

IES Celia Viñas - 1º ASI – Fundamentos de Programación
Más ejercicios resueltos de la Unidad 1
Ejercicios de Programación Modular (números 56 a 70)

Ejercicio 56 – Averiguar qué hace el algoritmo

Se trata de una calculadora muy simple, capaz sólo de realizar sumas y restas. Se utiliza escribiendo primero la operación que se desea realizar mediante un único carácter ('+' para la suma y '-' para la resta). Seguidamente, se introducen los dos operandos y se efectúa la operación mediante una llamada a la función "sumar" o a la función "restar", dependiendo de cuál haya sido el operador.

El proceso se repite hasta que se introduce como operación una 'Q'. Si la operación no es '+', '-' ni 'Q', el programa emite un mensaje de error.

Ejercicio 58a – Calcular salario – Solución con paso de parámetros por VALOR

```
real función calcular_salario (num_horas_trabajadas es entero, precio_por_hora es real)
variables
    h_extra es entero
    salario es real
inicio
    h_extra = 40 - num_horas_trabajadas // Calcula horas extra
    si (h_extra > 0) // Si hay horas extra, las restamos de las normales
        num_horas_trabajadas = num_horas_trabajadas - h_extra

    salario = num_horas_trabajadas * precio_por_hora // Calcula salario de horas normales
    si (h_extra > 0) // Calcula salario de las horas extra
        salario = salario + (h_extra * precio_por_hora * 1.5)

    return (salario)
fin
```

Ejercicio 58b – Calcular salario – Solución con paso de parámetros por REFERENCIA

```
real procedimiento calcular_salario (num_horas_trabajadas es entero, precio_por_hora es real,
                                     *salario es real)
variables
    h_extra es entero
inicio
    h_extra = 40 - num_horas_trabajadas // Calcula horas extra
    si (h_extra > 0) // Si hay horas extra, las restamos de las normales
        num_horas_trabajadas = num_horas_trabajadas - h_extra

    salario = num_horas_trabajadas * precio_por_hora // Calcula salario de horas normales
    si (h_extra > 0) // Calcula salario de las horas extra
        salario = salario + (h_extra * precio_por_hora * 1.5)
fin
```

Ejercicio 59 – Modificación del ejercicio 58

```
algoritmo calcular_salario_e_impuestos

variables
    h, n, i son enteros
    sbruto, sneto, imp, p son reales

inicio
    escribir("¿Cuántos trabajadores hay en la empresa?")
    leer(n)
    para i desde 1 hasta n hacer
        inicio
            escribir("Procesando el trabajador nº ", i)
            escribir("¿Cuántas horas ha trabajado a la semana?")
            leer(h)
            escribir("¿Cuánto cobra por cada hora trabajada?")
            leer(p)
            sbruto = calcular_salario_bruto(h, p)
            imp = calcular_impuestos(sbruto)
            sneto = calcular_salario_neto(sbruto, imp)
            escribir("El salario neto del trabajador es: ", sneto)
```

```

    fin      // fin del para
fin

real función calcular_salario_bruto (num_horas es entero, precio es real)
variables
    sal_bruto es real
inicio
    si (num_horas <= 40)                                // No hay horas extra
        sal_bruto = num_horas * precio
    si_no                                             // Sí hay horas extra
        sal_bruto = (40 * precio) + (num_horas - 40) * precio * 1.5

    devolver (sal_bruto)
fin

real función calcular_impuestos (salario es real)
variables
    imp es real
inicio
    si (salario > 1000)
        imp = 15
    si_no
        imp = 10

    devolver (imp)
fin

real función calcular_salario_netto (sbruto es real, imp es real)
variables
    sneto es real
inicio
    sneto = sbruto * imp / 100
    devolver (sneto)
fin

```

Ejercicio 60 – Polígonos

```

algoritmo polígonos
variables
    nlados es entero
    long_lado, apotema son reales
    nombre es cadena
inicio
    // Entrada de datos
    repetir                                // El n° de lados debe estar entre 3 y 12
    inicio
        escribir("Número de lados:")
        leer(nlados)
    fin
    mientras (nlados < 3) || (nlados > 12)

    escribir("Longitud del lado:")
    leer(long_lado)
    escribir("Longitud de la apotema:")
    leer(apotema)

    // Procesamiento de los datos
    area = calcular_área(nlados, long_lado, apotema)
    perimetro = calcular_perimetro(nlados, long_lado)
    nombre = nombre_poligono(nlados)

    // Salida de datos
    escribir("El polígono se llama ", nombre)
    escribir("Su área es:", area)
    escribir("Su perímetro es:", perimetro)
fin

// Esta función calcula el área del polígono y la devuelve
real función calcular_área(nl es entero, long es real, apotema es real)
variables

