Problemas-Tema-3.pdf

JuaneGC

Fundamentos de Programación

2º Grado en Ingeniería de las Tecnologías de Telecomunicación

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

EJERCICIO 4

/* Realizar un programa para que escriba todos los números primos entre 1 y N, siendo N un número introducido por el usuario. */

```
#include<stdio.h>

int primo(int n){

  int comprobante = 1, i = 2;

  while(comprobante == 1 && i <= n / 2){ /* Comprobamos hasta que el número o no sea primo o hasta que lleguemos a n / 2. */

    if(n % i == 0)
      comprobante = 0;            /* Al ser divisible por un i cualquiera, quiere decir que el número es primo. */
    else
      i++;

    }

  return(comprobante);            /* Devuelve una variable booleana que nos dice si el número es o no primo. */
}

int main(){

  int n, i;

  do{

    printf("Introduzca el número N del intervalo [1, N] en el que buscar primos: ");
    scanf("%d", &n);

  }while(n <= 1);
```

```
  for(i = 1; i <= n; i++){              /* Bucle que recorre el intervalo en el que buscar primos. */

    if(primo(i) == 1)
      printf("\nEl número %d es primo.\n", i);

  }

  return(0);
}
```

BBVA

1/6	BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.
Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.	

Ábrete la Cuenta Online de **BBVA** y llévate 1 año de **Wuolah PRO**

cómo??

Las ventajas de WUOLAH PRO

Di adiós a la publi en los apuntes y en la web

Descarga carpetas completas de un tirón

Acumula tickets para los sorteos

Ventajas Cuenta Online de BBVA

0€
Sin comisión de administración o mantenimiento de **cuenta**.
(0 % TIN **0 % TAE**)

0€
Sin comisión por emisión y mantenimiento de T**arjeta** Aqua débito.

0
Sin necesidad de domiciliar nómina o recibos.

EJERCICIO 5

/* Escribir una función en C para aceptar mensajes de confirmación. La función debe imprimir en la pantalla el mensaje ¿Confirmar (S/N)? y solo aceptará las pulsaciones de las telclas S y N (en mayúscula o minúscula). La función devolverá false (0) si se ha pulsado N o bien true (1) si se ha pulsado S. Hacer un main que muestre como se usa la función. */

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

void confirmacion(){

  int comprobante = -1;              /* Valor booleano que comprueba si se ha pulsado S o N. Con -1 comprueba que la tecla no es correcta. */
  char tecla;

  do{

    printf("\n¿Confirmar (S/N)? ");
    scanf("%c", &tecla);
    tecla = toupper(tecla);          /* Hacemos mayúscula el caracter introducido. */

  }while(tecla < 78 || (tecla > 78 && tecla < 83) || tecla > 83);

    if(tecla == 'S')
      comprobante = 1;               /* Como hemos introducido S, el valor es true. */
    else if(tecla == 'N')
      comprobante = 0;               /* Como hemos introducido N, el valor es false. */

  if(comprobante == 1)                 /* Comprobación de la respuesta. */
    printf("\n\nUsted ha confirmado.\n");
  else
    printf("\n\nUsted no ha confirmado.\n");

}
```

```
int main(){

  printf("\n### EJERCICIO 5 ###\n\n");
  printf("Este programa implementa la siguiente función: \n"
      "\t Cuando la llamamos nos preguntará un mensaje de confirmación.\n"
      "\t Deberá responder Si o No y esta guardará un valor true o false.\n");

  confirmacion();       /* Almacenamos la respuesta introducida (true o false). */

  return(0);
}
```

EJERCICIO 6

/* Implemente una función en C con un argumento de tipo caracter que:

* si el argumento es una letra en mayúscula, devuelve su correspondiente letra en minúscula,

* si el argumento no es una letra mayúscula, devuelve el mismo argumento.

Hacer un pequeño main que ilustre el uso de la función. */

```
#include<stdio.h>

#include<ctype.h>


char minuscula(char letra){


  char resultado;


  resultado = tolower(letra);       /* Si la la letra es mayúscula, la pasa a minúscula, sino la mantendrá como estaba.
*/


  return(resultado);

}


int main(){


  char letra1, letra2;             /* Aquí almacenaremos la letra antes y después de aplicar la función. */


  do{


    printf("\nIntroduzca una letra: ");

    scanf("%c", &letra1);


  }while(letra1 < 65 || (letra1 > 90 && letra1 < 97) || letra1 > 122); /* Buscamos que esté dentro de las letras de
ASCII. */


  letra2 = minuscula(letra1);
```

```
  printf("\nUsted introdujo %c y la función ha devuelto %c.\n\n", letra1, letra2);


  return(0);
}
```

EJERCICIO 7

/* Escribir en C la función int MCD(int a, int b) que devuelve el máximo común divisor de dos números enteros. Hacer un main para ilustrar su uso. */

```
#include<stdio.h>

int MCD(int a, int b){

  int mcd, i = 1;

  while(i <= a && i <= b){          /* Paramos el bucle cuando el contador se hace mayor que cualquiera de los números introducidos. */

    if(a % i == 0 && b % i == 0)       /* Si el contador del bucle es divisor lo almacenamos en la variable mcd. */
      mcd = i;

    i++;

  }

  return(mcd);
}

int main(){

  int x, y, mcd;

  printf("Introduzca los números de los que quiera averiguar el M.D.C.: ");
  scanf("%d %d", &x, &y);

  mcd = MCD(x, y);

  printf("\nEl M.C.D. de %d y %d es %d.\n\n", x, y, mcd);

  return(0);
```

```
}
```

EJERCICIO 8

/* Escribir un programa modular que presente un menú para calcular la potencia, factorial y combinatorio de ciertos valores leídos desde el

teclado y muestre en pantalla el resultado de la operación. Hacer primero la descomposición modular. */

