La enseñanza y el aprendizaje de la programación y la programación concurrente con DaVinci Concurrente

Beatriz O. Depetris, Daniel Aguil Mallea, Horacio Pendenti, Germán Tejero, Guillermo Feierherd, Guillermo Prisching

Instituto de Desarrollo Económico e Innovación
Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur
Hipólito Irigoyen 880 - Ushuaia - Tierra del Fuego

{bdepetris, daguil, hpendenti, ctejero, gfeierherd, gprisching}@untdf.edu.ar

Resumen

Las dificultades en la enseñanza y el aprendizaje de la programación, entendida en el sentido amplio de resolución de problemas computacionales, han llevado al desarrollo de herramientas que contribuyan a facilitar ambos procesos y a experimentar distintas metodologías de enseñanza.
El objetivo de este trabajo es presentar algunas conclusiones sobre las experiencias realizadas durante los dos últimos años en la enseñanza y el aprendizaje inicial de la programación y la programación concurrente utilizando DaVinci Concurrente, un entorno integrado de desarrollo (IDE) que facilita la comprensión de las situaciones problemáticas y permite la visualización de la ejecución de los algoritmos que la resuelven.
A partir de las ventajas y desventajas detectadas se proponen tanto mejoras al IDE como nuevas formas de uso, que permitan reforzar las primeras y disminuir las segundas. Algunas de esas mejoras están en proceso de desarrollo y algunos de los nuevos modos de uso se han comenzado a aplicar durante el corriente ciclo académico.

Palabras clave: Enseñanza de la programación, aprendizaje de la programación, programación concurrente, visualización de la ejecución de algoritmos

Introducción

La enseñanza y el aprendizaje de la programación de computadoras es uno de los temas más desafiantes de las ciencias de la computación [1], y ha dado lugar a numerosas investigaciones en la búsqueda de herramientas y metodologías que contribuyan a disminuir las dificultades que docentes y alumnos encuentran en ambos procesos.
Por otra parte, más allá de la importancia que el tema tiene dentro de la disciplina, en razón de constituir una habilidad imprescindible para un profesional informático, la programación de computadoras "suscita interés psicopedagógico debido a sus efectos en el desarrollo de las capacidades cognitivas." [2]
Existe cierto consenso en sostener que, tanto las dificultades como el interés, se deben a que aprender a programar no sólo significa adquirir un conocimiento nuevo, sino, fundamentalmente, aplicar ese conocimiento para resolver problemas. Se supone, además, que una vez adquirida la capacidad para resolver problemas en el ámbito de la programación de computadoras, la misma puede ser transferida a otros campos.
Las últimas consideraciones son válidas cuando la acción de programar computadoras se entiende en un sentido amplio, que incluye no sólo la codificación de una solución en un lenguaje que la computadora pueda entender, sino todas las instancias previas que llevan a la "resolución del problema computacional".

En términos generales la expresión "resolución de problemas" puede definirse como el conjunto de actividades no rutinarias que ponen en juego procesos de razonamiento que permiten, "partiendo de un estado inicial de información e incertidumbre" arribar a una meta, es decir, "a un estado final que se denomina solución, en el que la incertidumbre se ha reducido o eliminado." [3].

Más específicamente, Salgado Castillo describe en [3] el proceso completo de resolución de problemas de programación computacional como un proceso constituido por dos etapas claramente definidas: la modelación matemática del problema y su sistematización algorítmica.

Durante la primera etapa el primer paso es comprender la situación problemática, para, a partir de ello, lograr una interpretación matemática de la misma que concluya finalmente en una representación matemática.

Es evidente que ese primer paso requiere que el "solucionador" disponga de un conjunto de conocimientos previos que le permitan "comprender la situación", condición necesaria (aunque no suficiente) para poder luego interpretarla y finalmente lograr su representación matemática.

Por otra parte es importante señalar que los dos últimos pasos llevan implícita una "generalización del problema", por lo que para su concreción es imprescindible que el "solucionador" haya desarrollado ya alguna capacidad de abstracción.

