

PROGRAMACIÓN

(GRADOS EN INGENIERO MECÁNICO, ELÉCTRICO, ELECTRÓNICO INDUSTRIAL y QUÍMICO INDUSTRIAL)

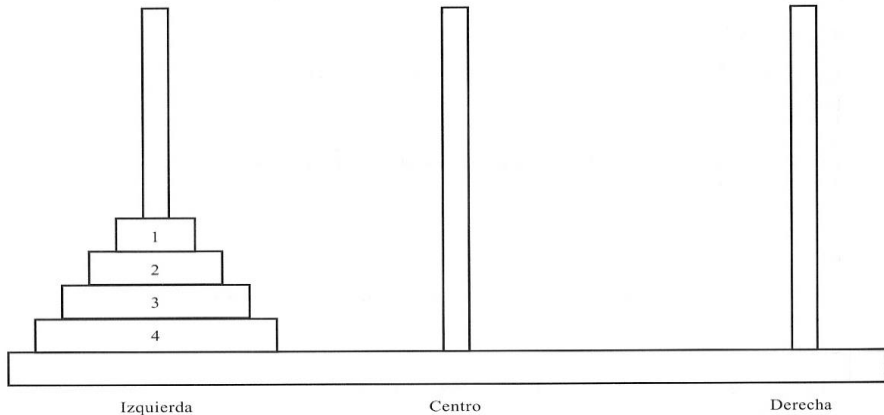
Sesión	7 (Diseño modular. Sintaxis de C: Recursividad + Módulos como Parámetros de otros módulos)	
Temporización	1 hora (no presencial)	
Objetivos formativos	<ul style="list-style-type: none"> • Conocer la estructura sintáctica de un programa modular en C: definiciones de funciones, prototipos de funciones, bibliotecas de funciones, paso de parámetros, tipo de dato puntero, funciones como parámetros, tipos de almacenamiento automático, externo, estático y registro. • Implementar programas modulares en lenguaje de programación C. Identificar y corregir errores sintácticos que surgen durante la codificación. • Construir módulos individuales (subprogramas), identificando su actividad funcional (entrada, salida, cálculo,...) y su interfaz para la comunicación de información con otros módulos (lista de parámetros formales + valor especial asociado al identificador del módulo). Distinguir entre parámetros de entrada y de salida. Documentar adecuadamente estos módulos mediante la notación sub-algorítmica. • Descomponer un problema en sub-problemas, identificando a partir del enunciado del mismo la información a procesar y la lista de tareas de manipulación de información; organizar esta lista de tareas de forma jerárquica, descomponiendo tareas complejas en tareas más simples. Representar mediante un diagrama en forma de árbol la estructura del programa. • <i>Resolver problemas sencillos mediante recursividad como alternativa a la construcción de repetición: expresar el problema de forma recursiva distinguiendo entre el caso general y el caso base.</i> • <i>Utilizar módulos como parámetros con el objetivo de separar algunos algoritmos de cálculo de las funciones sobre las que actúan.</i> • Probar con datos operacionales la correctitud de los módulos individuales (pruebas unitarias) y de los programas desarrollados e identificar y corregir los errores lógicos que surjan. 	
Competencias a desarrollar	<ul style="list-style-type: none"> • RD1: Poseer y comprender conocimientos • RD2: Aplicación de conocimientos • UAL1: Conocimientos básicos de la profesión • UAL3: Capacidad para resolver problemas • UAL6: Trabajo en equipo • FB3: Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en la ingeniería. 	X X X X X
Materiales	Sesiones de teoría 7.1 a 7.6 + bibliografía tema 3 + Internet IDEs: Dev-c++/Code::Blocks (freeware)	
Tarea	Desarrollar los programas correspondientes a los ejercicios propuestos en esta ficha de trabajo y presentar un informe según modelo que se adjunta.	
Fecha de entrega	No hay. Si tiene dudas al realizar algún ejercicio, consulte con el profesor.	

Criterios de éxito	<ul style="list-style-type: none"> • Terminar en el tiempo previsto la tarea. • Demostrar, en una prueba escrita u oral, mediante las respuestas a las preguntas del profesor que ha alcanzado los objetivos formativos. 	
Plan de trabajo	Actividad	Temporización
	Estudio de los ejemplos presentados en teoría de recursividad y de funciones como parámetros, codificación y prueba de los mismos (el código fuente en C se encuentra en esta ficha de trabajo).	10 mn
	Diseño de los algoritmos correspondientes a cada uno de los ejercicios propuestos. Nota: puede simultanear esta actividad con las dos siguientes (para cada ejercicio).	20 mn
	Implementación en lenguaje C de los programas correspondientes a los algoritmos diseñados	20 mn
	Pruebas: los programas desarrollados serán validados utilizando como mínimo los datos de prueba suministrados. Nota: en caso de detectar errores en esta fase de pruebas, estos deberán ser corregidos modificando el código fuente y/o el algoritmo correspondiente.	5 mn
	Elaboración de la documentación a presentar según modelo adjunto, así como de la respuesta a las cuestiones planteadas en el mismo.	5 mn

Ejemplos resueltos: Recursividad	
Estrategia de diseño de un módulo recursivo	<p>Leer detenidamente el enunciado del sub-problema e intentar expresarlo de forma recursiva. Una definición recursiva de un problema debe constar de al menos dos partes:</p> <ul style="list-style-type: none"> • Caso general (o recursivo). • Caso base (o condición de terminación). <p>Para obtener esta definición recursiva, puede servir de guía el plantearse las siguientes preguntas:</p> <ul style="list-style-type: none"> • ¿Cómo se puede definir el problema en términos de uno o más problemas más pequeños del mismo tipo que el original? → caso general. • ¿Qué instancias del problema harán de caso base? • Conforme el problema se reduce de tamaño, ¿se alcanzará el caso base? • ¿Cómo se usa la solución del caso base para construir una solución correcta al problema original? <p>Nótese que en todos los ejercicios de esta sección se pide la construcción de un programa completo que incluya una función recursiva.</p>

