Facultad de Informática

PROGRAMACIÓN I
Sesiones teóricas

Departamento de Ciencias de la Computación
y Tecnologías de la Información

SESIONES TEÓRICAS

SENTENCIAS DE CONTROL

- **SECUENCIAL**
- **ALTERNATIVA**
  - La sentencia IF-ELSE
  - La sentencia SWITCH
- **REPETITIVA**
  - Introducción
  - Bucle WHILE
  - Bucle DO-WHILE
  - Bucle FOR
  - Equivalencia entre bucles
  - Ejemplos
  - Errores en los bucles
  - Diseño de bucles

SENTENCIAS DE CONTROL

Teorema de Böhm y Jacopini (o de la programación estructurada)

- La **programación estructurada** se basa en **programas PROPIOS**
  - Tienen **un** solo **punto de entrada** y **un** solo **punto de salida**
  - **Toda acción** del algoritmo es **accesible** pues existe al menos un camino que va desde el inicio hasta el fin del algoritmo pasando a través de dicha acción
  - **No** tienen **bucles infinitos**
- **TEOREMA**: todo algoritmo (y por lo tanto, **cualquier programa**) se puede construir con **sólo 3 componentes estructurales**
  - **Secuencia**
  - **Selección**
  - **Repetición**

SENTENCIAS DE CONTROL

Clasificación de sentencias ejecutables en C

- **SIMPLES**
  - Asignación
  - Sentencia vacía
  - Activación de procedimiento
  - Sentencia GOTO
- **ESTRUCTURADAS**
  - Compuestas
  - Condicionales: IF-ELSE, SWITCH
  - Repetititvas: WHILE, DO WHILE, FOR

SENTENCIAS DE CONTROL

Sentencias Simples. Sentencia de Asignación

- **Sentencia de asignación:**
  - **Ejemplos**
    - x = a + b;
    - q = (j>=20) and (j <100);
    - v = sqrt(z) - (i * j);
    - NombreCompleto = " Sr/Sra " + Apellidos+ ´, ´ + Nombre;

- **La sentencia vacía corresponde simplemente a un ;**

SENTENCIAS DE CONTROL

Sentencias Simples. Sentencia GOTO

GOTO está **PROHIBIDA** dentro de una **programación estructurada ***

Si se quiere **alterar el orden** de ejecución secuencial se puede realizar con sentencias **CONDICIONALES**.

*** Opción de uso en aplicaciones muy especiales**

SENTENCIAS DE CONTROL

Sentencias Estructuradas

- **Sentencia compuesta**
  - Ejemplos:

```
{
  b=sqrt(8);
  a=cos(PI);
  c=a+b;
}
```

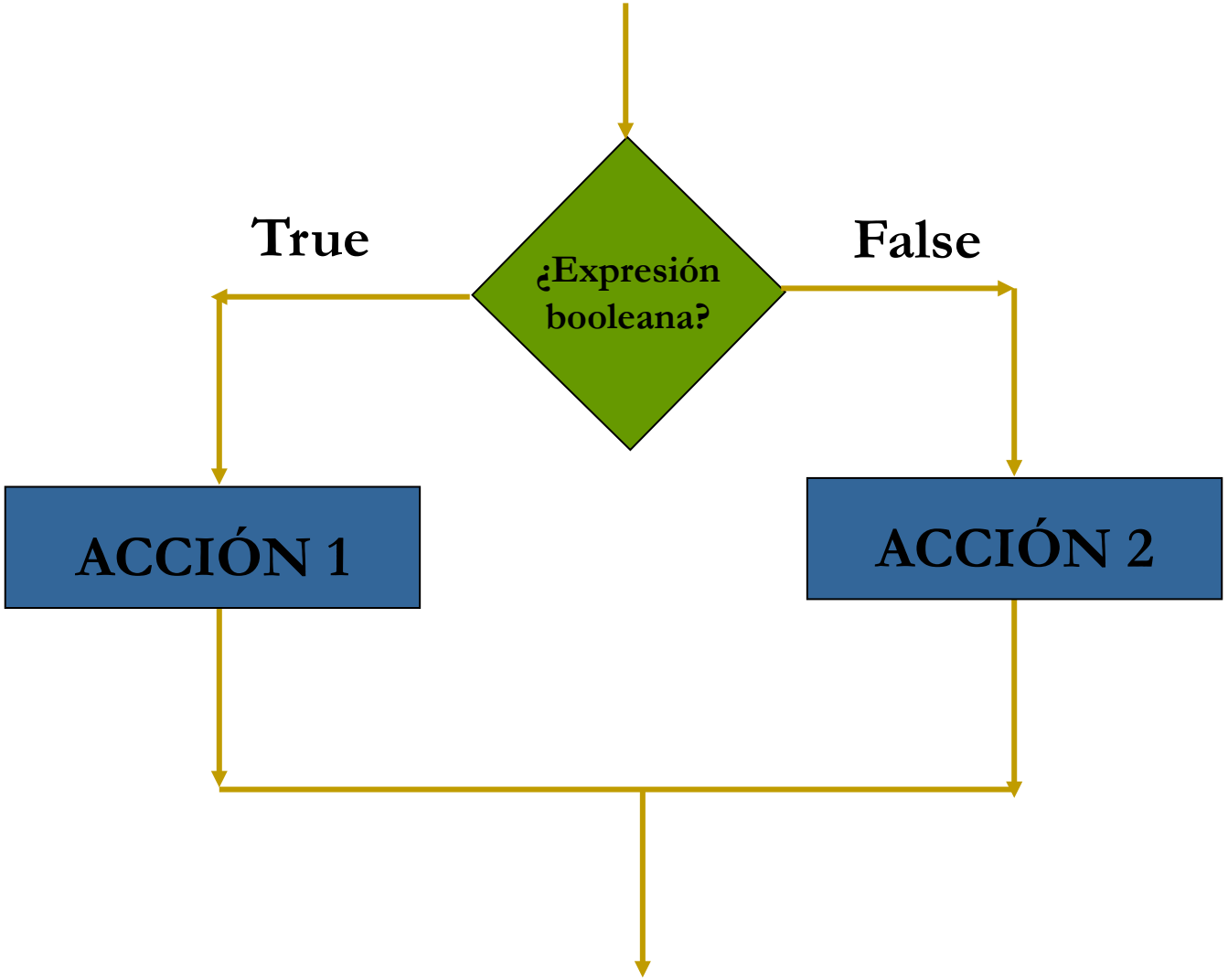
```
{
  scanf( "Valor" %d, &b);
  if (b>0)
      {
        a=cos(PI);
        c=a+b;
      }
}
```

