Problemario para CI-2126: Computación II
(versión 2)

Rosseline Rodríguez

Departamento de Computación y
Tecnología de la Información
Universidad Simón Bolívar

REPORTE INTERNO

Tabla de Contenido

ALCANCE DE VARIABLES. LOCALIDAD Y GLOBALIDAD

1.- En el siguiente programa usted debe verificar que las declaraciones de las variables son correctas, luego hacer la corrida del programa con el objeto de visualizar el alcance de las variables.

```
/* archivo p1 */
int k,x;

void r1(int a, int b)
{
      int i;
      i=0;
      a=2*b;
      r3(i);
      printf("%d %d,x,y);
}

main()
{
      k=1;
      x=2;
      r1(x,x);
      printf("%d",x);
}
```

```
/*archivo p2*/
int r2(int i)
{
      i=i+3;
      if (k <= 2)
            printf("%d %d",i,k)
      return(i+k+a);
}

void r3(int a)
{
      int x,y;
      a=i+k;
      x=r2(a);
      y=r2(2*a);
      printf ("%d %d",x,y);
}
```

2.- A continuación se presentan varios programas ilustrativos de lo que ocurre al combinar las reglas de alcance de variables y pasaje de parámetros en una forma no muy organizada. Su actividad concreta es dar los resultados de la corrida de estos programas atendiendo a dichas reglas. Escriba de nuevo los programas usando reglas de buen estilo de programación

```
a)
void pr(int x,int y,int *z);
main() {
      int a=5,b=8,c=3;
      pr(a,b,&c);pr(7,a+b+c,&a); pr(a*b,a /b,&c);
}
void pr(int x,int y,int *z)
{*z=x+y+*z; printf("%d%d%d",x,y,*z);}
b)
void r1(int *a; int b);
int r2(int i,int a,int k);
void r3(int *a, int i, int k);

main()
{
      int k=1,x=2;
      r1(&x,x,k);
      printf(“%d”,x);
}

void r1(int *a, int b,int y)
{
      int i=0;

      a=2*b;
     r3(&i,a,y);
      printf(“%d %d”,x,y);
}
```

```
c) main() {
      int i,j,k;
      i=0; j=1; k=2;
p2(0,&k);p2(1,&i); p2(2,&j);
}

void p1(int *i,int j,int k)
{     (*i)++;
      printf("%d %d %d",*i,j,k);
}

void p2(int h,int *j,int k) {
      int i=j;
      if (h==0)   p1(*j,j,k)
      else if (h==1)    p1(&i,*j,k);
                  else p3(&i,*j,k);
      printf("%d %d %d",i,j,k);
}
void p3(int *i)
{     (*i)++;}
```

3.- ¿Es este un programa correcto en C?, de no serlo señales los errores e indique posibles correcciones. Si fuera correcto, entonces haga la corrida del programa

```
main ()
{
      int f=2,h=10;
      a(f,h,'1'); a(h,f,'2');
}
void int A(int *x,int *y,char s)
{
      int b,c;

      b=x;
      if (G(b) >= 40) c=D(-5);
      else  c=D(b*2);
}
void D int p
{
f=f+1; p=p+f; printf(f);
}
void G(int *r)
{
      int a,f;
      a=10; f=a*a;
r=r*a; printf(r);
return(r);
}
```

APUNTADORES Y ARREGLOS

1.- Conteste cada una de las siguientes preguntas. Cada parte del ejercicio deberá utilizar los resultados de las partes anteriores donde sea apropiado:

a) Declare un arreglo de tipo `float`, llamado `numbers` con 10 elementos, e inicialice los elementos a los valores 0.0, 1.1, 2.2,...9.9. Suponga que la constante simbólica `SIZE` ha sido definida como 10.
b) Declare un apuntador `nPtr`, que apunte a un objeto de tipo `float`.
c) Imprima los elementos del arreglo `numbers`, utilizando notación de subíndice de arreglos. Utilice una estructura `for`, y suponga que se ha declarado la variable de control entera `i`.
d) Proporcione dos instrucciones por separado, que asignen la dirección inicial del arreglo `numbers` a la variable de apuntador `nPtr`.
e) Imprima los elementos del arreglo `numbers`, utilizando únicamente la variable `nPtr`.
f) Imprima los elementos del arreglo `numbers`, utilizando el nombre del arreglo como un apuntador.
g) Imprima los elementos del arreglo `numbers` mediante subíndices del apuntador `nPtr`.
h) Suponiendo que `nPtr` apunta al principio del arreglo `numbers`, ¿cuál es la dirección referenciada por `nPtr+8` ? ¿ cuál es el valor almacenado en esa posición?
i) Suponiendo que `nPtr` apunta a `numbers[5]`, ¿qué dirección es referenciada por `nPtr=nPtr-4` ? ¿cuál es el valor almacenado en esta posición?

2.- Encuentre el error en cada uno de los segmentos de programas siguientes. Suponga:

```
int *zPtr;
int *aPtr = NULL;
void *sPtr = NULL;
int number,i;
int z[5] = {1,2,3,4,5};
sPtr=z;

++z;
number = zPtr;    /* uso del apuntador para obtener el
                        primer valor del arreglo */
number = *zPtr[2]; /* asigna el valor 3 a number*/
number = *sPtr;/* asigna el valor apuntado por sPtr a number */
for (i=0;i<=5;i++) /* imprime el arreglo z */
      printf(“%d”,zPtr[i]);
```

3.- Para cada uno de los siguientes enunciados, escriba el trozo de código correspondiente. Suponga que se han declarado las variables **`num1`** y **`num2`** de tipo flotante.

a) Declare la variable **`fPtr`** como apuntador a un flotante
b) Asigne la dirección de la variable **`num1`** a la variable **`fPtr`**
c) Imprima el valor del objeto señalado por **`fPtr`**
d) Asigne el valor del objeto señalado por **`fPtr`** a la variable **`num2`**
e) Imprima el valor de **`num2`**
f) Imprima la dirección de **`num1`**. Utilice el especificador de conversión `%p`
g) Imprima la dirección almacenada en **`fPtr`** utilizando `%p` ¿Es el valor impreso el mismo de la dirección de **`num1`**?

4.- Realice la corrida del siguiente programa para observar los cuatro métodos para referenciar los elementos de un arreglo. Diga la diferencia entre los cuatro métodos:

```
#include <stdio.h>
main()
{
   int i, b[]={10,20,30,40,50};
   int *bPtr = b;

   printf("Arreglo impreso con notación de subíndices");
   for (i=0; i<=4; i++)
      printf("b[%d] = %d \n",i,b[i]);
```

```
   printf("\n");

   printf("Arreglo como apuntador y desplazamiento");
   for (i=0; i<=4; i++)
      printf("*(b+%d) = %d \n",i,*(b+i));
   printf("\n");

   printf("Notación apuntador-subíndices");
   for (i=0; i<=4; i++)
      printf("bPtr[%d] = %d \n",i,bPtr[i]);
   printf("\n");

   printf("Notación apuntador y desplazamiento");
   for (i=0; i<=4; i++)
      printf("*(bPtr+%d) = %d \n",i,*(bPtr+i));
}
```

5.- Para el siguiente programa, realice la corrida observando lo que ocurre en la pila de invocación. Modifique el programa para que los parámetros de la función sean todos por referencia. Realice la corrida con el programa modificado:

```
void pr(int x,int y,int z);

main() {
   int a=5,b=8,c=3;
   pr(a,b,c);
   printf("%d%d%d",a,b,c);
   pr(b,c,a);
   printf("%d%d%d",a,b,c);
   pr(c,b,a);
   printf("%d%d%d",a,b,c);
}
void pr(int x,int y,int z);
{
   z=x+y+z;
}
```

6.- Codifique en lenguaje C un extracto de código que dado un arreglo de enteros cuente por cada elemento o número del 0 al 9, cuantas veces se repite el elemento dentro del arreglo. Por ejemplo:
Entrada → Arreglo = [2,3,1,9,2,8,9,9,2,1,2]
Salida → 1 se repite 2 veces
2 se repite 4 veces
3 se repite 1 vez
9 se repite 3 veces
8 se repite 1 vez
los demás digitos aparecen 0 veces.

7.- El *modo* de un arreglo de números es el número m en el arreglo que se repite con mayor frecuencia. Si se repite mas de un número con frecuencias iguales, no existe un *modo*. Escriba un extracto de código en C que dado un arreglo de números, retorne el modo o un mensaje de que el mismo no existe.
Ejemplo: si se toma el ejemplo anterior el modo de arreglo sería 2.