```
#include<stdio.h>

#include<stdlib.h>

#include<math.h>


int potencia(int base, int exponente){


  int pot = 1, i;


  for(i = 1; i <= exponente; i++)          /* Una potencia es un producto de la base por si misma "exponente" veces.
*/
    pot *= base;


  return(pot);
}


int factorial(int n){


  int fact = 1, i;


  for(i = 1; i <= n; i++)              /* El factorial de n se define como n! = n * (n - 1)! */
    fact *= i;


  return(fact);
}


int combinatorio(int a, int b){


  int comb;
  int m, n;                        /* Necesitamos operar sabiendo que número es el mayor y cual es el menor. */
```

```
  if(a > b){

    m = a;
    n = b;

  }
  else{

    m = b;
    n = a;

  }

  comb = factorial(m) / (factorial(n) * factorial(m - n)); /* Aplicamos la fórmula del combinatorio: (m n) = m! / (n! *
(m - n)!). */

  return(comb);
}

int main(){

  int opcion;
  int base, exponente, poten;                /* Variables del apartado 1. */
  int n, fac;                            /* Variables del apartado 2. */
  int a, b, comb;                          /* Variables del apartado 3. */

  printf("### EJERCICIO 8 ###\n\n"
      "Este programa puede realizar las siguientes operaciones:\n\n"
      "\t1) POTENCIA\n"
      "\t2) FACTORIAL\n"
      "\t3) COMBINATORIO\n");

  do{
```

```
  printf("\nIntroduzca la opción que quiera realizar: ");
  scanf("%d", &opcion);

}while(opcion < 1 || opcion > 3);          /* Si la opción no es valida, repetimos la pregunta. */

switch(opcion){

  case 1:                          /* Caso del cálculo de la potencia. */
    system("cls");
    printf("Ha elegido la opción 1, la operación de potencia.\n");
    do{

      printf("\nIntroduzca la base y el exponente (positivos): ");
      scanf("%d %d", &base, &exponente);

    }while(base <= 0 || exponente <= 0);
    poten = potencia(base, exponente);
    printf("\nEl resultado de %d^%d es %d.\n\n", base, exponente, poten);
    break;

  case 2:                          /* Caso del cálculo del factorial. */
    system("cls");
    printf("Ha elegido la opción 2, la operación del factorial.\n");
    do{

      printf("\nIntroduzca el número positivo al que calcular el factorial: ");
      scanf("%d", &n);

    }while(n <= 0);
    fac = factorial(n);
    printf("\nEl resultado de %d! es %d.\n\n", n, fac);
    break;
```

```
    default:                        /* Caso del cálculo del combinatorio. Tomamos el default para ahorrar la
comprobación. */
        system("cls");
        printf("Ha elegido la opción 3, la operación del combinatorio.\n");
        do{

          printf("\nIntroduzca los dos números positivos para calcular el combinatorio: ");
          scanf("%d %d", &a, &b);

        }while(a <= 0 || b <= 0);
        comb = combinatorio(a, b);
        printf("\nEl resultado de combinar %d y %d es %d.\n\n", a, b, comb);

    }

    return(0);
}
```

EJERCICIO 9

/* Escribir una función que nos diga si un año dado (como un entero) es o no bisiesto. Los años bisiestos son aquellos que bien son múltiplo de 4 pero no de 100, o bien son múltiplo de 400. Además, hay que tener en cuenta que este calendario empezó a aplicarse a partir de 1582. */

```
#include<stdio.h>

int anobisiesto(int ano){

  int bisiesto = 0;                    /* Booleano para saber si es el año bisiesto. */

  if((ano % 4 == 0 && ano % 100 != 0) || ano % 400 ==0) /* Si se cumplen las condiciones es bisiesto. */
    bisiesto = 1;

  return(bisiesto);
}

int main(){

  int ano;
  int bisiesto;                    /* Variable booleana que indica si el año es o no bisiesto. */

  printf("Introduzca un año: ");
  scanf("%d", &ano);

  bisiesto = anobisiesto(ano);

  if(bisiesto == 1)
    printf("\nEl año sí es bisiesto.\n\n");
  else
    printf("\nEl año no es bisiesto.\n\n");

  return(0);
}
```

EJERCICIO 10

/* Escribir una función que, dados tres enteros representando el día, mes y año, nos diga si corresponden a una fecha correcta. */

```
#include<stdio.h>

int anobisiesto(int ano){

  int bisiesto = 0;                    /* Booleano para saber si es el año bisiesto. */

  if((ano % 4 == 0 && ano % 100 != 0) || ano % 400 ==0) /* Si se cumplen las condiciones es bisiesto. */
    bisiesto = 1;

  return(bisiesto);
}

int fecha(int dia, int mes, int ano){

  int correcta = 1;                    /* Valor booleano que determinará si la fecha es correcta. */
  int bisiesto;                        /* Valor booleano que determinará si el año es bisiesto. */

  bisiesto = anobisiesto(ano);         /* Comprobamos si el año introducido es bisiesto haciendo uso de la función anobisiesto. */

  if(mes < 8 && mes % 2 == 0 && dia > 30) /* Si el mes es anterior a agosto y este es par y si el día es mayor que 30, la fecha no es válida. */
    correcta = 0;
  else if(mes >= 8 && mes % 2 != 0 && dia > 30) /* Si el mes es agosto o posterior y es impar y si el día es mayor que 30, la fecha no es válida. */
    correcta = 0;
  else if(bisiesto == 0 && mes == 2 && dia > 28) /* Si el año no es bisiesto, y tenemos un día en febrero superior a 29, la fecha no es válida. */
    correcta = 0;
  else if(mes == 2 && mes == 2 && dia > 29) /* Si el año es bisiesto, y tenemos un día en febrero superior a 28, la fecha no es válida. */
    correcta = 0;
```

