

UTILIZACIÓN DEL PATRÓN WRAPPER PARA TRADUCIR A LENGUAJE JAVA LOS ALGORITMOS ESCRITOS EN LENGUAJE C QUE USAN PUNTEROS Y SUS IMPLICACIONES EDUCATIVAS

Por:

Milton Jesús Vera Contreras¹

RESUMEN

El presente documento muestra a estudiantes y profesores la posibilidad de utilizar el lenguaje Java como herramienta para el aprendizaje de la programación en sustitución al tradicional lenguaje C. Se centra en el problema particular del uso de punteros en la implementación de algoritmos en C al intentar migrarlos a Java [1, 2, 3]. Muestra, mediante la utilización del patrón estructural Wrapper [4], que los algoritmos implementados en C son fácilmente traducibles a lenguaje Java [2] sin modificaciones considerables en la codificación. Finalmente sugiere implicaciones del uso de Java como lenguaje en el aprendizaje de la programación.

Se pretende ir mucho más allá del acto investigativo y de la escritura del artículo, persiguiendo sembrar la importancia de arriesgarse a escribir, de proponer ideas, de crear y recrear la ciencia y la tecnología. En ocasiones, el estudiante y el profesor consideran como obvias [5] ciertas cosas dentro del acto educativo, el salón de clase, las cuales encierran, como en éste caso, situaciones ocultas que sólo con el tiempo se reflejan, situaciones que bien pueden ser tomadas por objeto de estudio de una investigación. Muchas de las ideas enunciadas en el presente artículo pretenden ser objeto de estudio en una propuesta de semillero de investigación, llamado ALGOMÁTICA [6], donde se busca conjugar el formalismo y la praxis de la programación de computadores, fomentando un espíritu investigativo al tiempo que se gestan situaciones propicias para crear ciencia y tecnología.

ABSTRACT

The present document shows students and professors the possibility of using the Java language as tool for

the learning of the programming in substitution to traditional language C. It is centered in the particular problem of the use of "pointers" in the implementation of algorithms in C, when trying to migrate it to Java[1, 2, 3]. Sample, by means of the use of the structural pattern Wrapper[4], that the algorithms implemented in C are easily translatable to Java language [2] without considerable modifications in the codification. Finally it suggests implications of the use of a specific language in the learning of the programming.

In addition, it is tried to seed the importance of risking to write, to propose ideas, to create and to recreate science and the technology. Sometimes, the student and the professor consider that certain things are obvious [5] within the educative act, the hall class, which lock up hidden situations that only with time they are reflected, situations that can well be taken by study object of an investigation. Many of the ideas enunciated in the present article try to be object of study in a nascent group of investigation, call ALGOMÁTICA [6], which is tried to conjugate the formalism and praxis of the programming of computers, fomenting a research spirit to the time which propitious situations are developed to create science and technology.

Palabras claves: Programación de Computadores, Programación Orientada a Objetos, Lenguaje Java, Patrones de Software, ALGOMÁTICA.

Key words: Programming of Computers, Object-oriented programming, Java Language, Patterns of Software.

INTRODUCCIÓN

La programación es un eje fundamental en la formación profesional del Ingeniero de Sistemas [7]. Para educar en ésta área, las universidades utilizaron a través de la historia diferentes lenguajes en diferentes temas específicos. En los últimos 10 años, el lenguaje

¹ Ing. de Sistemas, Profesor Instructor Departamento de Sistemas e Informática UFPS, Miembro Grupo de Investigación GIDIS, Semillero de Investigación ALGOMÁTICA. mjvera@bari.ufps.edu.co, indioguavita@gmail.com

C y posteriormente el lenguaje C++, fueron las tecnologías empleadas en la mayoría de las universidades, siendo Basic, Pascal, Fortran y Cobol otros lenguajes de épocas anteriores[3,8,9]. Actualmente existe una tendencia marcada a utilizar el lenguaje Java por los diferentes beneficios que ofrece, tanto para el aprendizaje inmediato como para el desenvolvimiento futuro del profesional, así como por las similitudes sintácticas y algorítmicas que tiene con lenguaje C, lo que facilita la transición [1]. Además, dos razones más contundentes para preferir Java son su carácter puramente orientado a objetos y su portabilidad, ésta última característica única entre los lenguajes disponibles, pues permite la compilación y ejecución de programas en prácticamente cualquiera de las plataformas existentes en el mercado[1, 8].