8.- Programe en C un extracto de código que dado dos cadenas de caracteres realice la concatenación de las mismas. Ejemplo: Entrada: (Palabra1 = "mar" Palabra2 = "ysol") Salida: "marysol"

9.- Defina un arreglo de caracteres y utilicelo para almacenar polinomios, de tal manera que cada índice del arreglo represente la potencia de la variable y el valor contenido en ese índice el coeficiente del mismo. Codifique en C un extracto de codigo que dados dos polinomios almacenados en esta forma sea capaz de realizar la suma de polinomios y almacenar en un tercer arreglo el polinomio resultado. Ejemplo:
Entrada:

`Polin1` = $x^5 + 5x^3 + 6x^2 + 1$
Almacenado en forma de arreglo sería = [1,0,6,5,0,1]
`Polin2` = $2x^5 + 3x^4 + x^2 + 3x$
El arreglo seria = [0,3,1,0,3,2]

Salida:

el polinomio resultado de la suma = $3x^5 + 3x^4 + 5x^3 + 7x^2 + 3x + 1$

10.- Implemente una función ordenar que reciba cuatro números en las variables a, b, c y d y los retorne ordenados de menor a mayor contenidos en las mismas variables. Ejemplo:
Función `ordenar (a,b,c,d)`
Valores de entrada: a = 4, b = 2, c = 9, d = 10
Valores de salida: a = 2, b = 4, c = 9, d = 10

11.- Implemente una función que dado un string (arreglo de caracteres) determine su longitud.
Ejemplo: función `contar_caracteres(arreglo)` donde la variable arreglo es de tipo `string`.
Valores de entrada: `arreglo` = 'Ingeniería Electrónica'
Valores de salida: 22

12.- Implementar una función `reemplazar` que dado un arreglo `vector1` de enteros de `n` posiciones sustituya todas las ocurrencias del valor `valor1` por el valor `valor2`. Ejemplo:
Función `Reemplazar (vector1,valor1,valor2)`
Valores de entrada: `vector1`[2,5,3,5,2,7,8]; `valor1` = 2; `valor2` = 15
Valores de salida: `vector1`[15,5,3,5,15,7,8]

13.- Dado un arreglo de enteros como entrada, usted debe ser capaz de recorrerlo y eliminar del mismo aquellos elementos que aparezcan repetidos secuencialmente, produciendo como salida el arreglo sin estas secuencias repetidas. Adicionalmente, debe indicar cuantos elementos en total debió eliminar para producir el nuevo arreglo. Hacer el programa principal para probar la función. Ejemplo:
Funcion `elimrepeticion(ARREGLO,long)`
Valores de entrada: `ARREGLO`= [1,1,1,2,1,5,6,6,2,2,8,9,2,1,4,4]
`Long` = 16
Valores de salida: `ARREGLO`= [1,2,1,5,6,2,8,9,2,1,4]
`Eliminados`= 5

PRE Y POST CONDICIÓN

1.- Escriba el encabezado (prototipo, descripción, pre y postcondición) de la función que resuelve cada uno de los siguientes problemas:

a) Dado un vector `V` ordenado insertar un elemento en la posición que corresponda para que mantenga el orden.
b) Dado un vector `V` encontrar el mayor elemento del vector.
c) Calcular el factorial de un número `N`
d) Dado un vector `V` de `N` posiciones sin elementos repetidos, ordenar dicho vector.
e) Calcular el valor absoluto de un entero `N`
f) Dado una lista de valores en la entrada agregar ordenadamente dicho valores a un vector `V` de `MAX` posiciones, suponga que la primeras `N` posiciones están ocupadas
g) Invertir los elementos de un vector `V` de `N` posiciones
h) Sumar los elementos de un vector `V` y retornar el resultado
i) Determinar si dos vectores `V1` y `V2` de `N` posiciones son iguales
j) Reemplazar en un vector `V` de `N` posiciones todas las ocurrencias del valor `x1` por el valor `x2`
k) Imprimir todos los elementos del vector `V` de `N` posiciones que son mayores al valor `X`.
l) Rotar `M` posiciones a la izquierda los elementos de un vector `V` de `N` casillas.
m) Compactar un vector `V` de `N` posiciones, es decir, todos los elementos diferentes de cero deben estar en la parte inicial
n) Encontrar el elemento que aparece un mayor número de veces en un vector de tamaño `N`.

USO DE ARREGLOS Y STRUCT PARA CREAR ESTRUCTURAS COMPLEJAS

1.- Realice una función que dadas dos matrices (de longitud fija `NxN`) ejecute la suma de las mismas. Implemente dos versiones de la misma función de la siguiente forma:

a) Que imprima el resultado por pantalla.
b) Que retorne la matriz resultado al programa principal.

2.- Diseñe una estructura que almacene los datos de un programa para controlar las pantallas de video que muestran las salidas y llegadas de vuelos en un aeropuerto. Debe contener la información de los vuelos que llegan o salen, es decir, el nombre de la línea aérea, ciudad de la que procede el avión, puerta en la que llegará el avión, hora de llegada.

3.- Crear una estructura de datos para almacenar los datos de cada empleado de una empresa. La información que se desea almacenar es la siguiente: Cédula de identidad del empleado, carnet en la empresa, Nombre completo, fecha de nacimiento, sueldo básico, sueldo bonos. Es recomendable posteriormente crear otra estructura donde puedan almacenarse todas las estructuras anteriores. Se desea que implemente posteriormente una serie de funciones para realizar ciertas operaciones básicas:

Función	**Datos de Entrada**	**Datos de Salida**
Crear un nuevo empleado	Datos del empleado	Empleado creado
Modificar un empleado	Empleado que se quiere modificar, datos a modificar.	Empleado Modificado.
Calcular el sueldo total de un empleado	Empleado a calcular el sueldo	Sueldo total = Sueldo básico + sueldo bonos
Calcular el promedio de sueldo total de la empresa	Conjunto de Empleados de la empresa	Sueldo total promedio = suma de sueldos totales de todos los empleados / total de empleados

Elabore un programa que integre la estructura y las funciones anteriores. Dicho programa debe presentar un menú en pantalla que permita utilizar las funciones implementadas, de esta forma las opciones serán las siguientes: crear un nuevo empleado, modificar un empleado, calcular el sueldo total de un empleado, y calcular el promedio de sueldo total de la empresa. Los resultados de cada una de las operaciones deben ser presentados por pantalla.

4.- Crear una estructura de datos para representar polinomios de hasta grado N e implementar las siguientes funciones que la manipulen:

Función	**Datos de Entrada**	**Datos de Salida**
Crear un Polinomio	Grado del polinomio, valores de los coeficientes	Polinomio Creado
Modificar un polinomio	Polinomio creado, modificaciones a realizar	Polinomio Modificado.
Multiplicar Polinomios	Polinomio 1 y Polinomio 2 a multiplicar	Polinomio Resultante
Calcular la primera derivada de un polinomio	Polinomio a calcular la derivada	Polinomio Resultante

Ejemplo: Sea el polinomio $P = 3 + X^2 + 4X^3 + 2X^5$, con grado = 5, los valores de los coeficientes de menor a mayor son: 3, 0, 1, 4, 0, 2. Elabore un programa que integre la estructura y las funciones anteriores. Dicho programa debe presentar un menú en pantalla que permita utilizar las funciones implementadas, de esta forma las opciones serán las siguientes: crear un polinomio, modificar un polinomio, multiplicar polinomios, calcular la primera derivada de un polinomio, salir del programa. Los resultados de cada una de las operaciones deben ser presentados por pantalla.

5.- Un deportista cumple durante un año con un conjunto de actividades. El mismo desea poder llevar un registro de las puntuaciones obtenidas cada mes del año en cada una de las actividades que debe ejecutar. Para ello se le ha ocurrido almacenar esto en una matriz donde las filas representan los meses del año, y las columnas las actividades. De esta forma

una posición `(i,j)` de la matriz indica una puntuación obtenida dada una actividad y un mes particular. Se desea que usted le proporcione un pequeño programa con el cual pueda ejecutar fácilmente las siguientes operaciones:

a) Crear una nueva matriz de registro. La cual crea una nueva matriz, con una etiqueta, que identifica al año particular del cual se quiere efectuar el registro. Nuevamente es importante mencionar que las filas vienen representadas por los meses del año, y las columnas por las actividades a registrar.
b) Cargar una nueva puntuación de una actividad, para un mes, y un año dados. Para ello deberá proporcionarse la matriz que identifica al año, el mes, la posición que identifica la actividad, y la puntuación a insertar.
c) Modificar la puntuación obtenida en un año particular, dados la actividad y el mes asociados. Debe proporcionarse la posición de la actividad y el nuevo valor a colocar.
d) Obtener los promedios del deportista por cada actividad al final del año, el cual es el resultado de sumar cada una de las puntuaciones correspondientes para todos los meses y dividir cada resultado entre doce.