```

```

    a es real
inicio
    a = nl * long * apotema / 2
    devolver (a)
fin

// Esta función calcula el perímetro del polígono y lo devuelve
real función calcular_perímetro(nl es entero, long es real)
variables
    p es real
inicio
    p = nl * long
    devolver (p)
fin

// Esta función determina el nombre del polígono y lo devuelve
cadena función nombre_polígono(nl es entero)
variables
    nombre es cadena
inicio
    según (nl) hacer
        inicio
            3: nombre = "Triángulo"
            4: nombre = "Cuadrado"
            5: nombre = "Pentágono"
            6: nombre = "Hexágono"
            7: nombre = "Heptágono"
            8: nombre = "Octógono"
            9: nombre = "Eneágono"
            10: nombre = "Decágono"
            11: nombre = "Endecágono"
            12: nombre = "Dodecágono"
        fin
    devolver (nombre)
fin

```

Ejercicio 61 – Devolución de monedas – Solución 1

```

algoritmo máquina_expendedora_2
variables
    dinero es entero
    d es real
    mon2e, mon1e son enteros
    mon50c, mon20c, mon10c son enteros
    mon5c, mon2c, mon1c son enteros
inicio
    escribir("¿Qué cantidad hay que devolver?")
    leer(d)
    dinero = d * 100
    escribir("Hay que devolver:")

    mon2e = calcular_monedas(&dinero, 200)
    escribir("Monedas de 2 euros:", mon2e)

    mon1e = calcular_monedas(&dinero, 100)
    escribir("Monedas de 1 euro:", mon1e)

    mon50c = calcular_monedas(&dinero, 50)
    escribir("Monedas de 50 cénts:", mon50c)

    mon20c = calcular_monedas(&dinero, 20)
    escribir("Monedas de 20 cénts:", mon20c)

    mon10c = calcular_monedas(&dinero, 10)
    escribir("Monedas de 10 cénts:", mon10c)

    mon5c = calcular_monedas(&dinero, 5)
    escribir("Monedas de 5 cénts:", mon5c)

    mon2c = calcular_monedas(&dinero, 2)

```

```

    escribir("Monedas de 2 cénts:", mon2c)

    mon1c = calcular_monedas(&dinero, 1)
    escribir("Monedas de 1 cént:", mon1c)
fin

/* Calcula la cantidad de monedas que hay que devolver del valor especificado en el argumento.
Modifica el valor de "dinero" para que el siguiente subalgoritmo haga correctamente sus
cálculos. Por eso "dinero" se pasa por referencia */
entero función calcular_monedas (*dinero es real, valor_moneda es real)
variables
    monedas es entero
inicio
    si (*dinero div valor_moneda != 0) entonces
        inicio
            monedas = *dinero div valor_moneda
            *dinero = *dinero - (monedas * valor_moneda)
        fin
    devolver(monedas)
fin

```

Ejercicio 61 – Devolución de monedas – Solución 2

```

algoritmo máquina_expendedora
variables
    d es real
    dinero es entero
    mon2e, mon1e son enteros                /* Monedas de 2 y 1 euro */
    mon50c, mon20c, mon10c son enteros      /* Monedas de 50, 20 y 10 céntimos */
    mon5c, mon2c, mon1c son enteros        /* Monedas de 5, 2, y 1 céntimos */
inicio
    escribir("¿Qué cantidad hay que devolver?")
    leer(d)
    dinero = d * 100                        /* Para evitar los decimales y poder usar div */
    calcular_devolución(dinero, &mon2e, &mon1e, &mon50c, &mon20, &mon10c, &mon5c, &mon2c, &mon1c)
    escribir("Hay que devolver:")
    escribir("Monedas de 2 euros:", mon2e)
    escribir("Monedas de 1 euro:", mon1e)
    escribir("Monedas de 50 cénts:", mon50c)
    escribir("Monedas de 20 cénts:", mon20c)
    escribir("Monedas de 10 cénts:", mon10c)
    escribir("Monedas de 5 cénts:", mon5c)
    escribir("Monedas de 2 cénts:", mon2c)
    escribir("Monedas de 1 cént:", mon1c)
fin

/* Este subalgoritmo calcula la devolución. Todos los parámetros se pasan por variable (menos
dinero) para poder devolver en ellos la cantidad de cada tipo de moneda */
procedimiento calcular_devolución(dinero es real, *mon2e, *mon1e, *mon50c, *mon20,
                                *mon10c, *mon5c, *mon2c, *mon1c son enteros)

inicio
    si (dinero div 200 != 0) entonces                /* Monedas de 2 euros */
        inicio
            *mon2e = dinero div 200
            dinero = dinero - (*mon2e) * 200
        fin
    si (dinero div 100 != 0) entonces                /* Monedas de 1 euro */
        inicio
            *mon1e = dinero div 100
            dinero = dinero - (*mon1e) * 100
        fin
    si (dinero div 50 != 0) entonces                /* Monedas de 50 céntimos */
        inicio
            mon50c = dinero div 50
            dinero = dinero - (*mon50c) * 50
        fin
    si (dinero div 20 != 0) entonces                /* Monedas de 20 céntimos */
        inicio
            *mon20c = dinero div 20
            dinero = dinero - (*mon20c) * 20
    fin

