PRÁCTICA 10

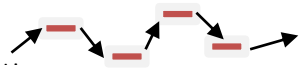
REPASO GENERAL

MAX/MIN 7sM&.

Function/Procedure

```
Persona=Record
    nombre: string[30];
    DNI: integer;
end;
```

9 5 9 2 ▮ ▮ ▮ ▮ ▮ ▮



Objetivos de la práctica:

Se espera que el alumno logre:

- **Interpretar consignas de problemas vinculados a los temas trabajados en el curso.**
- **Representar datos de acuerdo al problema y con los tipos y estructuras de datos vistos en el curso.**
- **Crear soluciones considerando la eficiencia y la secuencia lógica y algorítmica adecuada según el problema presentado.**

Nota: LEER completamente el enunciado antes de comenzar a escribir la solución. Modularizar las soluciones y definir las estructuras de datos utilizadas. Considerar que las listas pueden estar vacías y NO OLVIDE LIBERAR la memoria ocupada antes de finalizar el programa.

1. Un banco dispone de una lista en donde almacena la información de aquellos clientes que vienen a pagar impuestos. De cada cliente conoce: DNI, Nombre, Apellido, código de impuesto a pagar (1 a 9) y el monto a pagar. Se pide:

a. Realizar la atención de los clientes hasta que se recaude al menos 100.000 pesos o hasta que se terminen los clientes.

b. Al finalizar la atención informar el código de impuesto que más veces se pagó por los clientes que fueron atendidos.

c. Al finalizar la atención informar, en caso de que hayan quedado, la cantidad de clientes sin atender.

***Nota**: Asumir que cada cliente paga un solo impuesto.*

2. Se dispone de una estructura de datos con la información de personas que pagan su servicio de internet a una empresa. De cada persona se conoce nombre, apellido, categoría del servicio (1 doméstico, 2 empresa, 3 comercio, 4 sociedad de beneficencia), monto básico que debe pagar y la dirección donde se le brinda el servicio (a lo sumo 70 caracteres). El monto final a pagar por el servicio de internet se compone de un monto básico al que se le adiciona un importe extra de acuerdo a la categoría del servicio. Para ello la empresa dispone de una tabla adicional donde para cada categoría de servicio (1 doméstico, 2 empresa, 3 comercio, 4 sociedad de beneficencia) se indica el monto extra a cobrar. Se pide:

a. Calcular el monto recaudado por la empresa para cada categoría de servicio.

b. El nombre de la persona que más pago por el servicio de internet de categoría 2. En caso de que nadie haya pagado esta categoría de servicio, informar "*nadie pagó servicio de internet categoría 2*".

c. Para cada persona indicar si su dirección cumple la siguiente forma: **A % B % C** donde:

i. **A** debe ser una secuencia de letras mayúsculas de la *"A"..."G"* y caracteres dígitos pares.

ii. **%** es el carácter "%" que seguro existe.

iii. **B** debe ser exactamente las letras de **A**, en el mismo orden, donde para cada letra mayúscula de **A** aparece su minúscula en **B.**

iv. **C** debe ser una secuencia donde están una única vez, todos los caracteres dígitos que no aparecieron en **A**.

Ejemplo: DFG2A4EG % dfgaeg % 13057896 *{cumple con el patrón.}*

***Nota**: Cada persona paga un solo servicio.*

3. Se dispone de una lista con la información de los ingresantes a la Facultad en el año anterior. De cada ingresante se conoce: apellido, nombre, ciudad de origen, fecha de nacimiento (día, mes, año), si

presentó el título del colegio secundario y el código de la carrera en la que se inscribe (1: APU, 2: ATIC, 3: LI, 4: LS, 5: IC, 6: CDO). Con esta información de los ingresantes se pide que recorra la lista una vez para:

- **a.** Informar el apellido de los ingresantes cuya ciudad origen es "*Chacabuco*".
- **b.** Calcular e informar el año en que más ingresantes nacieron (asumir que los años de nacimientos de los ingresantes pueden variar entre 1975 y 2006).
- **c.** Informar la carrera con la mayor cantidad de inscriptos.
- **d.** Eliminar de la lista aquellos ingresantes que no presentaron el título.

