ESTRUCTURA DE DATOS

Trabajo Práctico Nº 4

Tema: TDA Simple

Apellido y Nombre: ..... Fecha:...../...../.....

EJEMPLOS

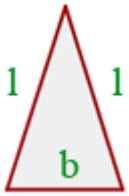
Un triángulo es un polígono de 3 lados que, en función de sus lados, puede clasificarse en equilátero, isósceles o escaleno. Considerando esto, defina e implemente:

Triángulo Equilátero

Perímetro

$$p = 3 \times lado$$

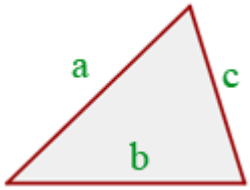
Triángulo Isósceles



Perímetro

$$p = 2 \times lado + b$$

Triángulo Escaleno



Perímetro

$$p = a + b + c$$

Semiperímetro $sp = \frac{p}{2}$

Área del triángulo $a = \sqrt{sp \times (sp - lado_1) \times (sp - lado_2) \times (sp - lado_3)}$
(fórmula de Herón)

a) el TDA triángulo utilizando arreglos,

b) el TDA triángulo utilizando registros y

c) las operaciones *crear triángulo*, *calcular perímetro* y *calcular área* para las definiciones de los ítems a) y b).

Especificación del TDA Triángulo

El TDA triángulo comprende 3 valores reales que representan las longitudes de los lados de un triángulo. Sobre este tipo pueden definirse las operaciones:

- Crear triángulo: dados 3 valores reales, devuelve un triángulo.
- Calcular perímetro: dado un triángulo, devuelve el valor de su perímetro.
- Calcular área: dado un triángulo, devuelve el valor de su superficie.

Implementación del TDA Triángulo con Arreglos

De acuerdo a la especificación, el tipo triángulo se representa mediante 3 valores reales que corresponden a las longitudes de los lados de la figura. Para implementar el concepto es necesario elegir una estructura de datos que permita almacenar y manipular estos 3 valores.

A continuación se presenta la implementación del tipo triángulo realizada mediante arreglos. En este caso, se utiliza un arreglo de 3 posiciones reales que representan los lados del triángulo; asimismo se desarrollan las operaciones correspondientes a la implementación elegida.

Definición de la estructura de datos que representa al tipo triángulo:

En Pseudodcódigo

```
t_triangulo=ARREGLO [1..3] de REALES
```

En C/C++

```
typedef float t_triangulo[3];
```

Operaciones definidas sobre el tipo triángulo:

```
En Pseudodcódigo
PROCEDIMIENTO crear(E/S t:t_triangulo)
INICIO
  ESCRIBIR "Ingrese lados: "
 LEER t[1],t[2],t[3]
FIN
```

```
En C/C++
void crear(t_triangulo &t)
{ cout << "Ingrese lados: ";
  cin >> t[0] >> t[1] >> t[2];
}
```

La operación *crear*, según se indicó en la especificación, permite crear un dato triángulo a partir de 3 valores. Por ello, esta operación se implementa mediante un *procedimiento* que permite almacenar los valores de longitud ingresados por el usuario.

```
FUNCIÓN perimetro(E t:t_triangulo):REAL
INICIO
  perimetro←t[1]+t[2]+t[3]
FIN
```

```
float perimetro(t_triangulo t)
{
  return t[0]+t[1]+t[2];
}
```

La operación *perímetro*, según se indicó en la especificación, permite calcular el perímetro de un triángulo (un valor real). Por ello, esta operación se implementa mediante una *función* real que recibe como entrada un dato tipo triangulo. La función utiliza los valores almacenados en la estructura para realizar el cálculo (fórmula general) y generar el resultado de salida.

```
FUNCIÓN area(E t:t_triangulo):REAL
VARIABLES
  sp:REAL
  a:REAL
INICIO
  sp←perimetro(t)/2
 a←sp*(sp-t[1])*(sp-t[2])*(sp-t[3])
 a←a^0.5
 area←a;
FIN
```

```
float area(t_triangulo t)
{ float sp;
  float a;
  sp=perimetro(t)/2;
  a=sp*(sp-t[0])*(sp-t[1])*(sp-t[2]);
  a=pow(a,0.5);
  return a;
}
```

La operación *area*, según se indicó en la especificación, permite calcular la superficie de un triángulo (un valor real). Por ello, esta operación se implementa mediante una *función* real que recibe como entrada un dato tipo triangulo. La función utiliza los valores almacenados en la estructura para realizar el cálculo (fórmula de Herón) y generar el resultado de salida.