```
  return(correcta);
}

int main(){

  int dia, mes, ano;
  int correcto;

  do{
    printf("Introduzca el día: ");
    scanf("%d", &dia);
  }while(dia <= 0 || dia > 31);

  do{
    printf("Introduca el mes: ");
    scanf("%d", &mes);
  }while(mes <= 0 || mes > 12);

  printf("Introduzca el año: ");
  scanf("%d", &ano);


  correcto = fecha(dia, mes, ano);

  if(correcto == 0)
    printf("\nLa fecha introducida no es correcta.\n\n");
  else
    printf("\nLa fecha introducida es correcta.\n\n");

  return(0);
}
```

EJERCICIO 11

/* Escribir en C la función int MCM(int a, int b) que devuelve el mínimo común múltiplo de dos números enteros. */

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

/* La función MCM funciona del siguiente modo:

-Introducimos las variables de entrada a y b.

-Abrimos un primer bucle que funcionará mientras no se haya encontrado el M.C.M. o mientras el iterador no alcance

el valor del número menor entre a y b.

-Dentro del bucle abrimos otro bucle que nuevamente, con otro iterador, estará activo mientras no se haya encontrado

el M.C.M. o mientras el iterador no alcance el valor del número menor entre a y b.

-Dentro de este segundo bucle comprobamos los distintos múltiplos del número a con el iterador fijado del primer bucle

con el múltiplo del número b con el iterador del segundo bucle e iremos variando este segundo.

-Una vez se ha recorrido el segundo iterador hasta llegar al número menor, si no se ha encontrado el M.C.M. durante las

iteraciones, salimos del bucle y reinicializamos el bucle para comenzar de nuevo aumentando en una unidad el iterador

del primer bucle. Así hasta que este último alcance el valor del número menor.

-Así, si encontramos el M.C.M. durante el proceso, los bucles se detenienen; o sino se llega hasta el valor mínimo de

las variables a y b de tal forma que el M.C.M. será su producto.

Finalmente, el resultado se almacena en la variable mcm y se devuelve al usuario. */

```
int MCM(int a, int b){

  int i = 0, j = 0;      /* Declaramos los iteradores de los bucles. */
  int comunmultiplo = 0; /* Declaramos una variable booleana que comprobará si hemos encontrado el mínimo común múltiplo. */
  int mcm;               /* En esta variable almacenamos el resultado del M.C.M. que devolverá la función. */

  while(comunmultiplo == 0 && (i <= a || i <= b)){
```

```
    i++;
    j = 0;

    while(comunmultiplo == 0 && (j <= a || j <= b)){

      j++;

      if(i * a == j * b)
        comunmultiplo = 1;

    }

  }

  mcm = j * b;

  return(mcm);
}

int main(){

  int x, y;           /* En las variables x e y almacenamos los números de los que queremos averiguar el M.C.M. */
  int mcm;            /* En la variable mcm almacenaremos el valor de M.C.M. de los números. */

  do{
    printf("Introduzca los números de los que quiere averiguar el M.C.M.: ");
    scanf("%d %d", &x, &y);
  }while(x <= 0 || y <= 0);

  mcm = MCM(x, y);      /* Usamos la función MCM para calcular el M.C.M. de x e y. */

  printf("\nEl M.C.M. de %d y %d es %d.\n\n", x, y, mcm);

  return(0);
```

```
}
```

EJERCICIO 13

/* Hacer una implementación recursiva para la función factorial(x). La definición recursiva de esta función dice que el factorial(0) es 1 y para cualquier otro x el factorial(x) es x*factorial(x - 1). */

```
#include<stdio.h>

/* La función factorial funciona del siguiente modo:
    -Primeramente se le introduce el valor x del que se quiera averiguar el factorial.
    -A continuación asigna como valor inicial al resultado la unidad ya que 0! = 1.
    -Posteriormente, siempre que x > 0 calculará el factorial recursivamente tal y como se indica en el enunciado.
    -Repetirá el bucle hasta que alcance x = 0, momento en el cual finaliza y pasará a devolver el resultado. */

int factorial(int x){

  int resultado = 1;

  if(x > 0)
    resultado = x * factorial(x - 1);

  return(resultado);

}

int main(){

  int x;              /* En la variable x almacenamos el número al que queremos calcular el factorial. */
  int fact;           /* En la variable fact almacenamos el resultado del factorial. */

  printf("Introduzca el número al que quiera calcular el factorial: ");
  scanf("%d", &x);

  fact = factorial(x);   /* Llamamos a la función factorial para calcular el factorial del número. */

  printf("\nEl resultado de %d! es %d.\n\n", x, fact);
```

```
  return(0);
}
```

EJERCICIO 14

/* Hacer una función que reciba un número entero y que devuelva el número de cifras que este número entero tiene.

Hacer un programa que muestre el uso de la función. */

```
#include<stdio.h>


/* La función numeroCifras funciona del siguiente modo:
    -Introducimos el número x del que queremos saber el número de cifras.
    -Declaramos la variable ncifras en la que almacenamos la salida del programa (el número de cifras).
    -Dividimos el entero entre 10 hasta que sea cero.
    -Mediante recursividad vamos sumando cifras hasta que el número es cero.
  Hecho esto, el programa devuelve el resultado final ncfiras al usuario. */


int numeroCifras(int x){

  int ncifras = 1;

  x /= 10;

  if(x != 0)
    ncifras += numeroCifras(x);

  return(ncifras);
}


int main(){

  int x;                /* En la variable x almacenamos el valor que introduciremos a la función. */
  int num_cifras;       /* En esta variable almacenamos la salida de la función. */

  printf("Introduzca un número entero: ");
  scanf("%d", &x);
```

```
  num_cifras = numeroCifras(x); /* Llamamos a la función numeroCifras para calcular el número de cifras de x. */

  if(num_cifras == 1)          /* Si el número de cifras es 1, imprimimos un mensaje en singular. */
    printf("\nEl número %d tiene %d cifra.\n\n", x, num_cifras);
  else
    printf("\nEl número %d tiene %d cifras.\n\n", x, num_cifras);

