Pontificia Universidad Católica Madre y Maestra
Facultad de Ciencias de La Ingeniería
Departamento de Ingeniería de Sistemas y Computación

Ejercicios/Problemas de Programación

Rev. febrero 2005

Este folleto es una compilación de ejercicios con miras a ayudar a los estudiantes de ISC-201 ("Algoritmos Fundamentales") a desarrollar destrezas en la elaboración de algoritmos e implementarlos en un lenguaje de computación. Puesto que dicha asignatura es impartida usando Lenguaje C, en la redacción de algunos problemas se hace referencia a algunas herramientas de dicho lenguaje. Sin embargo, los problemas se pueden resolver usando cualquier lenguaje de programación de aplicación general.

La división corresponde a los capítulos del programa de estudio. Los ejercicios pretenden estar enumerados en orden de menor a mayor complejidad.

El resultado de cada ejercicio es un programa de computadora, sin embargo, **es sumamente importante que el/la estudiante clarifique el algoritmo pensado para resolver el problema antes de codificar en un lenguaje específico.** En los casos complejos, es muy recomendado que el/los algoritmo(s) sea(n) escrito(s).

Rolando J. Batista
rbatista@pucmmsti.edu.do
Alejandro J. Liz R.
aliz@pucmmsti.edu.do

PARTE I. Programación Básica

⌖ Tipos de variables y expresiones.

Dados los valores para las siguientes variables, evalúe las expresiones que aparecen más abajo independientemente una de otra. Compruebe el resultado en el computador.
* Nota: Los valores ASCII de 'A' y '0' son 65 y 48 respectivamente.

int a = 8, b = 3, c = 12, d = 1, e = 5;
float x = 3.5, y = 2.1, z = 5;
char m = 'C', n = '2';

1. `d % b`
2. `++b * a - 9 * e`
3. `z / b + 1`
4. `e / b + 1`
5. `(int)z / (b * 2)`
6. `z / a * 2`
7. `c / (int)x % e`
8. `'0' > e || --d`
9. `(float)m / y + 3.52E3`
10. `c %= b * --e`
11. `(++e - 5) && (d--||c % b)`
12. `e = --n + 3`
13. `(n >= 2) ? n - '0' : n`
14. `b *= 0XB8 % a`
15. `((c & 25) | 4) >> b`
16. `(++e << 3) < c * 12 / n * e / 2`
17. `(c--%(int)x || (a-e)%e )? (c%b++ || --d) : (b*e+n%4)/(int)x`
18. Los operadores >> y << se pueden usar para dividir y multiplicar por dos en cifras enteras. ¿Por qué?
19. En una máquina de 16 bits, si se tiene la variable `a` declarada de la siguiente manera:
    `unsigned int a = 11;`
    ¿Cuál sería el valor de `~a`? ¿Qué diferencia habría si se ejecutan las mismas instrucciones en una máquina de 32 bits?
20. Investigue sobre el uso del operador `sizeof`.
21. ¿Cómo podríamos comprobar que un número es par o impar usando el operador &?

⌖ Ciclos, condiciones y funciones

1. Escriba un programa al cual se le digite un número e imprima por pantalla su nombre tantas veces como lo indique el número digitado.
2. Escriba un programa al cual se le digite dos números representando una longitud en pies y pulgadas e imprima dicha distancia expresada en metros.
3. Haga un programa al cual se le digite un número entero y despliegue por pantalla la suma de los números comprendidos entre 0 y el número digitado.
4. Haga un programa al cual se le digite una longitud expresada en pulgadas e imprima la misma longitud en yardas, pies y pulgadas. Por ejemplo, una longitud de 65 pulgadas sería expresada como 1 yarda, 2 pies y 5 pulgadas.
5. Escriba un programa que despliegue por pantalla los números positivos impares menores que 20 en orden ascendente.
6. Repita el ejercicio anterior, pero imprimiendo los números en orden descendente.
7. Haga una función llamada `maxval(n,m)` que retorne el mayor valor entre `n` y `m`.
8. Haga un programa que despliegue una tabla de 6 columnas con las equivalencias ASCII

