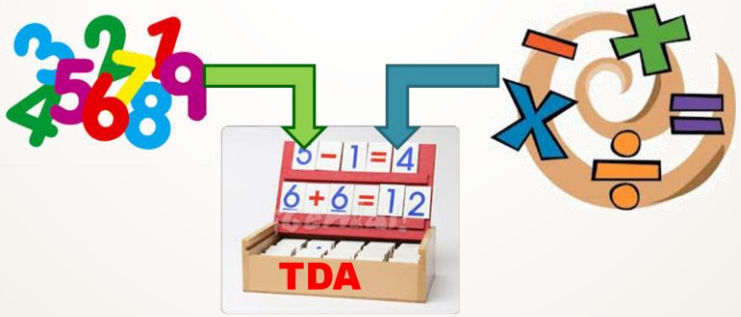
Analista Programador Universitario

Estructura de Datos

Unidad II: TDA Simple



Facultad de Ingeniería
Universidad Nacional de Jujuy

Tipos de Datos (1)

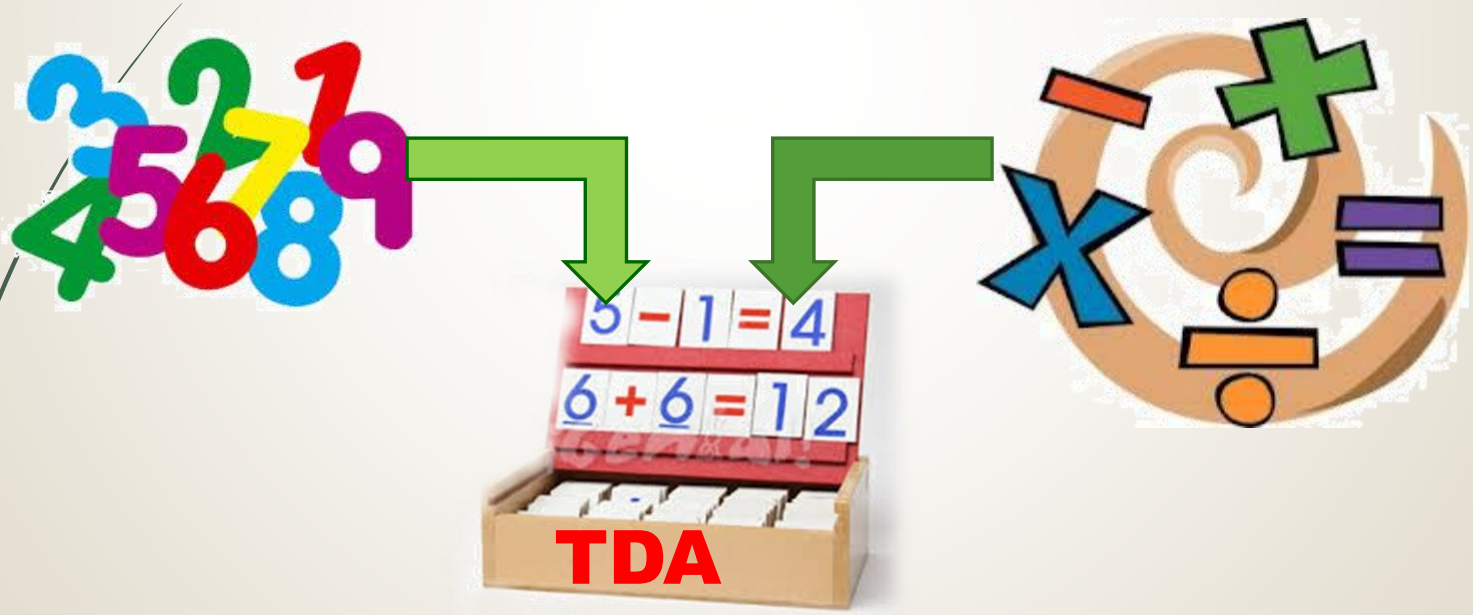
Tipos de Datos (2)

- Un *tipo de dato* hace referencia a un conjunto de valores.
- Un *tipo de dato abstracto* (TDA) comprende tanto el conjunto de valores como las operaciones que pueden aplicarse a ese conjunto.
- Una *estructura de datos* se refiere a la implementación física de un tipo de dato abstracto.

