

Introducción al lenguaje C - II

1ºDAM IoT

Conversión de tipos

Con frecuencia, se necesita convertir un valor de un tipo a otro sin cambiar el valor que representa.

Las conversiones de tipos pueden ser:

- **Implícitas:** Se ejecutan inmediatamente, nosotros no hacemos nada.
- **Explícitas:** Solicitadas por el programador. Se suele llamar casting.

Conversión implícita

Las variables del tipo char ó short se convierten en int.

Las variables del tipo float se convierten en double.

Si alguno de los operandos es de mayor precisión que los demás , estos se convierten al tipo de aquel y el resultado es del mismo tipo.

Si no se aplica la regla anterior y un operando es del tipo unsigned el otro se convierte en unsigned y el resultado es de este tipo.

Conversión implícita

```
int main()
{
    int numeroEntero1 = 12, numeroEntero2;
    double numeroDouble1 = 4, numeroDouble2;

    // Opero con variables del mismo tipo
    numeroEntero2 = numeroEntero1 + 1;
    printf("numeroEntero2 --> %i\n", numeroEntero2); // 13

    numeroDouble2 = numeroDouble1 + 1.5;
    printf("numeroDouble2 --> %lf\n", numeroDouble2); // 5.500000

    /*Valor de Entero se convierte a Double antes de la suma*/
    numeroDouble1 = numeroEntero1 + numeroDouble1;
    printf("numeroDouble1 --> %lf\n", numeroDouble1); // 16.000000

    /*Primero hace una división entera 12/5 = 2, se convierte a double 2.0 y lo asigna a un tipo Double*/
    numeroDouble1 = numeroEntero1 / 2;
    printf("numeroDouble1 --> %lf\n", numeroDouble1); // 6.000000

    /*Convierte 5 a tipo Double, hace la división 0.800000 y lo asigna a numeroDouble1*/
    numeroDouble1 = 4.0 / 5;
    printf("numeroDouble1 --> %lf\n", numeroDouble1); // 0.800000

    return 0;
}
```

Conversión implícita

Ejercicio: Dado el siguiente código. Piensa primero que devuelve cada uno de los printf's. Copia el código y ejecutalo. ¿La salida es la esperada?

```
#include <stdio.h>

int main()
{
    float numeroFloat1 = 5, numeroFloat2=10.33, resultadoFloat;
    int numeroEntero1 = 14, resultadoEntero;

    resultadoFloat = numeroFloat1 / 2;
    printf("resultadoFloat --> %f\n", resultadoFloat);
    resultadoFloat = numeroFloat1 / 2.0;
    printf("resultadoFloat --> %f\n", resultadoFloat);

    resultadoFloat = numeroEntero1 / 3;
    printf("resultadoFloat --> %f\n", resultadoFloat);
    resultadoFloat = numeroEntero1 / 3.0;
    printf("resultadoFloat --> %f\n", resultadoFloat);

    resultadoFloat = numeroEntero1 * numeroFloat2;
    printf("resultadoFloat --> %f\n", resultadoFloat);

    resultadoEntero = numeroEntero1 * numeroFloat2;
    printf("resultadoEntero --> %i\n", resultadoEntero);

    printf("numeroEntero1 * numeroFloat2 --> %f\n", numeroEntero1 * numeroFloat2);

    return 0;
}
```

Conversión implícita

Solución: Dado el siguiente código. Piensa primero que devuelve cada uno de los printf's. Copia el código y ejecutalo. ¿La salida es la esperada?

```
#include <stdio.h>

int main()
{
    float numeroFloat1 = 5, numeroFloat2=10.33, resultadoFloat;
    int numeroEntero1 = 14, resultadoEntero;

    resultadoFloat = numeroFloat1 / 2;
    printf("resultadoFloat --> %f\n", resultadoFloat); //2.500000
    resultadoFloat = numeroFloat1 / 2.0;
    printf("resultadoFloat --> %f\n", resultadoFloat); //2.500000

    resultadoFloat = numeroEntero1 / 3;
    printf("resultadoFloat --> %f\n", resultadoFloat); //4.000000
    resultadoFloat = numeroEntero1 / 3.0;
    printf("resultadoFloat --> %f\n", resultadoFloat); //4.666667

    resultadoFloat = numeroEntero1 * numeroFloat2;
    printf("resultadoFloat --> %f\n", resultadoFloat); //144.619995

    resultadoEntero = numeroEntero1 * numeroFloat2;
    printf("resultadoEntero --> %i\n", resultadoEntero); //144

    printf("numeroEntero1 * numeroFloat2 --> %f\n", numeroEntero1 * numeroFloat2); //144.619999

    return 0;
}
```

