PROGRAMACIÓN II – 2DO CUATRIMESTRE 2017

Profesor: Alejandro H. González

JTP: Silvana Lis Gallo

Ayudantes: Leandro Romanut, Lic. Juan Maulini - Sr. Mariano Sanchez - Sr. Leandro Libutti - Srta. Ruth Canaza

Horarios de teoría: Lunes de 8:00 a 10:00 hs. Aula 10_B

Miércoles de 8:00 a 10:00 hs. Aula 10B

Horarios de práctica: Lunes de10:00 a 12:00 hs. Aula 10_B y 7

Miércoles de 10:00 a 12:00 hs Aula 10_B y 7

PROGRAMACIÓN II -APROBACIÓN DE LA ASIGNATURA

Todos los alumnos entrarán al régimen de promoción (de carácter teórico-práctico)y deben cumplimentar las siguientes condiciones:

1. Se tomarán dos exámenes. El examen parcial 1 tiene los temas árboles y java, el examen parcial 2 tiene los temas componentes de la computadora y Assembler MX88.
2. Cada examen tiene su correspondiente recuperativo. En el recuperatorio se pueden rendir uno o los dos temas.
3. También hay un examen flotante donde se podrán rendir dos temas que hayan quedado sin aprobar de cualquiera de los 2 exámenes parciales anteriores.

PROGRAMACIÓN II -PROMOCION

4. Para acceder a la promoción será necesario aprobar los parciales prácticos.

5. Se tomarán además dos exámenes teóricos breves cada uno con su correspondiente recuperatorio, estos exámenes breves se consideran aprobado con nota 6 o superior. Estos exámenes serán corregidos una vez aprobada la parte práctica.

6. Se desarrollará durante el curso la entrega de una actividad grupal obligatoria con nota. El desarrollo del problema será un programa que funcione en computadora y se deberá aprobar también un coloquio referido al desarrollo realizado.

7. La nota final de la asignatura se compone de las notas de los exámenes teóricos breves aprobados y la nota de la actividad grupal con su correspondiente coloquio.

PROGRAMACIÓN II -APROBACIÓN DE LA ASIGNATURA

7. Los alumnos que obtienen una nota promedio de 6, o mayor a 6 tendrán aprobada la promoción.

8. Los alumnos que obtienen una nota promedio (de las 4 notas) de 4 o mayor a 4 y menor que 6 tendrán aprobada la cursada y deberán rendir examen final.

PROGRAMACIÓN II -APROBACIÓN DE LA ASIGNATURA

Para poder pasar la nota de promoción el alumno/a debe tener aprobada la materia correlativa correspondiente a Programación I.

La nota de promoción no será guardada llegado el caso que el alumno no posea la correlativa aprobada al momento de cerrar las notas en el sistema Siu-Guaraní.

PROGRAMACIÓN II -APROBACIÓN DE LA ASIGNATURA

Examen final – SOLO para los que no sacaron promoción

Los alumnos que no hayan aprobado el régimen de promoción, pero hayan aprobado los exámenes prácticos, obtendrán la aprobación de los Trabajos Prácticos y la habilitación para rendir el Examen Final de la asignatura.

El examen final es de carácter teórico-práctico.

PROGRAMACIÓN II – MATERIALES

Deben solicitar inscripción en el curso Programación II, donde el profesor es Alejandro González en el entorno virtual IDEAS desde https://ideas.info.unlp.edu.ar/login

También hay información de la cátedra y material de ayuda en el blog de Programación 2: http://blogs.unlp.edu.ar/programacion2/

PROGRAMACIÓN II - TEMAS DEL CUATRIMESTRE

Listas/repaso

Recursión

Arboles

Programación Orientada a Objetos (POO)

