Práctica – Lenguaje C

PARTE I – INTRODUCCION - OPERADORES – SALIDA POR PANTALLA

EJ. 0 – Transcribir en el entorno de desarrollo Code::Blocks el siguiente código literalmente, luego compilarlo:

```
#include <stdio.h>

void main()
Begin
    integer x, y = 9;
    x := 8;
    print('Bienvenido a Taller de Programacion 1\n');
    printf(" Hoy es x %d / y %d \n x+y = %d", x ,y, x+y);
    return 0;
End;
```

Corregir los errores sintácticos detectados. Ejecutar el programa y revisar resultados, corregir lo que considere necesario para que la salida sea la adecuada.

Ej. 1 – Completar la siguiente tabla para los tipos de datos en C:

Tipos de datos de C	Descripción	Equivalente en Pascal	Ejemplos de constantes
char			
float			
int			
long			
char *			

Ej. 2 – Sean las siguientes declaraciones de variables e inicializaciones. Completar el cuadro con la salida en cada caso, agregar salto de línea, tabulaciones o espacios que considere para que la visualización de los resultados sea legible:

```
int A = 34, B =1234, C = -3;
char ch1 ='A';
float F = 22.47;
```

`printf("A=%d",A);`	
`printf("A=%5d",A);`	
`printf("A=%4d",A);`	
`printf("A=%-5d",A);`	
`printf("A=%-4d",A);`	
`printf("A=%05d",A);`	
`printf("B=%-05d",A);`	
`printf("C=%+d",A);`	
`printf("B=%3d",B);`	
`printf("C=%c",C);`	
`printf("C=%3c",C);`	
`printf("C=%-3c",C);`	
`printf("F=%f",F);`	
`printf("F=%7.2f",F);`	

`printf("F=%07.2f",F);`	
`printf("F=%-7.2f",F);`	
`printf("F=%7.1f",F);`	
`printf("%c",ch1);`	
`printf("%3c",ch1);`	
`printf("%s",”ELEFANTE”);`	

Ej. 3 – Reescribir las siguientes sentencias según el ejemplo:

Sentencia	Equivale a
`n--;`	`n = n – 1;`
`n += 4;`	
`n -= a*2;`	
`n *= 3+2;`	

PARTE II – LECTURA POR TECLADO – ESTRUCTURAS DE DECISIÓN E ITERACIÓN

Ej. 4 – Desarrollar una macro que devuelva cada uno de los siguientes resultados:

a) el cuadrado de un número.

b) la suma de dos números.

c) el mínimo de dos números.

Ej. 5 – Dadas las declaraciones siguientes, completar la función ***scanf()*** con la expresión más apropiada

```
char x, cadena[25];
int n;
scanf("%d",……);
scanf("%c",……);
scanf("%s",……);
```

Ej. 6 – Indicar la salida de los siguientes programas, sin ejecutarlos:

a)

```
#include <stdio.h>
void main()  {
     int res, a=10 , b=4;
     res = 9 - b * b + a++;
     if ( res % 2 == 0 )
          printf ("El resultado %d es par. \n", res);
       else
          printf ("El resultado %d es impar. \n", res);
     printf ("El valor de a es %d. \n", a);
}
```

b)

```
#include <stdio.h>
int main(){
     int a = 4, b = 6;
      a -= b++ * 2;
     printf("a =  %d\t b= %d\n", a, b);
   return 0;
}
```

Ej. 7 –

a) Reescribir las siguientes sentencias a su equivalente con *if*:

- `x = ( y<z ) ? y : z ;`
- `printf ("%s", ( i%2 == 0 ) ? "es par" : "es impar" );`

b) Reescribir la siguiente sentencia utilizando el operador condicional:

```
if (x > 0)
    y = 1;
else
    y = 0;
```

Ej. 8 – Indicar la salida de los siguientes fragmentos de código, sin ejecutarlos:

```
#define n 5
 int suma, i;
```

a)
```
suma = 0;
   for (i=1; i<=n; i ++)
      suma += i;
   printf("%d\n", suma);
```

b)
```
suma = 0;
    for (i=1; i<n; i+=2 )
       suma += i;
    printf("%d -> %d\n", i,suma);
```

c)
```
for (suma = 1,i=n; i>0; i-- ) {
      suma *= i;
      printf("%d -> %d\n", i, suma);
}
```

Ej. 9 – Desarrollar programas para:

a) ingresar N números enteros y calcular y mostrar el promedio de los positivos.

b) ingresar N caracteres y determinar y mostrar la cantidad de vocales indicando si la A es la vocal que más veces apareció.