La utilización de Java como lenguaje para el aprendizaje de la programación de computadores reduce la complejidad pues evita comprender la arquitectura de la máquina real, particularmente evita los problemas que ocurrían en lenguaje C al utilizar apuntadores, puesto que Java ofrece una máquina virtual que realiza automáticamente la gestión de memoria[1,2,8]. Sin embargo, dos conceptos muy importantes en programación: el paso de argumentos por valor y por referencia, no consiguen comprenderse fácilmente con Java[2], donde todos los argumentos se pasan por valor pues son objetos que encapsulan su contenido[1]. Ésta notable diferencia entre los dos lenguajes se acentúa cuando se requiere traducir a Java un algoritmo escrito antes en C ó C++. Es, por lo tanto, de valioso interés didáctico tratar de traducir a Java algoritmos escritos en C que usen apuntadores y en éste artículo se pretende utilizar el patrón estructural Wrapper para conseguirlo.

Identificación del Problema

La Universidad Francisco de Paula Santander en su programa de Ingeniería de Sistemas, optó por introducir el lenguaje Java como herramienta en las 4 primeras asignaturas de programación a partir del

primer semestre académico de 2005, en reemplazo del lenguaje C (y C++). Dicha iniciativa se adoptó inmediatamente en el curso Programación III, Estructuras de Datos, y se inició en el segundo semestre del mismo año con el curso de Programación II, encontrándose grandes beneficios dentro de los que se destaca:

- El aprendizaje del paradigma orientado a objetos en paralelo con el tema de estudio de la asignatura. Cuando se empleaba lenguaje C, el aprendizaje de éste paradigma se aplazaba hasta el curso de programación IV donde se evolucionaba al C++.
- El estudio comparativo de dos lenguajes de programación en paralelo con el estudio de un área específica de la algorítmica. Para el caso particular de los estudiantes que reprobaron la asignatura cuando se utilizaba lenguaje C y para los que aprobaron la asignatura pero aún no terminan su ciclo de programación, por lo que participan en la transición.

De la misma manera se encontraron inconvenientes durante el aprendizaje, originando la necesidad de crear estrategias de programación y estrategias educativas para conseguir avanzar en los objetivos de la materia. Dentro de los inconvenientes encontrados se destacan tres:

- La diferencia en el modelo de administración de memoria empleado por los dos lenguajes en cuestión. Mientras lenguaje C emplea punteros y paso de argumentos por referencia y valor, lenguaje Java emplea referencias a objetos y paso de argumentos sólo por valor [1].
- La diferencia en librerías de clases proporcionadas por los dos lenguajes. Mientras Java es prolijo en clases que reducen sustancialmente la codificación de rutinas clásicas, C requiere un mayor trabajo de codificación.

Utilización del patrón Wrapper para traducir a lenguaje Java los algoritmos escritos en lenguaje C que usan punteros y sus implicaciones educativas

- La diferencia en el paradigma del lenguaje C, programación estructurada, empleado en el área prerrequisito, frente al nuevo paradigma, Orientado a Objetos, empleado por Java [8].

El presente artículo se centra en el primer problema, la restricción de acceso a memoria que presenta el lenguaje Java, y sugiere una solución algorítmica después de realizar un estudio formal de los dos lenguajes para el caso específico y plantea interrogantes e hipótesis que persiguen ser el inicio de una investigación más profunda.

Formulación del Problema

La diferencia en el modelo de administración de memoria empleado por lenguaje C frente a lenguaje Java en el contexto del aprendizaje de las estructuras de datos en lenguaje Java después de haber aprendido los conceptos fundamentales de programación en C lleva a plantear el siguiente interrogante:

¿Es posible traducir a lenguaje Java un algoritmo escrito en lenguaje C que use punteros, con el menor número posible de cambios en la codificación?

