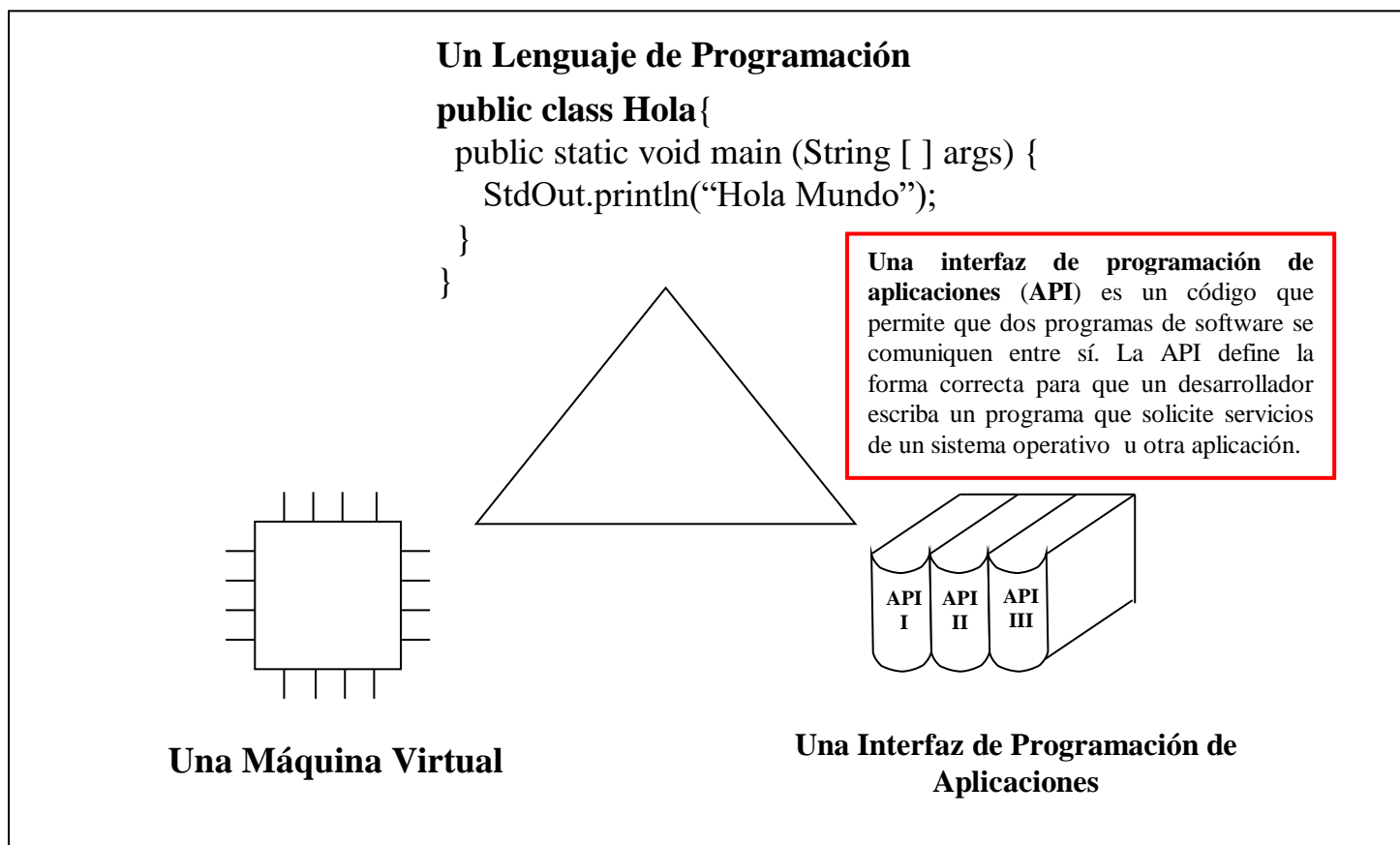


CAPÍTULO 0: JAVA Y PROGRAMAS QUE UTILIZAN ARREGLOS

0.1. Introducción al Java

Plataforma JAVA



JDK: Java™ Platform, Standard Edition Development Kit (JDK™)

- Es un ambiente de desarrollo para construir aplicaciones, applets y componentes usando el lenguaje de programación Java.
- Incluye herramientas útiles para desarrollar y testear programas escritos en Java y correrlos en la plataforma Java
- Enlace para descargar el JDK:

<https://www.java.com/es/download/faq/develop.xml>

JRE (Java Runtime Environment)

- Es lo que se obtiene al descargar el software de Java.
- JRE está formado por Java Virtual Machine (JVM), clases del núcleo de la plataforma Java y bibliotecas de la plataforma Java de soporte.
- JRE es la parte de tiempo de ejecución del software de Java, que es todo lo que necesita para ejecutarlo en el explorador web.
- Java Virtual Machine es sólo un aspecto del software de Java que interviene en interacción web. Java Virtual Machine está incorporado en la descarga del software de Java y ayuda a ejecutar las aplicaciones Java.

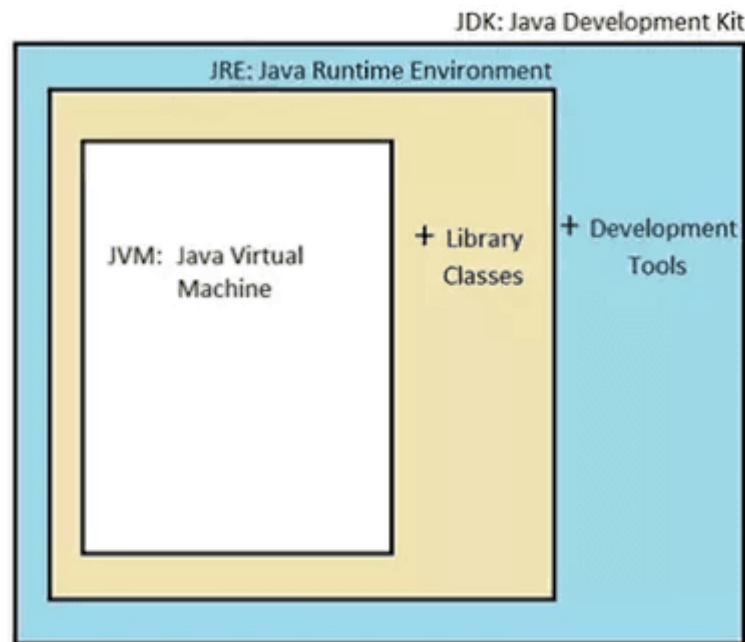
Diferencia entre JDK y JRE

- JDK es el kit de desarrollo de Java que sirve para construir programas usando el lenguaje de programación Java. Trae herramientas útiles como el compilador (javac), el desensamblador de binarios (javap), debugger, entre otras herramientas.
- La instalación de JDK ya contiene un JRE dentro de las carpetas.
- JRE es el entorno de ejecución de Java. Contiene a la JVM y otras herramientas que permiten la ejecución de las aplicaciones Java.
- JRE **no posee** compiladores ni herramientas para desarrollar las aplicaciones Java, solo posee las herramientas para ejecutarlas.
- Instalas el JDK cuando quieres desarrollar.
- Instalas solamente el JRE en los equipos donde solo vas a ejecutar aplicaciones Java.

Máquina virtual (JVM)

- Es un conjunto de programas de software que permiten la ejecución de instrucciones y que normalmente están escritos en byte code Java.
- Las máquinas virtuales de Java están disponibles para las plataformas de hardware y software de uso más frecuente.

- Una de las principales características de Java es que una vez nosotros codificamos un programa este puede ser ejecutado N cantidad de veces en diferentes plataformas. Esto es posible gracias a la máquina virtual de Java. Existe una *JVM* para cada entorno, para Windows, Linux, Mac, Android etc.



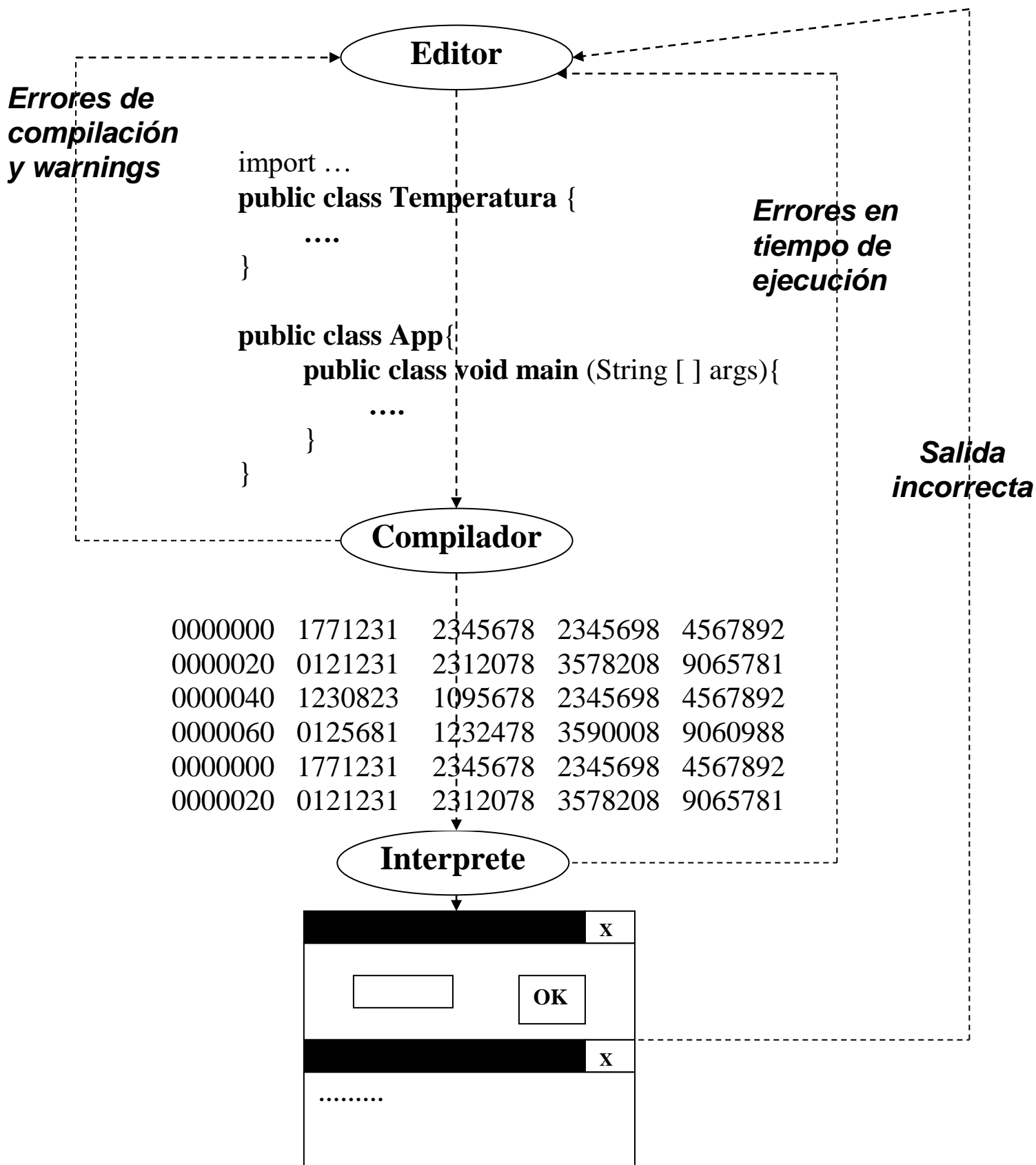
$JDK = JRE + Development\ Tools$

$JRE = JVM + Library\ Classes$

Compilación de un programa Java

- El proceso de compilación de un programa.java no entrega código que la máquina pueda comprender (unos y ceros), en cambio, produce archivos con extensión **.class**, estos archivos se componen de *Bytecode*.
- *Bytecode* es un conjunto de instrucciones altamente optimizadas, que se encuentran diseñadas para ser ejecutadas por la máquina virtual de Java.
- Es en esencia, la *JVM* es quien interpreta y ejecuta el código *Bytecode*, de tal manera que el dispositivo ya pueda realizar las tareas previamente escritas.

Ciclo de un programa Java



Documentación

Comentarios en Java

`/*`

Este es un comentario de bloque,
Puede tener más de 1 línea de largo.

`*/`

`//` Este es un comentario de EXACTAMENTE 1 línea.

JavaDoc

- Generador de documentación
- Herramienta que viene en el JDK
- Se usa cuando se prepara automáticamente la documentación generada.
- Javadoc es el estándar para documentar clases de Java.
- La mayoría de los IDEs utilizan javadoc para generar de forma automática documentación de clases.
- Documentación del código en formato HTML
- Los comentarios para las clases deben ir encerrados en `/**` `*/`.
- Un comentario en JavaDoc corresponde en una descripción general opcional seguido de opcionales secciones etiquetadas.
- Las etiquetas comunes en formato `@tag description` son las siguientes:
 - ☐ `@author name` (clase/interfaz)
 - ☐ `@version major.minor.patch` (clase/interfaz)
 - ☐ `@param name description` (método)
 - ☐ `@return description` (método)

Tipos de Datos Básicos

Java tiene 8 tipos primitivos:

- Enteros:

byte
short
int
long

}

Difieren en la magnitud
- Números de Punto flotante:

float
double

}

Difieren en la magnitud
- Caracteres: char
- Booleano: boolean

• Números Enteros

Tipo de Dato	Espacio en Memoria	Valor Mínimo	Valor Máximo
int	32 bits	-2147483648	2147483647
long	64 bits	-9,22337E+18	9,22337E+18

3	-8	4
-9	-16	3
4	-5	-8

• Números Reales

Tipo de Dato	Espacio en Memoria	Valor Mínimo (Valor Absoluto)	Valor Máximo (Valor Absoluto)	Dígitos Significativos
float	32 bits	1,40E-45	3,40E+38	6
double	64 bits	4,9E-324	1,8E+308	15



- **Booleano**

Tipo de Dato	Espacio en Memoria	Valores
boolean	8 bits	false(0), true(1)

- **Carácter**

Tipo de Dato	Espacio en Memoria	Descripción
char	16 bits	Un sólo Carácter

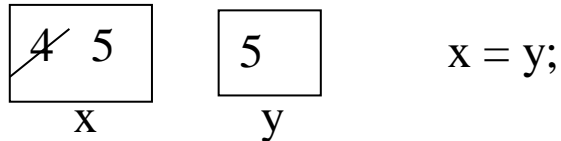
- **Cadena de caracteres**

Tipo de Dato	Espacio en Memoria	Descripción
String	Variable	Conjunto de caracteres

Variables

Las variables de los tipos primitivo almacenan **valores**.