4. Una empresa de la ciudad de La Plata, que realiza entregas de paquetes dentro del casco urbano, está interesada en procesar la información de sus paquetes. De cada paquete se conoce la fecha de envío, si pudo ser entregado al destinatario y la dirección que está compuesta por los campos: calle (de 1 a 122), número, piso y departamento.
Se dispone de una lista con los envíos del año pasado. Se requiere procesar la lista recorriéndola una sola vez para:

- **a.** Informar la cantidad de envíos realizados para cada calle.
- **b.** Informar el nombre del mes con mayor cantidad de paquetes enviados.
- **c.** Eliminar de la lista de envíos, aquellos que no fueron entregados al destinatario.
- **d.** Generar 10 nuevas listas con los envíos de las calles 11 a la 20. Cada lista debe estar ordenada por el número de la dirección.

5. Se desea modelar un nuevo juego que consiste en un tablero de 15 filas por 15 columnas y que se juega de a 2 jugadores. Cada casillero del tablero contiene un número del 0 al 10 y un valor que indica si la celda fue utilizada a lo largo de la partida.
La partida consiste de 10 rondas donde en cada una de las rondas los 2 jugadores eligen una celda del tablero. Las celdas elegidas en cada ronda se analizan y obtiene un punto el jugador que eligió la celda que contiene el mayor número. En caso de que ambos jugadores hayan, en una ronda, elegido una celda que contiene el mismo número ninguno suma puntos. En caso de que algún jugador seleccione una celda que ya haya sido utilizada a lo largo de la partida, dicho jugador pierde la ronda y se le suma un punto a su oponente. En caso de que en una ronda ambos jugadores seleccionen celdas que ya hayan sido utilizadas a lo largo de la partida, ninguno suma puntos. Se pide:

- **a.** Implemente un módulo que inicialice el tablero de juego. El módulo recibe el tablero y una lista con los valores que va a tener cada celda del tablero. Cada elemento de la lista almacena: fila, columna y número para la celda (0 a 10). Además, cada celda se debe marcar como no utilizada. En caso de que el valor para una celda no venga en la lista, dicha celda se inicializara con el valor 0 (cero).
- **b.** Implemente un módulo que reciba el tablero y simule el juego. Los valores que cada jugador va eligiendo son leídos de teclado. Al finalizar el juego, informar los puntos obtenidos por cada jugador y cuál es el ganador.

6. La facultad de informática decide abrir 10 cursos de actualización (cuyos códigos se numeran del 1 al 10) para sus graduados. A cada inscripto se le solicita la siguiente información: DNI, apellido, nombre, edad y el código del curso al que se desea inscribir (una persona se puede inscribir solo a un curso). La facultad dispone de una estructura en la que se almacena para cada código de curso el cupo máximo de personas que pueden inscribirse a ese curso. Se pide:

- **a.** Simular el proceso de inscripción de los graduados a los cursos. El proceso de inscripción finaliza cuando llega un graduado con apellido "zzz". Para cada inscripción se debe controlar que la cantidad de graduados ya inscriptos no supere el cupo máximo que cada curso tiene. En caso de que el curso solicitado se encuentre completo se debe informar que no hay lugar disponible en dicho curso.
- **b.** Una vez finalizada la inscripción:
  - i. Informar el código de aquellos cursos cuyo cupo máximo no se completó.
  - ii. Calcular e informar la cantidad de inscriptos al curso con código 2 que tienen entre 30 y 35 años.

7. Una empresa de gas tiene que revisar los montos facturados a sus clientes durante el mes Octubre. De cada factura se conoce el código de cliente, categoría de consumo (1 a 10), metros cúbicos (m^3) consumidos, monto total de la factura y cada monto facturado durante los 12 meses del año anterior. Se dispone de una lista con la información anterior y de una estructura que se accede por categoría de consumo al nombre de esta. Procesar la lista de facturas recorriéndola una sola vez para:

a. Separar la lista por las 3 condiciones, ordenándolas por código de cliente:
   - sin revisión: facturas cuyo monto es menor que el promedio del año anterior.
   - revisión: facturas cuyo monto es mayor que el promedio del año anterior y tiene más de 1000 m^3 consumidos.
   - refacturación: facturas cuyo monto es mayor que el promedio del año anterior y tiene menos de 1000 m^3 consumidos.

b. Calcular e informar el nombre y los m^3 consumidos para todas las categorías de consumo.