SENTENCIAS DE CONTROL

IF-ELSE

SENTENCIAS DE CONTROL

ALTERNATIVA. Sentencia IF-ELSE



SENTENCIAS DE CONTROL

Sentencia IF

if (expresión) sentencia;

if (expresión) {sentencias}

```
if (Valor > 10) dato=0;

if (Valor == MAX){
   Vmax = 1;
    error= “no”;
}
```

SENTENCIAS DE CONTROL

Operadores, recordatorio de características

RECORDEMOS

- La precedencia de los operadores relacionales es menor que la de los operadores aritméticos.

  Por ejemplo, i + j < k - 1 significa (i + j) <(k - 1).

- Los operadores relacionales son asociativos a la izquierda.

  La expresión:

  i <j <k la acepta el compilador, pero no prueba si j se encuentra entre i y k.

  Como el operador < es asociativo a la izquierda, esta expresión es equivalente a

  (i <j) <k *(ojo!! i<j, es 1 o 0)*

  La expresión correcta es i <j && j <k.

- -Los operadores de igualdad tienen una prioridad más baja que los operadores relacionales, por lo tanto la expresión

  i <j == j <k es equivalente a (i <j) == (j <k)

SENTENCIAS DE CONTROL

Sentencia IF- ELSE

`if(`*expresión*`)`*sentencia* `else` *sentencia*

`if(`*expression*`){`*sentencias*`}`
`else{`*sentencias*`}`

Ejemplos:

```
if(i>j) max=i; else max=j;
```

```
if(i>j)
   if(i>k)
     max=i;
   else
     max=k;
else
   if(j>k)
     max=j;
   else
     max=k;
```

```
if(i>j){
   if(i>k){
     max=i;
   }else{
     max=k;
   }
}else{
   if(j>k){
     max=j;
   }else{
     max=k;
   }
}
```

SENTENCIAS DE CONTROL

ALTERNATIVA. Sentencia IF

- La **condición** de una sentencia IF tiene que ser de tipo **BOOLEAN**.
- Las sentencias de las parte IF y ELSE pueden ser de **cualquier tipo.**
- **Se pueden anidar** varias sentencias IF-ELSE.

Es preciso tener en cuenta que la cláusula **ELSE** afecta siempre a la **sentencia IF más próxima** (forma de resolver el problema de la ambigüedad de la gramática)

SENTENCIAS DE CONTROL

Sentencia IF- ELSE

- Aunque la segunda instrucción **if** esté anidada dentro de la primera, los programadores de C no suelen endentarla. En su lugar, se alinean entre sí con el original:

```
if (n <0)
   printf ("n es menor que 0 \ n");
else if (n == 0)
   printf ("n es igual a 0 \ n");
else
   printf ("n es mayor que 0 \ n");
```

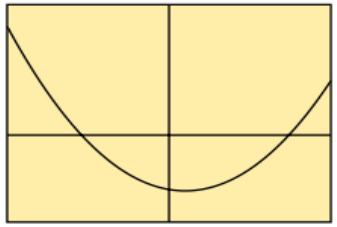
SENTENCIAS DE CONTROL

Sentencia IF- ELSE. Ejemplo. Raices ecuación 2ºGrado

El programa ejemplo propuesto resuelve una ecuación de segundo grado:

Ecuación cuadrática

$$ax^2 + bx + c = 0$$



Solución general x1,x2 →

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

El programa debe contemplar que:

