Introducción a la Algorítmica y Programación (3300)

Prof. Ariel Ferreira Szpiniak - aferreira@exa.unrc.edu.ar
Departamento de Computación
Facultad de Cs. Exactas, Fco-Qcas y Naturales
Universidad Nacional de Río Cuarto

Teoría 6

Composición Iterativa

(Repetición, Ciclos)

Novedades

UNRC

- El 1º de mayo la **UNRC** cumple 41 años, fue creada en **1971** por Decreto del P.E.N. dentro de un programa de adecuación de la enseñanza universitaria argentina a las necesidades del desarrollo y como respuesta a un fuerte movimiento social tanto local como regional que permitió la más grande conquista cultural de la región.
- Su creación fue un hito trascendente en el que participaron todos los sectores sociales de la comunidad local y regional con esfuerzo tenaz.
- Nuestra **Facultad** fue creada en **1975**, hace 37 años, y las **carreras de Computación** en **1992**.

Novedades

Escudo

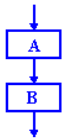
Significación del lema

CREER... CREAR... CRECER...
Representa la vocación de la juventud en devenir hombres y mujeres que en libertad y dignidad aspiran al desarrollo nacional.
CREER que es posible transformar para **CREAR** nuevas opciones para que todos podamos **CRECER**.
Son caminos a través de los cuales se manifiesta la UNRC, definiendo al mismo tiempo su relación con el hombre, con el mundo y con su época.

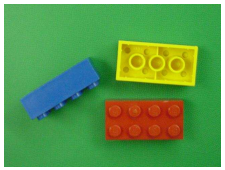
CREER... CREAR... CRECER

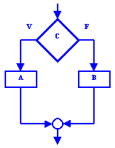
La UNRC está indisolublemente ligada a la región, en base a la cual define sus dimensiones, el ritmo de su expansión, su labor investigativa y vuelca su capacidad de innovación para contribuir al desarrollo integral.

Tipos de Composición de Algoritmos

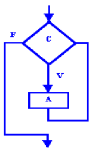


- Composición Secuencial





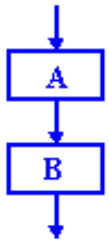
- Composición Condicional (Decisión, Selección)



- Composición Iterativa (Repetición)

Composición Secuencial

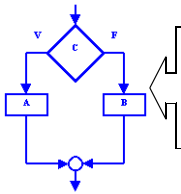
Es el tipo de composición más simple, está representada por una ***sucesión de acciones u operaciones*** (por ej. asignaciones), que se realizan una después de la otra, es decir, que el orden de ejecución coincide con el orden físico de aparición de las mismas, es decir, de arriba hacia abajo.



Las cajas A y B pueden ser definidas para ejecutar desde una simple acción hasta un módulo o algoritmo completo.

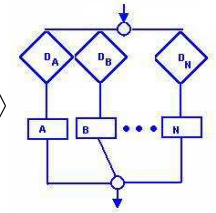
Composición Condicional

En un algoritmo representativo de un problema real es casi imposible que todo sea secuencial. Es necesario tomar ***decisiones*** en función de los datos del problema. La toma de decisión puede ser entre ***dos*** *o* ***más*** alternativas.



C es una condición que se evalúa. **A** es la acción que se ejecuta cuando la evaluación de este predicado resulta **verdadera** y **B** es la acción ejecutada cuando es **falsa**.

D_A, D_B,..., D_N son decisiones a tomar. **A** es la acción que se ejecuta cuando la **D_A** es verdadera. **B** es la acción ejecutada cuando **D_B** es verdadera. **N** es la acción ejecutada cuando **D_N** es verdadera.



Composición Iterativa
Introducción

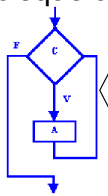
Hasta ahora hemos visto que un algoritmo está constituido estructuralmente por una serie de acciones organizadas de manera consecutiva (composición **secuencial**) o selectiva (composición **condicional**).

Sin embargo, en muchas ocasiones es necesario realizar las mismas acciones varias veces seguidas de manera consecutiva, ya sea una cantidad predeterminada de veces o no.

La composición **iterativa**, en todas sus alternativas, posibilita la ejecución repetida de un bloque de acciones.

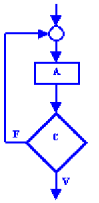
Composición Iterativa

En muchas ocasiones es necesario realizar una a varias acciones de manera repetida, ya sea una cantidad predeterminada de veces o no. La composición iterativa, en todas sus alternativas posibilita la ejecución repetida de un bloque de acciones. Veamos algunos ejemplos.



C es una condición que se evalúa. **A** es la acción o secuencia de acciones que se ejecuta cuando la evaluación de este predicado resulta **verdadera**. En caso de que la evaluación resulte **falsa**, **A** no se ejecuta más y **se continúa con la estructura siguiente**.

A es la acción o secuencia de acciones. **C** es una condición que se evalúa. Cuando la evaluación de este predicado resulta **falsa, A vuelve a ejecutarse**. Si la evaluación resulta **verdadera**, A no se ejecuta más y **se continúa con la estructura siguiente**.



Composición Iterativa
Introducción

- Analizaremos 4 formas diferentes de ejecutar acciones cíclicamente.