Es importante que recuerde el correcto uso de pasaje de parámetros y funciones. Ejemplo:

REGISTRO_1998 =

Meses	Actividades		
	Act1	Act2	Act3
Enero	10	5	9
Febrero	2	5	10
Marzo	8	3	1
Abril	9	9	8
.....	.....	.....	.....

Diseñe una estructura que almacene los datos de un programa para controlar las pantallas de video que muestran las salidas y llegadas de vuelos en un aeropuerto. Debe contener la información de los vuelos que llegan o salen, es decir, el nombre de la línea aérea, ciudad de la que procede el avión, puerta en la que llegará el avión, hora de llegada.

ARCHIVOS

A través de los archivos se pueden almacenar de forma persistente una cantidad cualquiera de elementos de un mismo tipo. Los archivos se pueden manipular a través de una variable conocida como File Pointer, la cual señala en todo tiempo el elemento visible actualmente del archivo, ya que sólo es visible un elemento a la vez. La declaración de una variable file pointer de tipo texto sería `FILE *fp;`

Para usar un archivo debe abrirse previamente con la instrucción fopen, la cual devuelve el apuntador a una estructura de tipo `FILE` o file pointer. En esta instrucción debe especificarse el nombre del archivo que se va abrir y el modo de apertura: "r" abierto para lectura, "w" creado para escritura, "a" para agregar al final, "r+" abierto para lectura y escritura, "w+" creado para lectura y escritura, "a+" para lectura o agregar al final. En la siguiente instrucción se abre el archivo cliente.dat para lectura: `cfPtr = fopen("cliente.dat","r")`

Con las funciones `fscanf` y `fprintf` se pueden leer o escribir valores en un archivo, y finalmente se debe cerrar para garantizar que las modificaciones realizadas no se pierdan con la función fclose.

El siguiente programa crea un archivo de enteros positivos, provistos por el teclado hasta que se introduzca un valor negativo.

```
main () {
   FILE *fp;
   int elem;
   char nombre[20];

   printf("Introduzca el nombre del archivo a crear:");
   scanf("%s",nombre);

   /* Crea el archivo para escritura */
   if ((fp=fopen(nombre,"w")) != NULL) {
      printf("Introduzca los valores enteros positivos:");
      scanf("%d",elem);
      while (elem>=0) {
         /*Escribe el elemento en el archivo*/
         fprintf(fp,"%d",elem);
         scanf("%d",elem);
      }
      fclose(fp); /*Cierra el archivo por seguridad*/
   }
   else
      printf("No se pudo crear el archivo");
}
```

Ejercicios:

1.- Indique cuáles de los siguientes enunciados son verdaderos y cuáles son falsos:

- La función fscanf no puede ser utilizada para leer datos de la entrada estándar.
- El programador debe utilizar fopen explícitamente, para abrir los flujos de entrada estándar, salida estándar y error estándar.
- Para cerrar un archivo un programa debe llamar en forma explícita a la función fclose.
- La función fprintf puede escribir a la salida estándar.
- La función fseek puede buscar únicamente en relación con el principio de un archivo.

2.- Encuentre el error en cada una de las siguientes trozos de programa y diga cómo corregirlo

```
{
FOPEN *fPtr;

      fprintf(fPtr,"%d%s%s\n",cuenta,compañía,cantidad);
}
```

```
{open("receive.dat","r+"); }
```

El archivo "tools.dat" debería ser abierto para añadir datos al archivo, sin descartar los datos actuales:

```
if ((tfPtr = fopen("tools.dat","w")) != NULL)
```

El archivo "courses.dat" debería ser abierto para agregar sin modificar el contenido actual.

```
if ((tfPtr = fopen("courses.dat","w+")) != NULL)
```

3.- Suponga que la siguiente estructura ha sido definida y que el archivo está abierto para escritura.

```
struct persona{
      char apellido[15];
      char nombre[15];
      char edad[2];
}
```

a) Escriba instrucciones que resuelvan las siguientes proposiciones:
b) Inicialice el archivo "NOMBRES.DAT" de tal forma que existan 100 registros con apellido = "noasignado", nombre="" y edad="0'.
c) Lea de la entrada 10 apellidos, nombres y edades y escríbalos al archivo.
d) Actualice un registro. Si no existe información en el registro, indique al usuario "No info".
e) Borre un registro que tenga información mediante la reinicialización de dicho registro en particular.

4.- Hacer un subprograma que dados los nombres de dos archivos de caracteres haga una mezcla de los dos en uno nuevo archivo, con la condición de que se intercalen las líneas. El nombre del nuevo archivo también es dado como parámetro.

5.- Se tiene un archivo cuyos registros poseen la siguiente estructura: Nombre, Edad, Sexo. Se desea elaborar un programa que liste todas las mujeres que sean mayores de una edad X. Dichos datos se encuentran dentro del archivo. El nombre del archivo es dado por el usuario.

7.- Elabore un programa que cree un archivo de caracteres con líneas de tamaño fijo 80, a partir de un archivo (también de texto) que contiene líneas de tamaño variable entre 1 y 80 caracteres. Para ello el archivo creado se rellena de blancos en las líneas con tamaño menor que 80.

8.- Elabore un programa que tome todos los múltiplos de 4 hasta un N dado, y los almacene en un archivo de caracteres .

9.- Se tiene un archivo de enteros. Se desea elaborar un programa que nos indique la posición del máximo y la del mínimo de dicha lista de enteros.

10.- Se desea hacer una copia de un archivo de caracteres con la condición de que sólo se almacenen (en un nuevo archivo) las líneas que se encuentran en posición par.

11.- Tomar dos archivos de texto y hacer una mezcla de los dos en un nuevo archivo, con la condición de que se intercalen las líneas.

12.- Se tiene un archivo cuya estructura es la siguiente: Nombre, Edad, Sexo. Se desea elaborar un programa que liste todas las mujeres que sean mayores de una edad X. Dichos datos se encuentran dentro del archivo.

13.- Elaborar un programa que nos permita guardar el CÓDIGO, NOMBRE y NOTA de las materias cursadas, en un archivo. Las operaciones a realizar sobre dicho archivo son: listar las materias que pertenezcan a cierto departamento (Ej: EC, MA, CI, etc), listar las materias en las cuales se haya obtenido determinada nota, y listar todas las materias cursadas.

14.- Escribir un programa que reciba como entrada el nombre de un archivo y cree un archivo copia.dat con el contenido del mismo.

15.- Los datos mensuales que maneja un sistema de nómina están almacenados en un archivo de texto de la siguiente manera:

NombreEmpleado	HorasTrabajadas	SueldoPorHora
Pepe Rodríguez	5	3000
Julio López	15	100

.. ..

El archivo se llama “master.dat”. Escribir un programa que calcule el promedio de horas trabajadas en el mes y la cantidad de dinero total a pagar en el mes.

16.- Con el mismo formato de archivo del problema anterior el gerente quiere obtener un archivo “master2.dat” que contenga solamente los empleados que hayan trabajado más de 20 horas y ganen menos de 1500 Bs. por hora.

17.- El sistema de control de inscripciones de D.A.C.E mantiene la información de los estudiantes en un archivo de texto de la siguiente manera:

CarnetEst	NombreEst	IndiceEst	SexoEst
90-22250	Ricado P.	3.61	M
90-22251	María P.	4.23	F
...	...	...	...

El archivo se llama “CEST.DAT”. Escribir un programa que:
Calcule el índice promedio de los estudiantes.
Calcule el número de estudiantes con carnet menor a 91.
Calcule el número de estudiantes masculinos con índice superior a 3.5.
Dado un número de carnet muestre el índice del estudiante.