Tipo de Dato Abstracto (1)

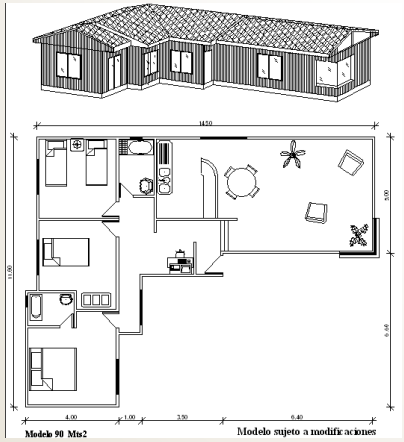
- Los TDA se caracterizan por el conjunto de valores que representan y las operaciones asociadas a ese conjunto.

TDA = datos + operaciones (procedimiento y funciones)



Tipo de Dato Abstracto (2)

- Al crear un TDA se identifican 2 etapas:
  - Especificación del TDA (definición de valores y operaciones)
  - Implementación del TDA (selección de las estructuras de datos y algoritmos)



ESPECIFICACIÓN

IMPLEMENTACIÓN

Especificación de TDA

- La especificación de un TDA consiste en describir el conjunto de valores que contemplará el tipo y las operaciones que podrán realizarse sobre los valores.
- La especificación puede realizarse de 2 maneras:
  - Especificación informal del TDA (descripción en lenguaje natural del TDA)
  - Especificación formal del TDA (descripción en lenguaje formal (matemático) del TDA)

Implementación de TDA

- La implementación de un TDA consiste en seleccionar las estructuras de datos que permitirán almacenar los valores del TDA y los algoritmos que realizarán las operaciones definidas.
- Es decir, la implementación comprende la selección de:
  - Estructuras de datos (registros, arreglos, listas)
  - Algoritmos (procedimientos, funciones)

Especificación: TDA Racional (1)

- Especificación formal del TDA racional

$$\mathbb{Q} \subset \mathrm{Frac}(\mathbb{Z}) = \left\{ \frac{p}{q} \mid p \in \mathbb{Z}, q \in \mathbb{Z}; q \neq 0 \right\}$$

  - Operaciones:
    - ✓ Crear racional $\frac{a}{b}$
    - ✓ Suma de racionales $\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}$
    - ✓ Producto de racionales $\frac{a}{b} \times \frac{c}{d} = \frac{ac}{bd}$

Especificación: TDA Racional (2)

- Especificación informal del TDA racional
  - TDA racional: dato numérico compuesto por 2 valores enteros, numerador y denominador (el denominador nunca puede ser cero)
  - Operaciones:
    - ✓ Crear racional: dados 2 enteros genera un N° racional.
    - ✓ Sumar racionales: dados 2 racionales, genera otro racional cuyo numerador es la suma de los productos cruzados de numerador y denominador, y cuyo denominador es el producto de los denominadores.
    - ✓ Multiplicar racionales: dados 2 racionales, genera otro racional cuyo numerador es producto de los numeradores y cuyo denominador es producto de los denominadores .

