Fundamentos de Programación

MT. Antonio Garay Espinoza

Robles Picazo Grecia Genesis

Numero de Control: 25130266

Semestre 1 Grupo G

Tarea Estructuras Repetitivas

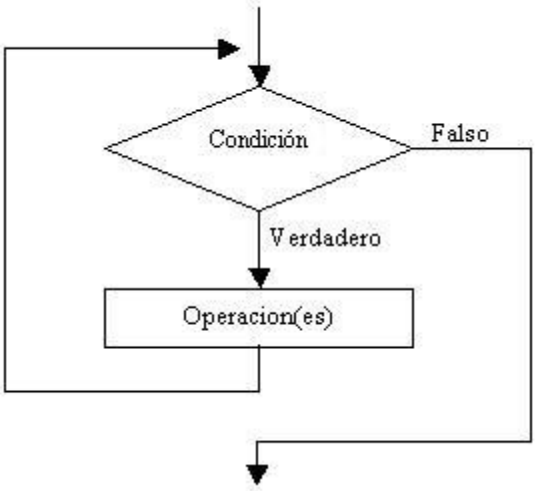
Contenido

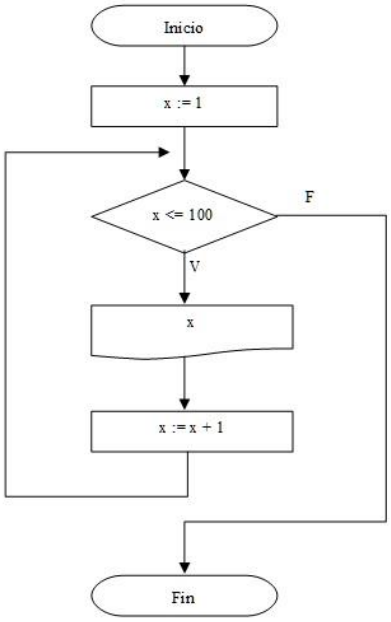
Estructuras de Repetición

Las estructuras de repetición permiten que un conjunto de instrucciones se ejecute múltiples veces sin necesidad de escribir el código repetidamente. Estas estructuras se basan en una condición lógica que determina si el ciclo continúa o finaliza

Estructura Mientras (While)

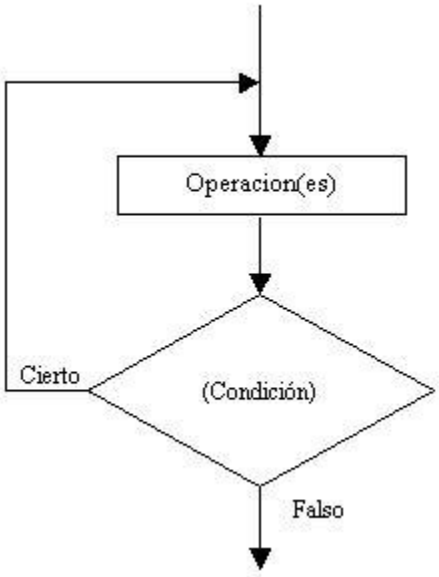
La estructura repetitiva mientras (en inglés while o dowhile: hacer mientras) es aquella en que el cuerpo del bucle se repite mientras se cumple una determinada condición. Cuando se ejecuta la instrucción mientras, la primera cosa que sucede es que se evalúa la condición (una expresión booleana). Si se evalúa falsa, no se toma ninguna acción y el programa prosigue en la siguiente instrucción del bucle. Si la expresión booleana es verdadera, entonces se ejecuta el cuerpo del bucle, después de lo cual se evalúa de nuevo la expresión booleana. Este proceso se repite una y otra vez mientras la expresión booleana (condición) sea verdadera.