18.- Con el formato de archivo anterior bienestar estudiantil desea obtener “CEST1.DAT”, “CEST2.DAT”, “CEST3.DAT”, “CEST4.DAT” tales que:

a) “CEST1.DAT” contenga la información de los estudiantes masculinos carnet 90 con índice mayor o igual a 3.50
b) “CEST2.DAT” contenga la información de los estudiantes con índice mayor a 4.25 que entraron después del año 92
c) “CEST3.DAT” contenga los nombres de las estudiantes 94 cuyos apellidos comiencen con A y B.
d) “CEST4.DAT” contenga los carnet de los estudiantes que se llamen José y tengan índice superior a 4.00

RECURSION

1.- Implementar las siguientes funciones en forma recursiva, en cada caso escriba la definición recursiva de manera matemática. No olvide colocar la pre y post-condición de cada función.:

a) `int contar (int v[],int elem)`: cuenta el número de veces que aparece `elem` en el arreglo `v`
b) Función que imprima los elementos de un arreglo de tamaño `N`
c) Función que imprima en orden inverso los elementos de un arreglo de tamaño `N`
d) Determinar si dos vectores `V1` y `V2` son iguales
e) Función que suma `N` enteros positivos leidos de la entrada
f) Decidir si un elemento `elem` está en un arreglo `V`, el arreglo está desordenado
g) Encontrar el menor valor de un vector `V`.
h) Decidir si un vector se encuentra ordenado
i) Calcular el número de valores diferentes que se encuentran en un vector
j) Dado un elemento y un valor `i`, encontrar la `i`-ésima ocurrencia del elemento en un arreglo `V`.
k) Encontrar el elemento que aparece un mayor número de veces en un arreglo `V`
l) Dados dos vectores `V1` y `V2` de tamaño `N`, encontrar el vector `V` que resulta de mezclar los dos vectores. `V1` y `V2` están ordenados ascendentemente.
m) Dado un vector `V` y elemento `e`, eliminar todas las ocurrencias del elemento en el vector `V`.
n) Dado un polinomio `P` de grado `m`, almacenado en un arreglo, encontrar la evaluación del polinomio para el valor `X`.
o) Dadas dos matrices A y B encontrar la suma de ellas
p) Dadas dos matrices `A` y `B`, encontrar el producto de ellas
q) Dados dos string `S1` y `S2`, escribir una función que devuelva `0` si son iguales, `1` si `S1 > S2` y `-1` si `S1 < S2`

2.- Escriba un algoritmo iterativo y uno recursivo para los siguientes problemas.

a) Calcular la evaluación de la serie de Taylor $\operatorname{sen} x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$, en un x dado

b) El número combinatorio $\binom{n}{m} = \frac{n!}{m!(n-m)!}$

3.- Escribir una función recursiva que devuelva `a*b` (producto de `a` y `b`) usando la adición, y suponga que `a` y `b` son enteros no negativos.

4.- Sea `sec1` y `sec2` secuencias de caracteres representadas con arreglos. Elaborar un programa que determine si `sec2` se encuentra contenido en `sec1`. Utilizar recursión.

5.- Escribir una función recursiva que devuelva a^b (potencia de `a` elevado a la `b`).

6.- Escriba una función recursiva que implemente el algoritmo de Euclides para el cálculo del máximo común divisor de dos números `p` y `q (p>q)`. Las condiciones del algoritmo son:
`MCD(p,q)=q` si el resto de la división `p/q` es cero
`MCD(p,q)=MCD(q,r)` para los otros casos, donde `r` es el resto de la división `p/q`

7.- Una forma de calcular el combinatorio de dos valores r y k es

$$\binom{r}{k} = \binom{r-1}{k} + \binom{r-1}{k-1} \quad con \quad r \geq k \geq 0 \quad y \quad \binom{n}{0} = \binom{n}{n} = 1 \quad \forall n .$$

Escriba una función recursiva que calcule el número combinatorio para `r` y `k`

8.- Sea **`p`** un arreglo de caracteres, **`p`** es un palíndrome si se cumple que al invertirlo se obtiene la misma secuencia de caracteres. Escriba la definición recursiva y una función en lenguaje C que retorne `TRUE` si `p` es un palídrome y `FALSE` en caso contrario

9.- Dados dos strings `S1` y `S2` escriba una función recursiva que verifique si `S2` se encuentra en `S1`. En caso de encontrarse devuelva la posición dentro del string `S1` donde comienza `S2` en caso contrario devuelva el mensaje "No encontrado".

10.- Dados tres string S1,S2,S3 y un número entero x. Escriba una función recursiva que verifique cuántas veces y en que posiciones S1 se encuentra a menos de x caracteres de distancia de S2 dentro del string S3. La función de recorrido del string S3 es recursiva y al finalizar el recorrido debe imprimir las posiciones en las que S1 está a menos de x caracteres de S2.

11.- Implemente la siguiente función en C:

$$f(x)=\begin{cases}1 & si \quad x>100\\ 0 & si \quad x=0\\ x*f(x+1) & si \quad no\end{cases}$$

12.- Se tiene un arreglo A de números enteros diferentes, y de longitud n, con la condición que el primer elemento es menor que el último elemento; es decir A[0]<A[n-1]. Se conoce con el nombre de ascenso una posición i donde A[i]<A[i+1]. Escriba una función recursiva que calcule una posición donde se encuentre un ascenso del arreglo. No se puede modificar los elementos del arreglo. Ejemplo: en el siguiente arreglo, las posiciones de los ascensos están subrayadas

arreglo:	4	2	15	7	1	5	8	6	18	11
posición:	0	1	2	3	4	5	6	7	8	9

13.- Escriba una función recursiva en C que invierta una cadena de caracteres. La función recibe como parámetros las dirección del primer elemento y la del último elemento de la cadena.

14.- Se define el punto de ensilladura de una matriz de tamaño mxn como la posición i,j que contiene al mínimo valor de la fila i que corresponde al máximo valor de la columna j. Por ejemplo, si la matriz es la siguiente (mxn=3x2):

1	5
14	9
8	6

La misma posee un punto de ensilladura en la posición 2,2 (mínimo valor de la fila 2 que corresponde al máximo valor de la columna 2). Escriba una función en C que escriba los valores de i y de j del punto de ensilladura de una matriz dada en el caso de existir. En caso contrario debe devolver los valores -1,-1.

15.- La figura del caballo (en ajedrez) posee un movimiento muy particular sobre el tablero, ya que se desplaza tres posiciones siguiendo exacta y únicamente movimiento NO diagonales, de tal manera que se desplaza dos posiciones en algun sentido y luego una posición hacia su derecha o hacia su izquierda deacuerdo a su sentido original (formando una especie de L). Sobre una matriz de 8x8 que simula el tablero del ajedrez (no importa el color del caballo, ni de los cuadros) y dadas las posiciones i,j (donde se encuentra inicialmente el caballo) y la posición f,c (ambas dentro de los límites de la matriz), decidir utilizando una función recursiva, si existe un camino desde i,j hasta f,c. Inicialmente el tablero tiene el valor de cero (0) en todas sus casillas y la función por usted diseñada debe ir marcando el camino recorrido por el caballo con el valor de uno (1) en cada casilla visitada. Puede haber varios caminos o ninguno. Recuerde que no se le está pidiendo un camino particular, únicamente la información de existencia o no de caminos.

16.- Dada una matriz nxn de números enteros y un par (x,y). Escriba una función recursiva que recorra la matriz partiendo de la posición (x,y) y colocando los valores en un vector. Obviamente el vector es de tamaño n*n. La regla para el recorrido es la siguiente: partiendo de la posición (x,y) debe moverse a la posición vecina que contenga el número de mayor magnitud y colocarlo en el vector. Esta posición se marca como visitada. Debe continuar hasta finalizar con la matriz sin moverse a posiciones ya visitadas. Puede moverse vertical, horizontal o diagonalmente.

17.- Elaborar un programa recursivo que reciba una matriz que simbolice un laberinto, de manera que una casilla con valor X identifique a una pared (no pase) y una casilla con el caracter 'blanco' identifique un camino libre (si paso). Además debe recibir una posición (i,j) que refleja la posición inicial dentro de la matriz. El programa debe indicar si el laberinto, representado por la matriz, tiene salida. La idea es hacer un programa que simule el recorrido dentro del laberinto a través de los caminos libres.

18.- Si se tiene la siguiente función recursiva para calcular la secuencia de Fibonacci hasta un entero n:

```
int fib (n)
int n;
{
      if (n < = 1)
            return(n);
      else
            return(fib(n-2)+fib(n-1));
}
```

Realice la corrida de la función anterior para un `n` dado (por ejemplo `n=6`) e indique los ambientes de función que se crean en la corrida. Indique también el valor de n en cada punto de la ejecución (recordar aquí la existencia de un ambiente global y ambientes locales de funciones por cada llamada).

19.- Supongamos que se quiere calcular la siguiente sumatoria $\sum_{x=i}^{n} x!$. Si se asume que el el límite superior e inferior de la sumatoria son variables, programe en C una función recursiva que dado i y n como entradas, sea capaz de retornar el resultado del cálculo de la función anterior. Coloque las precondiciones que considere necesarias para que su función trabaje correctamente.

USO DE TIPOS ABSTRACTOS DE DATOS

1.- Un conjunto es una colección de elementos donde cada uno de dichos elementos ocurre a lo sumo una vez. El orden de los elementos dentro del conjunto no tienen importancia. Se desea definir el `TAD CONJUNTO` para poder manejar objetos tipo conjunto, con las siguientes operaciones:

```
C_crear: void → CONJUNTO
C_destruir: CONJUNTO → void
C_unión: CONJUNTOxCONJUNTO → CONJUNTO
C_intersección: CONJUNTOxCONJUNTO → CONJUNTO
C_pertenece: ELEMxCONJUNTO → bool
C_vacío: CONJUNTO → bool
C_extraer: CONJUNTO → ELEM
```

a) Para cada una de las operaciones especificadas escriba sus prototipos.
b) Escriba las operaciones de intersección y unión. usando las demás primitivas del `TAD CONJUNTO`
c) Escriba la operación `C_iguales: CONJUNTOxCONJUNTO → bool` que diga si dos conjuntos son iguales.
d) Escriba la operación `C_subconjunto: CONJUNTOxCONJUNTO → bool` que diga si un conjunto es subconjunto de otro.

