TAREA #3

EJEMPLOS DE TIPOS DE DATOS ABSTRACTOS

CINTHIA GUADALUPE OLIVAS CALDERON NO. 17212165

- Ejemplo:

Crear el TDA RACIONAL, que corresponde al concepto matemático de un número racional. Un número racional es aquel que puede expresarse como el cociente de dos números enteros. Se definen que las operaciones disponibles para este TDA serán: la creación de un número racional a partir de dos enteros, la adición, la multiplicación, una prueba de igualdad y la impresión de un número racional en pantalla. A continuación el TDA especificado de manera semi formal:

```
 abstract typedef RACIONAL;
 condition RACIONAL[1]!=0;

/* definición del operador */
 abstract RACIONAL crearRacional(a,b)
 int a,b;
precondition b!=0;
postcondition crearRacional[0]==a;
              crearRacional[1]==b;

/* a+ b */
abstract RACIONAL sumar(a,b)
RACIONAL a,b;
postcondition sumar[1]==a[1]*b[1];
          sumar[0]==a[0]*b[1]+b[0]*a[1];

/* a*b */
abstract RACIONAL multiplicar(a,b)
RACIONAL a,b;
 postcondition multiplicar[0]==a[0]*b[0];
multiplicar[1]==a[1]*b[1];

 /* a==b */
 abstract sonIguales(a,b)
```

RACIONAL a,b;
postcondition sonIguales ==(a[0]*b[1]==b[0]*a[1]);

/* imprimir(a) */
abstract imprimirRacional(a)
RACIONAL a,b;

- En el caso del TDA Racional, por ejemplo, podría querer resolverse el siguiente problema:

Escriba un programa que reciba 10 números racionales por teclado y que muestre en pantalla el resultado de su suma.

Un ejemplo del diseño de una solución a este problema, utilizando la definición dada y pseudocodigo sería:

Programa: Suma de 10 números racionales

Entorno: Enteros n,d,i

Racional A, rSuma

Algoritmo:
i = 0

rSuma = crearRacional(0,1)
mientras (i < 10)

escribir “Introduzca el numerador y el denominador del Racional #” + i

leer n, d
A = crearRacional(n,d)

rSuma = sumar(rSuma,A)

i = i+1

finmientras

escribir “El resultado de la suma es ”
imprimirRacional(rSuma)

Finprograma

❖ Implementación del TDA pila

En este caso el costo de las operaciones es de O(1).

Implementación utilizando arreglos

Para implementar una pila utilizando un arreglo, basta con definir el arreglo del tipo de dato que se almacenará en la pila. Una variable de instancia indicará la posición del tope de la pila, lo cual permitirá realizar las operaciones de inserción y borrado, y también permitirá saber si la pila está vacía, definiendo que dicha variable vale -1 cuando no hay elementos en el arreglo.

```
class PilaArreglo
{
  private Object[] arreglo;
  private int tope;
  private int MAX_ELEM=100; // maximo numero de elementos en la pila

  public PilaArreglo()
  {
    arreglo=new Object[MAX_ELEM];
    tope=-1; // inicialmente la pila esta vacía
  }

  public void apilar(Object x)
  {
    if (tope+1<MAX_ELEM) // si esta llena se produce OVERFLOW
    {
```

```
    tope++;
    arreglo[tope]=x;
  }
}

public Object desapilar()
{
  if (!estaVacia()) // si esta vacia se produce UNDERFLOW
  {
    Object x=arreglo[tope];
    tope--;
    return x;
  }
}

public Object tope()
{
  if (!estaVacia()) // si esta vacia es un error
  {
    Object x=arreglo[tope];
    return x;
  }
}
```

```
  public boolean estaVacia()
  {
    if (tope==-1)
    {
      return true;
    }
    else
    {
      return false;
    }
  }
}
```

El inconveniente de esta implementación es que es necesario fijar de antemano el número máximo de elementos que puede contener la pila, MAX_ELEM, y por lo tanto al apilar un elemento es necesario controlar que no se inserte un elemento si la pila está llena. Sin embargo, en Java es posible solucionar este problema creando un nuevo arreglo más grande que el anterior, el doble por ejemplo, y copiando los elementos de un arreglo a otro:

```
public void apilar(Object x)
{
  if (tope+1<MAX_ELEM) // si esta llena se produce OVERFLOW
  {
    tope++;
    arreglo[tope]=x;
  }
```

```
  else
  {
    MAX_ELEM=MAX_ELEM*2;
    Object[] nuevo_arreglo=new Object[MAX_ELEM];
    for (int i=0; i<arreglo.length; i++)
    {
      nuevo_arreglo[i]=arreglo[i];
    }
    tope++;
    nuevo_arreglo[tope]=x;
    arreglo=nuevo_arreglo;
  }
}
```

REFERENCIAS BIBLIOGRÁFICAS

- José Fager W. Libardo Pantoja Yépez Marisol Villacrés Luz Andrea Páez Martínez Daniel Ochoa Ernesto Cuadros-Vargas. (2014). Tipos de Datos Abstractos.En Estructuras de datos(222). Latinoamérica: LATIn.
- Joyanes Aguilar, L., & Zahonero Martínez, I. (1998a). Estructura de datos. Algoritmos, abstracción y objetos. (Ed. rev.). Madrid, España: McGraw-Hill.