

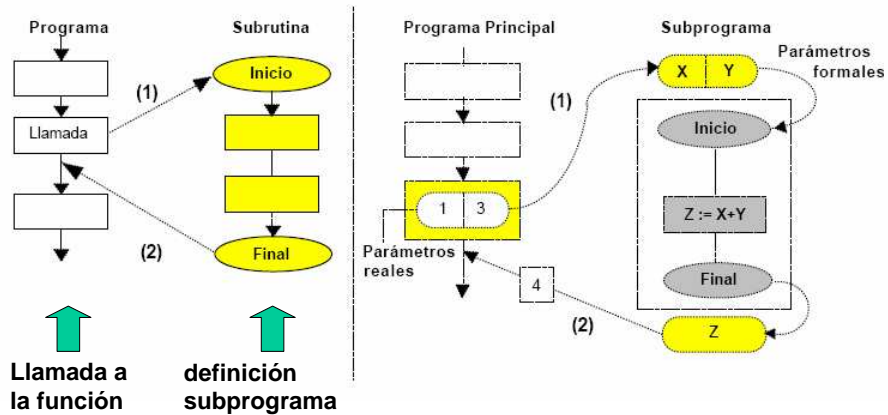
Funciones I

Parámetros por valor

Fundamentos de Programación

Fundamentos de Programación I

Un subprograma hace el papel de un programa. Puede tener una sección de declaraciones (variables, constantes, etc...) y posee también unos datos de entrada y de salida. Esto permite, como ya veremos, que el subprograma sea totalmente independiente del programa principal.



Una función es un subprograma que siempre tiene un parámetro de salida (cos(x), pow(2,3), ...).

```
Tipo nom_func (parámetros)    // → Cabecera de la función
{
  Declaracions                  // → Variables, ...
  ...
  Instruccions                  // → Cuerpo de la función
  ...
  return valor;                 // → Valor de devuelto
}
```

Tipo es el tipo de la variable de salida,
nom_func es un identificador que representa el nombre de la función,
parámetros es un conjunto de parámetros separados por comas, nombre de las variables con su tipo.
Mediante la instrucción **return** se envía el valor que devuelve la función al acabar.

- Cuando se hace una llamada a una función, lo primero que se hace es una asignación de los **parámetros reales** (los del programa princ.) a los **formales** (los de la función) → **operación de Paso de Parámetros (valor | referencia)**, y después comienzan a ejecutarse las instrucciones de la función ...
- Si la función no devuelve ningún valor, se declara de tipo **void**.

- **Ejercicio:** Este ejercicio consiste en la realización de un programa completo, correctamente modulado, utilizando funciones, que simule el funcionamiento de una calculadora electrónica. Así, implementaremos algunas de las operaciones básicas que se puede encontrar en una calculadora común. El programa cuando se ejecute mostrará un menú con todas las operaciones posibles. Cada una de estas funciones se implementará como una función independiente.

Operaciones a implementar como funciones:

- Suma, resta, multiplicación y división.
- Factorial de un número.
- Cálculo de un número aleatorio.
- Función que calcule la raíz cúbica de un número.

Función que suma dos números

```
float sumar ( float a, float b)
{
    int res; //Declaración de variables locales

    res = a + b;

    return res; //Valor que devuelve la función
}
```

Función que resta dos números

```
float restar ( float a, float b)
{
    int res; //Declaración de variables locales

    res = a - b;

    return res; //Valor que devuelve la función
}
```