2.- Se define el TAD VECTOR como una colección de datos unidimensional del mismo tipo ELEM, con las siguientes operaciones básicas:

V_Crear: ℵ → VECTOR
/* Devuelve un vector con el tamaño indicado */
V_Destruir: VECTOR→ void
/* Elimina un vector*/
V_AsignarElem: VECTOR x ELEM x ℵ → VECTOR
/* Inserta un elemento en la posición dada dentro del vector*/
V_Elemento: VECTOR x ℵ → ELEM
/* Retorna el elemento que se encuentra en la posición dada del vector */
V_Longitud: VECTOR → ℵ
/* Retorna la longitud del vector */
Suponga que las siguientes operaciones para el manejo del TAD ELEM están disponibles:
E_Iguales: ELEM x ELEM → BOOL
/* Retorna TRUE si los elementos dados son iguales */
E_Mayor: ELEM x ELEM → BOOL
/* Retorna TRUE si el primer elemento dado es mayor que el segundo*/
E_Swap: ELEM x ELEM
/* Intercambia ambos elementos */

a) Escriba la conceptualización de la estructura
b) Escriba el prototipo para cada función.
c) Usando las primitivas anteriores, codifique en C las siguientes operaciones:
V_Iguales: VECTOR x VECTOR→ BOOL
/* Retorna TRUE si los vectores son iguales */
V_Mínimo: VECTOR → ELEM
/* Retorna el mínimo elemento de un vector dado */
V_Ordenar_Ascendente : VECTOR → void
/* Ordena de manera ascendente el vector dado */
d) Escriba un función que sume dos vectores y retorne el vector resultado:
VECTOR sumaVector (VECTOR V1, VECTOR V2);

Suponiendo que está implementada esta definición de VECTOR, escriba una función que haciendo uso de las primitivas anteriores ejecute una suma de matrices.

3.- Se define el TAD MATRIZ de valores enteros, con las siguientes operaciones básicas:

M_Crear: ℵ x ℵ → MATRIZ
/* Devuelve una matriz con el tamaño bidimensional indicado */

M_Destruir: MATRIZ→ void
/* Elimina una matriz*/
M_AsignarElem: MATRIZ x ℵ x ℵ x ℵ → MATRIZ
/* Asigna a la casilla dada el entero dado */
M_Información: MATRIZ x ℵ x ℵ → ℵ
/* Retorna el valor que se encuentra en la casilla dada de la matriz */
M_Filas: MATRIZ → ℵ
/* Retorna el número de filas de la matriz */
M_Columnas: MATRIZ → ℵ
/* Retorna el número de columnas de la matriz */

a) Escriba la conceptualización de la estructura
b) Escriba el prototipo para cada función.
c) Usando las primitivas anteriores, codifique en C las siguientes operaciones:
M_Iguales: MATRIZ x MATRIZ → BOOL
/* Retorna TRUE si las matrices dadas son iguales */
M_Copiar: MATRIZ → MATRIZ
/* Devuelve una copia de la matriz dada */
M_Imprimir: MATRIZ → void
/* Imprime una matriz dada */
M_Cargar: FILE → MATRIZ
/* Llena una matriz con los datos del archivo indicado */
M_Salvar: MATRIZ x FILE → void
/* Escribe una matriz en el archivo indicado */
M_Sumar: MATRIZ x MATRIZ → MATRIZ
/* Retorna la matriz suma de las dos matrices dadas */
M_Multiplicar: MATRIZ x MATRIZ → MATRIZ
/* Retorna la matriz producto de las dos matrices dadas */
M_Traspuesta: MATRIZ → MATRIZ
/* Retorna la matriz traspuesta de una matriz dada */
M_CuadradoMágico: MATRIZ → BOOL
/* Retorna TRUE si las matriz es un cuadrado mágico. Es decir si el resultado de sumar los elementos de cada una de las filas es el mismo a a suma de los elementos de cada una de las columnas y a la suma de los elementos de cada una de las diagonales. Ejemplo:

8	1	6
3	5	7
4	9	2

Es un cuadrado mágico porque sus filas, sus columnas y sus diagonales suman 15 */

4.- Se define el TAD DICCIONARIO de valores enteros, con las siguientes operaciones básicas:
D_Crear: void → DICCIONARIO
/* Construye un diccionario sin palabras */
D_Destruir: DICCIONARIO → void
/* Destruye un diccionario*/
D_AgregarPal: DICCIONARIO x STRING x STRING → DICCIONARIO
/* Agrega una palabra y su significado al diccionario. La palabra no estaba en el diccionario */
D_AgregarSigPal: DICCIONARIO x STRING x STRING → DICCIONARIO
/* Agrega un nuevo significado a una palabra existente del diccionario */
D_SignificadoPos: DICCIONARIO x STRING x ℵ → STRING
/* Retorna el significado indicado por su posición de una palabra del diccionario */
D_CuantosSig: DICCIONARIO x STRING → ℵ
/* Retorna el número de significados de la palabra dada encontrados en el diccionario */
D_EliminarPal: DICCIONARIO x STRING → DICCIONARIO
/* Elimina una palabra con todos sus significados del diccionario */

a) Escriba la conceptualización de la estructura
b) Escriba el prototipo para cada función.
c) Usando las primitivas anteriores, codifique en C las siguientes operaciones:
D_Relacionados: DICCIONARIO x STRING x STRING → BOOL

/* Retorna TRUE si la palabra y el significado dados están relacionados */
D_Sinónimas: DICCIONARIO x STRING x STRING → BOOL
/* Retorna TRUE si las palabras dadas son sinónimas. Es decir comparte al menos un significado */
D_NumSigIguales: DICCIONARIO x STRING x STRING → ℵ
/* Retorna el número de significados que comparten dos palabras dadas*/

5.- El profesor Carlos X trabaja en una institución de Educación Superior y dicta varios cursos relacionados con la carrera de computación. Cada trimestre dicta aproximadamente de cuatro a cinco cursos. Por cada uno de los cursos Carlos X, construye unas listas de registros de sus estudiantes clasificados según el siguiente criterio: Lista1: los estudiantes con índice acumulado en un curso particular entre 1 y 3 inclusive; Lista2: los estudiantes con índice entre 3,1 y 4; Lista3: los estudiantes entre 4,1 y 5. En cada curso los estudiantes pueden ser cualquiera de los que están actualmente participando en la carrera, lo cual significa que puede tener alumnos repetidos en sus cursos (es decir el estudiante Pepe1, puede estar cursando durante un trimestre dos cursos o más con el profesor Carlos X). Al final de cada trimestre, Carlos X desea poder colocar las listas de los alumnos en 3 grandes listas clasificadas por las mismas categorías, para observar comportamientos generales de sus estudiantes (por ejemplo, para ver que estudiantes han podido estar dentro de un rango dado en todas las materias vistas con el en ese período). Ejemplo:

Curso1:	Curso2:	Curso3:
Lista1 = [Carlos1]	Lista1 = []	Lista1 = [Manuel1]
Lista2 = [Pepe1, Manuel1, Sandra1]	Lista2 = [Pepe1, Manuel1]	Lista2 = [Sandra1]
Lista3 = [Manuel2]	Lista3 = [Sandra1]	Lista3 = []

Listas finales:
Lista1 = [Carlos1, Manuel1]
Lista2 = [Pepe1, Manuel1, Sandra1]
Lista3 = [Manuel2, Sandra1]

De esta manera al final del período anterior, por ejemplo, el estudiante Pepe1, se mantuvo siempre dentro del rango de 3.1 a 4, en los cursos que tomo con el profesor Carlos X.
Se desea que usted ayude al profesor y que utilizando la definición del `TAD CONJUNTO` y sus primitivas, defina la rutina necesaria para crearle las listas finales por categoría que incluya a todos los cursos y estudiantes dictados por el durante un período dado.

6.- La arquitectura de la mayoría de los computadores está basada en un esquema monoprocesador. Sin embargo, muchos de estos computadores tienen que soportar el ser usados por múltiples usuarios a la vez. Para administrar el uso del procesador por los diversos usuarios se aplica un esquema de **tiempo compartido** en el cual se asigna un lapso de tiempo a cada proceso de los usuarios.

El programa del sistema de operación encargado de administrar el tiempo de CPU a los procesos se llama **scheduler**. Es el scheduler el que controla el lapso de tiempo durante el cual cada proceso de ejecuta. El scheduler maneja un conjunto de procesos activos. Este conjunto puede estar vacío en el momento en que todos los procesos del conjunto han terminado su ejecución. En cualquier instante un nuevo proceso puede incorporarse al conjunto, y se dice que el proceso pasa a estar activo (en ejecución). Los procesos salen del conjunto cuando termina su ejecución.