Mientras
Repetir
Iterar

Se aplican en casos donde se conoce o no la cantidad de veces que debemos ciclar

Para

Se aplica únicamente en el caso donde se conoce la cantidad de veces que debemos ciclar

Los lenguajes de programación generalmente implementan por lo menos una estructura de cada tipo.

Composición Iterativa
Introducción

La composición iterativa exige para su utilización **MUCHAS PRECAUCIONES**.

• Es preciso asegurarse a priori la coherencia en el encadenamiento sucesivo de las mismas acciones.

• Como veremos, éstas estructuras repetitivas presentan una **condición** que se debe satisfacer para continuar o para terminar. Un error en la formulación de esa condición puede producir a un fenómeno denominado **BUCLE o LOOP INFINITO** .

• Un **BUCLE** es la ejecución indefinida de una iteración.

La **iteración** también es llamada **ciclo**

Composición Iterativa
Introducción

Llamaremos iteración a toda repetición de la ejecución de una acción o de una secuencia de acciones.

Cuando se debe repetir una o varias acciones puede suceder alguna de las siguientes posibilidades:

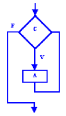
- No se conoce la cantidad exacta de iteraciones.
- Si se conoce la cantidad exacta de iteraciones.

Para cada uno de los casos existen composiciones iterativas que resultan más convenientes.

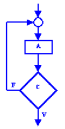
Composición Iterativa

Muchas veces no se conoce de antemano la cantidad de veces que hay que ejecutar una serie de acciones sino que solo se sabe bajo que condiciones se debe seguir ejecutando o cuando se debe finalizar la ejecución.
Para ese tipo de situaciones existen tres tipos de estructuras que permiten “ciclar” (ejecutar repetidamente) y parar solo cuando cierta condición es verdadera o es falsa.
Para este caso resultan adecuadas:

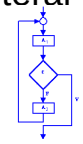
Mientras



Repetir

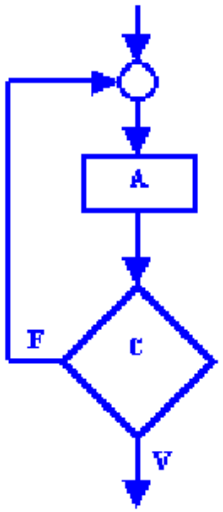


Iterar



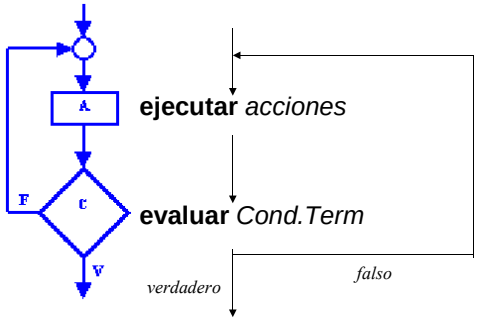
Composición Iterativa
Repetir

- El número de ejecuciones de las acciones no se expresa directamente.
- La *condición* expresa lo que se debe cumplir para que *termine* el ciclo. Por ello se denomina *condición de terminación*.
- Las *acciones* se ejecutan *una o más veces*.
- El comportamiento es el siguiente:
  - Se ejecutan las acciones una vez.
  - Luego se evalúa la condición de terminación.
  - Si la condición es falsa se vuelven a ejecutar las acciones.
  - Si la condición es verdadera termina (sale del ciclo).



Composición Iterativa
Repetir

En Notación Algorítmica la estructura **Repetir** se escribe de la siguiente manera:




```
repetir
  <acciones>
hasta que Cond.Term
```


Composición Iterativa
Repetir - Ejemplo

Problema: leer dos números enteros distintos por teclado y mostrarlos por pantalla en orden creciente.

```
Algoritmo OrdenCreciente
Lexico
  num1, num2 ∈ Z    {números a ordenar}
Inicio
  Escribir('Ingrese el primer número')
  Leer(num1)
  repetir
    Escribir('Ingrese otro diferente al primero')
    Leer(num2)
  hasta que num1<>num2
  si num1<num2
  entonces
    Escribir(num1)
    Escribir(num2)
  sino
    Escribir(num2)
    Escribir(num1)
  fsi
Fin
```


Composición Iterativa
Repetir - Ejemplos

Analizar los siguientes algoritmos y determinar que hacen.

```
Algoritmo BucleSioNo
Lexico
  i ∈ Z
Inicio
  i ← 1
  repetir
    escribir('Hola, soy un bucle')
  hasta que i≠i
  escribir('Terminé!!!')
Fin
```

¿cuántas veces hace el ciclo?

```
Algoritmo BucleSioNo2
Lexico
  i ∈ Z
Inicio
  i ← 2
  repetir
    escribir('Hola, soy otro bucle')
  hasta que i>9
  escribir('Terminé!!!')
Fin
```

¿cuántas veces hace el ciclo?

Composición Iterativa
Repetir - Ejemplos

Analizar los siguientes algoritmos y determinar que hacen.

```
Algoritmo MeCuré?
Lexico
  j ∈ Z
Inicio
  j ← 1
  repetir
    escribir('¿Cuántas veces repito')
    j ← j + 3
  hasta que j>0
  escribir('Terminé!!!')
Fin
```

¿cuántas veces hace el ciclo?

```
Algoritmo CuentaRegresiva
Lexico
  k ∈ Z
Inicio
  k ← 1
  repetir
    escribir('¿Salgo o no salgo?')
    k ← k + 1
  hasta que (k+5)>20
  escribir('Terminé!!!')
Fin
```

¿cuántas veces hace el ciclo?

