PRÁCTICA 6
Arreglos: Vectores

Aclaración: los ejercicios marcados con * se recomiendan realizar en forma obligatoria durante la semana correspondiente a la realización de la práctica, acorde a lo estipulado en el cronograma. Además, se recomienda consultar la solución realizada con los ayudantes durante la práctica y de ser posible, escribir el programa en Lazarus Pascal y probar su ejecución. El resto de los ejercicios es necesario realizarlos como parte del estudio y preparación para el parcial.

Objetivos de la práctica:
Se espera que el alumno logre:

- **Realizar adecuadas representaciones de objetos del mundo real con vectores.**
- **Aplicar las operaciones básicas de vectores tales como asignación de valores, manejo de dimensión lógica y física, en los casos que corresponda, etc.**
- **Utilizar los vectores como contadores en diferentes ejercicios modelos.**
- **Realizar operaciones clásicas sobre vectores como insertar y borrar elementos.**
- **Utilizar vectores de registros y ejercitar su manipulación a través de las operaciones requeridas en problemas sencillos.**

1. * Realizar un programa que implemente y use:

a) un módulo que cargue un vector con 150 números enteros y lo retorne.

b) un módulo que reciba el vector generado en a) y retorne el promedio de sus valores y el porcentaje de números negativos y positivos.

c) un módulo que reciba el vector generado en a) y dos valores enteros que representan un rango de valores y que retorne la cantidad de elementos del vector que estén dentro de ese rango.

d) un módulo que reciba el vector generado en a) y retorne los dos valores mínimos entre sus elementos junto con la posición donde se encuentran.

2. Una estación de servicio tiene 5 surtidores. Se requiere implementar un programa que lea datos de la carga de combustible por teclado (monto cargado y número de surtidor). La entrada de datos termina cuando se ingresa el surtidor 0. Finalizada la carga se pide calcular e informar:

a) Monto total vendido por cada surtidor.
b) Surtidor que más vendió.
c) Promedio de venta entre todos los surtidores.

3. * Se requiere analizar un fragmento de código fuente que contiene 1024 caracteres. Realizar un programa que lea de teclado y cargue un vector los caracteres por teclado. Finalizada la carga informar:

a) La cantidad y porcentaje de consonantes sobre el total de caracteres alfabéticos.
b) La cantidad y porcentaje de caracteres que son: dígitos, letras mayúsculas y letras minúsculas sobre el total de caracteres ingresados.

4. Realizar todas las modificaciones necesarias al ejercicio **3** para el caso que la carga de caracteres termine cuando se lea el carácter '.' o hasta que se almacenen los 1024 caracteres (tener en cuenta que en el vector se pueden cargar como máximo 1024 valores). **Declare una constante que represente el valor de fin de la lectura desde teclado (en este caso el punto).**

5. Definir un tipo de dato que permita almacenar una secuencia de 64 valores binarios e implementar módulos que realicen las operaciones AND, OR y NOT junto con dos operandos o uno, según corresponda.

6. *Una cátedra dispone de información de sus 1000 estudiantes. De cada estudiante se conoce: número de legajo, apellido, nombre y cantidad de asistencias a clase. Dicha información se encuentra ordenada por apellido de manera ascendente. Se pide que implemente:

a) Un módulo que retorne la posición del estudiante con un número de legajo recibido por parámetro o debe retornar -1 si no existe.

b) Un módulo que informe apellido, nombre y número de legajo de todos los estudiantes cuyo número de legajo es múltiplo de un número que se recibe como parámetro.

c) Un módulo que retorne la cantidad de estudiantes con cantidad de asistencias a clase en 0.

Nota: Realizar el programa principal que invoque los módulos desarrollados en los incisos previos.

7. Se dispone de un vector con **a lo sumo** 150 nombres de flores. Realizar un programa que lea desde el teclado un nombre de una flor e informe:

a) la posición del vector que coincida con este nombre.

b) Ídem **a)** pero asumiendo que los nombres están ordenados.

Nota: Tener en cuenta que dicho nombre puede no existir.

8. * Simular el funcionamiento de un **conjunto de caracteres** de la 'a' a la 'z' utilizando un arreglo. Defina un tipo de datos adecuado e implemente módulos que realicen las operaciones de unión y diferencia de dos conjuntos y una función que permite determinar si una letra pertenece al conjunto.

Nota: realice los chequeos correspondientes en cada módulo para procesar solo letras.