9. Se llama *media armónica* de dos números al resultado obtenido al calcular los inversos de los números, promediarlos y calcular el inverso del resultado. Escriba una función que acepte dos argumentos `double` y devuelva la media armónica de los números.
10. Escriba un programa que imprima una tabla de conversión de grados Farenheit a Celcius.. La fórmula de conversión es como sigue: 1.8 °C= °F - 32
11. Elabore un programa que pida un número entero por pantalla e imprima todos los números que lo dividen exactamente. El programa debe controlar que los valores digitados sean positivos. En caso de haber digitado un negativo, debe dar un mensaje de error.
12. Escriba un programa que lea dos números enteros positivos por pantalla y los multiplique por sumas sucesivas.
13. Repita el ejercicio anterior de forma que el programa funcione tanto para números positivos como para números negativos.
14. El banco le solicita un programa para cálculo en las libretas de ahorro. Si usted ingresa una cantidad en una libreta de ahorros, su capital irá incrementando a medidas que gana intereses mensuales. Haga un programa al cual se le digite el monto a ahorrar, la tasa de interés, los meses de ahorro y que imprima el monto al final del ahorro.
15. *Los conejos de Fibonacci*. El matemático Leonardo Fibonacci expuso el siguiente problema, haga un programa para solucionarlo. Suponga que una pareja de conejos tiene una par de crías cada mes y a la vez las crías se hacen fértiles al cabo de un mes. Si comenzamos con una pareja fértil y no muere, ¿Cuántos pares de conejos se tendrían al cabo de un año?
16. *El inventor del ajedrez*. De acuerdo a una leyenda, el rey estuvo tan complacido con el invento del juego del ajedrez que llamó al autor a su corte para que le indicara qué recompensa quería recibir por su creatividad. El inventor le dijo: "Todo lo que pido es un grano de trigo por la primera cuadrícula de mi tablero, dos granos por la segunda casilla, cuatro granos por la tercera y así sucesivamente hasta llegar a las 64 casillas, siempre doblando el número de granos de trigo". Haga un programa que le ayude al rey a tomar esta decisión. Haga una estimación del peso de un grano de trigo y "pese" la cantidad resultante. ¿Qué tamaño de palabra necesitaría tener la máquina en que se pueda correr dicho programa?
17. Escriba un programa al cual se le digiten diez valores y al final imprima por pantalla la suma de los valores, la suma de los cuadrados, el promedio, el máximo y el mínimo.
18. Escriba un programa que divida números enteros positivos por restas sucesivas.
19. Escriba un programa que solicite por pantalla el número de horas trabajadas en una semana y el salario por hora de un empleado. A continuación imprima el salario bruto, las retenciones y el salario neto. Suponga la siguiente información:
    a. Horas extras (por encima de 40), se pagan a razón de 1.5 * salario por hora.
    b. Retenciones: 10% de los primeros RD$1,100; 15% de los 1,500 siguientes y 25% del resto.
20. Haga una función float `eqcuad(a,b,c)` la cual retorne la solución positiva de una ecuación cuadrática cuyos parámetros con `a`,`b` y `c` respectivamente.
21. Haga una fución `tipotriang(a,b,c)` a la cual se le pase como parámetros las longitudes de los lados y retorne 1, 2 ó 3 según el triángulo sea equilátero, isósceles o escaleno.
22. El valor π está determinado por una serie como sigue:
    $\pi = 4(1- 1/3 + 1/5 - 1/7 + 1/9 ...)$
    Haga una función `pi(int n)` que retorne el valor de π evaluando `n` términos de la serie. Imprima por pantalla la evaluación con n = 5 .. 15.
23. La Ofina Nacional de Planificación le solicita un programa que haga una tabla de estimación de la población del país para los próximos diez años. El programa debe solicitar la población actual, las tasas de natalidad y de mortalidad esperadas Las tasas se expresan como porcentaje de la población. Por ejemplo, si la tasa de natalidad es de un 1.5%, entonces el número de personas nacidas el año n se espera que sea el 1.5% de la población del año n-1.
24. Escriba un programa en el cual se le digitará un valor de punto flotante que será considerado como el radio de un círculo. El programa imprimirá por pantalla la longitud de la circunferencia y el área del círculo. Para ello, realice una función `calcarea(r)`, y la función `calccirc(r)` que retornen el área y la circunferencia respectivamente. Use la función `pi(n)` resultante del ejercicio anterior.
25. Escriba una función `potencia(b,e)` que realice la operación b^e usando multiplicaciones sucesivas.
26. Usando las funciones `potencia()` y `pi()`, haga una función `vol(r)` que retorne el volumen de una esfera de radio `r`. La fórmula del volumen de la esfera es
    $V = (4/3)\ \pi r^3$
27. Su médico le ha indicado una dieta. Durante las próximas 4 semanas usted estará digiriendo menos comida de la que requeriría para permanecer en el peso actual. Por cada 3500 calorías que se dejan de consumir, se pierde una libra de peso. En la primera semana, usted comerá 1200 menos de las calorías diarias que necesita, la segunda semana 1000 menos, la tercera 850 y la cuarta 600. Escriba un programa que imprima esta información en una manera conveniente y que calcule las libras perdidas en la dieta.
28. Lamentablemente, el año siguiente a la dieta que el médico le indicó en el punto anterior, usted retomó sus libras de más. Sin embargo, desea rebajar nuevamente para ir de vacaciones a La Romana dentro de cuatro semanas, para lo cual ya compró su traje de baño. Para usar dicho traje de baño, no puede tener más de 140 libras. Escriba un programa al cual le digite su peso actual y el programa indique las calorías diarias que debe dejar de consumir durante el próximo mes. Igual que en la dieta del problema anterior, usted reducirá las calorías diarias consumidas por semana. En la primera semana debe perder una tercera parte del peso a rebajar. Durante la segunda y tercera semanas una cuarta parte en cada una y en la semana final perderá la sexta parte restante. Felices vacaciones.