A su vez, la segunda etapa (sistematización algorítmica) incluye la identificación de estructuras lógico-computacionales y la integración jerárquica de dichas estructuras.

El primero de los pasos de esta segunda etapa requiere reconocer las propiedades y funciones de cada estructura, diferenciando las esenciales de las que no lo son. No obstante, esta identificación no basta para lograr la solución a la situación problemática, pues para ello se deben concatenar adecuadamente algunas de las estructuras lógico-computacionales identificadas, conformando así el algoritmo que resuelve el problema planteado.

Situación clásica

En un artículo previo hemos sostenido que los alumnos que concurren a los cursos iniciales de programación de nuestra carrera (Expresión de Problemas y Algoritmos y Algorítmica y Programación I) muestran importantes debilidades para resolver aún los problemas más elementales. En esa oportunidad hemos atribuido estas debilidades a sus dificultades para *comprender la situación problemática* que les planteamos, consecuencia de una falta de los conocimientos previos necesarios, y a un bajo desarrollo de las capacidades vinculadas a la abstracción, que les impide generalizar y, por ende, *interpretar y representar matemáticamente la situación problemática*. [4]

Por otra parte, las conversaciones con colegas de otras regiones del país con los que interactuamos habitualmente, nos permiten asegurar que las características que detectamos en nuestros alumnos se repiten en la mayoría de los que participan de los cursos iniciales de programación en las universidades argentinas.

Ello ha llevado a que, desde hace varios años, en muchos cursos introductorios de programación (CS1 y CS2) se recurra al uso de distintas herramientas, cuyas características principales son la de posibilitar presentar problemas en un ambiente "reducido", y la de permitir visualizar distintos aspectos de los programas (en particular la ejecución de los mismos).

La primera de las características mencionadas, en nuestra opinión basada en la idea de "microcosmos" de Papert, busca restringir el conjunto de posibles *situaciones problemáticas*, facilitando su comprensión, pues para ello se requiere un mínimo de *conocimientos previos* que pueden ser fácilmente brindados a los alumnos en los cortos períodos de tiempo disponibles en estas asignaturas de introducción a la programación.

La segunda característica permite, básicamente, trabajar sobre el pensamiento abstracto en base a situaciones de experimentación concreta, buscando que en dicho proceso se "vayan desarrollando las

habilidades para entender conceptos abstractos, manipular símbolos, razonar lógicamente y generalizar." [5]

Situación actual

La situación descrita previamente se complica en la actualidad como consecuencia de la evolución del hardware.
La generalización de dispositivos con capacidad de realizar multiprocesamiento (en razón de disponer de varios núcleos o procesadores) ha llevado a que ya no resulte suficiente que el alumno (futuro profesional) profundice en la programación tradicional (procesos secuenciales) y adquiera sólo conocimientos teóricos básicos de programación concurrente y paralela, como sucedía hasta hace apenas unos años. Por el contrario, ahora debe tener un conocimiento mucho más profundo de estos últimos temas, y debe ser capaz de diseñar soluciones a los problemas que puedan implementarse mediantes procesos concurrentes y paralelos que aprovechen la multiplicidad de recursos disponibles en el hardware. Es evidente que esta nueva situación plantea nuevos desafíos a la enseñanza y el aprendizaje inicial de la programación. [6]
Estas nuevas condiciones, entre otras, fueron las que generaron la idea de evolucionar el Visual DaVinci, un entorno de programación desarrollado en el III-LIDI de la Universidad Nacional de La Plata [7] para facilitar la enseñanza y el aprendizaje inicial de la programación, y que habíamos utilizado durante varios años en nuestra universidad.
De esta evolución surgió DaVinci Concurrente (DVC), que ha sido utilizado en los cursos iniciales de programación y de programación concurrente durante los últimos tres años.