Ejercicio	Imprimir en pantalla los n primeros polinomios de Hermite para un valor de x dado por teclado. Polinomios de Hermite: $H_0(x)=1$ $H_1(x)=x$ $H_n(x)=2*x*H_{n-1}(x) - 2*(n-1)*H_{n-2}(x)$ Si (n>1)																					
Análisis (Pre-diseño)	Información: E: n (entero positivo o nulo) { nº de polinomios } x (real) S: valores de los n primeros polinomios de Hermite en x: $H_i(x)$ (i:1..n) Tareas: <ul style="list-style-type: none">Leer datos de entrada.<ul style="list-style-type: none">Leer entero positivo o nulo.Leer real.Escribir valores de polinomios.<ul style="list-style-type: none">Calcular valor del polinomio de Hermite n en x.																					
Diseño	<u>Diseño preliminar</u> <u>Diseño de datos</u> <u>Estructura del programa</u> <div><div>Módulo principal</div><div><div>Leer entero no negativo</div><div>Hermite</div></div></div> <u>Interfaces entre módulos</u> <table><tr><th>Nombre módulo</th><th>Tipo parám.</th><th>Nombre parám.</th><th>Tipo datos</th></tr><tr><td>Módulo principal</td><td></td><td></td><td></td></tr><tr><td>Leer entero no negativo</td><td>S</td><td>n</td><td>entero</td></tr><tr><td rowspan="3">Hermite</td><td rowspan="2">E</td><td>n</td><td>entero</td></tr><tr><td>x</td><td>real</td></tr><tr><td>S</td><td></td><td>real</td></tr></table> <u>Diseño detallado</u> Algoritmo Módulo_principal Var c: carácter i,n: entero x: real Inicio Repetir Escribir("POLINOMIOS DE HERMITE") Escribir("Introduzca valor de x: ") Leer(x) Escribir("Introduzca nº de polinomios: ") Llamar a leer_entero_no_negativo(n) i←0 Mientras(i≤n) Hacer Escribir("Polinomio nº ",i,"(",x,")=",hermite(i,x)) i←i+1 Fin mientras	Nombre módulo	Tipo parám.	Nombre parám.	Tipo datos	Módulo principal				Leer entero no negativo	S	n	entero	Hermite	E	n	entero	x	real	S		real
Nombre módulo	Tipo parám.	Nombre parám.	Tipo datos																			
Módulo principal																						
Leer entero no negativo	S	n	entero																			
Hermite	E	n	entero																			
		x	real																			
	S		real																			

	<p> Escribir("Ejecutar de nuevo (S/N): ") Leer(c) Hasta_que (c!='N')ó(c!='n') Fin_algoritmo_principal </p> <p> Procedimiento leer_entero_no_negativo(n: entero(S)) Inicio Repetir Escribir("Introduzca entero no negativo: ") Leer(n) Hasta_que (n≥0) Fin_procedimiento </p> <p> <u>Diseño del módulo recursivo:</u> definición recursiva del problema: $H(n,x)=\begin{cases} 1 & \text{Si } n=0 \\ x & \text{Si } n=1 \\ 2*x*hermite(n-1,x)-2*(n-1)*hermite(n-2,x) & \text{Si } n>1 \end{cases}$ } caso base } caso general </p> <p> Función hermite(n: entero(E), x: real(E)): real Inicio Si (n=0) Entonces Devolver (1) Sino Si (n=1) Entonces Devolver(x) Sino Devolver(2*x*hermite(n-1,x)-2*(n-1)*hermite(n-2,x)) Fin_si Fin_si Fin_función </p>
Codificación	<pre> #include <stdio.h> #include <stdlib.h> #include <conio.h> #include <ctype.h> #include <math.h> void leer_entero_no_negativo(int *n); double hermite(int n,double x); int main(){ char c; int i,n; double x; do{ system("cls"); printf("POLINOMIOS DE HERMITE\n"); printf("=====\n\n"); printf("Introduzca valor de x: "); scanf(" %lf", &x); printf("Introduzca num. de polinomios:\n"); leer_entero_no_negativo(&n); for(i=0;i<=n;++i){ printf("\nPolinomio num. %3d (%.21f) = %.21f", i,x,hermite(i,x)); } printf("\n\nDesea efectuar una nueva operacion (s/n)? "); c=toupper(getch()); }while (c!='N'); return 0; } void leer_entero_no_negativo(int *n){ do{ printf("\tIntroduzca entero no negativo: "); scanf(" %d",n); }while(*n<0); } double hermite(int n,double x){ if(n==0) </pre>

	<pre> return(1); else if(n==1) return(x); else return(2*x*hermite(n-1,x)-2*(n-1)*hermite(n-2,x)); } </pre>
Ejercicio	<p>Las Torres de Hanoi es un juego consistente en tres pivotes y un cierto número de discos de tamaños diferentes. Los discos tienen un agujero en el centro, con lo que se pueden apilar en cualquiera de los pivotes. Inicialmente los discos se amontonan en el pivote de la izquierda por tamaño decreciente, es decir, el más grande abajo y el más pequeño arriba:</p>  <p>El objeto del juego consiste en llevar los discos del pivote izquierdo al pivote derecho, de acuerdo con las siguientes restricciones: solo se puede mover un disco cada vez, cada disco debe encontrarse siempre en uno de los pivotes y no se puede colocar nunca un disco sobre otro más pequeño. Estrategia general: considerar uno de los pivotes como origen y el otro como destino; el tercer pivote se utilizará para almacenamiento auxiliar (temporal), con lo que podremos mover los discos sin poner ninguno sobre otro más pequeño. Supongamos que hay n discos, numerados del más pequeño al más grande como en la figura. Si los discos se encuentran apilados inicialmente en el pivote izquierdo, el problema de mover los n discos al pivote derecho se puede formular de la siguiente forma recursiva:</p> <ul style="list-style-type: none"> • Mover los $n-1$ discos superiores del pivote izquierdo (origen) al del centro. • Mover el n-ésimo disco (el más grande) al pivote de la derecha (destino). • Mover los $n-1$ discos del pivote del centro (auxiliar) al de la derecha. <p>Construir un programa que lea por teclado el número de discos e imprima en pantalla todos los movimientos necesarios para desplazar un número de discos dado por teclado, del pivote izquierdo al derecho. De cada movimiento se indicará el nº del disco desplazado (1,2,3,...,n), así como los pivotes origen y destino (izquierdo, centro, derecho).</p>
Análisis (Pre-diseño)	<p>Información:</p> <p>E: nº de discos: n (entero >0)</p> <p>S: movimientos necesarios para mover los n discos</p> <p>Nota: cada movimiento de un disco lo vamos a simular escribiendo en pantalla un texto del movimiento con el siguiente formato:</p> <p>Mover disco nº i del pivote $p1$ al pivote $p2$</p> <p>i: 1,...n</p> <p>$p1,p2$: izquierdo, centro y derecho</p>