Composición Iterativa
Repetir - Ejemplos

Analizar los siguientes algoritmos y determinar que hacen.

```
Algoritmo Mafalda
Lexico
  jj ∈ Z
Inicio
  jj ← 2
  escribir('Mi dicho preferido es:')
  repetir
    escribir('Paren el mundo ')
    escribir('me quiero bajar. ')
  hasta que jj>=1
  escribir('¿Les gustó?')
Fin
```

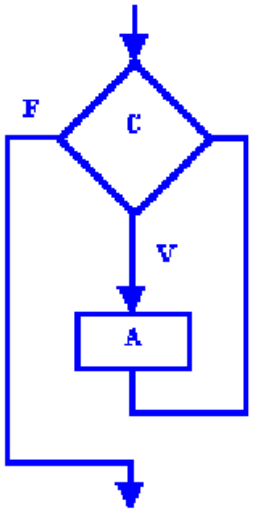
¿cuántas veces hace el ciclo?

```
Algoritmo CuentaRegresiva
Lexico
  i ∈ Z
Inicio
  i ← 10
  escribir('Iniciando conteo...')
  repetir
    escribir(i)
    i ← i - 1
  hasta que i<=0
  escribir('Despegue')
Fin
```

¿cuántas veces hace el ciclo?

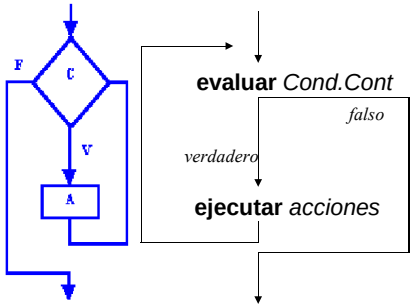
Composición Iterativa
Mientras

- El número de ejecuciones de las acciones no se expresa directamente.
- La *condición* expresa lo que se debe cumplir para que *continúe* el ciclo. Por ello se denomina *condición de continuación*.
- Las *acciones* se ejecutan *cero, una o más veces*.
- El comportamiento es el siguiente:
  - Se evalúa la condición de continuación.
  - Si la condición es verdadera, entra al ciclo y se ejecutan las acciones una vez.
  - Si la condición es falsa termina, es decir, sale del ciclo.
  - Si la condición es verdadera se vuelven a ejecutar las acciones.



Composición Iterativa
Mientras

En Notación Algorítmica la estructura **Mientras** se escribe de la siguiente manera:



mientras *Cond.Cont* **hacer**

<acciones>

fmientras

Composición Iterativa
Pascal - Mientras - Ejemplo

Problema: leer dos números enteros distintos por teclado y mostrarlos por pantalla en orden creciente.

```
Algoritmo OrdenCreciente
Lexico
  num1, num2 ∈ Z    {números a ordenar}
Inicio
  Escribir('Ingrese el primer número')
  Leer(num1)
  Escribir('Ingrese otro diferente al primero')
  Leer(num2)
  mientras num1=num2 hacer
    Escribir('Se necesita un número diferente de ', num1)
    Escribir('Vuelva a intentarlo')
    Leer(num2)
  fmientras
  si num1<num2
  entonces
    Escribir(num1)
    Escribir(num2)
  sino
    Escribir(num2)
    Escribir(num1)
  fsi
Fin
```


Composición Iterativa
Mientras - Ejemplos

Analizar los siguientes algoritmos y determinar que hacen. Compare semejanzas y diferencias con los ejemplos dados para el **Repetir**.

```
Algoritmo BucleSioNo
Lexico
  i ∈ Z
Inicio
  i ← 1
  mientras i=i hacer
    escribir('Hola, soy un bucle')
  fmientras
  escribir('Terminé!!!')
Fin
```

¿cuántas veces hace el ciclo?

```
Algoritmo BucleSioNo2
Lexico
  i ∈ Z
Inicio
  i ← 2
  mientras i<=9 hacer
    escribir('Hola, soy otro bucle')
  fmientras
  escribir('Terminé!!!')
Fin
```

¿cuántas veces hace el ciclo?

Composición Iterativa
Mientras - Ejemplos

Analizar los siguientes algoritmos y determinar que hacen. Compare semejanzas y diferencias con los ejemplos dados para el **Repetir**.

```
Algoritmo MeCuré?
Lexico
  j ∈ Z
Inicio
  j ← 1
  mientras j<=0 hacer
    escribir('¿Cuántas veces ciclo?')
    j ← j + 3
  fmientras
  escribir('Terminé!!!')
Fin
```

¿cuántas veces hace el ciclo?

```
Algoritmo CuentaRegresiva
Lexico
  k ∈ Z
Inicio
  k ← 1
  mientras (k+5)>20 hacer
    escribir('¿Salgo o no salgo?')
    k ← k + 1
  fmientras
  escribir('Terminé!!!')
Fin
```

¿cuántas veces hace el ciclo?