Casting

El casting, o simplemente cast, nos permite hacer una conversión explícita de un tipo de dato a otro, a criterio del programador siempre y cuando estos tipos sean compatibles. Forzamos la conversión por así decirlo.

La forma de escribir el cast es anteponer entre paréntesis el tipo de dato al que queremos convertir una variable o resultado de una operación.

variableInt = (int) variableFloat; → Se eliminan los decimales.

Casting

Un ejemplo de uso de casting.

```
#include <stdio.h>

int main(){

    float resultadoFloat;
    int numeroEntero1 = 14;

    //numeroEntero1 y 3 son enteros, realiza la operación como enteros, y el resultado 4 lo asigna a un tipo de dato float
    resultadoFloat = numeroEntero1 / 3;
    printf("resultadoFloat --> %f\n", resultadoFloat); //4.000000

    //numeroEntero1 lo convierte a float 14.0, realiza la división como float y la asigna a resultadoFloat
    resultadoFloat = (float) numeroEntero1 / 3;
    printf("resultadoFloat --> %f\n", resultadoFloat); //4.666667

    //En este caso el resultado de numeroEntero1 / 3= 4 lo convierte a float y lo asigna a numeroEntero1
    resultadoFloat = (float) (numeroEntero1 / 3);
    printf("resultadoFloat --> %f\n", resultadoFloat); //4.000000

    return 0;
}
```


Casting

Ejercicio: Dado el siguiente código. Piensa primero que devuelve cada uno de los printf. Copia el código y ejecútalo. ¿La salida es la esperada?

```
#include <stdio.h>

int main(){

    float resultadoFloat;
    int numeroEntero1;

    numeroEntero1 = (int)19.99 + (int)11.99;
    printf("numeroEntero1 --> %i\n", numeroEntero1);

    numeroEntero1 = 19.99 + 11.99;
    printf("numeroEntero1 --> %i\n", numeroEntero1);

    printf("Número PI %i\n", (int)3.14);

    double d , e , f = 2.33 ;
    int i = 6 ;
    e = f * i ;
    printf("e --> %lf\n", e);
    d = (int) ( f * i ) ;
    printf("d --> %lf\n", d);

    float resultado;
    int resultadoInt;
    resultado = (14/3 + 1.1) * 2;
    printf("resultado --> %f\n", resultado);
    resultado = ((float)14/3 + 1.1) * 2;
    printf("resultado --> %f\n", resultado);

    resultadoInt = (14/3 + 1.1) * 2;
    printf("resultadoInt --> %i\n", resultadoInt);
    resultadoInt = ((float)14/3 + 1.1) * 2;
    printf("resultadoInt --> %i\n", resultadoInt);

    return 0;
}
```