Ejemplo: x, y son enteros



Concepto de Casting

¿Desde qué punto de vista quiero ver: double o int?

int i;

double d;

- **Quiero verlo como entero:**

➤ ~~i = d;~~ No se puede perder información. Por lo tanto:

➤ ✓✓ i = (int) d; // **Se quiere ver como entero**

- **Quiero verlo como double:**

➤ ✓✓ d = i; // **No hay pérdida de información**

➤ ✓✓ d = (double) i;

Ejemplo

```
import ucn.StdIn;  
import ucn.StdOut;  
  
public class Promedio{  
  
    public static void main(String args[]){  
        StdOut.print("Ingrese nota 1:");  
        double N1 = stdIn.readDouble();  
  
        StdOut.print ("Ingrese nota 2:");  
        double N2 = stdIn.readDouble();  
  
        StdOut ("Ingrese nota 3:");  
        double N3 = stdIn.readDouble();  
  
        double NF = (N1 + N2 + N3) / 3;  
        StdOut.println("Nota final: " + NF);  
    } //Fin main  
  
} // Fin class Promedio
```

Operadores Relacionales

Operador	Uso	Devuelve verdadero si:
$>$	$v1 > v2$	<i>v1 es mayor que v2</i>
\geq	$v1 \geq v2$	<i>v1 es mayor o igual que v2</i>
$<$	$v1 < v2$	<i>v1 es menor que v2</i>
\leq	$v1 \leq v2$	<i>v1 es menor o igual que v2</i>
$==$	$v1 == v2$	<i>v1 y v2 son iguales</i>
$!=$	$v1 != v2$	<i>v1 y v2 son distintos</i>

Operadores Lógicos

Operador	Uso	Devuelve verdadero si...
&& (AND)	$v1 \&\& v2$	v1 y v2 son ambos verdaderos
 (OR)	$v1 v2$	v1 o v2 son verdaderos
! (NOT)	$!v$	v es falso

```
import ucn.Stdout;
import ucn.StdIn;

public class App{

    public static void main(String args[]){
        StdOut.print("Ingrese la primera nota: ");
        double nota1 = StdIn.readDouble();

        StdOut.print("Ingrese la segunda nota: ");
        double nota2 = StdIn.readDouble();

        StdOut.print("Ingrese la tercera nota: ");
        double nota3 = StdIn.readDouble();

        double notaFinal =(nota1 + nota2 + nota3) /3;

        if (notaFinal >= 3.95) {
            StdOut.println("Aprobo, nota = "+notaFinal);
        }
        else {
            StdOut.println("Reprobo, nota ="+notaFinal);
        }
    } //Fin main
} //Fin App
```

Diferencia entre el if al mismo nivel y el if anidado

Suponga que tiene que leer un dato entero y hacer algo diferente, dependiendo del valor del dato

```
....
if (dato == 1){
    StdOut.println("1");
}
if (dato == 2){
    StdOut.println("2");
}
if (dato == 3){
    StdOut.println("3");
}
else {
    StdOut.println("Ni 1,ni 2,ni 3");
}
```

≠

```
....
if (dato == 1){
    StdOut.println("1");
}
else {
    if (dato == 2){
        StdOut.println("2");
    }
    else {
        if (dato == 3){
            StdOut.println("3");
        }
        else {
            StdOut.println("Ni 1,
                               ni 2, ni 3");
        }
    }
}
```

¿Cómo funciona si dato es 1?

- Entra al primer if (si dato = 1). Como es si, imprime el 1 en la pantalla
- Entra al segundo if (si dato = 2). No se cumple la condición. No hace nada
- Entra el tercer if (si dato = 3). No se cumple la condición. Se va por el else e imprime: Ni 1, ni 2, ni 3

Impresión

1
Ni 1, ni 2, ni 3

¿Cómo funciona si dato es 1?

Entra al primer if (si dato = 1). Como es si, imprime el 1 en la pantalla y se va al final, ya que después viene el else, por lo tanto no entra ahí.

Impresión

1

if anidado versus switch

```
....  
if (dato == 1){  
    StdOut.println("1");  
}  
else {  
    if (dato == 2){  
        StdOut.println("2");  
    }  
    else {  
        if (dato == 3){  
            StdOut.println("3");  
        }  
        else {  
            StdOut.println("Ni 1,  
                             ni 2, ni 3");  
        }  
    }  
}
```

=

```
....  
switch (dato) {  
    case 1:  
        StdOut.println("1");  
        break;  
  
    case 2:  
        StdOut.println("2");  
        break;  
  
    case 3:  
        StdOut.println("3");  
        break;  
  
    default:  
        StdOut.println("Ni 1, ni 2,  
                        ni 3");  
}
```

Ejemplo: Problema del Fizz-Buzz test

Escriba un programa Java que imprima los números del 1 al 100, pero para los múltiplos de 3 debe imprimir “Fizz” en lugar del número, para los múltiplos de 5 imprimir “Buzz” en vez del número y para los números que son múltiplos de ambos (de 3 y 5) se debe imprimir “FizzBuzz” en vez del número

```
....  
public class App{  
    public static void main(String args[]){  
        for (int i = 1; i <= 100; i++){  
            if ((i%3 == 0) && (i %5 ==0)){  
                StdOut.println("FizzBuzz");  
            }  
            else{  
                if (i%3 == 0) {  
                    StdOut.println("Fizz");  
                }  
                else{  
                    if (i % 5 == 0) {  
                        StdOut.println("Buzz");  
                    }  
                    else{  
                        StdOut.println(i);  
                    }  
                }  
            }  
        }  
    }  
}
```

Ejercicio:

Cálculo del promedio de edad. Se leen las edades, hasta que se lee un -1, que implica el fin de datos

....

```
public class App{
    public static void main(String args[]){
        int contPersonas = 0;
        int sumEdad = 0;
        StdOut.print("Ingrese edad: ");
        int edad = StdIn.readInt();
        while(edad != -1) {
            sumEdad = sumEdad + edad;
            contPersonas++;
            StdOut.print("Ingrese edad: ");
            edad = StdIn.readInt();
        }
        if(contPersonas>0) {
            int promEdad = sumEdad/contPersonas;
            StdOut.println("Promedad:" + promEdad);
        }
        else{
            StdOut.println("No se leyeron edades");
        }
    }//Fin main
}//Fin App
```

Ejercicio

Determinar al mejor alumno (con la mejor nota), en 1000 estudiantes. Por cada alumno se debe leer su matrícula y su promedio de notas. Se debe desplegar el número de matrícula y la nota del mejor alumno.

```
import ucn.StdIn;
import ucn.Stdout;

public class MejorAlumno {
    public static void main(String args[]){
        double mayor = 0;
        String matrMayor = "";
        for(int i=1; i<=1000;i++){
            //Lectura de los datos
            StdOut.print("Ingrese matricula alumno: ");
            String mat = StdIn.readString();
            StdOut.print("Ingrese promedio alumno: ");
            double prom = StdIn.readDouble();
            if (prom > mayor) {
                mayor = prom;
                matrMayor = mat;
            }
        }
        StdOut.println("Mayor promedio es: " + mayor +
            " numero de matricula: " + matrMayor);
    }
}
```


Ejercicio

Leer registros de alumnos (matrícula, promedio) y determine los 2 números de matrículas correspondientes a los alumnos de mejores promedios. Fin de datos: matrícula = -1.

```
...
public class DosMejores{
    public static void main(String args[]){
        double mayor1 = 0;
        String mat1 = " ";
        double mayor2 = 0;
        String mat2 = " ";
        String mat = StdIn.readString();
        while (!mat.equals("-1 ")){
            double prom = StdIn.readDouble();
            if (prom > mayor1){
                mayor2 = mayor1;
                mat2 = mat1;
                mayor1 = prom;
                mat1 = mat;
            }
            else {
                if (prom > mayor2) {
                    mayor2 = prom;
                    mat2 = mat;
                }
            }
            mat = StdIn.readString();
        } //Fin del while
        StdOut.println("Primer prom: " + mayor1 + " matricula: " + mat1);
        StdOut.println("Segundo prom: " + mayor2 +
                                                                " matricula: " + mat2);
    } //Fin main
} //Fin DosMejores
```

Ejercicio

Genere el término M de la secuencia de Fibonacci. Suponga $M \geq 2$.

$S = 0, 1, 1, 2, 3, 5, 8, 13, \dots$ $M > 3$

```
import ucn.StdIn;
import ucn.Stdout;

public class Fibo{
    public static void main(String args[]){
        int anteriorUno = 0;
        int anteriorDos = 1;
        int nuevo;
        StdOut.print("Ingrese valor de m: ");
        int m = StdIn.readInt();
        for (int i = 3; i <= m; i++) {
            nuevo = anteriorUno + anteriorDos;
            anteriorUno = anteriorDos;
            anteriorDos = nuevo;
        }
        StdOut.println(" El Fibonacci de "+m+ " es "+ nuevo);
    } //Fin main
} //Fin Fibo
```

Ejercicios Propuestos

Problema 1

Para efectos de contabilización y cálculo de las ventas diarias relativas a un producto, se dispone de la siguiente información:

- Número de la factura
- Cantidad de frascos vendidos
- Tipo de frasco 1: Frasco de 3 kgs
2: Frasco de 5 kgs
3: Frasco de 9 kgs
4: Fin de datos

Los precios unitarios de venta son:

- Si el frasco es de tipo 1: \$1.8/kg
- Si el frasco es de tipo 2: \$1.5/kg
- Si el frasco es de tipo 3: \$1.2/kg

Se requiere:

a) Listado de toda la información, indicando por cada factura:

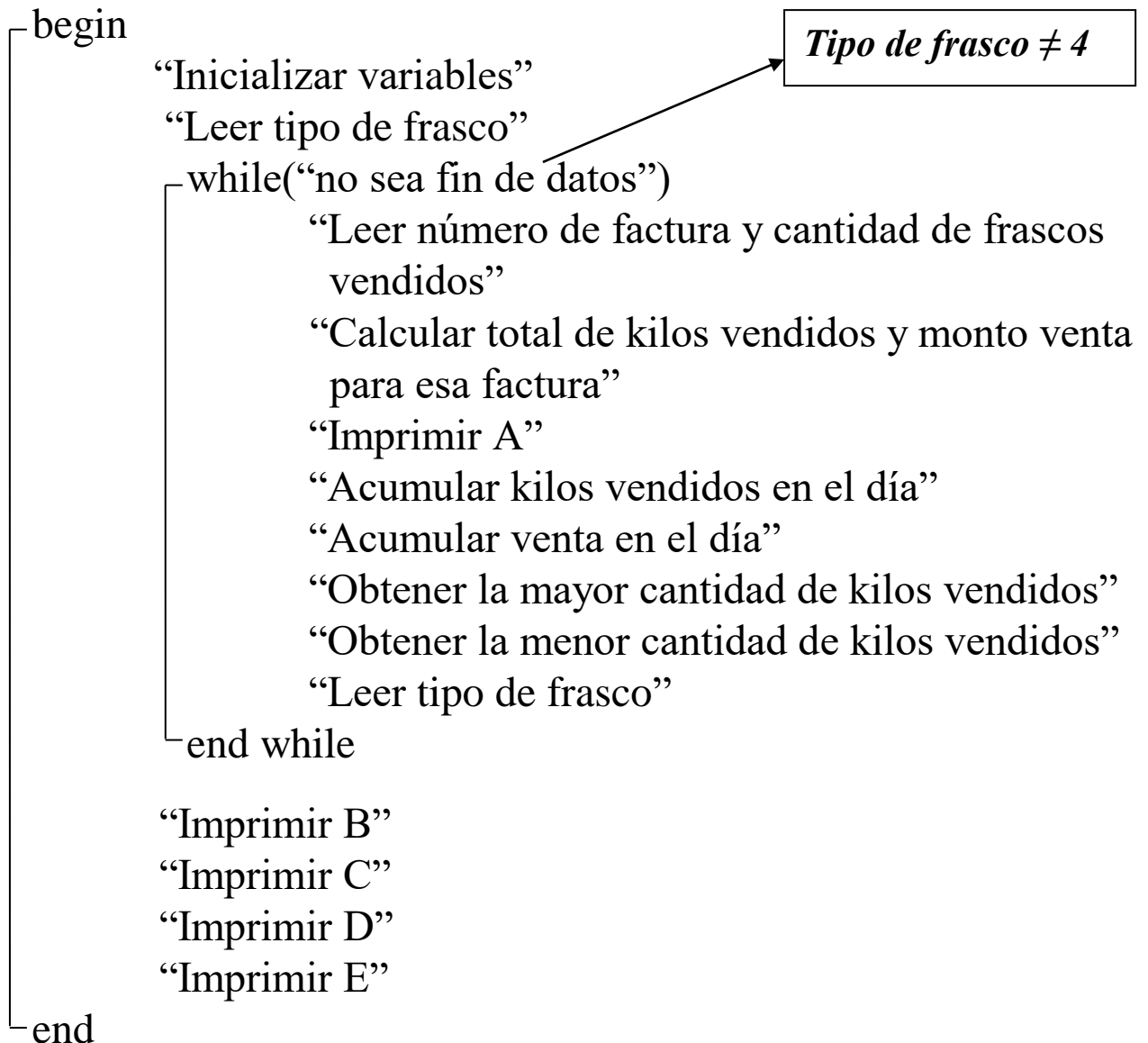
- Número de la factura
- Cantidad de frascos vendidos
- Total de kilos vendidos
- Monto de la venta

b) Cantidad total de kgs vendidos en el día

c) Monto total de ingresos diarios

- d) ¿Cuál fue la cantidad mayor de kgs vendidos y a qué número de factura pertenece?
- e) ¿Cuál fue la cantidad menor de kgs vendidos y a qué número de factura corresponde?