Composición Iterativa
Mientras - Ejemplos

Analizar los siguientes algoritmos y determinar que hacen. Compare semejanzas y diferencias con los ejemplos dados para el **Repetir**.

```
Algoritmo Mafalda
Léxico
  jj ∈ Z
Inicio
  jj ← 2
  escribir('Mi dicho preferido es:')
  mientras jj<1 hacer
    escribir('Paren el mundo ')
    escribir('me quiero bajar. ')
  fmientras
  escribir('¿Les gustó?')
Fin
```

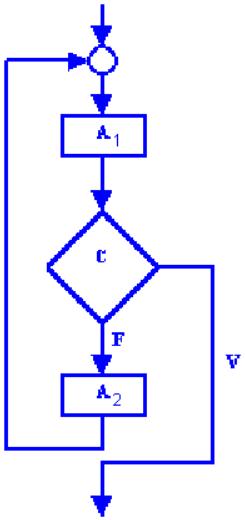
¿cuántas veces hace el ciclo?

```
Algoritmo CuentaRegresiva
Léxico
  i ∈ Z
Inicio
  i ← 10
  escribir('Iniciando conteo...')
  mientras i>=1 hacer
    escribir(i)
    i ← i - 1
  fmientras
  escribir('Despegue')
Fin
```

¿cuántas veces hace el ciclo?

Composición Iterativa
Iterar

- El número de ejecuciones de las acciones no se expresa directamente.
- La *condición* indica expresa lo que se debe cumplir para que *termine o pare* la iteración. Por ello se denomina *condición de parada*.
- Las *acciones* A1 se ejecutan *una o más veces*, pero las A2 se ejecutan *cero o más veces*.
- El comportamiento es el siguiente:
  - Se ejecutan las acciones A1 una vez.
  - Se evalúa la condición de parada.
  - Si la condición es falsa se ejecutan las acciones A2 una vez y vuelven a ejecutar las acciones A1 otra vez.
  - Si la condición es verdadera termina, decir, sale del ciclo.



Composición Iterativa

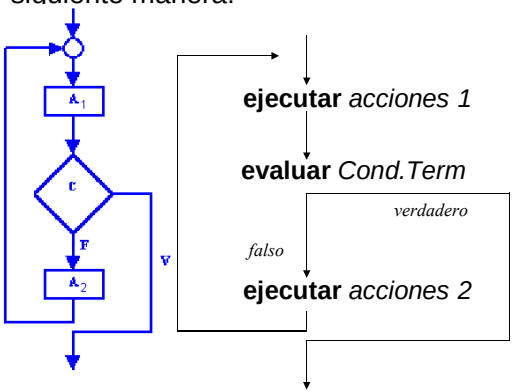
Iterar es la estructura más general, es decir, incluye a las dos primeras. Es una especie de "**mixtura**" entre la estrutura mientras y repetir.

Sin embargo *es la que menos* se usa debido, a por lo menos, dos razones:

- La mayoría de los lenguajes de programación no la soportan. O si lo hacen es a costa de "ensuciar" el código con principios "desterrados" de la programación estructurada como el **goto**. En otros casos también se use el **break**.
- Todos los lenguajes modernos soportan mientras y/o repetir.

Composición Iterativa
Iterar

En Notación Algorítmica la estructura **Mientras** se escribe de la siguiente manera:




```
iterar
  <acciones1>
  parada Cond.Term.
  <acciones2>
fiterar
```


Composición Iterativa
Pascal - Iterar - Ejemplo

Problema: leer dos números enteros distintos por teclado y mostrarlos por pantalla en orden creciente.

```
Algoritmo OrdenCreciente
Lexico
  num1, num2 ∈ Z    {números a ordenar}
Inicio
  Escribir('Ingrese el primer número')
  Leer(num1)
  Escribir('Ingrese otro diferente al primero')
  iterar
    Leer(num2)
    parada num1<>num2
    Escribir('Se necesita un número diferente de ', num1)
    Escribir('Vuelva a intentarlo')
  fiterar
  si num1<num2
  entonces
    Escribir(num1)
    Escribir(num2)
  sino
    Escribir(num2)
    Escribir(num1)
  fsi
Fin
```

Composición Iterativa
Iterar - Ejemplos

Analizar los algoritmos y determinar que hacen. Compare semejanzas y diferencias con los ejemplos para el **Mientras** y el **Repetir**.

```
Algoritmo BucleSioNo
Lexico
  i ∈ Z
Inicio
  i ← 1
  iterar
    escribir('Hola, soy un bucle')
    parada i≠i
    escribir('Hola, sigo aquí')
  fiterar
  escribir('Terminé!!!')
Fin
```

¿cuántas veces hace el ciclo?

```
Algoritmo BucleSioNo2
Lexico
  i ∈ Z
Inicio
  i ← 2
  iterar
    escribir('Hola, soy otro bucle')
    parada i>9
    escribir('Hola, sigo aquí')
  fiterar
  escribir('Terminé!!!')
Fin
```

¿cuántas veces hace el ciclo?

