Variables, operadores y tipo de datos. Especificadores de formato. Entrada y salida estándar

1. Indicar qué es lo que se almacena en cada una de las variables indicadas a continuación, si se realizan las operaciones que se indican en forma consecutiva:

```
int a=3,b=2,c=1, d,e;
float m=2.5, n=5.4, r,s;
```

Operación	Resultado	Operación	Resultado	Operación	Resultado	Operación	Resultado
d=m;		d=a%2;		s=m+n;		r=a+1;	
d=n;		e=b/c;		s=3+4-1;		r=a+1.0;	
e=3.7;		e=c/b;		s=3.0+4.0-1;		a++;	
d=a;		r=a+b;		a=m;		r++;	
d=a+b;		r=a/b;		a=m/2;		b--;	
d=a/b;		r=a/2;		a=m/2.0;		a+=5;	
d=a%b;		r=a/2.0;		a=3.0+4.0-1;		s*=5;	

2. Indicar cuál es la salida por pantalla para cada sentencia printf, siendo:

```
int a=5;
float b=13.546;
char c='A';
char d='a';

1) printf ("a vale %d", a);
2) printf ("a vale %O", a);
3) printf ("a vale %X", a);
4) printf ("a vale %f", a);
5) printf ("a vale %c", a);
6) printf ("b vale %d", b);
7) printf ("b vale %5.2d", b);
8) printf ("b vale %f", b);
9) printf ("b vale %.1f", b);
10) printf ("b vale %.2f", b);
11) printf ("b vale %6.4f", b);
12) printf ("b vale %6.1f", b);
13) printf ("b vale %c", b);
14) printf ("b vale %O", b);
15) printf ("b vale %X", b);
16) printf ("c vale %d", c);
17) printf ("c vale %f", c);
18) printf ("c vale %c", c);
19) printf ("c vale %O", c);
20) printf ("c vale %X", c);
21) printf ("d vale %d", d);
22) printf ("d vale %f", d);
23) printf ("d vale %c", d);
24) printf ("d vale %O", d);
25) printf ("d vale %X", d);
```

3. Indicar qué queda almacenado en la variable con la que se ingresan datos en cada caso, si se han definido las variables de la siguiente forma:

```
int a;
float b;
char c;

1) scanf ("%d", &a);
2) scanf ("%f", &a);
3) scanf ("%c", &a);
4) scanf ("%d", &b);
5) scanf ("%f", &b);
6) scanf ("%c", &b);
7) scanf ("%d", &c);
8) scanf ("%f", &c);
9) scanf ("%f", &c);
```

4. ¿Cuáles de los siguientes son tipos válidos?

```
1) unsigned char
2) long char
3) unsigned float
4) double char
5) signed long
6) unsigned short
7) signed long int
8) long double
9) long bool
```

Operadores de asignación compacta

5. Suponiendo los siguientes valores iniciales para las variables (int):

```
x = 2; y = 6; z = 9; r = 100; s = 10; a = 15; b = 3;
```

¿Cuál es el resultado de las siguientes expresiones?

```
a) x += 10;
b) s *= b;
c) r /= 0;
d) y += x + 10;
e) z -= a*b;
```

6. Usa expresiones equivalentes para las siguientes, usando operadores mixtos.

a)
```
x = 10 + x - y;
x += 10-y
x -= y+10
x += 10+y
```

b)
```
r = 100*r;
r *= 100*r
r *= 100
r += 100
```

c)
```
y = y/(10+x);
y /= 10*x
y /= 10 + y/x
y /= 10+x
```

d)
```
z = 3 * x + 6;
z += 6
z *= 3
no es posible
```

7. Evalúa cada una de las siguientes expresiones. Luego comprueba el resultado en el laboratorio.

```
1) 8 * 6 / 3 * 4
2) (8 * 6) / 3 * 4
3) 8 * 6 / (3 * 4)
4) (8 * 6 / 3 * 4)
5) (8 * 6) / (3 * 4)
6) 1 + 4 * 5 + 8 / 4 + 4
7) 1 + 4 * 5 - 8 / 4 + 4
```

8. Supone que las variables a, b y c tienen asignados los valores 49, 5 y 3 respectivamente. Encuentra:

```
1) a % b * c + 1
2) a % (b * c ) + 1
3) 24 / c * 4
4) (int) (a / b ) % 2
5) 48 / (c*2) * 4
```

9. Desarrolla un algoritmo que permita leer 2 valores y emitir por pantalla la suma de los dos, la resta, producto, división, promedio y el doble producto del primero menos la mitad del segundo.

10. Encuentra el error en cada uno de los siguientes programas e indica de qué tipo es.

```c
#include <stdio.h>

int main(){
    int n,total;
    float promedio;
    n=0;
    promedio=total/n;
    printf("Prom=%f\n",promedio);
    return 0;
}
```

```c
#include <stdio.h>

int main(void){
    integer x;
    real y;
    scanf ("%d",y);
    printf("%f",x)
    return 0;
}
```

11. Desarrolla un algoritmo que permita, dados ciertos centímetros como entrada de tipo flotante, emitir por pantalla su equivalencia en pies (enteros) y en pulgadas.

12. Construye un programa que pregunte los años que tienes y emita como respuesta el número de días vividos.

13. Construye un programa que dados el valor de 1 kg de determinada mercadería y la cantidad mercadería comprada, emite el valor del total a pagar.

14. Construye un programa que permita ingresar los valores de 2 de los ángulos interiores de un triángulo, y se emita por pantalla el valor del restante.

15. Construye un programa que permita ingresar las medidas de los lados de un rectángulo; el mismo debe emitir por pantalla su superficie y su perímetro.

16. Construye un programa que permita ingresar la superficie de un cuadrado (en m2), el mismo debe emitir por pantalla su perímetro.

17. Construye un programa que permita ingresar 2 tiempos, expresados en horas, minutos y segundos, el mismo debe emitir por pantalla la suma de ambos (también en horas, minutos y segundos).

Ejercicios integradores

18. Desarrollar un algoritmo que le permita leer un valor para el radio (R) y calcule el área (A) de un círculo y emitir su valor.

19. Determinar la hipotenusa de un triángulo rectángulo conocidas las longitudes de sus dos catetos. Desarrollar los correspondientes algoritmos.