```

```

fin
si (dinero div 10 != 0) entonces                                /* Monedas de 10 céntimos */
inicio
    *mon10c = dinero div 10
    dinero = dinero - (*mon10c) * 10
fin
si (dinero div 5 != 0) entonces                                /* Monedas de 5 céntimos */
inicio
    *mon5c = dinero div 5
    dinero = dinero - (*mon5c) * 5
fin
si (dinero div 2 != 0) entonces                                /* Monedas de 2 céntimos */
inicio
    *mon2c = dinero div 2
    dinero = dinero - (*mon2c) * 2
fin
si (dinero > 0) entonces                                        /* Lo que nos quede, en monedas de 1 céntimo */
    *mon1c = dinero div 1                                        /* Ni siquiera haría falta hacer la división */
fin

```

Ejercicio 62 – Predicción meteorológica

```

algoritmo predicción_meteorológica
variables
    presion, humedad son caracteres
    predicción es cadena
inicio
    escribir("¿Cómo es la presión atmosférica actual (A = alta, B = baja, M = media)?")
    leer(presion)
    escribir("¿Cómo es la humedad relativa actual (A = alta, B = baja, M = media)?")
    leer(humedad)
    prediccion = predecir_lluvia(presion, humedad)
    escribir("La probabilidad de lluvia es: ", predicción)
    prediccion = predecir_sol(presion, humedad)
    escribir("La probabilidad de que haga sol es: ", predicción)
    prediccion = predecir_frio(presion, humedad)
    escribir("La probabilidad de haga frio es: ", predicción)
fin

cadena función predecir_lluvia (P, H son caracteres)
variables
    result es cadena
inicio
    result = "Baja"
    si (P == "B") entonces                                     /* Resultado más general */
        inicio                                                 /* El resto de la función sirve */
            según (H) hacer                                     /* para ver si se modifica ese */
                inicio                                         /* resultado más general */
                    "A": result = "Muy alta"
                    "B": result = "Media"
                    "M": result = "Alta"
            fin
        fin
    si_no
        inicio
            si (P == "M") y (H == "M") entonces
                result = "Media"
            si_no
                result = "Baja"
            fin
    devolver (result)
fin

cadena función predecir_sol (P, H son caracteres)
variables
    result es cadena
inicio
    result = "Alta"
    si (P == "B") entonces                                     /* Resultado más general */

```

```

    inicio
        si (H == "A") entonces
            result = "Baja"
        si_no
            result = "Media"
    fin
    si_no
        inicio
            si (P = "M") y (H = "M") entonces
                result = "Media"
            si_no
                result = "Alta"
        fin
    devolver (result)
fin

cadena función predecir_frio (P, H son caracteres)
variables
    result es cadena
inicio
    result = "Baja" /* Resultado más general */
    si (P == "B") entonces
        inicio
            si (H == "A") entonces
                result = "Alta"
            si (H == "M") entonces
                result = "Alta"
        fin
    si (P == "M") entonces
        inicio
            si (H = "A") entonces
                result = "Alta"
            si (H = "M") entonces
                result = "Media"
        fin
    devolver (result)
fin

```

Ejercicio 63 – Reloj

```

algoritmo reloj_continuo
variables
    h, m, s son enteros
inicio
    escribir ("Introduzca la hora, el minuto y el segundo actual:")
    repetir
        leer(h, m, s) /* Comprobaremos la corrección de los datos leídos*/
    mientras que (h < 0) o (h > 23) o (m < 0) o (m > 59) o (s < 0) o (s > 59)

    repetir /* Comienzo del bucle del reloj */
        inicio
            esperar(1) /* Esperamos un segundo */
            actualizar_reloj(&h,&m,&s) /* Se incrementa la hora en un segundo */
            escribir("Hora actual: ", h, ":", m, ":", s)
        fin
    mientras que (0 == 0) /* Forzamos un bucle infinito */
fin

/* Este subalgoritmo incrementa la hora que se le pasa en un segundo. Los tres datos se pasan
por variable, porque los tres son susceptibles de modificarse */
procedimiento actualizar_reloj(*h, *m, *s son enteros)
inicio
    s = s + 1
    si (s >= 60) entonces
        inicio
            s = 0
            m = m + 1
            si (m >= 60) entonces
                inicio
                    h = h + 1
                fin
            fin
        fin
    fin