Refino



Problema 2

Se dispone de N conjuntos de datos. El valor de N es el primer valor a leer. El número de datos existente en cada conjunto, es el primer valor leído del conjunto. La configuración es la siguiente:

N	
N_1	
.....	} N_1 datos (conjunto 1)
.....	
.....	
N_2	
.....	} N_2 datos (conjunto 2)
.....	
.....	
N_N	
.....	} N_N datos (conjunto N)
.....	
.....	

En cada registro de datos viene un valor X . Sumar todos los valores de X para cada conjunto, e indicar en qué conjunto se encuentra el valor más alto de esta suma. Imprimir la suma de cada conjunto y el número del conjunto de mayor suma.

Refino

```
begin
    "Inicializar mayor"
    read(N)
    for (int i = 1; i <= N; i++) {
        "Inicializar acumulador en cero"
        read(NI)
        for J = 1 to NI
            "Leer dato"
            "Acumular dato"
        end for
        write(acumulador)
        "Determinar el mayor"
    }
    "Imprimir el conjunto que tiene la mayor suma"
end
```

Problema 3

Calcular e imprimir $S = \prod_{i=1}^N t_{2i-1}$ N debe leerse.

- a) Si se considera su forma recursiva: $t_1 = 5$; $t_i = t_{i-1} + 3$
 $i > 1$

Refino (a)

```
begin
    "Inicializar término"
    "Inicializar S en 1"
    "Leer N"
    for (int i = 1; i <= N; i++) {
        "Actualizar S"
        "Calcular nuevo término"
    }
    "Imprimir S"
end
```

- b) Si se considera su forma posicional: $t_i = 5 + 3(i-1)$

Refino (b)

```
begin
    "Leer N"
    "Inicializar S en 1"
    for (int i = 1; i <= 2*N; i = i+2) {
        "Calcular término"
        "Actualizar S"
    }
    "Imprimir S"
end
```

Problema 4

Calcular e imprimir S donde:

$$S = x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \dots$$

El valor de X se encuentra en un registro. El proceso termina cuando algún término de la sumatoria sea menor o igual a 0,00001.

$$S = \sum_{i=1} t_i \quad t_i = \frac{x^{2i-1}}{2i-1} \quad i \geq 1$$

Refino

```

begin
  "Leer x"
  "Inicializar término"
  "Inicializar S"
  "Inicializar I en 2"
  while (término sea > 0.00001)
    "Acumular término"
    "Calcular término"
    "Incrementar I en 1"
  end while
  "Imprimir S"
end

```


Problema 5: Calcular e imprimir: $A = N1! \times N2! \times N3!$
 Para valores de N1, N2 y N3 que se encuentran en un registro.

Refino versión1

Begin

“Leer N1,N2,N3”
 “Calcular factorial de N1”
 “Calcular factorial de N2”
 “Calcular factorial de N3”
 “Calcular A”
 “Imprimir A”

End

Refino version2

begin

read(N1,N2,N3)

for I = 1 to 3

if (I = 1) then

valor \leftarrow N1

else

if (I = 2) then

aux1 \leftarrow fact

valor \leftarrow N2

else

aux2 \leftarrow fact

valor \leftarrow N3

end if

end if

fact \leftarrow 1

for J = 2 to valor

fact \leftarrow fact * J

end for

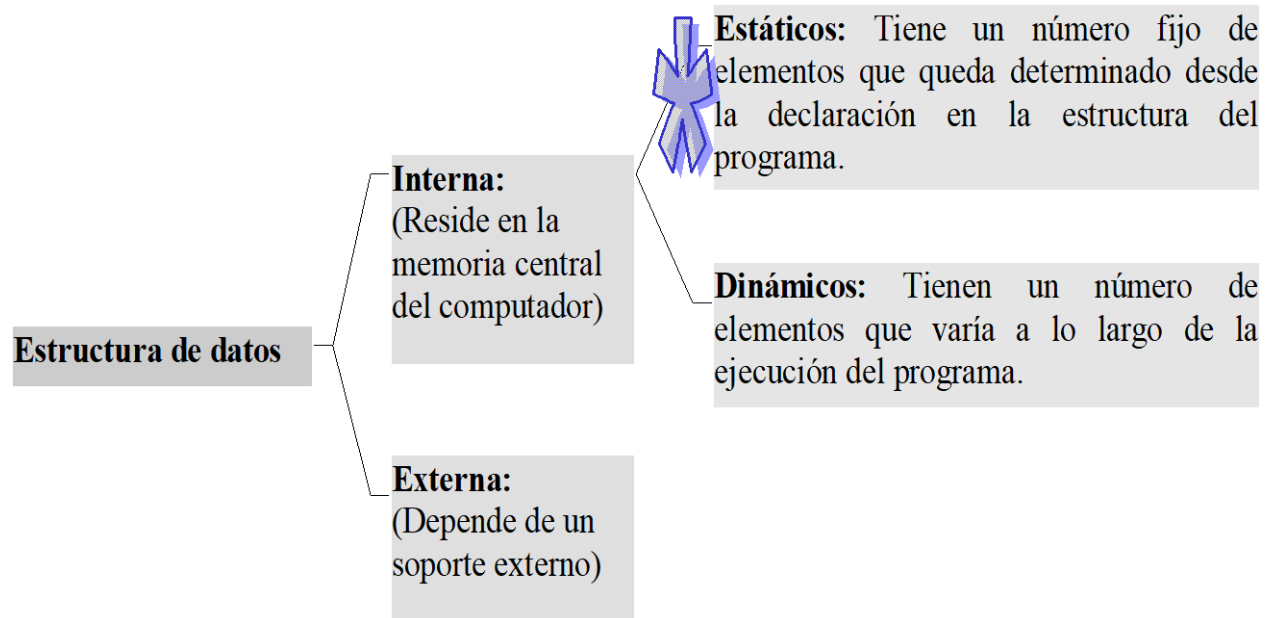
end for

A \leftarrow aux1 * aux2 * fact

write(A)

end

0.2. Concepto de Arreglos

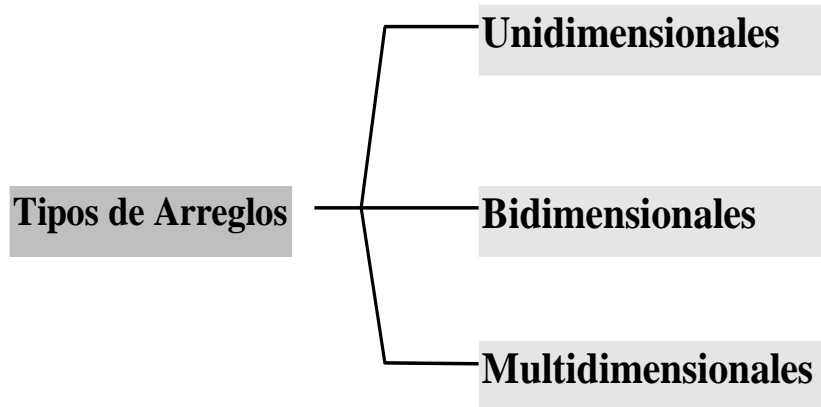


Un **arreglo** consiste en un número fijo, finito y ordenado de elementos, todos del mismo tipo y bajo un nombre común para todos ellos.

Nombre

Valor 1	Valor 2	Valor 3	Valor 4	Valor 5	Valor 6	Valor 7	...	Valor N
↑								↑
Primer elemento								enésimo elemento

Tipos de arreglos:



a) Arreglos unidimensionales (vectores)

- Son arreglos de una dimensión. También se denominan vectores.
- Tiene un solo índice. Cada componente del vector se direcciona mediante un **nombre** seguido del número correspondiente al **índice** entre paréntesis cuadrados.

1	2	3	4	5	6	7	8	9	10

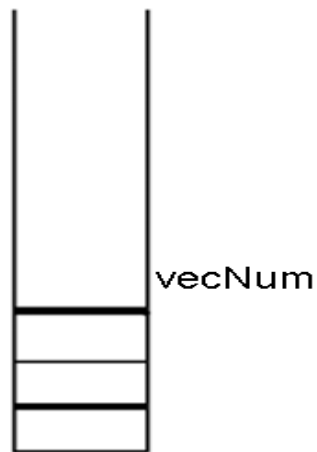
En **Java** el primer elemento está en la **posición 0**.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
1	2	3	4	5	6	7	8	9	10

- Para trabajar con vectores se necesita:
 - Declarar el vector
 - Crear el vector
 - Insertar elementos al vector
 - Obtener elementos del vector
- **Ejemplo:** para trabajar con un vector de números enteros.

- **Declaración del Vector**

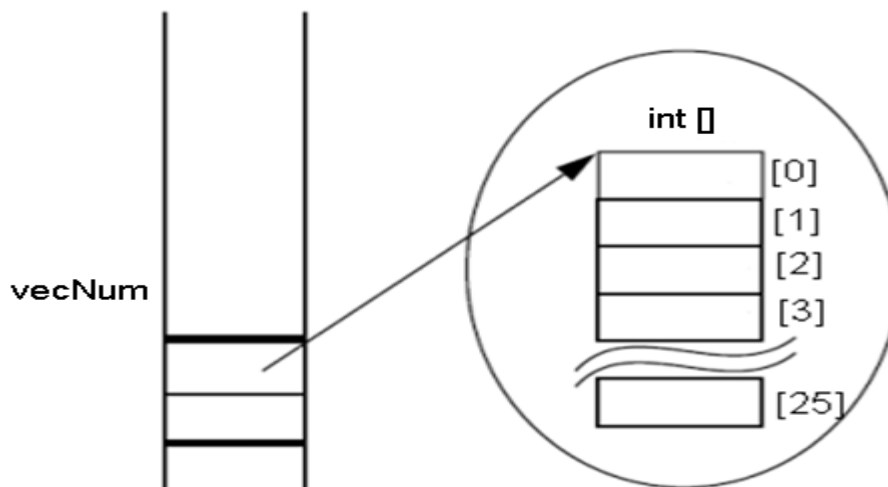
```
int [] vecNum;
```



→ Crea una referencia para utilizar el arreglo.

- **Creación del Vector**

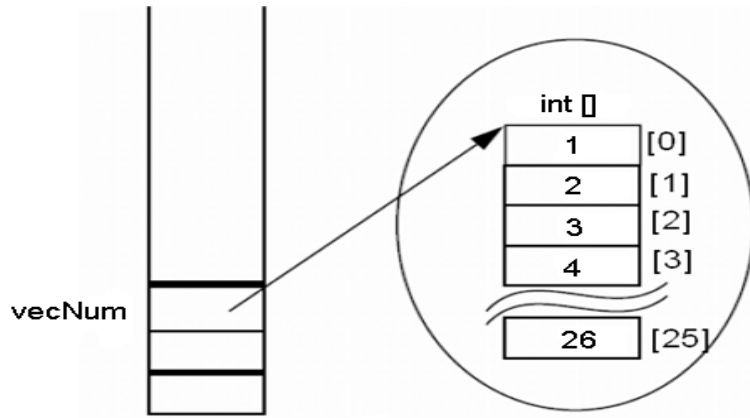
```
vecNum = new int[26];
```



→ Crea un arreglo que puede almacenar 26 elementos de tipo int

➤ **Inserción de elementos en el Vector**

```
for(int i=0; i<26;i++) {
    vecNum[i]=i+1;
}
```



➤ **Obtener elementos**

Desplegar los elementos del vector en la pantalla.

```
for(int i=0; i<26;i++) {
    StdOut.println(vecNum[i]);
}
```

Ejemplo:

Programa que lee la nota final de un alumno para cada una de 10 asignaturas, las almacena en un vector y calcula e imprime su media.

Refino:

PROGRAM NOTA MEDIA

Begin

“Leer notas y dejarlas en un vector de 10 posiciones”

“Sumar las notas”

“Calcular e imprimir el promedio”

end

En Java:

```
import ucn.StdIn;
import ucn.Stdout;
```

```
public class EjemploVector{
public static void main(String args[]){

    //Declaración y creación del vector
    double [] Notas = new double[10];

    //Lectura de las notas e ingreso al vector
    for(int i = 0; i < 10; i++) {
        StdOut.print (“Ingrese nota “ +(i+1));



```
Notas[i] = StdIn.readDouble();
```



```
double nota = StdIn.readDouble();
Notas[i] = nota;
```


        }

    //Calculo del promedio
    double suma = 0;
    for(int i = 0;i<10; i++) {
        suma = suma + Notas[i];
    }

    double prom = suma/10;
    StdOut.println(“Promedio: “ + prom);

} //Fin main
}//Fin EjemploVector
```

b) Arreglos Bidimensionales (Matrices)

- Son las tablas de 2 dimensiones.
- Tienen dos índices, por lo cual cada componente de la matriz se direccionan mediante su nombre seguido de los dos índices, donde cada uno se encuentra entre paréntesis cuadrados.

A		1	2	3	...	N
1						
2						
3						
...						
M						

MxN

Matriz A de M filas y N columnas

En Java, la primera fila es la 0 y la primera columna es la 0.