Hipótesis

Sí es posible traducir a lenguaje Java un algoritmo escrito en lenguaje C que use punteros, con un número mínimo de cambios en la codificación siempre que se cumplan las siguientes restricciones:

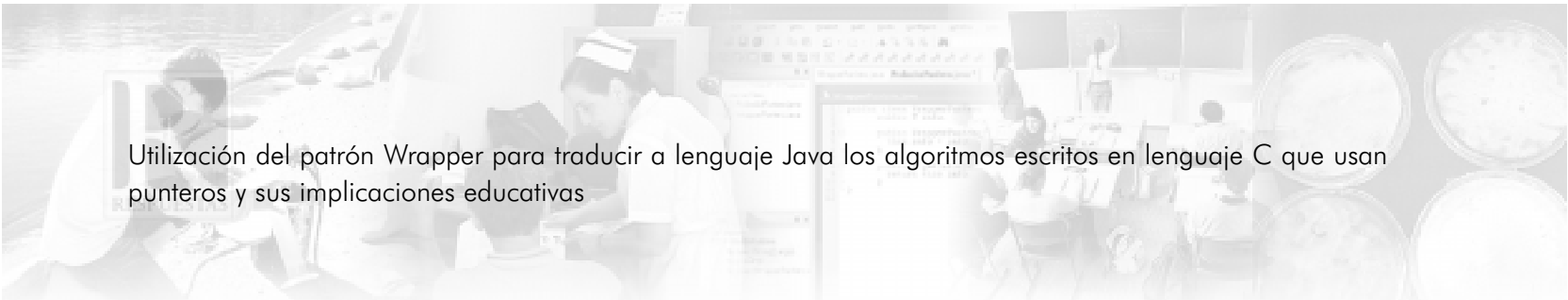
- El algoritmo inicial no emplea librerías específicas de lenguaje C.
- El algoritmo inicial separa la entrada, procesamiento y salida asegurando la portabilidad y encapsulamiento de estos elementos.
- El algoritmo procesa un conjunto de argumentos definidos formalmente y regresa un resultado.
- El algoritmo es invocado desde cualquier otro algoritmo que se encarga de obtener los datos de entrada y mostrar los datos de salida.

Cuando estas restricciones no se cumplen, una versión simplificada del programa, que las cumpla, puede conseguirse con transformaciones mínimas.

Marco Conceptual

Algoritmo: El término algoritmo se emplea para denotar un "Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema." [10]. El origen etimológico no es muy claro: "Quizá del Lat. tardío *algorismus*, y este abreviado del árabe clásico: Cálculo mediante cifras arábicas". El origen histórico data del año 780 DC en Arabia, por el matemático Muhammad ibn Musa abu Djafar Al-Khorezmi [11, 12]. Conocido como el primer matemático en la historia que propuso soluciones procedimentales ó algorítmicas a problemas matemáticos, permitiendo el fácil entendimiento por parte de no expertos. Se le reconoce un algoritmo para restar de un número otro mayor, lo que dió origen al uso del número cero: "Cuando en una resta nada queda, entonces escribe un pequeño círculo para que ese lugar no permanezca vacío" [11]. Antes de Al-Khorezmi, Euclides (300 A..C) ya había formulado su algoritmo para calcular el Máximo Común Divisor de dos números. Actualmente, la Algorítmica se considera dentro de las ciencias de la computación y es el pilar de la Ingeniería de Sistemas, la Informática y las tecnologías basadas en computadores [7, 13]. (Nota: Aunque para el objetivo del artículo es irrelevante el origen etimológico del término, se menciona por la relevancia que tiene la etimología al interior del Semillero de Investigación ALGOMÁTICA.)

Programación de Computadores: Mientras la Algorítmica se considera una ciencia formal, ligada a las matemáticas, la programación de computadores se constituye en la técnica que permite hacer funcionales los algoritmos [9, 13, 14, 15, 16]. La programación de computadores se ocupa de implementar los formalismos algorítmicos para que estos puedan emplearse en un computador y solucionar problemas con datos y/o información real.



Utilización del patrón Wrapper para traducir a lenguaje Java los algoritmos escritos en lenguaje C que usan punteros y sus implicaciones educativas

Lenguaje de Programación: Un lenguaje de programación es un conjunto de instrucciones básicas que pueden ser comprendidas por el computador o pueden traducirse a un lenguaje comprendido por el computador [14,15]. A través de la historia los lenguajes de programación han evolucionado y se han originado diferentes técnicas de programación, casi siempre ligadas a un lenguaje en particular. Como lenguajes históricamente importantes se tienen Fortran, Basic, Pascal, C, C++, Visual Basic y Java.

