

# Tema IX

INTRODUCCIÓN A LOS TIPOS DE DATOS ABSTRACTOS (TAD). ABSTRACCIÓN DE DATOS. CONCEPTO SOBRE TIPOS DE DATOS. MÓDULOS, INTERFAZ E IMPLEMENTACIÓN. ENCAPSULAMIENTO DE DATOS. DIFERENCIA ENTRE TIPO DE DATO Y TIPO ABSTRACTO DE DATOS. VENTAJAS DEL USO DE TAD. FORMAS DE ABSTRACCIÓN. REQUERIMIENTO Y DISEÑO DE UN TAD.

Prof. Oscar Adolfo Vallejos



# OBJETIVO

- Extender el concepto de tipo de dato definido por el usuario como una caracterización de elementos del mundo real, tendiendo al encapsulamiento de la representación y al comportamiento dentro de un tipo abstracto de datos (TAD).
- Enfatizar la importancia de la abstracción de los datos y de las operaciones para lograr la reutilización



# ABSTRACCIONES DE DATOS

- Caracterizar el mundo real para poder resolver problemas concretos mediante el empleo de herramientas informáticas.
- Reconocer objetos del mundo real y abstraer sus aspectos fundamentales y su comportamiento de modo de poder representarlos en un ordenador.
- La utilidad de la abstracción y la modelización de objetos es la posibilidad de reusar soluciones.



# CONCEPTOS SOBRE TIPOS DE DATOS

- En la memoria de datos de un ordenador no existen las “estructuras de datos”. Simplemente hay bits en 0 o 1.
- Las nociones de tipos, estructuras de datos, variables y constantes son abstracciones para acercar la especificación de los datos de problemas concretos al mundo real.
- Los lenguajes y SO se encargan de manejar naturalmente las conversiones entre el ámbito propio del especialista en informática y la realidad del hardware de los ordenadores
- El concepto de tipo de datos es una necesidad de los lenguajes de programación que conduce a identificar valores y operaciones posibles para variables y expresiones.



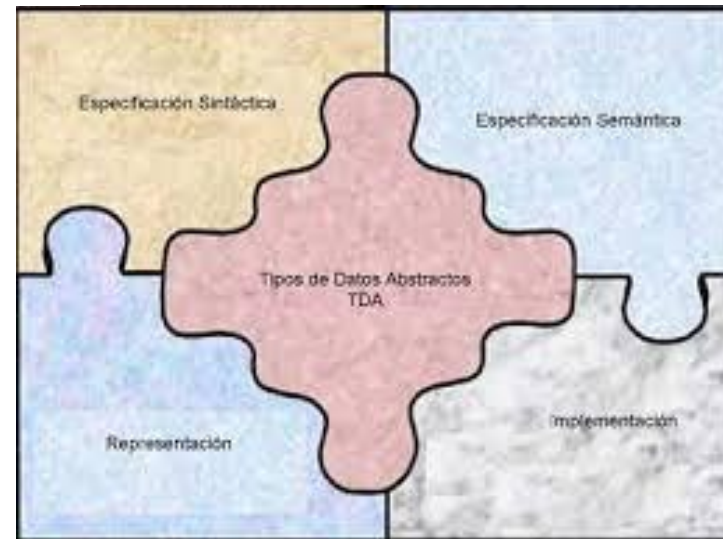
# SISTEMA DE TIPOS

- El sistema de tipos de un lenguaje es el conjunto de reglas para asociar tipos a expresiones en el lenguaje.
- La mayoría de los lenguajes impone la necesidad de declarar todas las variables asociándolas un tipo. Luego se verifica que las operaciones se hagan entre datos de mismo tipo.
- El sistema de tipos establece reglas, según operaciones que se realice con los datos.
- Algunos lenguajes resuelven la mezcla de tipos en expresiones mediante coersión (conversión forzada).
- Como principio general es conveniente que la mezcla de tipos no este permitido en las expresiones.
- La verificación de que los tipos usados sean los correctos según la operación en curso es una herramienta muy poderosa para la prevención y corrección de errores.