A		0	1	2	...	N-1
0						
1						
2						
...						
M-1						

MxN

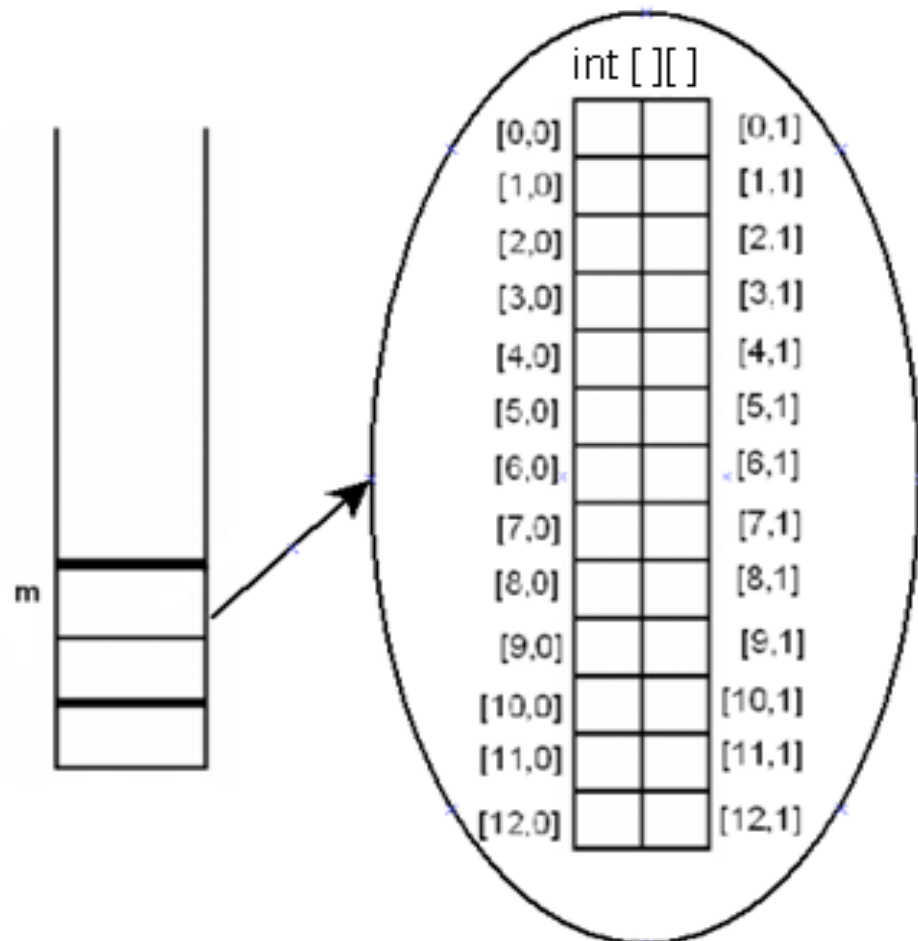
Matriz A de M filas y N columnas

- **Declaración de la matriz**

```
int [][] m;
```

- **Creación de la matriz**

```
m=new int[13][2];
```



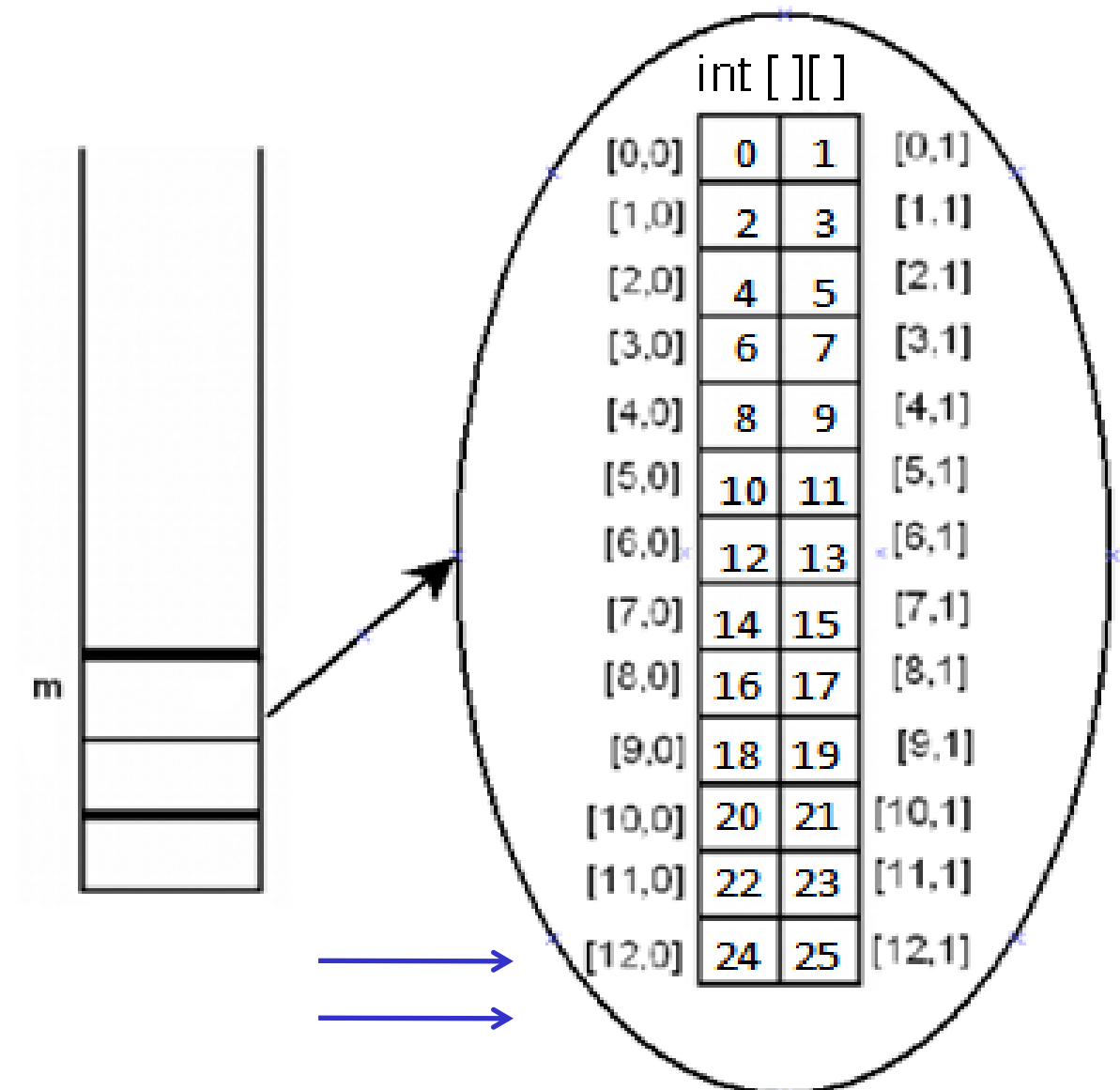
- **Direccinamiento de un elemento**

—————> **A**[fila] [columna]
 Identificador ↑ ↑
 Valor variable o expresión numérica entera

- Ejemplos:

.....

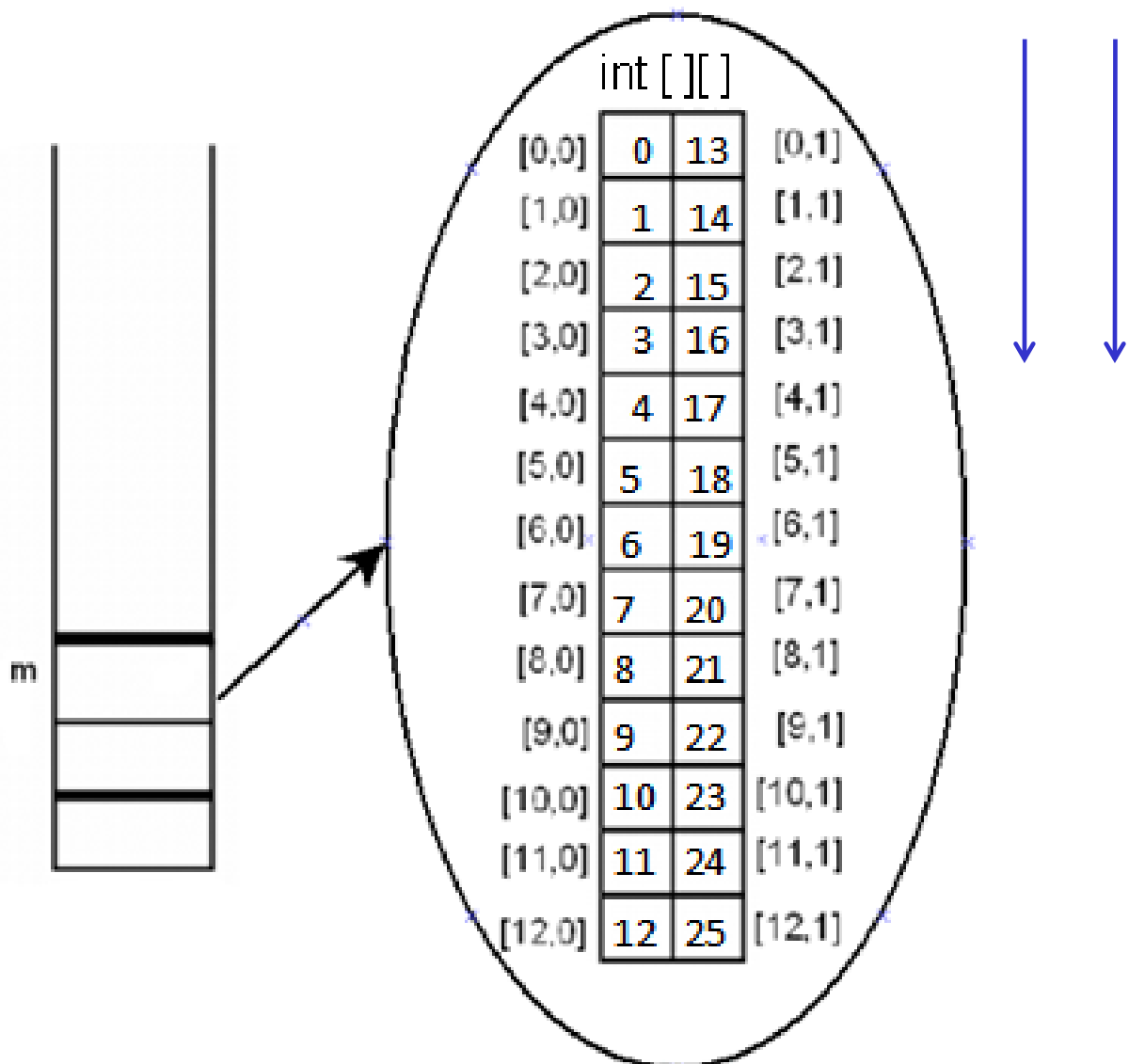
```
int val = 0;
for(int f=0; f<13; f++) {
    for(int c=0; c<2; c++){
        m[f][c]= val;
        val++;
    }
}
```



```

.....
int val = 0;
for(int c=0; c<2; c++) {
  for(int f=0; f<13; f++){
    m[f][c]= val;
    val++;
  }
}

```



Algoritmos de búsqueda

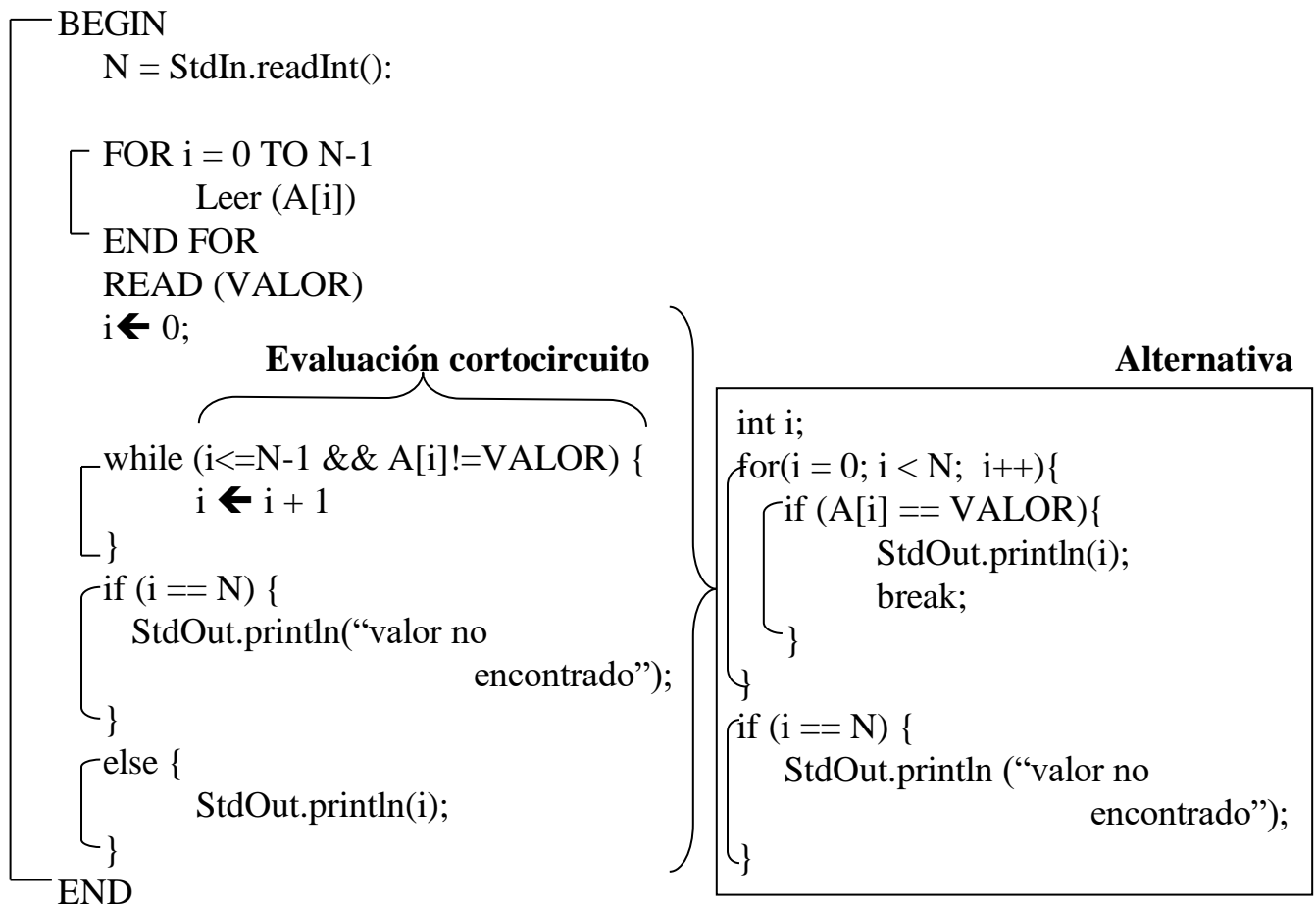
Concepto de búsqueda:

Entrada: A[I], I = 0, N-1 y Valor

Salida: Posición donde se encuentra Valor. Si no está, se envía un mensaje.