  return(0);
}
```

EJERCICIO 15

/* Escribir una función que calcule el producto de la primera y de la última cifra de un número entero recibido como argumento. Hacer un programa que muestre el uso de esta función. */

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

/* La función numeroCifras funciona del siguiente modo:
     -Introducimos el número x del que queremos saber el número de cifras.
     -Declaramos la variable ncifras en la que almacenamos la salida del programa (el número de cifras).
     -Dividimos el entero entre 10 hasta que sea cero.
     -Mediante recursividad vamos sumando cifras hasta que el número es cero.
   Hecho esto, el programa devuelve el resultado final ncfiras al usuario. */


int numeroCifras(int x){

  int ncifras = 1;

  x /= 10;

  if(x != 0)
    ncifras += numeroCifras(x);

  return(ncifras);
}

/* La función funciona del siguiente modo:
     -Le introducimos un número entero.
     -Calcula su número de cifras llamando a la función numeroCifras.
     -Obtenemos la cifra inicial y la cifra final.
     -Multiplicamos ambas cifras y las guardamos en la variable de salida.
   Hecho esto, el programa devuelve el resultado necesitado. */
```

```
int producto_primero_ultimo(int x){

  int ncifras, resultado;
  int x0, xf;                    /* En estas variables almacenaremos la cifra inicial y la final respectivamente. */

  ncifras = numeroCifras(x);        /* Calculamos el número de cifras que tiene nuestro número. */

  x0 = x / pow(10, ncifras - 1);     /* Al dividir entre 10^(ncifras - 1) obtenemos la cifra inicial. */
  xf = x - (x / 10) * 10;           /* Como trabajamos con enteros, al dividir entre 10 perdemos la última cifra y al
                                recuperarla multiplicando por 10 obtenemos el mismo número pero con la unidad nula. */
  resultado = x0 * xf;

  return(resultado);
}

int main(){

  int x;                        /* Este será el número que introduciremos de entrada a la función. */
  int resultado;                  /* Aquí almacenaremos la salida de la función. */

  printf("Introduzca un número entero: ");
  scanf("%d", &x);

  if(x < 0)
    x *= -1;                    /* En caso de que el número introducido sea negativo, lo volvemos positivo para usar la
función. */

  resultado = producto_primero_ultimo(x);

  printf("\nEl resultado de multiplicar la primera y la última cifra de %d es %d.\n\n", x, resultado);

  return(0);
}
```

EJERCICIO 16

/* Realizar un pequeño programa en C que cuenta el número de caracteres, palabras y líneas de un fichero de texto (que lee de stdin haciendo uso de redirección de entrada). */

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

/* La función funciona del siguiente modo:
    -Tiene como parámetros por referencia el número de caracteres, de palabras y de lineas que comienzan en 0.
    -Comienza un bucle que se detendrá cuando encontremos el caracter final de texto en ASCII.
    -Mientras tanto, cada vez que se lea un caracter aumentamos el contador en una unidad.
    -A continuación, si el caracter leido es un espacio aumentamos el número de palabras en uno.
    -La siguiente comprobación es que, si encontramos un salto de línea aumentamos el contador de lineas en uno.
  Una vez finaliza el bucle, las variables por referencia han almacenado la lectura correspondiente. */

void contadorcaracteres(int *ncaracteres, int *npalabras, int *nlineas){

  char caracter = 0;

  while(caracter != 3){

    scanf("%c", &caracter);
    *ncaracteres++;

    if(caracter == 32 && *ncaracteres > 0)
      *npalabras++;
    if(caracter == 10 && *ncaracteres > 0)
      *nlineas++;

  }

}

int main(){
```

```
  int ncaracteres = 0, npalabras = 0, nlineas = 0;

  contadorcaracteres(&ncaracteres, &npalabras, &nlineas);

  printf("\nEl archivo de texto contiene %d caracteres, %d palabras y %d líneas.\n\n", ncaracteres, npalabras,
nlineas);

  return(0);
}
```

EJERCICIO 17

/* La fuerza de atracción entre dos masas, m1 y m2 separadas por una distancia d, está dada por la fórmula:

F = G * m1 * m2 / d ^ 2. Donde G es la constante de gravitación universal G = 6.67*10^-11. Escriba un programa que lea

la masa de dos cuerpos y la distancia entre ellos y a continuación obtenga la fuerza gravitacional entre ellas. */

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

float fuerzaGravitatoria(float m1, float m2, float d){

  float fuerza;
  const float G = 6.67 * pow(10, -11);

  fuerza = G * (m1 * m2) / pow(d, 2);

  return(fuerza);
}

int main(){

  float m1, m2, d, fuerza;

  printf("Introduzca la masa del primer cuerpo (en kg): ");
  scanf("%f", &m1);
  printf("Introduzca la masa del segundo cuerpo (en kg): ");
  scanf("%f", &m2);
  printf("Introduzca la distancia entre ellos (en m): ");
  scanf("%f", &d);

  fuerza = fuerzaGravitatoria(m1, m2, d);

  printf("\nAmbos cuerpos se realizan una fuerza entre sí de %f N.\n\n", fuerza);
```

```
  return(0);
}
```