20. Dada una cantidad entera de segundos (menor a 86400) y conviértela en horas, minutos y segundo utilizando los operadores de cociente y resto enteros.

21. Dada una fecha en el formato DDMMAAAA y separarlo en Dia, Mes y Año utilizando operaciones aritméticas.

22. Desarrollar un algoritmo que permita leer un valor que represente una temperatura expresada en grados Celsius y convierta dicho valor en un valor expresado en grados Fahrenheit.

23. Desarrollar un algoritmo que permita calcular el área de un triángulo en función de las longitudes de sus lados previamente leídos desde el teclado.

24. Desarrollar un algoritmo que permita determinar el área y volumen de un cilindro cuyo radio (r) y altura (h) se leen desde teclado.

25. Una empresa paga sueldos calculando el valor del sueldo básico por la cantidad de horas trabajadas y un plus por antigüedad que corresponde al 2% por cada año trabajado. El sueldo básico es de $4500. Se desea ingresar la cantidad de horas trabajadas, la antigüedad y obtener el sueldo neto.

26. Se ingresa un número entero positivo de dos o más cifras y obtener su última cifra.

27. Dado un número entero de tres cifras mostrar sus cifras por separado.

28. Realizar un programa que devuelva la raíces de un polinomio de 2do grado ingresado los coeficientes a,b y c.

Ejercicios con condicionales

29. Indicar cual es la salida de los siguientes fragmentos de código:

a)
```
...
int x = 2 , y = 6 , z = 4 ;
if (x > y || x < z){
    printf ( "Verdadero" );
}
else{
    printf ( "Falso" );
}
...
```

b)
```
...
int x = 2 , y = 6 ;
if (x < y && x == y)printf (
"Verdadero" );
else printf ( "Falso" );
...
```

c)
```
...
int x = 2 , y = 6 ;
if ((x < y && x != y) || !(x == y))
   printf ( "Verdadero" );
else
   printf ( "Falso" );
...
```

d)
```
...
int x = 2 , y = 6 ;
if ((x < y && x != y) || !(x == y))
   printf ( "Verdadero" );
else
   printf ( "Falso" );
...
```

30. Completar en el contexto el siguiente fragmento con las condiciones correspondientes:

a)
```
...
int num;
printf ( "Ingresar un numero" );
scanf ( "%d" , &num);
if (_____________________)
   printf ( "%d no es multiplo de 5.\n" , num);
...
```

b)
```
...
int mes, dia;
printf ("Ingrese nro de dia:" );
scanf ("%d", &dia);
printf("Ingrese nro de mes:");
scanf ( "%d" , &mes);
if (_____________________)
   printf ( "Corresponde al primer trimestre" );
...
```

31. Desarrollar un algoritmo que ingrese por consola dos números enteros, a continuación, indique si son iguales o distintos.

32. Desarrollar un algoritmo que ingrese un numero por teclado y luego indique si es par o impar.

33. Desarrollar un algoritmo que Ingrese tres números e indique cuál es el menor, si los tres son iguales indicarlo con un mensaje.

34. Desarrolla un algoritmo que le permita leer dos valores A y B e indicar si la suma de los dos números es par.

35. Desarrollar un algoritmo que ingrese un número entero de cuatro dígitos e indique con un mensaje si es capicúa.

36. Una compañía dedicada a servicio de mensajería realiza envios al interior el costo fijo de traslado es de $1500 si es corta distancia y 2000 si es larga distancia, luego dependerá del peso de la mercadería enviada, los de corta distancia si el peso supera los 20 kilos se le cobran $800 por cada kilo de exceso y los de larga distancia se le cobra $800, cada 5 kilos excedidos. Desarrollar un algoritmo que ingrese el tipo de viaje y la cantidad de kilos y me devuelva el costo del viaje.

37. Desarrollar un algoritmo que ingrese tres caracteres y mostrarlos ordenados según el orden ascendente del alfabeto.

38. Desarrollar un algoritmo que ingrese dos números y luego un carácter que indique una operación (S-Suma, R-Resta, M-Multiplicación, D-División) y luego realice la operación correspondiente, tener en cuenta que no se pueden realizar divisiones por cero.

39. Desarrollar un algoritmo que ingrese nota entera (entre 1 y 10) de un alumno correspondiente al promedio obtenido, se pide mostrar el siguiente mensaje: "El alumno obtuvo un sobresaliente", según el promedio: 10-Sobresaliente, 8 y 9-Distinguido, 6 y 7-Bueno, 4 y 5-Aprobado, 1,2,3-Reprobado.

40. Desarrollar un algoritmo que ingrese fecha de nacimiento como tres enteros (DD, MM, AAAA), de una persona y mostrar por pantalla su fecha de nacimiento, de la siguiente forma "La persona nació el 5 de marzo de 1973".

41. Desarrolle un algoritmo que ingrese los valores de los lados de un triángulo. Valide si las medidas pueden formar un triángulo y luego indique a través de un mensaje que tipo de triángulo es (EQUILÁTERO, ISÓSCELES, O ESCALENO).

42. Construir un programa que ingrese un carácter y determine si es una vocal.

Ejercicios con ciclos

43. ¿Cuál es la salida de los siguientes fragmentos de código?

a)
```
...
int x;
x = 10;
while (x > 10){
   x = x - 3;
   printf("%d", x);
}
```

b)
```
...
int x;
x = 10;
while (x > 10){
   x = x - 3;
   printf("%d", x);
}
...
```

c)
```
...
int x;
x = 0;
do{
   printf("%d", x);
   x = x + 1;
}while (x!=5);
...
```

d)
```
...
int x = 0, y = 0;
do{
   x = x + 2;
   y = x - 2;
   printf("%d %d\n", x, y);
}while (y!=5);
...
```

e)
```
...
int i = 4, x = 5;
for (i = 0; i < 10; i++){
   if (i < x)
     printf("%d ", i);
   else
     printf("%d ", i - x);
}
...
```

f)
```
...
int i = 4, x = 5;
for (i = x; i < 10; i++){
   printf("%d, ", i);
}
...
```

44. Desarrollar un algoritmo que escriba por pantalla los primeros 100 números naturales.

45. Desarrollar un algoritmo que ingrese 10 números enteros y muestre por pantalla un mensaje únicamente cuando son positivos.