- **Datos se encuentran desordenados**

- **Búsqueda Secuencial**



Datos se encuentran ordenados de menor a mayor

□ Búsqueda Secuencial

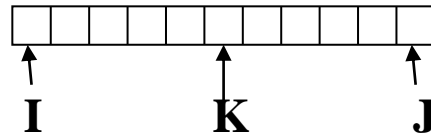
```
BEGIN
  READ (N)
  FOR I = 0 TO N-1
    READ (A[I])
  END FOR
  READ (valor)

  i ← 0;
  while (i <= N-1 && A[i] < valor) {
    i ← i + 1;
  }

  if (i == N || A[i] > valor) {
    StdOut.println ('valor
                     no encontrado');
  }
  else {
    StdOut.println (VALOR, I);
  }
END
```

❑ Búsqueda Binaria

➤ K: Posición de al medio del arreglo



➤ Si Valor = A[K] ==> Búsqueda Positiva

➤ Si Valor > A[K], Calcular $I \leftarrow K + 1$

➤ Si Valor < A[K], Calcular $J \leftarrow K - 1$

```

BEGIN
  READ (N)
  FOR I = 0 TO N-1
    READ (A[I])
  END FOR
  READ (VALOR)
  I ← 0;
  J ← N-1;
  do
    K ← ⌊ (I + J) / 2 ⌋
    IF (VALOR > A[K])
      I ← K + 1
    ELSE
      J ← K - 1
    ENDIF
  while (A[K] != VALOR && I <= J);
  IF (A[K] == VALOR) {
    StdOut.println (VALOR, K)
  }
  ELSE {
    StdOut.println ("VALOR NO ENCONTRADO");
  }
END

```

Problema:

Determinar el ganador de una elección. El programa recibe inicialmente el listado de 50 candidatos y luego 1000 votos (los votos contienen el nombre del candidato).

	candidatos	votos
0		
1		
2		
...		
49		

Refino:**PROGRAM VOTOS**

BEGIN

“Leer a los 50 candidatos y dejarlos en un vector de 50 posiciones”

“Inicializar el contador de votos de cada candidato en 0”

FOR I \leftarrow 1 TO 1000

READ (VOTO)

“Buscar a que candidato corresponde ese voto”

“Incrementar el contador correspondiente”

endfor

“Determinar al candidato con la mayor cantidad de votos”

“Imprimir al candidato ganador”

END

.....

public class Votos{

public static void main(String args[]){

String [] candidatos = new String[50];

int [] votos = new int[50];

for(int i = 1; i <= 50; i++) {

String nombre = StdIn.readString();

candidatos[i-1] = nombre;

votos[i-1] = 0;

}

for (int i = 1 ; i<=1000; i++){

String nombre = StdIn.readString();

int j= 0;

while(j < 50 && !candidatos[j].equals(nombre)){

j = j +1;

}

if (j == 50) {

StdOut.println

("Candidato no existe");

}

else {

votos[j]= votos[j] +1;

}

} //fin for

int mayor = 0;

String mejorCandidato = null;

for(int i = 0; i<= 49; i++){

if (votos[i] > mayor) {

mayor = votos[i];

mejorCandidato = candidatos[i];

}

}

StdOut.println (mejorCandidato + " " + mayor);

} // fin main

}//Fin votos

Alternativa

int i;

for(i = 0; i <= 49; i++){

if (candidatos[j].equals(nombre))

break;

}

if (i <= 49) {

votos [i] ++;

}

else{

StdOut.println

("Candidato no

existe");

Algoritmos de Ordenamiento

Entrada: Arreglo Desordenado

Salida: Arreglo Ordenado de menor a mayor

- **Ordenamiento Simple**

```

BEGIN
    //Lectura del vector
    StdOut.println ("Ingrese valor de N: ");
    int N =StdIn.readInt();
    for (int i=0; i <N; i++) {
        StdOut.print ("Ingrese valor de A["+i+"]: ");
        A[i]= stdIn.readInt();
    }
    //Ordenamiento del vector
    for (int i=0; i <=N-2; i++) {
        for (int j=i+1; j<=N-1; j++) {
            if (A[i] > A[j]) {
                AUX ← A[i];
                A[i] ← A[j];
                A[j] ← AUX;
            }
        }
    }
    //Despliegue del arreglo ordenado
    for (int i=0; i <N; i++) {
        StdOut.println (A[i]);
    }
END
  
```


- **Ordenamiento de Burbujas**

```

BEGIN
    //Lectura del vector
    StdOut.println ("Ingrese valor de N: ");
    int N = stdIn.readInt();
    for (int i=0; i <N; i++) {
        StdOut.print ("Ingrese valor de A["+i+"]: ");
        A[i]= StdIn.readInt();
    }
    //Ordenamiento del vector
    for(int i = 0; i <=N -2; i++){
        for( int L = N -1; L >= i+1; L--) {
            if (A[L] < A[L - 1]) {
                AUX = A[L];
                A[L] = A[L - 1];
                A[L - 1] = AUX;
            }
        }
    }
    //Despliegue del arreglo ordenado
    for (int i=0; i <N; i++) {
        StdOut.println (A[i]);
    }
END

```

Algoritmos de Inserción y Eliminación

• Eliminación

- Eliminar de un arreglo A de N elementos, el valor X.
- El arreglo y X deben ser leídos.
- Imprimir el arreglo resultante.
- El arreglo podría estar ordenado o no.

PROGRAM Eliminar

BEGIN

“Leer arreglo”

“Leer X”

“Buscar la posición donde está X en el arreglo”

IF (X está en el arreglo)

“Corrimiento”

$N \leftarrow N - 1$

“Imprimir arreglo resultante”

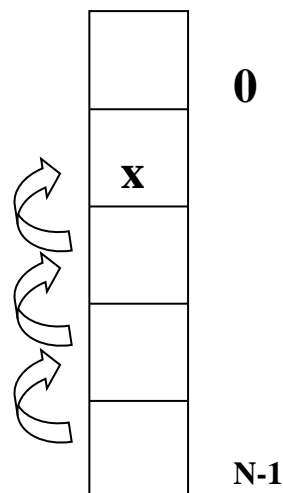
ENDIF

END

Primera posición a correr

Segunda posición a correr

Tercera posición a correr



```

import ucn.StdIn;
import ucn.StdOut;
public class Eliminar{
    public static void main(String args[]){
        StdOut.print ("Ingrese cantidad de elementos: ");
        int N = StdIn.readInt()
        int [] A = new int[N];
        for(int i = 0; i < N; i++){
            StdOut.print ("Ingrese datos del vector: ");
            A[i] = stdIn.readInt();
        }
        StdOut.print ("Ingrese valor a eliminar:");
        int X = StdIn.readInt();
        //Se busca el elemento a eliminar
        int i = 0;
        while (i < N && X != A[i]) {
            i++;
        }
        if (i == N) {
            StdOut.println ('Valor no encontrado');
        }
        else { //Corrimiento
            for(int k = i ; k <N-1; k++) {
                A[k] = A[k + 1];
            }
            N--; //El vector tiene un elemento menos
        }
        for (int I = 0 ; I<= N-1; I++) {
            StdOut.println (A[I]);
        }
    }//Fin main
}//Fin Eliminar

```

- **Inserción**

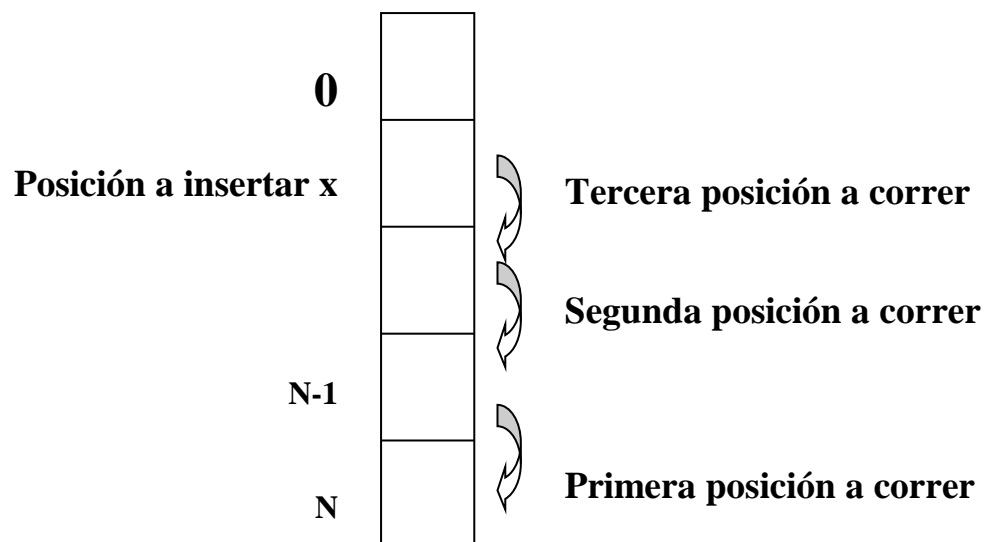
- Leer un arreglo de N elementos que está ordenado.
- Leer un valor X e insertarlo en el arreglo de manera que se conserve el orden.
- Imprimir el nuevo arreglo.

PROGRAM Insertar

```

BEGIN
  "Leer arreglo"
  "Leer X"
  "Buscar la posición donde se debe insertar X para
    que el arreglo siga ordenado"
  "Corrimiento"
  "Insertar X"
  N ← N + 1
  "Imprimir arreglo resultante"
END

```



```

import ucn.StdIn;
import ucn.StdOut;
public class Insertar{
    public static void main(String args[]){
        StdOut.print (“Ingrese cantidad de elementos: ”);
        int N = StdIn.readInt()
        int [] A = new int[N+1];

        for(int i = 0; i < N; i++){
            StdOut.print (“Ingrese datos del vector: ” );
            A[i] = StdIn.readInt();
        }
        StdOut.println (“Ingrese valor a insertar: ”);
        int x = stdIn.readInt();
        //Búsqueda de la posición donde hacer la inserción,
        //de manera de mantener el orden de los datos,
        //de mayor a menor
        int i = 0;
        while (i <= N-1 && x > A[i]){
            i++;
        }
        //Corrimiento
        for(int k = N ; k >= i + 1; k--){
            A[k] = A[k - 1];
        }
        A[i] = x;
        N ++;
        for(int i = 0; i < N; i++){
            StdOut.println (A[i]);
        }
    }
}//Fin main
}//Fin Insertar

```

Ejercicios propuestos con arreglos

1. Leer 1000 datos numéricos enteros desde pantalla y cargarlos en un vector de 12000 posiciones.
2. Leer 300 números reales desde pantalla y ubíquelos en las posiciones pares de un vector de 1000 posiciones.
3. Diseñe un algoritmo que genere un vector, de tal manera que en cada posición del vector coloque el subíndice respectivo. El tamaño del vector es N y debe ser leído desde pantalla.
4. Leer 100 valores enteros desde pantalla y almacenarlos en un vector A. Calcular e imprimir la suma de estos valores.

Program cuatro

```
begin  
  "Leer datos y cargarlos en vector"  
  "Sumar los elementos del vector"  
  "Imprimir el resultado"  
end
```

5. Leer desde pantalla un arreglo de reales cuyo tamaño es N, donde N viene como primer dato. Calcular e imprimir el promedio de los elementos del arreglo que se encuentran entre 10 y 25.

Program cinco

```
begin
  "Leer N"
  for I = 1 to N
    "Leer valor y dejarlo en el vector"
    "Acumular el valor si corresponde"
  end for
  "Calcular e imprimir el promedio si corresponde"
end
```

6. Leer a, b, c, d, e (reales) desde pantalla y cargarlos en las primeras 5 posiciones de un vector.
Calcular $((a + b) * c) / (d - e)$ y almacenar el resultado en la posición 6 del vector. Considere que $(d - e)$ puede ser 0.

Program seis

```
begin
  "Leer a, b, c, d, e y dejarlas en las cinco primeras
  posiciones del vector "
  if ( (d - e) <> 0) then
    "Calcular valor y dejarlo en posicion 6 del arreglo"
  end if
end
```

7. Leer desde pantalla los datos de un vector de enteros de tamaño N. Se debe calcular el promedio y luego determinar e imprimir los datos y la posición de los que sobrepasan en 5 al promedio. N es el primer dato a leer.

Program siete

```
begin
  "Leer N"
  for I = 1 to N
    "Leer dato y dejarlo en el vector"
    "Acumular dato"
  end for
  "Calcular promedio"
  for I = 1 to N
    "Determinar si el elemento I del vector es mayor
    en 5 unidades que el promedio"
    "Imprimir posición I si corresponde"
  end for
end
```


8. Inicializar el vector A, de tamaño N, con 1 en las posiciones correspondientes a K, donde K se determina según la serie de Fibonacci, cuyos 2 primeros términos son F1 y F2 ($F1 < F2$). Las otras posiciones del vector deben ser inicializadas con 0. Los valores de F1, F2 y N deben ser leídos desde pantalla.

