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
  - Estructura de repetición
    - Repita para
  - 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.

UNIVERSIDAD DE LOS ANDES

Tipos de datos

- Simples: Almacenan un solo valor (enteros, reales, caracteres, apuntadores y lógicos)
- Compuestos o estructurados: almacenan uno o más valores (arreglos, cadenas de caracteres, registros)
  - Se utilizan cuando se requiere el procesamiento de múltiples datos que tienen características comunes
  - Ejemplos: Un conjunto de números enteros, un conjunto de estudiantes, un conjunto de temperaturas, un conjunto de fechas, etc.
- **Arreglo:** Generalización del concepto de variable
  - Variable: puede tener máximo un (1) valor y cada variable se referencia con un nombre
  - Arreglo: representa un conjunto de valores (caracteres, enteros, reales, etc.) donde todos comparten el mismo nombre de referencia y cada valor se referencia con uno o más subíndices

UNIVERSIDAD DE LOS ANDES

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 cuatro (4) números enteros `int num1, num2, num3, num4;`

con un vector `int num[4];` num 

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

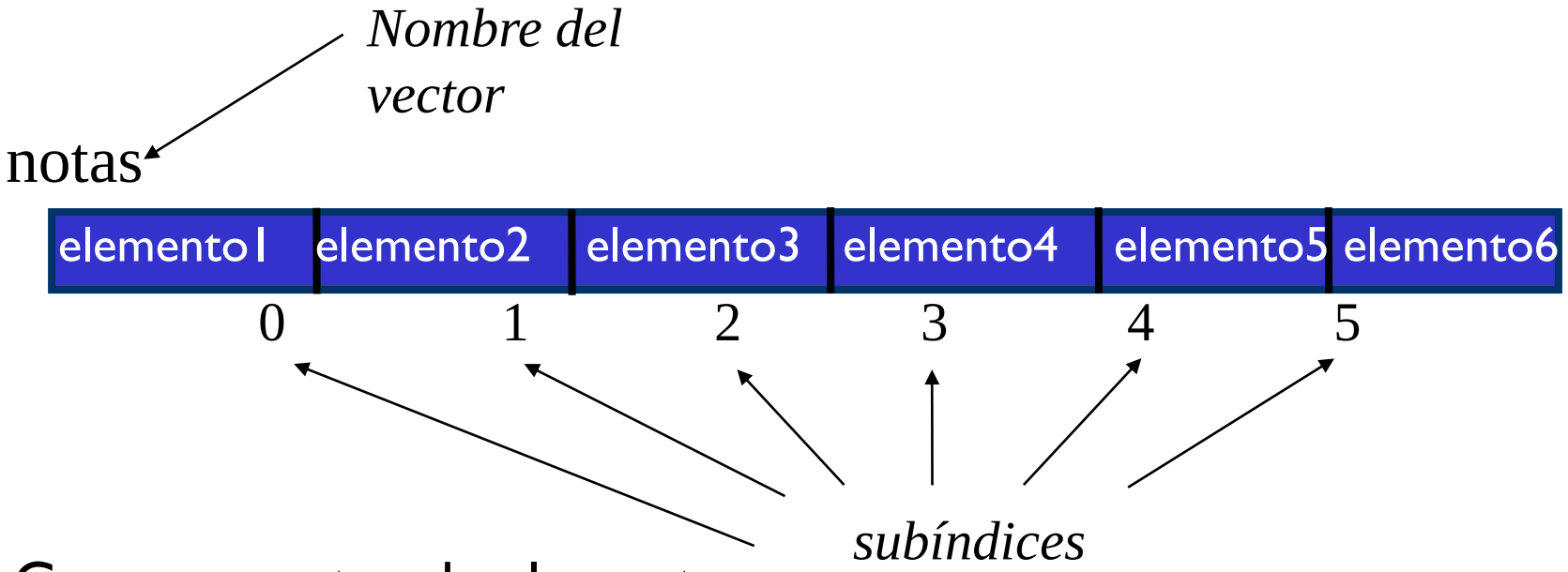
  nombreDelArreglo[posiciónDelElemento]=valor

Vector (Arreglo unidimensional / 1D)

- 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 vector

Notación algorítmica del vector

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



- Crear un vector de elementos

TipoDeDato nombreDelVector[númeroDeElementos]

Real notas[6]

