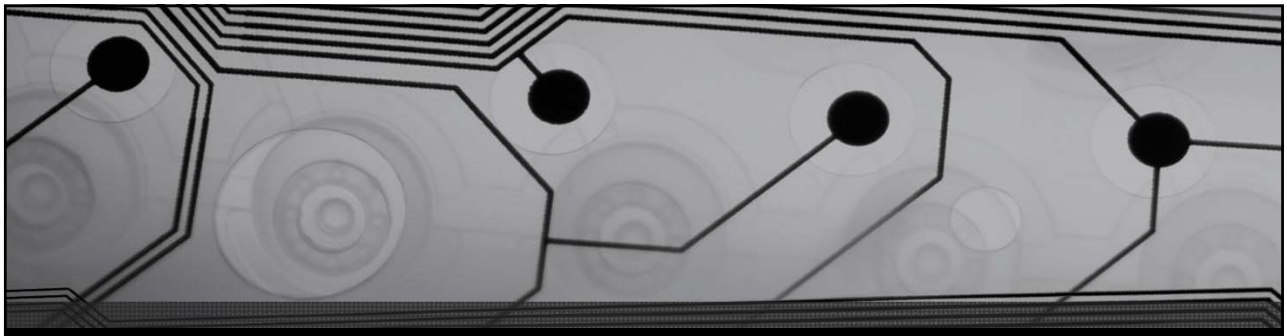
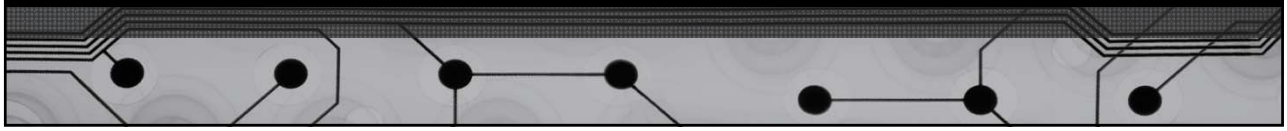


Abstracción en Lenguajes de Programación

Dr. Víctor de la Cueva
vcueva@itesm.mx



Abstracción

- Juega un papel sumamente importante para hacer que los programas puedan se leídos y entendidos fácilmente por la gente.
- La abstracción de los lenguajes cae en dos categorías:
  - Abstracción de datos: simplifica el comportamiento y los atributos de los datos (e.g números, cadenas de caracteres y árboles de búsqueda)
  - Abstracción de control: simplifica las propiedades de la transferencia de control, esto es, la modificación del camino de ejecución de un programa basado en una situación particular (e.g. ciclos, condicionales y llamadas a procedimientos)

Niveles de abstracción

- La abstracción también puede ser clasificada en términos de niveles, los cuales pueden ser vistos como una medida de la cantidad de información contenida (y escondida) en la abstracción:
  - Abstracción básica: abarca la mayor parte de la información de la máquina.
  - Abstracción estructurada: abarca información intermedia acerca de la estructura de un programa.
  - Abstracción unitaria: abarca la información de gran-escala de un programa.

Datos: Abstracción Básica

- Esconde la representación interna de los valores de los datos en una computadora, ej:
  - Los valores de datos enteros son normalmente almacenados en una computadora usando complementos a 2.
  - Los valores de punto flotante esconden la representación de máquina del IEEE doble precisión.
- Esos valores son normalmente llamados "primitivos" o "atómicos" porque el programador no puede acceder a parte de los bits que lo forman.

Variables

- Otra abstracción de datos es el uso de nombres simbólicos para esconder las localidades en la memoria de la computadora que contienen valores de los datos.
- Tales nombres son llamados variables.
- A la clase de valor de dato también se le da un nombre y es llamado tipo de dato.
- A los tipos de datos básicos se les da un nombre que es una variación de sus correspondientes valores matemáticos, e.g. `int`, `double` y `float`.
- A las variables se les dan nombres y tipos de datos usando una declaración (e.g. en Pascal se hacía `var x: integer;` y en C, `int x;`)
- También se proporcionaban operaciones básicas para tipos de datos básicos.