8. Una empresa de comidas rápidas ofrece 12 combos (comida, bebida y postre) y está interesada en realizar un análisis semanal de los pedidos que hacen sus clientes. De cada pedido se conoce el código del combo (de 1 a 12), día de la semana que se pidió (1=lunes, 2=martes, …, 7=domingo) y precio (sin descuento). Además, la empresa aplica, a manera de promoción, distintos descuentos sobre los combos según el día de la semana.
Se **dispone** de una lista de los pedidos realizados en la semana pasada. También se **dispone** de una estructura que se accede por código de combo a la descripción del mismo y otra estructura que se accede por el día para obtener el descuento (en $) a aplicar ese día. Se requiere procesar la lista de ventas recorriéndola **una sola vez** para:

a) Calcular e informar el monto facturado cada día, incluyendo el descuento.
b) Calcular e informar la **descripción** y monto facturado de los 2 combos más pedidos.
c) Generar 4 nuevas listas conservando su **orden original** e incluyendo los pedidos:
   1) Cuyo precio sea menor a $100.
   2) Cuyo precio sea mayor o igual a $100.
   3) Cuyo código de combo esté entre 1 y 3.
   4) Cuyo código de combo sea mayor o igual a 4.

9. Una empresa turística ofrece servicios de venta de paquetes a diferentes lugares del país. La misma cuenta con dos estructuras de datos: una estructura en donde almacena la información de sus clientes, donde de cada cliente se sabe su número de cliente, apellido, nombre, DNI y domicilio; y otra estructura en donde almacena la información de los 300 paquetes de turismo que dispone para la venta. Dichos paquetes se encuentran codificados del 1 al 300 y de cada uno se sabe el nombre del destino del paquete y cantidad de plazas disponibles para el mismo. La estructura con la información de los clientes **no** tiene orden. Realice un programa que:

a. Simule la venta de paquetes. De cada solicitud de compra de un paquete se lee el DNI de la persona que desea hacer la compra, el código de paquete que solicita y la cantidad de plazas que solicita. Una venta **no** podrá ser realizada si ocurre alguno de los siguientes motivos: que el DNI no exista en la estructura de clientes o que la cantidad de plazas libres del paquete solicitado no alcance para cubrir la cantidad de plazas solicitadas. En caso de que la venta pueda ser realizada se debe actualizar la cantidad de plazas disponibles del paquete que se vende. La venta de paquetes finaliza cuando se lee un DNI igual a 0. Una persona solicita un único paquete.

b. Una vez finalizada la venta de paquetes se pide:
   i. Realice un módulo que calcule e informe los dos códigos de paquetes con mayor cantidad de plazas disponibles.
   ii. Informe la cantidad de ventas que no pudieron ser realizadas porque el DNI no correspondía a un cliente y la cantidad de ventas que no se pudieron realizar porque la cantidad de plazas solicitadas era mayor que la cantidad de plazas libres del paquete.

10. Se dispone de cierta cantidad de mensajes de texto de una conocida red social. Cada mensaje cuenta con un texto, la cantidad de "me gusta" y la cantidad de veces que fue compartido con otros usuarios. Se dispone de un arreglo con capacidad máxima para 1000 mensajes ordenado por cantidad de "me gusta". Se pide:

a. implemente un módulo que reciba el arreglo de mensajes anterior y un mensaje de texto y lo agregue de manera que mantenga el orden.

b. implemente un módulo que reciba un arreglo de mensajes y lo reordene por la cantidad de veces que fue compartido.

Nota: *recuerde que el arreglo no necesariamente está completo y que no debe agregar mensajes si se alcanzó la capacidad máxima.*

11. ***Letter Dice*** es un juego para formar palabras con dados de letras donde participan 8 jugadores. Cada jugada se identifica con el número de jugada, el código del jugador (de 1 a 8), la palabra creada (hasta 15 letras en mayúsculas) y su longitud.

Se dispone de una estructura de datos que permite acceder al puntaje de una letra a través del carácter de esta y se requiere:

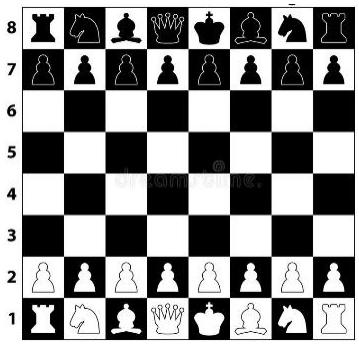
a) Leer de teclado para crear una lista con las jugadas, ordenada por número de jugada. Para cargar las palabras tenga en cuenta que cada letra se introduce en mayúscula sin necesidad de validarla. La carga finaliza cuando se ingresa 0 como numero de jugada.

Una vez creada la lista, recorrerla una única vez para:

b) Eliminar todas las jugadas del jugador número 3 ya que fue descalificado por hacer trampa en el juego.

c) Informar los 2 jugadores con más puntos en la partida. El puntaje de una palabra se obtiene sumando los valores de sus letras, según la estructura de puntaje.