Función que multiplica dos números

```
float multiplicar ( float a, float b)
{
    int res; //Declaración de variables locales

    res = a * b;

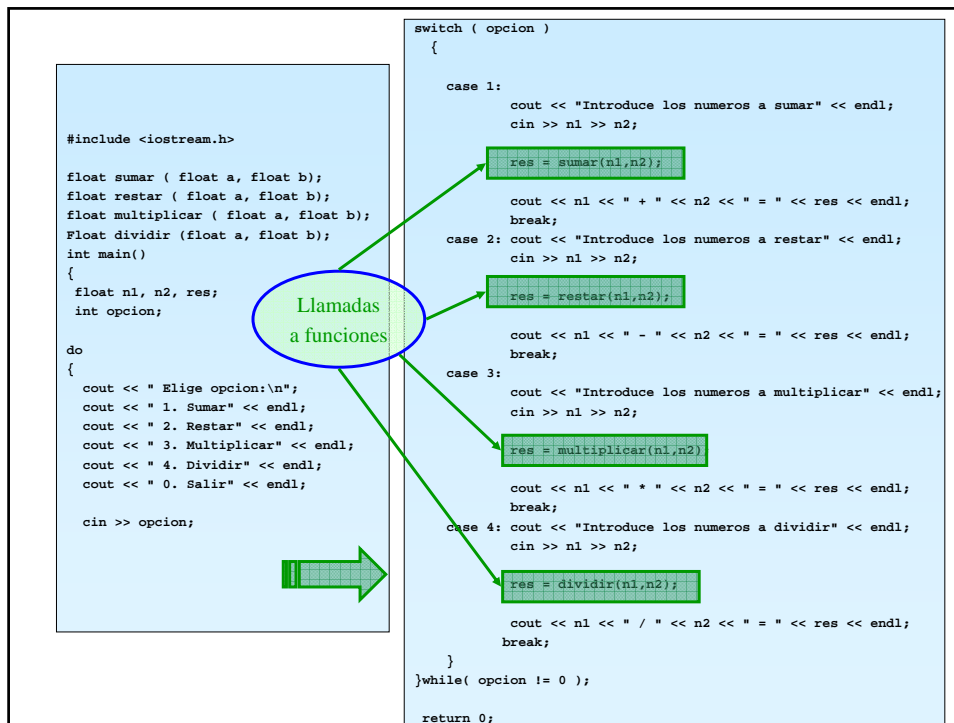
    return res; //Valor que devuelve la función
}
```

Función que divide dos números

```
float dividir ( float a, float b)
{
    int res; //Declaración de variables locales

    res = a / b;

    return res; //Valor que devuelve la función
}
```



Función que calcula el factorial

Entradas: número entero del que calcular el factorial (**n**)

Salidas: factorial del número (**n**)

Análisis: ¿Como se calcula el factorial de un número?

```

int factorial (int n)
{
    int res; // Variables locales
    int i;
    res = 1;
    for (i = 1 ; i <= n ; i++)
    {
        res = res * i;
    }
    return res; //Valor devuelto por la función
}

```

- Algo muy útil es la generación de números aleatorios en un intervalo determinado. Añadir a vuestra calculadora esta opción , indicando el intervalo en el cual debe estar el número generado [vmin,vmax], ambos incluidos. Ayuda: existe una función `int rand()` que devuelve un número entero en el intervalo `[0, RAND_MAX]`, dónde `RAND_MAX` es una constante.

Entradas: intervalo para la generación de número aleatorio [vmin, vmax]

Salidas: número aleatorio

Análisis: ¿Cómo calculo un número aleatorio utilizando la función `int rand()`

```
int NumeroAleatorio(int vmin, int vmax)
{
    int res;
    int n_alea;
    int tam;

    n_alea = rand();
    tam = vmax - vmin + 1;
    res = n_alea % tam + vmin;

    return res;
}
```

- Incorporar al programa anterior una función que calcule la raíz cúbica de un número x. Para ello se puede utilizar el siguiente método basado en aproximaciones sucesivas: sabiendo que α es una aproximación del resultado, β es una aproximación mejor dada por la siguiente expresión:

$$\beta = (2 * \alpha + x / (\alpha * \alpha)) / 3$$

con un error inferior a un valor introducido por teclado (error)

Entradas: número x del que se quiere calcular y el error

Salidas: raíz cúbica del número

Análisis: ???

```
int RaizCubica ( float x, float error)
{
    float alfa, beta;

    alfa = x / 3; // inicializo para la repetición, primera aproximación a la raíz

    do{
        beta = (2 * alfa + x / (alfa*alfa)) / 3 ;
        dif = alfa - beta;

        if (dif < 0)
            dif = - dif; //Valor absoluto

        alfa = beta;
    } while (dif > error); //Condición de final

    return beta;
}
```

- Ejercicio 1: Escribir un programa que muestre los valores de la recta $y = mx + b$ en el rango $[r0..r1]$. Los valores de los coeficientes m y b , así como el rango deben ser introducidos por teclado. Por pantalla deben aparecer una secuencia de líneas según el siguiente formato:

```
Recta: y= 2x+1 Rango:[1..3]
Valor de x = 1, Valor de y = 3
Valor de x = 2, Valor de y = 5
Valor de x = 3, Valor de y = 7
```