Cada proceso tiene un tiempo de ejecución durante el cual realiza su labor. Cada vez que el scheduler le asigna un lapso de tiempo al proceso, se decrementa en uno (1) el tiempo restante de ejecución. Cuando este llega a cero termina el proceso, por lo que sale del conjunto de procesos activos y debe ser destruído. Si no termina queda en espera de que el scheduler le asigne otro lapso de tiempo. El lapso de tiempo que el scheduler dedica a cada proceso es igual para todos sin importar el tipo de proceso que sea.

De esta forma, para implementar el scheduler es necesario usar dos TAD básicos: el TAD PROCESO y el TAD ACTIVOS.

El TAD PROCESO, el cual contiene la identificación del proceso y su tiempo de ejecución, posee las siguientes operaciones básicas:

```
PROCESO P_CrearNuevo (int tiempo);
/* devuelve un proceso con un ID y con tiempo de ejecución el indicado */
void P_Destruir (PROCESO p);
/* destruye el proceso p */
BOOL P_Ejecutar (PROCESO p, int t);
```

/* dado un proceso p, lo ejecuta un lapso de tiempo t, devuelve `TRUE` si terminó, `FALSE` si no*/

El TAD ACTIVOS contiene el conjunto de procesos activos y sus operaciones son las siguientes:

```
PROCESOS Act_Actual(ACTIVOS CA);
/* devuelve el proceso actual del conjunto de procesos activos */
ACTIVOS Act_Avanzar (ACTIVOS CA);
/* Cambia el proceso actual al siguiente */
BOOL Act_Vacio (ACTIVOS CA);
/* dice si no hay más procesos a servir, TRUE si el conjunto es vacío */
ACTIVOS Act_InsertarProceso (ACTIVOS CA, PROCESO p);
/* inserta el proceso p en el conjunto CA como anterior al actual */
ACTIVOS Act_eliminarProceso (ACTIVOS ca, int ID);
/* saca o elimina el proceso identificado como ID del conjunto CA */
```

a) Escriba la estructura del TAD PROCESO (conceptualización e implementación). Debe ser un apuntador a un `struct`.
b) El TAD ACTIVOS se diseña como una lista circular con un encabezado, que tiene un apuntador al proceso actual. Escriba la estructura del TAD (conceptualización e implementación).
c) Escriba la implementación de la operación `P_crearNuevo(int tiempo)`. Suponga que existe una función `int PID()` que devuelve un nuevo identificador para proceso.
d) Escriba la operación `BOOL Act_vacio (ACTIVOS CA);`
e) Suponga que están implementados los TAD PROCESO y ACTIVOS. Escriba la función scheduler (si da tiempo si no asignación).
f) Dejar para ellos como repaso la implementación del resto de las funciones que componen los TAD.

7.- Usando el `TAD CONJUNTO` elabore un programa que lo use como una librería. Dicho programa debe presentar un menú en pantalla con las siguientes opciones:
a) Crear un conjunto.
b) Insertar un elemento en un conjunto.
c) Eliminar un elemento en un conjunto.
d) Mostrar un conjunto.
e) Generar la intersección de dos conjuntos.
f) Salir del programa.

8.- Una cadena de comida rápida posee un servicio de atención en el carro, de tal forma que el cliente pueda pedir su comida para llevar sin bajarse de su automóvil. La empresa desea poder registrar este proceso de atención en el carro. En este sentido se quiere poder almacenar ciertos datos del auto (modelo, placa, color, año), datos del conductor (nombre, ci), y específicos del pedido (lista de comida, monto, forma de pago). A todo este conjunto de información le será llamado un individuo. Cada individuo tendrá además un estátus asociado que dependerá del punto de atención en el que se encuentre (inicialmente estatus = en espera, una vez que le haya sido tomado su pedido, estatus = con pedido, cuando haya efectuado el pago, estatus = cobrado, y una vez que le haya sido entregado el pedido, simplemente se elimina del registro). De esta forma la empresa lo ha contratado a usted para que realice las siguientes actividades:
a) Conceptualizar e implementar un TAD para el registro de toda esta información.
b) Implementar un conjunto de operaciones básicas para el TAD, de apoyo al proceso:
   - Crear un nuevo registro de atención.
   - Insertar un nuevo individuo, (estatus inicial = en espera).
   - Tomar el pedido de un individuo. Con esta función el individuo pasa de estatus = en espera al de con pedido.
   - Efectuar el cobro a un individuo, cambiando el estatus de con pedido a cobrado.
   - Entrega de pedido, acción con la cual se debe eliminar al individuo del registro.
   - Modificar algún dato asociado a un individuo, puede ser del auto, el conductor o el pedido. Para la modificación de este último debe validarse el estatus actual del individuo, el cual debe ser al menos con pedido.

IMPLEMENTACIÓN DE TIPOS ABSTRACTOS DE DATOS

1.- Se define el TAD LISTA con la siguiente estructura conceptual

$<e_1,e_2,e_3,...e_n>$ donde $n \geq 0$ y existe un elemento e_i, llamado ventana, que es el elemento visible de la lista

y cuyas operaciones son:

a) `L_Crear: void -> LISTA` /* Devuelve una lista vacía. La ventana está indefinida */
b) `L_Destruir: LISTA -> void` /* Destruye el objeto lista liberando toda la memoria ocupada. */
c) `L_InsertAnt: LISTA x ELEM -> LISTA` /* Devuelve una lista con el elemento insertado en la posición anterior a la ventana, la ventana queda sobre el elemento*/
d) `L_InsertPost: LISTA x ELEM -> LISTA` /* Devuelve una lista con el elemento insertado en la posición posterior a la ventana, la ventana queda sobre el elemento*/
e) `L_Eliminar: LISTA -> LISTA` /* Elimina el elemento actual de la ventana */
f) `L_Info: LISTA -> ELEM` /* Devuelve la información del elemento visible en la ventana*/
g) `L_PrimerLista: LISTA -> LISTA` /* Mueve la ventana al primer elemento de la lista*/
h) `L_UltimoLista: LISTA -> LISTA` /* Mueve la ventana al último elemento de la lista*/
i) `L_Siguiente: LISTA -> LISTA` /* Mueve la ventana al elemento siguiente de la lista*/
j) `L_Anterior: LISTA -> LISTA` /* Mueve la ventana al elemento anterior de la lista*/
k) `L_Vacia: LISTA -> bool` /* Dice si la lista es vacía */
l) `L_Final: LISTA -> BOOL` /* Devuelve TRUE si es el último elemento de la lista */

Usando estas primitivas implemente las siguientes operaciones sobre el TAD LISTA:

a) `L_Tamaño: LISTA -> int` /* Devuelve el tamaño de la lista */
b) `L_Proyectar: LISTA x int -> ELEM` /* Retorna el elemento de la posición dada */
c) `L_InsertPosic: LISTA x int -> LISTA` /* Inserta un elemento en la posición dada */
d) `L_ElimPosic: LISTA x int -> LISTA` /* Elimina el elemento de la posición dada */
e) `L_Imprimir: Lista -> void` /* Imprime el contenido de una lista */
f) `L_Copiar: LISTA -> Lista` /* Devuelve la copia de la lista */
g) `L_Concatenar: LISTA x LISTA -> LISTA` /*Devuelve una lista con la concatenación de las dos listas dadas*/
h) `L_Invertir: LISTA -> LISTA` /* Devuelve una nueva lista con el resultado de invertir la lista original */
i) `L_Localizar: LISTA x ELEM -> void` /* Coloca la ventana en el elemento buscado. Si el elemento no está, la ventana queda indefinida */
j) `L_EliminarTodos: LISTA x ELEM -> LISTA` /* Elimina todas las ocurrencias del elemento.*/
k) `L_Iguales: LISTA x LISTA -> BOOL` /*Dice si dos listas son iguales (tienen la misma longitud, y los elementos son iguales en la misma posición).*/
l) `L_Ordenada: LISTA -> BOOL` /* Dice si una lista está ordenada. Suponga que existe una función que dice si un elemento es menor que otro.*/
m) `L_Sublista: LISTA x LISTA -> BOOL` /* Dice si la segunda lista es sublista de la primera. */
n) `L_Agregar: LISTA x ELEM -> LISTA` /* Agrega el elemento al final de la lista */
o) `L_Sustituir: LISTA x ELEM -> LISTA` /* Sustituye el contenido actual de la ventana por el valor del elemento */
p) `L_Buscar: LISTA x ELEM -> BOOL` /* Dice si un elemento está en la lista */
q) `L_Anterior: LISTA -> void` /* Coloca la ventana en el elemento anterior de la lista. */
r) `L_Simplificar: LISTA -> LISTA` /* Elimina los elementos repetidos de la lista. Queda una sola ocurrencia de cada elemento */
s) `L_NumDiferentes: LISTA -> int` /* Devuelve el número de elementos diferentes de la lista */
t) `L_NumOcurrencias: LISTA x ELEM -> int` /* Devuelve el número de ocurrencias de un elemento dentro de la lista. */
u) `L_Rotar: LISTA x int -> LISTA` /* Devuelve la lista rotada n posiciones. Rotar una posición significa pasar el primer elemento de la lista a la última posición y desplazar todos los demás elementos una posición a la izquierda. */
v) `L_Ordenar: LISTA -> LISTA` /* Ordena ascendentemente la lista. */
w) `L_Mezclar: LISTA x LISTA -> LISTA` /* Devuelve la mezcla de dos lista ordenadas ascendentemente */