JAVA y POO

Organización de la Memoria

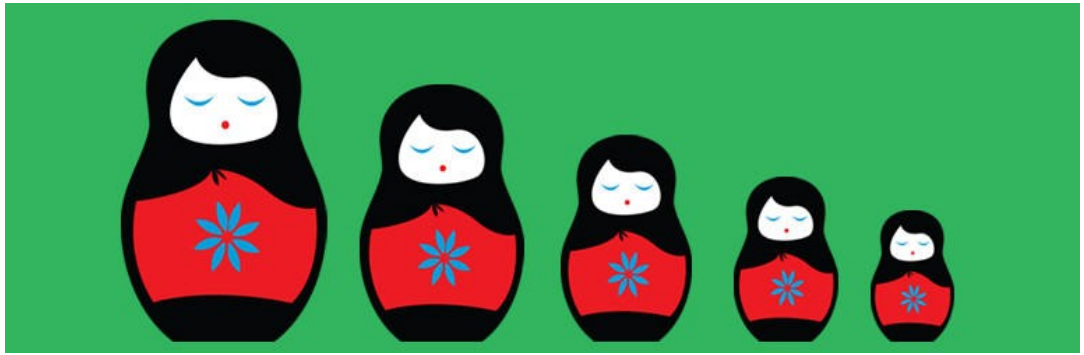
Subsistema de E/S: Periféricos

Simulador MSX88

Programación en Assembler

Conceptos de Programación Concurrente

RECURSION



TEMARIO - RECURSIÓN

- Concepto de Recursión
- Características de algoritmos recursivos
- Ejemplos

RECURSION

Metodologías para resolver problemas:

1. Diseño Top Down
2. Recursión
3. Abstracción de Datos
4. Diseño orientado a objetos

JUEGO

Se trata de un juego con discos de radio creciente que se apilan insertándose en una de las tres estacas de un tablero.

El objetivo del juego es crear la pila en otra de las estacas.

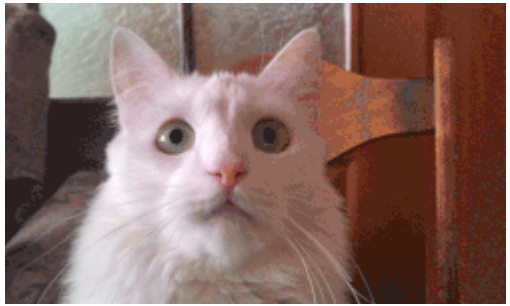
Las reglas son:

1. Sólo se puede mover un disco cada vez.

2. Un disco de mayor tamaño no puede descansar sobre uno más pequeño que él mismo.

3. Sólo se puede desplazar el disco que se encuentre arriba en cada varilla.

¿QUÉ ES LA RECURSIÓN?

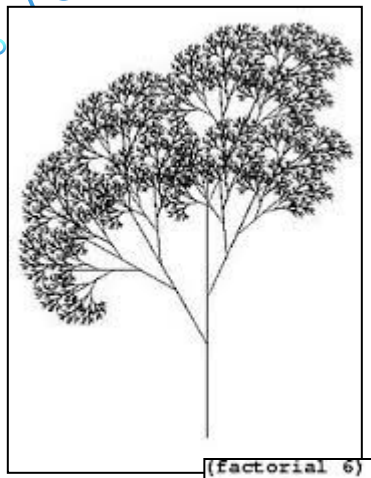


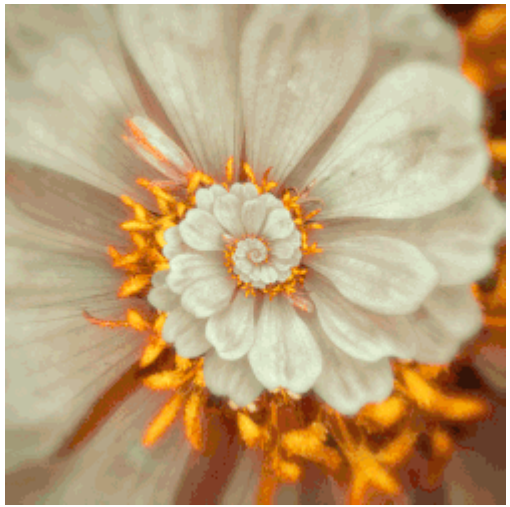
Vamos a mirar el siguiente video.
http://www.youtube.com/watch?v=aGlt2G-DC8c