Compilador e Interprete: Los compiladores e interpretes son programas de computador especializados en leer la implementación de algoritmos escritos en algún lenguaje de programación. Ambos verifican que la implementación cumpla con las reglas gramaticales, sintácticas semánticas y en general las reglas del lenguaje. Los compiladores realizan una verificación completa y generan posteriormente una versión ejecutable del algoritmo, la cual puede probarse en un computador como un programa independiente. Los intérpretes en cambio, de manera análoga a un intérprete de algún idioma, realizan la verificación y ejecución paso por paso, siendo cada paso una instrucción que por lo general es una línea dentro del código [14,15].

Identificador: Dentro de un lenguaje de Programación, un identificador es una palabra válida para el lenguaje que el programador define para nombrar un elemento de datos o de código en la implementación de un algoritmo [3, 14,15].

Localidad de Memoria: Una localidad de memoria es una abstracción de un espacio en la memoria física. Los bits de la memoria física se organizan en bytes y cada uno de estos se enumera desde 0, cero, para facilitar su acceso. Una localidad de memoria puede ocupar más de un byte por lo que el número del byte donde inicia la localidad de memoria se considera como su dirección base [1, 3].

Dirección de Memoria: En un programa de computador que se está ejecutando, una dirección de memoria es un número, por lo general escrito en base hexadecimal, que representa el número de byte en que inicia una localidad de memoria donde se encuentra un dato o un bloque de instrucciones [1, 3].

Tipo de dato: Un tipo de dato es un elemento semántico de los lenguajes de programación que permite interpretar una secuencia de bits. Bit es un acrónimo de las palabras en inglés binary-digit y se usa para denotar los valores del sistema de numeración binario, 1 y 0, que simbolizan los estados digitales usados en un computador para representar y almacenar datos [3, 13, 14]. Puesto que los datos se almacenan como bits, es necesario interpretarlos. La interpretación común de todo lenguaje de programación es en términos de números y caracteres. Los números pueden ser de cualquiera de los conjuntos numéricos definidos en matemática, y los caracteres se restringen a un sistema de codificación, siendo el sistema base el ASCII o el UNICODE. Los tipos de datos más comunes son el numérico entero, numérico real o de punto flotante, carácter simple, cadena de caracteres y booleano o lógico (falso, verdadero). La mayoría de los lenguajes de programación proporcionan elementos sintácticos y semánticos para especificar nuevos tipos de datos más complejos, conformados por los tipos de datos básicos [1,8].

Variable: Una variable es un elemento semántico de los lenguajes de programación que permite abstraer con facilidad una localidad de memoria. Toda variable se constituye de cuatro componentes: un nombre, una dirección de memoria, un tipo de dato y un valor. El nombre sirve como mnemónico para referirse al valor almacenado en una localidad de memoria. La localidad de memoria tiene una dirección y el valor es de un tipo de dato. Gráficamente, una variable en la memoria del computador puede verse como lo muestra la Figura 1. En la figura, el valor de la variable se muestra en dos sistemas de numeración, el decimal, para los humanos y el binario, para la máquina.

La dirección, en cambio, se muestra en hexadecimal aunque internamente lo que se manipula es su equivalente binario [14,15].

Figura 1. Esquema gráfico de una Variable.

Nombre	Dirección	Tipo de Dato	Valor
Suma	0EF01AH	Numérico Entero (int)	00001010
			10

Puntero o Apuntador: En lo referente al manejo de memoria, existen dos enfoques en los lenguajes de programación según se delegue o no al programador la administración de memoria [1]. Los lenguajes que no delegan la administración de la memoria, proporcionan mecanismos de control de memoria que se adjuntan a la versión ejecutable del programa. Los lenguajes de programación que delegan la administración de memoria, incluyen dentro de sus elementos semánticos y sintácticos el apuntador (del inglés pointer o puntero). Un puntero es un tipo de dato muy particular que almacena una dirección de memoria. Puesto que una dirección de memoria está determinada por un número en base hexadecimal, una variable del tipo puntero almacena un número y dicho número se interpreta como una dirección de memoria. El compilador se encarga de manejar los aspectos sintácticos y semánticos que posibilitan el acceso a memoria. Puesto que un puntero guarda una dirección de memoria, el lenguaje de programación debe proporcionar mecanismos para obtener la dirección de memoria de una variable y para obtener el valor almacenado en una dirección de memoria [1,14]. Aunque no es muy común, también es posible hallar elementos del lenguaje que permiten acceder al nombre dado a una dirección de memoria.