UNIVERSIDAD DE LOS ANDES

TAD Vector

05/07/11 Especificación (6) Vector[DIM]De TipoDeDato		
1	Sintáctica: creaVector(Entero+, TipoDeDato) -> Vector[DIM], leeElemento(Vector[DIM], Entero+) -> TipoDeDato, escribeElemento(Vector[DIM], Entero+, TipoDeDato) -> Vector[DIM],	creaVector(): Crea un arreglo unidimensional con el número de posiciones y tipo de dato suministrados por los parámetros, sin limpiar la memoria. leeElemento(): Devuelve el elemento que está en la posición solicitada en el parámetro. escribeElemento(): Escribe el elemento enviado por el parámetro en la posición indicada, actualizando el vector.
2	Declaraciones: pos, dim: Entero+ $(0 \leq pos \leq dim)$ el: TipoDeDato $\cup$ {TipoEleNoDefinido}	
3	Semántica: leeElemento(creaVector(dim, TipoDeDato), pos) = TipoEleNoDefinido leeElemento(escribeElemento(creaVector(dim, TipoDeDato), pos, el), pos) = el leeElemento(escribeElemento(Vector[dim], pos, el), pos) = el	

Declaración de un vector en C/C++

tipoDeDato nombreDelVector[númeroDeElementos];

- Ejemplos:

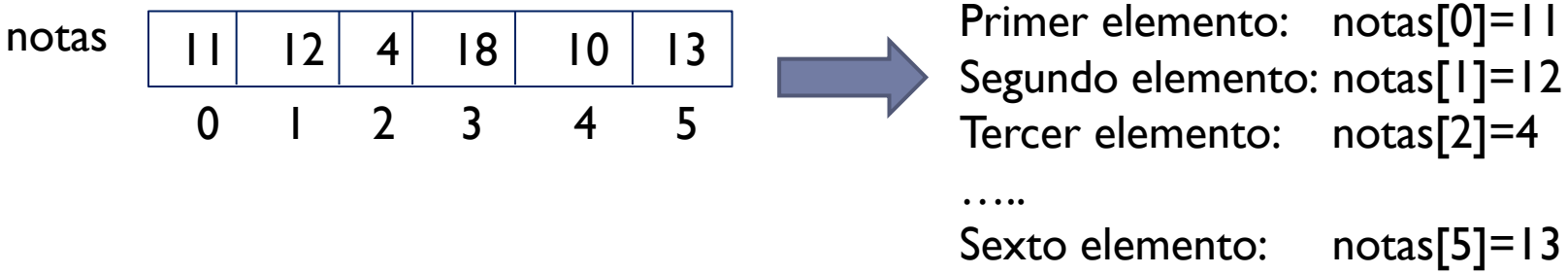
```
float   notas[6];        // Vector notas de 6 números reales
char    cdn[10];         // Vector cedulas de 10 caracteres
int     edades[100];     // Vector edades de 100 números enteros
int     x[12], y[30], z[90];     // Tres vectores de enteros
```

Codificación C/C++

```
#define TAM 10
#define  NUMHORAS  24

int     dia[NUMHORAS], P[TAM];
```

Acceso a los elementos de un vector



Cada elemento de un vector puede usarse como una variable cualquiera

Ejemplos:

```
Promedio=(notas[0]+ notas[1]+ notas[2]+ notas[3]+ notas[4]+ notas[5])/6;
printf("valor= %i", notas[3]);
cin>>notas[4];
cout<<" Nota promedio= "<<Promedio<<endl;
```