9. * Implementar 2 versiones de un módulo que cuente la cantidad de palabras que hay en un arreglo de caracteres con un máximo de 130 elementos:

a) Implementar una versión del módulo que tenga en cuenta una dimensión lógica para indicar la posición del último carácter.

b) Implementar una versión del módulo que asuma que el último carácter es punto.

Nota: en ambos casos, por seguridad, realice las verificaciones para no pasarse de la longitud máxima.

10. Construir un programa que implemente y use:

a) un módulo que lea números desde el teclado y los almacene en un vector hasta ingresar el valor 9999 (asuma como máximo 80 números).

b) un módulo que **informe** los números almacenados en las posiciones impares.

c) un módulo que **retorne** la posición en que se encuentra el primer elemento par. Si no hay ningún componente par deberá devolver el valor 0.

11. *Se dispone de un módulo que carga un vector con a lo sumo 600 nombres de personas, **ordenados** de forma ascendente. Implemente módulos que reciban dicho vector y permitan:

a) Devolver la posición en la que se encuentra una persona cuyo nombre se recibe como parámetro. *Tener en cuenta que dicha persona puede no existir.*

b) Insertar un nombre recibido en el vector conservando su orden. *Recuerde validar el espacio.*

c) Eliminar un nombre recibido del vector en caso de existir. *Considere que no hay nombres repetidos.*

d) Modifique c) considerando ahora que puede haber repetidos (elimine todas las ocurrencias).

12. *Construir un programa que lee desde teclado una secuencia de números reales hasta que se introduzca el 50. Informar los 8 números mayores de la secuencia.

Nota: Implemente la solución sin almacenar TODOS los números leídos.

13. La empresa “Documentado” se encuentra desarrollando un sistema de versionado de un documento. Para esto, se guardan hasta un máximo de 40 versiones del documento. De cada versión se conoce: el título, un enlace al texto completo y la fecha en formato UNIX (un entero). Esta información se encuentra ordenada por fecha. Se pide desarrollar:

a) Un módulo que reciba las últimas versiones de un documento, y una fecha en formato UNIX, y retorne la versión correspondiente a esa fecha. En caso de no encontrar una versión retornar una versión con título “Inexistente”.

b) Modificar el módulo del punto a) para que, en caso de no encontrar la versión, devolver la versión con la fecha más cercana.

14. El juego de la Oca es un juego de mesa clásico que implica tirar dados y mover fichas por un tablero con casillas especiales. En este ejercicio, se pide implementar una versión simplificada del juego de la Oca para 2 jugadores. El juego se desarrolla en un tablero lineal de 63 casillas, cada una de las cuales puede tener uno de los siguientes tipos: "libre", "impulso", "retroceso" o "trampa". Un casillero “libre” no produce ninguna acción; “impulso” indica un número que el jugador debe avanzar a partir del casillero actual; “retroceso” indica un número el jugador debe retroceder a partir del casillero actual; mientras que “trampa” indica “perder el turno”. Una "prenda" solo se aplica si el jugador llega a ella mediante un tiro de dado, no a través de casillas de "impulso" o "retroceso".

Mecánica del Juego:

1. El tablero se debe inicializar de manera dinámica antes de cada partida, asignando a cada casilla su tipo y prenda correspondiente. **Implementa una estrategia para asignar a cada casillero su “prenda”.** Por ejemplo, los casilleros impares pueden ser de retroceso, los pares de avance, los casilleros múltiplos de 5 pueden ser de tipo "perder turno", etc.
2. Al comenzar el juego, el programa **debe solicitar el nombre** de los dos jugadores.
3. En cada turno, el programa debe imprimir en pantalla el nombre del jugador cuyo turno es, el valor del dado arrojado y la prenda del casillero en el que se encuentra (si la hay).
4. El juego continúa hasta que uno de los jugadores llega a la casilla 63, "el jardín de la oca". Si un jugador cae en un casillero con un valor mayor a 63, debe pasar su turno sin avanzar.
5. El juego finaliza cuando un jugador llega al "jardín de la oca", y el programa debe anunciar al ganador.

Módulo tirarDado(): Implementa un módulo llamado tirarDado() que devuelva aleatoriamente un número del 1 al 6. Este módulo se utilizará para determinar el avance de los jugadores en cada turno.

Consideraciones para futuras variaciones: Diseña tu implementación de manera que permita futuras expansiones, como incrementar o decrementar la cantidad de casilleros, aceptar más de 2 jugadores y determinar quién comienza mediante una tirada de dado inicial.

Requisitos:

1. Asegúrate de que el juego siga las reglas y mecánicas descritas anteriormente.
2. Comenta tu código de manera clara para explicar cómo funciona y cómo podrían realizarse futuras expansiones.