Implementación del TDA Triángulo con Registros

Como ya se mencionó, para implementar el concepto triángulo es necesario elegir una estructura de datos que permita almacenar y manipular estos 3 valores. Una alternativa a la implementación presentada en el ejemplo anterior, puede plantearse con estructuras de registro.

A continuación se muestra la definición del tipo triángulo y, a modo ilustrativo, las operaciones *crear* y *area* adaptadas a la nueva implementación.

Definición de la estructura de datos que representa al tipo triángulo:

```
En Pseudodcódigo
t_triangulo=REGISTRO
             a:REAL
             b:REAL
             c:REAL
           FIN_REGISTRO
```

```
En C/C++
typedef struct t_triangulo {
                              float a;
                              float b;
                              float c;
                              };
```

Operaciones definidas sobre el tipo triángulo:

```
En Pseudodcódigo
PROCEDIMIENTO crear(E/S t:t_triangulo)
INICIO
  ESCRIBIR "Ingrese lados: "
 LEER t.a,t.b,t.c
FIN

FUNCIÓN area(E t:t_triangulo):REAL
VARIABLES
  sp:REAL
  a:REAL
INICIO
  sp←perimetro(t)/2
 a←sp*(sp-t.a)*(sp-t.b)*(sp-t.c)
 a←a^0.5
 area←a;
FIN
```

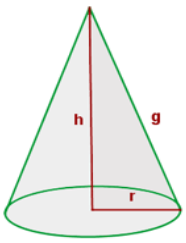
```
En C/C++
void crear(t_triangulo &t)
{ cout << "Ingrese lados: ";
  cin >> t.a >> t.b >> t.c;
}

float area(t_triangulo t)
{ float sp;
  float a;
  sp=perimetro(t)/2;
  a=sp*(sp-t.a)*(sp-t.b)*(sp-t.c);
  a=pow(a,0.5);
  return a;
}
```

Como puede observarse en las operaciones no se modificaron los datos que se reciben o los resultados que se generan, ya que sin importar la estructura de datos elegida para la implementación el concepto y las operaciones no varían. Los cambios sólo se reflejan en la forma en la que se accede a los datos almacenados, ya sean las posiciones de un arreglo o los campos de un registro.

EJERCICIOS

1) Un cono circular recto es un cuerpo geométrico que tiene una base circular y un eje como vértice que es perpendicular a la base. El eje es también la altura del cono. Considerando esto, implemente:
   a) el TDA cono utilizando arreglos,
   b) el TDA cono utilizando registros y
   c) las operaciones *crear cono*, *calcular generatriz*, *calcular área* y *calcular volumen* para las definiciones a) y b).



h: altura

r: radio

g: generatriz

$$g^2 = r^2 + h^2$$ **generatriz**

$$A_L = \pi \cdot r \cdot g$$ **área lateral**

$$A_T = \pi \cdot r \cdot (g + r)$$ **área total**

$$V = \frac{\pi \cdot r^2 \cdot h}{3}$$ **volúmen**

Nota: CONSIDERE QUE LA OPERACIÓN *CREAR CONO* COMPRUEBA QUE LOS VALORES INGRESADOS PARA DEFINIR EL *CONO* NO SEAN NEGATIVOS O CERO.

2) Un triángulo es un polígono de 3 lados que, en función del valor de sus lados, puede clasificarse en equilátero, isósceles o escaleno. En particular si el triángulo tiene un ángulo recto se denomina triángulo rectángulo y verifica el conocido teorema de Pitágoras. Defina e implemente el TDA triángulo con las siguientes operaciones:
   - Crear triángulo: dados 3 valores reales, devuelve un triángulo.
   - Validar triángulo: dado un triángulo, determina que el valor de sus lados sea correcto. Téngase en cuenta que para que un triángulo exista (el valor de sus lados sea válido) debe verificarse que la suma de cualquier par de sus lados sea mayor que el lado restante.
   - Tipo: identifica el tipo de triángulo (equilátero, isósceles o escaleno).
   - Pitágoras: determina si el triángulo es un triángulo rectángulo.