```

```

        si (h >= 24) entonces
            h = 0
        fin
    fin
fin

```

Ejercicio 64 – Caracteres ASCII

```

algoritmo escribir_códigos_ASCII
variables
    numero es enteros
    letra es carácter
inicio
    repetir
        inicio
            escribir ("Introduzca un número entre 0 y 255 (negativo para terminar)")
            leer(numero)
            si (numero >= 0) y (numero <=31) entonces /* Rango de caracteres no imprimibles*/
                escribir("Código ASCII no imprimible")
            si (numero >= 32) y (numero <=255) entonces /* Rango de caracteres imprimibles
        */
        inicio
            letra = carácter_ascii(numero)
            escribir("La letra que corresponde al código introducido es: ", letra)
        fin
        si (numero > 255) entonces /* Fuera de rango:imprimir tabla */
            escribir_tabla_ASCII() /* Lo haremos en un subalgoritmo */
        fin
    hasta que (numero < 0)
fin

procedimiento escribir_tabla_ASCII() /* No tiene argumentos */
variables
    i es entero
    letra es carácter
inicio
    escribir("La tabla de códigos ASCII imprimibles es:")
    para i desde 32 hasta 255 hacer
        inicio
            letra = carácter_ascii(i)
            escribir("Código: ", i, " - Letra: ", letra)
        fin
    fin
fin

```

Ejercicio 65 – Conversor de unidades de medida de información – Solución 1

```

algoritmo convertir_unidades
variables
    cantidad es real
    unidad_inicio es cadena
    unidad_destino es cadena
inicio
    escribir("Introduzca la cantidad de información:");
    leer(cantidad)
    escribir("¿En qué unidad está expresada?")
    leer(unidad_inicio)
    escribir("¿A qué unidad desea convertirla?")
    leer(unidad_destino)

    /* Convertimos la cantidad a bits */
    cantidad = convertir_a_bits(cantidad, unidad_origen)
    /* Convertimos los bits en la unidad buscada */
    cantidad = convertir_a_destino(cantidad, unidad_destino)

    escribir("El resultado es: ", cantidad)
fin

/* Convierte la cantidad (expresada en la unidad de medida que se pasa como parámetro) a bits
*/

real función convertir_a_bits(cantidad es real, unidad es cadena)

```

```

variables
    result es real
inicio
    según (unidad) hacer
        inicio
            "bit": result = cantidad
            "byte": result = cantidad * 8
            "KB": result = cantidad * 1024 * 8
            "MB": result = cantidad * 1024 * 1024 * 8
            "GB": result = cantidad * 1024 * 1024 * 1024 * 8
            "TB": result = cantidad * 1024 * 1024 * 1024 * 1024 * 8
        fin
    devolver (result)
fin

/* Convierte la cantidad (expresada bits) a la unidad de medida pasada como parámetro */
real función convertir_a_destino(cantidad es real, unidad es cadena)
variables
    result es real
inicio
    según (unidad) hacer
        inicio
            "bit": result = cantidad
            "byte": result = cantidad / 8
            "KB": result = cantidad / 1024 / 8
            "MB": result = cantidad / 1024 / 1024 / 8
            "GB": result = cantidad / 1024 / 1024 / 1024 / 8
            "TB": result = cantidad / 1024 / 1024 / 1024 / 1024 / 8
        fin
    devolver (result)
fin

```

Ejercicio 65 – Conversor de unidades de medida de información – Solución 2

```

algoritmo convertir_unidades
variables
    cantidad es real
    unidad_inicio es entero
    unidad_destino es entero
inicio
    escribir("Introduzca la cantidad de información:");
    leer(cantidad)
    escribir("¿En qué unidad está expresada (1 = bit, 2 = byte, 3 = KB, 4 = MB, 5 = GB)?")
    leer(unidad_inicio)
    escribir("¿A qué unidad desea convertirla (1 = bit, 2 = byte, 3 = KB, 4 = MB, 5 = GB)?")
    leer(unidad_destino)

    /* Comprobaremos si hay que multiplicar o dividir */
    si (unidad_inicio < unidad_destino)      /* Hay que dividir */
        inicio
            para i desde unidad_inicio hasta (unidad_destino-1) hacer
                cantidad = convertir_a_unidad_mayor(cantidad, i)
            fin

    si (unidad_inicio > unidad_destino)      /* Hay que multiplicar */
        inicio
            para i desde unidad_inicio hasta (unidad_destino+1) incr -1 hacer
                cantidad = convertir_a_unidad_menor(cantidad, i)
            fin

    escribir("El resultado es: ", cantidad)
fin

/* Convierte la cantidad (expresada en la unidad de medida que se pasa como parámetro) a la
unidad de medida inmediatamente superior. Por lo tanto, hay que dividir */
real función convertir_a_unidad_mayor(cantidad es real, unidad_origen es entero)
variables
    result es real
inicio
    si (unidad_origen == 1)    // Si la cantidad está expresada en bits, hay que dividir entre 8

