

ILERNA

Online

Videotutoría 7 (UF2): Programación modular

Módulo 03A: Programación

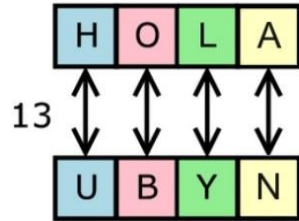


PAC Desarrollo

En el siglo I a.C., apareció un **cifrado por sustitución** conocido con el nombre genérico de **código César**.

Se trata de un *cifrado por sustitución*. Como tal, consiste en reasignar a cada letra del abecedario otra nueva resultante de desplazar éste un determinado número de lugares.

Para cifrar la palabra "HOLA", queremos realizar un desplazamiento de 13 unidades:



Se pide, realizar un programa que encripte un texto mediante el cifrado César. El usuario introducirá el texto, y el número de desplazamiento.

Ejercicio 2:

Realizar un programa que nos permita eliminar una posición dentro de una matriz dada la posición a eliminar.

Por ejemplo: Si tengo una matriz de este estilo:

```
1 1 1
1 1 1
1 1 1
```

Y queremos eliminar la posición (0,0), el resultado sería:

```
0 1 1
1 1 1
1 1 1
```

Únete al reto




¿Qué pasos debemos seguir?

- 1º) Crear un tablero de juego 3x3
- 2º) Ir rellenando ese tablero para inicializarlo
- 3º) Vamos a obtener la posición donde queremos poner nuestra 'X' o 'O'
- 4º) Para realizar el punto 3, debemos comprobar si podemos escribir en esa casilla
- 5º) Conocer el turno del jugador
- 6º) Comprobar quién ha ganado

UFI: Preguntas

¿Qué sentencia falta para que salga la salida indicada?

```
class BreakTest
{
    static void Main()
    {
        for (int i = 1; i <= 100; i++)
        {
            if (i == 5)
            {
                 ; 
            }
            Console.WriteLine(i);
        }

        // Keep the console open in debug mode.
        Console.WriteLine("Press any key to exit.");
        Console.ReadKey();
    }
}
/*
Output:
1
2
3
4
*/
```

- a) Continue;
- b) Break;**
- c) Return
- d) Goto

UFI: Preguntas

¿Qué tipo de operador es %?

- a) Aritmético
- b) Lógico
- c) De comparación
- d) booleano

UFI: Preguntas

¿Qué tipo de error es el que muestra la imagen?

```
0 referencias
static void Main(string[] args)
{
    int i, j;

    for (i = 1; i <5; i++)
    {
        for (j = 1; j <5; j++)
        }
    Console.ReadKey();
}
```

El término de expresión '}' no es válido
Se esperaba ;

UFI: Preguntas

En un bucle do-while:

- a) Entraremos 0 o más veces dependiendo de la condición del while
- b) Entraremos como mínimo una vez
- c) Entraremos 0 o más veces dependiendo de la condición del do
- d) Ninguna respuesta es correcta

UFI: Preguntas

¿Es correcto este código?

```
0 referencias
static void Main(string[] args)
{
    String cadena = "estudio en Ilernaonline";
    int i = cadena.Length;
    Console.WriteLine(i);
}
```

- a) No. No podemos asignar el método Length a una variable entera
- b) Sí. La salida será un número entero que indica el tamaño de la cadena
- c) No. Se produce un error en tiempo de ejecución
- d) Sí. La salida será un número entero que indica el tamaño de la cadena hasta el primer espacio en blanco

UF1: Preguntas

Realiza un pequeño código donde se explique la sentencia switch

UF1: Preguntas

Realiza un pequeño código donde se explique la sentencia switch

```
0 referencias
static void Main(string[] args)
{
    Console.WriteLine("Ingrese un valor entre 1 y 5:");
    int valor = int.Parse(Console.ReadLine());
    switch (valor)
    {
        case 1:
            Console.WriteLine("uno");
            break;
        case 2:
            Console.WriteLine("dos");
            break;
        case 3:
            Console.WriteLine("tres");
            break;
        case 4:
            Console.WriteLine("cuatro");
            break;
        case 5:
            Console.WriteLine("cinco");
            break;
        default:
            Console.WriteLine("Se ingreso un valor fuera de rango");
            break;
    }
    Console.ReadKey();
}
```

UFI: Preguntas

¿Qué elemento puede no existir en una sentencia de selección múltiple?