- Si tenemos a=0 nos queda la ecuación de una recta, b*x+c=0, con la solución única

  x=-c/b

- Si (b^2-4ac) es negativo, tenemos una solución compleja de la forma

  parteReal +/- parteImaginaria

SENTENCIAS DE CONTROL

Sentencia IF- ELSE. Ejemplo. Raices ecuación 2ºGrado

```
#include<stdio.h>
#include<math.h>
int main(void) {
   float a, b, c, disc, xR1, xR2, xI, xR;

   printf("Escriba el valor de a: ");
   scanf("%f", &a);
   printf("Escriba el valor de b: ");
   scanf("%f", &b);
   printf("Escriba el valor de c: ");
   scanf("%f", &c);
   if (a != 0) {
       disc = pow(b, 2.0) - 4 * a * c;
       if (disc > 0.0) {
           printf("Las dos raices son reales");
           xR1 = ((-b + sqrt(disc)) / (2.0 * a));
           xR2 = ((-b - sqrt(disc)) / (2.0 * a));
           printf("x1=%.2f   x2=%.2f", xR1, xR2);
       } else {
              if (disc == 0.0) {
              xR1 = (-b) / (2.0 * a);
              printf("La ecuacion solo tiene una raiz %.2f", xR1);
              } else {  /* disc <0 */
               xR = (-b / (2.0 * a));
               xI = (sqrt(-disc) / (2.0 * a));
               printf("La solución es compleja");
              printf("La parte real es %.2f y la imaginaria es +/-%.2fi", xR, xI);
            }
       }
   }
```

SENTENCIAS DE CONTROL

Sentencia IF- ELSE. Ejemplo. Raices ecuación 2ºGrado

```
else{
        printf("Es una recta, el punto de corte es:");
        xR1 = -c/b;
        printf("x=%.2f", xR1);
        printf("\n\n");
      }
   return 0;
}
```

Otro ejemplo
If-else de otra forma

SENTENCIAS DE CONTROL

If else, de otra forma

La sentencia

if (expresión) sentencia1; **else** sentencia2;

Se puede escribir como:

expresión **?** sentencia1**:** sentencia2

La sentencia se evalúa en etapas: expresión se evalúa primero; si su valor no es cero (true), entonces se ejecuta sentencia1. Si el valor de expre1 es cero (false), entonces se ejecuta sentencia2

Ejemplo:

La sentencia:

```
if(dato>100) printf("Dato superior a cien\n"); else printf("Dato inferior a 100\n");
```

Se puede escribir como:

```
dato>100 ? printf("Dato superior a cien\n"): printf("Dato inferior a 100\n");
```

SENTENCIAS DE CONTROL

If else, de otra forma

expresión **?** sentencia1**:** sentencia2

- Ejemplo de implementación de un "código ofuscado".

Se denomina creación de código ofuscado: "*al acto deliberado de realizar un cambio no destructivo, en el código fuente, con el fin de que no sea fácil de entender o leer*".

Ejemplo:

Se pide la nota del examen. Si es mayor que 5 se saca por pantalla el mensaje de aprobado con la nota; si es inferior a 5 se notifica suspenso y la nota. Si la nota es 0 se sube a 1

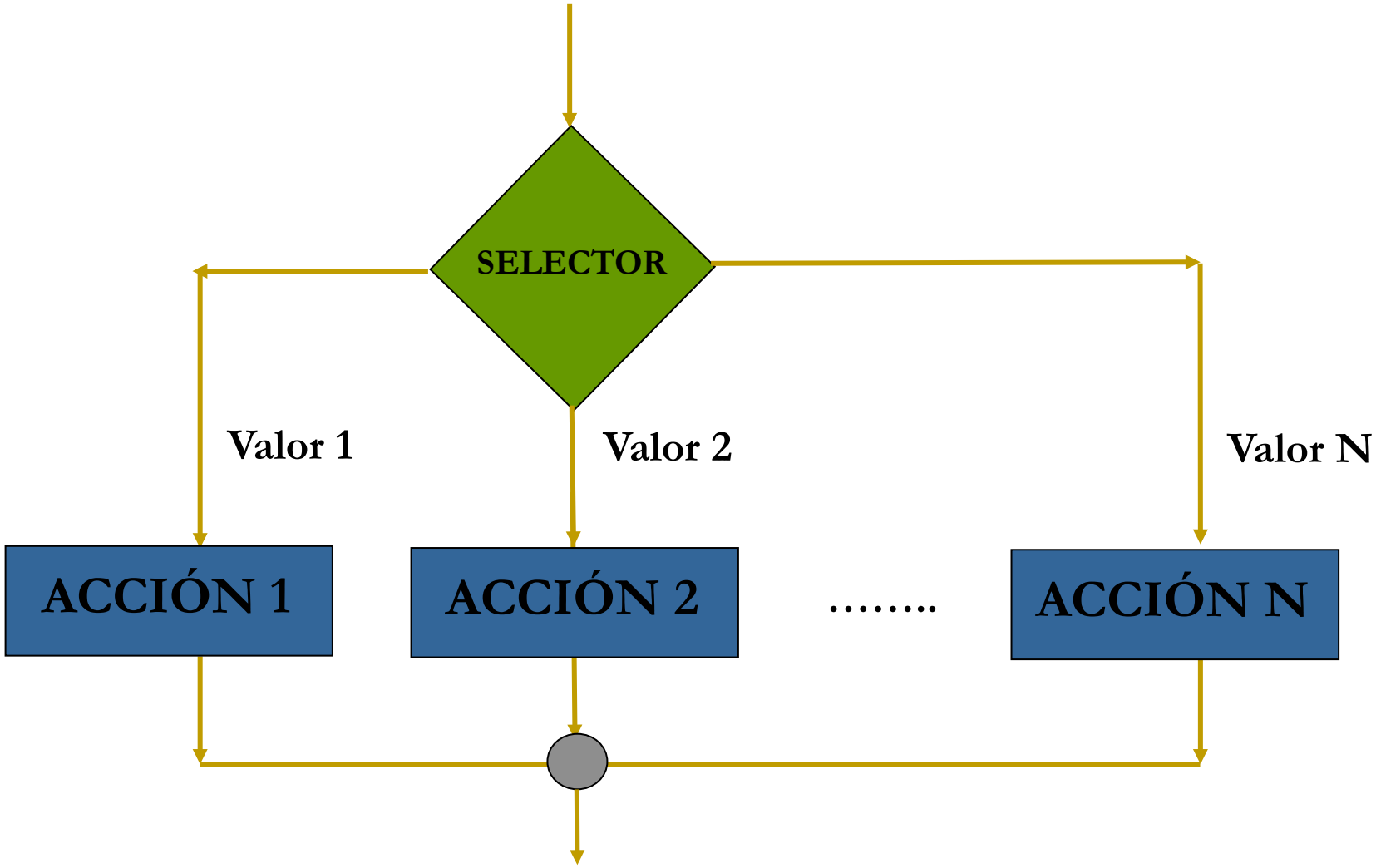
```
#define h printf
#define s scanf  /* estas definiciones se pueden incluir en stdio.h para ofuscar el código */

int main(){int n;h("Nota?\n");s("%d\n",&n);n>0?:n++;;
h("\n", n>5?h("Aprobado, nota=%d",n): h("Suspenso,=%d",n));return 0;}
```