2.- Implemente en C cada una de las primitivas del TAD LISTA

3.- Se define el TAD PILA como una colección de elementos $<e_1,e_2,e_3,...e_n>$ donde $n \geq 0$ y e_1 se llama el tope de la pila. Las operaciones primitivas de este TAD son:

a) `P_Top: PILA -> ELEM` /* Devuelve el tope de la pila */
b) `P_Pop: PILA -> PILA` /* Extrae el primer elemento de la pila */
c) `P_Push: PILA x ELEM -> PILA` /* Coloca un elemento en el tope de la pila */
d) `P_Empty: PILA -> BOOL` /* Dice si la pila está vacía */

Implemente el TAD PILA usando la definición y las operaciones del TAD LISTA

4.- Se define el TAD COLA como una colección de elementos $<e_1,e_2,e_3,...e_n>$ donde $n \geq 0$ y e_1 es el inicio de la cola y e_n es el final de la cola. Las operaciones primitivas de este TAD son

a) `C_Insert: COLA x ELEM -> COLA` /* Inserta un elemento al final de la cola*/
b) `C_Remove: COLA -> COLA` /* Elimina el primero de la cola */
c) `C_First: COLA -> ELEM` /* Devuelve el primer elemento de la cola */
d) `C_Empty: PILA -> BOOL` /* Dice si la cola está vacía */

Implemente el TAD PILA usando la definición y las operaciones del TAD LISTA

5.- ¿Qué puede modificar del TAD LISTA para que el manejo del TAD PILA y del TAD COLA sea mas eficiente.?

6.- Elaborar un programa que use el TAD PILA como una librería. El programa debe presentar un menú en pantalla con las siguientes opciones:

a) Crear una pila.
b) Insertar un elemento en una pila.
c) Eliminar un elemento de una pila.
d) Mostrar por pantalla los elementos de una pila.
e) Mostrar el tope la pila.
f) Decir si la pila está vacía.
g) Salir del programa.

7.- Usando las primitivas del TAD COLA elaborar un programa que use dicho TAD como una librería. Dicho programa debe presentar un menú en pantalla con las siguientes opciones:

a) Crear una cola.
b) Insertar un elemento en una cola.
c) Eliminar un elemento de una cola.
d) Mostrar por pantalla los elementos de una cola.
e) Mostrar el sucesor de un elemento de una cola.
f) Mostrar el antecesor de un elemento de una cola.
g) Salir del programa.

8.- Dé una implementación dinámica del TAD PILA sin usar el TAD LISTA. Recuerde implementar la estructura y las operaciones primitivas.

9.- Dé una implementación dinámica del TAD COLA sin usar el TAD LISTA. Recuerde implementar la estructura y las operaciones primitivas.

10.- Dé una implementación estática del TAD PILA usando un arreglo. ¿Qué ventajas y desventajas observa de esta implementación con respecto a la implementación dinámica?

11.- Dé una implementación estática del TAD COLA usando un arreglo. ¿Qué ventajas y desventajas observa de esta implementación con respecto a la implementación dinámica?

12.- Escriba una implementación dinámica del TAD PILA usando listas circulares. ¿Qué mejoras tiene esta implementación con respecto a las listas simples? ¿Qué desventajas observa?

13.- Escriba una implementación dinámica del TAD COLA usando listas circulares. ¿Qué mejoras tiene esta implementación con respecto a las listas simples? ¿Qué desventajas?

14.- Escriba una implementación dinámica del TAD PILA usando listas doblemente enlazadas. ¿Qué mejoras tiene esta implementación con respecto a las listas simples y a las listas circulares? ¿Qué desventajas?

15.- Escriba una implementación dinámica del TAD COLA usando listas doblemente enlazadas. ¿Qué mejoras tiene esta implementación con respecto a las listas simples y a las listas circulares? ¿Qué desventajas?

16.- Escriba una implementación dinámica del TAD PILA usando listas doblemente enlazadas circulares. ¿Qué mejoras tiene esta implementación con respecto a las otras implementaciones? ¿Qué complicaciones?

17.- Escriba una implementación dinámica del TAD COLA usando listas doblemente enlazadas circulares. ¿Qué mejoras tiene esta implementación con respecto a las otras implementaciones? ¿Qué complicaciones?

18.- En los Sistemas Distribuidos existen procesos, que permiten usar más eficientemente los recursos, cómo es el caso del proceso que controla la cola de un impresora. La impresora está conectada a uno de los computadores. Cualquiera de ellos puede enviar a travez de la red un "job de impresión", es decir un archivo a imprimir. En la mayoría de los casos estos se insertan al final de la cola y son servidos por orden de llegada, pero existe algunos trabajos que por su urgencia requieren ser impresos antes que todos los que aparecen en la cola. Se dice que el job tiene mayor prioridad que los demás. Las colas que permiten este tipo de inserción se conocen como "Colas de Prioridad" donde los job se sirven deacuerdo a la prioridad que tienen. Si hay un conjunto de procesos que tiene la misma prioridad entonces se sirven en orden de llegada.

a) Diseñe (conceptualice) e implemente en C, la estructura para el TAD JOB que contiene la información requerida para cada archivo a imprimir. Recuerde que debe escribir la estructura y las operaciones primitivas.
b) Escriba una estructura **estática** para el TAD COLAPRIO (conceptualización e implementación), con las operaciones
   - `CP_Encolar: COLAPRIO x JOB -> COLAPRIO` /*inserta en la cola deacuerdo a la prioridad del JOB*/
   - `CP_Desencolar: COLAPRIO -> JOB` /*Devuelve el primer job de la cola*/
   - `CP_Vacía:COLAPRIO -> BOOL` /* Dice si la cola está vacía */
c) Escriba una estructura **dinámica** para el TAD COLAPRIO (conceptualización e implementación), con las mismas operaciones:
d) Usando las operaciones del TAD JOB y TAD COLAPRIO escriba una función que trabaje como el servidor de la cola de impresión.

ORDEN DE ALGORITMOS (O grande)

1.- Calcular la complejidad de los siguientes trozos de código:

a)

```
X = 1;
Y = 2;
Z = 3;
```

b)

```
for (i = 0; i < 9; i++)
      a[i] = 0;
```

2.- Calcular la complejidad de la siguientes funciones:

a)

```
int fact(int num)
{
      int i, acum;
      i = 0;
      acum = 1;
      while (i < num)
      {
            i++;
            acum * = i;
      }
      return(acum);
}
```

b)

```
void prod_mat (int n)
{
      int i, j , k;
      for (i = 1; i < n; i++)
            for (j = 1; j < n; j++)
      {
                  c[i][j] = 0;
                  for (k = 1; k < n; k++)
                        c[i][j] = c[i][j] + a[i][k] * b[k][j];
            }
}
```

c)

```
void misterio (int n)
{
      int i, j , k;
      for (i = 1; i < n-1; i++);
      for (j = 1; j < n-1; j++)
            for (k = 1; k <= j; k++)
                  printf("Misterio");
}
```

d)

```
void muy_impar (int n)
{
      int i, j, x, y;
      for (i = 1; i < n; i ++)
            if impar(i)
                  for (j = i; j < n; j++)
                        x++;
```

```
                else
                      for (j = i; j < n; j++)
                            y++;
      }
```

e)

```
      int recursiva (int n)
      {
            if n < = 1
                  return(1);
            return(recursiva(n-1) + recursiva(n - 2));
      }
```

3.- Dada la función recursiva que devuelve a^b (potencia de a elevado a la b), representada por la siguiente definición:

```
      f(a,b) = 1 si b = 0
      f(a,b) = a* f(a,b-1) si b> 0
```

Codifique en C la función anterior. Calcule la complejidad de la función codificada.

