Flujo del programa

- El flujo del programa hace referencia al **orden** en que se **ejecutan** las instrucciones.
- El flujo por defecto de un programa es el **secuencial**:
  - El ordenador ejecuta cada sentencia y va a la siguiente hasta el final del programa.
- Este flujo puede **alterarse** mediante:
  - Sentencias de **selección**:
    - Sentencia **`if`**.
    - Sentencia **`case`**.
  - Sentencias de **iteración (bucles)**:
    - Sentencia **`while`**.
    - Sentencia **`repeat`**.
    - Sentencia **`for`**.

Selección: Sentencia `if` (1)

- La sentencia **`if`** permite la **ejecución condicional** de una o varias sentencia.
- Tiene diversas formas:
  - Ejecución condicional de **una** sentencia

```
if expresión_lógica then
      sentencia;
```

    Si la expresión lógica es cierta (**`true`**) se ejecuta la sentencia. En otro caso (**`false`**) , se omite su ejecución.

  - Ejecución condicional de **varias** sentencias
    -

```
if expresión_lógica then
begin
      sentencia1;
      ...
      sentencian;
end
```

Selección: Sentencia `if` (2)

- Ejecución **bicondicional** de sentencias
  - **`if`** *`expresión_lógica`* **`then`**
    `sentencia1`
    `else`
    `sentencia2``;`

    Si la *`expresión_lógica`* es **`false`** se ejecuta la sentencia que sigue al **`else`**. Nótese la falta del punto y coma tras *`sentencia1`*.

- Ejecución **multicondicional** de sentencias (**`if`** anidados)
  - **`if`** *`expresión_lógica1`* **`then`**
    `sentencia1`
    `else`
    `if` *`expresión_lógica2`* **`then`**
    `sentencia2``;`
    `else`
    `sentencia3``;`
  - En los **`if`** anidados cada **`else`** se corresponde con el **`if`** anterior.
- En ambos casos se permiten también los grupos de sentencias.

Selección: Sentencia `if` (3)

- Ejemplos:
  - Utilización de **`if`**

```
program edades;
var
   edad : integer ;
begin
   writeln('escribe tu edad : ');
   readln(edad);
   if edad >= 18 then
      writeln('!eres mayor de edad !');
   writeln('esta instrucción siempre se ejecuta')

end.
```

  - Utilización de **`if – else`**

```
program edades;
var
   edad : integer ;
begin
   writeln('escribe tu edad : ');
   readln(edad);
   if edad >= 18 then
      writeln('!eres mayor de edad !')
   else
      writeln('!eres menor de edad !')
   writeln('esta instrucción siempre se ejecuta')

end.
```

Selección: Sentencia `if` (4)

- Utilización de **`if – else`** con bloques

```
program edades;
var
   edad : integer ;
begin
   writeln('escribe tu edad : ');
   readln(edad);
   if edad >= 18 then
   begin
      writeln('!eres mayor de edad !');
      writeln('!ya puedes votar !`)
   end
   else
   begin
      writeln('!eres menor de edad !');
      writeln('!no puedes votar !`)
   end;
   writeln('esta instrucción siempre se ejecuta')

end.
```

Selección: Sentencia `if` (5)

- Utilización de **`if`** anidados

```
program nummayor;
var
   n1,n2,n3,mayor : integer ;
begin
   writeln('escribe tres numeros enteros : ');
   readln(n1,n2,n3);
   if n1>n2 then
      if n1>n3 then
                mayor:=n1
      else
                mayor:=n3
   else
      if n2>n3 then
                mayor:=n2
      else
                mayor:=n3;

   writeln('el mayor es ',mayor)
end.
```

Selección: Sentencia `case` (1)

- **Simplifica** en muchos casos los **`if`** anidados
- Forma general:
  - **case** *selector* **of**
    lista_de_constantes1:
    begin
    sentencia1;
    ...
    sentencian
    end;

    lista_de_constantes2:
    begin
    ...

    end;

    ...
    lista_de_constantesm:
    begin
    ...

    end
    else
    begin
    ...

    end;
    end;

Selección: Sentencia `case` (2)

- **Características** generales
  - La expresión *`selector`*, se evalúa y compara con la *`lista_de_constantes`*. Si la comparación tiene éxito se ejecutarán las instrucciones etiquetadas por *`lista_de_constantes`*.
  - Si el valor de *`selector`* no está comprendido en ninguna lista de constantes y no existe la cláusula **`else`**, sigue el flujo del programa; si existe la cláusula **`else`** se ejecutan las instrucciones a continuación de la cláusula **`else`**.
  - El selector debe ser un tipo ordinal (**`integer`**, **`char`**, **`boolean`** o enumerado). Los números reales no pueden utilizarse.
  - Todas las constantes en el **`case`** deben ser únicas y de un tipo ordinal compatible con el tipo del selector.
  - Cada sentencia, excepto la última, deben ir seguidas del punto y coma.
  - No debe escribirse punto y coma antes de la palabra **`else`**.