```



```

    result = cantidad / 8
si_no // En cualquier otro caso, hay que dividir entre 1024
    result = cantidad / 1024

    devolver (result)
fin

/* Convierte la cantidad (expresada en la unidad de medida que se pasa como parámetro) a la
unidad de medida inmediatamente inferior. Por lo tanto, hay que multiplicar */
real función convertir_a_unidad_mayor(cantidad es real, unidad_origen es entero)
variables
    result es real
inicio
    si (unidad_origen == 2) // Si la cantidad está expresada en bytes, hay que multiplicar por 8
        result = cantidad * 8
    si_no // En cualquier otro caso, hay que multiplicar por 1024
        result = cantidad * 1024

    devolver (result)
fin

```

Ejercicio 66 – Calcular edad

```

algoritmo calcular_edad
variables
    dia_nac, dia_hoy, mes_nac, mes_hoy, ano_nac, ano_hoy, edad son enteros
inicio
    escribir("Introduzca su fecha de nacimiento (día, mes y año)")
    leer(dia_nac, mes_nac, ano_nac)
    escribir("Introduzca la fecha del día de hoy (día, mes y año)")
    leer(dia_hoy, mes_hoy, ano_hoy)

    edad = calcular_edad(dia_nac, dia_hoy, mes_nac, mes_hoy, ano_nac, ano_hoy)
    escribir("Su edad es de ", edad, " años")
fin

entero función calcular_edad(dia_nac, dia_hoy, mes_nac, mes_hoy, ano_nac, ano_hoy son enteros)
variables
    edad es entero
inicio
    edad = ano_hoy - ano_nac // Caso general */
    si (mes_nac > mes_hoy) entonces // Este año, aún no ha sido el cumpleaños */
        edad = edad - 1
    si (mes_nac == mes_hoy) y (dia_nac > dia_hoy) // Tampoco ha sido aún el
    cumpleaños*/
        edad = edad - 1
    devolver (edad)
fin

```

Ejercicio 67 – Calcular descuentos

```

algoritmo calcular_descuentos
variables
    dia_nac, dia_hoy, mes_nac, mes_hoy, ano_nac, ano_hoy son enteros
    precio, descuento son reales
inicio
    escribir("Introduzca la fecha del día de hoy (día, mes y año)")
    leer(dia_hoy, mes_hoy, ano_hoy)

    repetir
    inicio
        escribir("Introduzca el precio de un artículo (negativo para terminar):")
        leer(precio)
        si (precio >= 0) entonces // El precio negativo es para terminar */
            inicio
                escribir("Introduzca la fecha de nacimiento del cliente (día, mes y año)")
                leer(dia_nac, mes_nac, ano_nac)
                edad = calcular_edad(dia_nac, dia_hoy, mes_nac, mes_hoy, ano_nac, ano_hoy)
                descuento = calcular_descuento(edad, precio)

```

```

        escribir("El descuento que se debe aplicar es de: ", descuento)
    fin
fin
mientras que (precio >= 0)
fin

entero función calcular_edad(dia_nac, dia_hoy, mes_nac, mes_hoy, ano_nac, ano_hoy son enteros)
variables
    edad es entero
inicio
    edad = ano_hoy - ano_nac                                /* Caso general */
    si (mes_nac > mes_hoy) entonces                            /* Este año, aún no ha sido el cumpleaños */
        edad = edad - 1
    si (mes_nac == mes_hoy) y (dia_nac > dia_hoy)                /* Tampoco ha sido aún el cumpleaños */
        edad = edad - 1
    devolver (edad)
fin

real función calcular_descuento(edad es entero, precio es real)
variables
    descuento es real
inicio
    descuento = 0                                            /* Caso general */
    si (edad > 65) entonces                                    /* Casos particulares */
        descuento = precio * 0.15
    si (edad < 25) entonces
        descuento = precio * 0.10

    devolver (descuento)
fin