Datos: Abstracción Estructurada

- La estructura de datos es el principal método para agrupar valores de datos relacionados en una sola unidad.
- Tres ejemplos clásicos son:
  - Record: agrupa la información relacionada, por ejemplo, con una sola persona, que puede contener: nombre, IMSS, RFC, CURP, dirección, etc., que son datos de diferentes tipos.
  - Array: agrupa datos del mismo tipo en una secuencia de ítems indexados, todos bajo el mismo nombre.
  - Text file: es una abstracción que representa una secuencia de caracteres para ser trasferidos hacia y desde un dispositivo de almacenamiento externo. Es independiente del tipo de medio de almacenamiento (e.g. disco magnético, disco óptimo, dispositivo de estado sólido, inclusive el teclado y una ventana (consola).

Estructura de Datos

- Una Estructura de Datos esconde un grupo de datos componentes permitiendo al programador verlo como un todo.
- A diferencia de los valores de datos primitivos, las estructuras de datos proporcionan al programador los medios de la construcción a partir de sus partes componentes, que pueden ser también estructuras de datos.
- También proporciona los medios para acceder a ellos y modificarlos.

Datos: Abstracción Unitaria

- En programas muy grandes es útil y necesario agrupar juntos los datos relacionados y las operaciones que se hacen sobre esos datos, ya sea en archivos separados o en estructuras del lenguaje separadas, dentro de un archivo.
- Típicamente, esas abstracciones incluyen convenciones de acceso y restricciones que soportan los datos escondidos.
- Estos mecanismos varían mucho de un lenguaje a otro pero, en general, permiten al programador definir nuevos tipos de datos (datos y operaciones), que esconden información en la misma forma que lo hacen los tipos de datos básicos en un lenguaje.
- Esta unidad de abstracción es normalmente asociada con el concepto de un tipo de dato abstracto.

Tipos de datos abstractos

- Definidas como un conjunto de valores de datos y las operaciones sobre ellos.
- Su principal características es la separación entre la interface (el conjunto de operaciones disponible para el usuario) y su implementación (la representación interna de valores de datos y operaciones).
- Ejemplos de abstracciones unitarias de gran-escala son:
  - Los módulos de ML, Heskell y Python
  - Los paquetes de LISP, Ada y Java
- Ejemplos de abstracciones unitarias de baja-escala: las clases de la OOP.
- Una propiedad adicional, cada vez con más importancia, es su reusabilidad.

Librerías

- Las abstracciones de datos unitarias se convierten en las bases para los mecanismos de las librerías del lenguaje.
- Estos mecanismos así como las librerías estándares, pueden o no ser parte del lenguaje.
- Se han desarrollado muchos estándares para las interfaces ya sea en forma independiente del lenguaje o atado a la definición del mismo.
- Cuando los programadores van a usar un nuevo recurso de software, típicamente estudian su **Application Programming Interface (API).**
- Una API da al usuario solamente la información necesaria sobre clases, métodos, funciones y características de desempeño, suficiente para usar esos recursos en forma efectiva.

Control: Abstracciones básicas

- Son los estatutos en el lenguaje que combinan un pequeño grupo de instrucciones de máquina en una instrucción abstracta que es más entendible para el programador que las instrucciones de máquina (e.g. la notación algebraica para las expresiones aritméticas y el estatuto de asignación).
- El término **syntactic sugar** es usado para referirse a cualquier mecanismo que permita al programador reemplazar una notación compleja con una más simple y más corta (e.g. la operación de asignación extendida +=).