Patrones de Diseño: En ingeniería de software y en programación, siempre se buscan soluciones generales a los problemas de modo que puedan reutilizarse en situaciones particulares, lo cual es propio de todas las ciencias. En la actualidad, uno de los logros en la materia son los patrones de diseño de software. Un

patrón de diseño de software es una solución genérica, modelada y/o implementada que puede reutilizarse en diferentes situaciones específicas. Los patrones se clasifican en patrones de creación, estructurales y de comportamiento [4].

Patron Wrapper: El patrón Wrapper o Adapter, comúnmente traducido como envoltorio es un patrón estructural. Los patrones estructurales permiten especificar la forma como se organizan las clases y los objetos para alcanzar una solución. El patrón Wrapper permite convertir la interfaz de acceso a un objeto a otra interfaz requerida. De ésta forma, dos interfaces incompatibles pueden cooperar entre sí [4].

MATERIALES Y MÉTODOS

Se tiene como objetivo implementar el patrón Wrapper para conseguir que un algoritmo escrito en lenguaje C pueda describirse en lenguaje Java con un cambio mínimo en el código. Siguiendo las restricciones de la hipótesis y los fundamentos conceptuales expuestos, se propone como método para demostrarla el siguiente silogismo:

- Un programa escrito en lenguaje C puede traducirse a otro lenguaje (bajo las restricciones establecidas en la hipótesis) si éste proporciona elementos semánticos similares a los que emplea el programa.
- Un puntero en lenguaje C es una construcción semántica del lenguaje C.
- En Java, puede construirse una clase (un elemento semántico) Puntero basada en el patrón estructural Wrapper que encapsule la misma abstracción que el puntero de lenguaje C.
- Como consecuencia lógica, todo programa escrito en C (que cumpla con las restricciones) se puede traducir a lenguaje Java con modificaciones mínimas en el código.

La verificación del razonamiento implica realizar un ejercicio real de codificación en lenguaje C y lenguaje Java donde se evidencien cada uno de los juicios expuestos en el silogismo.

Utilización del patrón Wrapper para traducir a lenguaje Java los algoritmos escritos en lenguaje C que usan punteros y sus implicaciones educativas

Metodológicamente, se seguirá el itinerario experimental propuesto a continuación:

Paso 1: Codificar en lenguaje C un algoritmo que utilice apuntadores.

Paso 2: Crear en lenguaje Java el envoltorio Puntero (wrapper Pointer) que proporcione la misma funcionalidad que el puntero de C.

Paso 3: Escribir en Java el mismo algoritmo escrito en C.

Paso 4: Aquellas instrucciones donde el programa escrito en C use punteros, cambiar el puntero por un objeto wrapper de la clase creada en el paso 2.

Paso 5: Probar la ejecución de las dos implementaciones.

Para tal fin, se requieren de los siguientes instrumentos tecnológicos:

- Hardware:
 - CPU versión mínima 586.
- Software:
 - Entorno integrado de desarrollo para Java JSDK versión 1.5 o posterior.
 - Entorno integrado de desarrollo para lenguaje Java, BlueJ, JCreator o NetBeans.
 - Entorno integrado de desarrollo para lenguaje C. Turbo C, DevCpp-GNU.
 - Plataforma operativa Windows o Linux.

Para realizar la verificación se empleará el algoritmo de intercambio swap, comúnmente empleado en algoritmos de ordenamiento. El algoritmo swap toma como argumento dos elementos del lenguaje que contienen direcciones de memoria diferentes e intercambia sus valores [3, 14].

RESULTADOS

La codificación en lenguaje C y Java de los algoritmos se presenta a continuación. Como convención se muestra en negrilla las palabras claves del lenguaje.