```

Ejercicio 68 – Ordenar tres números

```

algoritmo ordenar_números
variables
    a, b, c son enteros
inicio
    escribir("Introduce tres números enteros: ")
    leer(a, b, c)
    ordenar(&a, &b, &c)
    escribir("Los números ordenados son: ", a, b, c)
fin

procedimiento ordenar(*a es entero, *b es entero, *c es entero)
variables
    mayor, mediano, menor son enteros
inicio
    si (*a > *b) && (*b > *c)
        inicio
            mayor = *a
            mediano = *b
            menor = *c
        fin
    si (*a > *c) && (*c > *b)
        inicio
            mayor = *a
            mediano = *c
            menor = *b
        fin
    si (*b > *a) && (*a > *c)
        inicio
            mayor = *b
            mediano = *a
            menor = *c
        fin
    si (*b > *c) && (*c > *a)
        inicio
            mayor = *b
            mediano = *c

```

```

        menor = *a
    fin
    si (*c > *a) && (*a > *b)
    inicio
        mayor = *c
        mediano = *a
        menor = *b
    fin
    si (*c > *b) && (*b > *a)
    inicio
        mayor = *c
        mediano = *b
        menor = *a
    fin

    // Asignamos los resultados a los parámetros, para devolverlos al programa principal
    // en los parámetros a, b y c
    *a = menor
    *b = mediano
    *c = mayor
fin

```

Ejercicio 69 – Escribir números en forma de letras

algoritmo escribir_números

variables

```

    numero es entero
    unid, dec, cent son enteros    /* Unidades, decenas, centenas */
    umill, dmill son enteros      /* Unidades de millar y decenas de millar */

```

inicio

```

    escribir("Introduzca un número entero (máximo 5 cifras)")
    leer(numero)
    extraer_digitos(numero, &unid, &dec, &cent, &umill, &dmill) /* Separar el n° en dígitos */

    escribir_dec_millar(umill, dmill)
    escribir_uni_millar(umill, dmill)
    escribir_centenas(unid, dec, cent)
    escribir_decenas(unid, dec)
    escribir_unidades(unid, dec)

```

fin

/* Extraer los dígitos del número mediante divisiones sucesivas */

procedimiento extraer_digitos(numero es entero, *U es entero, *D es entero,
*C es entero, *UM es entero, *DM es entero)

inicio

```

    DM = numero div 10000          /* Extraer decenas de millar */
    numero = numero - (DM * 10000) /* Extraer unidades de millar */
    UM = numero div 1000
    numero = numero - (UM * 1000)  /* Extraer centenas */
    C = numero div 100
    numero = numero - (C * 100)    /* Extraer decenas */
    D = numero div 10
    numero = numero - (D * 10)     /* Extraer unidades */
    UM = numero

```

fin

/* Escribir las decenas de millar. Se escriben igual que las decenas convencionales */

procedimiento escribir_dec_millar(umill es entero, dmill es entero)

inicio

```

    escribir_decenas(umill, dmill) /* Reutilizamos la función escribir_decenas */

```

fin

/* Escribir unid. de millar. Se escriben como las unidades convencionales, seguidas de la palabra "mil". Hay varias excepciones. Cuando la unidad de millar es 1 se escribe diferente si hay decenas de millar ("un mil" 31050) que si no las hay ("mil" 1050). En otro caso (30050, 32050), se escribe la unidad seguida de la palabra "mil" */

procedimiento escribir_uni_millar(umill es entero, dmill es entero)

inicio

```

    si (umill == 1) entonces          /* Casos especiales */
    inicio