	<p>Tareas:</p> <ul style="list-style-type: none">• Leer datos de entrada.<ul style="list-style-type: none">• Leer nº de discos.• Mover los discos.<ul style="list-style-type: none">• Mover un disco.<ul style="list-style-type: none">▪ Escribir nombre del pivote.																																						
Diseño	<p><u>Diseño preliminar</u></p> <p><u>Diseño de datos</u></p> <p><u>Estructura del programa</u></p> <div><div>Módulo principal</div><div><div>Leer entero positivo</div><div><div>Mover discos</div><div>Mover un disco</div><div>Escribir pivote</div></div></div></div> <p><u>Interfaces entre módulos</u></p> <table><tr><th></th><th colspan="3">Parámetros formales</th></tr><tr><th>Nombre módulo</th><th>Tipo</th><th>Nombre</th><th>Tipo de dato</th></tr><tr><td>Módulo principal</td><td></td><td></td><td></td></tr><tr><td>Leer_entero_positivo</td><td>S</td><td>n</td><td>entero</td></tr><tr><td rowspan="4">Mover discos</td><td rowspan="4">E</td><td>n</td><td>entero</td></tr><tr><td>ori</td><td>carácter</td></tr><tr><td>des</td><td>carácter</td></tr><tr><td>aux</td><td>carácter</td></tr><tr><td rowspan="3">Mover un disco</td><td rowspan="3">E</td><td>n</td><td>entero</td></tr><tr><td>ori</td><td>carácter</td></tr><tr><td>des</td><td>carácter</td></tr><tr><td>Escribir pivote</td><td>E</td><td>c</td><td>carácter</td></tr></table> <p><u>Diseño detallado</u></p> <p>Algoritmo Módulo_principal</p> <p>Var c: caracter</p> <p> n: entero</p> <p>Inicio Repetir Escribir("TORRES DE HANOI")</p> <p> Escribir("Numero de discos: ")</p> <p> leer_entero_positivo(n)</p> <p> mover_discos(n, 'i', 'd', 'c')</p> <p> Escribir("Desea efectuar una nueva operacion (s/n)? ")</p> <p> Leer(c)</p> <p> Hasta_que (c='N')ó(c='n')</p> <p>Fin algoritmo_principal</p>		Parámetros formales			Nombre módulo	Tipo	Nombre	Tipo de dato	Módulo principal				Leer_entero_positivo	S	n	entero	Mover discos	E	n	entero	ori	carácter	des	carácter	aux	carácter	Mover un disco	E	n	entero	ori	carácter	des	carácter	Escribir pivote	E	c	carácter
	Parámetros formales																																						
Nombre módulo	Tipo	Nombre	Tipo de dato																																				
Módulo principal																																							
Leer_entero_positivo	S	n	entero																																				
Mover discos	E	n	entero																																				
		ori	carácter																																				
		des	carácter																																				
		aux	carácter																																				
Mover un disco	E	n	entero																																				
		ori	carácter																																				
		des	carácter																																				
Escribir pivote	E	c	carácter																																				

	<p>Procedimiento leer_entero_positivo(n: entero (S))</p> <p>Inicio Repetir Escribir("Introduzca entero positivo: ") Leer(n) Hasta_que (n>0)</p> <p>Fin_procedimiento</p> <p><u>Diseño del módulo recursivo:</u> definición recursiva del problema:</p> <pre>mover_discos(n,ori,des,aux): n=1 → mover_un_disco(1,ori,des) → caso base n>1 → mover_discos(n-1,ori,aux,des) } mover_un_disco(n,ori,des) } → caso general mover_discos(n-1,aux,des,ori) }</pre> <p>Procedimiento mover_discos(n: entero (E), ori: carácter (E), des: carácter (E), aux: carácter (E))</p> <p>Inicio Si (n=1) Entonces mover_un_disco(1,ori,des) Sino mover_discos(n-1,ori,aux,des) mover_un_disco(n,ori,des) mover_discos(n-1,aux,des,ori)</p> <p> Fin_si</p> <p>Fin_procedimiento</p> <p>Procedimiento mover_un_disco(n: entero(E),ori: carácter(E), des: carácter(E))</p> <p>Inicio Escribir("Mover disco nº ",n, " del pivote ") escribir_pivote(ori) Escribir(" al pivote ") escribir_pivote(des)</p> <p>Fin_procedimiento</p> <p>Procedimiento escribir_pivote(c: carácter (E))</p> <p>Inicio Según_sea (c) Hacer 'i': Escribir("izquierdo") 'd': Escribir("derecho") 'i': Escribir("centro") Fin_según_sea</p> <p>Fin_procedimiento</p>
Codificación	<pre>#include <stdio.h> #include <stdlib.h> #include <conio.h> #include <ctype.h> #include <math.h> void leer_entero_positivo(int *n); void mover_discos(int n,char ori,char des,char aux); void mover_un_disco(int n,char ori,char des); void escribir_pivote(char c); int main(){ char c; int n; do{ system("cls"); printf("TORRES DE HANOI\n"); printf("=====\n\n"); printf("Numero de discos:\n");</pre>

	<pre> leer_entero_positivo(&n); mover_discos(n, 'i', 'd', 'c'); printf("\n\nDesea efectuar una nueva operacion (s/n)? "); c=toupper(getch()); }while (c!='N'); return 0; } void leer_entero_positivo(int *n){ do{ printf("\tIntroduzca entero positivo: "); scanf(" %d",n); }while(*n<=0); } void mover_discos(int n,char ori,char des,char aux){ if(n==1) mover_un_disco(1, ori, des); else{ mover_discos(n-1,ori,aux,des); mover_un_disco(n,ori,des); mover_discos(n-1,aux,des,ori); } } void mover_un_disco(int n,char ori,char des){ printf("\nMover disco num. %3d del pivote ",n); escribir_pivote(ori); printf(" al pivote "); escribir_pivote(des); } void escribir_pivote(char c){ switch(c){ case 'i': printf("%10s","izquierdo"); break; case 'd': printf("%10s","derecho"); break; case 'c': printf("%10s","centro"); break; } } </pre>
--	--

- Funciones como parámetros.

