Modularización

El **Pensamiento Computacional** es la habilidad para resolver problemas, diseñar sistemas y entender el comportamiento de los sistemas complejos utilizando conceptos y técnicas de la computación. Es una forma de pensar que involucra la descomposición de problemas complejos en partes más pequeñas y manejables, la identificación de patrones y relaciones, la creación de algoritmos y la evaluación de soluciones.

Modularizar: Dividir o descomponer un problema complejo en partes más pequeñas y manejables.

EJEMPLO: Problema grande a resolver: Organizar una fiesta de cumpleaños para 20 personas en casa.

Problema 1: Hacer invitaciones

Problema 2: Comida y bebida

Problema 3: Decoración

Problema 4: Entretenimiento

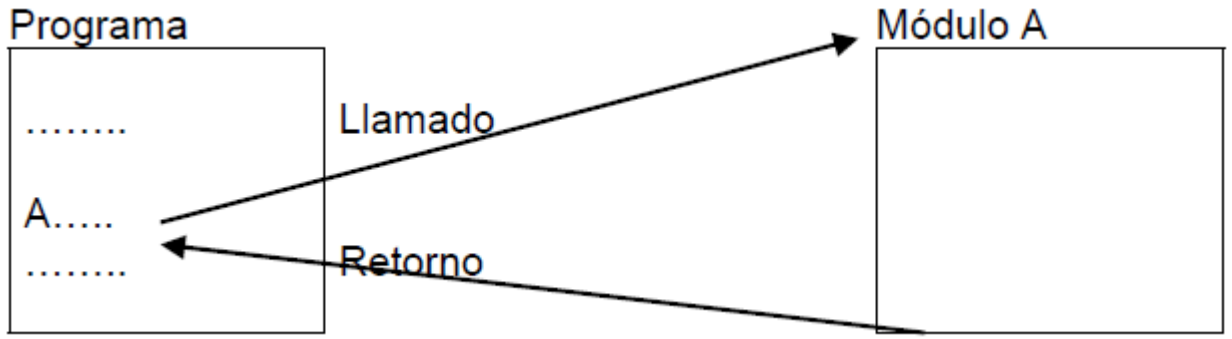
Problema 5: Logística

Modularización

- un problema complejo puede ser resuelto de manera más fácil y eficiente si se divide en problemas más pequeños y concentrándonos en cada etapa en la solución de ese "subproblema".
- Esto implica que el gran problema original será resuelto por medio de varios módulos, cada uno de los cuales se encarga de resolver un subproblema determinado.
- Los módulos se escriben sólo una vez, luego es posible hacer referencia a ellos ("llamarlos") desde diferentes puntos de un pseudocódigo. La ventaja obvia es que nos permite reutilización y evita la duplicación de códigos.
- Los módulos son independientes entre si, en el sentido de que se puede escribir y verificar cada módulo en forma separada sin preocuparse por los demás módulos. Por ello, es menos complicado localizar un error y también se puede modificar el código sin tener que tocar o rehacer varias partes del mismo.
- Notemos que al utilizar módulos se establece un límite para el alcance de las variables, unas tendrán efecto y valor sólo en el módulo y otras en el algoritmo principal, también es posible especificar que una variable tenga efecto en el algoritmo principal y todos los módulos.
- Los módulos pueden recibir valores del algoritmo principal (parámetros), trabajar con ellos y devolver un resultado al algoritmo principal: No existen limitaciones en cuanto a las acciones que pueda ejecutar un módulo.

Modularización

- Un módulo puede, a su vez, invocar o llamar a otros o a sus propios módulos, inclusive puede llamarse a sí mismo (esto se conoce como recursividad).
- Cada módulo debe tener un nombre que lo identifique.
- Puede tener una serie de parámetros asociados.
- El nombre del módulo es utilizado para la invocación del mismo.
- Cuando se invoca a un subprograma se transfiere el control al mismo y una vez finalizada la última instrucción del módulo el control retornará a la siguiente instrucción del programa o subprograma que lo llamó.



Modularización: Variables locales y globales

En programación existen dos tipos de variables, las llamadas locales y las variables globales.