Este artículo está organizado como sigue. En primer lugar se mencionan brevemente las principales características del Da Vinci Concurrente (DVC). Luego se presentan algunas conclusiones sobre su uso en las asignaturas iniciales de programación y de programación concurrente, incluyendo una breve síntesis de las mismas. Finalmente se enumeran algunos trabajos futuros para evolucionar el DVC así como nuevos modos de uso, surgidos en todo los casos como consecuencia de las experiencias realizadas.

Origen, objetivos y características del DVC

El DVC es un lenguaje de programación y un entorno integrado de desarrollo (IDE), obtenidos como resultado de dos tesis de grado realizadas en la Sede Ushuaia de la Universidad Nacional de la Patagonia San Juan Bosco (actualmente Universidad Nacional de Tierra del Fuego, Antártida e Islas del Atlántico Sur).
Las ideas que dieron lugar a las dos propuestas de tesis surgieron luego de varios años de utilizar el VDV en las primeras asignaturas de programación. Durante ese período, a partir de observaciones sobre el comportamiento de los alumnos, se fueron recolectando sugerencias de mejoras al VDV.
Por otra parte, la importancia que iba adquiriendo la programación concurrente y paralela, llevaron a pensar en la posibilidad de reducir las aún mayores dificultades que plantea su enseñanza y aprendizaje recurriendo a herramientas similares a las ya utilizadas, con cierto éxito, en los mismos procesos de la programación secuencial.
En el nuevo contexto que comenzaba a plantear el hardware, se entendió que una extensión del VDV, que además de mejorar algunos de sus aspectos permitiera la posibilidad de trabajar con varios robots, colaborando entre ellos para realizar una tarea, facilitaría a los alumnos la programación de los primeros algoritmos concurrentes. Las ventajas más obvias serían la de minimizar el tiempo de aprendizaje del lenguaje, y fundamentalmente, permitir la visualización de la ejecución de los mismos.
Planteados los requerimientos se analizó la posibilidad de modificar el producto existente o desarrollarlo por completo, optando finalmente por esta última alternativa. Como

resultado de los trabajos se obtuvo el DVC, un producto desarrollado bajo los principios del software libre y disponible con licencia GPL en http://davinci-c.sourceforge.net/

Como lenguaje el DVC mantiene algunas características del VDV, entre ellas una sintaxis similar a Pascal. Esto facilita, superadas las instancias iniciales del aprendizaje, comenzar a utilizar este último lenguaje en la codificación de soluciones a problemas de mayor diversidad que los que pueden plantearse con DVC.

Al igual que en el VDV, el conjunto de comandos básicos disponibles en DVC sigue siendo reducido, limitado a los imprescindibles para desplazar un robot en una ciudad compuesta de calles y avenidas perpendiculares entre sí. Estos incluyen *mover* (que mueve el robot hacia adelante hasta la siguiente intersección) y *derecha* (que hace girar el robot noventa (90) grados a la derecha). Adicionalmente se cuenta con los necesarios para que el robot pueda realizar algunas tareas elementales, como recoger o depositar objetos (papeles y flores), ubicados previamente en las intersecciones de la ciudad o transportados por el robot en una "bolsa" que lleva consigo.

No obstante el DVC incorpora nuevas facilidades, entre las que cabe mencionar nuevos tipos de datos (cadenas de caracteres), la posibilidad de leer variables en tiempo de ejecución, varias primitivas (números aleatorios, conversiones entre tipos de datos, etc.), nuevos mecanismos sintácticos para delimitar los bloques de código y una mejora en los mensajes de error,

En los cursos iniciales se trabaja con un único robot y en el introductorio a la programación concurrente con más de uno.

Experiencias con DVC

Se describen en esta sección las experiencias realizadas en los últimos años con DVC, en las dos asignaturas en las que ha sido empleado. Cada caso está precedido de una breve descripción de la materia, para posteriormente describir los modos de uso de la herramienta y las conclusiones a las que se ha arribado luego de aplicarlas. Estas conclusiones están basadas en la observación sistemática del comportamiento de los alumnos. Corresponde mencionar que se trata siempre de números reducidos de alumnos (no más de cincuenta alumnos en Expresión de Problemas y Algoritmos y no más de diez alumnos en Introducción a la Concurrencia).