46. Desarrollar un algoritmo que, ingrese un número entero entre 0 y 10 (validar este valor) y muestre por pantalla la tabla de multiplicar del número ingresado.

47. Desarrollar un algoritmo que ingrese números hasta ingresar un número negativo, se pide mostrar por pantalla el promedio.

48. Desarrollar un algoritmo que ingrese números hasta leer un cero, calcular y mostrar por pantalla cuántos de ellos son negativos y cuántos son positivos.

49. Desarrollar un algoritmo que calcule y visualice en pantalla una tabla con las 20 primeras potencias de 2.

50. Desarrollar un algoritmo que ingrese un número entero positivo y muestre por pantalla todos sus divisores.

51. Desarrollar un algoritmo que ingrese un entero positivo, y muestre por pantalla la suma de sus cifras.

52. Desarrollar un algoritmo que ingresa un número entero mayor o igual cero (validar) y luego muestre por pantalla la factorial del mismo. (https://www.smartick.es/blog/matematicas/numeros-enteros/factoriales/)

53. Desarrollar un algoritmo que escriba en pantalla todos los números pares comprendidos entre 1 y 50.

54. Desarrollar un algoritmo que ingrese dos números enteros, (primero < segundo), validar y muestre por pantalla los números enteros del primer número al segundo.

55. Desarrollar un algoritmo que pida al usuario dos números y una letra: “I”, “i” ó “p”, “P”, luego mostrar en pantalla los pares (si se pulsó la “p”, “p”) ó impares (si se pulsó la “i”, “I”), comprendidos entre el primer número y el segundo. Tener en cuenta que el primer número debe ser menor al segundo y validar que ingrese las letras correspondientes.

56. Se realizó un concurso de tiro al blanco. Existen 5 participantes y cada uno de ellos efectúa 10 disparos, registrando las coordenadas (x, y) de cada disparo. Indicar cuántos disparos se produjeron en cada cuadrante y cuantos dieron en el blanco.

Funciones

57. Desarrolle la función “mayorDeDos” que reciba dos enteros por parámetro y muestre por pantalla al mayor de ellos. Luego modifíquela a “getMayorDeDos” para que devuelva el mayor como resultado.

58. Desarrolle la función “potencia” que reciba por parámetros un entero X y una potencia Y; y devuelva por resultado XY.

59. Desarrolle la función “tablaDeMultiplicar” que reciba un entero por parámetro y muestre por pantalla su tabla de multiplicar de 0 a 10.

60. Desarrolle la función “sumaIntervalo” que reciba dos enteros por parámetro y devuelva por resultado la suma de todos los números enteros entre dichos valores (inclusive).

61. Desarrolle la función “menu” que muestre por pantalla 4 opciones, pida ingresar una de esas opciones y devuelva por resultado la opción elegida. La función debe validar que la opción ingresada sea valida si hay 4 opciones, no debo poder elegir la opción 6).

62. Desarrolle la función “esPrimo” que devuelva verdadero si el numero pasado por parámetro es un numero primo.

63. Desarrolle la función “múltiplo” que recibe dos valores enteros y emite “verdadero” si el primero es múltiplo del segundo. Desarrolle la función “areaRectangulo” que reciba 3 parámetros (base, altura y área) devolviendo el área en los parámetros.

64. Realiza una función de nombre Siguiente tal que, recibiendo un número primo mayor que uno, devuelva el número primo inmediatamente siguiente y superior a dicho número primo. Por ejemplo, si se invoca siguiente(7), la función devolverá el número 11.

65. Desarrolle la función “esBisiesto” que reciba un año por parámetro y devuelva “verdadero” si el año es bisiesto (Un año es bisiesto si es divisible por 400, o bien si es divisible por 4 pero no por 100).

66. Desarrolle la función “fechaValida” que reciba por parámetro un día, un mes y un año y devuelva por resultado “verdadero” si la fecha es válida (tener en cuenta años bisiestos).

67. Desarrolle la función void maxmin (int x1, int x2, int* max, int* min); que recibiendo por parámetros x1 y x2, devuelva el menor de ambos en min y el mayor en max.

68. Desarrolle la función void datoValidado (int *dato, int min, int max) que reciba un mínimo y un máximo por parámetro; que pida por teclado el ingreso de un valor, valide que el valor este entre min y max y lo devuelva en *dato.

69. Desarrollar una función de encabezado int ordenarMayor(int* v1, int* v2, int*v3) en la que la función ponga en V1 el menor valor de las tres variables, en v2 el del medio y en v3 el mayor. Noten el encabezado y el nombre de la función (que haga lo que el nombre de la función dice que hace).

70. Luego, análogo al punto anterior, desarrollen un función int ordenarMenor(int* v1, int* v2, int*v3).

71. Con las funciones de los dos puntos anteriores, queda la pregunta: ¿sería posible hacer una única función que pudiera ordenar las tres variables de menor a mayor o de mayor a menor según se requiera? Plantéenlo, desarróllenlo y hagan un programa que las utilice. Si pudieron hacerlo, dieron un gran paso hacia la anidación de funciones.

72. Desarrolle la función void acumulador(int* acumCat1, int* acumCat2, char categoría) y que acumule en acumCat1 si la categoría ingresada es ‘A’ o en acumCat2 si es ‘B’. Utilice la función en un programa con un ciclo para que acumule.

73. Escribe una función que reciba como parámetro de entrada un número entero y devuelva como resultado el número de cifras del número.

74. Escribe una función que reciba como parámetros de entrada un valor entero y compruebe si se encuentra comprendido entre dos valores constantes MIN y MAX definidos dentro de la propia función.

75. Escribe una función que reciba como parámetros de entrada tres números enteros que representan las longitudes de tres segmentos rectilíneos, y devuelva como resultado un valor de tipo lógico que indique si dichos segmentos pueden formar o no un triángulo (la condición necesaria pero no suficiente es que ninguno de los segmentos tenga una longitud superior a la suma de los otros dos).

76. Escribe un programa teniendo en cuenta las siguientes funciones
    - leeOpcion lee la opción deseada y comprueba su validez
    - menú muestra el menú en la pantalla
    - cuadrado, circulo, rectángulo, trapecio, triángulo calculan la superficie correspondiente.