Se puede pasar un puntero a una función como argumento a otra función, permitiendo transferir una función (huésped) a otra función (anfitriona) como si la primera fuera una variable. En C el nombre del identificador de una función representa la dirección de memoria que contiene la primera instrucción de la misma.

Tipo de parámetro	Pseudo-código	Sintaxis de C
Función: fH(...)	Definición: funciónA(fH(...):tipo2(E),...): tipo1 ... fH(...):tipo2 ... Llamada: ... funciónA(fH,...)	Prototipo: tipo1 funciónA(tipo2 (*f)(...), ...); tipo2 fH(...); Llamada: ... funciónA(fH,...)

Nota: tipo (*función)(...) /* puntero a una función que devuelve un dato del tipo indicado */
 tipo *función(...) /* función que devuelve un puntero al tipo indicado */

Ejemplo resuelto: Funciones como parámetros de otras funciones	
Estrategia de diseño del módulo	<p>Objetivo: separar algoritmos de cálculo (integración numérica, derivación numérica, cálculo de ceros, cálculo de máximos y mínimos, resolución de ecuaciones diferenciales ordinarias,...) de las funciones sobre las que se aplica el algoritmo.</p> <p>Las funciones actúan como “datos” del algoritmo → parámetros de Entrada</p>
Ejercicio	<p>Calcular el valor de la derivada de una función $f(x)$ cualquiera en un punto x_0 y con una precisión (Δx) dadas, utilizando la siguiente aproximación (fórmula de 5 puntos):</p> $\left. \frac{df(x)}{dx} \right _{x=x_0} = \lim_{\Delta x \rightarrow 0} \frac{f(x_0-2*\Delta x) - 8*f(x_0-\Delta x) + 8*f(x_0+\Delta x) - f(x_0+2*\Delta x)}{12*\Delta x}$
Actividad funcional	<p>La tarea funcional del módulo consiste en calcular numéricamente el valor de la derivada de una función dada en un punto con una precisión determinada. Nótese que este módulo tiene una funcionalidad limitada y específica (módulo de entrada de cálculo: derivador numérico).</p>
Interfaz (lista de parámetros formales)	<p>Para poder realizar su trabajo funcional, este módulo requiere que se le indique a priori la función a derivar ($f: \mathbb{R} \rightarrow \mathbb{R}$), el punto en el que se va a calcular y la precisión con la que se va a calcular, y el resultado que devuelve al módulo llamador es el valor de la derivada en dicho punto:</p> <p>E: función $f(x: \text{real} (E))$: real { parámetro funcional }</p> <p>x_0 (real)</p> <p>h (real)</p> <p>S: (valor de la derivada) (real)</p> <p>El sub-programa se va a construir como una función (el único resultado se devuelve como un valor especial asociado a su identificador de subprograma). Nótese que el primer parámetro es una función con una interfaz determinada.</p>
Diseño del módulo	<p>Función derivada(Función $f(x: \text{real} (E))$: real, x_0: real (E), h: real (E)): real</p> <p>Inicio Devolver((8*(f(x₀+h)-f(x₀-h))-(f(x₀+2*h)-f(x₀-2*h)))/(12*h))</p> <p>Fin_función</p>
Prueba del módulo	<p>Los subprogramas no se pueden ejecutar directamente. Para poder realizar pruebas unitarias de un módulo es necesario construir un programa completo que incluya dicho módulo. Como el módulo tiene un parámetro formal funcional, debemos pasarle como correspondiente parámetro real funcional un identificador de una función con la misma interfaz predefinida o definida por el programador (en cuyo caso también hay que construirla).</p> <div style="text-align: center;"> <pre> graph TD MP[Módulo Principal] --- L1[] L1 --- derivada L1 --- f1 L1 --- f2 L1 --- f3 </pre> </div> <p>Algoritmo modulo_principal()</p> <p>Var c: carácter</p> <p> i: entero</p> <p> x₀,h: real</p> <p>Inicio Repetir Escribir("DERIVACION NUMERICA ")</p>