El algoritmo swap escrito en lenguaje C:

```
#include <stdio.h>

void swap(int *p, int *q){
    int tmp = *p;
    *p = *q;
    *q = tmp;
}
```

El wrapper Puntero escrito en Java:

```
/**
 * Abstrae la dirección, la localidad y el valor, como en un puntero. Puesto que en lenguaje C cada puntero determina el tipo de dato de la variable a la cual apunta, se declara la clase Puntero como genérica, T.
 */
public class Puntero<T>{
    /**La referencia a un objeto de la clase T, genericidad*/
    protected T value;

    /**Constructor*/
    public Puntero(T value){
        this.value = value;
    } //Fin constructor

    /**Obtiene el objeto que referencia, como el "&" en C*/
    public T getValue(){
        return this.value;
    } //Fin getValue

    /**Modifica el objeto que referencia, como el "*" en C*/
    public void setValue(T value){
        this.value = value;
    } //Fin setValue
} //Fin clase Puntero
```

Utilización del patrón Wrapper para traducir a lenguaje Java los algoritmos escritos en lenguaje C que usan punteros y sus implicaciones educativas

El algoritmo swap escrito en lenguaje Java:

```
/**
El algoritmo se escribe fuera de una
clase, pero en Java debe existir una
clase. Obsérvese la similitud en el
código. En algoritmos más complejos
como los empleados en estructuras de
datos como listas, pilas, colas, árboles
y en general, la utilidad es bastante
grande porque permite llevar fácilmente
algoritmos de un lenguaje a otro.
*/
void swap(Puntero<Integer> p,
Puntero<Integer> q){
    int tmp = p.getValue();//int tmp =
    *p;
    p.setValue(q.getValue());// *p = *q;
    q.setValue(tmp); // *q = tmp;
} //fin swap
```

El programa de prueba escrito en lenguaje C:

```
int main(int argc, char * argv[]){
    int a, b;
    int *p=&a;
    int *q = &b;

    scanf("%d",&a);
    scanf("%d",&b);

    printf("antes");
    printf("p apunta a: %p y su valor es
%d",p,*p);
    printf("q apunta a: %p y su valor es
%d",q,*q);

    swap(p,q);

    printf("despues");
    printf("p apunta a: %p y su valor es
%d",p,*p);
    printf("q apunta a: %p y su valor es
```

```
%d",q,*q);
    exit(0);
}
```

El programa de prueba escrito en lenguaje Java:

```
/**
Se omiten las clases Teclado y Pantalla y se dejan como
abstracción para simplificar la lectura
*/
public class Main{
    public static void main(String args
[]) throws Exception{
        Teclado teclado = new Teclado();
        Integer a = teclado.leerEntero();
        Integer b = teclado.leerEntero();
        Puntero p = new Puntero(a);
        Puntero q = new Puntero(b);

        Pantalla.imprimir("antes");
        Pantalla.imprimir("p envuelve a " +
p.getValue());
        Pantalla.imprimir("q envuelve a " +
q.getValue());
        swap(p,q);
        Pantalla.imprimir("despues");
        Pantalla.imprimir("p envuelve a " +
p.getValue());
        Pantalla.imprimir("q envuelve a " +
q.getValue());

    } //fin main
} //Fin clase de prueba
```

DISCUSIÓN

Al probar las dos implementaciones, como se propuso, el resultado es el mismo. Aunque en el mundo del software la solución propuesta no es de mucha utilidad, si es un ejercicio muy interesante para que el estudiante consiga comprender la diferencia entre el uso de apuntador y de variables de referencia.

Declarar Puntero como una clase genérica, además de utilizar una de las principales abstracciones de la programación orientada a objetos en Java, evita la conversión de tipos de datos que opacaría la utilidad del wrapper.

En lo referente al diseño de patrones de software, se puede apreciar que el patrón Wrapper es bastante elemental y que su utilidad se encuentra en la migración de aplicaciones de una plataforma a otra, a fin de migrar la semántica de una plataforma a la sintaxis y semántica de otra, reutilizando el trabajo ya desarrollado en la aplicación original.

A partir de este razonamiento y la verificación efectuada, surgen diferentes interrogantes que es importante resolver y para lo cual pueden plantearse pequeños proyectos de investigación factibles de realizar dentro del grupo de clase incluyendo a los estudiantes en dichos proyectos.

En primer lugar, surge la incógnita de si los estudiantes realmente están comprendiendo el concepto de puntero en las asignaturas de programación o si sólo están mecanizando él ¿cómo hacer?, quedando por fuera el elemento formal tan indispensable para el profesional universitario.