4.- Para las siguientes operaciones del TAD LISTA indique cuál es el orden de la operación:

```
a) LISTA L_Crear();
b) LISTA L_InsertAnt(LISTA,ELEM);
c) LISTA L_InsertPost(LISTA,ELEM);
d) LISTA L_Eliminar(LISTA);
e) ELEM L_Info(LISTA);
f) LISTA L_PrimerLista(LISTA);
g) LISTA L_UltimoLista(LISTA);
h) void L_Siguiente(LISTA);
i) int L_Tamaño(LISTA);
j) BOOL L_Final(LISTA);
```

5.- Para las siguientes operaciones del TAD PILA indique cuál es el orden de la operación para cada una de las implementaciones de los ejercicios 3-17 de la sección pasada:

```
a) ELEM P_Top(PILA);
b) PILA P_Pop(PILA);
c) PILA P_Push(PILA,ELEM);
d) BOOL P_Empty(PILA);
```

6.- Para las siguientes operaciones del TAD COLA indique cuál es el orden de la operación para cada una de las implementaciones de los ejercicios 3-17 de la sección pasada:

```
a) COLA C_Insert(COLA,ELEM);
b) COLA C_Remove(COLA);
c) ELEM C_First(COLA);
d) BOOL C_Empty(PILA);
```

7.- Para cada una de las operaciones del TAD LISTA implementadas en el ejercicio 1 de la sección anterior, calcule el orden de la operación.

RECURSIÓN EN TAD.

1.- Para cada una de las siguientes operaciones del TAD LISTA escribir una implementación recursiva y diga el orden de la operación:

a) `LISTA L_Concatenar(LISTA, LISTA);` Devuelve una lista con la concatenación de las dos listas dadas
b) `LISTA L_Invertir(LISTA);` Devuelve una nueva lista con el resultado de invertir la lista original
c) `void L_Buscar(LISTA,ELEM);` Coloca la ventana en el elemento buscado. Si el elemento no está la ventana queda indefinida
d) `LISTA L_EliminarTodos(LISTA,ELEM);` Elimina todas las ocurrencias del elemento.
e) `BOOL L_Iguales(LISTA,LISTA);` Dice si dos listas son iguales (tienen la misma longitud, y los elementos son iguales en la misma posición).
f) `BOOL L_Ordenada(LISTA);` Dice si una lista está ordenada. Suponga que existe una función que dice si un elemento es menor que otro.
g) `void L_Destruir(LISTA);` Destruye el objeto lista liberando toda la memoria ocupada.
h) `BOOL L_Sublista(LISTA,LISTA);` Dice si la segunda lista es sublista de la primera.
i) `LISTA L_Agregar(LISTA,ELEM);` Agrega el elemento al final de la lista
j) `LISTA L_Simplificar(LISTA);` Elimina los elementos repetidos de la lista. Queda una sola ocurrencia de cada elemento
k) `int L_NumDiferentes(LISTA);` Devuelve el número de elementos diferentes de la lista
l) `int L_NumOcurrencias(LISTA,ELEM);` Devuelve el número de ocurrencias de un elemento dentro de la lista.
m) `LISTA L_Ordenar(LISTA);` Ordena ascendentemente la lista.
n) `LISTA L_Mezclar(LISTA,LISTA);` Devuelve la mezcla de dos lista ordenadas ascendentemente

ORDENAMIENTO

1.- Analíce el siguiente algoritmo, conocido como "Ordenamiento por inserción directa" e impleméntelo en lenguaje C. Este método se usa generalmente por los jugadores de cartas. Los items están divididos en una secuencia destino $a_1,...,a_{i-1}$ y una secuencia origen $a_i,..,a_n$. En cada paso empezando con i=2, se toma el elemento i de la secuencia origen y se transfiere a la secuencia destino insertándolo en el sitio apropiado.

```
Arreglo inicial :  44  55  12  42  94  18  06  67
i = 2              44  55  12  42  94  18  06  67
i = 3              12  44  55  42  94  18  06  67
i = 4              12  42  44  55  94  18  06  67
i = 5              12  42  44  55  94  18  06  67
i = 6              12  18  42  44  55  94  06  67
i = 7              06  12  18  42  44  55  94  67
i = 8              06  12  18  42  44  55  67  94
```

```
Algoritmo de Inserción directa
Para i←2 hasta n hacer
        x←a[i]
        a[0] ←x
        j←i-1
        Mientras x< a[j] hacer
                a[j+1] ←a[j]
                j←j-1
        a[j+1] ← x
fin
```

2.- Analíce el siguiente algoritmo, conocido como " Ordenamiento por selección directa " e impleméntelo en lenguaje C. Este método se basa en dos principios: seleccionar el artículo con cleve mínima, intercambiarlo con el primero. a continuación se repite el proceso con los items n-1,n2 etc hasta que quede un único item, el mayor.

```
Arreglo inicial :  44  55  12  42  94  18  06  67
                   06  55  12  42  94  18  44  67
                   06  12  55  42  94  18  44  67
                   06  12  18  42  94  55  44  67
                   06  12  18  42  94  55  44  67
                   06  12  18  42  44  55  94  67
                   06  12  18  42  44  55  94  67
                   06  12  18  42  44  55  67  94
```

```
Algoritmo de Selección directa

Para i←1 a n-1 hacer
        k←i
        x←a[i]
        Para j←i+1 a n hacer
                if a[j]<x entonces
                        k←j
                        x←a[j]
        a[k] ← a[i]
        a[i] ←x
fin
```

3.- Analíce el siguiente algoritmo, conocido como " Ordenamiento por intercambio directo (o método de la burbuja)" e impleméntelo en lenguaje C. La característica dominante es el intercambio de items. Se comparan items adyacentes hasta que todos estén ordenados. Se hacen repetidas pasadas sobre el arreglo hasta que el elemento de clave mínima quede en el extremo izquierdo. Si se mira el arreglo en condición vertical, los items se consideran burbujas con diferentes pesos, donde cada burbuja sube y se coloca en el peso que le corresponde

inicial	i=2	i=3	i=4	i=5	i=6	i=7	i=8
44	06	06	06	06	06	06	06
55	44	12	12	12	12	12	12
12	55	44	18	18	18	18	18
42	12	55	44	42	42	42	42
94	42	18	55	44	44	44	44
18	94	42	42	55	55	55	55
06	18	94	67	67	67	67	67
67	67	67	94	94	94	94	94

Algoritmo Burbuja

```
Para i ← 2 a n hacer
        Para j←n a i hacer
                if a[j-1]>a[j] entoncer
                        swap(a[j-1],a[j]
fin
```

4.- Analíce el siguiente algoritmo, conocido como " Método Rápido o Quicksort " e impleméntelo en lenguaje C. Se basa en el hecho de que los intercambios deben realizarse preferiblemente sobre distancias largas para que sean efectivos. Se toman arbitrariamente un elemento (x), se inspecciona el arreglo de izquierda a derecha hasta encontrar un item a_i>x y entonces se inspecciona el arreglo de derecha a izquierda hasta encontrar un item a_j<x. A continuación se itercambian los dos items y se continúa este proceso de inspección e intercambio hasta que los recorridos en ambas direcciones se encuentren en algún punto situado aproximadamente en la mitad del arreglo. Como resultado se obtiene el arreglo partido en dos: los menores que x y los mayores que x. X actúa como centinela en el proceso. Si se toma x=42 el arreglo se particiona en dos comparaciones

```
Arreglo inicial :  44  55  12  42  94  18  06  67
                   06  55  12  42  94  18  44  67
                   06  18  12  42  94  55  44  67
```

Luego se repite el proceso con cada partición. Note que es recursivo

```
Algoritmo Quicksort (izquierdo,derecho)
        i←izquierdo
        j←derecho
        x←a[(izquierdo+derechos)/2]
        hacer
                mientras a[i]<x hacer
                        i←i+1
                mientras x<a[j] hacer
                        j+1
                if i<=j entonces
                        swap(a[i],a[j])
                        i←i+1
                        j←j-1
        mientras i<=j
        if izquierdo < j entonces Quicksort(iz,j)
        if i<derecho entonces Quicksort(i,derecho)
fin
```

5.- Escriba un programa recursivo para ordenar un arreglo a del modo siguiente. Suponga que k es el índice del elemento intermedio del arreglo. Ordene los elementos hasta e incluyendo a[k]. Ordene los elementos después de a[k]. Intercale los dos subarreglos en un solo arreglo ordenado. Este método se denomina **ordenamiento por intercalación.** Ejemplo:

Entrada:

arreglo = [5,3,20,9,1,2,6,18,4]

Procedimiento:

Paso 1: k = 4
Paso 2: arreglo[k] = 1
resultado de ordenar la primera mitad del arreglo incluyendo a[k] → [1,3,5,9,20]
Paso 3:
Resultado de ordenar la segunda mitad desde a[k+1] hasta el último elemento → [2,4,6,18]
Paso 4 y salida: arreglo = [1,2,3,4,5,6,9,18,20]