**Universidad Alejandro de Humboldt**

**Faculta de Ingeniería**

**Carrera Ingeniería Informática**

**Trabajo Lenguaje De Programación II**

Autor: Ricardo Sánchez V- 29.583.472

Sección: 0501

Caracas, enero de 2021

Contenido

[Introducción 3](#_Toc62764213)

[Control de datos en procesamiento 4](#_Toc62764214)

[Ambiente de referencia 4](#_Toc62764215)

[Ambiente de referencia local y no local 4](#_Toc62764216)

[Ambiente de referencia local 4](#_Toc62764217)

[Ambiente no local de referencia 4](#_Toc62764218)

[Reglas de Alcance 4](#_Toc62764219)

[Alcance Dinámico 4](#_Toc62764220)

[Alcance estático 4](#_Toc62764221)

[Con Procedimientos Anidados 5](#_Toc62764222)

[Acceso en profundidad 5](#_Toc62764223)

[Paso de parámetro 5](#_Toc62764224)

[Por Valor 6](#_Toc62764225)

[Por Referencia 6](#_Toc62764226)

[Copiar-Restaurar 6](#_Toc62764227)

[Por nombre 6](#_Toc62764228)

[Programación modular 6](#_Toc62764229)

[Encapsulamiento 6](#_Toc62764230)

[Principios de independencia de la representación 7](#_Toc62764231)

[Lenguajes de programación Modular y comparación de esquemas de implementación 7](#_Toc62764232)

[Conclusión 8](#_Toc62764233)

[Bibliografía 9](#_Toc62764234)

# Introducción

En este trabajo se pretende explicar de manera profunda el control de datos de procesamiento y la programación modular, empezando por los conceptos simples como ambiente de referencia, pasando por sus etapas, sus reglas de alcance, paso o transferencia de parámetros ya sea por valor, por referencia, por nombre o copiar-restaurar más importantes, ya que en la actualidad el uso de este paradigma es muy importante, y se debe hondar en este tema lo más específicamente posible.

# Control de datos en procesamiento

## Ambiente de referencia

Cada programa o subprograma tiene un conjunto de asociaciones de identificador disponibles para su uso al hacer referencias durante su ejecución. Este conjunto de asociaciones de identificador se conoce como el ambiente de referencia del subprograma o programa. El ambiente de referencia de un subprograma es ordinariamente invariable durante su ejecución. Se establece cuando se crea la activación del subprograma, y pertenece sin cambio durante el tiempo de vida de la activación. Los valores contenidos en los diversos objetos de datos pueden cambiar, pero no así las asociaciones de nombres con objetos de datos y subprogramas.

## Ambiente de referencia local y no local

### Ambiente de referencia local

Es el conjunto de asociaciones creadas al entrar a un subprograma y que representan parámetros formales, variables locales y subprogramas definidos sólo dentro de ese subprograma conforma el ambiente local de referencia de esa activación del subprograma. El significado de una referencia a un nombre en el ambiente local se puede determinar sin salir de la activación de subprograma.

### Ambiente no local de referencia

El conjunto de asociaciones para identificadores que se pueden usar dentro de un subprograma pero que no se crean al entrar a él se conoce como el ambiente no local de referencia de subprograma.

## Reglas de Alcance

### Alcance Dinámico

Cada asociación tiene un alcance dinámico, el cuál es la parte de la ejecución del programa durante la que esa asociación existe como parte de un ambiente de referencia.

Así, el alcance dinámico de una asociación se compone del conjunto de actividades de subprogramas dentro de las cuales es visible.

### Alcance estático

Determina que declaración aplica a un nombre, examinando el programa (el texto).

Sin Procedimientos Anidados

* Todo nombre no local es global (estático).
* Se implementa fácilmente el paso de procedimientos como parámetros, o el retorno de estos como resultado.

### Con Procedimientos Anidados

* Para un nombre no local aplica la declaración más cercana de él (siguiendo la anidación en la declaración de los procedimientos).
* La anidación en la declaración no implica anidación en el llamado.
* Entonces decidir qué activación corresponde a una variable no local a un procedimiento es algo que se realiza en tiempo de ejecución.
* Profundidad de anidamiento: en un procedimiento asocia el nombre no local con la profundidad a la cual fue declarado.
* El link de acceso apunta al registro de la activación más reciente del procedimiento que lo anida.
* Recorriendo los links de acceso puede llegar al registro de activación que tiene el nombre (sabe en qué nivel de anidación lo está usando y en qué nivel fue declarado).

## Acceso en profundidad

Nuestras infraestructuras de tecnología son una compleja formación de elementos que en conjunto albergan uno de los activos más valiosos para las empresas: los datos. Podemos visualizar esta infraestructura como una serie de capas donde los datos ocupan el último nivel, precedidos de contenedores como lo son las localidades físicas, el perímetro, la red, los servidores y las aplicaciones.