En segundo lugar, queda el vacío de la pertinencia o no del lenguaje Java en el aprendizaje de la programación, la duda de, en qué momento del proceso de aprendizaje (programación I, II, III, IV, V) debe involucrarse éste lenguaje y más particularmente el paradigma de orientación a objetos. Se vislumbra entonces la necesidad de revisar muy detalladamente los contenidos de las asignaturas de programación, el estado actual de la ingeniería del software y su futuro próximo y las estrategias para cambiar un paradigma que no desaparecerá por completo pero que está en desuso en etapas superiores de la programación y el software.

De manera preliminar, se proponen tres prototipos de aprendizaje y consecuentemente de estudiantes en el ámbito de la programación de computadores.

- Un modelo de aprendizaje basado en la sintaxis. En éste modelo, el estudiante aprende a proponer soluciones algorítmicas ceñido a un esquema sintáctico proporcionado por el lenguaje específico que se emplea. Éste modelo ocasiona serios inconvenientes cuando en la tecnología se dan cambios de paradigma como el presentado en éste artículo.
- Un segundo modelo se basa en la praxis. En éste, el estudiante se apropia de un conocimiento sobre cómo hacer ciertas cosas, nuevamente en cierto lenguaje. Se encuentran expresiones como ¿cómo hago en java un getch? . Éste modelo, aunque facilita el aprendizaje de otros lenguajes, no es aceptable teniendo en cuenta que, nuevamente, un cambio de paradigma y un problema de programación diferente a los aprendidos en el primer lenguaje, determinan un retraso en el aprendizaje de nuevas tecnologías y tendencias.
- El tercer modelo, que es el considerado conveniente, corresponde a un modelo semántico, donde el estudiante, al igual que cuando aprende en su escuela a leer y a escribir, se apropia de un conocimiento sistémico, donde aprende a entrelazar la gramática, la sintaxis y la semántica, donde por ejemplo, puede reconocer la diferencia entre un verbo y un sustantivo así como entre dos palabras homófonas y además, es capaz de emplear dicho conocimiento para comunicarse.

Finalmente, el ejercicio pretende ir mucho más allá del acto investigativo y de la escritura del artículo, pretende crear y recrear la ciencia y la tecnología como estrategia didáctica para formar investigadores. El estudiante y el profesor obvian, dentro del acto educativo, el salón de clase, ideas que bien pueden

ser tomadas por objeto de estudio de una investigación. Muchas de las ideas enunciadas en el presente artículo, pretenden ser el punto de inicio para una propuesta de semillero de investigación llamado ALGOMÁTICA, donde se pretende conjugar el formalismo y la praxis de la programación de computadores, fomentando un espíritu investigativo al tiempo que se gestan situaciones propicias para crear ciencia y tecnología [6].

CONCLUSIONES

Un algoritmo implementado en lenguaje C que utilice apuntadores puede fácilmente escribirse en Java siempre que se elaboren los elementos semánticos que emulen la funcionalidad de los punteros.

Crear una clase wrapper en Java para simular los punteros de lenguaje C es un ejercicio interesante que permite evidenciar situaciones problemáticas en el aprendizaje de la programación de computadores.

Orientar la investigación en ingeniería de sistemas al aula y a situaciones sencillas como la presentada en el artículo es un ejercicio conveniente para sembrar actitud investigativa y creativa en los estudiantes desde los primeros semestres.

En ingeniería de sistemas puede hacerse investigación formal en el área de la programación a través de artículos de reflexión que siembren una actitud crítica ante la ciencia y la tecnología.

La experimentación en ingeniería de sistemas puede realizarse en el área de la programación como mecanismo de verificación de las propuestas que surgen de la investigación formal.

AGRADECIMIENTOS

Vicerrectoría de Investigación y Extensión por el Taller de Capacitación para la escritura de Artículos Científicos y Revistas Indexadas.

Departamento de Sistemas e Informática, Grupo de Investigación en Desarrollo e Ingeniería del Software: Ing. Judith del Pilar Rodríguez Tenjo e Ingeniero Oscar Alberto Gallardo Perez, por su invitación a formar parte del grupo y su invitación a participar en el Taller.

Revista RESPUESTAS y pares evaluadores por su valioso aporte en la formación del autor como escritor de artículos científicos.