Ojo, el código está probado, en versiones actuales de compiladores C puede dar un warning pero se compila y ejecuta.

SWITCH

SENTENCIAS DE CONTROL

ALTERNATIVA. Sentencia SWITCH



SENTENCIAS DE CONTROL

ALTERNATIVA. Sentencia SWITCH

Supongamos este trozo de código

```
if(Nota>=9)
   printf(“Sobresaliente”);
else if (Nota>=7)
  printf(“Notable”);
else if (Nota>=5)
  printf(“Aprobado”);
else if (Nota>=1)
  printf(“Suspenso”);
else

  printf(“Caso perdido”);
```

SENTENCIAS DE CONTROL

ALTERNATIVA. Sentencia SWITCH

Una alternativa es usar la sentencia **switch**.

```
switch (Nota) {
     case 10: case 9:printf(“Sobresaliente”);
                break;
     case 8: case 7: printf(“Notable”);
                break;
     case 6: case 5: printf(“Aprobado”);
                break;
     case 4:case3:case2:case1: printf(“Suspenso”)
                break;

     default: printf¨(“Caso perdido");
                break;
   }
```

SENTENCIAS DE CONTROL

ALTERNATIVA. Sentencia SWITCH

```
switch(expression){
      case constant-expression: statements
      …
      case constant-expression: statements
      default: statements
   }
```

Expression esto es, la expresión de control entre paréntesis, debe ser una expresión entera.

Los caracteres se tratan como enteros en C y, por lo tanto, se pueden usar en expresiones de control. Sin embargo, los números de punto flotante y las cadenas no.

SENTENCIAS DE CONTROL

ALTERNATIVA. Sentencia SWITCH

No se permiten etiquetas duplicadas.

El orden de los casos no importa.

```
swtich (dato){
    case 3: (sentencia vacía) ← no es el primer caso
    case 2:
    case 1: printf ("el dato vale  UNO");
            break;   ← indica salir del swtich, ya ha: "encontrado una salida"
    case 0: printf ("el dato vale CERO");
            break;
    default: printf ("ninguno");
            break;
 }
```

¿Qué ocurre si la entrada es 3 o 2?

SENTENCIAS DE CONTROL

ALTERNATIVA. Sentencia SWITCH

Se pueden poner las etiquetas en la misma línea (ojo, claridad del programa!)
No se requieren llaves alrededor de las declaraciones de un caso (en el ejemplo, el caso 1).

```
swtich (dato){
    case 3: case 2: case 1:
                    printf("el dato no es CERO\n");
                    printf("no se necesita poner llaves, imprimo este mensaje también);
                    break;
    case 0: printf ("el dato es igual a CERO\n");
            break;
    default: printf ("No me han programado este caso\n");
            break;
 }
```

Si falta el caso default y el valor de la expresión de control no coincide con ninguna etiqueta, el control del programa pasa a la siguiente declaración después del swtich.

SENTENCIAS DE CONTROL

ALTERNATIVA. Sentencia SWITCH

Si no se pone break (o alguna otra instrucción de salto) al final de un caso, el control del programa fluirá hacia el siguiente caso!!!

```
swtich (dato){
    case 3: case 2: case 1:
                        printf("UNO ");

    case 0:             printf ("CERO ");

    default:            printf ("ninguno");
                        break;
 }
```

La salida del programa, para cualquier valor, sería "UNO CERO ninguno"

¿por qué salen espaciadas las palabras?

