Tema 3:
Estructuras de control

Tema 3. Estructuras de control

3.1. Secuencial

3.2. Selección

3.3. Repetición

Objetivos

- **Objetivos del tema:**
  - Conocer y saber la **utilidad de las tres estructuras de control** (secuencial, alternativa y repetitiva) en el paradigma de la **programación estructura**.
  - Aprender a **utilizar** las sentencias asociadas a las **estructuras de control** y los **algoritmos fundamentales** basados en ellas.
  - Aprender a **identificar** distintos **tipos de problemas** y desarrollar **soluciones algorítmicas** que los resuelvan.
  - Todo ello utilizando el **lenguaje C** de forma **estructurada**.

Índice

- Secuencia
- Selección:
  - if
  - if ...else
  - switch
- Repetición:
  - while
  - do … while
  - for

Ejemplo. Programa con alternativa e iterativa

```
/* Programa que escribe los num primeros números naturales  */
#include <stdio.h>
int main()
{
   int contador, num;
   printf("Escribe un numero entero: ");
  scanf("%d", &num);
  if (num <= 0) printf("numero incorrecto");
  else {
          contador = 1;
          while (contador <= num)
          {
             printf ("%d \n", contador);
             contador++;
          }
       }
  return 0;
}
```

Estructura de Control: Secuencial

- **Finalidad**: conocer el conjunto de acciones y el orden en que deben realizarse: de manera automática, una a una, en el orden en que están escritas.
- **Representación**:
  - Lenguaje C

```
{
  sentencia 1;
  sentencia 2;
  .
  .
  .
  sentencia N;
}
```

Bloque o

Sentencia compuesta

Estructura de Control: Alternativa

- **Finalidad**: permite elegir, de entre distintos grupos de sentencias, cuál es la que se ejecuta.

- **Tipos de Alternativas (o condicionales):**
  - Simple
  - Doble
  - Múltiple

Estructura de Control: Alternativa Simple

- **Finalidad**: permite realizar un conjunto de acciones solo cuando se cumpla una determinada situación.

- **Representación** en el lenguaje C

```
if (condición) sentencia
```

La **condición** es una expresión entera lógica.

La **sentencia** puede ser simple o compuesta (bloque). Se ejecutará sólo si la condición es verdadera (toma un valor distinto de cero).

Alternativa Simple. Sintaxis

if (condición) sentencia

Con una sentencia:

```
if (condición)
      sentencia;
```

Con sentencia compuesta:

```
if (condición)
{
      sentencia1;
      sentencia2;
      …
 }
```

Estructura de Control: Alternativa Simple

Escribir un programa que lea un número entero e imprima su valor absoluto.

```
#include <stdio.h>
int main ( )
{
    int num;
    printf( “introduce un número entero: \n”);
    scanf(“%d”, &num);

    if (num < 0)  num = -num ;

    printf( “el valor absoluto es: %d\n”, num);
    return 0;
}
```

Estructura de Control: Alternativa Simple

Escribir un programa C que lea un número, entre 0 y 10, que indica una nota y escriba un mensaje de salida en el caso de estar aprobado.

```
#include <stdio.h>
int main ( )
{
   float nota;
   printf("Introduce la nota obtenida (0-10):");
   scanf("%f", &nota);
   if (nota>=5)
      printf("\n Enhorabuena, has aprobado");
   return 0;
}
```

¿Qué hace el programa si el alumno suspende?

¿Cómo sabemos si el programa funciona correctamente?

Estructura de Control: Alternativa Doble

- **Finalidad**: permite realizar un conjunto de acciones u otras en el caso de que se cumpla o no una determinada situación.
- **Representación** en el lenguaje C

```
if (condición) sentencia1 else sentencia2
```

condición es una expresión entera lógica.

sentencia1 es cualquier sentencia ejecutable (simple o compuesta), que se ejecutará si la condición es verdadera (toma un valor distinto de cero).

sentencia2 es cualquier sentencia ejecutable (simple o compuesta), que se ejecutará si la condición es falsa (igual a cero).