   Defina el *TDA triángulo* utilizando para su implementación a) ARREGLOS y b) REGISTROS.

3) Considerando que el concepto de fecha tiene 3 componentes (día, mes y año), implemente el TDA fecha con las siguientes operaciones:
   - Crear Fecha: dados 3 valores enteros, devuelve una fecha.
   - Validar Fecha: dada una fecha, determina si ésta contiene datos válidos.
   - Igualdad Fechas: dadas 2 fechas, determina si son iguales o no realizando la comparación componente a componente.
   - Comparar Fechas: dadas 2 fechas, determina si la primera es mayor que la segunda, la segunda es mayor que la primera o si son iguales. La comparación se realiza componente a componente.
   - Bisiesto: dada una fecha, determina si la fecha corresponde a un año bisiesto o no.
   - Días: dada una fecha, se determina su equivalente en días (a partir del 01 de enero del año correspondiente).
   - Mostrar fecha: dada una fecha, presenta su contenido por pantalla.

   Defina el *TDA fecha* utilizando para su implementación a) ARREGLOS y b) REGISTROS.

4) Un número racional puede comprenderse como el cociente de 2 números enteros, donde el primero de ellos representa el numerador y el segundo el denominador, siendo el denominador siempre distinto de cero. Sobre números racionales pueden aplicarse las siguientes operaciones:
   - Crear Racional: dados 2 números enteros, devuelve un racional.
   - Igualdad de Racionales: dados 2 racionales retorna verdadero solamente si los racionales son iguales.
   - Sumar Racionales: dados 2 racionales, devuelve otro racional que es la suma de los dados.
   - Restar Racionales: dados 2 racionales, devuelve otro racional que es la diferencia de los dados.
   - Producto Racionales: dados 2 racionales, devuelve otro racional que es el producto de los dados.
   - Cociente Racionales: dados 2 racionales, devuelve otro racional que es el cociente de los dados.
   - Potencia Racionales: dado un valor racional y un exponente entero positivo o cero, devuelve otro racional a cuyo numerador y denominador se aplicó la potencia.
   - Decimal: dado un número racional, devuelve el valor decimal correspondiente.
   - Mostrar Racional: dado un valor racional, presenta su contenido por pantalla.

   Considerando la definición y operaciones anteriores, implemente el *TDA racional* utilizando a) ARREGLOS y b) REGISTROS.

5) El concepto de polinomio se puede entender como una expresión algebraica entera, donde las letras y coeficientes están relacionados únicamente por adición, sustracción, multiplicación y potenciación con exponente natural. Las operaciones definidas para polinomios son:
   - Crear Polinomio: dados 3 números enteros, devuelve un polinomio.
   - Sumar Polinomios: dados 2 polinomios, devuelve otro polinomio que es la suma de los dados
   - Restar Polinomios: dados 2 polinomios, devuelve otro polinomio que es la diferencia de los dados.
   - Producto Polinomio/escalar: dados un polinomio y un escalar, devuelve otro polinomio que es el producto de los dados.
   - Cociente Polinomio/escalar: dados un polinomio y un escalar, devuelve otro polinomio que es el producto de los dados.
   - Grado de Polinomio: dado un polinomio, devuelve el grado del mismo.
   - Raíces de Polinomio: dado un polinomio, devuelve el valor de sus raíces si éstas existen.
   - Mostrar Polinomio: dado un polinomio, presenta su contenido por pantalla.

Considerando la definición y operaciones anteriores, implemente el TDA polinomio teniendo en cuenta que sólo se trabajará con polinomios de la forma:

$$P = a \times x^2 + b \times x + c$$

Para la implementación utilice a) ARREGLOS y b) REGISTROS.

6) Un número complejo puede ser definido como un par ordenado de números reales, donde la primera componente representa la parte real del número complejo mientras que la segunda componente corresponde a la parte imaginaria. Sobre números complejos pueden aplicarse las siguientes operaciones:

- Crear Complejos: dados 2 números reales, devuelve un complejo.
- Igualdad de Complejos: dados 2 complejos devuelve un valor booleano que es verdadero solamente si los complejos son iguales (la comparación se realiza componente a componente).
- Sumar Complejos: dados 2 complejos, devuelve otro complejo que es la suma de los dados.

$$(a, b) + (c, d) = (a + c\ , b + d)$$

- Restar Complejos: dados 2 complejos, devuelve otro complejo que es la diferencia de los dados.