	<pre> Escribir("Prueba funcion: x**3-3*x**2+5") Escribir("Introduzca valor x0: ") Leer(x0) h←0.1 Desde i=1 hasta 6 Hacer Escribir("h=",h, " derivada=",derivada(f1,x0,h)) h←h/10 i←i+1 Fin_desde Escribir("Prueba funcion: exp(-x**2)") Escribir("Introduzca valor x0: ") Leer(x0) h←0.1 Desde i=1 hasta 6 Hacer Escribir("h=",h, " derivada=",derivada(f2,x0,h)) h←h/10 i←i+1 Fin_desde Escribir("Prueba funcion: seno(x)*exp(-x)") Escribir("Introduzca valor x0: ") Leer(x0) h←0.1 Desde i=1 hasta 6 Hacer Escribir("h=",h, " derivada=",derivada(f3,x0,h)) h←h/10 i←i+1 Fin_desde Escribir("Ejecutar de nuevo (s/n)? ") Leer(c) Hasta_que (c='N')ó(c='n') Fin_algoritmo_principal Función derivada(Función f(x: real (E)):real, x0: real (E), h: real (E)) Inicio Devolver((8*(f(x0+h)-f(x0-h))-(f(x0+2*h)-f(x0-2*h)))/(12*h)) Fin_función Función f1(x: real (E)): real Devolver (x*x*x-3*x*x+5) Fin_función Función f2(x: real (E)): real Devolver (exp(-x*x)) Fin_función Función f3(x: real (E)): real Devolver (sin(x)*exp(-x)) Fin_función </pre>
Codificación	<pre> #include <stdio.h> #include <stdlib.h> #include <conio.h> #include <ctype.h> #include <math.h> double derivada(double (*f)(double x), double x0, double h); </pre>

```

double f1(double x);
double f2(double x);
double f3(double x);

int main(){
    char c;
    int i;
    double x0,h;

    do{ system("cls");
        printf("DERIVACION NUMERICA\n");
        printf("=====\n\n");
        printf("Prueba derivacion numerica de funcion: x**3-3*x**2+5\n");
        printf("Introduzca valor x0: ");
        scanf(" %lf",&x0);
        h=0.1;
        for(i=1;i<7;++i){
            printf("\ni=%2d h=%.6lf\n",i,h);
            derivada=%.6lf",i,h,derivada(f1,x0,h));
            h=h/10;
        }
        printf("\n\nPrueba derivacion numerica de funcion: exp(-x**2)\n");
        printf("Introduzca valor x0: ");
        scanf(" %lf",&x0);
        h=0.1;
        for(i=1;i<7;++i){
            printf("\ni=%2d h=%.6lf\n",i,h);
            derivada=%.6lf",i,h,derivada(f2,x0,h));
            h=h/10;
        }
        printf("\n\nPrueba derivacion numerica de funcion: seno(x)*exp(-x)\n");
        printf("Introduzca valor x0: ");
        scanf(" %lf",&x0);
        h=0.1;
        for(i=1;i<7;++i){
            printf("\ni=%2d h=%.6lf\n",i,h);
            derivada=%.6lf",i,h,derivada(f3,x0,h));
            h=h/10;
        }
        printf("\n\nDesea efectuar una nueva operacion (s/n)? ");
        c=toupper(getch());
    }while (c!='N');
    return 0;
}

double derivada(double (*f)(double x), double x0, double h){
    return ((8*(f(x0+h)-f(x0-h)))-(f(x0+2*h)-f(x0-2*h)))/(12*h);
}

double f1(double x){
    return (x*x*x-3*x*x+5);
}

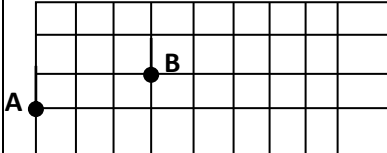
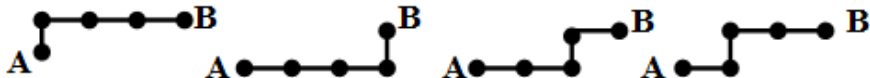
double f2(double x){
    return (exp(-x*x));
}

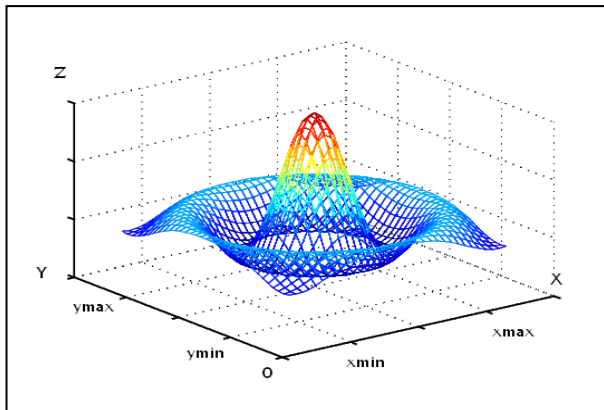
double f3(double x){
    return(sin(x)*exp(-x));
}

```

Ejercicios: desarrollo de programas

Ejercicio 1	Construya un módulo recursivo que devuelva la suma de las cifras de un número natural dado como argumento. Ejemplos: <div>n=12345 → suma de las cifras=15</div> <div>n=8902 → suma de las cifras=19</div> Construir un módulo no recursivo que resuelva el ejercicio anterior. ¿Qué ventajas e inconvenientes presenta la solución recursiva frente a la no recursiva?																							
Datos de prueba	<div>n=12345 →</div> <div>n=8902 →</div> <div>n=7 →</div> <div>n=100 →</div> <div>n=0 →</div>	<div>Suma de las cifras= 15</div> <div>Suma de las cifras= 19</div> <div>Suma de las cifras= 7</div> <div>Suma de las cifras= 1</div> <div>Suma de las cifras= 0</div>																						
Ejercicio 2	Construya un módulo recursivo que escriba en pantalla la representación binaria (en base 2) de un entero decimal positivo (base 10) dado. Nota: para convertir un número decimal en binario, dividir sucesivamente por 2 el número y los cocientes resultantes; la representación binaria se obtiene escribiendo en primer lugar el último cociente, seguido de los restos de las sucesivas divisiones en orden inverso a como se han generado. Ejemplo: <div><div><div>26</div><div>6</div><div>0</div></div><div><div>2</div><div>13</div><div>1</div></div><div><div>2</div><div>6</div><div>0</div></div><div><div>2</div><div>3</div><div>1</div></div><div><div>2</div><div>1</div></div></div> <div>(26)₁₀ = (11010)₂</div>																							
Datos de prueba		<table><tr><th>Decimal</th><th>Binario</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr><tr><td>15</td><td>1111</td></tr><tr><td>26</td><td>11010</td></tr><tr><td>128</td><td>10000000</td></tr><tr><td>1000</td><td>1111101000</td></tr></table>	Decimal	Binario	0	0	1	1	15	1111	26	11010	128	10000000	1000	1111101000								
Decimal	Binario																							
0	0																							
1	1																							
15	1111																							
26	11010																							
128	10000000																							
1000	1111101000																							
Ejercicio 3	<div>Construya un módulo recursivo que devuelva el resultado de la multiplicación de dos números naturales mediante duplicación y mediación (método del campesino ruso). El método consiste en lo siguiente:</div> <div><table><tr><th>A</th><th>B</th><th>Sumandos</th></tr><tr><td>27</td><td>82</td><td>82</td></tr><tr><td>13</td><td>164</td><td>164</td></tr><tr><td>6</td><td>328</td><td></td></tr><tr><td>3</td><td>656</td><td>656</td></tr><tr><td>1</td><td>1312</td><td>1312</td></tr><tr><td>Resultado</td><td></td><td>2214</td></tr></table><div><ul style="list-style-type: none">Se escriben los dos números a multiplicar A y B en la parte superior de sendas columnas.Se divide A entre 2 sucesivamente, ignorando el resto, hasta llegar a la unidad, escribiendo los resultados en la columna A.Se multiplica B por 2 tantas veces como se ha dividido A entre 2, escribiendo los resultados sucesivos en la columna B.Se suman todos los números de la columna B que estén al lado de un número impar de la columna A, y ése es el resultado.</div></div> <div>Suponiendo que el lenguaje de implementación no soporta el mecanismo de la</div>			A	B	Sumandos	27	82	82	13	164	164	6	328		3	656	656	1	1312	1312	Resultado		2214
A	B	Sumandos																						
27	82	82																						
13	164	164																						
6	328																							
3	656	656																						
1	1312	1312																						
Resultado		2214																						