Program ocho

```
begin
  "Leer N, f1 y f2"
  "Inicializar vector A con ceros"
  A[f1] ← 1
  A[f2] ← 1
  "Calcular valor (Fibonacci)"
  while (valor <= N)
    A[valor] ← 1
    "Actualizar f1 y f2"
    "Calcular valor"
  end while
end
```

9. Leer un vector de reales de tamaño N. N es el primer dato.
Listar los 3 valores mayores y los 2 menores.

Program nueve_version1

```
begin
  "Leer N"
  "Leer vector "
  "Ordenar vector de mayor a menor"
  "Imprimir los tres mayores"
  "Imprimir los dos menores"
end
```

Program nueve_version2

```
begin
  "Leer N"
  "Leer vector"
  "Inicializar los tres mayores y los dos menores"
  "Determinar los tres mayores"
  "Imprimir los tres mayores"
  "Determinar los dos menores"
  "Imprimir los dos menores"
end
```

10. Calcular e imprimir:

$$S = \frac{\sum_{i=1}^N (X_i - \bar{X})^2}{N} \quad ; \quad \bar{X} = \frac{\sum_{i=1}^N X_i}{N}$$

Valores de N y X_i deben ser leídos desde pantalla.

Program diez

```
begin
  "Leer datos del vector"
  "Sumar los elementos del vector"
  "Sacar promedio"
  "Calcular e imprimir S"
end
```

11. Inicializar la matriz de 10 x 10 con 1 en la diagonal principal y 0 en las restantes posiciones.

12. Dada la matriz A de NxM, sumar los elementos de las filas y el resultado almacenarlo en el vector FILA(I), con I=0,N-1

Program doce

```
begin
  for I = 0 to N-1
    "Sumar los elementos de la fila I"
    Fila[I] ← suma
  end for
end
```

13. Calcular la suma de dos matrices M1 de NxM y M2 de NxM. Ambas matrices deben leerse previamente. Se debe imprimir la matriz resultante.

Program catorce

```
begin
    "Leer N y M"
    "Leer M1"
    "Leer M2"
    "Sumar matrices"
    "Imprimir matriz resultante"
End
```

14. Realizar un algoritmo que, utilizando un vector, genere los 100 primeros números de la serie de Fibonacci, los cuales deben quedar almacenados en el vector.

15. Escribir un algoritmo que permita rotar hacia la derecha k veces los valores de un vector V de 1000 posiciones. El valor de k debe leerse. Ejemplo: Para K = 1

a	b	c	d	e	f
---	---	---	---	---	---

b	c	d	e	f	a
---	---	---	---	---	---

Program

```
begin
    "Leer vector"
    "Leer K"
    "Rotar el vector"
End
```

16. Invertir un vector dado, sin usar otro vector.

Program

```

begin
  "Leer vector"
  "Invertir vector"
end

```

17. Dada una matriz de NxM

- a) Almacenar en un vector llamado FILA, la suma de cada fila de la matriz
- b) Almacenar en un vector llamado COLUMNA, la suma de cada columna de la matriz
- c) Determinar las posiciones de la matriz que contienen valores menores que 7

Program

```

begin
  for I = 0 to N-1
    "Sumar fila i-esima de la matriz"
    Fila[I] ← sumaF
  end for
  for J = 0 to M-1
    "Sumar columna i-esima de la matriz"
    Columna[J] ← sumaC
  end for
  "Recorrer la matriz y determinar las posiciones
  que contienen valores menores que siete "
end

```

- 18.** Leer la edad y nombre de 200 personas y almacenarlas en los vectores EDAD y NOMBRE respectivamente, de manera que EDAD[i] corresponda a NOMBRE[i]. Luego ordenar los vectores por edad en orden decreciente, teniendo cuidado de no perder la relación Edad-Nombre.

Program

```
begin
    "Leer vectores edad y nombre"
    "Ordenar vector edad, cambiando cuando corresponda
    también en el vector nombre"
end
```

- 19.** Se tiene un vector V con 500 números, determinar cuántos números se repiten y eliminar todas las repeticiones.
- 20.** Un alumno de Ingeniería Civil de la U.C.N., al término de su carrera desea saber cómo fue su desempeño en la Universidad. Los datos se leen desde pantalla, donde el fin de datos es nota = -1. Para esto solicitó a un alumno de Programación avanzada que le confeccione un algoritmo que determine:
- a) Cuántas de sus notas son mayores que 5.5
 - b) Cuántas de sus notas son menores que 4.5
 - c) El promedio de notas
 - d) El porcentaje que representan las notas mayores que 6.0
 - e) La nota menor y la nota mayor
 - f) Un listado de notas de mayor a menor.

Program

```

begin
    "Leer datos y cargarlos en un vector"
    "Determinar cuántas notas son mayores que 5.5"
    "Determinar cuántas notas son menores que 4.5"
    "Determinar el promedio de notas"
    "Determinar el porcentaje de notas mayores que 6.0"
    "Determinar la nota mayor"
    "Determinar la nota menor"
    "Obtener un listado de notas de mayor a menor"
end

```

- 21.** Se dispone de un archivo que contiene una matriz A de $N \times M$ y un vector B de M elementos. Calcular e imprimir el producto entre el arreglo bidimensional y el vector. Precediendo a los datos, el archivo contiene los valores de N y M. El resultado debe almacenarse en un vector C de largo N.

Program

```

begin
    "Leer datos"
    "Calcular producto y dejarlo en un vector C"
    "Imprimir el vector C"
End

```

- 22.** Leer los datos de 2 matrices, A de $N_1 \times N_2$ y B de $N_2 \times N_3$. Calcular e imprimir la matriz $C = A * B$. Los primeros datos que se leen son N_1 , N_2 y N_3 .

Nota: $C_{ij} = \sum_{h=1}^{N2} a_{ih} b_{hj}$

Program

```
begin
    "Leer datos (N1, N2, N3 y las matrices A y B)"
    "Multiplicar matrices"
    "Imprimir matriz resultante"
end
```


Más Ejercicios Propuestos

1. Dado el siguiente programa en Java, indique lo que imprime. (Rutee). Suponga que el valor de N es 5.

```

...
public class Problema1 {
    public static void main (String [] args) {
        StdOut.print("Ingrese N:");
        int N = StdIn.readInt();
        int [] vector = new int[N];
        int [] vector1 = new int[N];
        double sum = 0;
        for (int i=0; i<N; i++){
            vector[i] = Math.pow(i,2);
            vector1[i] = vector[i] + 5;
            vector[i] = vector1[i] * 2;
            StdOut.println("vector["+i+"]=" + vector[i]);
            sum = sum +vector[i];
        }
        if (N >0 ){
            double prom = sum /N;
            StdOut.println("Promedio: " + prom);
        }
        else {
            StdOut.println("No hay datos");
        }
    } //Fin main
} //Fin Problema1

```

Para este problema, ¿era necesario trabajar con vectores? Justifique su respuesta.

2.

- a. Dado el siguiente programa en Java, indique lo que imprime. (Rutee). Suponga que al leer datos desde pantalla, ingresaría los valores que se indican a continuación y en el orden señalado:
- | | | | | | | |
|---|---|----|----|----|---|---|
| 2 | 7 | 6 | 18 | 19 | 4 | 5 |
| 3 | 2 | 12 | 8 | 3 | 4 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |

```
....
public class Problema2 {
    public static void main (String [] args) {
        int [][]matriz = new int[3][3];
        int [][] otraMatriz = new int[3][3];
        int [][] matrizSum = new int[3][3];
        for(int i= 0; i <3; i++){
            for(int j= 0; j <3; j++){
                matriz[i][j] = StdIn.readInt();
            }
        }
        for(int j= 0; j <3; j++){
            for(int i= 0; i <3; i++){
                otraMatriz[i][j] = StdIn.readInt();
            }
        }
        for(int i= 0; i <3; i++){
            for(int j= 0; j <3; j++){
                matrizSum[i][j] = matriz[i][j] + otraMatriz[i][j];
            }
        }
        int sum = 0;
        for(int i= 0; i <3; i++){
            sum = sum + matrizSum[i][i];
        }
        StdOut.println("Suma: " + sum);
    } //fin main
} //Fin Problema2
```

- b. ¿Qué representa el valor de sum que se imprimió?
- c. Agregue el código que sea necesario para imprimir la suma de los elementos de la diagonal secundaria de matrizSum

3. Leer datos desde pantalla, y determinar cuál es el dato que más se repite. Son N datos, donde N es el primer dato a leer.

	num	cont
0		
1		
...		
n-1		

4. Construir un programa en Java que lea el nombre, rut y PPA de todos los alumnos del curso de Programación Avanzada del segundo semestre del 2016, fin de datos, nombre = “fin”, y despliegue:
- PPA y nombre del alumno, cuyo rut se ingresa por pantalla
 - Nombre y PPA del alumno que tiene el menor PPA
 - Porcentaje de alumnos que tienen un PPA igual a un determinado PPA que se lee desde pantalla (con respecto al total de alumnos)
5. Construya un programa que a partir de un archivo de ventas (ventas.txt) de una empresa, determine los diez productos más vendidos, ordenando los resultados de menor a mayor; y los productos que no se vendieron. El archivo de ventas contiene por cada registro el código del producto (entero) y la cantidad vendida. Además, se tiene un archivo inventarios.txt, que contiene en cada registro el código del producto y su stock.

Considere que cada vez que procesa un registro del archivo de ventas debe actualizar el stock del producto correspondiente.

Producto	Stock	cantidad

0.3. Subprogramas en Java

Ejemplo 1 programa Phyton

```
# -*- coding: utf-8 -*-
```

```
import numpy as np
```

```
def imprimirMatriz(m):
```

Parámetro formal

```
    for x in range(4):
        for y in range(4):
            if m[y][x] > 50:
                print("-", end="")
            elif m[y][x] < 50:
                print("-", end="")
            else:
                print("=", end="")
        print()
    print()
```

**Definición
del
subprograma**

```
m = np.zeros([4, 4])
```

```
for x in range(4):
    for y in range(4):
        m[y][x] = np.random.uniform(0, 100)
```

```
imprimirMatriz(m)
```

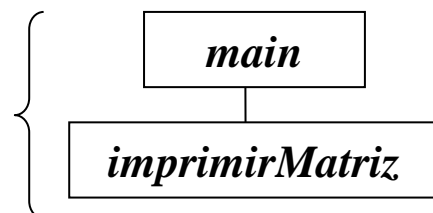
Parámetro real

```
for x in range(4):
    for y in range(4):
        m[y][x] = 2 * m[y][x]
```

```
imprimirMatriz(m)
```

**Invocación o
llamado del
subprograma**

Estructura del programa



Equivalente en Java

```

import ucn.Stdout;
public class Ejemplo {
    public static void imprimirMatriz(int[][] m){
        for(int i=0; i<4; i++) {
            for(int j=0; j<4; j++) {
                if(m[i][j] > 50) {
                    StdOut.print("+");
                }else {
                    if(m[i][j] < 50) {
                        StdOut.print("-");
                    }else {
                        StdOut.print("=");
                    }
                }
            }
            StdOut.println();
        }
        StdOut.println();
    }

    public static void main(String[] args) {
        int[][] matriz = new int[4][4];
        for(int i=0; i<4; i++) {
            for(int j=0; j<4; j++) {
                matriz[i][j] = (int) Math.random()*100;
            }
        }
        imprimirMatriz(matriz);
        for(int i=0; i<4; i++) {
            for(int j=0; j<4; j++) {
                matriz[i][j] = 2 * matriz[i][j];
            }
        }
        imprimirMatriz(matriz);
    }
}

```

Procedimiento

Definición del subprograma

Parámetro formal

Parámetro real

Invocación o llamado del subprograma

Ejemplo 2

```
# -*- coding: utf-8 -*-
```

```
import numpy as np
```

```
def imprimirMatriz(m):
    for x in range(4):
        for y in range(4):
            if m[y][x] > 50:
                print("+", end="")
            elif m[y][x] < 50:
                print("-", end="")
            else:
                print("=", end="")
        print()
    print()
```

```
def amplificarMatriz(m, k):
    for x in range(4):
        for y in range(4):
            m[y][x] = k * m[y][x]
```

*Otro
subprograma*

```
m = np.zeros([4, 4])
```

```
for x in range(4):
    for y in range(4):
        m[y][x] = np.random.uniform(0, 100)
```

```
imprimirMatriz(m)
```

```
amplificarMatriz(m, 2)
```

```
imprimirMatriz(m)
```

```
amplificarMatriz(m, 2)
```

```
imprimirMatriz(m)
```

*Invocación del
subprograma*

*Podríamos amplificar
la matriz por 3*

3

```
# -*- coding: utf-8 -*-
```

```
import numpy as np
```

```
def imprimirMatriz(m, filas, columnas):
    for x in range(columnas):
        for y in range(filas):
            if m[y][x] > 50:
                print("+", end="")
            elif m[y][x] < 50:
                print("-", end="")
            else:
                print("=", end="")
        print()
```

*Subprograma
Procedimiento*

```
def amplificarMatriz(m, filas, columnas, k):
    for x in range(columnas):
        for y in range(filas):
            m[y][x] = k * m[y][x]
```

*Subprograma
Procedimiento*

```
def sumarValores(m, filas, columnas):
    suma = 0
    for x in range(columnas):
        for y in range(filas):
            suma = suma + m[y][x]
```

*Subprograma
Función*

```
return suma
```

```
m = np.zeros([4, 4])
```

```
for x in range(4):
    for y in range(4):
        m[y][x] = np.random.uniform(0, 100)
```

```
imprimirMatriz(m, 4, 4)
s = sumarValores(m, 4, 4);
```