```

```

        si (dmill > 1) entonces          /* 21xxx, 31xxx, 41xxx, etc... */
            escribir(" un mil ")
        si_no
            escribir(" mil ")           /* 11xxx, 1xxx */
    fin
    si_no                                /* Casos generales */
    inicio
        escribir_unidades(umill, dmill)
        si (umill !=0) o (dmill !=0) entonces
            escribir(" mil ")
    fin
fin

/* Escribe las centenas. La única excepción es 100 */
procedimiento escribir_centenas(unid es entero, dec es entero, cent es entero)
inicio
    si (unid == 0) y (dec == 0) y (cent == 1) entonces /* Caso excepcional */
        escribir ("cien")
    si_no                                           /* Caso general */
        según (cent) hacer
        inicio
            1: escribir("ciento")
            2: escribir("doscientos")
            3: escribir("trescientos")
            ...
            9: escribir("novecientos")
        fin
fin

/* Escribir las decenas. Los números del 10 al 19 se tratan como excepciones. También el 20
frente al caso general (2x = "veinti..."). En el resto de decenas, se escribe una " y " si la
unidad no es cero (ej: 3x = "treinta y ...") */
procedimiento escribir_decenas(unid es entero, dec es entero)
inicio
    si (dec == 1) entonces                        /* Casos especiales (de 10 a 19) */
        según (unid) hacer
        inicio
            0: escribir("diez")
            1: escribir(" escribir ("Tu puntuación final: ", puntos_humano)
escribir ("Mi puntuación funal: ", puntos_ordenad6or)
si (puntos_humano > puntos_ordenador) entonces
    escribir(";Has ganado, enhorabuena!")
si_no
    si (puntos_ordenador < puntos_humano) entonces
        escribir("Te he ganado. Inténtalo en otra ocasión.")
    si_no
        escribir("Hemos empatado. Intenta ganarme otro día.")once")
            2: escribir("doce")
            3: escribir("trece")
            ...
            9: escribir("diecinueve")
        fin
    si (dec == 2) entonces                        /* De 20 a 29 */
    inicio
        si (unid == 0) entonces
            escribir ("veinte")
        si_no
            escribir ("veinti")
    fin
    si (dec >= 3) entonces                        /* Resto de casos (de 30 a 99) */
    inicio
        según (dec) hacer
        inicio
            3: escribir("treinta")
            4: escribir("cuarenta")
            5: escribir("cincuenta")
            ...
            9: escribir("noventa")
        fin
        si (unid != 0) entonces
            escribir (" y ")

```

```

    fin
fin /* procedimiento escribir_decenas */

/* Escribe las unidades. Excepción: cuando la decena es 1 (10, 11, 12, etc). Estos casos
se escriben en el procedimiento escribir_decena */
procedimiento escribir_unidades(unid es entero, dec es entero)
inicio
    si (dec != 1) entonces                                /* Si el n° está entre 10 y 19, no escribir nada */
        según (unid) hacer                                /* En cualquier otro caso, escribir la unidad */
            inicio
                1: escribir("uno")
                2: escribir("dos")
                3: escribir("tres")
                ...
                9: escribir("nueve")
            fin
fin
fin

```

Ejercicio 70 – Convertir números binarios a decimales

```

algoritmo convertir_binario_a_decimal
variables
    bin, dec son enteros
inicio
    escribir("Introduzca un número binario (máximo 8 cifras)")
    leer(bin)

    dec = convertir(bin)
    escribir("El número decimal es:", dec)
fin

/* Esta función recibe como parámetro un número binario en una variable de tipo entero
y lo convierte a decimal, devolviéndolo también como un entero */

entero función convertir(bin es entero)
variables
    r, posicion, digito son enteros
inicio
    r = 0                                /* Resultado (acumulador) */
    posicion = 0                         /* "Peso" del primer dígito binario */

    mientras (bin != 0) hacer
        inicio
            digito = bin % 10            /* Extraemos un dígito */
            r = r + (digito * 2 ^ posicion) /* Acumulamos el resultado */
            bin = bin div 10              /* Eliminamos el dígito recién procesado */
            posicion = posicion + 1       /* Actualizamos la posición */
        fin

    devolver (r)
fin

```

Ejercicio 71 – Convertir a modular el juego de los dados

```

algoritmo dados
variables
    puntos_maquina, puntos_humano son enteros
    turno es cadena
    tirada es carácter
    dado1, dado2, dado3 son enteros
    premio es entero
inicio
    puntos_ordenador = 0
    puntos_humano = 0
    turno = "HUMANO"
repetir
    inicio
        tirada = manejar_turno(turno)

```

```

    si (tirada != 'Q') entonces
    inicio
        tirar_dados(&dado1, &dado2, &dado3)
        premio = comprobar_dados(dado1, dado2, dado3)
        sumar_premio(&turno, premio, &puntos_humano, &puntos_maquina)
    fin
fin
mientras que (tirada != 'Q')
    escribir_resultado(puntos_humano, puntos_maquina)
fin

/* Esta función muestra el turno y lee la tirada del jugador humano (si es su turno).
   Devuelve la tirada del humano */
carácter función manejar_turno(turno es cadena)
variables
    tirada es carácter
inicio
    si (turno == "HUMANO") entonces
    inicio
        escribir ("Es tu turno. Pulsa Enter para tirar los dados o Q para terminar")
        leer (tirada)
    fin
    si_no
    inicio
        escribir ("Es mi turno.")
        tirada = ' '
    fin
    devolver(tirada)
fin

/* Esta función sirve para tirar los tres dados al azar. Devuelve el valor de los
   dados dentro de las variables pasadas por referencia */
procedimiento tirar_dados(*dado1 es entero, *dado2 es entero, *dado3 es entero)
inicio
    *dado1 = aleatorio(6) + 1
    *dado2 = aleatorio(6) + 1
    *dado3 = aleatorio(6) + 1
    escribir("La tirada ha sido: ", *dado1, *dado2, *dado3)
fin

entero función comprobar_premio(dado1 es entero, dado2 es entero, dado3 es entero)
variables
    premio es entero
inicio
    premio = 0
    si (dado1 == dado2) o (dado1 == dado3) entonces
        premio = 10
    si (dado1 == dado2) y (dado1 == dado3) entonces
    inicio
        si (dado1 == 6) entonces
            premio = 50
        si_no
            premio = 20
    fin
    devolver(premio)
fin

procedimiento sumar_premio(*turno es cadena, premio es entero,
                           *puntos_humano es entero, *puntos_maquina es entero)
inicio
    si (turno == "HUMANO") entonces
    inicio
        *puntos_humano = *puntos_humano + premio
    fin
fin