$$(a, b) - (c, d) = (a - c, b - d)$$

- Producto Escalar: dados un complejo y un valor escalar, devuelve otro complejo cuyas componentes se multiplicaron por el escalar.

$$(a, b) \times N = (a \times N, b \times N)$$

- Producto Complejos: dados 2 complejos, devuelve otro complejo que es el producto de los dados.

$$(a, b) \times (c, d) = (a \times c - b \times d, a \times d + b \times c)$$

- Conjugado Complejo: dado un complejo, devuelve su conjugado (la parte imaginaria cambia de signo).

$$\overline{(a, b)} = (a, -b)$$

- Módulo Complejo: dado un complejo, devuelve el valor de su módulo.

$$|(a, b)| = \sqrt{a^2 + b^2}$$

- Mostrar Complejo: dado un valor complejo, presenta su contenido por pantalla.

Implemente el *TDA complejo* y sus operaciones utilizando a) ARREGLOS y b) REGISTROS.

7) Un conjunto se define como una colección homogénea de elementos (no repetidos) sobre el que pueden aplicarse las siguientes operaciones:

- Crear Conjunto: genera un conjunto vacío.
- Agregar elementos: dado un conjunto y un elemento permite agregar éste último al conjunto (sin repeticiones).
- Intersección Conjuntos: dados 2 conjuntos, devuelve otro conjunto que contiene sólo los elementos comunes a los conjuntos dados.
- Unión Conjuntos: dados 2 conjuntos, devuelve otro conjunto que contiene todos los elementos (no repetidos) de los conjuntos dados.
- Pertenencia: dados un conjunto y un elemento, determina si el valor se encuentra presente o no en el conjunto.
- Diferencia Conjuntos: dados 2 conjuntos, devuelve otro que contiene los elementos del primer conjunto que no se encuentran en el segundo.
- Mostrar Conjunto: dado un conjunto, presenta su contenido por pantalla.

Considerando esto, implemente el *TDA conjunto* utilizando a) ARREGLOS y b) LISTAS ENLAZADAS.

8) Un programa es, básicamente, un conjunto de instrucciones y datos que permiten procesar información para resolver diversas situaciones o problemas. Entre los tipos de datos más utilizados en programación se encuentran las cadenas de caracteres que permiten almacenar colecciones de valores alfanuméricos. Considerando esto, defina el *TDA tcadena* e implemente las siguientes operaciones:

- Generar Cadena Vacía: genera una cadena vacía o nula.
- Leer Cadena: permite cargar una cadena de caracteres con la entrada del usuario.
- Longitud de Cadena: permite determinar la cantidad de caracteres almacenados en una cadena.

- Comparar Cadenas: dadas 2 cadenas de caracteres determina si éstas son iguales (valor de retorno cero), si la primera es mayor que la segunda (valor de retorno 1) o si la segunda es la mayor (valor de retorno -1).
- Vacía: determina si una cadena está vacía o es nula.
- Mostrar Cadena: dada una cadena de caracteres, muestra su contenido en pantalla.
- Alfabético: permite determinar si una cadena contiene exclusivamente símbolos alfabéticos.
- Mayúsculas: permite convertir el contenido de una cadena de caracteres a mayúsculas.
- Minúsculas: permite convertir el contenido de una cadena de caracteres a minúsculas.

Realice la implementación utilizando LISTAS ENLAZADAS. Considere que la función *getchar()* permite leer caracteres individuales y que el símbolo '\n' identifica la entrada ENTER o INTRO del teclado. Alternativamente, puede usar la función *getche*() para leer caracteres individuales y el símbolo '\r' para identificar la entrada ENTER o INTRO.