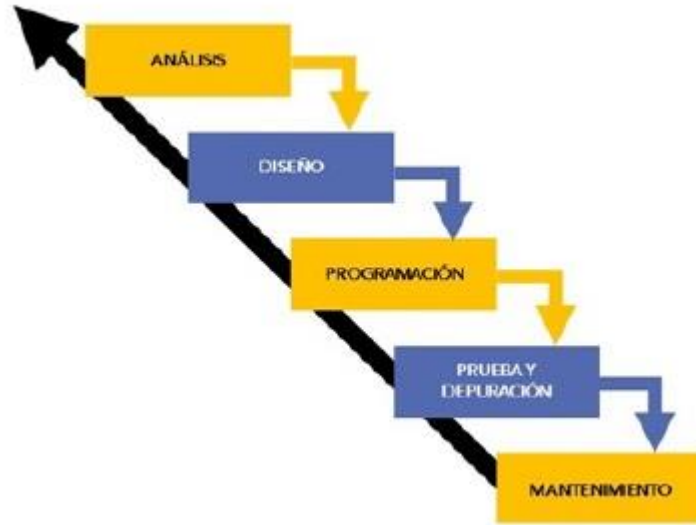
- a) La expresión.
- b) Los case.
- c) La cláusula default.
- d) El break de dos o más cases

UFI: Preguntas

¿Cuál es la primera fase del ciclo de vida de un programa informático?

UFI: Preguntas

¿Cuál es la primera fase del ciclo de vida de un programa informático?



UF1: Preguntas

¿Cuál es la salida del siguiente código?

```
static void Main(string[] args)
{
    int x = 10;
    int y = 9;

    if (x>y && x)
    {
        Console.WriteLine(x);
    }

    Console.ReadKey();
}
```

UFI: Preguntas

Según el siguiente código responde a las siguientes cuestiones:

```
0 referencias
static void Main(string[] args)
{
    string cadena;
    int desplazamiento;
    Console.WriteLine("Introduzca cadena");
    cadena = Console.ReadLine();
    Console.WriteLine("-Desplazamiento");
    desplazamiento = int.Parse(Console.ReadLine());

    if (desplazamiento > 26) {
        desplazamiento %= 26;

        for (int i = 0; i < cadena [ ] i++) {
            Console.Write("{0}", ( [ ])(cadena[i] + desplazamiento));
        }
    }

    Console.ReadKey();
}
```

Según el código indicado, ¿qué instrucción falta en el primer hueco?

- a) charAt()
- b) indexOf()
- c) substring()
- d) length()

UFI: Preguntas

Según el siguiente código responde a las siguientes cuestiones:

```
0 referencias
static void Main(string[] args)
{
    string cadena;
    int desplazamiento;
    Console.WriteLine("Introduzca cadena");
    cadena = Console.ReadLine();
    Console.WriteLine("-Desplazamiento");
    desplazamiento = int.Parse(Console.ReadLine());

    if (desplazamiento > 26) {
        desplazamiento %= 26;

        for (int i = 0; i < cadena [ ] i++) {
            Console.Write("{0}", ( [ ])(cadena[i] + desplazamiento));
        }
    }

    Console.ReadKey();
}
```

Según el código indicado, ¿qué instrucción falta en el segundo hueco?

- a) (char)
- b) cadena[i]
- c) (double)
- d) (int)

UFI: Preguntas

Según el siguiente código responde a las siguientes cuestiones:

```
0 referencias
static void Main(string[] args)
{
    string cadena;
    int desplazamiento;
    Console.WriteLine("Introduzca cadena");
    cadena = Console.ReadLine();
    Console.WriteLine("-Desplazamiento");
    desplazamiento = int.Parse(Console.ReadLine());

    if (desplazamiento > 26) {
        desplazamiento %= 26;

        for (int i = 0; i < cadena.Length; i++) {
            Console.Write("{0}", (cadena[i] + desplazamiento));
        }
    }

    Console.ReadKey();
}
```

La línea 'desplazamiento %=26;', ¿es correcta?

- a) No existe este tipo de instrucciones en C#.
- b) Sí. Estamos realizando un % de una variable
- c) Sí. Estamos realizando una división y guardando el resto de esta
- d) Sí. Estamos realizando una división y guardando el cociente de esta

UFI: Preguntas

Según el siguiente código responde a las siguientes cuestiones:

```
0 referencias
static void Main(string[] args)
{
    string cadena;
    int desplazamiento;
    Console.WriteLine("Introduzca cadena");
    cadena = Console.ReadLine();
    Console.WriteLine("-Desplazamiento");
    desplazamiento = int.Parse(Console.ReadLine());

    if (desplazamiento > 26) {
        desplazamiento %= 26;

        for (int i = 0; i < cadena  i++) {
            Console.Write("{0}", ((cadena[i] + desplazamiento));
        }
    }

    Console.ReadKey();
}
```

¿Qué realiza el código?