Iniciación de los elementos de un vector

Todos los elementos del vector tienen asignado valores iniciales

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

No todos los elementos tienen valor inicial

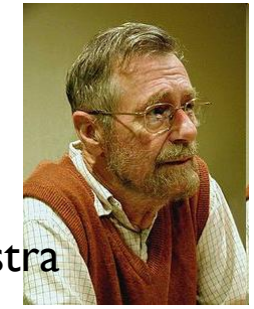
UNIVERSIDAD DE LOS ANDES

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 

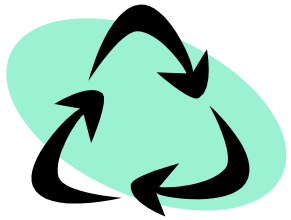
Dijkstra 

Hoare 

UNIVERSIDAD DE LOS ANDES

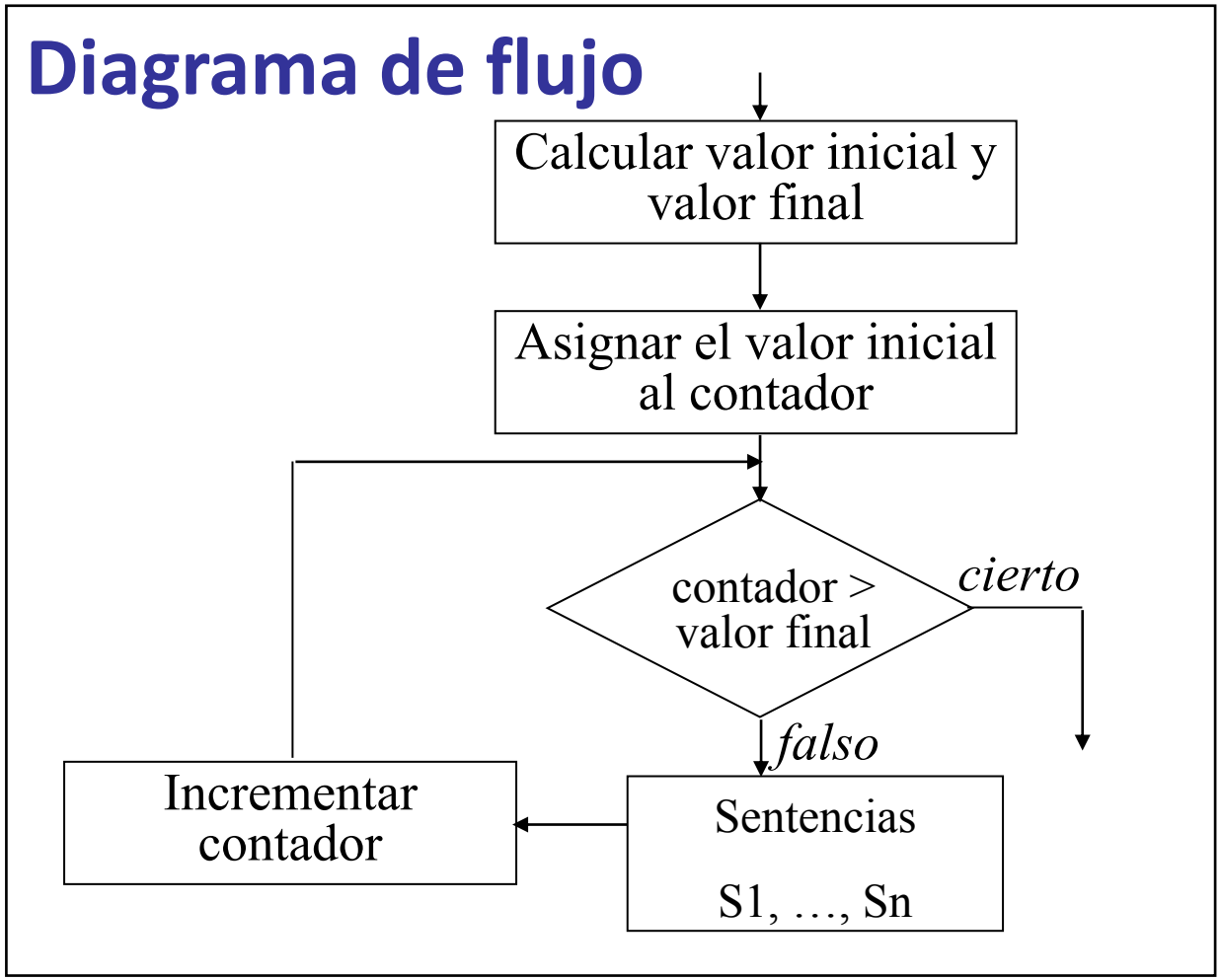
Estructuras de repetición

- 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 para

Diagrama de flujo



REPITA PARA contador DESDE valor inicial HASTA valor final CON INCREMENTO o DECREMENTO dado