Alternativa Doble. Sintaxis

if (condición**)** sentencia1 **else** sentencia2

Con una sentencia:

```
if (condición)
      sentencia1;
else
      sentencia2;
```

Con sentencia compuesta:

```
if (condición)
   {
        sentencia1.1;
        sentencia1.2;
        …
    }
else
   {
        sentencia2.1;
        sentencia2.2;
        …
    }
```

Estructura de Control: Alternativa Simple

Escribir un programa C que lea un número, entre 0 y 10, que indica una nota y escriba un mensaje de salida indicando si está aprobado o suspenso.

```
#include <stdio.h>
int main ( )
{
   float nota;
   printf("Introduce la nota obtenida (0-10):");
   scanf("%f", &nota);
   if (nota>=5)
      printf("\n Enhorabuena, has aprobado");
   else printf("\n Estas suspenso");
   return 0;
}
```

Estructura de Control: Alternativa Doble

Escribir un programa C que lea dos números enteros distintos e indique cuál de los dos es mayor, el primer número o el segundo.

```
#include <stdio.h>
int main ( )
{
   int n1, n2;
   printf( “introduce dos números enteros: \n”);
   scanf(“%d%d”, &n1, &n2);

   if (n1 > n2)
      printf( “el mayor es el primero: %d”, n1);
   else
      printf( “el mayor es el segundo: %d”, n2);
   return 0;
}
```

Estructura de Control: Alternativa Doble Anidada

Escribir un programa C que lea dos números enteros ~~distintos~~ e indique cuál de los dos es mayor, el primer número, el segundo o si son iguales.

```
#include <stdio.h>
int main ( )
{
    int n1, n2;
    printf( “introduce dos números enteros: \n”);
    scanf(“%d %d”, &n1, &n2);

    if (n1 > n2)
       printf( “el mayor es el primero: %d”, n1);
    else if (n2 > n1)
            printf( “el mayor es el segundo: %d”, n2);
         else printf( “los dos números son iguales);
    return 0;
}
```

Estructura de Control: Alternativa Doble Anidada

Alternativa if doble anidada

- Una de las sentencias (sentencia1 y/o sentencia2) es a su vez una sentencia if.
- Cuando se anidan alternativas dobles **puede haber ambigüedad**, la regla de correspondencia es:

  un **else** se asocia al **if** anterior más cercano que no tenga asociado otro **else.** Para romper la regla se usan { }

Estructura de Control: Alternativas

Escribir un programa C que lea tres números enteros distintos e indique cuál es el mayor de los tres con un mensaje en pantalla

Estructura de Control: Alternativa Múltiple

Estructura if-else-if

- Se utiliza para expresar más claramente las distintas alternativas.
- Si hay que **elegir** entre **una lista de opciones**, y únicamente una de ellas es válida:

```
if (cond1) sentencia1;
else

  if (cond2) sentencia2;
    else

       if (cond3) sentencia3;
            else  sentencia4;

}
```

Equivale a:

```
if (cond1) sentencia1;
else    if (cond2) sentencia2;
else    if (cond3) sentencia3;
else  sentencia4;
```

Estructura de Control: Alternativa Múltiple

El formato de la sentencia **switch** es:

```
switch ( selector ) {
    case constante1: [sentencia 1]     [break;]
    case constante2: [sentencia 2]     [break;]
     ...
    [default:  sentencia n]            [break];
}
```

Selector: expresión de tipo ordinal (int o char). Se evalúa y se compara con cada una de las etiquetas case.

Cada **constante** es un valor único del tipo del selector.

default: Si el valor del selector no coincide con ninguna de las constantes se ejecuta *sentencia n*

Se evalúa el selector y se **compara**, por orden, con cada una de las **constantes del case**. Si se encuentra el valor, se ejecutan la secuencia de instrucciones asociadas con el case, **hasta** que se encuentra la instrucción **break** o el **final** de la instrucción **switch**.