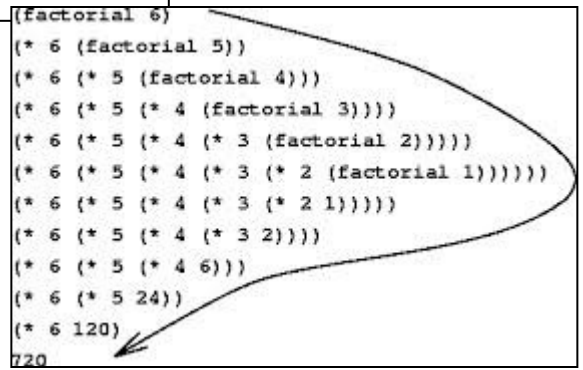
El mismo trata de la resolución de un problema conocido llamado “Las Torres de Hanoi”. Es un rompecabezas o juego matemático inventado en 1883 por el matemático francés Édouard Lucas..

¿CUÁLES SON LAS CARACTERÍSTICAS DE UN PROBLEMA RECURSIVO?

Observemos las siguientes imagenes:







PROBLEMA A RESOLVER **SIN RECURSIÓN**

Realizar un procedimiento que se llame **CuentaRegresiva** que muestre por pantalla la cuenta atrás de un número que nosotros pasemos como parámetro a la función.

```
Procedure CuentaRegresiva( n:integer);
var i: integer
begin
 For i:=n downto 0 do
    writeln (n)
end;
```

PROBLEMA A RESOLVER **CON RECURSIÓN**

Realizar un procedimiento que se llame **CuentaRegresiva** que muestre por pantalla la cuenta atrás de un número que nosotros pasemos como parámetro a la función.

```
Procedure CuentaRegresiva( n:integer);
begin
    If n>=0 then
      begin
         writeln (n);
         CuentaRegresiva(n-1);
      end
end;
```

RECURSION – CONCEPTO

Una solución recursiva resuelve un problema por resolución de instancias más pequeñas del mismo problema y hay un caso especial.

Suponga que quiere resolver el problema de contar cuantas monedas hay en una pila de monedas

```
cant=0
while (hay monedas en la pila) do
    cuento una más (cant + 1)
    saco una moneda de la pila
```

¿Cómo se piensa una solución recursiva?

RECURSION - CARACTERÍSTICAS

Toda solución recursiva tiene:
- un caso base o degenerado
- el problema es el mismo
- pero de menor tamaño

contar monedas de la pila

```
cant=0
Contar monedas de la pila
  si (hay monedas en la pila)
      cuento una más (cant + 1)
      saco una moneda de la pila
      contar monedas de la pila
```

RECURSION – EJEMPLO 1

Observaciones importantes:

→Se llama a sí mismo.

→En cada llamada el tamaño el problema se achica (la pila disminuye).

→Existe un caso distinto que se resuelve de manera particular y directa (sin recursión).

RECURSION – EJEMPLO 1

Formalicemos un poco mas el algoritmo

```
Procedure cuenta (var cant:integer);
Begin
  if (la pila tiene monedas) then
    begin
      cant:= cant + 1;
      <sacar una moneda de la pila>
      cuenta (cant);
   end;
End;
```

¿Dónde se inicializa cant?
¿Cómo se ejecuta un módulo recursivo?

RECURSION – EJEMPLO 1

Pensemos en una pila que no sabemos cuantos elementos puede tener
¿Qué estructura de datos la puede representar?

Lista

```
Type pila = ^listap;
      listap = record
                valor: integer;
                sig: pila;
       end;
```

¿Cómo debemos construirla si debe funcionar como una pila?