Casting

Solución: Dado el siguiente código. Piensa primero que devuelve cada uno de los printf's. Copia el código y ejecútalo. ¿La salida es la esperada?

```
#include <stdio.h>

int main(){

    float resultadoFloat;
    int numeroEntero1;

    numeroEntero1 = (int)19.99 + (int)11.99;
    printf("numeroEntero1 --> %i\n", numeroEntero1); //30

    numeroEntero1 = 19.99 + 11.99;
    printf("numeroEntero1 --> %i\n", numeroEntero1); //31

    printf("Número PI %i\n", (int)3.14); //3

    double d , e , f = 2.33 ;
    int i = 6 ;
    e = f * i ;
    printf("e --> %lf\n", e); //13.980000
    d = (int) ( f * i ) ;
    printf("d --> %lf\n", d); //13.000000

    float resultado;
    int resultadoInt;
    resultado = (14/3 + 1.1) * 2;
    printf("resultado --> %f\n", resultado); //10.200000
    resultado = ((float)14/3 + 1.1) * 2;
    printf("resultado --> %f\n", resultado); //11.533333

    resultadoInt = (14/3 + 1.1) * 2;
    printf("resultadoInt --> %i\n", resultadoInt); //10
    resultadoInt = ((float)14/3 + 1.1) * 2;
    printf("resultadoInt --> %i\n", resultadoInt); //11

    return 0;
}
```

Operadores asignación y aritméticos

Operador asignación → =

Operadores aritméticos → +, -, *, /, %

```
int a = 10, b = 2, resultado;  
resultado = a + b;  
resultado = a - b;  
resultado = a / b;  
resultado = a * b;  
resultado = a % b; //Módulo
```

Operadores de asignación y aritméticos

Operadores de asignación y aritméticos:

- **+=** → $a+=b$; → $a = a + b$;
- **-=** → $a-=b$; → $a = a - b$;
- ***=** → $a*=b$; → $a = a * b$;
- **/=** → $a/=b$; → $a = a / b$;
- **%=** → $a+=b$; → $a = a \% b$;

```
#include <stdio.h>

int main()
{
    int resultado;
    resultado = 0;
    resultado = resultado + 1;
    printf("Resultado %d\n", resultado);
    resultado = 0;
    resultado += 1;
    printf("Resultado %d\n", resultado);
    return 0;
}
```

Resultado 1
Resultado 1

Operadores relacionales

Operador relacionales:

- menor que → **<**
- mayor que → **>**
- menor o igual que → **<=**
- mayor o igual que → **>=**
- igual que → **==**
- distinto que → **!=**

```
// 0 valor falso
// 1 valor verdadero
int a = 10, b = 2;
printf("%i\n", a < b); //0
printf("%i\n", a <= b); //0
printf("%i\n", a > b); //1
printf("%i\n", a >= b); //1
printf("%i\n", a == b); //0
printf("%i\n", a != b); //1
```

Operadores lógicos

Operadores lógicos:

- Y, AND → **&&**
- O, OR → **||**
- Negación, NOT → **!**

Operadores de incremento y decremento

Operadores de incremento y decremento:

- **++**

- $a = a + 1; \rightarrow a++;$
- $a = a + 1; \rightarrow ++a;$

- **--**

- $a = a - 1; \rightarrow a--;$
- $a = a - 1; \rightarrow --a;$

```
int a = 10, b = 2;
a++;
printf("a++ --> %d\n", a); //11
++a;
printf("++a --> %d\n", a); //12
b--;
printf("b-- --> %d\n", b); //1
--b;
printf("--b --> %d\n", b); //0
```

```
a++ --> 11
++a --> 12
b-- --> 1
--b --> 0
```

Precedencia/Prioridad de operadores

Nivel	Operadores	Orden de evaluación
1	() . [] >	izquierda-derecha
2	* & ! ~ ++ -- + - (conversión de tipo sizeof)	derecha-izquierda
3	* / %	izquierda-derecha
4	+ -	izquierda-derecha
5	<< >>	izquierda-derecha
6	< <= > >=	izquierda-derecha
7	= = !=	izquierda-derecha
8	&	izquierda-derecha
9	^	izquierda-derecha
10		izquierda-derecha
11	&&	izquierda-derecha
12		izquierda-derecha
13	?:	derecha-izquierda
14	= *= /= += -= %= <<= >>= &= ^= =	derecha-izquierda
15	,	izquierda-derecha