	recursividad, construya un módulo que resuelva el ejercicio anterior. ¿Qué ventajas e inconvenientes presenta la solución recursiva frente a la no recursiva?																																																																				
Datos de prueba	<table><tr><th>A</th><th>B</th><th>Sumandos</th></tr><tr><td>27</td><td>82</td><td>82</td></tr><tr><td>13</td><td>164</td><td>164</td></tr><tr><td>6</td><td>328</td><td></td></tr><tr><td>3</td><td>656</td><td>656</td></tr><tr><td>1</td><td>1312</td><td>1312</td></tr><tr><td colspan="2">Resultado</td><td>2214</td></tr></table>	A	B	Sumandos	27	82	82	13	164	164	6	328		3	656	656	1	1312	1312	Resultado		2214	<table><tr><th>A</th><th>B</th><th>Sumandos</th></tr><tr><td>32</td><td>16</td><td></td></tr><tr><td>16</td><td>32</td><td></td></tr><tr><td>8</td><td>64</td><td></td></tr><tr><td>4</td><td>128</td><td></td></tr><tr><td>2</td><td>256</td><td></td></tr><tr><td>1</td><td>512</td><td>512</td></tr><tr><td colspan="2">Resultado</td><td>512</td></tr></table>	A	B	Sumandos	32	16		16	32		8	64		4	128		2	256		1	512	512	Resultado		512	<table><tr><th>A</th><th>B</th><th>Sumandos</th></tr><tr><td>17</td><td>15</td><td>15</td></tr><tr><td>8</td><td>30</td><td></td></tr><tr><td>4</td><td>60</td><td></td></tr><tr><td>2</td><td>120</td><td></td></tr><tr><td>1</td><td>240</td><td>240</td></tr><tr><td colspan="2">Resultado</td><td>255</td></tr></table>	A	B	Sumandos	17	15	15	8	30		4	60		2	120		1	240	240	Resultado		255
A	B	Sumandos																																																																			
27	82	82																																																																			
13	164	164																																																																			
6	328																																																																				
3	656	656																																																																			
1	1312	1312																																																																			
Resultado		2214																																																																			
A	B	Sumandos																																																																			
32	16																																																																				
16	32																																																																				
8	64																																																																				
4	128																																																																				
2	256																																																																				
1	512	512																																																																			
Resultado		512																																																																			
A	B	Sumandos																																																																			
17	15	15																																																																			
8	30																																																																				
4	60																																																																				
2	120																																																																				
1	240	240																																																																			
Resultado		255																																																																			
Ejercicio 4	<p>Construir un módulo recursivo que cuente y devuelva el número de caminos de dirección nordeste desde un punto a otro en una reja rectangular que representa una red de tuberías. Ej.: Un camino dirección nordeste es aquel por el cual se va desde un punto de partida a otro punto de la red moviéndose únicamente hacia el norte (hacia arriba) y hacia el este (hacia la derecha). En el ejemplo hay cuatro caminos nordeste:</p> <div><p>A=(1,2) B=(4,3)</p><div></div></div>																																																																				
Datos de prueba	<table><tr><th>A</th><th>B</th><th>Nº de caminos NE</th></tr><tr><td>(1,2)</td><td>(4,3)</td><td>4</td></tr><tr><td>(3,3)</td><td>(3,3)</td><td>0</td></tr><tr><td>(4,3)</td><td>(1,2)</td><td>0</td></tr><tr><td>(1,2)</td><td>(10,2)</td><td>1</td></tr><tr><td>(1,2)</td><td>(1,10)</td><td>1</td></tr><tr><td>(1,1)</td><td>(10,10)</td><td>48620</td></tr></table>	A	B	Nº de caminos NE	(1,2)	(4,3)	4	(3,3)	(3,3)	0	(4,3)	(1,2)	0	(1,2)	(10,2)	1	(1,2)	(1,10)	1	(1,1)	(10,10)	48620																																															
A	B	Nº de caminos NE																																																																			
(1,2)	(4,3)	4																																																																			
(3,3)	(3,3)	0																																																																			
(4,3)	(1,2)	0																																																																			
(1,2)	(10,2)	1																																																																			
(1,2)	(1,10)	1																																																																			
(1,1)	(10,10)	48620																																																																			
Ejercicio 5	Calcular el área bajo la curva de una función <i>f(x)</i> cualquiera en un intervalo (integral definida) mediante el método trapezoidal (ver ejercicio 2-43).																																																																				
Datos de prueba	<div>$f(x) = x^3 - 3x^2 + 5$<table><tr><th>a</th><th>b</th><th>base</th><th>Area</th></tr><tr><td rowspan="6">0</td><td rowspan="6">2</td><td>0.4</td><td>6.000000</td></tr><tr><td>0.3</td><td>5.995500</td></tr><tr><td>0.2</td><td>6.000000</td></tr><tr><td>0.1</td><td>6.000000</td></tr><tr><td>1e-2</td><td>6.000000</td></tr><tr><td>1e-5</td><td>6.000000</td></tr><tr><td rowspan="6">1.5</td><td rowspan="6">4</td><td>0.4</td><td>14.937250</td></tr><tr><td>0.3</td><td>14.794450</td></tr><tr><td>0.2</td><td>14.692450</td></tr><tr><td>0.1</td><td>14.631250</td></tr><tr><td>1e-2</td><td>14.609594</td></tr><tr><td>1e-5</td><td>14.609375</td></tr></table></div>			a	b	base	Area	0	2	0.4	6.000000	0.3	5.995500	0.2	6.000000	0.1	6.000000	1e-2	6.000000	1e-5	6.000000	1.5	4	0.4	14.937250	0.3	14.794450	0.2	14.692450	0.1	14.631250	1e-2	14.609594	1e-5	14.609375																																		
a	b	base	Area																																																																		
0	2	0.4	6.000000																																																																		
		0.3	5.995500																																																																		
		0.2	6.000000																																																																		
		0.1	6.000000																																																																		
		1e-2	6.000000																																																																		
		1e-5	6.000000																																																																		
1.5	4	0.4	14.937250																																																																		
		0.3	14.794450																																																																		
		0.2	14.692450																																																																		
		0.1	14.631250																																																																		
		1e-2	14.609594																																																																		
		1e-5	14.609375																																																																		