Programa ejemplo

Ejemplo de SWITCH

```
#include<stdio.h>

int main()
  {
     char ch;
     printf("Introduzca una vocal: ");
     scanf("%c", &ch);
     switch(ch) {
          case 'a': printf("Se ha pulsado una letra a\n");
           break;
          case 'e': printf("Se ha pulsado una letra e\n");
           break;
          case 'i':  printf("Se ha pulsado una letra i\n");
           break;
          case 'o': printf("Se ha pulsado una letra o\n");
           break;
          case 'u': printf("Se ha pulsado una letra u\n");
           break;
          default: printf("Error, %c, no es una vocal\n", ch);
    }
    return(0);
}
```

Ejemplo sentencia switch con if-else

Ejemplo Adivinar_un_Número

Objetivo: adivinar un número conociendo su paridad y el resto de dividirlo por cinco

```
*********************************************
          NÚMERO      RESTO      PARIDAD
            1           1       1 impar
            2           2       0 par
            3           3       1 impar
            4           4       0 par
            5           0       1 impar
            6           1       0 par
            7           2       1 impar
            8           3       0 par
            9           4       1 impar
           10           0       0 par
*****************************************
```

Ejemplo1 sentencia switch con if-else

```
#include <stdio.h>
#include <stdlib.h>
int main()
 {
    int Paridad, Resto;
    printf("Piense un numero de 1 a 10\n");
    printf("Diga si es impar(1) o par (2)\n");
    scanf("%d",&Paridad);
    printf("Diga el resto de dividirlo por 5\n");
    scanf("%d",&Resto);
    printf("El numero pensado ha sido: el ");
    switch(Resto){
          case 0: if (Paridad == 1) printf("5"); else printf("10");
                  break;
          case 1: if (Paridad == 1) printf("1"); else printf("6");
                  break;
          case 2: if (Paridad == 1) printf("7´"); else printf("2");
                  break;
         case 3: if (Paridad == 1) printf("3"); else printf("8");
                  break;
          case 4: if (Paridad == 1) printf("9"); else printf("4");
                  break;
        default: printf("\n \n Error, el resto debe estar entre 0 y 4 \n");
        break; }
    return(0);
 }
```

Ejemplo sentencia switch con if-else

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int Paridad,Resto, N;
   printf("Piense un númro del 1 al 10 \n");
   printf("Diga si es impar(1) o par(2):\n");
   scanf("%d",&Paridad);
   printf("Diga el resto de dividir por 5:\n");
   scanf("%d",&Resto);
   printf("El número pensado es:");
   if (Paridad==1){
      switch(Resto){
         case 0: N=5;
         break;
         case 1: N=1;
         break;
         case 2: N=7;
         break;
         case 3: N=3;
         break;
         case 4: N=9;
         break;
         default: printf("ERROR");
         break;
      }
    }
```

Ejemplo2 sentencia if-else con switch