Entradas: Parámetro m y b , rango $[r0, r1]$
Salidas: Puntos que pertenecen a la recta
Análisis :

```
#include <iostream>

int main()
{
    int m, b; //coeficientes de la ecuacion de la recta
    int r0,r1; //puntos extremos del intervalo
    int x, y; //para el bucle for

    cout << "Introducir coeficientes de la recta: y=mx+b\n";
    cin >> m >> b;

    cout << "Recta: "<< "y=" << m << "x + " << b ;
    cout << "\tRango: [" << r0 << " " << r1 << "]\n";
    //Calculo los puntos de la recta
    for (x = r0 ; x <= r1; x++)
    {
        //Calculo valores de y para cada valor de x
        y = m * x + b;

        //Muestro resultado
        cout << " Valor de x: " << x << ", Valor de y: " << y << endl;
    }

    return 0;
}
```

```

#include <iostream.h>

//Prototipos de funciones
int recta(int x, int m, int b);

int main()
{
    int m, b; //coeficientes de la ecuacion de la recta
    int r0,r1; //puntos extremos del intervalo
    int x, y; //para el bucle for

    cout << "Introducir coeficientes de la recta: y=mx+b\n";
    cin >> m >> b;

    cout << "Recta: "<< "y=" << m << "x + " << b ;
    cout << "\tRango: [" << r0 << " " << r1 << "]\n";
    //Calculo los puntos de la recta
    for (x = r0 ; x <= r1; x++)
    {
        //Calculo valores de y para cada valor de x
        y =recta ( x, m, b);
        //Muestro resultado
        cout << " Valor de x: " << x << " , Valor de y: " << y <<endl;
    }

    return 0;
}

```

Llamada a la función

```

//Función que calcula la coordena y que pasa por una recta
int recta(int x, int m, int b)
{
    int y;

    y = m*x + b;

    return y;
}

```

- Ejercicio 2: Dado el día, el mes y el año, hacer una función que calcule el día de la semana (0 = domingo) correspondiente. La fórmula que permite conocer esto es:

```
x = 14 - mes / 12;
y = anyo - x;
z = mes + (12 * x) - 2 ;
n = dia + y + (y / 4) - (y / 100) + (y / 400) + (31 * z / 12)
n % 7 indica el día de la semana ( 0 = domingo)
```

Entradas: día (d), mes (m) y año (a)

Salidas: día de la semana (n) (1..7)

Análisis: Qué parámetros se deben pasar a la función y que devuelve???

```
#include <iostream.h>
//prototipo de funciones
int CalcularDiaSemana(int d, int m, int a);

int main()
{
    int d, m a;
    int n;

    cout << "Introduce el día mes año\n (d m a):\n";
    cin >> d >> m >> a;

    n = CalcularDiaSemana(d,m,a);

    cout << "El día de la semana es: " << n << endl;

    return 0;
}

//Calcula el día de la semana
int CalcularDiaSemana(int dia, int mes, int anyo)
{
    int x, y, z, n;

    x = 14 - mes / 12;
    y = anyo - x;
    z = mes + (12 * x) - 2 ;
    n = dia + y + (y / 4) - (y / 100) + (y / 400) + (31 * z / 12)

    return n % 7;
}
```

Llamada a la función

- Ejercicio 3: Invertir las cifras de un número entero introducido por teclado. La operación invertir debe aparecer como una función

Entradas: Número a invertir (n)

Salidas: Número invertido (ninv)

Análisis: ¿Cómo se puede invertir un número entero?

```
#include <iostream.h>
//prototipo de funciones
int invertir (int dato);

int main()
{
    int x; //Dato de entrada
    int r; //resultado

    cout << "Introduce el numero entero a invertir\n";
    cin >> x;

    r = invertir(x);

    cout << "El numero invertido es: " << r << endl;

    return 0;
}

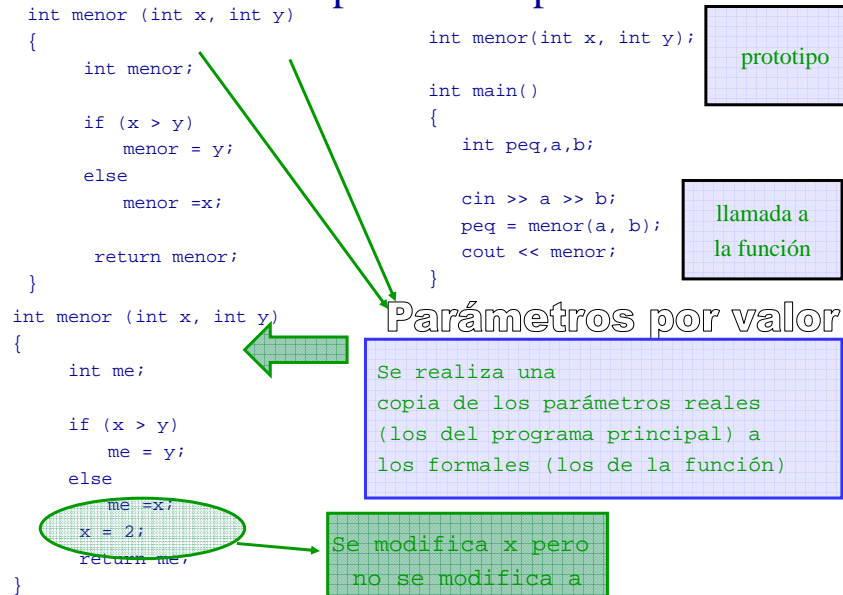
//Invierto las cifras del 'dato'
int invertir(int dato)
{
    int res; //para invertir dato
    int coc, resto; //divisiones por 10
    res = 0;

    do
    {
        cociente = dato / 10;
        resto = dato % 10; //Cifra eliminada al dividir por 10
        //Construyo el nuevo numero
        res = res * 10 + resto;
        dato = cociente
        //Repito el proceso con una cifra menos
    }while (dato != 0);
    return res;
}
```