    El menú por mostrar sería algo como lo que sigue:

```
**********************************
****** CÁLCULO DE SUPERFICIES ******
1. Cuadrado (lado*lado)
2. Círculo (pi*radio*radio)
3. Rectángulo (base*altura)
4. Trapecio (base1+base2)*altura/2)
5. Triángulo (base*altura)/2)
0. Salir del programa
**********************************
```

77. Construir una función que permita procesar los datos de productos vendidos. La función pedirá por teclado:

    El id de producto (número entero > 0 y menor a 1000). Si es invalido, saltearlo; si es válido, pedir lo demás:
    - Precio de costo, ej: 5.30
    - Precio de venta, ej: 9.50
    - Cantidad vendida, ej: 25

    La función debe:
    - Mostrar por pantalla la ganancia de cada id de producto.
    - Calcular las ganancias totales de todos los id de productos.
    - Calcular la cantidad de id de productos procesados.

78. Desarrollar un programa que utilice la función anterior y que emita el total de ids de productos procesados y las ganancias que se obtuvieron por todas las ventas. Ej: se procesaron 43 productos, las ganancias obtenidas fueron $2398.

79. El videojuego Mortal Kombat es un juego de lucha entre dos jugadores. Desarrollar el juego de lucha entre dos jugadores que:
    - Al iniciar el programa, pida los datos de cada jugador: vida (int), ataque (int), defensa (int).
    - Implemente la función atacar() que debe recibir los parámetros necesarios para modelar el ataque de un jugador a otro.
    - Al iniciar la pelea, el jugador1 inicia atacando al jugador2. Si el jugador2 muere, se termina el juego. Si no, el jugador2 devuelve el ataque. Si el jugador1 muere, termina el juego.
    - Al finalizar la pelea, se debe mostrar qué jugador ganó y con cuánta vida quedó.

Ejercicios de punteros

80. ¿De qué tipo es cada una de las siguientes variables?

```
int *a, b;
int *a,*b;
```

81. Si se declara: float x, *p; ¿Cuál de las siguientes expresiones es correcta?

- p=&x;
- Ninguna de las restantes respuestas es correcta x=p*;
- &x=p;
- &p=x;

82. Explica el error.

```
char c = 'A';
double *p = &c;
```

83. Un programa en C contiene las siguientes sentencias: a. ¿Qué valor tiene a al finalizar el programa?

```
float a = 0.001, b = 0.003;
float c, *pa, *pb;
...
pa = &a;
*pa = 2 * a;
pb = &b;
c = 3 * (*pb - *pa);
```

Responda:

a. ¿Qué valor tiene b al finalizar el programa?
b. ¿Qué valor tiene c al finalizar el programa?
c. ¿Qué valor tiene (*pa) al finalizar el programa?
d. ¿Qué valor tiene (*pb) al finalizar el programa?

84. El siguiente código contiene un error, ¿cuál es?:

```
int main (){
    int x = 5;
    float y = 5;
    int *xPtr = NULL;
    xPtr = &y;
    printf ("%d", *xPtr);
    return 0;
}
```

Ejercicios con vectores

85. Escribe un programa que almacene, en un vector con dimensión 10, los números 10 primeros números de la Quiniela. El ingreso de los datos será indicando posición-numero, de modo que puedo ingresar la posición 5 antes que la posición 2.

86. Escribe un programa con un menú que defina las siguientes funciones y emita los resultados, la dimensión del vector será de 10 (elementos):
    - Cargar m elementos de un vector por teclado. Deberá considerar que no podrá superar el máximo de elementos de vector: void cargar(int vect[], int unNum);
    - Leer elementos de un vector hasta encontrar el número entero a. Deberá retornar el número de elementos que ha leído sin contar el entero a. int leerMarca(int vect[]);
    - Dado un elemento y dado un vector de enteros, desarrolle una función que devuelva el número de apariciones del elemento en el vector. int veces(int valor, int vect[], int unNum);

Invertir los elementos del vector sin utilizar otro vector. Por ejemplo, el vector formado por los enteros: 1 2 3, debe quedar 3 2 1. void invertirOrden(int vect[], int unNum);

87. Continúa agregando funciones al ejercicio anterior que permitan:
    - Calcular y emitir la suma de sus elementos.
    - Calcular y emitir el mínimo del vector.
    - Calcular y emitir el promedio de los valores del vector
    - Emitir los valores de aquellos que superaron ese promedio.
    - Emitir los valores del vector que son múltiplos del último número ingresado en el mismo.
    - Emitir el valor máximo e indicar la cantidad de veces que apareció y el número de orden en que fue ingresado.
    - Emitir los valores que son pares.
    - Emitir los valores que son impares.
    - Emitir aquellos que estén ubicados en posición par.

88. Escribe un programa que, a partir de un vector vacío de enteros, permita insertar nuevos números en posiciones válidas del mismo. El programa emitirá repetidamente al usuario un menú con cuatro opciones:
89. Insertar delante: Esta opción agrega un número en la primera posición del vector. Deberá desplazar, si fuese necesario, el resto de los elementos una posición a la derecha. Si el vector estuviese lleno, se perdería su último elemento.
90. Insertar detrás: agrega un nuevo elemento al final del vector, es decir, en la última celda. Si el vector estuviese lleno, el elemento se inserta en la última celda del vector y se perdería el elemento que estaba allí anteriormente.
91. Insertar en una posición dada: Dado un índice del vector, se debe insertar un nuevo elemento en dicha posición, siempre que el nuevo elemento sea contiguo a los ya existentes. Por ejemplo, si en el vector hay tres elementos (que ocupan las posiciones 0, 1 y 2) no se permite agregar un elemento en la posición 7 porque no forma una secuencia continua, pero sí se permite en la posición 3. Por otra parte, si la posición dada está ocupada, los elementos se deben desplazar una posición a la derecha y, si el vector estaba lleno, se pierde el último elemento.
92. Finalizar el programa.