29. Los *números de Pitágoras* pueden ser descritos de la siguiente manera: a2 + $b^2 = c^2$, donde *a*, *b* y *c* son enteros y b y c son consecutivos. a y b son catetos y c es la hipotenusa. Escriba un programa para encontrar cinco ternas de *números de Pitágoras*. Por ejemplo:
$3^2 + 4^2 = 5^2$, $5^2 + 12^2 = 13^2$
30. Usando ciclos, escriba un programa que imprima por pantalla los siguientes gráficos (no use la función estándar gotoxy()).

a)

```
******
*****
****
***
**
*
```

b)

```
     *
    ***
   *****
  *******
*********
```

c)

```
     *
    ***
   *****
  *******
*********
  *******
   *****
    ***
     *
```

31. Haga un programa que realice lo siguiente: capturar por pantalla un número N. Luego capturar por pantalla N números. Al finalizar la digitación de los N números, despliegue por pantalla la media de los números digitados, el mínimo y el máximo.
32. Escriba un programa que despliegue los *números primos* entre 1 y 100. Un número es *primo* si es divisible exactamente por él mismo y la unidad solamente.
33. El valor de *e* (usado como base en los logaritmos naturales) está definida por una serie como sigue:
e = 1+1/1+1/2!+1/3!+1/4!+1/5!+...+1/n!
Haga una función que retorne el valor de ***e*** considerando **n** términos de la serie.
34. Hacer una función `cdig`(), cuyo prototipo es:
`int cdig(int n,int d);`
Se desea que dicha función retorne las veces que el dígito **d** está contenido en el número **n**. Por ejemplo, cdig(1241,1)=2, cdig(28858,8)=3, cdig(890,4) = 0.
35. Se hace un triángulo con latas de leche condensada, de la siguiente manera: 1 lata en el tope, 3 en el nivel siguiente, 5 en el siguiente etc. Realice una función para cada uno de los acápites siguientes:
    a. ¿Cuántos niveles se tienen con **n** latas?
    b. ¿Cuántas latas sobran haciendo el triángulo si se tienen n latas (con 13 latas se tienen 3 niveles y sobran 4).
36. Todo número entero puede ser factorizado en potencias de factores primos. Escriba un programa que pida números enteros hasta digitar 0. Para cada número, escriba por pantalla la descomposición de la siguiente manera.

```
Digite un número: 60
60 =
    2^2*
    3*
    5
Digite un número: 0
```

37. Números dorsales son aquellos que como 1233 son iguales a la suma de los cuadrados de las unidades de centenas: 1233 = 12*12 + 33*33. Realice la función *int dorsal(int n)* que retorne 1 si n es dorsal y 0 si no lo es.
38. Escriba un programa que lea dos números enteros y encuentre el *máximo común divisor* (el mayor entero que dividirá exactamente ambos números).
39. Escriba un programa que lea dos números enteros e informe por pantalla si ambos son *relativamente primos*. Dos números son *relativamente primos* si no tienen divisores comunes (excepto la unidad).
40. El *mínimo común múltiplo* de dos enteros es el producto de los dos números dividido por el *máximo común divisor*. Haga una función `mcm(a,b)` que retorne el *mínimo común múltiplo* de `a` y `b`.
41. Escriba un programa que capture un número decimal e imprima su equivalencia en binario, octal y hexadecimal. En `printf(),` no use `%o` ni `%x`.
42. Dado dos números enteros `a` y `b` que contengan cifras en notación binaria, haga una función `int sumbin(a,b)` que retorne la suma de `a` y `b` en notación binaria sin hacer conversiones de notación binaria a otra notación (decimal, octal, etc).
43. La *Conjetura de Goldbach* establece que cualquier número par mayor que puede ser representado como la suma de dos números primos (Se puede emplear 2 veces el mismo número). Confirme esta conjetura para los primeros 100 números impares.
44. Un *número perfecto* es un entero que es igual a la suma de todos sus factores excepto él mismo. Escriba una función `perf(n)` que retorne 1 ó 0 según `n` sea perfecto o no.
45. Se desea jugar con la máquina a "adivinar el número". Para ello, elabore un programa en el cual la máquina seleccione internamente un número aleatorio entre 0 y 200. El programa solicitará al usuario que adivine el número que escogió el programa. Dicho programa debe informar si el número digitado por el usuario es mayor o menor que el número a adivinar. Esto se repetirá hasta que el usuario haya adivinado el número. El programa informará la cantidad de intentos realizados por el usuario. Una posible salida sería como la siguiente:

```
Adivina el número secreto: 65 [ENTER]
>> 65 es menor que el número secreto
Adivina el número secreto: 90 [ENTER]
>> 90 es mayor que el número secreto
Adivina el número secreto: 80 [ENTER]
>> 80 es menor que el número secreto
Adivina el número secreto: 83 [ENTER]
>> Excelente!!! 83 es el número secreto.
   Adivinaste en 5 intentos.
```