Estructura de Control: Alternativa Múltiple

Dado el siguiente programa codificarlo empleando la sentencia switch.

```
#include <stdio.h>
int main()
{
   int num;
   printf( "Introduce un número " );
   scanf( "%d", &num );
   if ( num==1 )  printf( "Es un  uno\n" );
   else if ( num==2 )  printf( "Es un dos\n" );
        else  if (num==3)  printf ( "Es un tres\n" );
              else  printf( "No era ni 1, ni 2, ni 3\n" );
   return 0;
}
```

Estructura de Control: Alternativa Múltiple

Y codificado con la estructura if-else-if …

```
#include <stdio.h>
int main()
{
   int num;
   printf("Introduce un número ");
   scanf("%d", &num);
   if (num==1)  printf("Es un  uno\n");
   else  if (num==2)  printf("Es un dos\n");
   else  if (num==3)  printf("Es un tres\n");
   else  printf ("No era ni 1, ni 2, ni 3\n");
   return 0;
}
```

Estructura de Control: Alternativa Múltiple

```
#include <stdio.h>
 int main()
{
   int num;
   printf( "Introduce un número " );
   scanf( "%d", &num );
   switch( num ) {
         case 1: printf( "Es un uno\n" );
                 break;
         case 2: printf( "Es un dos\n" );
                 break;
         case 3: printf( "Es un tres\n" );
                 break;
        default: printf( "No es ni 1, ni 2, ni 3\n" );
  }
   return 0;
}
```

Estructura de Control: Iterativas

- **Finalidad**:

  permite la repetición de una acción o un grupo de acciones .

- **Componentes de un ciclo**:
  - valores iniciales
  - condición de finalización
  - cuerpo del ciclo
- **Tipos de acciones iterativas**.
  - Mientas: while
  - Hacer … Mientras : do … while
  - Para: for

Estructura de Control: Iterativa Mientras

Representación en el lenguaje C

```
while (condición)   sentencia
```

condición es una expresión entera lógica que controla la secuencia de repetición. Se evalúa antes de ejecutar la sentencia.

sentencia simple o compuesta, se ejecuta cuando la condición es verdadera (distinta de cero).

El cuerpo del bucle **while** se ejecutará **cero** o más veces

Estructura de Control Iterativa

Ejemplo. Escribir los 10 primeros números naturales

```
#include <stdio.h>
int main()
{
  int contador;
  contador = 1;
  while (contador <= 10)
  {
       printf ("%d \n", contador);
       contador++;
  }
  return 0;
}
```

Estructura de Control Iterativa

Ejemplo. Escribir los 10 primeros números naturales

```
#include <stdio.h>
int main()
{
   int contador;
   contador = 0;
   while (contador < 10)
   {
       contador++;
       printf ("%d \n", contador);
   }
   return 0;
}
```

Estructura de Control Iterativa

Escribir un programa que sume los N primeros números naturales. El valor de N se leerá desde teclado y es un valor mayor o igual que 1. 1+2+3+ …+N

```
#include <stdio.h>
int main()
{
     int   n, numero, suma;
     printf( “introduce  el valor de N: \n”);
     scanf(“%d”, &n);
     suma = 0;
     numero = 1;
     while (numero <= n)
     {
        suma = suma + numero;
        ++numero;
     }
     printf (“La suma es = %d \n”,  suma);
     return 0;
}
```

Estructura de Control Iterativa

Escribir un programa que sume los N primeros números naturales. El valor de N se leerá desde teclado y es un valor mayor o igual que 1. 1+2+3+ …+N

```
#include <stdio.h>
int main()
{
    int   n, numero, suma;
    printf( "introduce  el valor de N: \n");
    scanf("%d", &n);
    suma = 0;
    numero = 0;
    while (numero < n)
    {
          ++numero;
          suma = suma + numero;
    }
    printf ("La suma es = %d \n",  suma);
    return 0;
}
```

Esquemas Algorítmicos: Tipos de Ciclos

Podemos clasificar los ciclos en:

- Ciclo recorrido, controlado por contador
- Ciclo controlado por centinela
- Ciclos anidados

Esquemas Algorítmicos: Ciclo contador

Ciclo recorrido, controlado por contador:

- Una variable "recorre" un intervalo de valores, realizando una iteración para cada uno de los valores del intervalo recorrido.
- Se suele utilizar para:
  - Lectura de una secuencia de valores de entrada (sabiendo cuántos valores la forman)
  - Recorrido de un intervalo de valores (conociendo valor inicial y valor final)
  - Contar las veces que se produce un suceso

Esquemas Algorítmicos: Ciclo contador

Esquema de ciclo contador

(o ciclo recorrido, controlado por contador)

```
// el ciclo se ejecuta
// desde valor_inicial hasta valor_final

cont=   /*valor inicial*/;
while (cont<=  /*valor final*/)
{
    /*Sentencias*/
    ...

    cont= cont+1;
}
```

Esquemas Algorítmicos: Ciclo contador

Ciclo contador:

El bucle se ejecuta un número conocido de veces.