De esta forma, cada capa de nuestra infraestructura representa una barrera para el atacante en su camino hacia el objetivo final de acceder a los datos confidenciales, de manera que si falla cualquiera de los controles en una capa haya defensas adicionales que contengan la amenaza y minimicen las probabilidades de una brecha.

### Paso de parámetro

Cuando un programa principal invoca un subprograma la información se transmite o comunica a través de la lista de parámetros, y se establece una correspondencia automática entre los parámetros actuales y ficticios.

Los parámetros ficticios toman el valor de los parámetros actuales.

Existen diferentes métodos para la transmisión o el paso de parámetros a los subprogramas.

Es preciso conocer el método adoptado por cada lenguaje de programación, ya que esto puede afectar la lógica del programa.

Dicho de otro modo, un mismo programa puede producir diferentes resultados bajo diferentes sistemas de pasos de parámetros.

### Por Valor

En síntesis, en el método de paso por valor una vez que arranca la ejecución del procedimiento, cualquier cambio en el valor de un parámetro ficticio no se refleja en un cambio en el correspondiente parámetro actual. Esto es, cuando es su programa se termine, el argumento actual tendrá exactamente el mismo valor que cuando es subprograma comenzó, independientemente de lo que se haya sucedido al parámetro ficticio. Los parámetros de entrada se denominan parámetros valor.

### Por Referencia

En el método de pasó por referencia o dirección, se envía la posición o dirección (no el valor) del parámetro actual, por lo que un cambio en el parámetro ficticio se refleja en un cambio en el correspondiente parámetro actual, ya que ambos tienen la misma dirección de memoria. Los parámetros de salida o entrada/salida, se denominan parámetros variables.

### Copiar-Restaurar

* El procedimiento llamador evalúa los parámetros actuales, y pasa los n-valores al procedimiento llamado.
* Cuando el control retorna, los n-valores de los parámetros formales son copiados en los parámetros actuales (de aquellos que tienen l valor).

### Por nombre

* El procedimiento actúa como una macro, cuyo cuerpo se copia en el procedimiento llamador, los parámetros actuales sustituyen a los formales.

# Programación modular

### Encapsulamiento

La encapsulación es un mecanismo que consiste en organizar datos y métodos de una estructura, conciliando el modo en el que el objeto se implementa, es decir, evitando el acceso a datos por cualquier otro medio distinto a los especificados. Por lo tanto, la encapsulación garantiza la integridad de los datos que contiene un objeto.

### 

### Principios de independencia de la representación

La independencia funcional se adquiere desarrollando módulos con una clara función evitando una excesiva interacción con otros módulos. Este concepto está derivado del modularidad, la abstracción y el ocultamiento de la información.

### Lenguajes de programación Modular y comparación de esquemas de implementación

Entre los lenguajes de programación que admiten el concepto de programación modular se encuentran C, Ada, PL/I, Erlang, Pascal, Algol, COBOL, RPG, Haskell, Python, HyperTalk, IBM/360 Assembler, MATLAB, Ruby, IBM RPG, SmallTalk, Morpho, Java (los paquetes se consideran módulos), Perl, etc.

# Conclusión

Como pudimos ver la programación modular también tiene importancia en la actualidad, ya que disminuye mucho la complejidad de cualquier tipo de algoritmo haciendo mucho más sencilla la programación y por supuesto sirve para simplificar el diseño. Por otro lado,**la programación modular también hace más pequeño el tamaño total del programa** y sirve para favorecer el trabajo en equipo. Además, la programación modular también ahorra mucho tiempo de programación porque permite reusar los códigos y favorece el trabajo en equipo en cada uno de los módulos para después aplicarlo al conjunto.

# Bibliografía

Baldemarfj (2014). Paradigmas y lenguajes de comunicación [Articulo en Línea] Disponible en: <https://www.clubensayos.com/Tecnolog%C3%ADa/PARADIGMAS-Y-LENGUAJES-DE-PROGRAMACION/1523624.html> [Consulta: 2021, enero 28]

Constanza M. (2018). Compiladores: ambientes de ejecución [Articulo en Línea] Disponible en: <http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:compi:comp_sesion23-24-25_2008_1.pdf> [Consulta: 2021, enero 28]

Flores L. (2014). Encapsulamiento [Articulo en Línea] Disponible en: <https://sites.google.com/a/innovavirtual.org/tecceilpvi/home/vi-ciclo-2014/poo/encapsulamiento> [Consulta: 2021, enero 28]

Guzmán E. (2015). Subprogramas: funciones y procedimientos [Articulo en Línea] Disponible en: <http://webdelprofesor.ula.ve/ingenieria/eliana/prog/unidad6.pdf> [Consulta: 2021, enero 28]

UPT (2009). Diseño de software [Articulo en Línea] Disponible en: <https://es.slideshare.net/simonaparicio/diseo-de-software-2593855> [Consulta: 2021, enero 28]