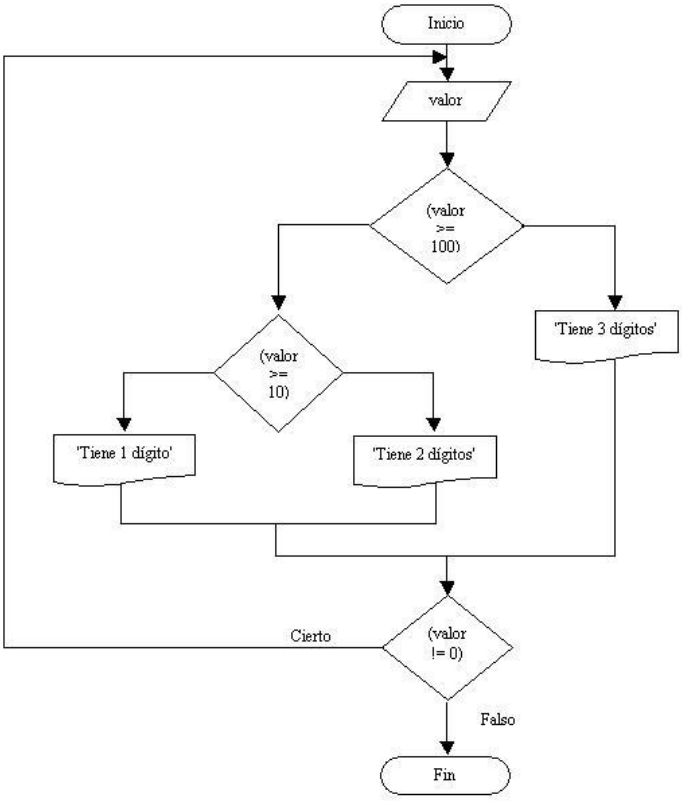




Estructura hacer-mientras ("do-while").

El bucle mientras al igual que el bucle desde que se verá con posterioridad evalúan la expresión al comienzo del bucle de repetición; siempre se utilizan para crear bucle pre-test. Los bucles pre-test se denominan también bucles controlados por la entrada. En numerosas ocasiones se necesita que el conjunto de sentencias que componen el cuerpo del bucle se ejecuten al menos una vez sea cual sea el valor de la expresión o condición de evaluación. Estos bucles se denominan bucles post-test o bucles controlados por la salida. Un caso típico es el bucle hacer-mientras (do-while) existente en lenguajes como C/C++, Java o C#. El bucle hacer-mientras es análogo al bucle mientras y el cuerpo del bucle se ejecuta una y otra vez mientras la condición (expresión booleana) sea verdadera. Existe, sin embargo, una gran diferencia y es que el cuerpo del bucle está encerrado entre las palabras reservadas hacer y mientras, de modo que las sentencias de dicho cuerpo se ejecutan, al menos una vez, antes de que se evalúe la expresión booleana. En otras palabras, el cuerpo del bucle siempre se ejecuta, al menos una vez, incluso aunque la expresión booleana sea falsa.