```

```

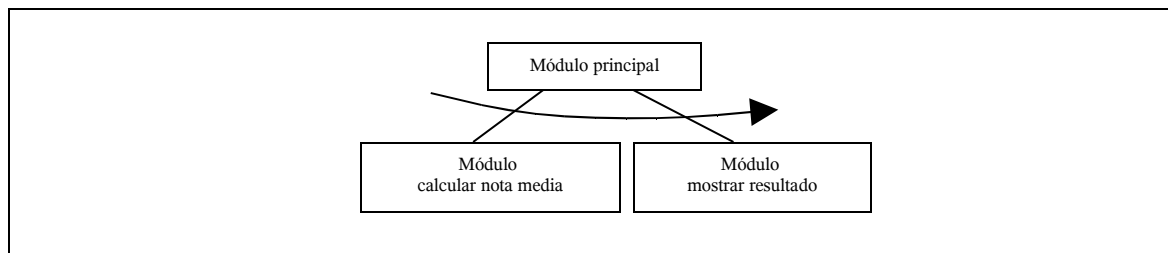
        *turno = "ORDENADOR"
    fin
    si_no
    inicio
        *puntos_ordenador = *puntos_ordenador + premio
        *turno = "HUMANO"
    fin
fin

procedimiento escribir_resultado(puntos_humano es entero, puntos_maquina es entero)
inicio
    escribir ("Tu puntuación final: ", puntos_humano)
    escribir ("Mi puntuación final: ", puntos_maquina)
    si (puntos_humano > puntos_maquina) entonces
        escribir("¡Has ganado, enhorabuena!")
    si_no
        si (puntos_maquina < puntos_humano) entonces
            escribir("Te he ganado. Inténtalo en otra ocasión.")
        si_no
            escribir("Hemos empatado. ¡Intenta ganarme otro día!")
fin

```

Ejercicio 72 – Convertir a modular el programa de las notas de los alumnos

Se han hecho dos subalgoritmos: uno para calcular la nota media y otro para mostrar el resultado. La estructura modular sería ésta:



```

algoritmo notas
variables
    contraseña es cadena
    n_alumnos, alum son enteros
    nota_media es real
inicio
    escribir("Bienvenido al sistema. Por favor, escriba la contraseña.")
    repetir
        leer(contraseña)
    mientras que (contraseña == "12345")

    escribir("Introduzca el número de alumnos")
    leer (n_alumnos)
    para alum desde 1 hasta n_alumnos hacer
        inicio
            nota_media = calcular_nota_media()
            mostrar_resultado_alum(nota_media)
        fin
    fin
fin

real función calcular_nota_media()
variable
    asig es entero
    nota es real
    nota_media es real
inicio
    nota_media = 0.0
    escribir ("Introduciendo notas del alumno/a nº ", alum)

```

```

para asig desde 1 hasta 5 hacer
inicio
    según (asig) hacer
        inicio
            1: escribir("Escriba la nota de los ejercicios")
            2: escribir("Escriba la nota del examen")
            3: escribir("Escriba la nota de clase")
            4: escribir("Escriba la nota del trabajo obligatorio")
            5: escribir("Escriba la nota de actitud")
        fin
    leer (nota)
    según (asig) hacer
        inicio
            1: nota_media = nota_media + (nota * 0.10)
            2: nota_media = nota_media + (nota * 0.30)
            3: nota_media = nota_media + (nota * 0.15)
            4: nota_media = nota_media + (nota * 0.25)
            5: nota_media = nota_media + (nota * 0.20)
        fin
    fin

    devolver(nota_media)
fin

procedimiento mostrar_resultado_alum(nota_media es real)
inicio
    escribir(nota_media)
    si (nota_media >= 5) entonces
        escribir ("El alumno/a está APROBADO/A")
    si_no
        escribir ("El alumno/a está SUSPENSO/A")
fin

```