Es importante destacar también que ambas asignaturas han surgido durante la última modificación del Plan de Estudios de la carrera (plan 2010), y en los dos casos buscaron generar nuevos espacios curriculares, que, incorporados en etapas tempranas de la carrera, posibilitaran la maduración posterior de los conocimientos.

Expresión de Problemas y Algoritmos (EPA)

Es una asignatura que corresponde al primer cuatrimestre del primer año de las carreras de Licenciatura en Sistemas y Analista Universitario de Sistemas, con una carga horaria de 6 horas semanales durante quince semanas (90 horas totales). Antes de su inclusión en el Plan de Estudios los temas estaban incorporados a la asignatura Algorítmica y Programación, que se dictaba recién en el segundo cuatrimestre de primer año. Estos temas son los siguientes:

- Análisis y resolución de problemas.
- Especificación simbólica.
- Expresión de soluciones en un lenguaje algorítmico.

El DVC se utiliza en la cátedra durante las últimas trece semanas. En el segundo cuatrimestre, en la asignatura Algorítmica y Programación I, los alumnos reemplazan este lenguaje por el lenguaje Pascal.

Es para los alumnos la primera actividad de resolución de problemas computacionales.

La experiencia obtenida en el uso de la herramienta durante los ciclos académicos 2013 y 2014, ha dejado en evidencia la utilidad de la misma al momento de incorporar los diferentes conceptos vinculados a la

resolución de algoritmos con una computadora, más concretamente con la programación estructurada.
Entre las ventajas más importantes podemos mencionar:

a. Incremento en el porcentaje de alumnos que aprobaron la asignatura. La Tabla 1 compara el porcentaje promedio de aprobados y aprobados por promoción durante los tres primeros años de dictado de EPA (2010 a 2012), con los porcentajes de los dos últimos años en los que se utilizó DVC.

Ciclos	Porcentaje de Alumnos Aprobados	
	Total	Promoción
Promedio anteriores	47%	27%
2013	61%	56%
2014	56%	54%

Tabla 1

b. Autonomía de los alumnos. Se logró un trabajo más autónomo por parte de los alumnos, con menor dependencia del docente. Esta autonomía es atribuible a las mayor claridad y especificidad de los mensajes de error, tanto en tiempo de compilación como de ejecución, que permiten que los alumnos puedan resolver muchos de los errores cometidos sin consultar al docente. Esto ha sido muy importante en el presente ciclo, pues se ha mantenido la estructura de cátedra (un auxiliar en las clases prácticas), pero un ingreso muy importante ha provocado que en el aula haya casi cincuenta alumnos (entre ingresantes nuevos y recursantes).

c. Ampliación del espectro de problemas. Como consecuencia de la incorporación de nuevos tipos de datos y la posibilidad de lectura de variables en tiempo de ejecución se amplió el espectro de problemas a resolver en las guías de trabajos prácticos. Esto permite proponer a los alumnos problemas más variados y de mayor complejidad, posibilitando utilizar DVC hasta el final de la asignatura. En ciclos anteriores debía cambiarse a Pascal en la última parte del cuatrimestre.