EJERCICIO 18

/* Escribe un programa que lea tres números reales que representan una hora en formato horas, minutos y segundos, le sume un segundo y saque el resultado en formato HH:MM:SS. */

```
#include<stdio.h>

/* La función sexagesimal funciona del siguiente modo:
    -Primeramente tiene como argumentos de entrada el tiempo en segundos y, por referencia, las tres salidas de horas, minutos
     y segundos.
    -Dicho esto, comienza calculando el número de minutos como el cociente del tiempo entre 60. A su vez, el resto nos devuelve
     el número final de segundos.
    -Posteriormente aplicamos el mismo razonamiento, pero para el paso de minutos a segundos.
  Una vez se realizan las operaciones, se obtiene la salida por paso por referencia. */

void sexagesimal(int tiempo, int *horas, int *minutos, int *segundos){

  *minutos = tiempo / 60;
  *segundos = tiempo % 60;
  *horas = *minutos / 60;
  *minutos %= 60;

}

/* La función tiempoSuma1 funciona del siguiente modo:
    -Como argumentos de entrada tiene el tiempo en horas, minutos, segundos y las tres salidas ss, mm y hh por referencia.
    -Suma una unidad a los segundos, y ahora comprueba que el contador no pueda marcar 60 al sumar 1.
    -Si ocurre que obtenemos un 60, llamamos a la función sexagesimal para que reorganice el tiempo nuevamente.
    -Sino guardamos directamente en la salida los tiempos correspondientes. */

void tiempoSuma1(int horas, int minutos, int segundos, int *hh, int *mm, int *ss){

  segundos++;
```

```
  if(segundos == 60 || (segundos == 60 && minutos == 59)){

    segundos += horas * 3600 + minutos * 60;
    sexagesimal(segundos, &*hh, &*mm, &*ss);

  }
  else{

    *hh = horas;
    *mm = minutos;
    *ss = segundos;

  }

}

int main(){

  int horas, minutos, segundos;    /* Entradas de la función. */
  int hh = 0, mm = 0, ss = 0;      /* Salidas de la función. */

  printf("Introduzca el número de horas: ");
  scanf("%d", &horas);

  do{

    printf("Introduzca el número de minutos: ");
    scanf("%d", &minutos);

  }while(minutos < 0 || minutos >= 60);

  do{

    printf("Introduzca el número de segundos: ");
```

```
    scanf("%d", &segundos);

  }while(segundos < 0 || segundos >= 60);

  tiempoSuma1(horas, minutos, segundos, &hh, &mm, &ss);

  printf("\nTenemos ahora un tiempo de %d:%d:%d.\n\n", hh, mm, ss);

  return(0);
}
```

EJERCICIO 19

/* Escribe un programa que lea un número natural que representa un número de segundos y lo descomponga

en horas, minutos y segundos. */

```
#include<stdio.h>

/* La función sexagesimal funciona del siguiente modo:
    -Primeramente tiene como argumentos de entrada el tiempo en segundos y, por referencia, las tres salidas de horas, minutos
     y segundos.
    -Dicho esto, comienza calculando el número de minutos como el cociente del tiempo entre 60. A su vez, el resto nos devuelve
     el número final de segundos.
    -Posteriormente aplicamos el mismo razonamiento, pero para el paso de minutos a segundos.
  Una vez se realizan las operaciones, se obtiene la salida por paso por referencia. */

void sexagesimal(int tiempo, int *horas, int *minutos, int *segundos){

  *minutos = tiempo / 60;
  *segundos = tiempo % 60;
  *horas = *minutos / 60;
  *minutos %= 60;

}

int main(){

  int tiempo;
  int horas = 0, minutos = 0, segundos = 0;

  printf("Introduzca el número de segundos: ");
  scanf("%d", &tiempo);

  sexagesimal(tiempo, &horas, &minutos, &segundos);
```

```
    printf("\nTenemos un tiempo %d:%d:%d.\n\n", horas, minutos, segundos);

    return(0);
}
```

EJERCICIO 20

/* Sabiendo que las distancias de los planetas del sistema solar en millones de kilometros al sol son:

1 Mercurio 59

2 Venus 108

3 Tierra 150

4 Marte 228

5 Júpiter 750

6 Saturno 1431

7 Urano 2877

8 Neptuno 4509

9 Plutón 5916

Hacer un programa que muestre un menú de planetas para que el usuario seleccione un planeta y el programa nos diga

su distancia al sol. (Utilizar la sentencia switch). */

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

/* [IDENTIFICADOR: menu(int *opcion)]
  [TIPO DEVUELTO: n/a]
  [COMETIDO: Mostrar el menú de inicio de programa con la elección correspondiente.]
  [ENTRADAS: opcion(por referencia)]
  [SALIDAS: n/a] */

void menu(){

  printf("Este programa nos dice la distancia al Sol del planeta elegido. Puede seleccionar los siguientes:\n\n"
      "\t1. Mercurio.\n"
      "\t2. Venus. \n"
      "\t3. Tierra. \n"
      "\t4. Marte. \n"
      "\t5. Jupiter. \n"
```

```
        "\t6. Saturno. \n"
        "\t7. Urano. \n"
        "\t8. Nepturno. \n"
        "\t9. Plutón. \n");

}

/* [IDENTIFICADOR: distanciaMercurio_Sol()]
   [TIPO DEVUELTO: n/a]
   [COMETIDO: Mostrar la distancia de Mercurio al Sol.]
   [ENTRADAS: n/a]
   [SALIDAS: n/a] */

void distanciaMercurio_Sol(){

   system("cls");
   printf("\nLa distancia de Mercurio al Sol es de 59 Gm.\n\n");

}

/* [IDENTIFICADOR: distanciaVenus_Sol()]
   [TIPO DEVUELTO: n/a]
   [COMETIDO: Mostrar la distancia de Venus al Sol.]
   [ENTRADAS: n/a]
   [SALIDAS: n/a] */

void distanciaVenus_Sol(){

   system("cls");
   printf("\nLa distancia de Venus al Sol es de 108 Gm.\n\n");

}

/* [IDENTIFICADOR: distanciaTierra_Sol()]
```

```
   [TIPO DEVUELTO: n/a]
   [COMETIDO: Mostrar la distancia de Tierra al Sol.]
   [ENTRADAS: n/a]
   [SALIDAS: n/a] */

void distanciaTierra_Sol(){

   system("cls");
   printf("\nLa distancia de Tierra al Sol es de 150 Gm.\n\n");

}

/* [IDENTIFICADOR: distanciaMarte_Sol()]
   [TIPO DEVUELTO: n/a]
   [COMETIDO: Mostrar la distancia de Marte al Sol.]
   [ENTRADAS: n/a]
   [SALIDAS: n/a] */

void distanciaMarte_Sol(){

   system("cls");
   printf("\nLa distancia de Marte al Sol es de 228 Gm.\n\n");