	<div><div>$f(x) = \exp(-x^2)$</div><table><tr><th>a</th><th>b</th><th>base</th><th>Area</th></tr><tr><td>-1</td><td>1</td><td>1e-5</td><td>1.493648</td></tr><tr><td>-5</td><td>5</td><td>1e-5</td><td>1.772454</td></tr><tr><td>-10</td><td>10</td><td>1e-5</td><td>1.772454</td></tr></table></div> <div><div>$f(x) = \text{seno}(x)*\exp(-x)$</div><table><tr><th>a</th><th>b</th><th>base</th><th>Area</th></tr><tr><td>0</td><td>1.5708</td><td>1e-5</td><td>0.396061</td></tr><tr><td>0</td><td>3.1416</td><td>1e-5</td><td>0.521607</td></tr><tr><td>0</td><td>6.2832</td><td>1e-5</td><td>0.499066</td></tr></table></div>	a	b	base	Area	-1	1	1e-5	1.493648	-5	5	1e-5	1.772454	-10	10	1e-5	1.772454	a	b	base	Area	0	1.5708	1e-5	0.396061	0	3.1416	1e-5	0.521607	0	6.2832	1e-5	0.499066																				
a	b	base	Area																																																		
-1	1	1e-5	1.493648																																																		
-5	5	1e-5	1.772454																																																		
-10	10	1e-5	1.772454																																																		
a	b	base	Area																																																		
0	1.5708	1e-5	0.396061																																																		
0	3.1416	1e-5	0.521607																																																		
0	6.2832	1e-5	0.499066																																																		
Ejercicio 6	<div><div>Construya un módulo que calcule numéricamente la integral doble de una función $f(x,y)$ cualquiera sobre un rectángulo, esto es, que determine el volumen encerrado por la superficie $f(x,y)$ y el plano XY dentro de los límites del rectángulo:</div><div><div>$\int_{x_{\min}}^{x_{\max}} \int_{y_{\min}}^{y_{\max}} f(x,y) \, dx \, dy = \sum_{i=1}^m \sum_{j=1}^n f(x_i, y_j) \cdot A_{ij}$</div></div><div><div>Nota: para el cálculo de la integral doble sobre un rectángulo, se divide el rectángulo en pequeñas áreas rectangulares y se aproxima el volumen encerrado por la superficie y cada área por un prisma rectangular de altura el valor de la función en el punto central del área. El número de divisiones horizontales m y verticales n también será pasado como parámetro.</div><div>Construya a continuación un programa que calcule las siguientes integrales:</div><table><tr><th>$f(x,y)$</th><th>x_{\min}</th><th>x_{\max}</th><th>y_{\min}</th><th>y_{\max}</th><th>m</th><th>n</th></tr><tr><td>$\log(x+2y)$</td><td>1.4</td><td>2.0</td><td>1.0</td><td>1.5</td><td>100</td><td>100</td></tr><tr><td>x^2+y^2</td><td>2.0</td><td>4.0</td><td>1.0</td><td>2.0</td><td>50</td><td>50</td></tr></table></div></div>	$f(x,y)$	x_{\min}	x_{\max}	y_{\min}	y_{\max}	m	n	$\log(x+2y)$	1.4	2.0	1.0	1.5	100	100	x^2+y^2	2.0	4.0	1.0	2.0	50	50																															
$f(x,y)$	x_{\min}	x_{\max}	y_{\min}	y_{\max}	m	n																																															
$\log(x+2y)$	1.4	2.0	1.0	1.5	100	100																																															
x^2+y^2	2.0	4.0	1.0	2.0	50	50																																															
Datos de prueba	<table><tr><th>$f(x,y)$</th><th>x_{\min}</th><th>x_{\max}</th><th>y_{\min}</th><th>y_{\max}</th><th>m</th><th>n</th><th>Integral doble</th></tr><tr><td rowspan="3">$\log(x+2y)$</td><td rowspan="3">1.4</td><td rowspan="3">2.0</td><td rowspan="3">1.0</td><td rowspan="3">1.5</td><td>100</td><td>100</td><td>0.429554625823904</td></tr><tr><td>1000</td><td>1000</td><td>0.429554528530990</td></tr><tr><td></td><td></td><td></td></tr><tr><td rowspan="3">x^2+y^2</td><td rowspan="3">2.0</td><td rowspan="3">4.0</td><td rowspan="3">1.0</td><td rowspan="3">2.0</td><td>50</td><td>50</td><td>23.333000000000016</td></tr><tr><td>100</td><td>100</td><td>23.333249999999968</td></tr><tr><td>1000</td><td>1000</td><td>23.333332500000225</td></tr><tr><td>$x+y$</td><td>1</td><td>2</td><td>0</td><td>3</td><td>10</td><td>10</td><td>9.000000000000000</td></tr><tr><td>1</td><td>0</td><td>2</td><td>1</td><td>3</td><td>10</td><td>10</td><td>4.000000000000000</td></tr></table>	$f(x,y)$	x_{\min}	x_{\max}	y_{\min}	y_{\max}	m	n	Integral doble	$\log(x+2y)$	1.4	2.0	1.0	1.5	100	100	0.429554625823904	1000	1000	0.429554528530990				x^2+y^2	2.0	4.0	1.0	2.0	50	50	23.333000000000016	100	100	23.333249999999968	1000	1000	23.333332500000225	$x+y$	1	2	0	3	10	10	9.000000000000000	1	0	2	1	3	10	10	4.000000000000000
$f(x,y)$	x_{\min}	x_{\max}	y_{\min}	y_{\max}	m	n	Integral doble																																														
$\log(x+2y)$	1.4	2.0	1.0	1.5	100	100	0.429554625823904																																														
					1000	1000	0.429554528530990																																														
x^2+y^2	2.0	4.0	1.0	2.0	50	50	23.333000000000016																																														
					100	100	23.333249999999968																																														
					1000	1000	23.333332500000225																																														
$x+y$	1	2	0	3	10	10	9.000000000000000																																														
1	0	2	1	3	10	10	4.000000000000000																																														