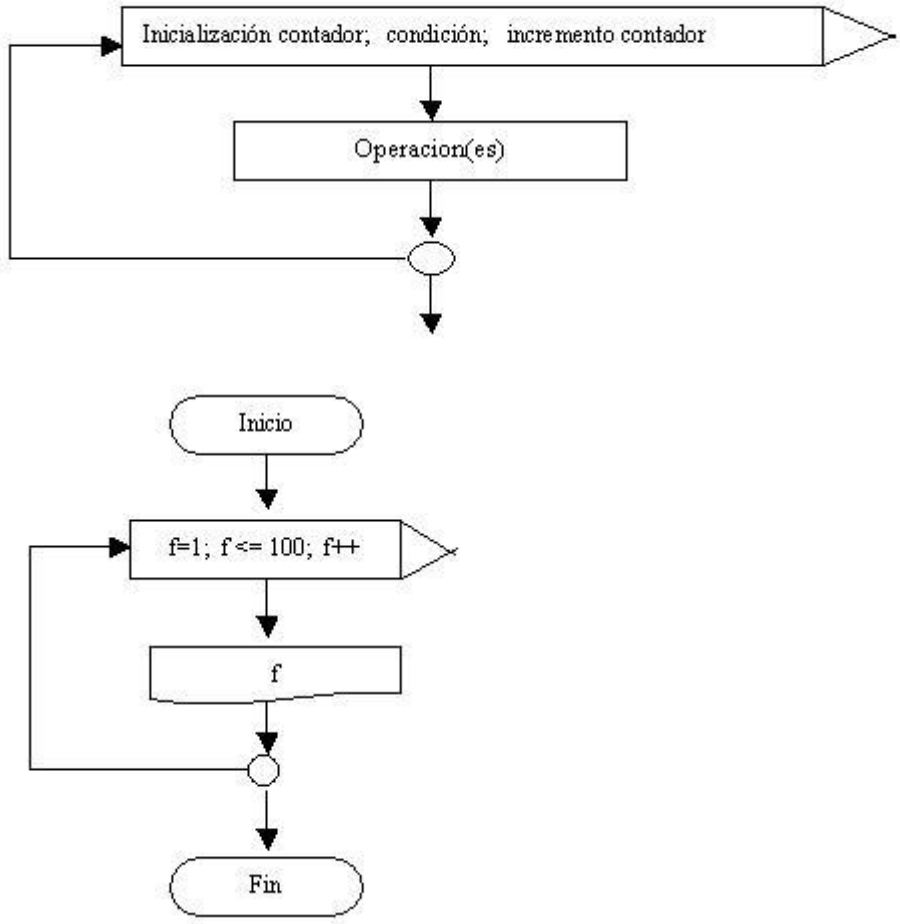




Regla: El bucle hacer-mientras se termina de ejecutar cuando el valor de la condición es falsa. La elección entre un bucle mientras y un bucle hacer-mientras depende del problema de cómputo a resolver. En la mayoría de los casos, la condición de entrada del bucle mientras es la elección correcta. Por ejemplo, si el bucle se utiliza para recorrer una lista de números (o una lista de cualquier tipo de objetos), la lista puede estar vacía, en cuyo caso las sentencias del bucle nunca se ejecutarán. Si se aplica un bucle hacer-mientras nos conduce a un código de errores.

Estructura Desde/para(FOR)

Estructura desde/para ("for") En muchas ocasiones se conoce de antemano el número de veces que se desean ejecutar las acciones de un bucle. En estos casos, en el que el número de iteraciones es fijo, se debe usar la estructura desde o para (for, en inglés). La estructura desde ejecuta las acciones del cuerpo del bucle un número especificado de veces y de modo automático controla el número de iteraciones o pasos a través del cuerpo del bucle.



La estructura desde comienza con un valor inicial de la variable índice y las acciones especificadas se ejecutan, a menos que el valor inicial sea mayor que el valor final. La variable índice se incrementa en uno y si este nuevo valor no excede al final, se ejecutan de nuevo las acciones. Por consiguiente, las acciones específicas en el bucle se ejecutan para cada valor de la variable índice desde el valor inicial hasta el valor final con el incremento de uno en uno. El incremento de la variable índice siempre es 1 si no se indica expresamente lo contrario. Dependiendo del tipo de lenguaje, es posible que el incremento sea distinto de uno, positivo o negativo.

Diferencias entre cada una

Ciclo for

- Se usa cuando se conoce el número exacto de veces que se debe repetir una acción.
- Tiene tres partes: inicialización, condición y actualización.
- La condición se evalúa antes de cada iteración.
- Si la condición es falsa desde el inicio, el bloque de código no se ejecuta.

Ciclo while

- Se usa cuando no se sabe exactamente cuántas veces se ejecutará el código, pero la condición debe cumplirse antes de entrar en el ciclo.
- Solo tiene una condición, que se evalúa antes de ejecutar el bloque de código.
- Si la condición es falsa desde el inicio, el código no se ejecuta nunca.

Ciclo do-while

- Similar a while, pero la gran diferencia es que el bloque de código se ejecuta al menos una vez antes de evaluar la condición.
- Se usa cuando se quiere garantizar que el código se ejecute al menos una vez, sin importar la condición.