}

/* [IDENTIFICADOR: distanciaJupiter_Sol()]
   [TIPO DEVUELTO: n/a]
   [COMETIDO: Mostrar la distancia de Jupiter al Sol.]
   [ENTRADAS: n/a]
   [SALIDAS: n/a] */

void distanciaJupiter_Sol(){

   system("cls");
```

```
  printf("\nLa distancia de Mercurio al Sol es de 750 Gm.\n\n");

}

/* [IDENTIFICADOR: distanciaSaturno_Sol()]
  [TIPO DEVUELTO: n/a]
  [COMETIDO: Mostrar la distancia de Saturno al Sol.]
  [ENTRADAS: n/a]
  [SALIDAS: n/a] */

void distanciaSaturno_Sol(){

  system("cls");
  printf("\nLa distancia de Saturno al Sol es de 1431 Gm.\n\n");

}

/* [IDENTIFICADOR: distanciaUrano_Sol()]
  [TIPO DEVUELTO: n/a]
  [COMETIDO: Mostrar la distancia de Urano al Sol.]
  [ENTRADAS: n/a]
  [SALIDAS: n/a] */

void distanciaUrano_Sol(){

  system("cls");
  printf("\nLa distancia de Urano al Sol es de 2877 Gm.\n\n");

}

/* [IDENTIFICADOR: distanciaNeptuno_Sol()]
  [TIPO DEVUELTO: n/a]
  [COMETIDO: Mostrar la distancia de Neptuno al Sol.]
  [ENTRADAS: n/a]
```

```
   [SALIDAS: n/a] */

void distanciaNeptuno_Sol(){

   system("cls");
   printf("\nLa distancia de Nepturno al Sol es de 4059 Gm.\n\n");

}

/* [IDENTIFICADOR: distanciaPluton_Sol()]
   [TIPO DEVUELTO: n/a]
   [COMETIDO: Mostrar la distancia de Plutón al Sol.]
   [ENTRADAS: n/a]
   [SALIDAS: n/a] */

void distanciaPluton_Sol(){

   system("cls");
   printf("\nLa distancia de Plutón al Sol es de 5916 Gm.\n\n");

}

int main(){

   int opcion;

   menu();

   do{
      printf("\nIntroduzca el número del planeta: ");
      scanf("%d", &opcion);
   }while(opcion < 1 || opcion > 9);

   switch(opcion){
```

```
    case 1: distanciaMercurio_Sol();   break;
    case 2: distanciaVenus_Sol();      break;
    case 3: distanciaTierra_Sol();     break;
    case 4: distanciaMarte_Sol();      break;
    case 5: distanciaJupiter_Sol();    break;
    case 6: distanciaSaturno_Sol();    break;
    case 7: distanciaUrano_Sol();      break;
    case 8: distanciaNeptuno_Sol();    break;
    case 9: distanciaPluton_Sol();     break;

    default: printf("\nNo ha introducido una opción correcta, lo siento.\n\n"); break;

  }

  return(0);
}
```

EJERCICIO 21

/* Hacer una función que reciba cuatro números enteros que son el numerador y denominador de dos fracciones, sume estas dos fracciones y devuelva dos números enteros que será el numerador y el denominador de la fracción suma. Hacer también un programa principal para ilustrar el uso de la función. */

```
#include<stdio.h>

/* IDENTIFICADOR: int MCD;
  TIPO DEVUELTO: int;
  COMETIDO: Hallar el M.C.D. de dos números;
  ENTRADAS: int a, int b;
  SALIDAS: int mcd; */

int MCD(int a, int b){

  int mcd, i = 1;

  while(i <= a && i <= b){

    if(a % i == 0 && b % i == 0)
      mcd = i;

    i++;

  }

  return(mcd);
}

/* IDENTIFICADOR: int MCM;
  TIPO DEVUELTO: int;
  COMETIDO: Hallar el M.C.M. de dos números;
  ENTRADAS: int a, int b;
  SALIDAS: int mcm; */
```

```
int MCM(int a, int b){

  int i = 0, j = 0;
  int comunmultiplo = 0;
  int mcm;

  while(comunmultiplo == 0 && (i <= a || i <= b)){

    i++;
    j = 0;

    while(comunmultiplo == 0 && (j <= a || j <= b)){

      j++;

      if(i * a == j * b)
        comunmultiplo = 1;

    }

  }

  mcm = j * b;

  return(mcm);
}

/* IDENTIFICADOR: void sumaFracciones;
  TIPO DEVUELTO: N/A;
  COMETIDO: Sumar dos fracciones a partir de sus numeradores y denominadores.;
  ENTRADAS: int num1, int den1, int num2, int den2;
  SALIDAS: (por referencia) int *numSuma, int *denSuma; */
```

```
void sumaFracciones(int num1, int den1, int num2, int den2, int *numSuma, int *denSuma){

  int num, den;
  int simplificador;

  den = MCM(den1, den2);

  num = num1 * den / den1 + num2 * den / den2;

  simplificador = MCD(den, num);

  *numSuma = num / simplificador;
  *denSuma = den / simplificador;

}

int main(){

  int num1, den1, num2, den2;
  int numSuma, denSuma;

  do{
    printf("Introduzca el numerador y denominador de la primera fracción: ");
    scanf("%d %d", &num1, &den1);
  }while(den1 == 0);

  do{
    printf("Introduzca el numerador y denominador de la segunda fracción: ");
    scanf("%d %d", &num2, &den2);
  }while(den2 == 0);

  sumaFracciones(num1, den1, num2, den2, &numSuma, &denSuma);

  if(denSuma == 1)
```

```
    printf("\nEl resultado de %d/%d + %d/%d es %d.\n\n", num1, den1, num2, den2, numSuma);
  else
    printf("\nEl resultado de %d/%d + %d/%d es %d/%d.\n\n", num1, den1, num2, den2, numSuma, denSuma);

  return(0);
}
```

EJERCICIO 22

/* Realizar un programa que pida números enteros hasta que se introduzca el cero. El programa debe sumar todos los números pares

y los números impares para al finalizar mostrar estas sumas. */