93. Desarrolle una función que reciba un vector de 10 números enteros y muestre los valores por pantalla, eliminando las repeticiones.
94. ¿Qué está mal en el siguiente fragmento de código?

```
int b[10];
for (i=0;i<=10;i++)b[i]=b[i+1];
```

95. Construye un programa, utilizando funciones, que lea como máximo 10 números enteros, el ciclo termina al introducir un cero. A continuación, lea otro entero y compruebe si ese entero está o no entre los anteriores. En caso de que se encuentre, contar y emitir el número de veces que aparece.
96. Escribe un programa que imprima los elementos de un vector de enteros en orden inverso utilizando punteros (no utilizar subíndices []).

```
int v[10] = {1,2,3,4,5,6,7,8,9,10};
int *p;
```

97. Escribe una función que reciba un vector de enteros y su tamaño, y retorne la cantidad de números impares que contiene. Trabaja con notación de punteros.

98. Dadas las siguientes declaraciones:

```
int v[3] = {10,20,30};
int *p;
p = v;
```

Explica que imprimiría el printf en cada caso de los siguientes:

a) (*p)++; printf (“%d”, *p);
b) *(p++); printf (“%d”, *p);
c) *p++; printf (“%d”, *p);

99. Construye una función tal que dados dos vectores de 5 elementos cada uno, los concatene en un tercer un vector de 10 elementos.
Ej: `V1= 2-56-7-8-30; V2= 7-80-2-4-13; V3= 2-56-7-8-30-7-80-2-4-13;`

100. Ídem anterior, pero los elementos de los dos vectores deben concatenarse intercalados.
Ej: `V4= 2-7-56-80-7-2-8-4-30-13;`

101. Se ingresan los N y M elementos de los arreglos unidimensionales A y B, respectivamente. Desarrollar 3 funciones que devuelvan la unión, diferencia e intersección de dichos vectores.

102. Escriba una función que devuelva un vector de 6 posiciones que deben tener los valores de la lotería generados aleatoriamente entre 1 y 54 (utilice la función rand).

103. Para probar un congelador, la fábrica registra en un listado la temperatura en el interior durante todos los días del mes de junio. Escriba una función que reciba un vector con todas estas temperaturas (generalmente, negativas) y devuelva la mínima temperatura. Luego escriba una segunda función que diga en qué día del mes se produjo la temperatura mínima.

104. Desarrolle una función que pueda procesar cualquier vector de enteros, y emita el promedio de los valores que estén cargados.

105. Desarrollar una función que reciba un vector, su tamaño y un número; y devuelva la posición del número que se encuentra o -1 si no se encuentra.

106. Desarrolle una función recursiva que reciba un vector de enteros y devuelva la suma de los elementos.

Ejercicios con cadenas

107. Desarrolle una función que reciba una cadena por parámetro y devuelva su tamaño. No puede usar funciones de cadenas en el desarrollo.

108. Desarrolle una función que reciba una cadena por parámetro y devuelva la cantidad de vocales que contiene. No puede usar funciones de cadenas en el desarrollo.

109. Desarrolle una función que reciba una cadena por parámetro y devuelva la cantidad de consonantes. No puede usar funciones de cadenas en el desarrollo.

110. Desarrolle una función que reciba una cadena por parámetro y un carácter, y que reemplace todas las ocurrencias de ese carácter por X.

111. Desarrolle una función que reciba dos cadena por parámetro y devuelva: 1 si la primer cadena es mayor, 0 si ambas son iguales o -1 si la primera es menor. No puede usar funciones de cadenas en el desarrollo.

112. Desarrolle una función que reciba dos cadenas por parámetro y retorne 1 si la segunda cadena es subcadena en la primera y cero en caso contrario.

113. Desarrolle una función que reciba dos cadenas y devuelve 1 si la cadena 2 está dentro de la cadena 1 (si es subcadena) y 0 (cero) si no lo es.

114. Desarrolle una función que reciba una cadena y la muestre invertida por pantalla.

115. Desarrolle una función que reciba una cadena y la devuelva sin espacios (compactarla).

116. Desarrolle una función que reciba una cadena y devuelva un 1 si es palíndromo (palabra capicúa) o cero si no lo es.

117. Desarrolle una función que reciba una cadena y devuelva la cantidad de mayúsculas que contiene.

118. Desarrolle una función que reciba una cadena y convierta todas sus minúsculas en mayúsculas.

119. Se declara: char s[4]; y se ejecuta la siguiente instrucción: scanf("%s",s); se introduce por teclado: PEPITO GRILLO, e inmediatamente se ejecuta la instrucción puts(s); ¿Cuál de los siguientes literales se visualizará en la pantalla?
PEPI
PEPITO GRILLO
PEPITO
PEPIT
Ninguno, porque hay un error de sintaxis en alguna instrucción.

120. Construye un programa que permita ingresar una frase (hasta 256 caracteres) y resuelve:
- Contar la cantidad de letras.
- Contar la cantidad de palabras.
- Determinar cuál es la vocal que aparece con mayor frecuencia.
- Determinar cuántas veces aparece determinada letra, leída de teclado.
- Averiguar qué cantidad de letras tiene la palabra más larga.
- Leídas dos letras de teclado, se pide determinar la cantidad de veces que la primera letra precede a la segunda.

121. Escriba un programa que implemente una función que reciba una frase por teclado y la muestre ordenada por pantalla en forma inversa (si la frase es “Aguante Guns N Roses”, la salida seria “Roses N Guns Aguante”).

Ejercicios avanzados

122. Programar el juego TA-TE-TI. Para ello se requiere modelar un tablero de 3 x 3 en el que el jugador 1 ingresará una coordenada en la que guardará una ‘x’. Luego el jugador 2 ingresara otra coordenada en la que se guardara una ‘o’. Si la posición ya está ocupada, el programa deberá pedir el reingreso de otra coordenada. Una vez completados todos los casilleros (el programa de darse cuenta cuando esta condición se dé), el programa debe decir qué jugador gano la partida o si hubo empate (ningún jugador ganó).