Diferencias principales entre los ciclos

Característica	`for`	`while`	`do-while`
¿Cuándo se usa?	Cuando se sabe cuántas veces se repetirá.	Cuando la cantidad de repeticiones depende de una condición.	Cuando se necesita ejecutar el código al menos una vez.
¿Cuándo evalúa la condición?	Antes de cada iteración.	Antes de cada iteración.	Después de ejecutar la primera vez.
¿Puede no ejecutarse nunca?	Sí, si la condición es falsa al inicio.	Sí, si la condición es falsa al inicio.	No, siempre se ejecuta al menos una vez.

¿Como se escriben en lenguaje Java?

WHILE

Este es sin menor duda la forma más básica de un bucle, lo único que hace es que ejecuta un bloque de código una y otra vez mientras una condición sea verdad.

```
1 while (condicion){
2  sentencias;
3}
```

No puede ser tan fácil, ¿o sí?...
La verdad es que lo es, es por eso que es el bucle más poderoso, pero la verdad es que mucha gente lo acaba usando es un estilo más o menos así:

```
1 int i = 0;              //Este es nuestro contador
2while (i < tope){      //Lo vamos a seguir haciendo tope veces
3  sentencias;          //Ejecutamos algo
4i++;                   //Aumentamos el contador en uno
5}
```

Pero el problema es que como te das cuenta resulta poner mucho código para repetir cosas, por eso se inventó un nuevo bucle.

FOR

Este es lo mismo que el bucle while, simplemente con otra sintaxis, es más esta es la sintaxis:

```
1 for (inicialización; condición; incremento){
2  sentencia;
3}
```

Así el último código que vimos con los whiles, se vería así con for:

```
1 for (i=0; i < tope; i++){
2  sentencia;
3}
```

DO WHILE

Al contrario que los bucles for y while que comprueban la condición en lo alto de la misma, el bucle do/while comprueba la condición en la parte baja del mismo, lo cual provoca que el bucle se ejecute como mínimo una vez. La sintaxis del bucle do/while es:

```
1 do{
2   sentencia;
3}
4while(condicion);
```

Break y Continue

Las sentencias de control break y continue permiten modificar y controlar la ejecución de los bucles anteriormente descritos.

- La sentencia break provoca la salida del bucle en el cual se encuentra y la ejecución de la sentencia que se encuentra a continuación del bucle.
- La sentencia continue provoca que el programa vaya directamente a comprobar la condición del bucle en los bucles while y do/while, o bien, que ejecute el incremento y después compruebe la condición en el caso del bucle for.