```
    else {
    switch(Resto) {
            case 0: N=10;
            break;
            case 1: N=6;
            break;
            case 2: N=2;
            break;
            case 3: N=8;
            break;
            case 4: N=4;
            break;
            default: printf("ERROR");
             break;
        }
      }
    printf("el número es %d", N);
    return 0;
}
```

Bucles

SENTENCIAS DE CONTROL

REPETITIVA. Introducción

- Permiten programar la repetición de una o más sentencias (**cuerpo del bucle**) mediante la construcción denominada **ciclo** o **bucle.**

- En C, cada bucle tiene una expresión de control. El número de veces que se repite el cuerpo del bucle está determinado esta **sección de control del bucle.** Si la expresión es verdadera (tiene un valor que no es cero), el bucle continúa ejecutándose.

- Cada vez que se ejecuta el cuerpo del bucle (una iteración del bucle), se evalúa la expresión de control.

- Debe existir una **condición de final del bucle.**

TEMA 2: SENTENCIAS DE CONTROL

2.3 REPETITIVA. Introducción

- Dos tipos de bucles: el **ciclo condicional** y el **ciclo con contador**
  - Ciclo condicional

Repetir MIENTRAS condición sentencia1 sentencia2 …… sentenciaN **WHILE**	**Repetir** sentencia1 sentencia2 …… sentenciaN **DO WHILE** **MIENTRAS** condición

  - Ciclo con contador

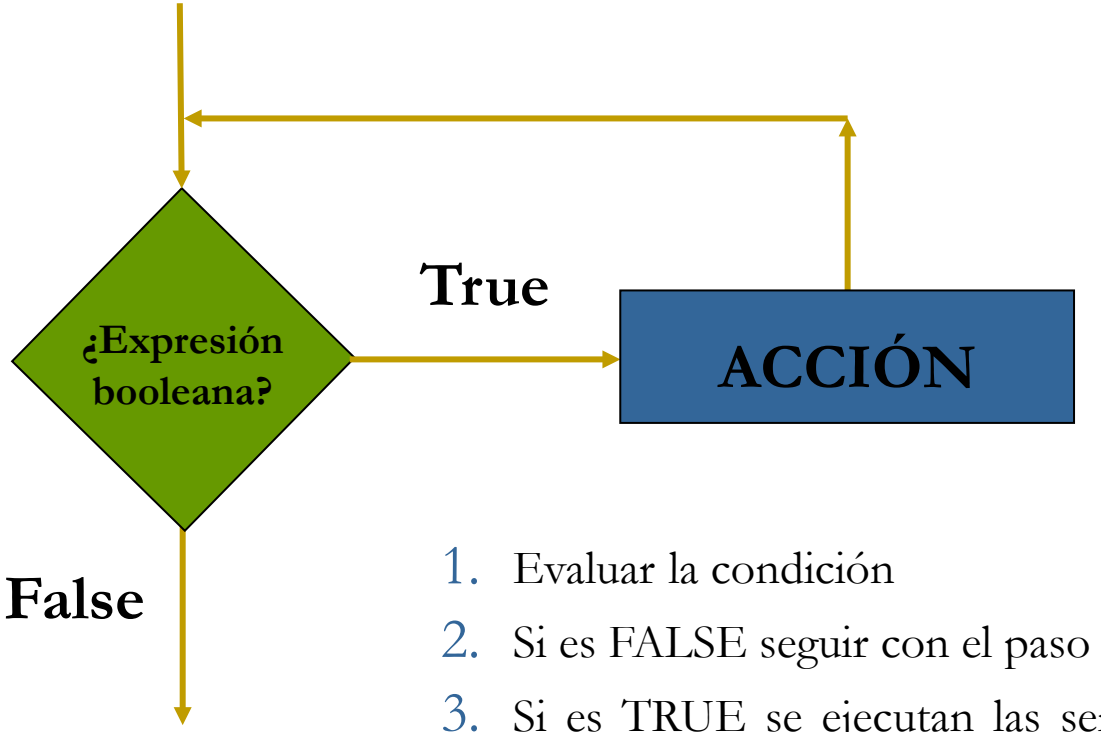
Repetir para VariableControl=valor_inicial, valor_final , incremento
sentencia1
sentencia2
……
sentenciaN
FOR

SENTENCIAS DE CONTROL

Bucle WHILE

SENTENCIAS DE CONTROL

REPETITIVA. Bucle WHILE



1. Evaluar la condición
2. Si es FALSE seguir con el paso 5
3. Si es TRUE se ejecutan las sentencias del cuerpo del bucle
4. Volver al paso 1
5. Continuar con el resto del programa

SENTENCIAS DE CONTROL

REPETITIVA. Bucle WHILE

- Usar una instrucción **while** es la forma más fácil de configurar un bucle.
- La sentencia while tiene la forma:

while (*expresión*) *sentencia*

while (*expresión*) {*acciones*}

donde *expresión* es la expresión controladora de la sentencia/s que se realizan en el cuerpo del bucle

Ejemplo:

```
while(i<n)  /*expression de control*/
  i=i*2;   /*cuerpo del bucle*/
```

SENTENCIAS DE CONTROL

REPETITIVA. Bucle WHILE

Bucle WHILE simple

WHILE (condición) acción**;**

Bucle WHILE con secuencia de acciones

```
WHILE (condición)
   {  acción1;
      acción2;
      ……..
      acciónN;
   }
```

SENTENCIAS DE CONTROL

REPETITIVA. Bucle WHILE

- En el ejemplo anterior consideremos el caso: `n=10`;

```
i=1;
while(i<n)
  i=i*2;
```

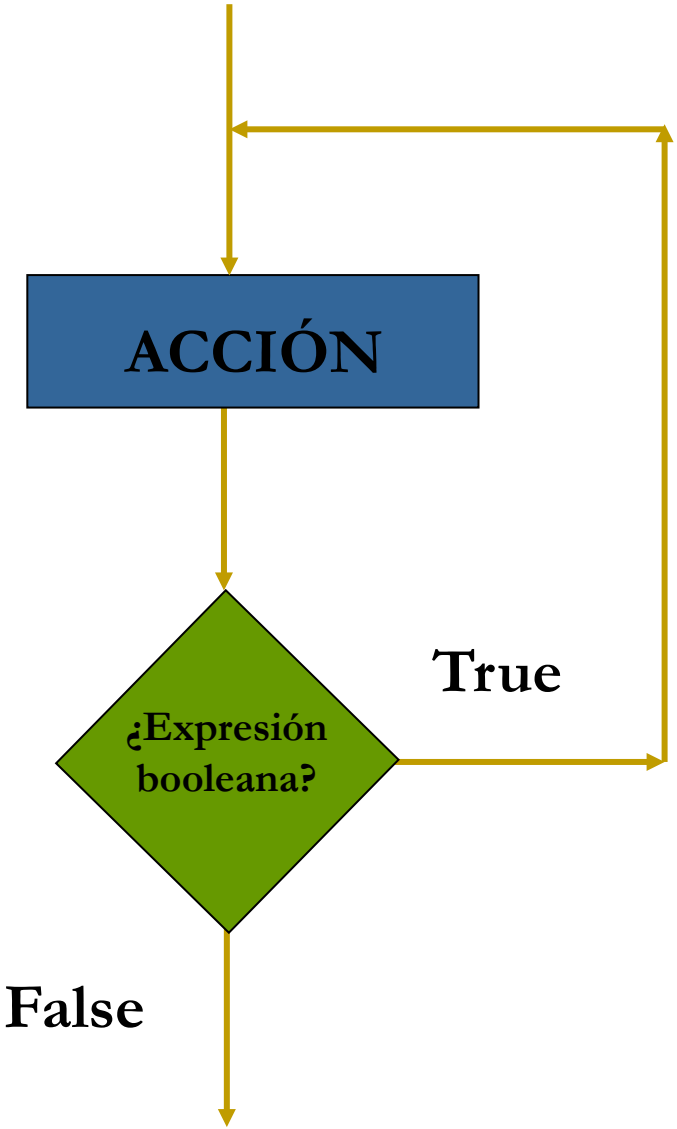
- La secuencia lógica de pasos es:

`i=1;`	`i` es 1.
¿es `i < n`?	Si; continua.
`i=i*2;`	`i` es ahora 2.
¿es `i < n`?	Si; continua.
`i=i*2;`	`i` es 4.
¿es `i < n`?	Si; continua.
`i=i*2;`	`i` es 8.
¿es `i < n`?	Si; continua.
`i=i*2;`	`i` es 16.
¿es `i < n`?	No; salir del bucle.

Bucle DO-WHILE

SENTENCIAS DE CONTROL

REPETITIVA. Bucle DO-WHILE



1. Ejecutar la acción
2. Evaluar la condición
3. Si es TRUE volver al paso 1
4. Si es FALSE seguir con el paso 5
5. Continuar con el resto del programa

SENTENCIAS DE CONTROL

REPETITIVA. Bucle DO WHILE

Bucle DO WHILE simple

DO acción; **WHILE** (condición);

Bucle DO WHILE con secuencia de acciones