Llamada a la función

Código de la función

Resumen parámetros por valor



- **Ejercicio 2:** Escribir un programa que permita **codificar** y **decodificar** una cadena de caracteres acabada en '\n'. La codificación se realizará mediante la suma de un carácter (que llamaremos clave de codificación) a cada uno de los caracteres de la cadena inicial. El programa mostrará un menú con las siguientes opciones:

Selecciona la tarea a realizar:

- Cambiar clave
- Codificar una cadena
- Decodificar una cadena

Cualquier otro carácter para salir

Entradas: cadena de 4 caracteres y un carácter clave

Salidas: cadena de 4 caracteres codificada o decodificada

Análisis: Este programa hará uso de 2 funciones ?? ¿Cómo estructuramos el programa? ¿Qué parámetros necesita cada una de las funciones??

```

#include <iostream.h>

//Prototipos de funciones
char Codificar(char ch, char clave);
char Decodificar(char ch, char clave);

int main()
{
    char clave, ch, ch_codif;
    bool salir = false;
    char opcion;
    bool fin;

    cout << "Introducir clave de codificacion\n";
    cin >> clave;

    do
    {
        cout << "Selecciona la tarea a realizar:" << endl;
        cout << " a. Cambiar clave" << endl;
        cout << " b. Codificar una cadena " << endl;
        cout << " c. Decodificar una cadena" << endl;
        cout << "Pulsa cualquier otro caracter para salir " << endl;
        cin >> opcion;
        cin.ignore();

        switch (opcion)
        {
            case 'a': cout << "Introducir una nueva clave de codificacion" << endl;
                      cin.get(clave);
                      cout << "Clave cambiada" << endl;
                      break;
        }
    }
}

```

```

case 'b':
    fin = false;
    cout << "Introduce una cadena finalizada con el caracter fin de linea " << endl;
    do
    {
        cin.get(ch);
        if (ch != '\n')
        {
            ch_codif = Codificar(ch,clave);
            cout << ch_codif;
        }
        else
            fin = true;
    } while(fin != true);
    cout << endl;
    break;

case 'c':
    fin = false;
    cout << "Introduce una cadena finalizada con el caracter fin de linea para decodificar " << endl;
    do
    {
        cin.get(ch);
        if (ch != '\n')
        {
            ch_codif = Decodificar(ch,clave);
            cout << ch_codif;
        }
        else
            fin = true;
    } while(fin != true);

    cout << endl;
    break;
    default : salir = true;
}

} while (salir != true);

return 0;
}

```

Llamada a la función

```

/*****
* Funcion codificar carácter
* Descripcion: Transforma un carácter en otro sumandole una clave
* Parámetros:
* Nombre: E/S
* ch      E
* clave   E
*
* Valor devuelto:
* char Es de tipo carácter (valor devuelto)
*****/

char Codificar( char ch, char clave)
{
    char mich;

    mich = char( (int(ch)+ int(clave) - int(' ')) % (int('-')+ 1 - int(' ')) + int(' '));

    return mich;
}

/*****
* Funcion codificar carácter
* Descripcion: Transforma un carácter en otro restandole una clave
* Parámetros:
* Nombre: E/S
* ch      E
* clave   E
*
* Valor devuelto:
* char Es de tipo carácter (valor devuelto)
*****/

char Decodificar( char ch, char clave)
{
    char mich ;

    mich = char( (int(ch) -int(clave) - int(' ')) % (int('-')+ 1 - int(' ')) + int(' '));

    return mich;
}

```