UNIVERSIDAD DE LOS ANDES

Repita para

- Se utiliza cuando se conoce con anterioridad el número de veces que el lazo se va a repetir
- Requiere:
  - **Nombre** del contador del lazo
  - El **valor inicial** del contador del lazo (exp1 o Vi)
  - El **incremento o decremento** con el cual, cada vez que se termine una repetición, el contador del lazo será modificado (exp3 o inc)
  - La **condición** que compruebe la existencia del valor final del contador del lazo (exp2 o Vf)

$$[X1,...,Xn = E1,...,En]c = Vi, Vf, inc$$

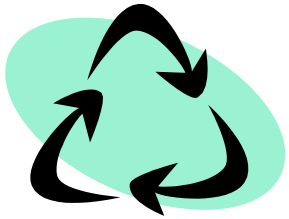
REPITA PARA c DESDE Vi HASTA Vf CON INCREMENTO inc

UNIVERSIDAD DE LOS ANDES

Repita para

Pseudocódigo en español

Repita para (exp1; exp2; exp3)
    S1
    ….
    Sn
Fin_rp



Código en C/C++

```
for(exp1; exp2; exp3)
{
        S1
        ....
        Sn
}
```

REPITA PARA c DESDE Vi HASTA Vf CON INCREMENTO inc

[X1,...,Xn = E1,...,En]c = Vi,Vf, inc

Nombre de la variable de control

Valor final de la variable de control

for (c = 1; c <=10; c = c + 1)

Valor inicial de la variable de control

Incremento de la variable de control

UNIVERSIDAD DE LOS ANDES

Repita para

Pseudocódigo	Código en C/C++

[Desplegar i] i =1, 9,1	for (i = 1; i < 10; i++)
	  printf("%d\n", i);
	// i es la variable de control o contador

[Desplegar "Numero???"
  numero= valor suministrado
]i =0, n, 1
	for (i=0; i < n-1; i++)
	{ **// i es la variable de control**
	  printf("numero???");
	  scanf("%d", &numero);
	  i = i + 2; NOOOOOO
	}

[Desplegar "X[", j, "]= ", X[j]] j = 0, 10, 1	for(j = 0; j < 10; i++) cout<<"X["<<j<<"]= "<<X[j] <<endl;

NOTA: Cuando hay mas de una sentencia asociada al *for* se escribe { }

UNIVERSIDAD DE LOS ANDES

Repita hasta

Pseudocódigo	Corrida en frío
[Desplegar i] i =1, 3,1 for(i=1; i < 4; i=i+1) cout << i << endl;	i = 1 **(1 < 4) = cierto** **1** i = 2 **(2 < 4) = cierto** **2** i = 3 **(3 < 4) = cierto** **3** i = 4 **(4 < 4) = falso**

UNIVERSIDAD DE LOS ANDES

Repita para en C/C++

- for(i=1; i < 4; i++)
- for (; ;) // LAZO INFINITO
- for((i = 0; i < 1000; i++) // Variar el contador (i) de 0 a
  // 999 en incrementos de 1

- for (conta = 1; conta <= 10; conta ++) // Variar el contador
  // (conta) de 1 a 10
  // en incrementos de 1

- for (i = 100; i >= 1; i--) // Variar el contador (i) de 100 a 1
  // en decrementos de 1

UNIVERSIDAD DE LOS ANDES

Ejemplos

- `for (I = 7; I <= 77; I +=7)` `// Variar el contador (I) de 7 a 77 en incrementos de 7`
- `for (I = 20; I >= 2; I -=2)` `// Variar el contador (I) de 20 a 2 en decrementos de –2`
- `for (j = 2; j <= 20; j +=3)` `// Variar el contador (j) a lo largo de la siguiente secuencia de valores: 2, 5, 8, 11, 14, 17, 20`
- `for (j = 99; j >= 0; j -=11)` `// Variar el contador (j) a lo largo de la siguiente secuencia de valores: 99, 88, 77, 66, 55, 44, 33, 22, 11, 0`

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 entre 2 y 100 acumular su valor en el acumulador de pares (ap)			
Salida	Mensaje indicando ap	Acumulador de pares	Entero	>0

UNIVERSIDAD DE LOS ANDES

Ejercicio resuelto 1. Diseño

	sumaPares	
	{pre: }	{pos: AP > 0}
1 2 3	AP=0 [AP = AP + num] num = 2, 100, 2 Desplegar "Suma de los números pares menores que 101=", AP	•**AP: Entero+. Acumulador de números pares** •**num: Entero+. Contador de números pares**
1	-> Suma de los números pares menores que 101 = 2550	Caso exitoso