Variables Locales: Son aquellas que se encuentran dentro de un modulo y son distintas de las variables que están en el algoritmo principal.
Si en el algoritmo principal tratamos de utilizar estas variables o imprimirlas, no obtendremos nada, ya que para el algoritmo estas variables son locales y desde su punto de vista NO EXISTEN.
Variables Globales: Son las que se definen o están declaradas en el algoritmo principal y tiene efecto tanto en el algoritmo principal como en cualquiera de sus módulos.
Comparación: Una variable local (de un módulo) no tiene ningún significado en el algoritmo principal y otros módulos. Si un módulo asigna un valor a una de sus variables locales, este valor no es accesible desde otros módulos, es decir, no pueden utilizar este valor. Las variables globales tienen la ventaja de compartir información de diferentes módulos.
En resumen: las variables locales son las que se definen en módulos y solo tienen valor dentro de él. Las variables globales son definidas en el algoritmo principal y tienen valor y se pueden utilizar en cualquier parte de algoritmo o en cualquier subprograma.

Conceptualmente, puede decirse que desarrollar módulos con independencia funcional, variables locales y mínima comunicación externa mediante parámetros es una buena práctica de programación, que favorece la reutilización y el mantenimiento del software.

Modularización: Parámetros

Generalmente, los módulos, necesitan para operar algunos datos que están disponibles en el algoritmo que los invoca. Por esta razón, es necesario “comunicar” los módulos. La forma en que se realiza esta comunicación es a través de parámetros.

Se denomina parámetros a la serie de datos con los que se comunican los módulos. Los parámetros de un módulo deben definirse en el encabezado del mismo. Cada parámetro debe especificar el tipo de dato con el que se corresponde.

Los parámetros que se definen en el llamado del módulo reciben el nombre de **argumentos o parámetros actuales/reales**, en tanto que los parámetros descritos en el encabezado del módulo invocado (procedimiento o función) se denominan **parámetros formales**.

Un **parámetro formal** cumple, para el modulo en el cual está definido, la misma función que cualquier otra variable. Podemos asignarle cualquier valor y utilizarlo en cualquier expresión (siempre de su mismo tipo). Son parámetros formales todos los identificadores (nombres) incluidos entre paréntesis a continuación del nombre del subprograma.

Modularización: Parámetros

Declaracion de un Módulo

funcion <nombreFunción>(<ParámetrosFormales>)
    Instrucción
    Retorno <valor|variable|calculo>
Fin_Funcion

Invocacion a un Módulo

Inicio
    v= nombreFunción(Argumentos)
Fin

Correspondencia entre parámetros actuales y formales

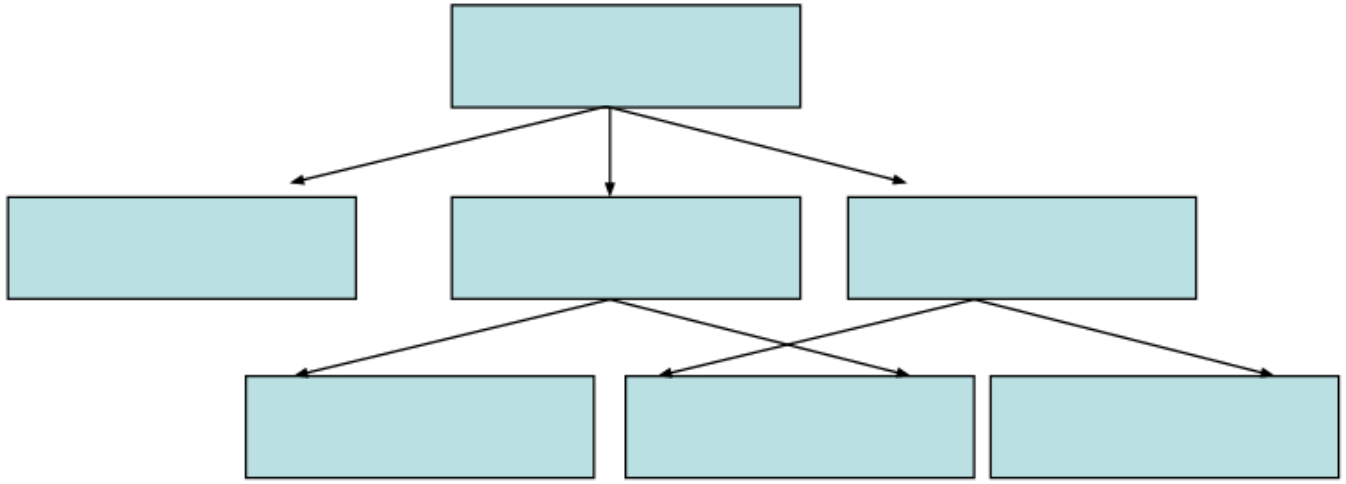
Cuando se invoca un módulo, y éste posee parámetros, la correspondencia entre los parámetros actuales y formales es **por posición**, esto significa que el primer parámetro actual corresponde al primero formal; el segundo parámetro actual corresponde al segundo formal, y así sucesivamente.
Los parámetros y los argumentos deben coincidir en, orden, tipo y catidad.