c) mostrar el valor medio de dos enteros leídos

d) leer N caracteres y hallar y escribir la cantidad de letras mayúsculas ingresadas.

e) leer 3 valores correspondientes a un ángulo en grados, minutos y segundos y obtener y mostrar el valor equivalente en radianes

f) Ingresar tres letras y escribirlas ordenadas alfabéticamente. Resolver este problema de dos formas diferentes. (Utilizando estructuras de decisión independientes y estructuras de decisión anidadas).

g) Leer a, b y c coeficientes de una ecuación cuadrática (ax^2+bx+c). Determinar si tiene raíces reales o imaginarias ($b^2-4ac < 0$).

h) Informar el importe de un pasaje aéreo teniendo en cuenta que el mismo depende de la clase ($100000-turista y $200000-bussines) y recibe un descuento del 10% si viaja la primera semana del mes. Ingresar la clase (T ó B) y el día (1..31) indicar el importe del pasaje

i) Determinar si una persona es niño, adolescente, adulto joven o adulto, según la edad ingresada por el usuario. Niño se es hasta los 11, adolescente hasta los 17 y adulto joven hasta los 25. [Nota: todas las edades dadas se incluyen en su categoría].

Ej. 10 – Transformar las siguientes sentencias utilizando la instrucción **switch** (x es int, F es char).

a)
```
if (x == 4)
   y = 7;
else
    if (x == 5)
       y = 9;
    else
       if(x == 9)
          y = 14;
       else
          y = 22;
```

b)
```
if (F == 's' || F == 'S')
     r = x + y;
else
   ff (F == 'r' || F == 'R')
       r = x - y;
   else
       if (F == 'm' || F == 'M')
           r = x * y;
       else
          if (F == 'd' || F == 'D')
              y = x/y;
         else
              y = 0;
```

Ej. 11 – Detectar y corregir los errores de los siguientes fragmentos de código:

a) mostrar los números naturales divisibles por 5 y menores a 25
```
#define N 50
int i;
     for(i = 5; i <= N; i+=5);
          printf("%d", &i);
```

b) intercambiar dos enteros
```
void InterCambia (int *a, int *b){
  int aux;
  aux = *a;
  a = *b;
  b = &aux;
}
```

c) retornar el mayor valor de un arreglo de N valores naturales
```
int Mayor (int V[], int N){
  int i=0, max=0;
  while (i<N)
       if (V[i] > max)
          return V[i];
       else
           i++;
}
```

PARTE III – FUNCIONES – PUNTEROS.

Ej. 12 – Resolver nuevamente los incisos del **ej. 8** desarrollando funciones.

Ej. 13 – Para N personas se registró el peso y la altura, calcular sus índices de masa corporal y determinar e informar el porcentaje de los que tienen un índice normal (18 <= ICM <= 21). Implementar y utilizar función IMC correctamente parametrizada que devuelva verdadero si el índice es normal y falso en caso contrario

Ej. 14 – Escribir un programa que funcione como una calculadora que realice operaciones de suma, resta, multiplicación y división. Además, permitir elevar un número a una potencia entera ingresada por el usuario utilizando funciones para cada operación.

Ej. 15 – Desarrollar un programa que convierta una temperatura de grados Celsius a Fahrenheit o viceversa, según la elección del usuario. [NOTA: La conversión se realiza mediante la siguiente fórmula: ºF = ºC x 1.8 + 32]

Ej. 16 – Escribe una función llamada **esPalindromo** que tome una cadena como parámetro y determine si es un palíndroma (se lee igual de izquierda a derecha que de derecha a izquierda). Utiliza esta función para verificar si una palabra ingresada por el usuario lo es o no.

Ej. 17 – Escribe una función llamada **esCapicua** que tome un entero como parámetro y determine si es o no capicúa.

Ej. 18 – Desarrollar un programa que imprima el equivalente en letras de un número introducido de un rango de 0 a 999.
Por ejemplo: se introdujo **119**, imprime **ciento diecinueve**.

Ej. 19 – Dadas las siguientes declaraciones:

```
float  *pf1, *pf2, *pf3, a1, a2;
char *pch1, *pch2, ch1, ch2;
```

Indicar cuáles de las siguientes sentencias son correctas y cuáles no. Justificar.

a) `*pf1 = *pf2 + 1;`
b) `pf1 += pf2;`
c) `printf("%f", p1);`
d) `a1 = &pf1;`
e) `pf2  = a2;`
f) `pf2  = NULL;`
g) `*pf3 = a2 * 2;`
h) `*pf1 = *pf3;`
i) `pf3 = pf2;`
j) `scanf("%c",pch2);`
k) `pch2 = 'a';`
l) `pch = &ch1;`