123. Mc Donald necesita registrar los ingresos de sus 5 sucursales de Mar Del Plata por semana. Para ello necesita un programa que pueda cargar, en forma aleatoria, los ingresos por día de cada sucursal. Una vez cargados los datos, se pide:
- Calcular qué sucursal tuvo más ingresos en un día y cual fue ese día
- Calcular qué sucursal tuvo más ingresos en la semana
- Calcular el promedio de ventas por día en Mar Del Plata
- Calcular el promedio de ventas de Mc Donald en toda la semana laboral

124. La zapatería de barrio Alto Calzado necesita un programa para administrar su local. Para ello el programa debe cumplir las siguientes funciones:
a) cargar los datos de sus 5 vendedores (DNI, nombre, apellido, genero, pares Vendidos).
b) cargar los 10 tipos de calzado que venden (idCalzado, descripción, su stock remanente).
c) Al vender un par de zapatillas

Armar un programa con las estructuras necesarias para contabilizar las ventas hechas por los vendedores (ingresando DNI y idCalzado de la venta) y mantenga actualizado el stock remanente de pares de zapatillas.

125. La batalla naval es un juego que se basa en un tablero de 10 por 10 casilleros en el que se ubican 5 barcos: un portaviones (5 casilleros), 2 destructores (3 casilleros) y dos patrulleros (2 casilleros). Se pide modelar un programa en el:
   a) El jugador A ingrese la posición de sus 5 barcos ingresando los casilleros que ocupa cada uno. Se asumirá que el jugador será honesto e ingresará las posiciones contiguas que ocupa cada barco.
   b) Luego el jugador B ingresará coordenadas con la intensión de hundir los barcos. Tras cada ingreso, el juego deberá indicar si fue "agua" (marcando con un carácter 'o' en el tablero) o "golpe" (marcado con una 'x' en el tablero). En cada movimiento deberá mostrarse el tablero con todos los disparos realizados, pero obviamente sin mostrar los barcos.
   c) Al hundir todos los barcos, el juego deberá contabilizar la cantidad total de disparos y la cantidad de disparos fallidos. Con la intensión de que luego al invertir los roles, el jugador que menos disparos hizo es el ganador.

Ejercicios con estructuras

126. Definir un tipo de dato llamado `Fecha` compuesto por 3 campos llamados `día`, `mes` y `año`. Además defina y desarrolle sus operaciones `getDia()`, `getMes()`, `getAnio()` y `isBisiesto()`.

127. Dada la estructura `struct producto {int id; float precio};` desarrollar una función que reciba dicha estructura como puntero y aplique un descuento del 20% al precio.

128. Dada la estructura del ejercicio 2, desarrolle una función que reciba dos estructuras producto y devuelva aquella que tiene el precio más caro.

129. Dada la estructura del ejercicio 2, desarrolle la función `int getMenor(struct producto p[], int dim);` que devuelva la posición del producto que tenga menor precio.

130. Desarrollar una función que reciba únicamente una estructura por puntero y le cargue los valores que corresponda. La estructura es libre, pero debe cargar mínimo 2 enteros y un flotante.

131. Desarrolle una función que reciba 3 parámetros (de tipo a elección) y devuelva una estructura con esos valores cargados en sus campos.

132. Desarrollar un programa que lea registros de alumnos y procese sus notas. Se ingresarán 10 alumnos de los cuales se ingresa su legajo y 3 notas. Se pide calcular el promedio de las 3 notas y guardar el promedio de cada alumno para listarlo al final en la forma "legajo:promedio" y además mostrar el mayor promedio al final de ese listado resaltado.

    Defina una estructura alumno que tenga un campo `int legajo` y otro `enum` aprobado que deberá considerar los valores "`DESAPROBADO`", "`APROBADO PARCIAL`" y "`APROBADO`". Desarrolle un programa que pida 5 legajos por teclado y cargue en la estructura el estado de aprobación.

Ejercicios con archivos de texto

Importante: se asume que los archivos `.txt` son de texto y los archivos `.dat` son binarios.

133. Preguntas
   a) La función `fclose()`, utiliza el nombre externo de un archivo o la variable de este?
   b) Y la función `fprintf()`?
   c) ¿Cuál es la función que usa ambos nombres, el externo y el de la variable asociada?
   d) Siendo `arch` una variable `FILE`, explique la diferencia entre:
   ```
   arch = fopen("archivo.txt","rt");
   arch = fopen("archivo.txt","wt");
   ```
   e) La instrucción `arch = fopen("archivo.txt","wt");` presenta algún peligro potencial?

134. Se tiene el archivo `pro1.txt`. Escriba las instrucciones para abrirlo en modo lectura y cerrarlo.

135. Desarrolle una función `contarVocales` que reciba un archivo de texto ya abierto y devuelva la cantidad de vocales que hay en el archivo.

136. Desarrolle un programa que abra el archivo `cuento.txt` y lo muestre por pantalla.

137. Desarrolle una función `fLog()` que reciba una cadena y grabe en un archivo de texto `debug.log` el día, la hora y la cadena. La función valida si el archivo existe y lo crea si es necesario. Para trabajar el día y la hora, revisar este link.

138. Desarrolle una función que lea el archivo `config.txt` que tiene el siguiente contenido:
   ```
   Archivo de configuración
   Tamaño de arrays: 25
   Año: 2019
   Máximo de líneas: 150
   ```
   Y devuelva los datos leídos en un `struct`. La función **no debe** recibir datos por parámetro.

139. Desarrolle un programa que pida por teclado el número de legajo de un alumno y dos notas; que calcule el promedio y guarde toda la información en un archivo de texto `alumnos.txt`. El ingreso de datos finaliza con un legajo igual a `-1`.
140. Modifique el programa anterior para que lea el archivo `alumnos.txt` y lo muestre por pantalla.

141. Desarrolle la función `mostrarArchivoTokenizado()` que recibe un archivo de texto ya abierto y muestre por pantalla el contenido del archivo, una palabra por línea. NOTA: los separadores son todos los signos de puntuación. Luego impleméntela en un programa.

142. Basado en el ejercicio anterior, desarrolle la función `tokenizarArchivo()` que reciba un archivo de texto ya abierto y cree uno nuevo llamado `archTokenizado.txt` que contenga el contenido del primero pero con una palabra por línea.

143. Dado un archivo `promedios.txt` con la siguiente forma:
   ```
   11111  4.50
   11112  8.70
   33331  1.20
   ```
   Desarrolle un programa lea dicho archivo y guarde en otro nuevo `aprobados.txt` aquellas líneas cuyo promedio es mayor a `7`. El nuevo archivo debe contener el titulo `Los aprobados son`.