Él último elemento en apilar es el primero en salir

RECURSION – EJEMPLO 1

Lista

```
Type pila = ^listap;
     listap = record
                valor: integer;
                sig: pila;
              end;
```

Agrego por el frente, (puntero al primero)

<Sacar un elemento de la pila>
Saco por el frente el primer elemento

¿Cómo es la implementación de sacar?

RECURSION – EJEMPLO 1

```
Programa recursivo;
Procedure cuenta (P:pila; var cant:integer);
Begin
 if (la pila P tiene monedas) then
   begin
     cant:= cant + 1;
     <sacar una moneda de la pila P>
     cuenta (P,cant);
  end;
Var
 cantidad:integer; P:pila
Begin
 cargarPila(P); cantidad:=0;
 cuenta (P, cantidad);
End.
```

```
Programa secuencial;
Procedure cuenta (P:pila; var cant:integer);
Begin
  while (la pila P tiene monedas) do
    begin
      cant:= cant + 1;
      <sacar una moneda de la pila P>
    end;
 end;
Var
  cantidad:integer; P:pila
Begin
  cargarPila(P); cantidad:=0;
  cuenta (P, cantidad);
End.
```

RECURSION – EJEMPLO 1

```
Programa recursivo;
Procedure cuenta (P:pila; var cant:integer);
Begin
  if (la pila P tiene monedas) then
    begin
      cant:= cant + 1;
      <sacar una moneda de la pila P>
      cuenta (P, cant);
   end;
Var
  cantidad:integer; P:pila
Begin
  cargarPila(P); cantidad:=0;
  cuenta (P, cantidad);
End.
```

```
Programa secuencial;
Procedure cuenta (P:pila; var cant:integer);
Begin
  cant:=0;
  while (la pila P tiene monedas) do
    begin
      cant:= cant + 1;
      <sacar una moneda de la pila P>
    end;
 end;
Var
  cantidad:integer; P:pila
Begin
  cargarPila(P);
  cuenta (P, cantidad);
End.
```

IMPLEMENTACIÓN RECURSIVA

```
Program Recursivo;
Type pila = ^listap;
       listap = record
                  valor: integer;
                  sig: pila;
         end;
Procedure cuenta (P:pila; var cant:integer);
Begin
  if (P <> nil) then
    begin
      cant:= cant + 1;
      P:= P^.sig;
      cuenta (P, cant);
   end;
```

```
Var
  cantidad:integer; P:pila
Begin {del programa principal}
  cargarPila(P); cantidad:=0;
  cuenta (P, cantidad);
End.
```

¿Cómo funciona la recursión en memoria RAM?

RECURSION - DEFINICIÓN

Una solución para ser recursiva debe cumplir con que:

1. El problema debe resolverse en instancias del mismo problema
2. En cada llamado debe reducirse el espacio de solución del problema
3. Debe existir un caso base o degenerado que se resuelve sin recursión.

RECURSION – EJEMPLO 2

Suponga que se quiere calcular el factorial de un número

Solución secuencial

```
function factorial (n:integer):integer;
Var
  i,fac:integer;
Begin
 fac:=1;
 for i:= n downto 2 do
   fac:= fac*i;
 factorial:= fac;
End;
```

Solución recursiva

```
function factorial (n:integer):integer;
Var
  i,fac:integer;

Begin

End;
```

Caso base?

$$N! \begin{cases} 1 & \text{si } N<=1 \\ N*(N-1)! & \text{si } N>1 \end{cases}$$

RECURSION – EJEMPLO 2

Probar código en líena en http://ideone.com

```
Function factorial ( n: integer ) :  integer;
{ Recibe un entero en n y devuelve un valor integer
   con su factorial}
 begin
   if ( n <=1 ) then factorial :=1
                  else  factorial := n * factorial ( n - 1);
 end;
```