Selección: Sentencia `case` (3)

- Ejemplo:
  - **`Con if`** anidados:

```
if operator = '*' then
        result := number1 * number2
else
        if operator = '/' then
                result := number1 / number2
        else
                if operator = '+' then
                        result := number1 + number2
                else
                        if operator = '-' then
                                result := number1 - number2
                        else
                                invalid_operator = 1;
```

  - Sentencia **`case`**:

```
case operator of
   '*' :
        result:= number1 * number2;
   '/' :
        result:= number1 / number2;
   '+' :
        result:= number1 + number2;
   '-' :
        result:= number1 - number2;
   else
        invalid_operator := 1
end;
```

Iteración: Sentencia `for` (1)

- **Repite** un grupo de instrucciones un número **conocido** de veces.
- El **formato** de la sentencia **`for`** es:

```
for contador:=expresión1 to expresión2 do
begin
    instrucción1;
    instrucción2;
    ...
    instrucciónn
end;
```

- Características:
  - ¿Cómo se ejecuta la sentencia **`for`** ?
    1. Se asigna a *`contador`* el valor inicial *`expresión1`*.
    2. Se ejecuta el cuerpo del bucle (las intrucciones de su interior).
    3. Se incrementa *`contador`* en uno.
    4. Si *`contador`* es menor o igual que *`expresión2`* se vuelve a 2.
  - El contador se puede decrementar sustituyendo la palabra **`to`** por la palabra **`downto`**.
  - El contador no puede ser de tipo **`real`**.

Iteración: Sentencia `for` (2)

- Ejemplos:

```
program ej_for;
var
   valor_final,contador : integer;
begin
   write('escribe el número de iteraciones : ');
   readln(valor_final);
   for contador:=1 to valor_final do
     writeln('iteración : ',contador)
end.
```

```
program ej_for;
var
   valor_final,contador : integer;
begin
   write('escribe el número de iteraciones : ');
   readln(valor_final);
   for contador:=valor_final downto 1 do
     writeln('iteración : ',contador)
end.
```

Iteración: Sentencia `while` (1)

- **Repite** un grupo de instrucciones **mientras** una condición sea **cierta**.
- El **formato** de la sentencia **`while`** (mientras) es:

```
while condición do
begin
    instrucción1;
    instrucción2;
    ...
    instrucciónn
end;
```

- Características.
  - ¿Cómo se ejecuta la sentencia **`while`** ?
    1. Si condición es **`true`**
       - se ejecuta el cuerpo del **`while`** (las instrucciones del interior).
       - se vuelve a 1.

       En otro caso
       - sigue la ejecución normal del programa
  - Nótese que el **`while`** se ejecutará **indefinidamente** a menos que alguna sentencia en el interior del bucle **modifique la condición** haciendo que su valor pase a falso.

Iteración: Sentencia `while` (2)

- Ejemplo:
  - Encontrar el primer término de la serie armónica que sobrepasa un límite prefijado

```
program armonica;
var
   cuentaterminos:integer;
   suma,limite:real;
begin
     cuentaterminos:=1;
     suma:=1;
     writeln('Introduzca el limite');
     readln(limite);

     while(suma <= limite) do
     begin
          cuentaterminos:=cuentaterminos+1;
          suma:=suma+1/cuentaterminos
     end;

    writeln('Límite sobrepasado en el término ',cuentaterminos)
end.
```

Iteración: Sentencia `repeat` (1)

- **Repite** un grupo de instrucciones **hasta** que una condición sea cierta
- El **formato** de la sentencia **`repeat`** (repite) es:

```
repeat
    instrucción1;
    instrucción2;
    ...
    instrucciónn
until condición;
```

- Características:
  - ¿Cómo se ejecuta la sentencia **`repeat`** ?
    1. Se ejecutan el cuerpo del **`repeat`** (las instrucciones de su interior).
    2. si condición es **`false`**
       - se vuelve a 1.

       en otro caso
       - sigue la ejecución normal del programa
  - Nótese que el **`repeat`** se **ejecutará indefinidamente** a menos que alguna sentencia en el interior del bucle **modifique** la condición haciendo que su valor pase a **`true`**.

Iteración: Sentencia `repeat` (2)

- Ejemplo:
  - Sumar un conjunto de números dados por el usuario hasta que introduzca el valor 0.

```
program sumanums
var
   num,suma:integer;
begin
   suma:=0;
   repeat
          writeln('Introduzca un número (0 para terminar)');
          readln(num);
          suma:=suma+num;
   until num=0;
   writeln(La suma total es: ',suma);
end.
```

Iteración: Bucles anidados

- Bucles anidados:
  - El cuerpo de un bucle puede contener **cualquier** tipo de sentencias: secuenciales (simples o compuestas), condicionales (**`if`**, **`case`**) o iterativas (**`for`**, **`while`**, **`repeat`**).
  - Cuando un bucle está contenido en otro bucle se dice que está **anidado**.
  - Ejemplo:

```
program tablas;
var
   i,j:integer;
begin
for i:=1 to 10 do
begin
   writeln('La tabla del ',i);
   for j:=1 to 10 do
          writeln(i,' por ',j,'= ',i*j);
   writeln
end;
end.
```

Iteración: Reglas generales

- El bucle **`for:`**
  - Se suele reservar para situaciones en que el número de repeticiones se conoce **antes** de que empiece el bucle.
- Los bucles **`while`**, **`repeat:`**
  - Se suelen reservar para aquellas situaciones en las que el número de repeticiones no se conoce **antes** de que empiece el bucle.
  - El cuerpo del bucle **`repeat`** siempre **se ejecutará al menos una vez**.
  - El cuerpo del bucle **`while`** puede que **no se ejecute** si la condición es falsa.
  - En la práctica es más común encontrarse con el bucle **`while`** que con el bucle **`repeat.`**