Composición Iterativa
Iterar - Ejemplos

Analizar los algoritmos y determinar que hacen. Compare semejanzas y diferencias con los ejemplos para el **Mientras** y el **Repetir**.

```
Algoritmo MeCuré?
Lexico
  j ∈ Z
Inicio
  j ← 1
  iterar
    escribir('¿Cuántas veces ciclo?')
    j ← j + 3
    parada j>0
    j ← j - 2
  fiterar
  escribir('Terminé!!!')
Fin
```

¿cuántas veces hace el ciclo?

```
Algoritmo MarchaAtras
Lexico
  k ∈ Z
Inicio
  k ← 1
  iterar
    escribir('¿Salgo o no salgo?')
    k ← k + 1
    parada (k+5)>20
    nada
  fiterar
  escribir('Terminé!!!')
Fin
```

¿cuántas veces hace el ciclo?

Composición Iterativa
Iterar - Ejemplos

Analizar los algoritmos y determinar que hacen. Compare semejanzas y diferencias con los ejemplos para el **Mientras** y el **Repetir**.

```
Algoritmo Mafalda
Lexico
  jj ∈ Z
Inicio
  jj ← 2
  escribir('Mi dicho preferido es:')
  iterar
    nada
    parada jj>=1
    escribir('Paren el mundo ')
    escribir('me quiero bajar. ')
  fiterar
  escribir('¿Les gustó?')
Fin
```

¿cuántas veces hace el ciclo?

```
Algoritmo CuentaRegresiva
Lexico
  i ∈ Z
Inicio
  i ← 10
  escribir('Iniciando conteo...')
  iterar
    escribir(i)
    i ← i - 1
    parada i<=0
    escribir(i)
    i ← i - 1
  fiterar
  escribir('Despegue')
Fin
```

¿cuántas veces hace el ciclo?

Composición Iterativa
Para

para **i** **desde** vi **hasta** vf **paso** **k** **hacer**
  <acciones>
fpara

- El número de ejecuciones de las acciones se expresa directamente.
- Las *acciones* se ejecutan *cero, una o más veces*.
- El paso *k es un número entero* (puede ser positivo o negativo).
- La variable *i* es denominada variable de control. Toma valores entre *vi (valor inicial)* y *vf (valor final)*.

Composición Iterativa
Para

```
para i desde vi hasta vf paso k hacer
  <acciones>
fpara
```

- El comportamiento es el siguiente:
  - **Si el paso es positivo**:
    - Se evalúa el valor de *i.* Si *i* es menor o igual que *vf* se ejecutan las acciones. Luego se incrementa el valor de *i* según el paso *k*.
    - Si *i* es mayor que *vf* termina (sale del ciclo).
  - **Si el paso es negativo**:
    - Se evalúa el valor de *i.* Si *i* es mayor o igual que *vf* se ejecutan las acciones. Luego se decrementa el valor de *i* según el paso *k*.
    - Si *i* es menor que *vf* termina (sale del ciclo).

Composición Iterativa
Para - Ejemplo

Problema: realizar y mostrar la sumatoria de los ***n*** primeros números enteros. El valor ***n*** es ingresado por teclado por el usuario.

```
Algoritmo SumatoriaEnteros
Lexico
  cotaSup ∈ Z     {cantidad de iteraciones del para}
  i ∈ Z           {variable de control del para}
  s ∈ Z           {sumatoria}
Inicio
  Escribir('Ingrese el valor de n')
  Leer(cotaSup)
  s ← 0
  para i desde 0 hasta cotaSup paso 1 hacer
    s ← s + i
  fpara
  Escribir('La suma de los ', cotaSup, 'primeros
           números enteros es: ', s)
Fin
```


Composición Iterativa
Para - Ejemplo

Analizar los siguientes algoritmos y determinar que hacen.

```
Algoritmo ContarHasta10
Lexico
  i ∈ Z
Inicio
  para i desde 1 hasta 10 paso 1 hacer
    escribir('Hola, soy el ', i)
  fpara
  escribir('Terminé!!!')
Fin
```

```
Algoritmo Pares
Lexico
  j ∈ Z
Inicio
  para j desde 2 hasta 80 paso 2 hacer
    escribir('Hola, soy el ', j)
  fpara
  escribir('Terminé!!!')
Fin
```


Composición Iterativa
Para - Ejemplo

Analizar los siguientes algoritmos y determinar que hacen.

```
Algoritmo CuentaRegresiva
Lexico
  i ∈ Z
Inicio
  escribir('Iniciando conteo...')
  para i desde 10 hasta 1 paso -1 hacer
    escribir(i)
  fpara
  escribir('Despegue')
Fin
```

```
Algoritmo CuentaRegresiva
Lexico
  i ∈ Z
Inicio
  escribir('Iniciando conteo...')
  para i desde 10 hasta 0 paso -1 hacer
    escribir(i)
  fpara
  escribir('Despegue')
Fin
```

Composición Iterativa
Para - Ejemplo

Analizar los siguientes algoritmos y determinar que hacen.

```
Algoritmo ParesDesde0
Lexico
  j ∈ Z
Inicio
  para j desde 0 hasta 80 paso 2 hacer
    escribir('Hola, soy el ', j)
  fpara
  escribir('Terminé!!!')
Fin
```

```
Algoritmo BucleSioNo
Lexico
  k ∈ Z
Inicio
  escribir('Hola!!!!')
  para k desde 1 hasta 10 paso -1 hacer
    escribir('Soy un bucle')
  fpara
  escribir('Terminé!!!')
Fin
```


Composición Iterativa
Para - Ejemplo

Analizar los siguientes algoritmos y determinar que hacen.

```
Algoritmo BucleSioNo2
Lexico
  k ∈ Z
Inicio
  escribir('Hola!!!!')
  para k desde 1 hasta 10 paso -2 hacer
    escribir('Soy un bucle')
  fpara
  escribir('Terminé!!!')
Fin
```

```
Algoritmo BucleSioNo3
Lexico
  k ∈ Z
Inicio
  escribir('Hola!!!!')
  para k desde 5 hasta 4 paso 1 hacer
    escribir('Soy un bucle')
  fpara
  escribir('Terminé!!!')
Fin
```

2

Ejercicios

Ahora que ya conocemos las estructuras iterativas vamos a resolver los siguientes ejercicios, validando que los datos ingresados sean correctos.