Ejemplo esquemático:

```
cont = 1;
while   (cont  <= 10)
{
    .
    .
    .
   cont = cont + 1;
}
```

* cont - variable de control del bucle (contador)
* En el ejm: el cuerpo del bucle se ejecuta 10 veces

Esquemas Algorítmicos: Ciclo contador

Ciclo contador. Ejemplo:

Realizar un programa que sume los **20 valores** introducidos por teclado.

```
#include <stdio.h>
void main()
{
    int num, suma, cont;
    cont = 1;
    suma = 0;
    while (cont < =20)
    {
        printf("Escribe otro numero\n");
        scanf("%d", &num);
        suma= suma + num;
        cont = cont + 1;
    }
    printf("La suma es: %d \n",suma);
}
```

Esquemas Algorítmicos: Tipos de Ciclos

- Ciclo controlado por centinela: el bucle termina cuando sucede algo en el interior del cuerpo del bucle que indica que se debe salir del bucle.

  - Casos particulares:
    - Centinela especial: EOF
    - Consulta explícita

Esquemas Algorítmicos: Ciclos Centinela

Ciclos controlados por centinela:

- El centinela es un dato especial que nos indica que no hay más datos para procesar.
- El centinela no se procesa (no es un dato válido para el proceso que se pide)

Ejm.- para marcar el final de una serie de datos que se leen desde teclado. Por ejemplo, si todos los datos son positivos, el final de la secuencia se marca con un número cero o negativo.

Esquemas Algorítmicos: Ciclos Centinela

Esquema de ciclos controlados por centinela:

```
scanf( /*lee valor para dato*/ )
while (dato!=centinela)
 {
     /*procesar dato*/
     ...

      scanf( /*lee valor para dato*/ );
  }
```

/* observar que la lectura se hace al final del ciclo */

Esquemas Algorítmicos: Ciclos Centinela

Ciclos controlados por centinela. Ejemplo:
Sumar los valores (enteros positivos) que se introducen desde el teclado. El final de la secuencia lo marca un cero (el centinela **que no se procesa**).

```
#include <stdio.h>
void main()
{
    int num, suma;
    suma = 0;
    printf("Escribe un número\n");
    scanf("%d", &num);
    while (num != 0)
    {
       suma= suma + num;
       printf("Escribe otro número\n");
       scanf("%d", &num);
    }
    printf("La suma es: %d \n",suma);
}
```

Esquemas Algorítmicos: Ciclos Centinela

Esquema de ciclo centinela con búsqueda.

se trata de buscar en una secuencia un valor que cumple una determinada condición, puede no existir el valor buscado. Si existe el valor buscado, se deberá parar el proceso y no llegar al final de la secuencia.

```
scanf( /*lee valor para dato*/ );
encont=0;
while (dato!=centinela && !encont)
{
    /*Sentencias*/

    if (/*condicion de búsqueda*/) encont=1;
    else scanf(/*lee valor para dato*/);
}
if (encont)  printf ("Existe ...");
else printf ("NO existe ...");
```

/* cuando encuentra el valor buscado acaba el while */

Esquemas Algorítmicos: Ciclos Centinela

Ciclo centinela con búsqueda. Ejemplo.

se introducen por teclado, en una línea, valores enteros positivos, el final de la secuencia lo marca un número -1 (es el centinela). Escribir un programa que nos indique **si existe** algún número **par** en la secuencia de entrada.

```
#include <stdio.h>
void main()
{
  int existe = 0, num;
  printf("Escribe números (fin -1):");
  scanf("%d", &num);
  while (!existe &&  num != -1)
       if (num % 2 == 0) existe=1;
       else scanf("%d", &num);
  if (existe) printf ("Hay algún par");
  else printf ("No hay pares");
}  /* acabará el ciclo cuando lea el primer número par, si no hay
      pares llegará al final de la secuencia. */
```