```
DO {
      acción1;
      acción2;
       ……..
     acciónN;
} WHILE (condición);
```

REPETITIVA. Bucle WHILE

```
/* Calcula el número de dígitos en un entero */

#include <stdio.h>

int main(void)
{
  int digitos = 0, n;

  printf(“Introduzca un número entero positivo:“);
  scanf("%d", &n);

  do {
    n=n/10;
    digitos=digitos+1;
  } while (n > 0);

  printf(“El número de dígitos es: %d\n", digitos);

  return 0;
}
```

SENTENCIAS DE CONTROL

Bucle FOR

SENTENCIAS DE CONTROL

Bucle FOR

La declaración **for** es la mas adecuada para bucles que tienen una variable de "conteo“ y conocemos el número de repeticiones de antemano…

La forma forma general de la declaración **for** es:

for(*expr1* **;** *expr2* **;** *expr3* **)** *statement*

expr1, expr2 y expr3 son expresiones.

Ejemplo:

```
for (i = 10; i> 0; i--)
printf (“n% d y contando \ n", i);
```

- La **variable de control** debe ser de tipo **ORDINAL**, así como el **valor inicial**.
- La comprobación de **final de bucle** se realiza **antes** de ejecutarse las acciones.
- En *C99*, la primera expresión en una instrucción for puede reemplazarse por una declaración.

```
for (int i = 0; i <n; i ++)
```

La variable i no necesita así haber sido declarada antes.

- OJO, la variable “i” solo es “visible” dentro del lazo.

SENTENCIAS DE CONTROL

Bucle FOR

Bucle FOR simple

FOR (v_ordinal=valor_inicial; condición_valor_final; incremento)
acción**;**

Bucle FOR con secuencia de acciones

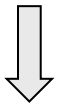
```
FOR (v_ordinal=valor_inicial; condición_ valor_final, incremento)
  {
        acción1;
        acción2;
           …
        acción
  }
```

SENTENCIAS DE CONTROL
Bucles For y While

La sentencia for está estrechamente relacionada con la sentencia while.

Excepto en algunos casos raros, un bucle for siempre se puede reemplazar por un bucle while equivalente:

```
for(expr1;expr2;expr3)statement
```



```
expr1;
while(expr2){
    statement
    expr3;
}
```

expr1 sería un paso de inicialización que se realiza solo una vez, antes de que el bucle comience a ejecutarse.

expr2 controla la terminación del bucle (el bucle continúa ejecutándose siempre que el valor de *expr2* sea distinto de cero).

expr3 es una operación que se realiza al final de cada iteración de bucle.

SENTENCIAS DE CONTROL

Bucle FOR

La declaración for suele ser la mejor opción para los bucles que "cuentan" (incrementa una variable) o "cuenta regresiva" (disminuye una variable).

Una declaración que cuenta hacia arriba o hacia abajo un total de n veces tendrá una de las siguientes formas:

Contando de **0** a **n – 1**:	for (i = 0; i <n; i ++) ...
Contando de **1** a **n**:	for (i = 1; i <= n; i ++) ...
Cuenta regresiva de **n – 1** a **0**:	for (i = n - 1; i> = 0; i--) ...
Cuenta regresiva de **n** a **1**:	for (i = n; i> 0; i--) ...

Bucle FOR

En C99, la primera expresión en una instrucción **for** puede reemplazarse por una declaración. Esta característica le permite al programador declarar una variable para ser utilizada por el bucle:

```
for (int i = 0; i <n; i ++) {

…
```

PERO:

No se puede acceder a una variable declarada por una instrucción **for** fuera del cuerpo del bucle (decimos que no es visible fuera del bucle):

```
for (int i = 0; i <n; i ++) {
    printf ("% d", i);  /* i es visible dentro del bucle */
    ...
}
printf ("% d", i); /* INCORRECTO,  no se compila */
```

Bucle FOR. Ejemplo

```
#include <stdio.h>

int main(void)
{
  int i, n;

  printf(“Este programa calcula los cuadrados de 1 a N.\n");
  printf(“Indique el valor de N: ");
  scanf("%d", &n);

  for (i = 1; i <= n; i++)
    printf("%10d %10d\n", i, i * i);

  return 0;
}
```

Bucle FOR. ejemplo

sum.c

```
/* Suma una serie de números enteros */

#include <stdio.h>

int main()
{
  int n, sum = 0;

  printf(“Este programa suma una serie de números enteros\n");
  printf(“Introduzca los números (0 para finalizar): ");

  scanf("%d", &n);
  while (n != 0) {
    sum += n;
    scanf("%d", &n);
  }
  printf(“La suma es: %d\n", sum);

  return 0;
}
```

Bucle FOR. ejemplo

Cálculo del número e

$$e = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n$$

```
#include <stdlib.h>

int main( )
  {
    float sumatorio, producto, num;
    producto=1;
    sumatorio=1;
    for(num=1;num<=100;num++)
      {
        producto=producto*num;
        sumatorio=sumatorio+(1/producto);
      }
    printf("El numero e es: %.10f",sumatorio);
    system("pause");
    return 0;
  }
```

Sentencia continue

La declaración “continue” es similar a “break”, con la particularidad:

Break transfiere el control el final de un bucle.

Continue transfiere el control a un punto justo antes del final del cuerpo del bucle.

Por lo tanto: con break, el control abandona el bucle ; con continue, el control permanece dentro del bucle.

Hay otra diferencia entre break y continue, break se puede usar en swich (necesario como se ha visto)

Sentencia continue. Ejemplo

```
n = 0;
sum = 0;
while (n < 10) {
  scanf("%d", &i);
  if (i == 0)
     continue;
  sum += i;
  n++;
        /* continue provoca un salto hasta este punto */
}
```

El mismo ejemplo sin emplear continue sería:

```
n = 0;
sum = 0;
while (n < 10) {
  scanf("%d", &i);
  if (i != 0) {
    sum += i;
    n++;
  }
}
```

ESTRUCTURAS SIMPLES DE DATOS
ARRAYS. Ejemplo: Triángulo de Floyd

El **Triángulo de Floyd**, llamado así en honor a Robert Floyd, es un triángulo rectángulo formado con números naturales. Para crear un triángulo de Floyd, se comienza con un 1 en la esquina superior izquierda y se continúa escribiendo la secuencia de los números naturales de manera que cada línea contenga un número más que la anterior:

```
#include <stdio.h>
int main() {
    int i, j, k, tamano;
    k = 1;
    printf("Introduzca el tamaño deseado\n");
    scanf("%d",&tamano);
    printf("El triángulo de Floyd es;\n");
    for (i=1;i<=tamano;i++) {
        for (j=k;j<=k+i-1;j++) {
            printf("%i ",j);
        }
        printf("\n");
        k = k+i;
    }
    return 0;
}
```

ESTRUCTURAS SIMPLES DE DATOS
ARRAYS. Ejemplo: Triángulo de Floyd

La salida para N=7 sería:

- Introduzca el tamano deseado
- 7
- El triangulo de Floyd es;
- 1
- 2 3
- 4 5 6
- 7 8 9 10
- 11 12 13 14 15
- 16 17 18 19 20 21
- 22 23 24 25 26 27 28
-
- Process returned 0 (0x0) execution time : 2.420 s
- Press any key to continue.

SENTENCIAS DE CONTROL

REPETITIVA. Diseño de bucles

Seleccionar TIPO de bucle

Si se conoce o se puede calcular las veces que se va a repetir → **FOR**

Si **NO** se conoce el número de repeticiones:

DO WHILE Si hay que ejecutar la sentencia del bucle al menos 1 vez.

WHILE Si primero hay que evaluar condición

Inicializar variable de control si es bucle WHILE o DO WHILE

Construir correctamente la condición si WHILE o DO WHILE

Comprobar con ejemplos

Alterar variable de control en el cuerpo si WHILE o DO WHILE

NO SE DEBE ALTERAR EN BUCLES FOR, ES AUTOMÁTICO

Comprobar que se alcanza la condición de terminación

SENTENCIAS DE CONTROL

REPETITIVA. Errores en los bucles

- Tres cosas:

  Inicialización

  WHILE **condición**

  ……..

  Alteración de la condición

- **No inicialización de variable de control → Resultado indeterminado**
- **No alteración de la variable de control en el cuerpo → bucle infinito**
- **No se alcanza la condición de terminación → bucle infinito**
- **Mala sintaxis → Resultados inesperados**
- **¡¡¡¡CONDICIÓN MAL CONSTRUIDA!!!!**