Ej. 20 – Corregir el siguiente programa para retornar la suma y la diferencia de los valores de las variables a y b declaradas en el main().

```
#include <stdio.h>
  int main(){
    int a = 30, b = 20;
    sumaresta(a, b, s, r);
    printf(" valor de a+b es %d\t valor de a-b es %d\n", &s, &r);
    return 0;
  }
  void sumaresta(int x, int y, int *s, int *r){
      s = a+b;
      r = a-b;
  }
```

PARTE IV – ARCHIVOS TEXTO - CADENAS DE CARACTERES – AREGLOS - MATRICES

Ej. 21 – Dado un vector de enteros de n elementos, desarrollar una función que encuentre y devuelva la posición del máximo. Implementar 3 veces utilizando for, while y do while.

Ej. 22 – Desarrollar programas modularizados para:

a) Leer 10 cadenas de caracteres por teclado (pueden tener espacios) y grabarlas en un archivo de texto CADENAS.TXT (una por línea). Desde el archivo creado, mostrar la cadena más larga.

b) Leer un archivo de texto NUMEROS.TXT que contiene un número entero en cada línea y almacenar los números pares es un vector y los impares en otro ignorando los 0.
Luego regrabar el archivo con los pares primero y los impares después.
NOTA: Suponer que en el archivo hay como máximo 50 elementos

c) Leer de un archivo de texto FRAC.TXT que contiene dos fracciones por línea: n1, d1, n2, d2. Para cada línea obtener y mostrar la suma de ambas fracciones, como número real con tres decimales. Si el resultado es un número entero, mostrarlo como tal.
Ejemplos: 2 3 1 2 resultado 1.166 ; 4 3 2 3 resultado 2 ; 5 12 1 18 resultado 0.472

Ej. 23 – Enumerar los valores de todos los componentes de los siguientes arreglos. Indicar cuáles arreglos de caracteres podrían ser utilizados correctamente como cadenas.

a) `int v1[4] = {0};`
b) `int v2[1]={6};`
c) `int v3[] = {2,4,6};`
d) `char s1[4] = {'h','o','y'};`
e) `char s2[] = {'h','o','y'};`
f) `char s3[4]= {'h','o','y','\0'};`

Ej. 24– Sea A un vector de N elementos enteros, escribir funciones que permitan:
- **a)** hallar el promedio de los elementos positivos (función float).
- **b)** hallar el mínimo de los impares (función int)
- **c)** hallar el máximo de las posiciones pares (función void)
- **d)** generar un vector B con los elementos de A que sean mayores a K (K debe leerse)

Escribir un programa que lea N y un vector N elementos desde un archivo de texto DATOS.TXT, y luego mediante un menú repetitivo invoque a la función correspondiente (desarrolladas de a) a d)) mostrando el resultado.

Ej. 25 – Desarrollar una función que pase la primera letra a mayúscula de cada palabra de una cadena de caracteres.
Por ejemplo "**calle jujuy 1085 6 a"** pasa a **"Calle Jujuy 1085 6 A".**

Ej. 26 – Sea A una matriz de NxM elementos reales, escribir funciones que permitan:
- **a)** hallar el promedio de los elementos positivos (función float).
- **b)** hallar el mínimo de las columnas impares (función int)
- **c)** hallar el máximo de las filas pares (función void)
- **d)** generar un archivo de texto que contengas una línea por cada fila de la matriz que tienen todos los valores pares, colocando en la línea NROFILA SUMPARES

Escribir un programa que lea N, M y la matriz desde un archivo de texto DATOS.TXT, y luego mediante un menú repetitivo invoque a la función correspondiente (desarrolladas de a) a d)) mostrando el resultado.

Ej. 27 – En una escuela secundaria para el último año se tiene un archivo de texto NOTAS.TXT con las notas finales de las 10 materias, el formato una línea por alumno con Número de matrícula (0 a 99) y las 10 notas finales separadas por espacios (pueden no estar los 100 alumnos ni vienen ordenados)
Desarrollar una función para generar una matriz de 100x10 con las notas finales donde las filas representan el número de matrícula y las columnas las materias (las filas que no representen alumnos pues no se encuentran los datos deben quedar en 0) y luego desarrollar:
- **a)** una función (int) que devuelva la cantidad de alumnos recibidos (todas las notas más de 4)
- **b)** una función (int) que devuelva la cantidad de alumnos que deben rendir más de 2 materias (deben rendir aquellas en las que no tienen al menos 4).
- **c)** una función (void) que devuelva en dos parámetros el alumno recibido con mayor nota promedio (número de matrícula y nota promedio final)