Implementación: TDA Racional (0)

- ¿Cuáles son las alternativas para la implementación?
  - Registros
  - Arreglos

Implementación: TDA Racional (1)

- Implementación del TDA Racional mediante registros

```
tracional=REGISTRO
            numerador: ENTERO
            denominador: ENTERO
        FIN_REGISTRO
```

Implementación: TDA Racional (2)

- Implementación de la operación suma de racionales

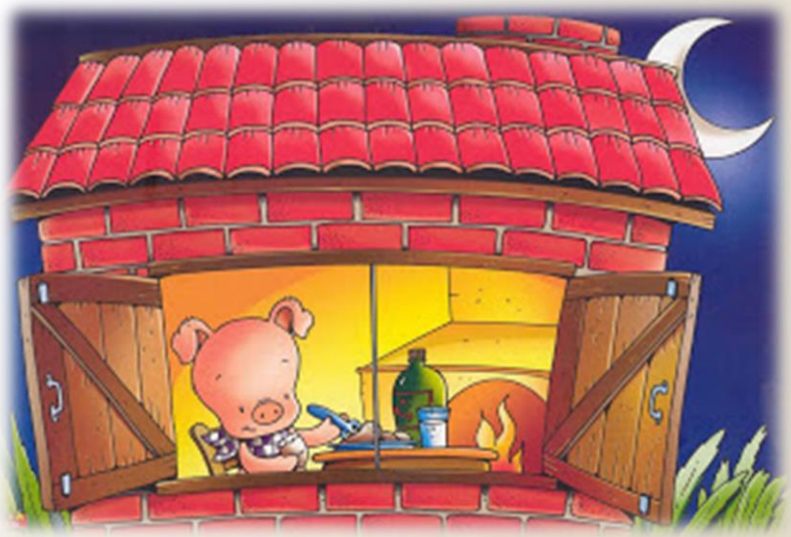
```
PROCEDIMIENTO Sumar(E r1:tracional,E r2:tracional,
                    E/S r3:tracional)
INICIO
r3.numerador ← r1.numerador * r2.denominador +
               r2.numerador * r1.denominador
r3.denominador ← r1.denominador * r2.denominador
FIN
```

$$\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}$$

Implementación: TDA Racional (3)

- Implementación del TDA Racional mediante arreglos.

```
tracional=ARREGLO[1..2] de ENTERO
```



Implementación: TDA Racional (4)

- Implementación de la operación suma de racionales

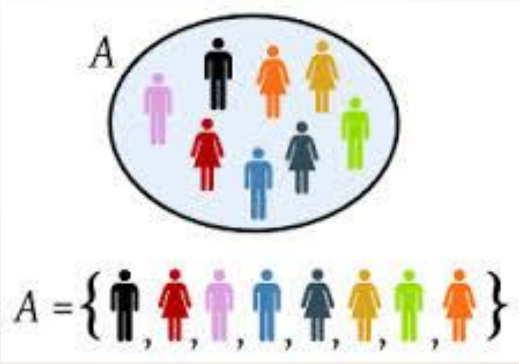
```
PROCEDIMIENTO Sumar(E r1:tracional,E r2:tracional,
                    E/S r3:tracional)
INICIO
   r3[1] ← r1[1] * r2[2] + r2[1] * r1[2]
   r3[2] ← r1[2] * r2[2]
FIN
```

$$\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}$$

Especificación: TDA Conjunto (1)

- Especificación del TDA Conjunto
  - Un conjunto es una colección homogénea de elementos, cuyo orden no es relevante, y dónde no se permiten las repeticiones.
  - Un conjunto puede representarse como:

$$\{elemento_1, elemento_2, \ldots, elemento_n\}$$



Especificación: TDA Conjunto (2)

- Especificación de operaciones del TDA Conjunto
  - Conjunto vacío: genera un conjunto vacío (sin elementos)
  - Unión: dados 2 conjuntos, genera un nuevo conjunto que contiene los elementos (sin repeticiones) de los conjuntos originales.
  - Intersección: dados 2 conjuntos, genera un nuevo conjunto que contiene los elementos comunes a ambos conjuntos.
  - Diferencia: dados 2 conjuntos, genera un nuevo conjunto que contiene los elementos del primer conjunto que no pertenezcan al segundo conjunto.
  - Pertenencia: dado un conjunto y un valor, determina si el valor se encuentra o no entre los elementos del conjunto.