Por otra parte, confirmando nuestra opinión sobre el uso cuidadoso que debe hacerse de este tipo de herramientas, hemos detectado que en alguna medida las dificultades causadas por una limitada capacidad de abstracción sólo se han desplazado en el tiempo, y aparecen nuevamente cuando el alumno comienza a resolver problemas más ligados al ámbito profesional, con un lenguaje de programación que posee pocas o ninguna facilidad para visualizar la ejecución. Esto se nota tanto sobre el final de EPA (cuando en función de las nuevas características del lenguaje se plantean problemas que no implican movimientos del robot en la ciudad y que en consecuencia no pueden visualizarse), como en la asignatura siguiente (Algorítmica y Programación I), en la que se utiliza desde el comienzo el lenguaje Pascal.
Otra situación que hemos advertido es que, al comienzo de Algorítmica y Programación I, cuando se realiza un repaso de los conceptos que han adquirido en el cuatrimestre anterior, y pese a no introducirse temas conceptuales nuevos, la mayoría de los alumnos debe realizar un esfuerzo importante para aplicar correctamente dichos conceptos en problemas que no permiten la visualización de la ejecución.
Esta situación nos llevó a realizar, a partir de este ciclo lectivo, algunos cambios metodológicos en el uso del DVC en la asignatura EPA. Estos cambios buscan mejorar la capacidad de abstracción en los alumnos,

proponiendo una gradualidad entre los problemas que pueden resolverse visualizando su ejecución y aquellos en los que la visualización de la ejecución no es posible.
Las estrategias que se están aplicando durante la presente cursada son:

a. Incrementar sustancialmente los ejercicios de los siguientes tipos (que deben resolver sin recurrir a la computadora):
    i. ¿Qué hace este programa cuando utiliza los siguientes datos?
    ii. Determine errores de ejecución en el siguiente programa cuando se lo utiliza con los siguientes datos.
    iii. ¿Qué conjunto de datos provocará que falle la ejecución de este programa?
    iv. ¿Cuál de los siguientes programas resuelve el problema X? De existir más de uno que lo haga determine cuál es el más eficiente, justificando su respuesta.

    Obviamente se informará a los alumnos la importancia que se le asigna a resolver estos ejercicios sin hacer uso de la computadora, aún cuando las características de los problemas planteados permitan su visualización mediante DVC.
    Se espera que este tipo de actividades incremente la capacidad de abstracción en los alumnos, y disminuya el uso del procedimiento "prueba y error", que suelen aplicar para obtener la solución a los problemas visualizables.
    Se les recordará que este tipo de ejercitación, que se incrementará en esta cursada pero que se emplea desde hace varios años, también se incluye en los exámenes parciales, y que la experiencia demuestra que son los problemas que más inciden en la desaprobación.

b. Organizar un torneo, a partir del segundo mes de clases, en el que competirán equipos compuestos por dos o tres alumnos. A partir de un problema propuesto por "el robot" cada grupo deberá resolverlo durante la clase práctica, sin hacer uso de la computadora, ni para compilarlo ni para ejecutarlo. Finalizada la clase deberá entregarlo vía la plataforma que se utiliza para soportar la actividad educativa. Los equipos que hayan entregado programas que compilen y ejecuten correctamente obtendrán dos puntos. Durante la semana siguiente a la que se planteó el ejercicio el código correcto estará disponible en la plataforma y podrá ser visto por todos los alumnos, hayan participado o no de la práctica. El o los grupos ganadores del torneo obtendrán medallas virtuales (oro, plata y bronce), que se utilizará como nota de concepto para mejorar sus notas finales en la cursada de la asignatura.

Introducción a la Concurrencia

Es una asignatura del primer cuatrimestre del tercer año de las carreras Licenciatura en Sistemas y Analista Universitario de Sistemas. con una carga horaria de 4 horas semanales (60 horas totales). Antes de su inclusión en el plan de estudios los temas estaban incorporados a la asignatura Sistemas Operativos, que se continúa dictando en el segundo cuatrimestre de tercer año. Estos temas son:

- Conceptos de concurrencia.
- Especificación de la ejecución concurrente.
- Comunicación y sincronización.
- Concurrencia con variables compartidas.
- Concurrencia con pasaje de mensajes.
- Sistemas multiprocesador para concurrencia real.
- Lenguajes de programación concurrente.

- Diseño y programación de algoritmos concurrentes.