Operadores

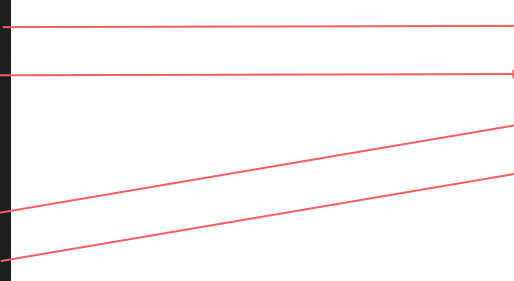
Operadores de incremento y decremento: ¡Cuidado con este caso! ¿Cuál sería el resultado?

```
int i = 1, j, k;  
j = i++;  
printf("Resultado %d\n", j);  
printf("Resultado %d\n", i);  
i = 1;  
k = ++i;  
printf("Resultado %d\n", k);  
printf("Resultado %d\n", i);
```

Precedencia/Prioridad de operadores

Operadores de incremento y decremento: ¡Cuidado con este caso! ¿Cuál sería el resultado?

```
int i = 1, j, k;  
j = i++;  
printf("Resultado %d\n", j);  
printf("Resultado %d\n", i);  
i = 1;  
k = ++i;  
printf("Resultado %d\n", k);  
printf("Resultado %d\n", i);
```



```
Resultado 1  
Resultado 2  
Resultado 2  
Resultado 2
```

Condicionales -if

Condicional simple

```
if (expresion){  
    instrucciones;  
}
```

Condicional simple

```
if (expresion)  
    1 instrucción;
```

Condicional doble

```
if (expresion){  
    instrucciones;  
}  
else{  
    instrucciones;  
}
```

Condicionales -if

Ejemplo: Dado un sueldo comprobar si es mayor que 3000.

```
#include <stdio.h>

int main()
{
    float sueldo;
    printf("Ingrese el sueldo:");
    scanf("%f", &sueldo);
    if (sueldo > 3000)
        // Sin llaves
        // Sólo hay una instrucción dentro del if
        printf("Ganas mucho");
    printf("Fin del programa");
    return 0;
}
```

```
#include <stdio.h>

int main()
{
    float sueldo;
    printf("Ingrese el sueldo:");
    scanf("%f", &sueldo);
    if (sueldo > 3000){
        //Con llaves, lo más recomendable
        //para no equivocarnos
        printf("Ganas mucho");
    }
    printf("Fin del programa");
    return 0;
}
```

Condicionales -if

Ejemplo: Dado un sueldo comprobar si es mayor que 3000 e informar también cuando sea menor.

```
#include <stdio.h>

int main()
{
    float sueldo;
    printf("Ingrese el sueldo:");
    scanf("%f", &sueldo);
    if (sueldo > 3000)
    {
        printf("Ganas mucho");
    }
    else
    {
        printf("Ya ganarás más");
    }
    printf("Fin del programa");
    return 0;
}
```

```
#include <stdio.h>

int main()
{
    float sueldo;
    printf("Ingrese el sueldo:");
    scanf("%f", &sueldo);
    if (sueldo > 3000)
        printf("Ganas mucho");
    else
        printf("Ya ganarás más");
    printf("Fin del programa");
    return 0;
}
```

Condicionales -if

Ejercicio: Solicita un número al usuario e dile si es par o impar.

Condicionales -if

Solución: Solicita un número al usuario e dile si es par o impar.

```
#include <stdio.h>

int main(){
    int numero;
    printf("Introduce un número: ");
    scanf("%i",&numero);
    if(numero%2==0){
        printf("El número es par ");
    }else{
        printf("El número es impar");
    }
    return 0;
}
```

Condicionales - if

Condicional compuesta

```
if (expresion){  
    instrucciones;  
}  
else if(expresion){  
    instrucciones;  
}  
else{  
    instrucciones;  
}
```

Condicional anidadas

```
if (expresion){  
    instrucciones;  
}  
else if(expresion){  
    if (expresion){  
        instrucciones;  
    }  
    else{  
        instrucciones;  
    }  
}  
else{  
    instrucciones;  
}
```