BIBLIOGRAFÍA

[1] ECKEL, Bruce, Thinking Java. Segunda Edición. PEARSON EDUCACIÓN, S.A. Madrid, 2002. ISBN:84-205-3 192-8 (Traducción del Original en Inglés ISBN0-13-027363-5).

[2] MEZA, Federico, GARRIDO, Ruth y ASTUDILLO, Cesar. Un Lenguaje de Bajo Nivel como Apoyo al Aprendizaje en el Primer Curso de Programación de las Carreras de Ingeniería. Revista de La Sociedad Chilena. Sociedad Chilena de Ciencia de la Computación (SCCC), Documento en línea disponible en <http://www.sccc.cl/revistaSCCC2003.zip>. Febrero 17 de 2005.

[3] TENENBAUM, AARON M; LANGSAM, YELIDYAH; PULIDO CEJUDO, JAVIER; AUGENSTEIN, MOSHE J. ESTRUCTURAS DE DATOS EN C. 1994. ISBN: 968-880-798-2

[4] LARMAN, CRAIG HERNANDEZ RODRIGUEZ, LUZ MARIA. UML y Patrones: Introduccion al Analisis y Diseño Orientado a Objetos. ISBN: 970-17-0261-1. México 1999.

[5] BAEZA Yates, Ricardo, Diseñemos Todo de Nuevo: Reflexiones sobre la Computación y su Enseñanza. Revista Colombiana de Computación Vol. 1, N° 1 Diciembre 2000, Págs. 7-28. ISSN 1657 – 2831.

[6] VERA Contreras, Milton Jesús. ALGOMÁTICA. Una propuesta temática para semilleros de Investigación

Utilización del patrón Wrapper para traducir a lenguaje Java los algoritmos escritos en lenguaje C que usan punteros y sus implicaciones educativas

en Matemática e Informática. Documento Electrónico publicado en PIAGEV. Disponible en <<http://dptosist.ufps.edu.co/algomatica.pdf>>. Octubre de 2004

[7] ACOFI, ICFES. Marco de Fundamentación Conceptual Especificaciones de Prueba ECAES Ingeniería de Sistemas versión 6. Documento electrónico disponible en <http://200.14.205.63:8080/portalicfes/home_2/rec/arc_4400.pdf>. Agosto de 2005.

[8] DEITEL, Cómo Programar en Java. Cuarta Edición. México 1998. ISBN: 0-13-263401-5.

[9] JOYANES, AGUILAR LUIS. FUNDAMENTOS DE PROGRAMACION : ALGORITMOS Y ESTRUCTURAS DE DATOS. 1996 . ISBN: 84-481-0603-2.

[10] Diccionario de la Real Academia de la Lengua Española DRAE. Aplicación de consulta en línea disponible en <<http://buscon.rae.es/diccionario/drae.htm>>.

[11] BAEZA Yates, Ricardo. Un Matemático Olvidado, Revista Ciencia al Día. Volumen 1 No 1. Documento en línea disponible en <<http://www.ciencia.cl/CienciaAlDia/volumen1/numero1/articulos/articulo2.html>>. Octubre 12 de 2004.

[12] BAEZA Yates, Ricardo. Un Matemático Olvidado, Dpto. de Ciencias de la Computación Univ. de Chile Santiago, Chile, Documento en línea disponible en <http://www.dcc.uchile.cl/~rbaeza/inf/alk.html>. Octubre 12 de 2004.

[13] KNUTH, D. E. The Art of Computer Programming Vol 3: Sorting and Searching, Addison Wesley, Reading, MA. ISBN 0201485419 .

[14] Universidad Nacional de Colombia. Curso Virtual de Programación de computadores. Documento electrónico disponible en <http://www.virtual.unal.edu.co/cursos/ingenieria/2001839/index.html>. Agosto de 2004.

[15] Universidad Nacional de Colombia. Curso Virtual de Análisis y Diseño de Algoritmos. Documento electrónico disponible en <http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060024/index.html>. Agosto de 2004.

[16] WIRTH, NIKLAUS. ALGORITMOS Y ESTRUCTURAS DE DATOS. 1989. ISBN: 968-880-113-5

Fecha recibido: Marzo 14 de 2006

Fecha aceptación: Junio 15 de 2006