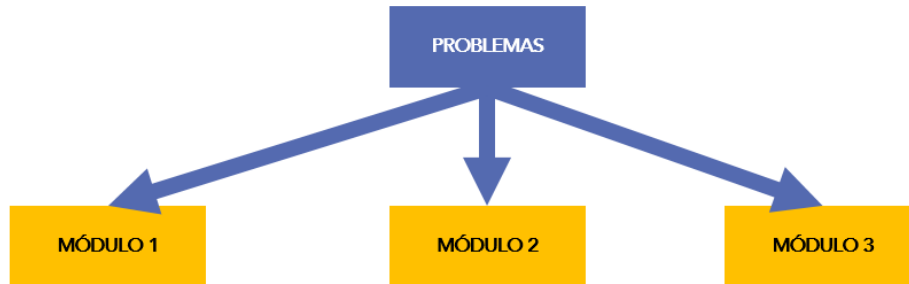
UF2: Programación modular

Concepto

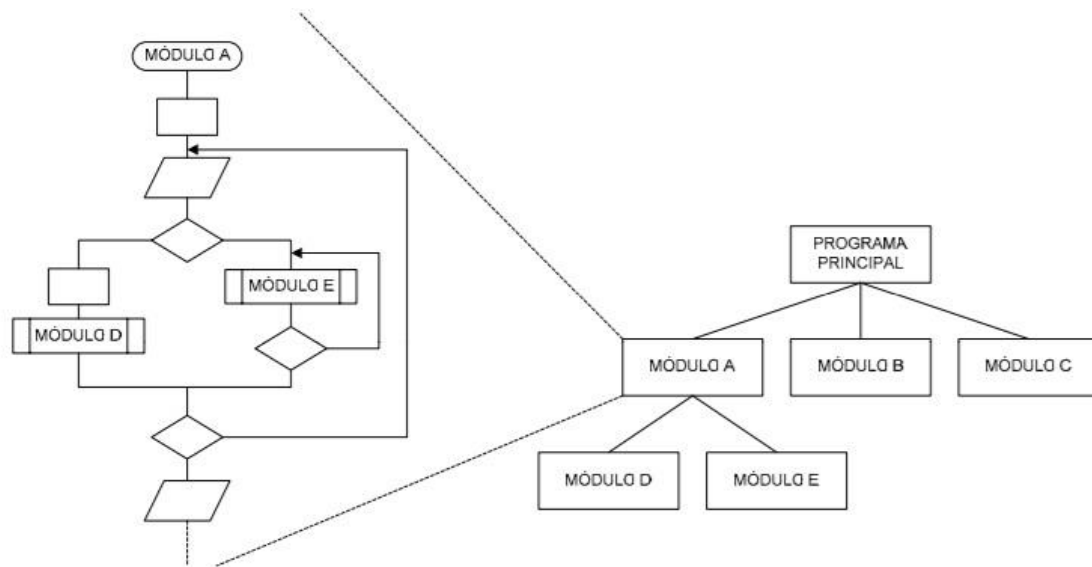
- La programación modular consiste en dividir el problema original en diversos subproblemas, que se pueden resolver por separado, para, después, recomponer los resultados y obtener la solución al problema.
- Un subproblema se denomina módulo (función o procedimiento), y es una parte del problema que se puede resolver de manera independiente.

Top down

- El diseño descendente es una técnica que permite diseñar la solución de un problema con base en la modularización o segmentación, dándole un enfoque de arriba hacia abajo (top down).
- Esta solución se divide en módulos que se estructuran e integran jerárquicamente.
- Este diseño se basa en el principio "divide y vencerás".



Ejemplo



Criterios de descomposición modular

- Tiene que haber una relación entre el tamaño de los módulos y la complejidad de la aplicación
- Si un programa se descompone en demasiadas unidades, decrece la efectividad: **Si un programa se descompone en demasiadas unidades, decrece la efectividad.**
- *Algunos criterios para su descomposición (recomendaciones):*
 1. Descomposición por tamaño (aprox. 50 líneas/módulo)
 2. Niveles de anidamiento (aprox. menos de 7 niveles)
 3. Un módulo debe realizar una única tarea y comunicarse lo menos posible con el resto de módulos Si aún habiéndolo dividido puede hacer varias tareas es que aún se puede dividir más. Cuanto más independiente mejor.

Módulos

FUNCIONES

PROCEDIMIENTOS

Módulo = Método

FUNCIÓN si declaramos la función int, el dato que declare de vuelta también ha de serlo

```
Ámbito de la declaración Tipo Función
Nombre_función (parámetros) {
//declaración de variables locales
//Instrucciones
//retorno del tipo
}
```

Utilizamos la palabra reservada *function* con una lista de parámetros (variables) que vamos a utilizar en ellas

Cuando la función llega a su fin, retornará un valor del mismo tipo de la función con la directiva *return* (SIEMPRE)

PROCEDIMIENTO

```
Ámbito de la declaración void Se declara vacío.
Nombre_procedimiento{
// Instrucciones
}
```

Un procedimiento no devuelve un valor y por tanto no incluye la directiva *return*

No nos devuelve ningún tipo de parámetro el procedimiento, ni hay que declararla de ningún tipo. Por lo tanto no incluye la directiva *return*.