Implementación: TDA Conjunto (0)

- ¿Cuáles son las alternativas para la implementación?
  - Arreglos
  - Listas

Implementación: TDA Conjunto (1)

- Implementación del TDA Conjunto mediante arreglos

```
const int MAX=20;
typedef int telementos[MAX];
typedef struct tconjunto { telementos valores;
                           int ocupado; };
```

Implementación: TDA Conjunto (2)

- Implementación de la operación pertenencia de conjuntos

```
bool pertenece(tconjunto a, int num)
{ int i;
  bool existe=false;
  for(i=0;i<=a.ocupado && a.valores[i]!=num;i++);
  if (i <= a.ocupado)
    existe=true;
  return existe;
}
```

Implementación: TDA Conjunto (3)

- Implementación de la operación intersección de conjuntos

```
void intersec(tconjunto a,tconjunto b,tconjunto &c)
{ int i;
  crear_vacio(c);
  for(i=0;i<=a.ocupado;i++)
     if (pertenece(b,a.valores[i])==true)
        agregar_elemento(c,a.valores[i]);
}
```

Implementación: TDA Conjunto (4)

- Implementación del TDA Conjunto mediante listas simples (con punteros de *inicio* y *final*)

```
typedef struct telemento *pelemento
typedef struct telemento
                { int dato;
                  pelemento sig;
                };
typedef struct tconjunto
                { pelemento inicio;
                  pelemento final;
                };
```

Implementación: TDA Conjunto (5)

- Implementación de la operación pertenencia de conjuntos

```
bool pertenece(tconjunto a, int num)
{ pelemento i;
  bool existe=false;
  for(i=a.inicio;i!=NULL && i->dato!=num;i=i->sig);
  if (i!= NULL)
    existe=true;
  return existe;
}
```

Implementación: TDA Conjunto (6)

- Implementación de la operación intersección de conjuntos

```
void intersec(tconjunto a, tconjunto b, tconjunto &c)
{ pelemento i, nuevo;
   crear_vacio(c);
   for(i=a.inicio;i!=NULL;i=i->sig)
      if (pertenece(b,i->dato)==true)
          { crear_nodo(nuevo,i->dato);
            agregar_elemento(c,nuevo);}
}
```

```
void crear_nodo (pelemento &nuevo, int valor)
{
   nuevo=new telemento;
   if (nuevo==NULL)
     cout << "No hay memoria" << endl;
   else
    { nuevo->dato=valor;
      nuevo->sig=NULL;
    }
}
```

Ejemplo TDA

- Considerando que la magnitud tiempo puede definirse por las componentes hora, minuto y segundo, defina el TDA tiempo y la operación validar tiempo.
  - ¿Cuáles son los valores de este tipo?
  - ¿Cuáles son las operaciones que pueden aplicarse?
  - ¿Cómo puede implementar el TDA?
    - ✓ arreglos?
    - ✓ registros?
    - ✓ listas?

Ventajas de TDA

- El uso de TDA presenta las siguientes ventajas:
  - Permite representar entidades del mundo real.
  - Facilita la comprobación de tipos.
  - En lenguajes tipados, mejora el tiempo de compilación.
  - Independiza la *especificación* del TDA de su *implementación* (estructuras de datos), lo que facilita futuras modificaciones
  - Favorece la reusabilidad.

Bibliografía

- Joyanes Aguilar *et al*. Estructuras de Datos en C++. Mc Graw Hill. 2007.
- De Giusti, Armando *et al.* Algoritmos, datos y programas, conceptos básicos. Editorial Exacta. 1998.
- Joyanes Aguilar, Luis. Fundamentos de Programación. Mc Graw Hill. 1996.
- Hernández, Roberto *et al*. Estructuras de Datos y Algoritmos. Prentice Hall. 2001.