# MÓDULOS, INTERFAZ E IMPLEMENTACIÓN

- *Ya hemos visto las ventajas de descomponer (modularizar) un sistema de software en procedimientos y funciones. Esencialmente se logra abstraer las operaciones, de modo de descomponer funcionalmente un problema complejo.*
- *Idealmente el modulo es una caja negra con una función interna y una interfaz de vinculación con otros módulos.*
- *La interfaz es una especificación de las funcionalidades del módulo y puede contener las declaraciones de tipos y variables que deben ser conocidas externamente.*
- *La implementación del módulo abarca el código de los procedimientos que concretan las funcionalidades internas.*
- *La interfaz se conoce como la parte pública del módulo, mientras que la implementación es la parte privada.*





## ENCAPSULAMIENTO DE DATOS

- A partir de la creación de especificaciones abstractas verdaderas o abstracción de datos es lograr encapsulamiento (empaquetamiento) de los datos: se define un nuevo tipo y se integran en un módulo todas las operaciones que se pueden hacer con él.
- si el lenguaje permite separar la parte visible (interfaz) de la implementación, se tendrá ocultamiento de datos (data hiding).
- Y si se logra que la solución del cuerpo del módulo pueda modificar la representación del objeto – dato, sin cambiar la parte visible del módulo, se tendrá independencia de la representación.





## DIFERENCIA ENTRE TIPO DE DATO Y TIPO ABSTRACTO DE DATO

- Un objeto del mundo real es algo más complejo que las representaciones de números o caracteres que se han visto en los tipos de datos simples.
- No alcanza con pensar en la estructura de registro (tipo) que sirve para representarlos para caracterizar estos objetos: no es necesario agregar al tipo las operaciones que caracterizan su comportamiento.



El concepto de tipo abstracto de dato es un tipo de datos definido por el programador que incluye:

- ❑ Especificación de la representación de los elementos del tipo
- ❑ Especificación de las operaciones permitidas con el tipo
- ❑ Encapsulamiento de todo lo anterior, de manera que el usuario no pueda manipular los datos del objeto, excepto por el uso de las operaciones definidas en el punto anterior.
- ❑ La forma de programación utilizada

Programa = Datos + Algoritmos



## Conclusión

- Dado un problema a resolver, se busca representar, los objetos que participan en el mismo e incorporar los algoritmos necesarios para llegar a la solución.
- La concepción de este algoritmo se realiza pensando en la solución del problema en su conjunto y en tal sentido se aplican técnicas de modularización a fin de reducir la complejidad del problema.



Se refina la ecuación anterior se puede mejorar:

$$\text{Algoritmo} = \text{Algoritmo de datos} + \text{Algoritmo de control}$$

Algoritmo de dato es la parte del algoritmo encargada de manipular las estructuras de datos del problema.

Algoritmos de control es la parte que representa el método de solución del problema, independiente de las estructuras de datos seleccionadas.

Dado que los TAD reúne en su definición la representación y el comportamiento de los objetos del mundo real se puede escribir la ecuación inicial como:

$$\text{Programa} = \text{Datos} + \text{Algoritmos de Datos} + \text{Algoritmos de Control}$$

$$\text{Programa} = \text{TAD} + \text{Algoritmo de control}$$



## VENTAJAS DEL USO DE TAD (RESPECTO DE LA PROGRAMACIÓN CONVENCIONAL)

- Es posible el desarrollo de algoritmos sin utilizar tipos abstractos de datos.
- El TAD lleva en si mismo la representación y el comportamiento de sus objetos y la independencia que este posee del programa o módulo que lo utiliza permite desarrollar y verificar su código de manera aislada. Es decir es posible implementar y probar el nuevo TAD de una forma totalmente independiente del programa que lo va a utilizar.
- Esta independencia facilita la re – usabilidad del código.
- El modulo que referencia al TAD, lo utiliza como una caja negra de la que se obtienen los resultados a través de operaciones predefinidas. Esto permite que las modificaciones internas de los TAD no afectan a quienes lo utilizan (la interfaz no se debe modificar).

Con el uso de TAD, el programador diferencia dos etapas:

- 1.- En el momento de diseñar y desarrollar el TAD no interesa conocer la aplicación que lo utilizara.
- 2.- En el momento de utilizar el TAD no interesa saber como funcionara internamente, solo bastará con conocer las operaciones que permiten manejarlo



## En resumen ...

1. Permite una mejor conceptualización y modelización del mundo real. Mejora la representación y la comprensibilidad. Clarifica los objetos basados en estructuras y comportamientos comunes.
2. Mejora la robustez del sistema. Si hay características subyacentes en los lenguajes permiten la especificación del tipo de cada variable, los tipos abstractos de datos permiten la comprobación de tipos para evitar errores de tipo en tiempo de ejecución.
3. Mejora el rendimiento (prestaciones). Para sistemas tipeados, el conocimiento de los objetos permite la optimización de tiempo de compilación.
4. Separa la implementación de la especificación. Permite la modificación y mejora de la implementación sin afectar al interfaz público del tipo abstracto de dato.
5. Permite la extensibilidad del sistema. Los componentes de software reutilizables son más fáciles de crear y mantener.
6. Recoge mejor la semántica del tipo. Los tipos abstractos de datos agrupan o localizan las operaciones y la representación de atributos.

