Problemas de Cálculo con C

1. Escriba una función que, pasándole un número entero de hasta 10 cifras decimales, retorne un número entero con las cifras decimales en orden inverso.

   Solución:

```
unsigned int invierte(unsigned int x) {
    int y, u;
    y=0;
    while( x > 0 ) {
        u= x % 10;
        y=10*y+u;
        x= x / 10;
    }
    return y;
}
```

2. Supongamos que tenemos un archivo de datos `datos.dat` con N pares de puntos que representan el comportamiento de una función $y(x)$ en un intervalo dado. La primera columna del archivo corresponde a los valores x_i (que están ordenados de menor a mayor) mientras que la segunda muestra los valores de la función $y(x_i)$. Escribir un programa que lea esos puntos y obtenga los valores x_i en los que la función muestra los mínimos y máximos relativos dentro del intervalo. El valor de N debe ser introducido por línea de comandos como argumento de la función `main()` cuando se ejecuta el programa. El programa debe mostrar en pantalla el resultado del siguiente modo (es un ejemplo):

   ```
   La función tiene tres mínimos relativos en los puntos x = 0.1, 1.2, 4.5
   La función tiene dos máximos relativos en los puntos x = 0.9, 2.9
   ```

   Del mismo modo el programa también deberá indicar si la función no tiene mínimos ni máximos.

   Solución: Ver código `funcion_maximos_minimos.c`

3. El objetivo de este ejercicio es definir una función (que llamaremos escribe_valores) que escriba en un archivo de texto los valores de cualquier función matemática $f(x)$ en los puntos x_i pertenecientes al intervalo $[a, b]$ y separados por una distancia Δx. La función debe escribir en el archivo las parejas de puntos $(x_i, f(x_i))$ para luego poder ser representados con Gnuplot.

   Los argumentos de la función escribe_valores serán los extremos del intervalo así como la separación entre los puntos, que serán pasados por valor, mientras que la función matemática que se quiere representar, que habrá sido definida en otra parte del programa, será pasada por referencia. Finalmente, se pasará también como argumento el nombre del archivo en el que se quieren escribir los datos. Una posible llamada a la función desde el programa principal podría ser

```
escribe_valores(1.0, 2.0, 0.1, &parabola, "datos.dat")
```

   donde previamente se debería haber definido la función parábola, por ejemplo:

```
double parabola(double x) {
        return (2*x*x-1);
}
```

Solución: Ver código `funcion_exportar_puntos.c`

4. Construya una función en C que, dado un vector de $n+1$ coeficientes $a[\,]$ y un valor de la variable x, evalúe la función polinomial

$$P(x) = a[0] + a[1] * x + a[2] * x * x + \cdots + a[n] * \underbrace{x * \cdots * x}_{n \text{ factores}}$$

Solución:

```
double evalPoly(int n, double a[], double x) {
    int m;
    double y=a[0], xm=1;
    for(m=1; m<=n; m++) {
        xm=xm*x;
        y+=a[m]*xm;
    }
    return y;
}
```

5. Construya una función en C que, dados los $2n+1$ coeficientes a_0, $a_1, \ldots a_n$ y $b_1, \ldots, b_n$, evalúe el polinomio trigonométrico:

$$F(x) = a_0 + \sum_{i=1}^{n} (a_i \cos ix + b_i \sin ix)$$

Solución:

```
double evalTrig(int n, double a[], double b[], double x) {
    int m;
    double y=a[0];
    for(m=1; m<=n; m++) {
        y+=a[m]*cos(m*x)+b[m]*sin(m*x);
    }
    return y;
}
```

6. Escribir una función que, dados el día y mes de un año, y la hora, minutos y segundos transcurridos de ese día, calcule el número de días y fracción transcurridos desde el 1 de enero (fecha juliana), retornando un único valor de tipo `double`. [**Sugerencia:** usar un array global con el número de días de cada mes] [**Nota:** no es necesario tener en cuenta que el año sea bisiesto]

Solución:

```
static int dias_mes[]={31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
double fechaJ(int mes, int dia, int hora, int mint, int segd) {
    int n;
    double jdate=0;
    for(n=0; n<mes-1; n++) {
        jdate+=dias_mes[n];
```

```
    }
    jdate+=dia-1;
    jdate+=((double)hora+(double)mint/60+(double)segd/3600)/24;
    return jdate;
}
```

7. La sucesión de Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ... es la sucesión infinita de números naturales en la que sus dos primeros términos son unos y a partir de ahí cada elemento es la suma de los dos anteriores: $F_{n>2} = F_{n-1} + F_{n-2}$. Se sabe que $\varphi = \lim_{n\to\infty} \frac{F_{n+1}}{F_n}$, donde φ es el número áureo $\varphi = \frac{1+\sqrt{5}}{2} \simeq 1.6180339887\ldots$, un número irracional con propiedades matemáticas muy interesantes y que representa una proporción que aparece frecuentemente en la naturaleza.