144. Escribe un programa para actualizar, mensualmente, el archivo de `sueldos.txt` de una empresa. Cada grupo de datos en el archivo contiene el nombre del empleado, el sueldo semanal, y finalmente los meses y años de antigüedad en la empresa. Al fin del mes, cuando se ejecuta el programa, se debe incrementar la antigüedad en meses (y en años si corresponde), y deben darse aumentos. Si el empleado tiene al menos 5 años, recibe un aumento del 30%, y si supera los 10 años, se le aumenta un 35% adicional.

145. Cada grupo de datos para el archivo `clientes.txt` contiene el nombre del cliente, dirección, género y saldo de la cuenta en un supermercado. Escribe un programa que:
   a) Cree dos archivos separados, `mujer.txt` y `varon.txt` con todos los datos de los clientes almacenados en el archivo correcto.
   b) Muestre en pantalla la siguiente información:
      i) Número total de clientes
      ii) Número total de varones
      iii) Número total de mujeres
      iv) Promedio de saldos para hombres
      v) Promedio de saldos para mujeres

146. Desarrolle un programa que pida por teclado un legajo, nombre, edad guarde la información en un archivo de texto con la siguiente forma:
Registro #`unNum`
Legajo: `unLegajo`
Nombre: `unNombre`
Edad: `unaEdad`
`Línea vacía`
El ingreso de datos se finaliza con un legajo igual a cero. Los datos ingresados se guardan en el archivo `datos.txt`.
Luego desarrollar otra función que lea dicho archivo y lo muestre por pantalla con el formato `unNum-unLegajo|unNombre|unaEdad`. Implementar la función al final del programa para que muestre el contenido del archivo.

Ejercicios con archivos binarios

147. Escribe la declaración del tipo estructurado que se guardará en un archivo de datos de los socios del club "San Antonio de Padua". Cada registro contendrá los siguientes campos (elementos de la estructura): nombre, género, edad, fecha último pago de abono.

148. Para el archivo del ejercicio anterior realiza un programa que permita crear el archivo y guardar registros. Luego, construye una función `mostrarData()` que abra el archivo para lectura, muestre sus registros en pantalla y cierre el archivo.

149. Desarrolle un programa que genere un archivo binario `personas.dat` cuya estructura interna sea: `identificador(int)`, `nombre(15)`, `apellido(15)`, `genero(char)`, `fechaNacimiento(int DDMMAAAA)`, `localidad(15)`, `salario(float)`. Una vez creado el archivo, el programa debe mostrarlo por pantalla. Utilizaremos este archivo en ejercicios siguientes.

150. Utilizando el archivo `personas.dat` del ejercicio 149, desarrolle una función que reciba el archivo abierto y una posición, y devuelva el registro que se encuentra en dicha posición.

151. Utilizando el archivo `personas.dat` del ejercicio 149, desarrolle una función que reciba el archivo abierto y devuelva el ultimo registro del archivo.

152. Utilizando el archivo `personas.dat` del ejercicio 149, desarrolle una función que reciba el archivo abierto y devuelva los últimos 5 registros del archivo.

153. Utilizando el archivo `personas.dat` del ejercicio 149, desarrolle una función que reciba el archivo abierto y una posición y devuelva el registro de dicha posición y los 5 registros siguientes.

154. Utilizando el archivo `personas.dat` del ejercicio 149, desarrolle un programa (con funciones) que guarde en el archivo `adultos.dat` a todos los mayores de edad. NOTA: para saber la fecha actual, ingresarla por teclado como un int.

155. Utilizando el archivo `personas.dat` del ejercicio 149, desarrolle un programa que incremente el salario de cada persona en un 10% (modificando el archivo original).

156. Utilizando el archivo `personas.dat` del ejercicio 149, desarrolle una función devuelva el tamaño del archivo en bytes.

157. Utilizando el archivo `personas.dat` del ejercicio 149, desarrolle una función devuelva la cantidad de registros en el archivo.

158. El archivo binario `estudia.dat`, contiene registros con los siguientes datos de los estudiantes de una universidad:
    - Nombre
    - Número de Legajo
    - Cantidad de materias aprobadas
    - Promedio de calificaciones

    El archivo se encuentra ordenado en orden ascendente por número de legajo. El programa debe presentar un menú con las siguientes opciones:
    - Mostrar todos los datos de un estudiante, cuyo nro. de legajo se ingresa por teclado.
    - Mostrar todos los datos de los estudiantes cuyo promedio sea mayor o igual al ingresado por el usuario, y además que tengan no menos de 10 materias aprobadas.

    NOTA: Previamente deberá realizar un programa auxiliar para la creación del archivo de estudiantes.

Ejercicios integradores

159. Explica las siguientes funciones, aportando ejemplos de cada una: `feof()`, `fscanf()`, `fseek()`y `fread()`.

160. Dado un archivo de texto, escribir un programa que cuente el número de palabras que aparezcan en dicho archivo.

161. Escribe un programa que copie en un archivo otros dos archivos, uno a continuación de otro.

162. Se tiene el archivo binario `acciones.dat` cuya estructura de registros es `idAccion(int)`, `valor(float)`. Se pide un programa que implemente una función que reciba el archivo abierto y devuelva un vector con los precios máximos y mínimos de cada acción.

163. Tenemos un archivo de registros cuya estructura están definidas según la declaración:

```
struct registro1{
      char partido[60];
      char localidad[60];
      int candidatos;
      }
```

Se quiere sustituir la estructura del registro por el siguiente:

```
struct registro2{
      char partido[60];
      char localidad[60];
      int candidatos;
      int elegidos;
      }
```

Escribe un programa que modifique la estructura y traslade toda la información del archivo primitivo al nuevo, inicializando a cero el nuevo campo creado.

164. Dada una palabra ingresada por teclado, el programa debe informar si esta palabra se encuentra escondida en el texto del archivo `cuento.txt`. Una palabra esta escondida cuando sus letras se encuentran en el texto (no necesariamente consecutivamente), por ejemplo, la palabra "elefante" está escondida en la frase "**el** h**e**rmano de **F**r**an**cisco **t**uvo su**e**rte" (ver negritas). De encontrarse, debe informarse la cantidad de caracteres entre los que se encuentra escondida.