46. Haga el mismo juego anterior con la salvedad de que sea el computador que adivine un número seleccionado por el usuario. El usuario debe informar al programa si el número propuesto por el computador es mayor o menor. Use un algoritmo de forma que el programa incurra en la menor cantidad de oportunidades posibles. El programa debe detectar e informar si el usuario engañó (mintió) en algún momento. Un ejemplo del formato de la salida del programa sería como sigue:

```
Tome un número en la mente entre 0 y 200
Pulse [ENTER] cuando esté listo.
Es su número secreto el 120?
>> Si es menor, pulse <
>> Si es mayor, pulse >
>> Si adiviné, pulse =
Respuesta: <

Es su número secreto el 80?
>> Si es menor, pulse <
>> Si es mayor, pulse >
>> Si adiviné, pulse =
Respuesta: >

Es su número secreto el 100?
>> Si es menor, pulse <
>> Si es mayor, pulse >
>> Si adiviné, pulse =
Respuesta: <

Es su número secreto el 95?
>> Si es menor, pulse <
>> Si es mayor, pulse >
>> Si adiviné, pulse =
Respuesta: =

Adiviné en 4 oportunidades!!
```

47. Un vendedor le ha solicitado un programa para saber cómo entregar "la devuelta" a los compradores, de manera que le entregue la menor cantidad de monedas posible. Por ejemplo, si un comprador le entrega una moneda de RD$1.00 para pagar un artículo de 21 centavos, el programa debe informarle que debe devolver:
    1 moneda de 50 centavos
    1 moneda de 25 centavos y
    4 centavos
    Elabore dicho programa.

48. Haga el programa anterior de manera que considere billetes.

49. Los *números mellizos* son aquellos que son primos y tienen entre sí una diferencia de dos. Haga un programa que despliegue 5 pares de *números mellizos.*

50. *Problemas de Calendario*. Un año es bisiesto (año de 366 días, en el cual febrero tiene 29 en vez de 28 días) siempre que sea un múltiplo de 4, excepto los años que sean múltiplos de 100, los cuales serán bisiestos solamente si son también múltiplos de 400. Sabiendo que en enero 1 de 1800 fue un miércoles:
    a) Haga una función `bisiesto(a)`, la cual retorne `1` si a corresponde al número de un año bisiesto, de lo contrario retorna `0`.
    b) Haga un programa que lea cualquier fecha (día/mes/año) e imprima el día de la semana correspondiente.
    c) Escriba un programa que imprima en los próximos 10 los años en los cuales habrá "*fin de semana largo*" debido al día de fiesta de la independencia nacional (27 de febrero).
    d) Haga una función `void princal(a)`, la cual imprima por pantalla un calendario correspondiente al año **a**.

51. Un cubo narcisista es aquel que como 370 y 371 son iguales a la suma de los cubos de las cifras decimales que lo componen. Es decir, 370 == 3*3*3 + 7*7*7 + 0*0*0 y 371 == 3*3*3 + 7*7*7 + 1*1*1 Haga la función que determine si un entero **n** es narcisista.

52. La persistencia de un entero es el número de veces que hay que multiplicar sus dígitos hasta alcanzar un solo dígito. Así la persistencia de 715 => 7*1*5 => 35 => 3*5 => 15 => 1*5 = 5 es **3**. Realice una función que dado un entero determine su persistencia.

⌖ Punteros, Arreglos y Matrices

Nota: Las funciones para manipular arreglos se pueden implementar tanto usando índices como usando punteros. Considérese ejercicios de punteros la implementación de dichas funciones de la segunda forma.

1. Haga una función `swap(int a,b)` la cual intercambie los valores de las variables `a` y `b`.
2. Haga una función void `encender(n,b)` de manera que ponga '1' en bit `b` del número `n`. Nótese que la función no tiene valor de retorno.
3. Haga una función void `invbits(int n),` de manera que invierta los bits del número `n`. Por ejemplo, si n = 19 (10011), al ejecutar `invbits(n),` n debe valer 25 (11001).
4. Haga un programa en el cual se digiten la temperatura para cada hora del día. Imprima por pantalla la temperatura promedio para ese día, la temperatura mayor y menor informando la hora a la cual ocurrieron.
5. En un programa se declara un arreglo de 50 elementos de tipo enteros y se inicializa con valores aleatorios entre 0 y 300. Con dicho arreglo, creando funciones para cada operación, haga lo siguiente:
    a. Imprima por pantalla el menor y el menor valor del arreglo.
    b. Calcule el promedio, desviación estándar, moda y mediana de los valores.
    c. Identifique los grupos de 3 que sean números adyacentes (27, 28 y 29 es un grupo de adyacentes)

6. Haga un programa que sume dos arreglos de cualquier tamaño.

7. Haga una función `grafarr(ar, n, c)` a la cual se le pase un arreglo `arr` de `n` números reales y los grafique usando el caracter `c`. La función debe calcular la escala a usar dependiendo del máximo y el mínimo valor contenido en el arreglo. Si el caracter usado fuese asterisco, la apariencia de la salida sería como la siguiente:

```
1 *************** 15.8
2 ******************** 21.3
3 ********************** 24.9
4 ************************ 28.5
5 ****************************** 36.0
6 ********************** 26.6
7 ****************** 19.3
8 ********* 10.1
9 **** 3.5
```