Esquemas Algorítmicos: Ciclos Centinela EOF

Ciclo controlados por centinela: EOF (**E**nd **O**f **F**ile)

- Cuando se introducen datos por teclado, con las teclas **Ctrl+z** finalizamos el fichero de entrada.
- Cuando scanf no encuentra datos para asignar a variables, estamos en el "final de fichero", y devuelve un valor especial, la constante definida en stdio.h de identificador **EOF** que podemos utilizar como centinela.

Esquemas Algorítmicos: Ciclos Centinela EOF

Ciclo controlados por centinela (EOF). Esquema

```
 int fin;
…
fin= scanf( /*lee valores para variables*/ );
while (fin!=EOF)
{
      /*procesar valores leidos*/
      ...

      fin= scanf( /*lee valores para variables*/ );
}
```

/* la entrada de datos por teclado debe finalizar con Ctrl+z */

Esquemas Algorítmicos: Ciclos Centinela EOF

Ciclo controlados por centinela (EOF). Ejemplo

Sumar los valores que se introducen desde el teclado.

```
#include <stdio.h>
void main()
{
    int num, suma, fin;
    suma = 0;
    printf("Escribe un numero\n");
    fin = scanf("%d", &num);
    while (fin != EOF)
    {
       suma= suma + num;
       printf("Escribe otro numero (^Z para acabar): ");
       fin = scanf("%d", &num);
    }
    printf("La suma es: %d \n",suma);
}
```

Esquemas Algorítmicos: Ciclo Consulta

Ciclo controlados por consulta explícita:

- El programa pregunta al usuario si desea continuar la ejecución del bucle.
- La contestación del usuario normalmente se almacena en una variable tipo char o int
- Es el usuario, con su contestación, quien decide la salida de la iteración.

Esquemas Algorítmicos: Ciclos Consulta

Realizar un programa que sume los valores introducidos por teclado. El programa finaliza cuando el usuario no desea introducir más números.

```
#include <stdio.h>
void main()
{
   int num, suma = 0, resp;
   printf("\n quieres sumar numeros? (1 = si, 0=no): ");
   scanf("%d", &resp);
   while (resp== 1)
   {
      printf("\n Escribe un numero");
      scanf("%d", &num);
      suma= suma + num;
      printf("\n quieres seguir sumando? (1 = si, 0=no): ");
      scanf("%d", &resp);
   }
   printf("\n La suma es:  %d", suma);
}
```

Esquemas Algorítmicos: Ciclos Consulta

Realizar un programa que sume los valores introducidos por teclado. El programa finaliza cuando el usuario no desea introducir más números.

```
#include <stdio.h>
void main()
{
    int num, suma = 0;
    char resp;
    printf("\n quieres sumar numeros? s/n");
    scanf("%c", &resp);
    while (resp=='s')
    {
       printf("\n Escribe un numero");
       scanf("%d", &num);
       suma= suma + num;
       printf("\n quieres seguir sumando? s/n");
       fflush (stdin);    // vacía el buffer de entrada
       scanf("%c", &resp);
    }
    printf("\n La suma es:  %d", suma);
}
```

Esquemas Algorítmicos: Ciclos Anidados

Ciclo anidados:

Cuando el cuerpo de la sentencia iterativa contiene otra sentencia iterativa.

Esquemas Algorítmicos: Ciclos Anidados

Ciclo anidados. Ejemplo:

Imprimir un rectángulo de 3 asteriscos de base por 4 asteriscos de altura:

```
…
altura = 1;
while (altura <= 4)
{
      base = 1;
      while (base <= 3)
      {
            printf  (“*”);
            base = base + 1;
      }
      printf (“\n”);
      altura = altura + 1;
}
…
```

Estructura de Control Iterativa

El formato del bucle **do...while** es el siguiente:

```
do  sentencia  while (condición);
```

condición es una expresión entera lógica que controla la secuencia de repetición. Se repite el bucle mientras la condición es cierta.

sentencia simple o compuesta se ejecuta al menos una vez antes de evaluar la condición.