   Construir un programa en C que calcule los términos de esta sucesión hasta obtener el valor de φ con una precisión de n cifras decimales. El programa debe pedir al usuario la precisión deseada. [**Ayuda:** Para saber que hemos llegado a la precisión deseada podemos imponer la siguiente condición en cada iteración: cuando el módulo de la diferencia entre dos valores F_{n+1}/F_n consecutivos de sea menor que 10^{-n}, eso significará que hemos llegado a la precisión deseada]

   Solución: Ver código `fibonacci.c`

8. Explicar detalladamente qué realiza el siguiente programa:

```
#include <stdio.h>
#include <math.h>
#include <stdio.h>

double *funcion(double *v) {
    int i, j;
    double num;
    num=v[0];
    i=0;
    for(j=1; j<6; j++) {
        if (*(v+j) < num) {
            num=*(v+j);
            i=j;
        }
    }
    return (v+i);
}

int main(int argc, char** argv) {
    double *p;
    double v[6]={-22.3, 5.4, 11.6, -2.1, -4.7, 10};
    p=funcion(v);
    printf ("el resultado del programa es %g\n", *p);
    return 0;
}
```

   Solución: El programa calcula el menor valor de una lista de reales definida en el vector `v`. Para ello llama a `funcion`, que devuelve un puntero a ese valor.

9. Escribir una función que, dada una lista de números complejos en la forma de un array de estructuras

```
struct complexf {
        float re, im;
};
```

devuelva el módulo del número complejo resultante de sumar todos ellos.

Solución:

```
float mod_suma(int N, struct complexf x[]) {
    int i;
    float m;
    struct complexf s;
        s.re=0.0;
        s.im=0.0;
    for(i=0; i < N; i++) {
        s.re+=x[i].re;
        s.im+=x[i].im;
    }
    m=sqrt(s.re*s.re + s.im*s.im);
    return p;
}
```

10. Escribir una función que, dada una lista de números complejos en la forma de un array de estructuras

```
struct complexf {
        float re, im;
};
```

devuelva el número complejo resultante de multiplicar todos ellos.

Solución:

```
struct complexf complex_prod(int N, struct complexf x[]) {
    int i;
    float aux;
    struct complexf p;
        p.re=1.0;
        p.im=0.0;
    for(i=0; i < N; i++) {
        aux=p.re;
        p.re=p.re*x[i].re-p.im*x[i].im;
        p.im=aux*x[i].im+p.im*x[i].re;
    }
    return p;
}
```

11. Supongamos que tenemos un archivo de datos "datos.dat" con N pares de puntos que representan el comportamiento de una función en un intervalo dado. Escribir un programa que lea esos puntos y obtenga los valores de x en los que la función alcanza el valor máximo y el valor mínimo dentro del intervalo.

Solución:

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main(int argc, char** argv) {
    int i, n, npts;
    double xx, yy, ymin, ymax, xmin, xmax;
    FILE *fin;
    fin=fopen("datos.dat", "r");
    fscanf(fin, "%lf %lf", &xx, &yy);
    xmin=xmax=xx;
    ymin=ymax=yy;
    do{
        n=fscanf(fin, "%lf %lf", &xx, &yy);
        if (yy < ymin) {ymin=yy; xmin=xx;}
        if (yy > ymax) {ymax=yy; xmax=xx;}
    } while(n==2);
    fclose(fin);
    printf("el minimo global de la funcion es (%g, %g)\n", xmin, ymin);
    printf("el maximo global de la funcion es (%g, %g)\n", xmax, ymax);
    return 0;
}
```

12. Los números primos son aquellos números naturales mayores que 1, cuyos únicos divisores (el resto de la división es cero) son ellos mismos y el 1. Por ejemplo, el 5 es primo ya que sus únicos divisores son el 1 y el 5. El 6 no es primo ya que, aparte del 1 y del 6, tiene como divisores el 2 y el 3. Escriba un programa en C que calcule e imprima en un archivo, los N primeros números primos. Tanto el valor de N como el nombre del archivo de texto en el que imprimirán los números primos, deberán ser introducidos por línea de comandos como argumentos de la función `main()`.

    Solución:

```
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char **argv) {
    int i, cont, num, N;
    FILE* fout;
    N=atoi(argv[1]);
    fout=fopen(argv[2],"wb");
    cont=0;
    num=2;
    do {
        for(i=2; i<=num; i++) {
            if(num%i==0) break;
        }
        if(i==num) {
            fprintf(fout,"%d\n",num);
            cont++;
        }
        num++;
    }
    while (cont<N);
    fclose(fout);
    return 0;
}
```

13. Escriba la implementación de la función

```
double lever(double c0, double c1, double c2, double *x2);
```

que devuelva los valores x_1 (como valor retornado) y x_2 (en el último argumento) tales que se cumpla el sistema de ecuaciones:

$$x_1 + x_2 = 1$$
$$c_1 x_1 + c_2 x_2 = c_0$$

Solución:

```
double lever(double c0, double c1, double c2, double *x2) {
    double x1;
        x1=(c0-c2)/(c1-c2);
        *x2=1-x1;
    return x1;
}
```

14. El más irracional de los números es la razón aurea, $\varphi \simeq 1.6180339887\ldots$ Este número se puede calcular como la fracción continua

$$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1+\ldots}}}}$$

Escríbase una función en C que calcule el valor de esta constante matemática de la forma iterativa mostrada en esta fracción continua, parando cuando dos iteraciones consecutivas proporcionen valores que se diferencien en menos de 10^{-15}.

Solución:

```
double phi() {
    double f=1, fa;
    do {
        fa=f;
        f=1+1/f;
    } while( fabs(fa-f)>1e-15 );
    return f;
}
```