8. Una compañía paga a sus vendedores un 8% de comisión en las ventas diarias de menos de 15,000 y un 9.5% en las ventas de 15,000 o más. Dicha compañía le solicita un programa para ser usado en el día de pago semanal al cual se le ingresará las ventas de los 6 días laborables y el sistema calculará tanto las comisiones del vendedor como la suma que recibe la compañía al deducir las comisiones. La salida podría ser la siguiente:

Día	Venta	Comisión	Diferencia
1	9400	752	8648
2	16800	1596	15204
3	3500	280	3220
4	15000	1425	13575
5	17200	1634	15566
6	1900	152	1748
---	---	---	---
	63800	5839	57961

9. Haga un programa que multiplique arreglos.
10. En el ajedrez, la reina se puede mover, vertical, horizontal y diagonalmente con el alcance que quiera. Escriba un programa que lea una fila y columna de un tablero de ajedrez que determine la posición de la reina. Despliegue por pantalla el tablero de ajedrez, indicando con:
    "R" la posición que ocupa la reina.
    "*" las cuadrículas a las cuales la reina se podría mover y
    "+" las demás cuadrículas. Por ejemplo, un tablero en el cual la reina esté ubicada en la segunda fila y tercera columna, aparecería de la siguiente manera:

```
+  *  *  *  +  +  +  +
*  *  R  *  *  *  *  *
+  *  *  *  +  +  +  +
*  +  *  +  *  +  +  +
+  +  *  +  +  *  +  +
+  +  *  +  +  +  *  +
+  +  *  +  +  +  +  *
+  +  *  +  +  +  +  +
```

11. Escriba un programa que multiplique dos matrices de dimensión n.

12. *Infección en la piel.* Simule una infección en la piel usando un arreglo de 11 x 11 celdas. Cada celda representa una célula en la piel. Comience con la celda del centro "infectada" y las demás "sanas". En cada tiempo de la simulación una celda infectada tiene 0.5 de probabilidad de infectar cada uno de sus vecinos "sanos". Después de seis tiempos de simulación, una célula infectada se hace inmune por cuatro tiempos de simulación y luego se hace saludable nuevamente. Haga un programa que simule este problema, imprimiendo por pantalla el estado de las células para cada paso de simulación, indicando si están infectadas, saludables o inmunes.

Cadenas de Caracteres

1. Las siguientes funciones son estándares en la manipulación de caracteres del Lenguaje C. Investigue qué hacen, sus parámetros y valores de retorno. Haga su propia versión de cada función:
   `strlen` `strchr`
   `strcpy` `strcat`

2. Haga una función que ponga en mayúsculas todas las letras de un string pasado como parámetro

3. Haga una función similar a la anterior, pero convirtiendo a minúsculas.

4. Haga una función `contarc(s,c)`, de manera que retorne cuántas veces se repite el caracter `c` en el string `s`.
   La función estaría declarada de la siguiente manera:
   ```
   int contar(s,c)
   char s[], c;
   ```

5. Haga una función `suschar(st,c,k).` Dicha función debe sustituir en el string `st` todas las ocurrencias del caracter `c` por el caracter `k`. Por ejemplo

6. Haga una función que reciba un string de parámetro y convierta a mayúsculas las letras que estén en minúsculas y que convierta a minúsculas las que están en mayúsculas. La función debe dejar intactos los números y los signos de puntuación que estén contenidos en el string.

7. Realice una función que determine si una cadana de caracteres sólo contiene caracteres numéricos.

8. Dado una cadena de caracteres conteniendo un número expresado en una base entre 2 y 36, llevar dicho número a otra base especificada. El prototipo de la función podría ser **void cambiobase(char *n1, int b1, char *n2, int b2)**, donde **n1** contiene el número en base **b1** y va ser llevado a base **b2** y se almacenará en **n2.**

9. Las siguientes funciones son estándares en la manipulación de caracteres del Lenguaje C. Investigue qué hacen, sus parámetros y valores de retorno. Haga su propia versión de cada función:

```
strcmp          stricmp
strrchr         atoi
itoa
```

10. La función `instring` está declarada de la siguiente manera:

```
int instring(c, s, p)
char *c, *s;
int p;
```

Se desea que dicha función inserte el string `s` en el string `c` a partir de la posición `p`.
Por ejemplo, si c = "LA BLANCA", s = "VACA ", p = 3, el resultado en c sería "LA VACA BLANCA". En caso de que la posición contenida en p sea mayor que la longitud de c, la función retornaría -1.

11. Se desea que la función `postring(s,p)`, en caso de que el string `p` esté contenido en el string `s`, retorne el número de la posición dentro de `s` en la cual comienza `p`. De lo contrario, retorna -1.

```
int postring(s,p)
char s[], p[];
```

12. Haga una función `numpal(s)`, la cual retorne el número de palabras que contiene el string `s`. Considerar que las palabras pueden estar separadas por más de un espacio en blanco. Por ejemplo, si `s` = "Las mil y una noches ", la función debe retornar 5.