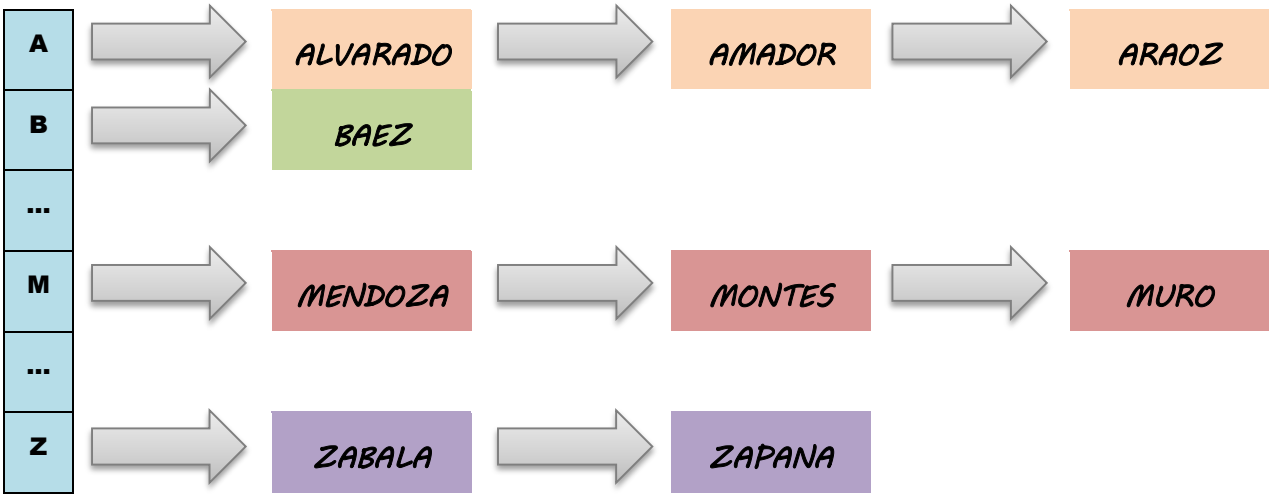
9) En Informática, al ejecutar un programa, las variables de éste se almacenan en una tabla especial llamada *tabla de símbolos*. En esta tabla se registra, por cada variable, nombre simbólico o identificador, tipo de dato, tamaño (en bytes), valor y ámbito (global, local). Considerando esto, defina el *TDA tabla_simbolos* e implemente las siguientes operaciones:

- Validar nombre de variable: dada una variable, verifica que el nombre de la variable sólo contenga letras, dígitos o el guion bajo, y que el primer carácter del nombre sea una letra (mayúscula o minúscula).
- Cantidad de variables de la tabla de símbolos: calcula la cantidad de variables almacenadas en la tabla.
- Total de bytes de la tabla de símbolos: calcula el tamaño (en bytes) de la tabla de símbolos.
- Ordenar tabla de símbolos: ordena las variables de la tabla de símbolos según su tamaño.
- Mostrar tabla de símbolos: muestra el contenido de la tabla de símbolos.

Realice la implementación utilizando a) ARREGLOS (suponga un máximo de 50 elementos) y b) LISTAS ENLAZADAS.

10) Una agenda es una colección de contactos que se organizan de acuerdo a un orden alfabético de modo que la búsqueda se realice rápidamente.

En informática, una agenda puede definirse como una colección de elementos (clave, valor) que se organizan según un criterio de orden preestablecido. En una agenda, la clave hace referencia a la primera letra del apellido del contacto, mientras que valor representa contacto (apellido, nombre y N° de teléfono) propiamente dicho. A fin de agilizar el proceso de búsqueda, las estructuras de datos que implementan agendas agrupan los valores por clave. Por ejemplo, los contactos con apellido "ALVARADO", "AMADOR", "ARAOZ" se agrupan bajo la clave "A", mientras que "MENDOZA", "MONTES", "MURO" se agrupan bajo la clave "M". La siguiente figura muestra un ejemplo de organización de una agenda.



Considerando la definición del TDA agenda se especifican las siguientes operaciones:

- Tamaño: retorna el número de contactos de la agenda.
- Vacío: determina si la agenda está vacía.
- Existe: determina si un contacto pertenece o no a la agenda.
- Consulta contacto: muestra los datos de un contacto.
- Contactos por clave: muestra todos los contactos que corresponden a una clave determinada (letra).
- Agregar contacto: agrega un contacto a la agenda de acuerdo a la clave que corresponda. Por ejemplo, "NIEVA" se guardará en la categoría "N". En cada categoría los datos se disponen en orden.
- Eliminar contacto: remueve un contacto, retornando el contacto extraído.
- Listar: muestra todos los contactos de la agenda.
- Contactos por clave: determina la cantidad de contacto por cada clave de la agenda.

Teniendo en cuenta lo descripto, combine registros, arreglos y listas enlazadas para implementar el TDA agenda.