Condicionales - if

Ejemplo: Solicitar 3 números que se supondrán distintos. Decir cual es el mayor de ellos.

```
int num1, num2, num3;
printf("Ingrese primer valor:");
scanf("%i", &num1);
printf("Ingrese segunda valor:");
scanf("%i", &num2);
printf("Ingrese tercer valor:");
scanf("%i", &num3);
if (num1 > num2)
{
    if (num1 > num3)
    {
        printf("%i", num1);
    }
    else
    {
        printf("%i", num3);
    }
}
else
{
    if (num2 > num3)
    {
        printf("%i", num2);
    }
    else
    {
        printf("%i", num3);
    }
}
```

Otra solución utilizando operadores lógicos.

```
int num1, num2, num3;
printf("Ingrese primer valor:");
scanf("%i", &num1);
printf("Ingrese segundo valor:");
scanf("%i", &num2);
printf("Ingrese tercer valor:");
scanf("%i", &num3);
if (num1 > num2 && num1 > num3)
{
    printf("%i", num1);
}
else
{
    if (num2 > num3)
    {
        printf("%i", num2);
    }
    else
    {
        printf("%i", num3);
    }
}
```

Condicionales - if

¿Qué imprime por pantalla el siguiente código?

```
#include <stdio.h>

int main()
{
    int i = 2;
    if (i = 1)
    {
        printf("Es 1");
    }
    else if (i == 1)
    {
        printf("Es 2");
    }
    else
    {
        printf("Ni 1 ni 2");
    }
    return 0;
}
```

Condicionales - if

¿Qué imprime por pantalla el siguiente código?

```
#include <stdio.h>

int main()
{
    int i = 2;
    if (i = 1)
    {
        printf("Es 1");
    }
    else if (i == 1)
    {
        printf("Es 2");
    }
    else
    {
        printf("Ni 1 ni 2");
    }
    return 0;
}
```

En la condición hay una asignación. Como la asignación es correcta devuelve verdadero e imprime "Es 1".

Condicionales-if

Ejercicio: Solicita al usuario una nota numérica del 1 al 10 y dile la nota que ha sacado. 1-4 SUSPENSO, 5-6 APROBADO, 7-8 NOTABLE, 9-10 SOBRESALIENTE. Cualquier otra nota informa de que es errónea.

Condicionales-if

Solución: Solicita al usuario una nota numérica del 1 al 10 y dile la nota que ha sacado. 1-4 SUSPENSO, 5-6 APROBADO, 7-8 NOTABLE, 9-10 SOBRESALIENTE. Cualquier otra nota informa de que es errónea.

```
#include <stdio.h>

int main(){
    int nota;
    printf("Introduce un número: ");
    scanf("%i",&nota);
    if(nota==1 || nota==2 || nota ==3 || nota ==4){
        printf("SUSPENSO");
    }else if(nota==5 || nota==6){
        printf("APROBADO");
    }else if(nota==7 || nota==8){
        printf("NOTABLE");
    }else if(nota==9 || nota==10){
        printf("SOBRESALIENTE");
    }else{
        printf("Has introducido una nota errónea");
    }
    return 0;
}
```

Condicionales -switch

Comprueba si una expresión coincide con uno de los enteros del case. También es válido para el tipo de dato char.

La proposición “break” nos sirve para salir del switch, ya que si no lo ponemos continúa con las instrucciones del siguiente “case”.

```
switch (variable){  
    case valor1:  
        instrucciones;  
        break;  
    case valor2:  
        instrucciones;  
        break;  
    default:  
        instrucciones;  
}
```

Condicionales -switch

Ejemplo: Solicitar un valor numérico del 1 al 5 e imprimir por pantalla el valor en texto.

```
int valor;
printf("Ingrese un valor entre 1 y 5:");
scanf("%i", &valor);
switch (valor)
{
    case 1:
        printf("Uno");
        break;
    case 2:
        printf("Dos");
        break;
    case 3:
        printf("Tres");
        break;
    case 4:
        printf("Cuatro");
        break;
    case 5:
        printf("Cinco");
        break;
    default:
        printf("El valor esta fuera de rango");
}
```

Importante añadir el “break” para salir de ese caso cuando terminemos.

Opción por defecto, aquí entramos cuando no se cumple ninguna de las anteriores. No es necesario break al ser la última opción.