Control: Abstracciones estructuradas

- Dividen un programa en grupos de instrucciones que son anidadas dentro de verificación de condiciones (pruebas) que gobiernan su ejecución.
- Ayudan al programador a expresar la lógica de las estructuras primarias de control de secuenciación, selección e iteración (ciclos).
- Otra estructura de este tipo es el iterador.
  - Típicamente se encuentra en los OOL.
  - Es un objeto que se asocia con una colección: arreglo, lista, conjunto o árbol.
  - El programador abre un iterador sobre una colección y visita todos sus elementos corriendo métodos del iterador en el contexto de un ciclo.

```
        LEA  R1, LIST      ; Load the base address of the array (the first cell)
        AND  R2, R2, #0    ; Set the sum to 0
        AND  R3, R3, #0    ; Set the counter to 10 (to count down)
        ADD  R3, R3, #10
WHILE   LDR  R4, R1, #0    ; Top of the loop: load the datum from the current
                           ; array cell
        BRZP INC           ; If it's >= 0, skip next two steps
        NOT  R4, R4        ; It was < 0, so negate it using twos complement
                           ; operations
        ADD  R4, R4, #1
INC     ADD  R2, R2, R4    ; Increment the sum
        ADD  R1, R1, #1    ; Increment the address to move to the next array
                           ; cell
        ADD  R3, R3, #-1   ; Decrement the counter
        BRP  WHILE         ; Goto the top of the loop if the counter > 0
        ST   R2, SUM       ; Store the sum in memory
```

```
int sum = 0;
for (int i = 0; i < 10; i++){
    int data = list[i];
    if (data < 0)
        data = -data;
    sum += data;
}
```

Procedimientos

- Un mecanismo poderoso de control estructurado es el procedimiento (también llamado subrutina o subprograma).
  - Permite al programador considerar una secuencia de acciones como una sola acción que puede ser llamado o invocada desde muchos puntos de un programa.
  - Un procedimiento se define para relacionarse con un nombre y con las acciones que debe realizar, esto se llama declaración del procedimiento (similar a la declaración de una variable o tipo).
  - El procedimiento debe ser llamado en el punto donde se deben realizar las acciones, esto se llama invocación o activación del procedimiento.
  - Puede tener parámetros que en ejecución se sustituyen por los argumentos (o parámetros actuales).

Funciones

- Es un mecanismo de abstracción íntimamente relacionado con un procedimiento.
- Se puede ver como un procedimiento que regresa un valor al momento de ejecutarse.
- La importancia de la función es mucho mayor que la de los procedimientos debido a que se relacionan más con el concepto matemático de función, esto implica que pueden ser entendidas independientemente del concepto de computadora.
- Además, las funciones pueden ser combinadas en abstracciones de nivel superior conocidas como funciones de orden superior.
  - Son capaces de aceptar funciones como argumentos y regresar funciones como valores (e.g las funciones `map` y `reduce` de Scheme).
  - `(map abs (list 33 -10 66 88 -4))` , regresa `(33 10 66 88 4)`
  - `(reduce + (map abs (list 33 -10 66 88 -4))`, regresa 201

Control: Abstracciones Unitarias

- El control puede ser abstraído para incluir una colección de procedimientos que proporcionan servicios relacionados lógicamente, a otras partes de un programa y que forman una unidad, o parte de programa stand-alone.
  - Ej, un programa para administrar datos puede requerir calcular medidas estadísticas tales como media, mediana y desviación estándar.
  - Estos procedimientos que proporcionan estas operaciones pueden ser agrupados en una unidad de programa (**program unit**) que puede ser trasladada y usada por otras partes del programa por medio de una interface cuidadosamente controlada.
- Otra abstracción de control, que es difícil colocar en un nivel de abstracción, es la de los mecanismos de programación paralela:
  - Java tiene mecanismos para declarar hilos (threads), caminos de control ejecutados separadamente dentro del sistema Java, y procesos (otros programas ejecutándose fuera del sistema Java).
  - Ada proporciona el mecanismo task para ejecución paralela, que son esencialmente unidades de abstracción.

Referencias

- R. Sethi. Programming Languages: concepts and constructs. Addison-Wesley, 2nd edition (1996).
- K.C. Louden and K. A. Lambert. Programming Languages: Principles and Practice. Cengage 3rd edition (2011).