- Desarrollar un algoritmo que solicite al usuario la base y altura de un triángulo, calcule, almacene y muestre el área del mismo.

- Desarrollar un algoritmo que siga los siguientes pasos:
  1. Pida al usuario un número y lo almacene en la variable num.
  2. A continuación le sume 1 unidad al número ingresado por el usuario (num) y lo muestre por pantalla con el siguiente mensaje: "Mi número es", num, "TE GANE!!!"
  3. A continuación le pregunte al usuario si quiere jugar otra vez (S / N).
  4. Si el usuario responde `S` vuelve al paso 1, sino termina.

Nota: use la estructura repetitiva que considere más adecuada.

Composición Iterativa
Pascal

La estructura **Repetir**, en **Pascal**, se escribe de la siguiente manera:

Notación Algorítmica

```
repetir

  <acciones>

hasta que Cond.Term
```

PASCAL

```
REPEAT

  <acciones>

UNTIL Cond.Term;
```

Composición Iterativa
Pascal - Repetir - Ejemplo

Problema: leer dos números enteros distintos por teclado y mostrarlos por pantalla en orden creciente.

```
Algoritmo OrdenCreciente
Lexico
  num1, num2 ∈ Z    {números a ordenar}
Inicio
  Escribir('Ingrese el primer número')
  Leer(num1)
  repetir
    Escribir('Ingrese otro diferente al primero')
    Leer(num2)
  hasta que num1<>num2
  si num1<num2
  entonces
    Escribir(num1)
    Escribir(num2)
  sino
    Escribir(num2)
    Escribir(num1)
  fsi
Fin
```


Composición Iterativa
Pascal - Repetir - Ejemplo

```
PROGRAM OrdenCreciente;
VAR
  num1, num2: Integer;  {números a ordenar}
BEGIN
  WRITELN('Ingrese el primer número');
  READLN(num1);
  REPEAT
    WRITELN('Ingrese otro diferente al primero');
    READLN(num2)
  UNTIL num1<>num2;
  IF num1<num2
  THEN EGIN
    WRITELN(num1);
    WRITELN(num2)
  END
  ELSE BEGIN
    WRITELN(num2);
    WRITELN(num1)
  END
END.
```


Composición Iterativa
Pascal

La estructura **Mientras**, en **Pascal**, se escribe de la siguiente manera:

Notación Algorítmica	PASCAL
mientras *Cond.Cont* **hacer** *<acciones>* **fmientras**	**WHILE** *Cond.Cont* **DO BEGIN** *<acciones>* **END;**

Composición Iterativa
Pascal - Mientras - Ejemplo

Problema: leer dos números enteros distintos por teclado y mostrarlos por pantalla en orden creciente.

```
Algoritmo OrdenCreciente
Lexico
  num1, num2 ∈ Z    {números a ordenar}
Inicio
  Escribir('Ingrese el primer número')
  Leer(num1)
  Escribir('Ingrese otro diferente al primero')
  Leer(num2)
  mientras num1=num2 hacer
    Escribir('Se necesita un número diferente de ', num1)
    Escribir('Vuelva a intentarlo')
    Leer(num2)
  fmientras
  si num1<num2
  entonces
    Escribir(num1)
    Escribir(num2)
  sino
    Escribir(num2)
    Escribir(num1)
  fsi
Fin
```

Composición Iterativa
Pascal - Mientras - Ejemplo

```
PROGRAM OrdenCreciente;
VAR
  num1, num2: Integer;  {números a ordenar}
BEGIN
  WRITELN('Ingrese el primer número');
  READLN(num1);
  WRITELN('Ingrese otro diferente al primero');
  READLN(num2);
  WHILE num1=num2 DO BEGIN
    WRITELN('Se necesita un número diferente de ', num1);
    WRITELN('Vuelva a intentarlo');
    READLN(num2)
  END;
  IF num1<num2
  THEN BEGIN
    WRITELN (num1);
    WRITELN(num2)
  END
  ELSE BEGIN
    WRITELN (num2);
    WRITELN(num1)
  END
END.
```


Composición Iterativa
Pascal

La estructura **Iterar**, en **Pascal**, *se puede escribir* de la siguiente manera:

Notación Algorítmica	**PASCAL**
iterar *<acciones1>* **parada** *Cond.Term.* *<acciones2>* **fiterar**	*<acciones1>* **WHILE** NOT *Cond.Term* **DO BEGIN** *<acciones2>* *<acciones1>* **END;**