```
#include<stdio.h>

/* IDENTIFICADOR: sumador;
  TIPO DEVUELTO: N/A;
  COMETIDO: Acumular la entrada en un contador de suma par o impar.;
  ENTRADAS: int entrada;
  SALIDAS: (por referencia) int *sumaPares, int *sumaImpares; */

void sumador(int entrada, int *sumaPares, int *sumaImpares){

  if(entrada % 2 == 0)
    *sumaPares += entrada;
  else
    *sumaImpares += entrada;

}

int main(){

  int sumaPares = 0, sumaImpares = 0, entrada;

  while(entrada > 0){

    do{
      printf("\nIntroduzca un número positivo (para finalizar debe introducir un cero): ");
      scanf("%d", &entrada);
    }while(entrada < 0);
```

```
    sumador(entrada, &sumaPares, &sumaImpares);

  }

  printf("\nEl resultado de la suma de los pares es %d.\n", sumaPares);
  printf("\nEl resultado de la suma de los impares es %d.\n\n", sumaImpares);

  return(0);
}
```

EJERCICIO 23

/* Hacer una función para calcular el cociente y el resto de una divisón entera. La función recibirá los dos números dividendo y divisor y devolverá los dos números cociente y resto. No se puede utilizar la función módulo y la divisón entera hay que hacerlo por restas sucesivas. Hacer un main que ilustre como se usa la función. */

```
#include<stdio.h>

/* IDENTIFICADOR: division;
  TIPO DEVUELTO: void;
  COMETIDO: Realiza de forma manual una división.;
  ENTRADAS: El dividendo y el divisor de la división.;
  SALIDAS: El cociente y el resto (por referencia).; */

void division(int dividendo, int divisor, int *cociente, int *resto){

  int i = dividendo;                    /* Esta variable nos sirve como controlador de la recursividad. */

  if(i >= divisor){

    *cociente = *cociente + 1;          /* El cociente aumenta en uno ya que "cabe" un divisor más en el dividendo. */
    i -= divisor;                       /* Nuestro controlador actuará como el nuevo dividendo. */

    division(i, divisor, cociente, resto);      /* Llamamos nuevamente a la función con los nuevos datos. */

  }
  else
    *resto = i;                         /* El resto será el valor final del dividendo durante la recursividad. */

}

int main(){

  int dividendo, divisor, cociente = 0, resto = 0;
```

```
  do{
    printf("Introduzca el dividendo: ");
    scanf("%d", &dividendo);
  }while(dividendo < 0);                    /* Controlamos que el dividendo nunca sea negativo. */

  do{
    printf("Introduzca el divisor: ");
    scanf("%d", &divisor);
  }while(divisor < 0);                /* Controlamos que el divisor nunca sea mayor que el dividendo. */

  division(dividendo, divisor, &cociente, &resto);

  printf("\n\nEl resultado de dividir %d con %d nos devuelve: \n"
       "\tCociente = %d.\n"
       "\tResto = %d.\n\n", dividendo, divisor, cociente, resto);

  return(0);
}
```

EJERCICIO 24

/* Realizar un programa que permita calcular el área de un cuadrado, un círculo o un triángulo equilátero.

El programa ha de presentar un menú pedir los datos necesarios y hacer el cálculo. */

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

#define M_PI 3.14159265358979323846

/* IDENTIFICADOR: menu;
  TIPO DEVUELTO: N/A;
  COMETIDO: Mostrar el menu de uso del programa;
  ENTRADAS: N/A;
  SALIDAS: opción (por referencia); */

int menu(){

  int opcion;

  printf("Elija una de las tres opciones: \n\n"
       "\t1. ÁREA DE UN CUADRADO.\n"
       "\t2. ÁREA DE UN CÍRCULO.\n"
       "\t3. ÁREA DE UN TRIÁNGULO EQUILATERO.\n");
  do{
    printf("\nIntroduzca la opción: ");
    scanf("%d", &opcion);
  }while(opcion < 1 || opcion > 3);

  return(opcion);

}

/* IDENTIFICADOR: areaCuadrado;
```

```
  TIPO DEVUELTO: float;
  COMETIDO: Calcular el área de un cuadrado;
  ENTRADAS: lado del cuadrado;
  SALIDAS: área; */

float areaCuadrado(float lado){

  float area;

  area = pow(lado, 2);

  return(area);
}

/* IDENTIFICADOR: areaCirculo;
  TIPO DEVUELTO: float;
  COMETIDO: Calcular el área de un círculo;
  ENTRADAS: radio del círculo;
  SALIDAS: área; */

float areaCirculo(float radio){

  float area;

  area = M_PI * pow(radio, 2);

  return(area);
}

/* IDENTIFICADOR: areaTriangulo;
  TIPO DEVUELTO: float;
  COMETIDO: Clcular el área de un triángulo equilatero;
  ENTRADAS: lado del triángulo equilatero;
  SALIDAS: área; */
```

```
float areaTriangulo(float lado){

  float area;

  area = (lado * lado * sin(M_PI / 3)) / 2;

  return(area);
}

int main(){

  int opcion;
  float ladoCuadrado, radio, ladoTriangulo;
  float area;

  opcion = menu();

  switch(opcion){

    case 1:
      system("cls");
      do{
        printf("Introduzca el lado del cuadrado: ");
        scanf("%f", &ladoCuadrado);
      }while(ladoCuadrado <= 0);
      area = areaCuadrado(ladoCuadrado);
      break;

    case 2:
      system("cls");
      do{
        printf("Introduzca el radio del círculo: ");
        scanf("%f", &radio);
```

```
        }while(radio <= 0);
        area = areaCirculo(radio);
        break;

      case 3:
        system("cls");
        do{
          printf("Introduzca el lado del triángulo equilatero: ");
          scanf("%f", &ladoTriangulo);
        }while(ladoTriangulo <= 0);
        area = areaTriangulo(ladoTriangulo);
        break;

      case 4:
        system("cls");
        printf("\nNo ha introducido una opción valida.\n\n");

    }

    printf("\nEl área es de %f metros cuadrados.\n\n", area);

    return(0);
}
```