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	
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;

   for (num = 2; num <= 100; num += 2)       AP += num;
   printf("Suma de numeros pares = %d\n", AP);
   return 0;
}
```

UNIVERSIDAD DE LOS ANDES

Resumen

Tipo nombreDelArreglo[tamañoMáximo]= {listaValoresIniciales};

nombreDelArreglo[posiciónDelElemento]

nombreDelArreglo[posiciónDelElemento]=valor

REPITA PARA c DESDE Vi HASTA Vf CON INCREMENTO inc

[X1,...,Xn = E1,...,En]c =Vi,Vf, inc

Código en C/C++

```
for(exp1; exp2; exp3)
{
        S1
        ....
        Sn
}
```

UNIVERSIDAD DE LOS ANDES

Ejercicio resuelto 2

- Enunciado del problema

  Dadas las notas de n estudiantes de las materias de Programación 1, Física1 y Cálculo 10. Calcular el promedio de notas de cada estudiante

- Análisis E-P-S

Entrada	Variable	Descripción	Tipo de dato	Rango válido
	n	Numero de estudiantes	Entero+	≥0
	notaPR1[n]	Nota de Programación 1	VectorDeReal	[0, 20]
	notaFI1[n]	Nota de Física1	VectorDeReal	[0, 20]
	notaCA1[n]	Nota de Cálculo 1	VectorDeReal	[0, 20]
Proceso	Para cada estudiante hacer: Calcular promedio[i] = $\frac{notaPR1[i]+notaFI1[i]+notaCA1[i]}{3}$			
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

promedioPR1FI1CA1

{pre: n ∈ ℵ} {pos: n∈ ℵ, ∀i, notaPR1$_i$, notaFI1$_i$, notaCA1$_i$, media$_i$ ∈ℜ }

```
1  Desplegar "Introduzca el número de estudiantes ≥ 1"
2  n = valor suministrado
3  Si ( n ≥ 1)
     [ Desplegar "Para el estudiante ", k
       Desplegar "Introduzca la nota de PR1"
         notaPR1[k] = valor suministrado
       Si (0≤ notaPR1[k]≤ 20)
          Desplegar "Introduzca la nota de Fisica1"
          notaFI1[k] = valor suministrado
          Si (0≤ notaFI1≤ 20)
             Desplegar "Introduzca la nota de Cálculo 1"
             notaCA1[k] = valor suministrado
             Si (0≤ notaCA1≤ 20)
                Desplegar "Promedio de notas del estudiante ", k, " = ", media[k]=
(notaPR1[k]+notaFI1[k]+notaCA1[k])/3
              sino
                Desplegar "Nota fuera de rango"
             fsi
           sino
             Desplegar "Nota fuera de rango"
           fsi
       sino
```

- **n:** Natural. Número total de estudiantes de la asignatura.
- **k:** Natural. Contador de estudiantes.
- notaPR1, notaFI1, notaCA1: VectorDeReal[n]. Notas de las materias Programación 1, Física1 y Cálculo 1 de cada estudiante. Cada nota debe estar en el rango de 0 a 20.
- media: VectorDeReal[n]. Promedio de las notas del estudiante, debe estar entre 0 y 20.

Ejercicio resuelto 2. Diseño (cont)

promedioPR1Esto1CA40 {pre: n ∈ ℵ} {pos: n∈ ℵ, ∀i, notaPR1$_i$, notaEsto1$_i$, notaCA40$_i$, media$_i$ ∈ℜ }		
	Desplegar "Nota fuera de rango" fsi] k = 1, n, 1 sino Desplegar "El numero de estudiantes debe ser mayor que 0" fsi	
1	n = 1, notaPR1[1]=12, notaFI1[1]=10, notaCA1[1]=11 => Promedio de notas del estudiante 1=11	Caso límite
2	n = 2, notaPR1[1]=11, notaFI1[1]=11, notaCA1[1]=11 => Promedio de notas del estudiante 1=11 notaPR1[2]=1, notaFI1[2]=1, notaCA1[2]=1 => Promedio de notas del estudiante 2=1	Caso exitoso
3	n=0 => El numero de estudiantes debe ser mayor que cero	No hay estudiantes
4	n=1, notaPR1[1]=34 => Nota fuera de rango	Nota de PR1 inválida
5	n=1, notaPR1[1]=19, notaFI1[1]=56 => Nota fuera de rango	Nota de FI1 inválida
6	n=1, notaPR1[1]=13, notaFI1[1]=1, notaCA1[1]=-5=> Nota fuera de rango	Nota de CA1 inválida

UNIVERSIDAD DE LOS ANDES

Codificación en C++