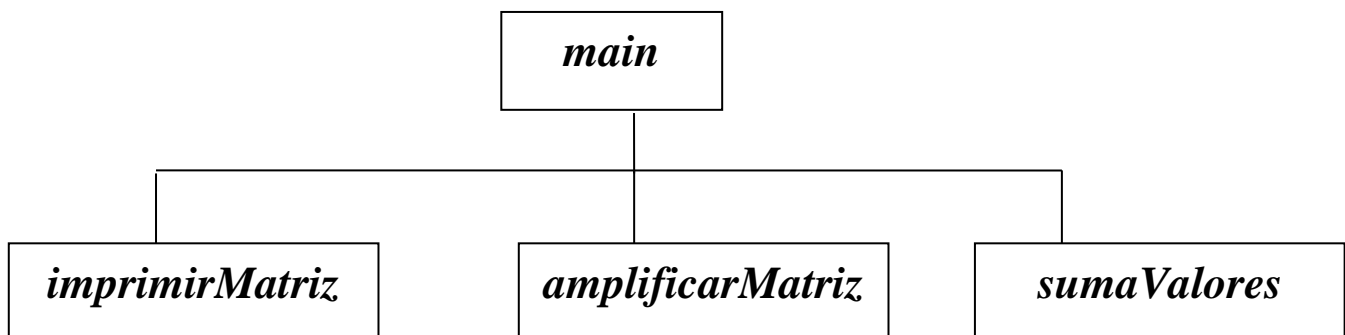
```
print("suma total=" + str(s))

amplificarMatriz(m, 4, 4, 2)
imprimirMatriz(m, 4, 4)
print("suma total=" + str(sumarValores(m, 4, 4)))

amplificarMatriz(m, 4, 4, 2)
imprimirMatriz(m, 4, 4)
print("suma total=" + str(sumarValores(m, 4, 4)))

amplificarMatriz(m, 4, 4, 0.4)
imprimirMatriz(m, 4, 4)
print("suma total=" + str(sumarValores(m, 4, 4)))
```

Estructura del programa



```
import ucn.StdOut;

public class Ejemplo {

    public static void imprimirMatriz(int[][] m,
                                      int filas, int columnas) {
        for(int i=0; i<filas; i++) {
            for(int j=0; j<columnas; j++) {
                if(m[i][j] > 50) {
                    StdOut.print("+");
                }else {
                    if(m[i][j] < 50)
                        StdOut.print("-");
                    else
                        StdOut.print("=");
                }
            }
            StdOut.println();
        }
        StdOut.println();
    }

    public static void amplificarMatriz(int[][] m,
                                       int filas, int columnas, int k) {
        for(int i=0; i<filas; i++)
            for(int j=0; j<columnas; j++)
                m[i][j] = k * m[i][j];
    }
}
```

Función

```

public static int sumarValores(int[][] m,
                                int filas, int columnas) {
    int suma = 0;
    for(int i=0; i<filas; i++)
        for(int j=0; j<columnas; j++)
            suma = suma + m[i][j];
    // suma += m[i][j];
    return suma;
}

public static void main(String[] args) {
    int[][] matriz = new int[4][4];

    for(int i=0; i<4; i++){
        for(int j=0; j<4; j++){
            matriz[i][j]=(int) Math.random()*100;
        }
    }

    imprimirMatriz(matriz, 4, 4);
    int s = sumarValores(matriz, 4, 4);
    StdOut.println("Suma Total: " + s);

    amplificarMatriz(matriz, 4, 4, 2);
    imprimirMatriz(matriz, 4, 4);
    StdOut.println("Suma Total: " +
        sumarValores(matriz, 4, 4));

    amplificarMatriz(matriz, 4, 4, 2);
    imprimirMatriz(matriz, 4, 4);
    StdOut.println("Suma Total: " +
        sumarValores(matriz, 4, 4));

    amplificarMatriz(matriz, 4, 4, 4);
    imprimirMatriz(matriz, 4, 4);
    StdOut.println("Suma Total: " +
        sumarValores(matriz, 4, 4));
}

```

Ejemplo 3

```
# -*- coding: utf-8 -*-
```

```
def esPrimo(n):  
    for x in range(2, n):  
        if n % x == 0:  
            return False  
    return True  
  
for i in range(1, 100):  
    print("es primo el " + str(i) + " = " +  
          str(esPrimo(i)))
```

Se imprime

```
es primo el 1 = True  
es primo el 2 = True  
es primo el 3 = True  
es primo el 4 = False  
es primo el 5 = True  
es primo el 6 = False  
es primo el 7 = True  
es primo el 8 = False  
...(etc)...
```

En Java:

```
import ucn.Stdout;

public class Ejemplo {

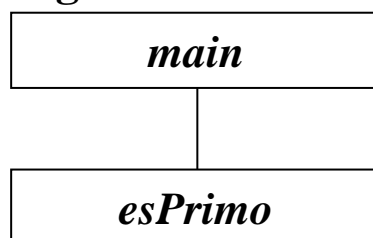
    public static boolean esPrimo(int n) {
        for(int x=2; x<n; x++)
            if(n % x == 0)
                return false;
        return true;
    }

    public static void main(String[] args) {

        for(int i=1; i<100; i++)
            StdOut.println("Es primo el: " + i +
                           " = " + esPrimo(i));
    }
}
```

```
Es primo el: 1 = true
Es primo el: 2 = true
Es primo el: 3 = true
Es primo el: 4 = false
Es primo el: 5 = true
Es primo el: 6 = false
Es primo el: 7 = true
Es primo el: 8 = false
...(etc)...
```

Estructura del programa



Problema

Creemos una función que me retorne el enésimo número primo, considerando que los primos son:

1, 2, 3, 5, 7, etc.

O sea, creemos una función que reciba un parámetro, y que cumpla lo siguiente:

Parámetro	Número retornado
1	1
2	2
5	7
17	53
19	61

```
# -*- coding: utf-8 -*-
def esPrimo(n):
    for x in range(2, n):
        if n % x == 0:
            return False
    return True

def enesimoPrimo(n):
    faltan = n
    candidato = 0

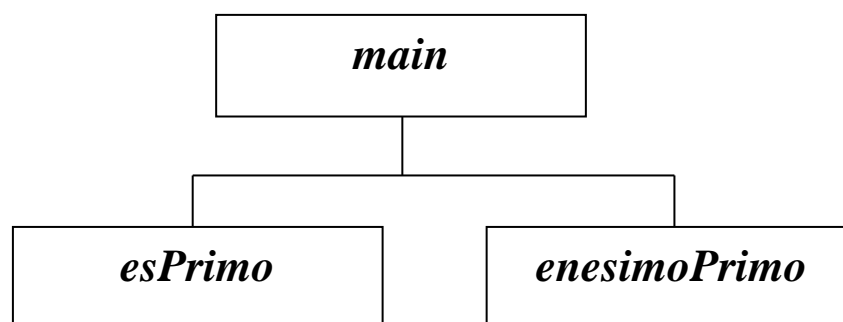
    while faltan > 0:
        candidato = candidato + 1
        if esPrimo(candidato):
            faltan = faltan - 1
    return candidato

for i in range(1, 20):
    print("primo " + str(i) + " = " +
          str(enesimoPrimo(i)))
```

*Función que
retorna el
enésimo primo*

```
primo 1 = 1  
primo 2 = 2  
primo 3 = 3  
primo 4 = 5  
primo 5 = 7  
primo 6 = 11  
primo 7 = 13  
primo 8 = 17  
primo 9 = 19  
primo 10 = 23  
primo 11 = 29  
primo 12 = 31  
primo 13 = 37  
primo 14 = 41  
primo 15 = 43  
primo 16 = 47  
primo 17 = 53  
primo 18 = 59  
primo 19 = 61
```

Estructura del programa



En Java:

```
import ucn.Stdout;

public class Ejemplo {
    public static boolean esPrimo(int n) {
        for(int x=2; x<n; x++)
            if(n % x == 0)
                return false;
        return true;
    }

    public static int enesimoPrimo(int n) {
        int faltan = n;
        int candidato = 0;

        while(faltan > 0) {
            candidato = candidato + 1;
            if(esPrimo(candidato)) {
                faltan = faltan - 1;
            }
        }
        return candidato;
    }

    public static void main(String[] args) {
        for(int i=1; i<20; i++)
            StdOut.println("El primo: " + i +
                           " = " + enesimoPrimo(i));
    }
}
```

*Función que
retorna el
enésimo primo*

El primo 1 = 1	El primo 8 = 17	El primo 15 = 43
El primo 2 = 2	El primo 9 = 19	El primo 16 = 47
El primo 3 = 3	El primo 10 = 23	El primo 17 = 53
El primo 4 = 5	El primo 11 = 29	El primo 18 = 59
El primo 5 = 7	El primo 12 = 31	El primo 19 = 61
El primo 6 = 11	El primo 13 = 37	
El primo 7 = 13	El primo 14 = 41	

Traspaso de parámetros

- En **Java** el traspaso de parámetros es **por valor**:
 - Implica que para el subprograma se crea una copia de la variable que se ha traspasado sin afectar el valor de la variable “original”. Sólo interesa el valor, no las modificaciones que pueda tener dentro del subprograma.
 - Son parámetros unidireccionales, que pasan información desde el programa que invoca al subprograma.

parámetro real



parámetro formal

➤ **Ejemplo:**

parámetro formal

```

public class Ejemplo{
    public static void trazaLinea (int N){
        for(int i = 1; i<= N; i++) {
            StdOut.print (“*”);
        }
        StdOut.println (“”);
    }
}
  
```

```

public static void main(String [] Args){
    int num = 9;

    trazaLinea(num);

    StdOut.println ("HOLA");

    trazaLinea(5);
    trazaLinea(num+2);
}

```

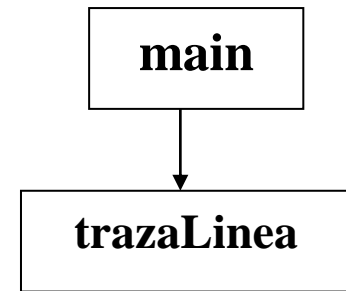
parámetro real

Estructura del programa

```

*****
HOLA
*****
*****

```



¿Qué sucede si observamos el siguiente main?

```

public static void main(String [] Args){
    for(int i = 1; i <= 4; i++){
        trazaLinea(i);
    }
}

```

Se imprime:

```

*
**
***
****

```

- **Por referencia:**

- En ciertas circunstancias se deseará escribir un subprograma que sea capaz de cambiar el valor de una variable definida desde donde se llama al subprograma.
- Para que un parámetro pueda cambiar el valor de la variable definida en el enunciado de llamada, debe ser declarada como *parámetro variable (por referencia)* en el encabezado del subprograma.
- Implica que en el subprograma se puede afectar el valor de la variable “original”. Se pasa una referencia a la posición de memoria donde se encuentra dicho valor.
- Se utilizan tanto para recibir como para transmitir información entre el programa que llama y el subprograma

parámetro real



parámetro formal

Otro ejemplo

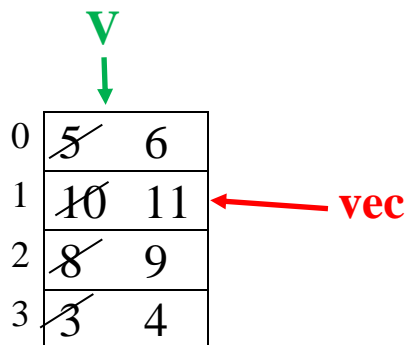
```
public class Ejemplo1Subprogramas{
```

```
    public static void leeVector(int N,int []V) {
        for(int i = 0; i <= N-1; i++){
            StdOut.print ("Ingrese elemento " + i + " del vector");
            V[i] = StdIn.readInt();
        }
    }
```

```
    public static int [] leeVector(int N){
        int [] V = new int[N];
        for(int i = 0; i <= N-1; i++){
            StdOut.println("Ingrese elemento " + i + " del vector");
            V[i] = StdIn.readInt();
        }
        return V;
    }
```

Alternativa como función

N = 4

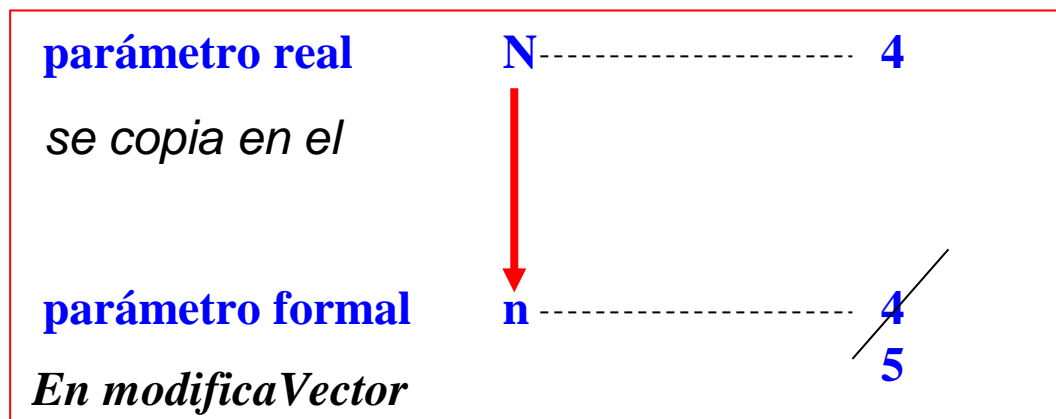


0	5	6
1	10	11
2	8	9
3	3	4

```
public static void imprimeVector(int [] V, int n){
    for( int i = 0; i <=n-1; i++) {
        StdOut.println(“V[“+i+”]: “+ V[i]);
    }
    StdOut.println (“”);
}
```

```
public static int calculaSuma(int []V, int n){  
    int suma = 0;  
    for( int i = 0; i <=n-1; i++) {  
        suma = suma + V[i];  
    }  
    return suma;  
}
```

```
public static void modificaVector (int []V, int n){
    for( int i = 0; i <=n-1; i++) {
        V[i]++;
    }
    //n++;
    //Este cambio de n no se reflejaría en el main,
    //ya que el traspaso de parámetros es por valor.
} //fin modificaVector
```



```
public static void main(String [] args) {
```

```
    StdIn.print (“Ingrese tamaño del vector: “);  
    int N = StdOut.readInt(); // N = 4
```

```
    int [] vec = new int[N];  
    leeVector(N, vec);
```

Llamada a la función

```
    int [] vec = leeVector(N);
```



```
    imprimeVector (vec, N);
```

```
    modificaVector(vec, N);
```

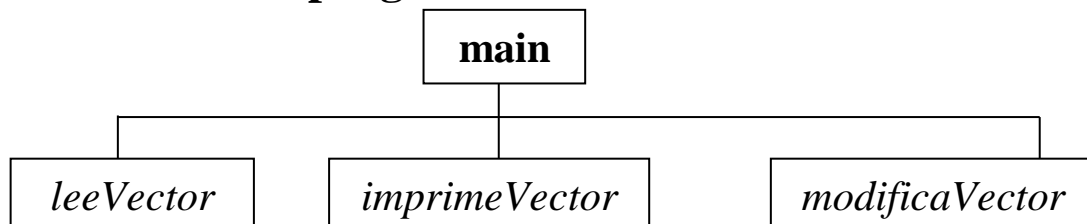
```
    Ejemplo1Subprogramas.imprimeVector (vec, N);  
    //Idem a imprimeVector...
```

```
    StdOut.println (“Suma elementos del vector:”+  
                    calculaSuma(vec, N);
```

```
    } //Fin main
```

```
} //Fin Ejemplo1Subprograma
```

Estructura del programa



0.4 Archivos

Ejemplo

Leer un archivo con datos de nombre de alumno, matrícula y tres notas, y calcular el promedio para cada alumno y escribirlo en un archivo junto con el nombre y la matrícula. El archivo de entrada se llama datos.txt y el de salida Salida.txt.