Condicionales -switch

¿Qué imprimirá por pantalla el siguiente código?

```
#include <stdio.h>

int main()
{
    int i = 2;
    switch (i)
    {
        case 1:
            printf("Es un 1");
            break;
        case 2:
            printf("Es un 2");
        default:
            printf("No es ni 1 ni 2");
            break;
    }
    return 0;
}
```


Condicionales -switch

¿Qué imprimirá por pantalla el siguiente código?

```
#include <stdio.h>

int main()
{
    int i = 2;
    switch (i)
    {
        case 1:
            printf("Es un 1");
            break;
        case 2:
            printf("Es un 2");
        default:
            printf("No es ni 1 ni 2");
            break;
    }
    return 0;
}
```

Imprime:

Es un 2No es ni 1 ni 2

Observa que entra en el case 2 pero después del printf no encuentra break por lo que sigue su ejecución.

Escribe las dos frases juntas porque no hemos puesto \n.

Condicionales -switch

Un ejemplo evaluando una variable de tipo char.

```
#include <stdio.h>

int main()
{
    char i = 'A';
    switch (i)
    {
        case 'A':
            printf("Es una A");
            break;
        case 'B':
            printf("Es una B");
            break;
        default:
            printf("Ni es una A ni es una B");
            break;
    }
    return 0;
}
```

Condicionales -switch

Ejercicio: Solicita al usuario una nota numérica del 1 al 10 y dile la nota que ha sacado. 1-4 SUSPENSO, 5-6 APROBADO, 7-8 NOTABLE, 9-10 SOBRESALIENTE. Cualquier otra nota informa de que es errónea.

Condicionales -switch

Solución: Solicita al usuario una nota numérica del 1 al 10 y dile la nota que ha sacado. 1-4 SUSPENSO, 5-6 APROBADO, 7-8 NOTABLE, 9-10 SOBRESALIENTE. Cualquier otra nota informa de que es errónea.

```
int nota;  
printf("Introduce la nota numérica: ");  
scanf("%i",&nota);  
switch (nota)  
{  
    case 1:  
        printf("SUSPENSO");  
        break;  
    case 2:  
        printf("SUSPENSO");  
        break;  
    case 3:  
        printf("SUSPENSO");  
        break;  
    case 4:  
        printf("SUSPENSO");  
        break;  
    case 5:  
        printf("APROBADO");  
        break;  
    case 6:  
        printf("APROBADO");  
        break;  
    case 7:  
        printf("NOTABLE");  
        break;  
    case 8:  
        printf("NOTABLE");  
        break;  
    case 9:  
        printf("SOBRESALIENTE");  
        break;  
    case 10:  
        printf("SOBRESALIENTE");  
        break;  
    default:  
        printf("Has introducido una nota errónea");  
        break; //Este break no haría falta  
}
```

Condicionales -switch

OTRA Solución: Solicita al usuario una nota numérica del 1 al 10 y dile la nota que ha sacado. 1-4 SUSPENSO, 5-6 APROBADO, 7-8 NOTABLE, 9-10 SOBRESALIENTE. Cualquier otra nota informa de que es errónea.

Puedo jugar con los breaks. Observa, que hay casos en los que no se hace nada y como no tienen break continuaría la ejecución hasta encontrar el siguiente break.

```
#include <stdio.h>

int main(){
    int nota;
    printf("Introduce la nota numérica: ");
    scanf("%i",&nota);
    switch (nota)
    {
        case 1:
        case 2:
        case 3:
        case 4:
            printf("SUSPENSO");
            break;
        case 5:
        case 6:
            printf("APROBADO");
            break;
        case 7:
        case 8:
            printf("NOTABLE");
            break;
        case 9:
        case 10:
            printf("SOBRESALIENTE");
            break;
        default:
            printf("Has introducido una nota errónea");
            break; //Este break no haría falta
    }
    return 0;
}
```

Estructuras repetitivas - while

Mientras que se cumple una expresión se ejecuta el bucle.

```
while (expresion){  
    instrucciones;  
}
```

```
int x;  
x = 1;  
while (x <= 100)  
{  
    printf("%i", x);  
    printf(" - ");  
    x = x + 1;  
}
```

Estructuras repetitivas - while

Ejercicio: Pregunta al usuario cuantos números va a introducir. Realiza el sumatorio todos los números e imprímelos al final por pantalla.