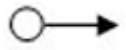
Modularización: Diagrama de Estructura

Un diagrama de estructura (DE) permite modelar un programa como una jerarquía de módulos.

Para construirlo aplicamos las técnicas de DESCOMPOSICIÓN, ABSTRACCIÓN y RECONOCIMIENTO DE PATRONES.

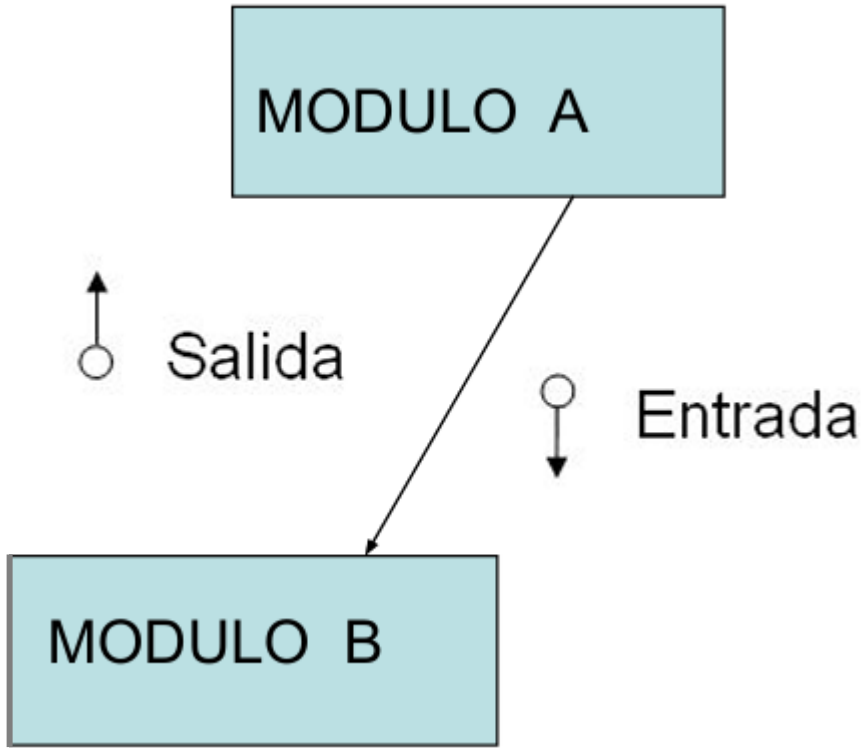
REPRESENTACIÓN:
El **módulo** (*rectángulo*) de un nivel de la jerarquía **invoca** (*flecha*) uno a más módulos del nivel inmediatamente inferior. Los módulos se comunican/envían información mediante **cuplas (parámetros)**.





Cupla

Modularización: Diagrama de Estructura



EJEMPLO.
El **MODULO A** de un primer nivel de la jerarquía invoca al **MODULO B** del nivel inferior y comparte información mediante cuplas.

La **cupla de entrada** envía información **desde** el MODULO A al MODULO B y la **cupla de salida envía** información desde el MODULO B al MODULO A.