El cuerpo del bucle do…while se ejecutará al menos una vez.

Estructura de Control Iterativa

Ejemplo. Escribir los 10 primeros números naturales

```
#include <stdio.h>
void main()
{
     int contador;
     contador=1;
     do
     {
         printf (“%d \n”, contador);
         contador++;
     } while (contador <= 10) ;
}
```

Estructura de Control Iterativa

El formato general de la sentencia **for** es:

```
for ( inicialización; condición iteración; incremento)
    sentencia
```

inicialización donde se inicializa la variable de control del bucle.

condición iteración contiene una expresión lógica-relacional que hace que el bucle realice las iteraciones de las sentencias, mientras que la expresión sea verdadera.

incremento incrementa o decrementa la variable de control del bucle.

sentencia simple o compuesta (cuerpo del ciclo), se ejecutará por cada iteración del bucle. En estas sentencias **no se deben modificar** las variables que intervienen en "inicialización" o en "condición iteración"

Estructura de Control Iterativa

for (inicialización**;** condición iteración**;** incremento**)** sentencia

Orden de ejecución:

1.- se inicializan variables según el código de inicialización.

2.- se verifica la "condición iteración", si es verdadera se ejecuta la sentencia que forma el cuerpo del bucle.

4.- se ejecuta "incremento".

5.- se vuelve al paso 2 hasta que "condición iteración" sea falsa.

Estructura de Control Iterativa

Ejemplo. Escribir los 10 primeros números naturales

```
#include <stdio.h>
void main()
{
   int contador;
   for (contador=1;  contador<=10;  contador++)
   {
      printf (“%d \n”, contador);
   }

}
```

Estructura de Control Iterativa

for (inicialización**;** condición iteración**;** incremento**)** sentencia;

La condición se evalúa siempre al principio del bucle, es decir, puede que no se ejecute ninguna vez la sentencia que forma el cuerpo del bucle

Equivale a la estructura while:

```
inicialización;
while (condición iteración)
{
        sentencia;
        incremento;
}
```

Programación Estructurada

Las reglas de la programación estructurada, que estamos viendo en este curso, son:

- Todo programa consiste en una serie de acciones que se ejecutan en **secuencia, una detrás de otra.**
- Cualquier acción puede ser sustituida por dos o más en secuencia.
- Cualquier acción puede ser sustituida por cualquier estructura de control y sólo se consideran tres estructuras de control: **la secuencia, la selección y la repetición.**
- **Todas las estructuras de control tienen un solo punto de entrada y un solo punto de salida**.
- Las dos reglas anteriores pueden aplicarse tantas veces como se desee y en cualquier orden.

Estructura de Control Iterativa

INSTRUCCIONES de SALTO:

- C tiene cuatro sentencias que ejecutan un salto incondicional:

  return, goto, break y continue

- return se usa para el retorno de una función (lo estudiaremos en el tema siguiente).
- break se puede usar en la sentencia de alternativa múltiple switch

- En la programación estructurada, que estamos viendo en este curso, **no** se debe **utilizar** ninguna de las sentencias: **return, goto, break y continue**, en los bucles ni en ninguna otra parte del programa (salvo break en switch y return en las funciones y con determinadas condiciones)

Estructura de Control Iterativa

Los siguientes ciclos NO son correctos, desde el punto de vista de la programación estructurada que estamos estudiando.

```
#include <stdio.h>
 void main()
{
   int n;
   for (n = 1; n <= 10; n++)
   {
        if  (…)  n = 11;
        printf (“ …”);
   }

}
```

```
#include <stdio.h>
 void main()
{
   int n;
   for (n = 1; n <= 10; n++)
   {
        if  (…)  continue;
        printf (“ …”);
   }

}
```

```
#include <stdio.h>
 void main()
{
   int n;
   for (n = 1; n <= 10; n++)
   {
        if  (…)   break;
        printf (“ …”);
   }

}
```

Tampoco se deben modificar, en el cuerpo del bucle, ninguna de las variables que intervienen en la cabecera del for:

for (inicialización**;** condición iteración**;** incremento**)**