Ejercicio 7	<div>Construir un programa que permita obtener, mediante aproximaciones sucesivas, una raíz real de una función $f(x)$ dada (esto es, una solución aproximada de $f(x)=0$), utilizando la técnica de Newton-Raphson que emplea la siguiente relación:</div> <div>$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$<div><div>$x_i$: valor estimado actual</div><div>x_{i+1}: siguiente valor calculado</div><div>$f(x_i)$: función evaluada en x_i</div></div></div> <div>Para muchas funciones, esta secuencia de aproximaciones x_1, x_2, x_3, \dots converge a la solución, teniendo como única condición que la primera aproximación (x_0) esté lo suficientemente cerca de la solución. El proceso requiere como entrada una estimación inicial x_0, y deberá concluirse cuando $x_{i+1}-x_i <E$, o el número de iteraciones exceda algún límite superior dado o $f'(x_i)=0$. Los valores x_0, E y el número máximo de iteraciones serán introducidos por teclado, mostrándose en pantalla bien la solución aproximada o bien un mensaje indicando que ésta no se ha conseguido y la causa (excedido nº de iteraciones o encontrada una derivada nula), presentando en pantalla el valor de la última aproximación calculada.</div>																																																		
Datos de prueba		<table><tr><th>$f(x)$</th><th>x_0</th><th>E</th><th>n</th><th>Raíz</th></tr><tr><td rowspan="2">$x^3 - 3x^2 + 5$</td><td>3</td><td>1e-6</td><td>100</td><td>-1.103803</td></tr><tr><td>1000</td><td>1e-6</td><td>100</td><td>-1.103803</td></tr><tr><td rowspan="3">$x^2 - 1$</td><td>5</td><td>1e-6</td><td>100</td><td>1.000000</td></tr><tr><td>-5</td><td>1e-6</td><td>100</td><td>-1.000000</td></tr><tr><td>0</td><td>1e-6</td><td>100</td><td>Derivada nula</td></tr><tr><td>$x^2 + 1$</td><td>5</td><td>1e-6</td><td>100</td><td>Excedido nº iteraciones</td></tr><tr><td>$\exp(-x^2)$</td><td>100</td><td>1e-6</td><td>100</td><td>Derivada nula</td></tr><tr><td rowspan="2">$\text{seno}(x) \cdot \exp(-x)$</td><td>1.5</td><td>1e-6</td><td>100</td><td>3.141593</td></tr><tr><td>5</td><td>1e-6</td><td>100</td><td>6.283185</td></tr></table>	$f(x)$	x_0	E	n	Raíz	$x^3 - 3x^2 + 5$	3	1e-6	100	-1.103803	1000	1e-6	100	-1.103803	$x^2 - 1$	5	1e-6	100	1.000000	-5	1e-6	100	-1.000000	0	1e-6	100	Derivada nula	$x^2 + 1$	5	1e-6	100	Excedido nº iteraciones	$\exp(-x^2)$	100	1e-6	100	Derivada nula	$\text{seno}(x) \cdot \exp(-x)$	1.5	1e-6	100	3.141593	5	1e-6	100	6.283185			
$f(x)$	x_0	E	n	Raíz																																															
$x^3 - 3x^2 + 5$	3	1e-6	100	-1.103803																																															
	1000	1e-6	100	-1.103803																																															
$x^2 - 1$	5	1e-6	100	1.000000																																															
	-5	1e-6	100	-1.000000																																															
	0	1e-6	100	Derivada nula																																															
$x^2 + 1$	5	1e-6	100	Excedido nº iteraciones																																															
$\exp(-x^2)$	100	1e-6	100	Derivada nula																																															
$\text{seno}(x) \cdot \exp(-x)$	1.5	1e-6	100	3.141593																																															
	5	1e-6	100	6.283185																																															

Asignatura	Programación		
Plan de Estudios	Grados en Ingeniero Mecánico, Eléctrico, Electrónico Industrial y Químico Industrial		
Actividad	Trabajo individual	Sesión	7
Tiempo empleado			

Apellidos, nombre	DNI	Firma

1.- Ejercicios.

1. En lenguaje de programación C, ¿qué es un puntero a una función? ¿Cómo se declara sintácticamente? ¿Para qué se utiliza? ¿Qué representa el nombre o identificador de una función?
2. Implementar en C los siguientes módulos:

Definición del módulo
Función fejexbr2(f(x:real(E),y:real(E)):real(E), g(x:real(E)):real(E), x0:real(E),y0:real(E)):real Inicio Devolver(f(x0,y0)*g(x0)/(f(y0,x0)*g(y0))) Fin_función
Función h(función f(x: real (E), y: real (E)):carácter (E), x0: real (E), y0: real (E)): logico Inicio Si (f(x0,y0)=f(y0,x0)) Entonces Devolver(verdadero) Sino Devolver(falso) Fin_si Fin_función

2.- Resultados de aprendizaje (reflexión): marque con una cruz los objetivos que cree haber alcanzado tras realizar esta actividad, y rellene en el campo de observaciones aquellos aspectos que cree que necesita mejorar (si los hubiera):

Objetivos formativos	Cumplimiento
<ul style="list-style-type: none"> • Conocer la estructura sintáctica de un programa modular en C: definiciones de funciones, prototipos de funciones, bibliotecas de funciones, paso de parámetros, tipo de dato puntero, funciones como parámetros, tipos de almacenamiento automático, externo, estático y registro. • Implementar programas modulares en lenguaje de programación C. Identificar y corregir errores sintácticos que surgen durante la codificación. • Construir módulos individuales (subprogramas), identificando su actividad funcional (entrada, salida, cálculo,...) y su interfaz para la comunicación de información con otros módulos (lista de parámetros formales + valor especial asociado al identificador del módulo). Distinguir entre parámetros de entrada y de salida. Documentar adecuadamente estos módulos mediante la notación sub-algorítmica. 	

<ul style="list-style-type: none">• Descomponer un problema en sub-problemas, identificando a partir del enunciado del mismo la información a procesar y la lista de tareas de manipulación de información; organizar esta lista de tareas de forma jerárquica, descomponiendo tareas complejas en tareas más simples. Representar mediante un diagrama en forma de árbol la estructura del programa.• <i>Resolver problemas sencillos mediante recursividad como alternativa a la construcción de repetición: expresar el problema de forma recursiva distinguiendo entre el caso general y el caso base.</i>• <i>Utilizar módulos como parámetros con el objetivo de separar algunos algoritmos de cálculo de las funciones sobre las que actúan.</i>• Probar con datos operacionales la correctitud de los módulos individuales (pruebas unitarias) y de los programas desarrollados e identificar y corregir los errores lógicos que surjan.	
Observaciones	