Composición Iterativa
Pascal - Iterar - Ejemplo

Problema: leer dos números enteros distintos por teclado y mostrarlos por pantalla en orden creciente.

```
Algoritmo OrdenCreciente
Lexico
  num1, num2 ∈ Z    {números a ordenar}
Inicio
  Escribir('Ingrese el primer número')
  Leer(num1)
  Escribir('Ingrese otro diferente al primero')
  iterar
    Leer(num2)
    parada num1<>num2
    Escribir('Se necesita un número diferente de ', num1)
    Escribir('Vuelva a intentarlo')
  fiterar
  si num1<num2
  entonces
    Escribir(num1)
    Escribir(num2)
  sino
    Escribir(num2)
    Escribir(num1)
  fsi
Fin
```


Composición Iterativa
Pascal - Iterar - Ejemplo

```
PROGRAM OrdenCreciente;
VAR
  num1, num2: Integer;  {números a ordenar}
BEGIN
  WRITELN('Ingrese el primer número');
  READLN(num1);
  WRITELN('Ingrese otro diferente al primero');
  READLN(num2);
  WHILE NOT(num1<>num2) DO BEGIN
    WRITELN('Se necesita un número diferente de ', num1);
    WRITELN('Vuelva a intentarlo');
    READLN(num2)
  END;
  IF num1<num2
  THEN BEGIN
    WRITELN(num1);
    WRITELN(num2)
  END
  ELSE BEGIN
    WRITELN(num2);
    WRITELN(num1)
  END
END.
```

Pascal no implementa el **ITERAR**, por lo tanto si se desea usar este tipo de construcción, hay que simularlo.

Como podemos observar, el programa es igual al del ejemplo del **WHILE.**

Composición Iterativa
Pascal

La estructura **Para**, en **Pascal**, se escribe de la siguiente manera:

Notación Algorítmica

```
para i desde vi hasta vf paso k hacer
  <acciones>
fpara
```

PASCAL

```
FOR i := vi TO vf DO
BEGIN
  <acciones>
END;
```

¡OJO! En Pascal solo existe el paso 1 y –1. Para indicar el paso 1 se coloca la palabra TO y para el –1 DOWNTO.

DOWNTO

Composición Iterativa
Pascal - Para - Ejemplo

Problema: realizar y mostrar la sumatoria de los ***n*** primeros números enteros. El valor ***n*** es ingresado por teclado por el usuario.

```
Algoritmo SumatoriaEnteros
Lexico
  cotaSup ∈ Z     {cantidad de iteraciones del para}
  i ∈ Z           {variable de control del para}
  s ∈ Z           {sumatoria}
Inicio
  Escribir('Ingrese el valor de n')
  Leer(cotaSup)
  s ← 0
  para i desde 0 hasta cotaSup paso 1 hacer
    s ← s + i
  fpara
  Escribir('La suma de los ', cotaSup, 'primeros
           números enteros es: ', s)
Fin
```


Composición Iterativa
Pascal - Para - Ejemplo

```
PROGRAM SumatoriaEnteros
VAR
  cotaSup: INTEGER;     {cantidad de iteraciones del para}
  i: INTEGER;           {variable de control del para}
  s: INTEGER;           {sumatoria}
BEGIN
  WRITELN('Ingrese el valor de n');
  READLN(cotaSup);
  s := 0;
  FOR i:= 0 TO cotaSup DO BEGIN
    s := s + i
  END;
  WRITELN('La suma de los ', cotasup, 'primeros
           números enteros es: ', s)
END;
```

Si el paso es distinto a 1 o –1 hay que buscar otra alternativa como por ejemplo utilizar un WHILE.

Composición Iterativa
Equivalencias

repetir <acciones> **hasta que** cond.term	≡	<acciones> **mientras** ¬cond.term **hacer** <acciones> **fmientras**
iterar <acciones1> **parada** cond.term <acciones2> **fiterar**	≡	<acciones1> **mientras** ¬cond.term **hacer** <acciones2> <acciones1> **fmientras**
para i **desde** a **hasta** n **paso** k **hacer** <acciones> **fpara**	≡	i ← a **mientras** i≤n **hacer** <acciones> i ← i+k **fmientras**

Composición Iterativa
Equivalencias (cont.)

mientras cond.cont **hacer** acciones **fmientras**	≡	**iterar** **parada** ¬cond.cont acciones **fiterar**
repetir acciones **hasta que** cond.term	≡	**iterar** acciones **parada** cond.term **fiterar**
para i **desde** a **hasta** n **paso** k **hacer** acciones **fpara**	≡	i ← a **iterar** **parada** i>n acciones i ← i+k **fiterar**

Composición Iterativa
Riesgos - Ejemplos

Escribir un algoritmo que dado un **cesto** de papas permita **pelar** un número suficiente de **papas** para hacer un rico puré. Debe tenerse en cuenta que el cesto de papas puede estar vacío en un momento dado y que es posible volver a llenarlo.