13. Haga una función `rev(s)` que invierta el string `s`.

14. Haga una función `strder(s,t)`, la cual retorne la posición de la ocurrencia de `t` en `s` que esté más a la derecha. Por ejemplo, si s = "Las ballenas asesinas" y t = "na", en este caso la función retorna 18.

15. Haga una función `strfin(s,t)` la cual retorne 1 si el string `t` se encuentra al final del string `s`, de lo contrario retorne 0.

16. La función `elimstr()` está declarada como sigue:

```
elimstr(st, pos, numcar)
char st;
int pos, numcar;
```

Esta función debe eliminar en el string `st` tantos caracteres como lo indique `numcar` a partir de la posición `pos`. Por ejemplo, si `st` = "La Bilirrubina", entonces, luego de ejecutar elimstr(st, 1,10), el contenido de `st` sería "Lina".

17. Haga una función `elimpal(s,n)` de manera que elimine la `n`-ésima palagra del string `s`.
Por ejemplo, si `s` = "El lobo feroz", `elimpal(s,2)` hace que el valor de `s` sea "El feroz".

18. Usando redireccionamiento, haga un programa que cuente la cantidad de palabras que tiene un archivo en texto.

19. Similar al ejercicio anterior, haga un programa que cuente la cantidad de veces que se usan las palabras while, for, if en un programa en C.

20. Haga una función `sumrom(r, n1, n2)` que coloque en el string `r` la suma de los número romanos contenidos en `n1` y `n2` los cuales a la vez son también dos string.

21. Similar al ejercicio anterior, haga una función para restar.

22. Hacer un programa al cual se le digite un número entre 0 y 10,000 e imprima por pantalla el número en letras hasta digitar un 0. La salida del programa podría ser como sigue:

```
Digite el número: 1947
1947 es mil novecientos cuarenta y siete
Digite el número: 5015
5015 es cinco mil quince
Digite el número: 0
```

Para estos fines, haga una función `numletras(n,s)`, el cual coloque en la cadena de caracteres `s` el número `n`.

23. Puesto que los tipos de datos para aritmética de enteros tienen tamaños limitados, se desea hacer un programa que trabaje las cifras enteras con cadenas de caracteres, de forma que no haya limitación del tamaño del número. Dicho programa podría tener una estructura como la siguiente:

```
void divis(char *r, char *a,char *b);
void mult(char *r,char *a, char *b);
void suma(char *r, char *a, char *b);
void resta(char *r, char *a, char *b);

int main(int argc, char *argv[])
{
   char n1[]= "2897339855874",
        n2[]= "98347028",r[30];

   divis(r,n1,n2);
   printf("%s / %s = %s\n",n1,n2,r);

   mult(r,n1,n2);
   printf("%s * %s = %s\n",n1,n2,r);

   resta(r,n1,n2);
   printf("%s - %s = %s\n",n1,n2,r);

   suma(r,n1,n2);
   printf("%s + %s = %s\n",n1,n2,r);

}
```

Haga las funciones `divis`, `mult`, `suma` y `resta`.

24. Haga un programa que calcule la frecuencia de cada letra de un archivo de texto, usando redireccionamiento.

25. Haga una función `squeeze2(s1, s2)` que elimine del string `s1` todos los caracteres que estén presentes en `s2`.

Recursividad

1. El *factorial* de un número está definido de la siguiente manera:
   6! = 6*5*4*3*2*1 por lo tanto,
   6! = 6*5!
   considerando que:
   1! = 1 y 0! = 1;

Haga una función recursiva `fact(n)` que retorne el factorial de `n`.

2. Haga una función que sume dos números enteros positivos recursivamente.
3. Repita el ejercicio anterior de forma que la función sume enteros positivos y negativos.
4. Haga una función que multiplique de forma recursiva dos números enteros positivos por sumas sucesivas.
5. Haga una función recursiva que retorne la longitud de un string pasado como parámetro.
6. Implemente la función potencia(b,e) recursivamente (la función retorna b^e)

7. La *serie Fibonacci* se rigue por el siguiente comportamiento:
   $F_1 = 1$
   $F_2 = 1$
   $F_{i+2} = F_{i+1} + F_i \quad (i >= 1)$
   a) Elabore una función recursiva `fib(i)` que retorne el elemento `i` de la *serie Fibonacci*.
   b) Imprima los 15 primeros números de la serie.
   c) Alguien notó que si multiplicamos un número *Fibonacci* por 1.618, se consigue una aproximación al siguiente número en la serie. Pruebe esto y calcule la diferencia para cada par de los primeros 20 números de la serie.

8. Usando el operador módulo (%), se puede calcular el *máximo común divisor* de un número.
   Haga un algoritmo con tal fin.
   Implemente el algoritmo en una función recursiva. Dicha función estaría declarada de la siguiente manera:
   ```
   int mcd(a,b)
   int a,b;
   ```

9. La función `printbin` está declarada de la siguiente manera:
   ```
   void printbin(n)
   int n;
   ```
   Haga la función `printbin`, de forma que imprima por pantalla el número entero `n` en notación binaria usando un algoritmo recursivo.