Estructuras repetitivas – while

Solución: Pregunta al usuario cuantos números va a introducir. Realiza el sumatorio todos los números e imprímelos al final por pantalla.

```
#include <stdio.h>

int main()
{
    int x, cantidad, sumatorio, numero;
    sumatorio = 0;
    printf("¿Cuántos números vas a introducir?");
    scanf("%i", &cantidad);
    x = 1;
    while (x <= cantidad)
    {
        printf(">");
        scanf("%i", &numero);
        sumatorio += numero;
        x = x + 1;
    }
    printf("Has introducido %i números y la suma es de %i", cantidad, sumatorio);
    return 0;
}
```


Estructuras repetitivas - do ... while

Mientras que se cumple la condición se ejecuta el bucle. Pero al menos se ejecuta una vez.

```
do {  
    instrucciones;  
} while (expresion);
```

```
int x;  
x = 1;  
do  
{  
    printf("%i", x);  
    printf(" - ");  
    x = x + 1;  
} while (x <= 100);  
// Retroceso, para que no aparezca el último guión  
printf("\b\b ");
```

Estructuras repetitivas – do ... while

Ejercicio: Pide al usuario un carácter, hasta que el carácter introducido sea una x (mayúscula o minúscula).

Estructuras repetitivas – do ... while

Solución: Pide al usuario un carácter, hasta que el carácter introducido sea una x (mayúscula o minúscula).

```
#include <stdio.h>

int main()
{
    char character;
    do
    {
        printf(">");
        scanf("%c", &character);
        fflush(stdin);
    } while (character != 'x' && character != 'X');
    printf("Salgo del programa");
    return 0;
}
```

Estructuras repetitivas - for

Mientras que se cumple la condición se ejecuta el bucle. Pero al menos se ejecuta una vez.

```
for ( expresión1 ; expresión2 ; expresion3 ) {  
    instrucciones;  
}
```

- expresión1 → Inicialización variable
- expresión2 → Condición de continuidad
- expresión3 → Incremento

```
int f;  
for (f = 1; f <= 100; f++)  
{  
    printf("%i", f);  
    printf("-");  
}  
// Borro el último guión  
printf("\b ");  
return 0;
```

Estructuras repetitivas - for

¡Cuidado! Las 3 expresiones del for no son obligatorias. Yo os recomiendo escribirlas siempre.

- `for (; exp2 ; exp3)`
- `for (exp1 ; ;)`
- `for (; ;)`

Estructuras repetitivas - for

Ejercicio: Solicita un número al usuario. Muestra para los 20 primeros números si son divisibles por el número dado por el usuario.

```
Introduce un número
5
El número 1 no es divisible por 5
El número 2 no es divisible por 5
El número 3 no es divisible por 5
El número 4 no es divisible por 5
El número 5 es divisible por 5
El número 6 no es divisible por 5
El número 7 no es divisible por 5
El número 8 no es divisible por 5
El número 9 no es divisible por 5
El número 10 es divisible por 5
El número 11 no es divisible por 5
El número 12 no es divisible por 5
El número 13 no es divisible por 5
```

Estructuras repetitivas – for

Solución: Solicita un número al usuario. Muestra para los 20 primeros números si son divisibles por el número dado por el usuario.

```
#include <stdio.h>

int main()
{
    int numero;
    printf("Introduce un número\n");
    scanf("%i", &numero);
    int f;
    for (f = 1; f <= 100; f++)
    {
        if (f % numero == 0)
        {
            printf("El número %i es divisible por %i\n", f, numero);
        }
        else
        {
            printf("El número %i no es divisible por %i\n", f, numero);
        }
    }
    return 0;
}
```