```
#include <iostream>
using namespace std;
#define ME 50
int    main ( )
{    unsigned int n;
     float      notaPR1[ME], notaFI1[ME], notaCA1[ME], media[ME];
     cout << "Introduzca el numero de estudiantes, entero mayor que 0 y menor que \n"<< ME;
     cin >> n;
     if (n >= 1)
       for(int k=1; k<=n; k++)     // OJO: declaración de k como Entero
       {  cout << "Introduzca la nota de Programacion 1 para el estudiante"<<k << endl;
          cin >> notaPR1[k];
          if(notaPR1[k] < 20.0 && notaPR1[k] > 0.0)
             cout << "Introduzca la nota de Fisica1" << endl;
             cin >> notaFI1[k];
             if(notaFI1[k] < 20.0 && notaFI1 [k] > 20.0)
                                cout << "Introduzca la nota de Calculo 1" << endl;
                cin >> notaCA1[k];
                if(notaCA1[k] < 20.0 && notaCA1[k] > 20.0)
                  media[k]=(notaPR1[k]+notaFI1[k]+notaCA1 [k])/3.0;
```

UNIVERSIDAD DE LOS ANDES

Ejercicio resuelto 2

```
            cout << "Promedio del estudiante " << k << "= " << media[k] << endl;
          else
            cout << "Nota fuera de rango" << endl;
        else
          cout << "Nota fuera de rango" << endl;
      else
        cout << "Nota fuera de rango" << endl;
    }
   else
     cout << "El numero de estudiantes debe ser mayor que cero" << endl;
  return 0;
}
```

Proponga otros diseños junto con sus implementaciones al mismo problema

- ¿Qué es un arreglo y sus tipos?
- ¿Cómo es el TAD Vector?
- ¿Cómo se declaran e inician los vectores?
- ¿Qué es una estructura de repetición?
- Cuáles son las estructuras de repetición?
- ¿Cómo se define una estructura de repetición controlada por un contador?
- ¿Cómo se utiliza un acumulador y un centinela en un repita para?
- ¿Cómo se prueba en frío un programa con repita para?
- ¿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. Hallar el resultado de la siguiente sucesión:

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{N}$$

   hasta que 1/N converja hacia e/100 (o sea igual o menor que un número e cualquiera dividido entre 100)

2. Calcular la sumatoria de los 100 primeros números naturales
3. Calcular independientemente la suma de los números pares e impares comprendidos entre 1 y n

UNIVERSIDAD DE LOS ANDES

Ejercicios

4. Dada una lista de números suministrados por teclado, determinar cual de ellos es el menor y cual es el mayor
5. Dadas las notas de n estudiantes correspondientes al segundo examen de PR1 en el rango de 0 a 20. Calcular el número de estudiantes sobresalientes (16-20), el número de estudiantes satisfactorios (10-15) y el número de estudiantes no satisfactorios (0-9)
6. Calcular el máximo común divisor de n pares de números (x1, y1), (x2, y2), …, (xn, yn) mediante el algoritmo de Euclides:

Sean los números A y B. El método para hallar el máximo común divisor (mcd) de dos números A y B por el método de Euclides es:

- Dividir el número mayor por el menor. Si el residuo de la división es 0, el número menor es el mcd
- Si la división no es exacta, se divide el número menor por el residuo de la división anterior
- Se siguen los pasos anteriores hasta obtener un resto cero. El último divisor es el mcd buscado

UNIVERSIDAD DE LOS ANDES

Ejercicios

7. Leer valores que representan años e indique si son o no años bisiestos. El programa seguirá leyendo años hasta un máximo de 10 o hasta que haya leído 3 años bisiestos

   Recuerde, una vez más, la regla: "Un año es bisiesto si es divisible por 400, o bien si es divisible por 4 pero no por 100"

   Por ejemplo, el año 2000 es bisiesto (es divisible por 400), el año 1992 es bisiesto (es divisible por 4 y no por 100), y el año 2100 no es bisiesto (es divisible por 4 y también por 100)

8. Leer fechas con el formato dd, mm, aaaa indicando si el día es válido con respecto al mes hasta que ya no se desee verificar mas fechas
9. Ordenar en forma ascendente (de menor a mayor) un conjunto de n números reales, desplegando el conjunto desordenado y el mismo conjunto ordenado