165. Dado el archivo abreviaturas.txt cuyo formato es `palabra:abreviatura`, desarrollar un programa que lea el contenido del archivo `cuento.txt` y genere uno nuevo `cuentoAbreviado.txt` en el que se reemplacen las palabras que pueden ser abreviadas.

166. Dado un valor entero por teclado, elaborar un programa que busque en un archivo (`numeros.dat`) de enteros ese valor, y en caso de encontrarlo, que diga en qué posición o posiciones del archivo se encuentra.

167. Se define el tipo siguiente:

```
struct alumno {
      char apellidos[60];
      char nombre[20];
      float notajunio,notasept,notaprac;
      }
```

En un archivo con datos de esta estructura disponemos de los alumnos matriculados en una asignatura. Se quiere obtener un archivo de texto en el que aparezca en cada línea el nombre de un alumno con el formato apellidos, nombre, nota junio.

168. En un archivo se tiene grabada una sucesión de números enteros positivos o nulos, correspondientes a las puntuaciones de varios jueces sobre un mismo ejercicio. Escriba un algoritmo para calcular el promedio de los valores estrictamente positivos. También debe calcular la cantidad de ceros.

169. Escribe un programa que permita llevar los gastos e ingresos menores a $25000 de una empresa. El programa presentará un menú como el siguiente:

```
CONTROL DE GASTOS
==================
1. Ingresos
2. Gastos
3. Salida
==================
```

Donde las funciones realizan las siguientes operaciones:

- Menú muestra el menú en la pantalla.

- lee_opcion lee la opción del teclado.
- Ingresos lee nuevos ingresos y los suma al saldo actual.
- Gastos lee nuevos gastos y los resta del saldo actual (que nunca podrá ser negativo).
- Salida muestra el saldo actual (ingresos - gastos).

170. Dados dos archivos binarios y secuenciales: “A.DAT” y “B.DAT” con el siguiente formato de registros:

```
struct Treg {
      int Clave: integer;
      char Dato[10];
      }
```

Ambos archivos están clasificados en forma ascendente por el campo Clave y dentro de un mismo archivo no hay claves duplicadas.
Escribe cada una de las siguientes funciones:

- Unión: se genera un archivo C resultado de la unión entre A y B, es decir el archivo C contendrá todos los registros consignados en los archivos de entrada. En caso de existir claves repetidas en el Archivo A y el B, se grabará un solo registro en C utilizando los datos del archivo B.
- Intersección: se generará el archivo C conteniendo solo los registros del Archivo A cuyas claves también aparezcan en el Archivo B.
- Diferencia: el archivo C tendrá los registros que estén en el primer archivo que se pase como parámetro y cuyas claves no aparezcan en el archivo que se pase como segundo parámetro.

171. Un club cuenta con un archivo maestro de acceso secuencial denominado `socios.dat`. Este archivo se encuentra ordenado por código de socio y tiene los siguientes campos: `Código de socio`, `Nombre` y `apellido`, `Dirección`, `Deuda` (array de 12 posiciones, cada posición contendrá la deuda correspondiente a ese mes).
Se cuenta además con un archivo `bajas.dat` de acceso secuencial ordenado, con los códigos de socio a ser dados de baja. Se desea:

- Actualizar el archivo SOCIOS con el de BAJAS.
- Emitir un listado ordenado por código de socio con: código de socio, nombre y apellido, dirección y total adeudado de aquellos socios que hayan sido dados de baja.

Se supone que no hay duplicados en el archivo Socios, pero si puede haberlo en el de Bajas.

Se cuenta además con un archivo `pagos.dat` de acceso secuencial, con los datos de los pagos realizados por los socios. Este archivo tiene registros con la siguiente estructura:

- Código de socio
- Mes
- Monto abonado

Este archivo está ordenado en forma ascendente por código de socio.
Se desea:

- Actualizar el archivo SOCIOS con la información del archivo Pagos.
- Emitir un listado ordenado por código de socio con: código de socio, nombre y apellido, total adeudado con aquellos socios que mantengan deudas con el club.

Se supone que no hay duplicados en el archivo Socios, pero si puede haber varios registros para un mismo socio en el archivo de pagos.

172. Dados dos archivos binarios `a.dat` (acceso directo y secuencial) y `b.dat` (acceso secuencial) con el siguiente formato de registros:

```
struct Treg {
   int Clave: integer;
```

```
char Dato[10];
}
```

Se desea realizar un algoritmo que actualice los datos del archivo A con los del B. Teniendo en cuenta que la clave en A coincide con el número de registro. Toda clave en B estará en A.

173. Desarrollar un algoritmo que permita la actualización de un archivo de solicitudes de planes de ahorro para compra de automóviles. Se cuenta con:

- Un archivo secuencial ordenado en forma ascendente por el campo `nroSolicitud` donde están registradas todas las solicitudes vigentes. Los registros de este archivo tienen la siguiente estructura:

```
registroSolicitud:
      nroSolicitud: entero
      titular: cadena de 40 caracteres
      marca: cadena de 20 caracteres (12 marcas diferentes)
      modelo: carácter (válidos: A,B,C,D)
      color: entero (1 a 8)
```

- Un archivo también secuencial que informa las bajas a procesar. Este archivo está ordenado en forma ascendente por el campo nroSolicitudBaja y sus registros tienen la siguiente estructura:

```
registroBaja
      nroSolicitudBaja: entero
      causa: carácter (R,F,O)
```

Se quiere generar un nuevo archivo de solicitudes resultado de efectuar las bajas correspondientes. (Este archivo tendrá la misma estructura que el archivo de solicitudes.)

Se podrán rechazar bajas por causa inválida, o baja a registro inexistente. Las bajas rechazadas serán listadas indicando el porqué del rechazo.

Al final del proceso se emitirá un informe indicando para cada marca, la cantidad de solicitudes existentes por modelo y color.

También al final del proceso se indicará la cantidad de registros leídos en cada uno de los archivos de entrada, la cantidad de registros grabados en el archivo actualizado, la cantidad de bajas rechazadas y listadas y la/s marca/s con mayor cantidad de solicitudes vigentes (en el actualizado).