10. Haga una función recursiva llamada `printinvstr(st)`, la cual reciba como parámetro un string y lo imprima en orden invertido.

11. Haga una función `numbits(unsigned int n)` de forma que retorne el número de bits en 1 que tiene el número n. La función debe ser recursiva. Por ejemplo, el número 9 en binario es 1001, por lo tanto, numbits(9) debe retornar 2.

Pista: Investigue sobre el operador & (and para bits).

12. Implemente la función de invertir un string usando un algoritmo recursivo. Por ejemplo, si el string inicialmente es "ABRACADABRA", luego de ejecutar la función `rev(s)`, el string s debe ser "ARBADACARBA".

13. Implemente la función `cdig()` usando un algoritmo recursivo.

Estructuras

1. Haga un programa al cual se le digite una hora en formato decimal y los convierta a hora, minutos y segundos. La conversión debe realizarla en una función que reciba como parámetros tanto la hora en formato decimal, así como una estructura que contenga hora, minutos y segundos. La salida sería como la siguiente:

```
Digite la hora: 6.5 [ENTER]
6.5 horas = 6:30:00
```

2. Haga un programa que imprima un archivo de texto por pantalla haciendo pausa cada 20 líneas. Para este programa no use redireccionamiento, sino las funciones estándares de manejo de archivos y las funciones estándares de manejo de cadenas de caracteres.

3. Resuelva este problema usando la información expuesta en "Problemas de Calendario".
   Se tiene una estructura como la siguiente:
   ```
   struct tipofecha {
       int dia,mes,ano;
   };
   ```
   Usando dicha estructura, haga la función `calcdias(f1, f2)`, la cual retorne los días transcurridos entre `f1` y `f2`. La declaración de la función sería como sigue:
   ```
   long int calcdias(f1, f2)
   struct tipofecha f1, f2;
   ```

4. *Aritmética de complejos*.
   a) Declare una estructura `struct complex` para representar números complejos.
   b) Elabore cuatro funciones que tomen como parámetro dos variables del tipo `struct complex` (suma, resta, multiplicación y división) para hacer operaciones con números complejos usando la estructura anterior.

5. En una clase de laboratorio hay 10 estudiantes. Cada estudiante fue evaluado en 5 prácticas. Luego de digitar todos los datos, se desea que aparezca por pantalla las siguientes informaciones.
   a) Promedio de la clase para cada práctica
   b) Promedio de prácticas para cada estudiante
   c) La calificación final del estudiante será el promedio de sus cuatro mejores calificaciones.
   d) Promedio final de las calificaciones de los 10 estudiantes
   e) Nombre de los estudiantes con calificaciones sobre el promedio de la clase.

6. *Geometría Plana*. En lo adelante, se hace referencia a las siguientes estructuras:

```
struct PUNTO {
   float x,y;
}

struct RECTA {
   struct PUNTO p1, p2;
}

struct TRIANGULO {
   struct PUNTO p1,p2,p3;
}
```

7. Haga la función `pendiente(RECTA r)`, la cual retorne la pendiente de la recta pasada como parámetro.

8. Usando la función `pendiente(r)`, haga una función `rectasrel(RECTA r1, r2)`, la cual retorne 0 si r1 y r2 coinciden, 1 si son rectas paralelas, 2 si son perpendiculares, de lo contrario retorna -1.

9. Escriba una función dettriang(PUNTO p1,p2, p3) que retorne 1 ó 0, según los puntos p1, p2 y p3 determinen un triángulo o no.

10. Implemente la función `tipotriangulo()`, pero en vez de pasar como parámetro las longitudes de los lados, la función recibirá un puntero a una estructura `TRIANGULO`.

11. Haga una función int `intersec(RECTA r1,r2; PUNTO p)`, la cual coloque en la variable `p` el punto en la cual se intersectan. En caso de que sean paralelas o sean la misma recta, la función retorna 0, de lo contrario retorna 1.

12. Haga una función `caja(pos1, pos2)` que dibuje un rectángulo en la pantalla con el caracter '*'. En dicha función, `pos1` contiene la coordenada de la esquina superior izquierda del rectángulo y `pos2` contiene la esquina inferior derecha.

⌖ Archivos

1. Usando redireccionamiento, haga un programa que cuente las líneas de un archivo en texto.

2. Se desea calcular la frecuencia con que ocurre cada letra en un archivo de texto. Haga un programa para ello usando las funciones estándares de manejo de archivos.

3. Haga un programa que tome un archivo con un código fuente en C y lo envíe a otro archivo de texto habiendo eliminado los comentarios (Texto contenido entre /* y */).

4. Haga un programa cola.c el cual sea llamado con un parámetro conteniendo el nombre de un archivo en texto e imprima las 10 últimas líneas de dicho archivo.

5. Haga un programa contpal.c, el cual cuente las palabras que contiene un archivo en texto.

6. Haga un programa que construya un histograma con las longitudes de las palabras contenidas en un archivo de texto. Para construir el histograma, use la función `grafarr`().

Parte II. Otros ejercicios

1. Haga una función `aleat(n)` que retorne números aleatorios entre 0 y n. Una estrategia es usar los "ticks" del reloj del computador.