Program Archivo

Begin

“Definir el archivo de lectura datos.txt”

“Definir el archivo de salida Salida.txt”

“Mientras existan registros en el archivo datos.txt”

Begin

“Leer registro del archivo datos.txt”

“Obtener cada uno de los campos del registro”

“Calcular promedio”

“Formar un registro de salida con el nombre, matrícula y promedio”

“Grabar el registro en el archivo Resultados.txt”

End;

“Cerrar archivos”

End.

En Python:

```

archivoEntrada = open('datos.txt', 'r')
archivoSalida = open('Salida.txt', 'w')
for linea in archivoEntrada.readlines():
    nombre, matricula, nota1, nota2, nota3 = linea.split(",")
    promedio = (float(nota1) + float(nota2) + float(nota3)) / 3
    archivoSalida.write(nombre + ', ' + matricula + ', ' + str(promedio) + '\n')
archivoEntrada.close()
archivoSalida.close()

```

Se obtienen como String

```

linea=nombre+', '+ matricula+ ', ' + str(promedio)+ '\n'
archivoSalida.write(linea)

```

Archivo “datos.txt”

```

Daniela Fernandez,89.446-5,3.0,4.0,5.0
Jose Ramirez,92.889-0,3.5,3.0,5.5
Sergio Perez,89.234-7,4.0,4.5,3.7

```

Archivo “Salida.txt”

```

Daniela Fernandez,89.446-5,4.0
Jose Ramirez,92.889-0,4.0
Sergio Perez,89.234-7,4.0666666666666666

```


En Java:

```
import java.io.IOException;
```

```
import ucn.ArchivoEntrada;
```

```
import ucn.ArchivoSalida;
```

```
import ucn.Registro;
```

```
import ucn.StdIn;
```

```
import ucn.Stdout;
```

```
public class Ejemplo{
```

```
    public static void main(String args[]){
```

```
        ArchivoEntrada arch1=new ArchivoEntrada("Datos.txt");
```

```
        ArchivoSalida arch2=new ArchivoSalida("Resultado.txt");
```

```
        while(!arch1.isEndFile()){//Mientras no sea fin de archivo
```

```
            //Se obtiene el registro
```

```
            Registro regEnt = arch1.getRegistro();
```

```
            //Se separa el registro en los campos que lo constituyen
```

```
            String nombre = regEnt.getString();
```

```
            String matricula = regEnt.getString();
```

```
            double n1 = regEnt.getDouble();
```

```
            double n2 = regEnt.getDouble();
```

```
            double n3 = regEnt.getDouble();
```

```
            //Se calcula el promedio
```

```
            double promAl = (n1+n2+n3)/3;
```

```
//Se crea un registro de salida con 3 campos  
Registro registroSalida = new Registro(3);
```

```
//Se agregan cada uno de los campos  
//al registro de salida  
registroSalida.agregarCampo(nombre);  
registroSalida.agregarCampo(matricula);  
registroSalida.agregarCampo(promAl);
```

```
//Se graba el registro de salida en arch2  
arch2.writeRegistro(registroSalida);
```

```
} //Fin while
```

```
//Se cierran los archivos  
arch1.close();  
arch2.close();
```

```
} // fin main
```

```
} // fin Ejemplo
```

0.4. Ejercicios

Problema

Se tiene un archivo con los datos de una matriz que contiene enteros. El primer registro de la matriz contiene la cantidad de filas y la cantidad de columnas de la matriz. La estructura de c/u de los siguientes registros es:

- Letra: Puede ser f o c.
 - Si es f, significa que los datos del registro corresponden a los datos de una fila.
 - Si es c, significa que los datos del registro corresponden a los datos de una columna
- Número: Corresponde al número de la fila o columna según corresponda
- Datos: Corresponde a los datos de la fila o columna según corresponda.

Nota: Suponga que los datos en el archivo están correctos

Ejemplo de Matriz.txt

3,4

f,0,1,2,3,1

c,1,2,5,8

f,2,7,8,9,3

c,2,3,6,9

c,0,1,4,7

f,1,4,5,6,2

c,3,1,2,3

Matriz M1 de 3x4

1	2	3	1
4	5	6	2
7	8	9	3

Parte 1 (sobre la línea diagonal)

Parte 2 (bajo la línea diagonal)

1	4	7
2	5	8
3	6	9
1	2	3

Traspuesta de M1

- Los elementos de la parte 1 son los que están sobre la línea, incluyendo los de la línea
- Los elementos de la parte 2 son los que están bajo la línea, incluyendo los de la línea
- La línea pasa por los elementos que tienen el mismo índice de fila y de columna

Otro ejemplo de Matriz.txt

5,3

f,0,1,2,3

c,1,2,4,2,9,1

f,1,1,4,5

c,0,1,1,6,8,2

c,2,3,5,1,3,3

f,4,2,1,3

f,3,8,9,3

Matriz M2 de 5x3

1	2	3
1	4	5
6	2	1
8	9	3
2	1	3

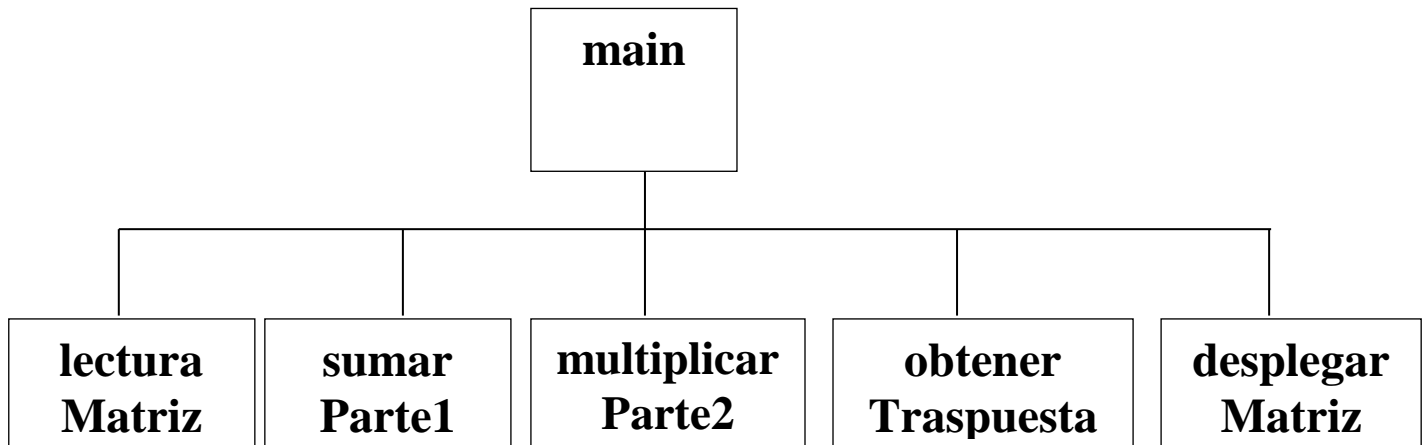
Su programa debe:

1. Desplegar la matriz (en forma de matriz) leída desde el archivo
2. Hacer la traspuesta de la matriz y desplegarla (en forma de matriz)
3. Sumar los elementos de la parte 1 de la matriz y desplegar el resultado
4. Multiplicar los elementos de la parte 2 de la matriz y desplegar el resultado

Se pide:

- a) Dibuje la estructura del programa
- b) Código Java

Estructura del programa



public class Matrices {

```

public static void lecturaMatriz(ArchivoEntrada arch, int [][] matriz,
    int cantFilas, int cantColumnas) throws IOException{
    while(!arch.isEndFile()){
        Registro reg = arch.getRegistro();
        char filaColumna = reg.getChar();
        int num = reg.getInt();
        if (filaColumna == 'f'){
            for(int col = 0; col < cantColumnas; col++){
                matriz[num][col]= reg.getInt();
            }
        }
        else{
            if(filaColumna == 'c'){
                for(int fil = 0; fil < cantFilas; fil++){
                    matriz[fil][num]= reg.getInt();
                }
            }
        }
    }
    arch.close();
}

```

```

public static int sumarParte1(int [][] matriz, int cantFilas,
                                int cantColumnas){
    int suma = 0;
    for(int i = 0; i < cantFilas; i++){
        for(int j = i; j < cantColumnas; j++){
            suma = suma + matriz[i][j];
        }
    }
    return suma;
}

```

```

public static void obtenerTranspuesta(int [][]matriz,
    int cantFilas, int cantColumnas, int[][] matrizTranspuesta){
    for(int i = 0; i < cantFilas; i++){
        for(int j = 0; j < cantColumnas ; j++){
            matrizTranspuesta[j][i]= matriz[i][j];
        }
    }
}

```

```

public static void desplegarMatriz(int [][]matriz, int cantFilas,
    int cantColumnas){
    for(int i = 0; i < cantFilas; i++){
        for(int j = 0; j < cantColumnas; j++){
            StdOut.print(matriz[i][j]+ " ");
        }
        StdOut.println("");
    }
}

```

```

public static int multiplicarParte2(int [][]matriz, int cantFilas,
                                     int cantColumnas){

    int mult = 1;
    if (cantFilas > cantColumnas){
        for(int i = 0; i < cantColumnas; i++){
            for(int j = 0; j <=i; j++){
                mult = mult * matriz[i][j];
            }
        }
        for(int i = cantColumnas; i < cantFilas; i++){
            for(int j = 0; j < cantColumnas; j++){
                mult = mult * matriz[i][j];
            }
        }
    }
    else{ //cantFilas <= cantColumnas
        for(int i = 0; i < cantFilas; i++){
            for(int j = 0; j <=i; j++){
                mult = mult * matriz[i][j];
            }
        }
    }

    return mult;
}

```

```

public static void main(String[] args) throws IOException{

    //Lectura Matriz desde el archivo Matriz.txt
    ArchivoEntrada arch = new ArchivoEntrada("Matriz1.txt");
    Registro reg = arch.getRegistro();
    int cantFilas=reg.getInt();
    int cantColumnas= reg.getInt();

    int [][]matriz = new int [cantFilas][cantColumnas];

    lecturaMatriz(arch, matriz, cantFilas, cantColumnas);

    desplegarMatriz(matriz, cantFilas, cantColumnas);

    int [][] matrizTraspuesta =
                                new int[cantColumnas][cantFilas];

    obtenerTraspuesta(matriz, cantFilas,
                                cantColumnas, matrizTraspuesta);

    desplegarMatriz(matrizTraspuesta, cantColumnas, cantFilas);

    int sumaParte1 = sumarParte1(matriz, cantFilas,
                                cantColumnas);
    StdOut.println("Suma parte 1: " + sumaParte1);

    int multParte2= multiplicarParte2(matriz, cantFilas,
                                cantColumnas);
    StdOut.println("Multiplicación parte 2: " + multParte2);

} //Fin main

//Fin Matrices

```


Ejercicio Propuesto

Se necesita un programa que controle los proyectos de una empresa dedicada al rubro de la minería. La empresa almacena los proyectos en ejecución y los nombres de los integrantes del equipo de trabajo en un archivo llamado “proyectos.txt”. Este archivo tiene en cada registro: el código del proyecto (numero correlativo que empieza en 0), el nombre de un integrante del proyecto y porcentaje de dedicación al proyecto (entero). Debe considerar que los códigos de proyectos se pueden repetir dado que un proyecto puede tener varios integrantes, que una persona puede estar trabajando en más de un proyecto. Los proyectos en el archivo vienen en cualquier orden. Un ejemplo del archivo “proyectos.txt” es:

0,daniel contreras,25

2,jorge rivera,30

1,daniel contreras,20

0,jorge rivera,40

2,daniel contreras,25

...

Así en el proyecto código=0, trabajan daniel contreras con un 25% de dedicación de su tiempo y jorge rivera con un 40%.

Una vez terminada la lectura de los datos, se debe desplegar:

- (1) El porcentaje de dedicación a cada uno de los proyectos de una persona cuyo nombre se lee desde pantalla
- (2) El nombre del integrante que se encuentra más ocupado, esto es el que tiene el mayor porcentaje de su tiempo ocupado. Considere que hay un solo mayor.
- (3) La cantidad de proyectos
- (4) La cantidad de integrantes

- (5) El promedio de integrantes por proyecto, por ejemplo, un valor de 4 significa que en promedio en cada proyecto trabajan 4 personas.
- (6) El porcentaje de proyectos que tienen entre 10 y 15 integrantes

Considere:

- El archivo proyectos.txt no tiene datos erróneos
- Un proyecto tiene a lo más 15 integrantes.
- Hay a lo más 50 proyectos y 200 empleados

Se pide:

- i. Dibuje la(s) estructura(s) de datos a utilizar, junto con una breve descripción.
- ii. Dibuje las estructura del programa
- iii. Código Java de su aplicación.