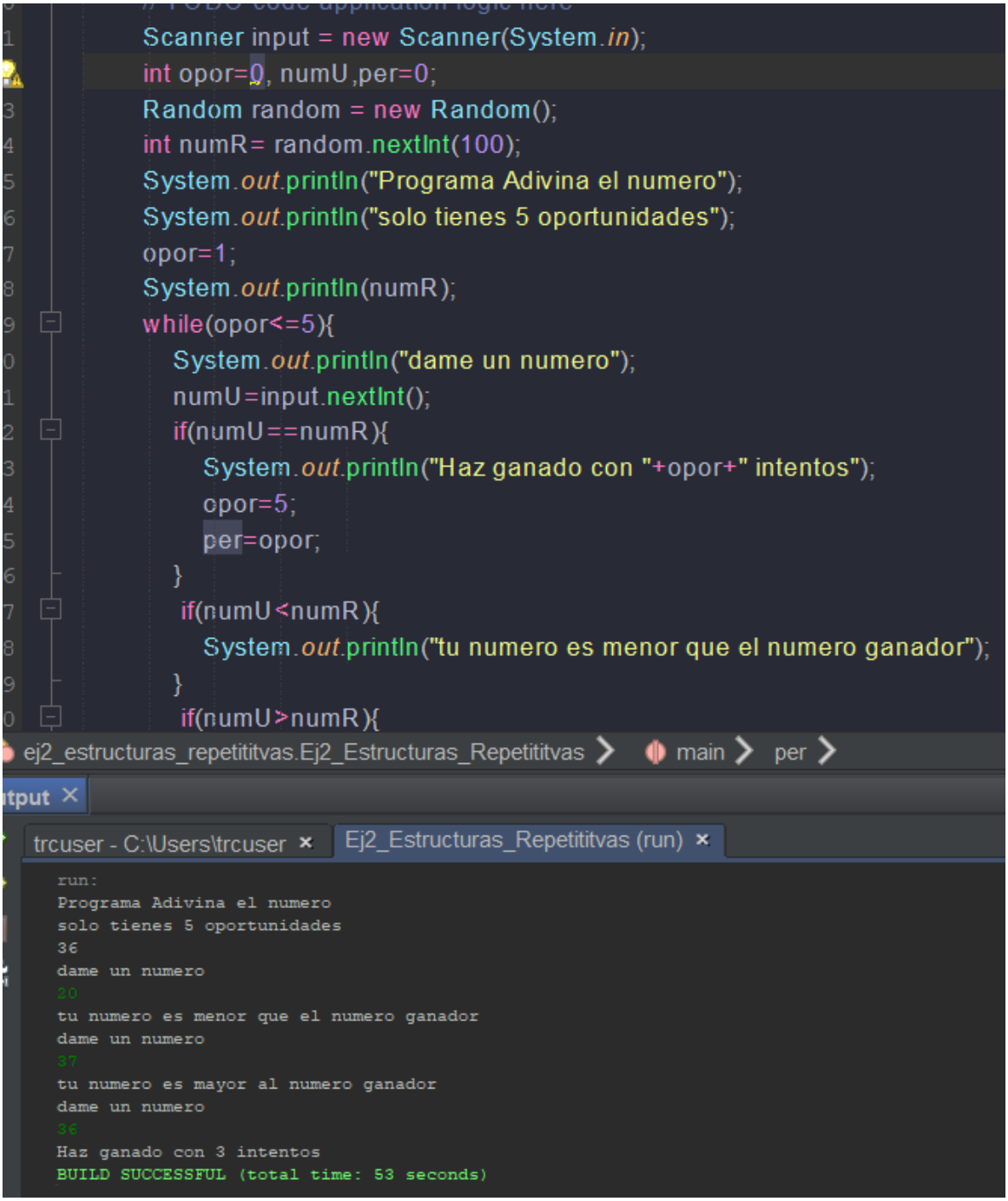
Ejemplos Sencillos de cada Estructura Repetitiva

1.- En una clase realice una sentencia repetitiva que pida al usuario 5 números y muestre la suma de estos números.