El DVC se utiliza aproximadamente durante las primeras 6 semanas de dictado de la materia.
La experiencia obtenida en el uso de la herramienta durante los años 2013 y 2014 ha permitido determinar algunas ventajas y desventajas acerca de su uso.
Entre las ventajas más importantes podemos mencionar:

a. Posibilidad de instalación en múltiples plataformas. Esta facilidad resulta de importancia ya que, a diferencia de lo que ocurre en los primeros años, la mayoría de los alumnos utilizan sus propios equipos y en ellos tienen instalados una diversidad de sistemas operativos (Windows, distintas distribuciones de Linux, Mac OS X)
b. Tiempo mínimo de aprendizaje del lenguaje. Si bien algunos alumnos ya han utilizado VDV o DVC, otros no lo han hecho nunca, y aún en esos casos logran aprender a utilizarlo en muy pocas horas.
c. Importancia de la visualización de la ejecución. Para la comprensión del problema de la concurrencia resulta de gran relevancia, especialmente en los inicios, poder comprobar visualmente la ejecución de un algoritmo. El hecho que dos o más robots "se pasen por encima" en una misma esquina, cuando dicha superposición resulta incompatible con las características del problema que se está resolviendo, tiene un impacto fuerte en la comprensión del problema por parte del alumno y favorece la búsqueda de mecanismos de sincronización que impidan que esto ocurra. Este impacto no se logra con lenguajes que carecen de un seguimiento visual.
   No obstante, teniendo en cuenta que en algunos casos esta "superposición" es posible y deseable, el sistema puede configurarse para que dicha situación produzca un error (y finalice la ejecución) o se ignore (permitiendo continuar la ejecución).
d. Posibilidad de repetir una traza de ejecución. Uno de los problemas nuevos más importantes que enfrenta el alumno es el de encontrar defectos en las soluciones concurrentes. Esta dificultad es consecuencia del "no determinismo" que caracteriza a las mismas. Dicha condición es provocada por el tipo de planificador de corto plazo que emplea el sistema operativo y por la carga del sistema (conjunto de procesos que compiten por el uso de recursos) al momento de ejecutar el programa concurrente.
   El DVC contempla la posibilidad de elegir distintos algoritmos de planificación de corto plazo: aleatorio, round-robin (con posibilidad de modificar el quantum) y repetitivo.
   El planificador repetitivo es el que resulta más útil a la hora de enseñar y aprender programación concurrente, pues permite repetir una traza (una particular intercalación de las instrucciones de los distintos procesos secuenciales que conforman la solución concurrente), obtenida como resultado de una ejecución previa o forzada por el instructor (o el alumno). En particular la segunda posibilidad permite demostrar que una solución concurrente es defectuosa, forzando la ejecución de la traza que provoca la situación conflictiva. Esta instancia podría no producirse, aún ejecutando el programa concurrente un número elevado de veces.

Por su parte, entre las principales desventajas cabe señalar:

a. Un único mecanismo de sincronización. La versión actual de DVC sólo incorpora semáforos como mecanismo de sincronización. Esto

obliga a abandonarlo rápidamente cuando deben introducirse otros mecanismos (básicamente monitores y mensajes). Este cambio implica un quiebre en la dinámica de la práctica que resulta perjudicial pues al menos una de las clases prácticas (2 horas) se pierde en la transición destinada a introducir un nuevo lenguaje.

b. Limitado manejo del "no determinismo". Si bien, como ha sido mencionado entre las ventajas, el DVC permite repetir una traza de ejecución o forzar una que lleve a una situación conflictiva, es esa posibilidad la única para tratar con el "no determinismo" de las soluciones concurrentes. Además, por el momento, la verificación de que la traza forzada resulte compatible con los procesos concurrentes definidos (es decir, pueda efectivamente producirse en una ejecución particular de los mismos bajo condiciones de planificación y carga del sistema determinadas), es responsabilidad exclusiva de quien la establece.
   Por otra parte, aunque el no determinismo depende del tipo de planificador y de la carga del sistema, sólo puede actuarse sobre las primeras de las variables eligiendo un número reducido de alternativas.

