estudiantes de la asignatura. ● **k:** Natural. Contador de estudiantes. ● notas: MatrizDeReal[n][4]. Notas de las materias Programación 1, Física11, Cálculo 10 y su promedio para cada estudiante. Cada nota debe estar en el rango de 0 a 20.
1 2	n = 1, notas[1][1]=12, notas[1][2]=10, notas[1][3]=11 => Promedio de notas del estudiante 1=11 n = 2, notas[1][1]=11, notas[1][2]=11, notas[1][3]=11 => Promedio de notas del estudiante 1=11 notas[2][1]=1, notas[2][2]=1, notas[2][3]=1 => Promedio de notas del estudiante 2=1	Caso límite Caso exitoso

Compare esta solución con la de la clase 7 para el mismo enunciado

UNIVERSIDAD DE LOS ANDES

Codificación en C++

```
#include <iostream>
using namespace std;
#define  ME  50
int    main ( )
{      int n, k;
       float       notas[ME][4];
       do
       {      cout << "Introduzca el numero de estudiantes, entero mayor que 0 y menor que \n"<< ME;
              cin >> n;
       } while(n < 0 || n > ME);
       for(k=1; k<=n; k++)
       {      cout << "Introduzca para el estudiante"<<k << endl;
              do
              {      cout<<"Introduzca la nota de Programamcion 1"<<endl;
                     cin >> notas[k][0];
              } while(notas[k][0] < 0.0 || notas[k][0] > 20.0);
              do
              {      cout << "Introduzca la nota de Fisica1" << endl;
                     cin >> notas[k][1];
              } while(notas[k][1] < 0.0 || notas[k][1] > 20.0);
```

UNIVERSIDAD DE LOS ANDES

Ejercicio resuelto 2

```
        do
        {   cout << "Introduzca la nota de Calculo 1" << endl;
            cin >> notas[k][2];
        } while(notas[k][2] < 0.0 || notas[k][2] > 20.0);
        notas[k][3] = (notas[k][0]+notas[k][1]+notas[k][2])/3.0;
        cout << "Promedio del estudiante " << k << "= " << notas[k][3] << endl;
    }
    return 0;
}
```

Proponga otros diseños junto con sus implementaciones al mismo problema

- ¿Qué es un arreglo multidimensional?
- ¿Cómo se declaran e inician los arreglos?
- ¿Cómo se define una estructura de repetición hacer mientras y su diferencia con el repita hasta?
- ¿Cómo se prueba en frío un programa con hacer mientras?
- ¿Cómo se anidan las estructuras de decisión y de repetición?

Resumen

¿Cuáles son los conceptos relevantes de esta clase?

UNIVERSIDAD DE LOS ANDES

Ejercicios

- **Para cada uno de los enunciados dados a continuación realizar: Análisis en E-P-S, diseño en pseudocódigo y codificación en C o C++**

1. Ordene descendentemente un conjunto de valores dados que pertenecen al intervalo [2, 120]
2. Encuentre todas las letras 'p' en un conjunto de letras dadas
3. Ordene de menor a mayor un conjunto de letras dadas
4. Encuentre la menor y mayor letra de un conjunto de letras
5. Encuentre el menor valor para un punto en el espacio definido por 3 coordenadas X, Y, Z. Se define el menor punto en 3D a aquel punto que esté mas cerca del origen (0,0,0), es decir X = 0, Y = 0, Z = 0