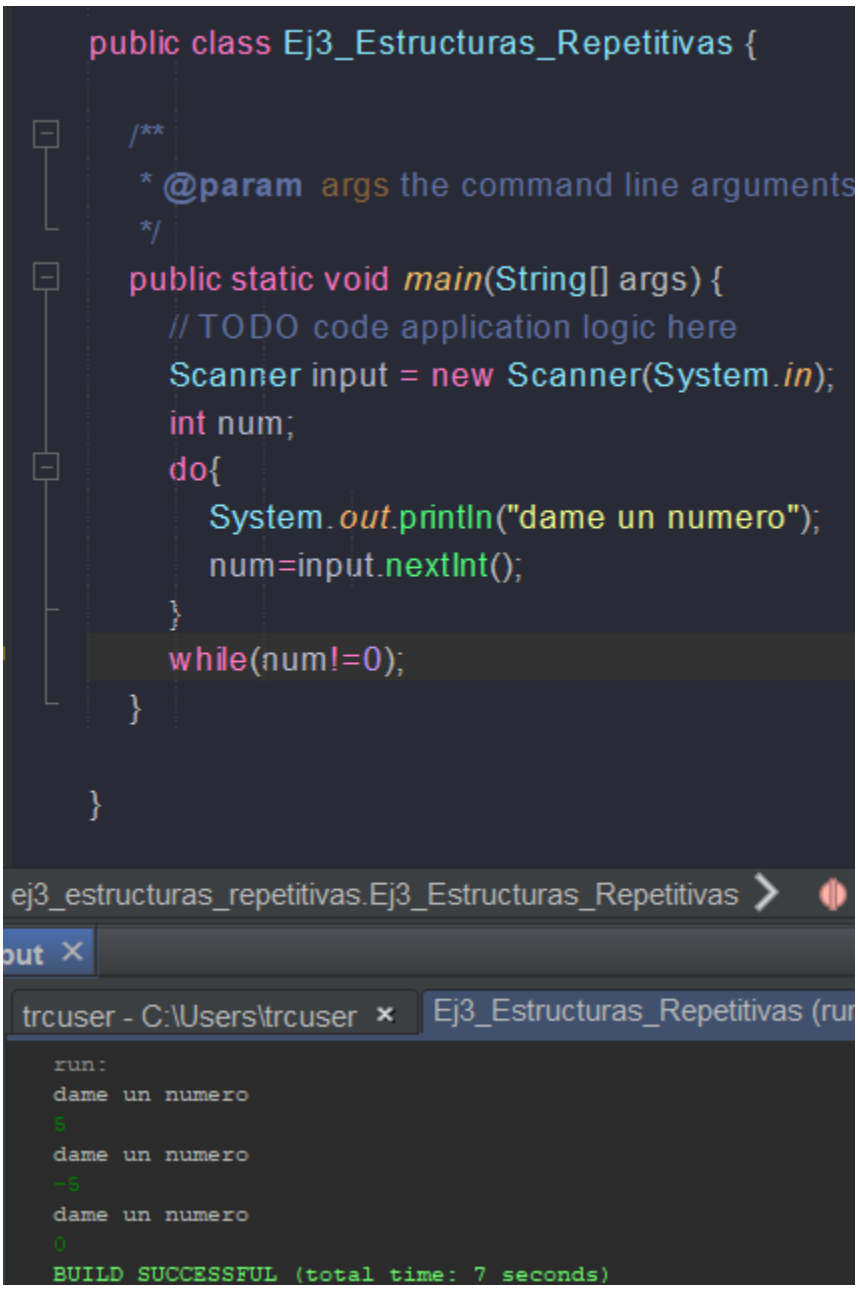
```
public class Ej1_Estructuras_Repetitivas {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        double num;
        double res=0;
        Scanner input = new Scanner(System.in);
        for( int i=0; i<=4;i++){
            System.out.println("dame un numero");
            num=input.nextDouble();
            res=num+i;
        }
        System.out.println("tu resultado es "+res);
    }
```

```
trcuser - C:\Users\trcuser ×  Ej1_Estructuras_Repetitivas (run)
run:
dame un numero
1
dame un numero
1
dame un numero
1
dame un numero
1
dame un numero
1
tu resultado es 5.0
BUILD SUCCESSFUL (total time: 5 seconds)
```

2.- El programa de adivinar el numero da instrucción y advierte al usuario que solo tiene 5 oportunidades, pide un numero al usuario y este va comparando con el numero aleatorio que escogió, le da pistas si es mayor o menor el número que ingreso el usuario, también lleva un contador de cada oportunidad que tiene para asi decirle en caso de que gane en que oportunidad de las 5 gano, OJO si se pasa de 5 oportunidades ahí termina el programa, ya en caso de que gane se felicita al usuario.



3.- Programa 1 ciclo do-while Hacer un programa que lea consecutivamente números enteros, positivos o negativos, el ciclo terminará cuando se lea un cero (0)



Conclusión

Las estructuras repetitivas son necesarias en el mundo de la programación y en la vida diaria las aplicamos en cosas muy cotidianas como cuando vamos a un super a hacer mandando nos cobran la cuenta de muchos productos asi los cajeros con cada cliente, que las cuentan son larguísimas seria muy complicado realizar código pidiendo lo mismo infinidad de veces, para eso se crearon las estructuras repetitivas que nos facilita ese tipo de programas