## FORMAS DE ABSTRACCIÓN: TIPOS DE DATOS ABSTRACTOS Y TIPOS DE DATOS LÓGICOS

- Cuando trabajamos con pilas ...
- 1,. Se definieron las características del objeto (ej.: cola) sin discutir su implementación.
- 2.- Se definieron las operaciones permitidas sobre el objeto (Ej.: poner, sacar) indicando como invocarlas, es decir, indicando la interfaz.
- (abstracción en dos direcciones: datos y operaciones).
- Este tipo de datos se denomina TDL (Tipo de dato lógico) pues existe una conexión lógica, conocida por el usuario entre los tipos, por ejemplo, la declaración del tipo de pila, y los procedimientos o funciones asociadas con dicho tipo.
- Con los TDL no se llega a un verdadero tipo abstracto de datos porque no existe una vinculación estructural entre dicho tipo y las operaciones asociadas al mismo; es más: los datos declarados dentro del tipo podrían ser accedidos por otras operaciones y, asimismo, las operaciones asociadas con el tipo podrían ser utilizadas para otros datos.



# REQUERIMIENTOS Y DISEÑO DE UN TAD


- Disponer de un TAD posibilita tener código reusable.

Esto requiere:


- Poder encapsular dentro de un módulo del lenguaje
- Poder declarar tipos protegidos de modo que la representación interna este Oculta de la parte visible
- Poder heredar el TAD, es decir, crear instancias a partir de ese molde.



El diseño de un TAD lleva consigo a selección de:

- ❑ Una representación interna, lo que implica conocer las estructuras de datos adecuadas para representar la información.
  - ❑ Las operaciones a proveer para el nuevo tipo y el grado de parametrización de las mismas.
- 

Estas operaciones se clasifican en:

- ❑ Operaciones para crear o inicializar objetos (Ej. Cree un pila vacía)
  - ❑ Operaciones para modificar los objetos del TAD que permiten quitar o agregar elementos del TAD (Ej. Push).
  - ❑ Operaciones que permiten analizar los elementos del TAD (Ej.: verificar que si la pila esta vacía).
- 

# CREAR TU PROPIA BIBLIOTECA EN C

- Existen bibliotecas estándares en C que ya vienen incluida en la mayoría de los compiladores, como son *stdio.h*, *math.h*, *time.h*.
- La biblioteca, o también mal conocida como librería (del ingles *library*) nos permite el uso de estructuras y funciones, previamente definidas en un programa, sin la necesidad de escribir su código en nuestro programa. (TAD).
- Posteriormente, desde nuestro programa deberemos invocar dicha librería. En resumen, bastara con situar en la cabecera del programa el nombre de la biblioteca para poder utilizar todas las funciones y estructuras contenidas en la misma.



# PASOS PARA CREAR UN BIBLIOTECA

- **Genera las funciones que te interesan y escribelas todas juntas (codigo y cabeceras) en un mismo archivo de texto** (Puedes usar el editor de texto del compilador, el bloc de notas, igual da...).
- **El fichero creado anteriormente, guardalo con extension *.h***, se deberá guardar en la carpeta *include* del compilador o se puede guardar el fichero en la misma carpeta del codigo que queramos compilar.
- **Llamar a la biblioteca en el programa.** Deberemos colocar en la cabecera del programa, junto a los llamamiento de otras bibliotecas:



## EJEMPLO

```
int multiplica(int A, int B);  
int multiplica(int A, int B)  
{  
    return(A*B);  
}
```

```
#include <stdio.h>  
#include "milibreria.h"  
  
int main(void)  
{  
    int X, Y;  
    scanf("%d %d", &X, &Y);  
    printf("X*Y=%d \n", multiplica(X, Y));  
}
```

`include <milibreria.h>` Cuando el fichero *milibreria.h* se encuentre en la carpeta include de nuestro compilador.

`#include "milibreria.h"` Cuando el fichero *milibreria.h* esté en el mismo directorio que el archivo que queremos compilar.

# BIBLIOGRAFÍA

- ESTRUCTURA DE DATOS EN C. Luis Joyanes y Otros. 2007. Editorial MCGRAW-HILL. ISBN: 978-84-5645-9.
- INTRODUCCION AL DISEÑO Y ANALISIS DE ALGORITMOS. UN ENFOQUE ESTRATEGICO. R.C.T. Lee; S.S. Tseng; R.C. Chang; Y.T. Tsai 2007. MCGRAW-HILL. ISBN: 978-970-10-6124-4.