Versión 1
mientras cesto no vacío **hacer**
pelar una papa
pelar una papa
fmientras

Versión 2
repetir
pelar una papa
pelar una papa
hasta que cesto no vacío

Versión 3
repetir
pelar una papa
pelar una papa
hasta que cesto vacío

Versión 4
mientras número de papas insuficiente **hacer**
si cesto vacío
entonces
llenar cesto
fsi
pelar una papa
fmientras

Versión 5
mientras número de papas insuficiente **hacer**
si cesto no vacío
entonces
pelar una papa
sino
llenar cesto
fsi
fmientras

Composición Iterativa
Riesgos - Ejemplos

Versión 6
mientras nro de papas insuficiente **hacer**
si cesto no vacío
entonces
pelar una papa
fsi
fmientras

Versión 7
repetir
si cesto no vacío
entonces
pelar una papa
fsi
hasta que nro de papas suficiente

Versión 8
mientras cesto lleno **hacer**
si nro de papas insuficiente
entonces
pelar una papa
sino
llenar cesto
pelar una papa
fsi
fmientras

Versión 9
mientras nro de papas insuficiente o cesto lleno **hacer**
pelar una papa
fmientras

Versión 10
repetir
pelar una papa
hasta que nro de papas insuficiente o cesto vacío

Bibliografía

– Biondi, J. y Clavel, G. “Introducción a la Programación. Tomo 1: Algorítmica y Lenguajes”. (pags. 70 - 122)
– Wirth, N. “Introducción a la Programación Sistemática”. Ateneo. (pags. 27 - 44)
– Quetglás, Toledo, Cerverón. "Fundamentos de Informática y Programación“. Capítulo 3. http://robotica.uv.es/Libro/Indice.html
• Estructuras Repetitivas (pags 98 – 110)
– Watt, David: Programming Language Concepts and Paradigms, Prentice-Hall International Series in Computer Science (1990). Cap 10: The Imperative Programming Paradigm.
– Scholl, P. y Peyrin, J.-P. “Esquemas Algorítmicos Fundamentales: Secuencias e iteración”. (pags 107 – 128). Aparecen los ciclos en forma simultánea con el concepto de secuencias (este concepto lo veremos más adelante).

– Pascal
• introturbopascal.pdf (aula virtual)
• laprogramacionenlenguajepascal.pdf (aula virtual)
• pascalyturbopascal.pdf (aula virtual)
• Biondi, J. y Clavel, G. “Introducción a la Programación. Tomo 1: Algorítmica y Lenguajes”: (pags. 243 - 252)
• Joyanes Aguilar, L., “Programación en Turbo Pascal”. Mc Graw Hill, 1993.
• Wirth, N. and K. Jensen, “Pascal: User Manual and Report”, 4°ed., New York, Springer-Verlag, 1991 (traducción de la primera edición “Pascal: Manual del Usuario e Informe”, Buenos Aires, El Ateneo, 1984).

Introducción a la Algorítmica y Programación (3300)

Prof. Ariel Ferreira Szpiniak - aferreira@exa.unrc.edu.ar
Departamento de Computación
Facultad de Cs. Exactas, Fco-Qcas y Naturales
Universidad Nacional de Río Cuarto

Teoría 6 (continuación)

Composición Iterativa

(Repetición, Ciclos)

Ciclos en C - mientras

```
//PROGRAM OrdenCreciente
#include <stdio.h>
int  num1, num2;      //números a ordenar}
void main(){
  printf("Ingrese el primer número");
  scanf("%d",&num1);
  printf("Ingrese otro diferente al primero");
  scanf("%d",&num2);
  while (num1==num2){
    printf("Se necesita un número diferente de %d", num1);
    printf("Vuelva a intentarlo");
    scanf("%d",&num2);
  }
  if (num1<num2){
    printf("Primer numero: %d",num1);
    printf("Segundo número: %d",num2);
  }
  else{
    printf("Segundo numero: %d",num2);
    printf("Primer numero: %d",num1);
  }
}
```

Ciclos en C - repetir

```
//PROGRAM OrdenCreciente
#include <stdio.h>
int  num1, num2; //números a ordenar
void main(){
  printf("Ingrese el primer número");
  scanf("%d",&num1);
  do {
    printf("Ingrese otro diferente al primero");
    scanf("%d",&num2);
  } while (num1==num2);
  if (num1<num2){
    printf("Primer numero: %d",num1);
    printf("Segundo número: %d",num2);
  }
  else{
    printf("Segundo numero: %d",num2);
    printf("Primer numero: %d",num1);
  }
}
```

Ciclos en C - iterar

```
//PROGRAM OrdenCreciente
#include <stdio.h>
int  num1, num2;      //números a ordenar}
void main(){
  printf("Ingrese el primer número");
  scanf("%d",&num1);
  printf("Ingrese otro diferente al primero");
  scanf("%d",&num2);
  while (!(num1!=num2)){
    printf("Se necesita un número diferente de %d", num1);
    printf("Vuelva a intentarlo");
    scanf("%d",&num2);
  }
  if (num1<num2){
    printf("Primer numero: %d",num1);
    printf("Segundo número: %d",num2);
  }
  else{
    printf("Segundo numero: %d",num2);
    printf("Primer numero: %d",num1);
  }
}
```

Ciclos en C - para

```
//PROGRAM SumatoriaEnteros
#include <stdio.h>
int  cotaSup;       //cantidad de iteraciones del para
int  i;     //variable de control del para
int  s;     //sumatoria
void main(){
  printf("Ingrese el valor de n: \n");
  scanf("%d",&cotaSup);
  s = 0;
  int i;
  for (i=0; i<=cotaSup; i++){
    s = s + i;
  }
  printf("La suma de los %d primeros números enteros es: %d \n",
cotaSup, s);
}
```