PARTE V – REGISTROS – TIPO PUNTERO – MEMORIA DINÁMICA

Ej. 28 – Desarrollar un programa que contenga una función *void* que reciba una cadena que representa una fecha de la forma aaaammdd y devuelva un struct con 3 campos: dd, mm y aaaa. La lectura de la cadena y la impresión del registro deben estar en el *main().*

Ej. 29 – Desarrollar un programa correctamente modularizado que solicite por teclado N y los siguientes datos de N alumnos: Nombre (cadena de 30), NumeroMatricula (entero), Carrera (cadena de 30) y almacene los datos de los alumnos en un arreglo de structs. Luego, muestre por pantalla a partir de los datos ingresados (con un alumno por línea) solo aquellos que estudian Ingeniería en Informática.

Ej. 30 – Escribir un programa que cree dinámicamente 3 variables enteras y ponga en ellas 3 valores leídos por teclado, calcule y muestre su suma y producto. Finalmente libere la memoria reservada.

Ej. 31 – Escribir un programa que cree un arreglo estático de punteros a enteros, y luego cargue en él N enteros (N y los enteros se encuentran en un archivo de texto). Mostrar aquellos que sean positivos. Al finalizar, liberar la memoria solicitada en tiempo de ejecución.

Ej. 32 –

a) Desarrollar un programa que lea los datos de dos personas, en variables dinámicas (tipo struct), y muestre el nombre del más joven. Utilizar los siguientes tipos que deben estar en el archivo tipos.h:

```
typedef struct {
   char Nombre[21];
   int  Edad;
 } strPersona;
typedef strPersona * ptPersona;
```

b.- Modificar el tipo strPersona del inciso a.- agregando un campo sig de tipo puntero a strPersona. Enlazar pt1 con pt2, haciendo que pt2 sea el siguiente de pt1 y que pt2 no tenga siguiente.

c.- Dados los datos cargados en a.- y los enlaces efectuados en b.-, indicar si son correctas o no las siguientes sentencias y en caso de ser correctas indicar el efecto que producen:

i) `printf("%d", pt1.Edad);`
ii) `printf("%s", pt1->sig->Nombre);`
iii) `pt1->sig = pt1;`
iv) `pt2->sig = pt1->sig;`
v) `pt1->Nombre = pt1->sig->Nombre;`

PARTE VI – ARCHIVOS BINARIOS

Ej. 33 – Desarrollar un programa que grabe en un archivo binario NUMEN.DAT números enteros ingresados por teclado, finaliza el ingreso cuando aparecen dos números consecutivos iguales (esos datos van al archivo). Mostrar el archivo generado.

Ej. 34 – Dado un archivo binario DATOS.DAT en el que cada registro es de 17 dígitos para tres campos, 5 iniciales de ciudad, 5 superficie y 7 cantidad de habitantes. Escribir un programa que por pantalla muestre la ciudad con más densidad de habitantes y la de menor densidad.

Ej. 35 – Modifique el programa anterior para generar un vector de registros con las 10 primeras ciudades del archivo. Luego encuentre y muestra las iniciales de las ciudades que tengan la mayor y la menor de densidad poblacional entre esas 10.

Ej. 36 – Una empresa distribuidora de repuestos de automotores tiene un archivo binario PENDIFACTU.DAT con las facturas pendientes de cobro, con el siguiente formato de registro.

```
 typedef struct {
int NroFactura;  //ordenado por este criterio
float Precio;   //Precio unitario del articulo
} RegistroFactu;
```

Las facturas ya cobradas están registradas en otro archivo binario COBROS.DAT de enteros (ordenado) que representa el número de factura (podría haber facturas erróneas)
Crear un archivo PeENDIFACTUNUEVO.DAT con la misma estructura de PENDIFACTU.DAT tal que solo queden las facturas no cobradas, informar finalmente el importe total cobrado, el número de factura con mayor importe no cobrada y la cantidad de facturas erróneas que había en COBROS.DAT

Ej. 37 – Dado un archivo de texto que representa la facturación diaria de una empresa. Hay dos tipos de líneas, la de cabecera (una por factura y es la primera), y las de detalle que están a continuación y representan los artículos de la factura. La cabecera empieza con la letra **F** seguido 12 dígitos del número de factura y 30 caracteres el nombre del cliente.
La de detalle empieza con la letra **D** seguido de 6 dígitos código de artículo, 20 caracteres de descripción de artículo, 7 dígitos para el precio los dos últimos son decimales, y 5 dígitos de cantidad.
Desarrollar un programa que lea el archivo de facturación y genere un archivo binario que por cada factura genere un registro con número de factura, cantidad de ítems e importe total.