2. Haga un programa que simule 1000 lanzamientos de un dado. Despliegue por pantalla la frecuencia con que salieron cada una de las seis caras del dado.

3. Haga un programa para jugar *Tic-Tac-Toe* contra el computador. El juego del *Tic-Tac-Toe* es el mismo de *"Tres en raya"* usando "x" y "o", por ejemplo:

```
X   0   X
0   X   0
0   X   X
```

4. Haga un programa que simule una máquina traganíqueles.

5. *Blackjack*. En el juego del *Blackjack* se le entrega dos cartas a un jugador y tiene la opción de pedir más cartas a la banca. El objetivo del juego es acumular un total de la suma de las cartas tan alto como sea posible siempre que no pase de 21. En este último caso, el jugador "se pasa" y pierde la mano.
   En cada mano, el jugador tiene la opción de pedir más cartas o "quedarse" con las dos cartas iniciales que le entregaron. El jugador pide cartas para tratar de acercarse tanto como pueda a sumar 21.
   "La Banca" juega igual que el jugador, sin embargo la banca no tiene opciones: mientras las dos cartas de la banca suman 16 ó menos, la banca debe tomar otra carta. La banca no solicita cartas cuando su suma está entre 17 y 21 inclusive. La banca pierde cuando se pasa de 21 o cuando la suma del jugador se aproxima más a 21 que la suma de la banca.
   Las cartas 2 a 10 son contadas en virtud de sus números, sin embargo, el alfil, la reina y el rey (J,Q,K) se cuentan como 10. El as se cuenta como 1 ó como 11. Se cuenta como 11 siempre que no "pase" al jugador de 21, de lo contrario se cuenta como 1. Nótese que un caso particular es cuando un jugador tiene dos aces, en cuyoo caso se cuenta solamente una como 11.
   Haga un programa en el cual el usuario juegue *Blackjack* con la máquina, la cual hará el papel de la banca. Deber aparecer en la pantalla la información de cuantos juegos ganados y cuantos perdidos tiene el usuario. Debe controlar las bajaras que se van usando, las cuales deben ser entregadas al azar.
   Cuando queden 5 cartas o menos, el programa debe "bajarar" las cartas y disponer de todas las cartas nuevamente.

6. Cinco reinas pueden ser ubicadas en un tablero de ajedrez de forma que todas las casillas sean dominadas por lo menos por una reina. Escriba un programa que determine cómo ubicar las cinco reinas en un tablero de ajedrez.

7. Repita el ejercicio anterior:
   a) con doce caballos
   b) con doce alfiles.

8. *El tour del caballo*. Escriba un programa de forma que un caballo de ajedre visite las 64 cuadrículas del tablero una sola vez. Es decir, el caballo nunca tocará dos veces una misma cuadrícula. Seleccione la cuadrícula inicial aleatoriamente.

9. Tres misioneros, tres caníbales y un bote están a la orilla de un río. El bote no puede llevar más de dos personas. Si en un momento dado los caníbales son más que los misioneros en cualquier orilla del río, los caníbales se comen a los misioneros. Escriba un programa que calcule cómo cruzar todos los caníbales y los misioneros sin que ningún misionero sea digerido por los caníbales.

10. La *Torre de Hanoi*.

11. *El juego de la vida*. En la *Scientific American*, febrero de 1971, la siguiente situación es descrita.
    En un momento *t*, una casilla está "viva" ya sea porque:
    a) estaba vacía en el momento *t* - 1 y exactamente tres de sus vecinos estaban vivos, o
    b) estaba viva en el momento *t*- 1 y dos o tres de sus vecinos estaban vivos.
    De lo contrario, la casilla queda vacía o "muerta".
    Los vecinos de una casilla son las ocho cuadrículas que la rodean.
    En la situación siguiente, las X's son los vecinos de Y:

    X X X
    X Y X
    X X X

    Es importante calcular la configuración para cada ciclo tan eficientemente como sea posible. Su trabajo será simular una configuración de 15 x 15 celdas, escogiendo aleatoriamente 20 celdas "vivas" para iniciar la simulación. Escriba un programa que simule 5 ciclos.

Direcciones Internet de Interés:

http://vinny.csd.mu.edu/learn.html
"LEARN C/C++ Today"
Información para aprender C/C++. Documentación sobre compiladores recomendados, otras direcciones Internet, libros, etc.

http://arachnid.cs.cf.ac.uk/Dave/C/CE.html
"Programming in C"
Tutorial completo para aprender C. Está dividido por temas. Incluye ejercicios con sus soluciones.

http://199.72.148.16/classes/c/c1.htm
"Introduction to C Programming". Lecciones tutoriales para Lenguaje C.

Notas a tomar en cuenta en los programas

- Para las constantes siempre use #define
- Recuerde indentar el programa
- Ponga un breve comentario en el encabezado de cada función donde se explique qué hace la función, valor de retorno y descripción de los parámetros.
- Cuando se le solicite escribir una función, haga un programa en el cual pruebe que dicha función está correcta.