

Funciones II

Fundamentos de Programación
Fundamentos de Programación I

- Ejercicio 1: Escribe una función que transforme un punto en coordenadas polares a cartesianas

Entradas: Un punto como coordenadas polares (**modulo y ángulo**)

Salidas: Coordenadas x e y (x,y)

Análisis: ¿cuáles son las expresiones que te permiten pasar de coordenadas polares a cartesianas??

```

#include <iostream.h>
#include <math.h>

//Prototipo de la función
void PolaresACartesianas(float modulo, float angulo, float &x, float &y);

int main()
{
    float m,ang;
    float posx, posy;

    cout << "Introducir el modulo y el angulo del punto en coordenadas polares:\n";

    cin >> m >> ang;

    //Calculo la posicion x, y

    PolaresACartesianas(m, ang, posx, posy);

    cout << "Las coordenadas cartesianas correspondientes a: " <<m << ', ' << ang << endl;
    cout << "("<< posx << ", " << posy<<")"<< endl;

    return;
}

void PolaresACartesianas(float modulo, float angulo, float &x, float &y)
{
    x = modulo*cos(angulo);
    y = modulo*sin(angulo);

    return;
}

```

- **Ejercicio 2:** Escribe una función que, dados dos enteros positivos x e y , calcule el cociente de la división entera y el resto utilizando únicamente restas.

Entradas: Dos números enteros (a y b)

Salidas: El cociente y el resto que han sido calculados

Análisis: ¿Cuál es el algoritmo que permite calcular la división entera mediante restas??

```

/*****
* Funcion division entera
* Descripcion: Calcula el cociente y el resto de la division entera
* Parámetros:
* Nombre: E/S
* a      E
* b      E
* resto  S
*
* Valor devuelto:
* char Es de tipo carácter (valor devuelto)
*****/
int DivisionEntera(int a, int b, int &resto)
{
    int cociente;

    cociente=0;

    while( a > b)
    {
        a = a -b;
        cociente ++;
    }
    resto = a;

    return cociente;
}

```

```

#include <iostream.h>
//Prototipo de la funcion

int DivisionEntera(int a, int b, int &resto);

int main()
{
    int dividendo, divisor;
    int cociente, resto;
    //Leo datos
    cout << "Introducir dividendo y divisor:" << endl;
    cin >> a;
    cin >> b;

    cociente = DivisionEntera(dividendo, divisor, resto);
    //Escribo datos
    cout << "El cociente es:" << cociente << y el resto:" << resto << endl;

    return 0;
}

```

Llamada a la función

- Ejercicio 3: Realizar una función recursiva que calcule el sumatorio y el productorio de los n primeros números naturales.

```
/******  
Funcion que calcula el sumatorio de forma recursiva  
******/  
int Sumatorio(int n)  
{  
    int res;  
  
    if (n == 1)  
        res = 1;  
    else  
        res = n + sumatorio( n -1);  
  
    return res;  
}  
/******  
Funcion que calcula el productorio de forma recursiva  
******/  
int Productorio(int n)  
{  
    int res;  
  
    if ( n == 1)  
        res = 1;  
    else  
        res = n * Productorio ( n-1);  
  
    return res;  
}
```

```
#include <iostream.h>

//Prototipo de la funcion
int Sumatorio(int n);
int Productorio(int n);
int main()
{
    int num;
    int suma, producto;

    cout << "Este programa calcula el sumatorio y productorio de n numeros naturales\n";
    cout << "Utilizando una función recursiva. Introduce n:\n";

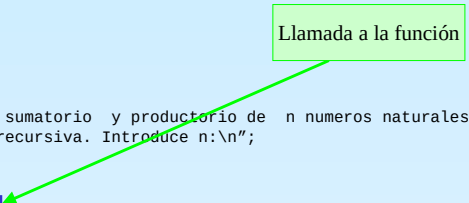
    cin >> n;

    suma = Sumatorio(num);
    producto = Productorio(num);

    cout << "El productorio es:" << producto;
    cout << "El sumatorio es:" << suma;

    return 0;
}
```

Llamada a la función



- Ejercicio 4: Realizar una función recursiva que calcule recursivamente la suma de dos números enteros utilizando solamente operaciones de incremento y decremento.

```

/*****
* Funcion que suma dos numero enteros
* Descripcion: Calcula el cociente y el resto de la division entera
* Parámetros:
* Nombre: E/S
* a      E
* b      E
*
* Valor devuelto:
* int (valor devuelto)
*****/
int Suma(int a, int b)
{
    if ( a == 0)
        res = b;
    else
        res = suma(a-1,b+1)

    return res;
}

```

- **Ejercicio 5: Recursividad indirecta.** Escribe una función que devuelva true si un número entero pasado como parámetro es par y false en caso contrario.

```

#include <iostream.h>

// Prototipos
bool par(int n);
bool impar(int n);

int main()
{
    int a;

    cout << "Introduce un número entero : ";

    cin >> a ;
    cin.ignore();

    if(par(a))
        cout << " ... es par " << endl;
    else
        cout << " ... es impar " << endl;

    system("pause");

    return 0;
}

bool par(int n)
{
    if (n == 0)
        return true;
    else
        return impar(n-1);
}

bool impar(int n)
{
    if (n == 0)
        return false;
    else
        return par(n-1);
}

```

- **Ejercicio2:** Se desea calcular la trayectoria de un determinado móvil sometido a un movimiento uniforme ($v = e/t$). El móvil estará situado en una posición inicial (x_0, y_0) y deberá recorrer 100 metros. Calcular las posiciones intermedias para los siguientes móviles:
 - Una hormiga (0.5 km/hora)
 - una persona (3.5 km/hora)
 - Un caballo (30 km/hora)

Nota: Para la resolución del ejercicio diseñar una función que dada una posición inicial, una velocidad y un intervalo de tiempo calcule la posición alcanzada por el móvil.