12. Construir un programa que simule un juego de ajedrez. Cada pieza se identifica con un número (1: Alfil, 2: Caballo, 3: Peón, 4: Torre, 5: Reina y 6: Rey) y un color (1: blanco y 2: negro). Con estos valores se construye un identificador de la pieza que consiste en un número de 2 dígitos donde la decena corresponde a la pieza y la unidad al color, así por ejemplo el número 52 corresponde a reina negra y 21 corresponde a caballo blanco.



El tablero posee 64 casillas (8x8) donde cada una almacena el identificador de pieza. En particular un cero en una casilla indica que está vacía mientras que otro valor indica que hay una pieza.

Un movimiento implica desplazar una pieza de una casilla origen a otra casilla destino, dejando la casilla origen vacía. Cada casilla de un movimiento se identifica con una fila y una columna del tablero.

Se **dispone** de un módulo que inicializa el tablero y de una lista de movimientos que realiza cada jugador de forma alternada hasta terminar el juego. Se pide recorrer la lista de movimientos una única vez para:

a) implementar los módulos ***ObtenerPieza*** y ***ObtenerColor*** que reciben un identificador de pieza y retornan el número de pieza y color respectivamente. Utilizando los módulos anteriores implementar los módulos ***PiezaEsBlanca***, ***PiezaEsNegra***, ***PiezaEsRey*** y ***HayPieza*** que reciben un identificador de pieza y retornan verdadero o falso según la pieza coincida con lo indicado en el nombre del módulo.

b) Generar 2 listas (una por color) para almacenar las piezas comidas. Si la casilla destino de un movimiento está ocupada con una pieza, ésta debe quedar en la lista asociada con su color. Conservar el orden en que se comen las piezas dentro de cada lista.

c) Informar la pieza que más movimientos tuvo independientemente del color de esta.

d) Informar el resultado del juego. Para esto implemente un módulo que reciba el tablero y retorne el resultado del juego (0=empate, 1=blanco, 2=negro). Tenga en cuenta que si hay 2 reyes en el tablero fue empate y si solo queda uno es el ganador.

Nota: modularice adecuadamente y libere la memoria de las estructuras dinámicas.

13. Una plataforma de streaming desea implementar un sistema para gestionar su catálogo de series vistas por los usuarios. Cada vez que un usuario finaliza una serie, se registra la siguiente información: nombre de la serie, nombre del usuario y calificación del usuario a la serie (1..5). Se lee información de los usuarios que finalizaron series hasta que llega el nombre de usuario “ZZZ”. Se pide:

a. Crear una lista doblemente enlazada con los registros de series vistas ordenada por el nombre de la serie.

b. Recorrer la lista una sola vez para:

i. Calcular la calificación que mayor cantidad de series recibe por parte de los usuarios.

ii. Generar e imprimir una lista simple con el promedio de calificaciones para cada nombre de serie.

14. Dada la siguiente declaración de tipos, complete cada sección indicada con [completar]agregando el código necesario para realizar las operaciones solicitadas:

```
1.  Program Ejercicio14P10;
2.  Type
3.     ListaDE =^nodo;
4.     nodo = record
5.        dato:integer;
6.        ant: ListaDE;
7.        sig: ListaDE;
8.     end;
9.     punteros = record
10.       pri1, pri2: ListaDE;
11.    end;
12.
13. Var p: punteros;
14.
15. Procedure AgregarAdelante([completar]; [completar]);
16. var nuevo: [completar];
17. begin
18.   New(nuevo);
19.   nuevo^.dato := [completar];
20.   nuevo^.[completar] := nil;
21.   nuevo^.[completar] := nil;
22.   if p.[completar] = nil then begin // caso de lista vacía
23.     p.[completar] := nuevo;
24.     p.[completar] := nuevo;
25.   end
26.   else begin // caso de lista NO vacía
27.     nuevo^.sig := p. [completar];
28.     p.pri1^.ant := nuevo;
29.     lista.pri1 := [completar];
30.   end;
31. end;
32.
33. procedure ImprimirIzquierdaADerecha([completar]);
34. var act: [completar];
35. begin
36.   act := [completar];
37.   while act <> [completar] do begin
38.     Write(act^.[completar]);
39.     act := act^.[completar];
40. end;
```

```
41. procedure ImprimirDerechaAIzquierda([completar]);
42. var actual: [completar];
43. begin
44.   actual := [completar];
45.   while actual <> [completar] do begin
46.     Write(actual^.[completar]);
47.     actual := actual^.[completar];
48. end;
```