Trabajos futuros

Los trabajos futuros incluyen tanto mejoras a incorporar al DVC como continuar experimentando nuevas formas de uso. Algunas de estas últimas ya se están aplicando durante el actual ciclo académico en Expresión de Problemas y Algoritmos. Otras terminarán de definirse y comenzarán a utilizarse a partir del año próximo.

Mejoras del entorno.

a. Depuración de programas: si bien se ha dotado al entorno con herramientas que facilitan la depuración de los programas, tales como ejecución paso a paso y la posibilidad de observar el valor de las variables, la experiencia muestra que sería conveniente incorporar nuevas facilidades. Entre las previstas está la posibilidad de observar los valores la cantidad de objetos que aún quedan en las intersecciones.
b. Primitivas para concurrencia: el lenguaje ofrece como único mecanismo de sincronización los semáforos binarios y generales. Es imprescindible agregar, al menos, monitores y mensajes (sincrónicos y asincrónicos).
c. Dispositivos móviles: una de las primeras preguntas que realizan los alumnos es si pueden instalar la aplicación en sus dispositivos móviles. En nuestro caso particular la mayoría de los alumnos cuentan con un smartphone, y pasan mucho tiempo realizando diversas actividades con él. Está en estudio desarrollar una aplicación, que en principio podría contar con una funcionalidad reducida, que puedan ejecutar en su smartphone y les permita programar, visualizar la ejecución y compartir sus soluciones.
d. Manipulación de la traza en la concurrencia. En una primera etapa se espera poder contar con la funcionalidad que permita verificar la validez de una traza que es introducida por el usuario para forzar su ejecución utilizando el planificador repetitivo.

Nuevas formas de uso.

De la experiencia resumida en el presente trabajo se manifiesta una nueva forma de utilización de la herramienta que pretende recortar el tiempo que el alumno necesita para potenciar su capacidad de abstracción. Se incorporará una práctica, que el alumno deberá resolver con el lenguaje, en la cual no hará falta la interacción con el robot, la ciudad u objetos en ella.
La práctica estará basada en ejercicios que habitualmente se resuelven en la primera parte

de la materia Algorítmica y Programación I y pretenderá actuar como un atenuante en el cambio de contexto entre ambas materias.

Referencias

1. Hassinen Marko and Mäyrä Hannu. 2006. Learning programming by programming: a case study. In Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006 (Baltic Sea '06). ACM, New York, NY, USA, 117-119. DOI=10.1145/1315803.1315824 http://doi.acm.org/10.1145/1315803.1315824
2. Moroni Norma y Señas Perla. 2002. La visualización de algoritmos como recurso para la enseñanza de la programación. En Proceedings del IV Workshop de Investigadores en Ciencias de la Computación. Tandil (BA). Disponible en http://hdl.handle.net/10915/21882.
3. Salgado Castillo Antonio y col. 2013. Didáctica de la resolución de problemas de programación Computacional. En Pedagogía Universitaria, Vol. XVIII N° 4
4. Depetris Beatriz y col. 2013. Experiencias con Da Vinci Concurrente en la enseñanza inicial de la programación y la programación concurrente. En Proceedings del VIII Congreso de Tecnología en Educación y Educación en Tecnología. Santiago del Estero. Disponible en http://hdl.handle.net/10915/27581.
5. Dann Wanda and Cooper Stephen. 2009. Education: Alice 3: concrete to abstract. Communications of ACM 52, 8, 27–29, (August 2009)
6. De Giusti Armando y Frati Fernando. 2010. ¿Concurrencia y Paralelismo en el primer curso de Algorítmica? En Proceedings del V Congreso de Tecnología en Educación y Educación en Tecnología.
7. Champredonde Raúl y De Giusti Armando. 1997. Design and Implementation of Visual Da Vinci. In: III Congreso Argentino en Ciencias de la Computación (CACiC '97), La Plata (BA)