EJERCICIO 25

/* Hacer un programa que calcule el área y el perímetro de un rectángulo dada la base y la altura del rectángulo. */

```
#include<stdio.h>

/* IDENTIFICADOR: rectangulo;
  TIPO DEVUELTO: N/A;
  COMETIDO: Calcular el área y el perímetro de un rectángulo.;
  ENTRADAS: Los dos lados del rectángulo.;
  SALIDAS: El área y el perímetro (por referencia).; */

void rectangulo(float lado1, float lado2, float *area, float *perimetro){

  *area = lado1 * lado2;

  *perimetro = 2 * lado1 + 2 * lado2;

}

int main(){

  float lado1, lado2;
  float area, perimetro;

  do{
    printf("Introduzca los dos lados del rectángulo: ");
    scanf("%f %f", &lado1, &lado2);
  }while(lado1 <= 0 || lado2 <= 0);

  rectangulo(lado1, lado2, &area, &perimetro);

  printf("\nEl área del rectángulo es %f metros cuadrados.\n", area);
  printf("\nEl perímetro del rectangulo es %f metros.\n\n", perimetro);
```


```
  return(0);
}
```

EJERCICIO 26

/* Crear un programa que encuentre el máxima común divisor de dos números usando el algoritmo de Euclides:
Dado dos números enteros positivos m y n, tal que m > n, para encontrar su máximo común divisor, es decir, el mayor entero positivo que divide a ambos;
  * Dividir m por n para obtener el resto r(0 = r < n).
  * Si r = 0, el MCD es n.
  * Si no, el máximo común divisor es MCD(n, r). */

```
#include<stdio.h>

/* IDENTIFICADOR: MCD;
  TIPO DEVUELTO: int;
  COMETIDO: Calcular el M.C.D. de dos números.;
  ENTRADAS: Los números m y n (m > n) a los que calcular el M.C.D.;
  SALIDAS: El M.C.D.; */

int MCD(int m, int n){

  int mcd, r;

  r = m % n;

  if(r == 0)
    mcd = n;
  else
    mcd = MCD(n, r);

  return(mcd);
}

int main(){

  int a, b, mcd;
```

```
  do{
    printf("Introduzca los dos números a los que quiere calcular el M.C.D.: ");
    scanf("%d %d", &a, &b);
  }while(a <= 0 || b <= 0);

  if(a > b)
    mcd = MCD(a, b);
  else
    mcd = MCD(b, a);

  printf("\nEl M.C.D. de %d y %d es %d.\n\n", a, b, mcd);

  return(0);
}
```

EJERCICIO 27

/* Escribir un programa (con las funciones que se estime pertinentes) que escriba todos los números perfectos

menores que 1000. Se dice que un número es perfecto cuando es igual a la suma de todos sus divisores (incluyendo el

1 y excluyendo el propio número). */

```
#include<stdio.h>

/* IDENTIFICADOR: sumaDivisores;
  TIPO DEVUELTO: int;
  COMETIDO: Calcular la suma de los divisores.;
  ENTRADAS: El número n a calcular la suma de divisores y el iterador i (debe comenzar en 1).;
  SALIDAS: La suma de divisores del número n.; */

int sumaDivisores(int n, int i){

  int suma = 0, j;

  j = i;

  while(j < n && j != 1 && n % j != 0)
    j++;

  if(j == 1){

    j++;
    suma = 1 + sumaDivisores(n, j);

  }
  else if(j < n){

    suma = j + sumaDivisores(n, j + 1);

  }
```

```
  return(suma);
}

int main(){

  int n, i;

  do{
    printf("Hasta que número quiere buscar números perfectos: ");
    scanf("%d", &n);
  }while(n <= 0);

  for(i = 2; i <= n; i++){

    if(sumaDivisores(i, 1) == i)
      printf("\nEl número %d es un número perfecto.\n\n", i);

  }

  return(0);
}
```

EJERCICIO 28

/* El método de Newton para hallar la raíz cuadrada de un número real positivo X tiene gran interés ya que utiliza sólo sumas, multiplicaciones y divisiones (algunas de ellas divisiones por 2 que son inmediatas para la máquina). El método consiste en lo siguiente: para calcular la raíz cuadrada de un número X tal que el cuadrado de la solución difiera de X menos de un cierto error E, comenzamos con la aproximación a = X /2. Si |(a * a) - X| < E paramos los cálculos y el resultado es a. Si no, reemplazamos a con la siguiente aproximación definida por (a + X / a) / 2. Entonces comprobamos si esta aproximación es lo suficientemente buena de la misma manera que antes lo hicimos. Si lo es, el cálculo finaliza y si no prosigue iterativamente, estando garantizado que la aproximación converge. Escribe una función que implemente este método dados X y el error E aceptable, y un programa que la utilice. */

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

/* IDENTIFICADOR: raizCuadrada;
  TIPO DEVUELTO: float;
  COMETIDO: Calcular una raiz cuadrada con un error dado mediante el método de Newton.
  ENTRADAS: El número y el error.
  SALIDAS: La raíz. */

float raizCuadrada(float x, float error){

  float a, error_a;

  a = x / 2;
  error_a = fabs(pow(a, 2) - x);

  while(error_a > error){

    a = (a + x / a) / 2;
    error_a = fabs(pow(a, 2) - x);

  }
```

```
  return(a);
}

int main(){

  float x, error, raiz;

  do{
    printf("Introduzca el número al que quiere calcular su raíz cuadrada: ");
    scanf("%f", &x);
  }while(x < 0);

  do{
    printf("Introduzca el máximo error admitido: ");
    scanf("%f", &error);
  }while(error <= 0);

  raiz = raizCuadrada(x, error);

  printf("\nLa raíz cuadrada de %f es %f±%f.\n\n", x, raiz, error);

  return(0);
}
```