The student as course content designer

Sylvia da Rosa Zipitría

Instituto de Computación, Facultad de Ingeniería, Universidad de la República
darosa@fing.edu.uy

Abstract. This article describes the process of applying a proposal of students of the course Didactics of Algorithms and Data Structures (DAED). DAED is a course of the degree in Computer Engineering. It is based on the epistemology of Jean Piaget as a theoretical framework and on the theory of didactical situations of Guy Brousseau as a didactic model. In the course, students learn the theoretical concepts through practical learning experiences. The final coursework is to redesign learning activities for specific content of their choice. The article highlights a student proposal of learning activities designed to teach about Dynamic Memory and Pointers, and the importance of this learning experience of the DAED students.

Keywords: didactics, programming, epistemology

El rol del estudiante como diseñador de contenidos

Sylvia da Rosa Zipitría

Instituto de Computación, Facultad de Ingeniería, Universidad de la República
darosa@fing.edu.uy

Abstract. Este artículo describe el proceso de aplicación de una propuesta de estudiantes del curso Didáctica de Algoritmos y Estructuras de Datos (DAED). DAED es un curso de la carrera de Ingenieríen en Computación que se basa en la epistemología de Jean Piaget como marco teórico y en la teoría de situaciones didácticas de Guy Brousseau como modelo didáctico. De acuerdo a ello, los conceptos teóricos se introducen mediante experiencias prácticas. El trabajo final consiste en rediseñar actividades de aprendizaje para contenidos específicos que los participantes eligen. El artículo destaca una propuesta estudiantil para enseñar el tema Memoria Dinámica y Punteros, y la importancia de esta experiencia de aprendizaje para los estudiantes DAED.

Keywords: didáctica, programación, epistemología

1 Introducción

En la edición de 2017 del curso "Didáctica de Algoritmos y Estructuras de Datos" (DAED) uno de los grupos formado por estudiantes de la carrera de Ingeniería en Computación, eligió como tema de su trabajo final "Memoria dinámica. Tipo puntero" que es parte del curso introductorio de programación (Programación1) de su carrera. En su trabajo, los autores critican la forma tradicional en que se introduce el tema y plantean un enfoque diferente, que fundamentan en lo que estudiaron en el curso DAED, especialmente la teoría de Guy Brousseau[1, 14]. Como consecuencia, el material del curso Programación1 sobre este tema ha sido revisado y complementado con ejercicios inspirados en esta propuesta.

Esta experiencia nos enseñó que los estudiantes de las distintas carreras de computación que se imparten en las universidades, pueden convertirse en excelentes generadores de contenidos. Ellos conocen los temas y sobre todo tienen frescas las dificultades que entraña su estudio por primera vez. Al proveerles de formación didáctica como la que se brinda en el curso DAED, pueden aplicar los conocimientos teóricos para reflexionar sobre su experiencia y lo que significa construir conocimiento sobre un tema. Al brindar a estudiantes en computación un curso como DAED, obtienen formación básica en el área de didáctica de la informática en la cual pueden profundizar y desarrollarse profesionalmente.

En las siguientes secciones se describe brevemente la teoría de Guy Brousseau, y luego se presenta el trabajo realizado para el curso DAED mostrando el material del curso tradicional Programación1 que los autores critican y los ejercicios

complementarios inspirados en su propuesta. A continuación se incluye el análisis de respuestas de estudiantes de Programación1 a los ejercicios complementarios. Dicho análisis provee elementos interesantes, que confirman lo acertado del enfoque de los estudiantes de DAED. Finalmente se presentan algunas conclusiones y las referencias a los artículos citados.

2 La teoría de las situaciones de Guy Brousseau

En la teoría epistemológica de Jean Piaget el conocimiento es una construcción individual profunda que se realiza poniendo en juego los mecanismos cognitivos para interactuar con el ambiente, ya sea resolviendo un problema o realizando una tarea[3–5]. Una de las aplicaciones didácticas más notables es la teoría de las situaciones de Guy Brousseau, que plantea que el rol del docente es por un lado, presentar al estudiante el objeto de estudio de modo que la interacción posibilite la construcción de conocimiento y por otro, ayudarlo a pasar de ese conocimiento *instrumental* a conceptos formales[1].

¿Cómo presentar el objeto de estudio? Es necesario partir de problemas, actividades o tareas que disparen una interacción del estudiante con el tema, lo relacione con sus conocimientos previos, quiera saber más, busque información, reflexione y se haga preguntas ... Esa interacción tiene el objetivo de ayudar al alumno a construir un conocimiento personalizado y contextualizado, es decir, el alumno construirá un conocimiento sobre el tema poniendo en juego sus instrumentos cognitivos e integrándolo a sus propias estructuras cognitivas. Podemos decir que es así como comienzan a investigar los investigadores de cualquier área o disciplina, para luego, a la hora de comunicar sus resultados, transformar el conocimiento que han construido en conocimiento formalizado y culturizado, es decir, comunicable. Así lo reciben los académicos y los profesores. ¿Qué debe hacer un profesor? Debe hacer de alguna manera una transformación inversa, es decir, utilizar el conocimiento formalizado y culturizado para elaborar una presentación para el estudiante que le permita indagar, reflexionar, criticar, enfrentarse a obstáculos, generar dudas, sacar conclusiones, es decir construir un conocimiento personalizado[14]. Esto es bien diferente del modo tradicional de "dar clase" donde lo que en general hace el docente es "exponer" los temas o "dar el teórico" y eventualmente plantear preguntas o poner ejercicios, refiriendo a los conceptos en su forma acabada y formalizada. Seguir las pautas didácticas de Guy Brousseau requiere mucho trabajo previo del docente: preparar las preguntas, o actividades, formular los problemas de modo que el alumno por sí sólo o en discusión con otros (pero sin el docente), pueda abordar el tema y avanzar en conocerlo. Introducir prácticas didácticas basadas en ello no es fácil, porque contradice lo que los docentes están acostumbrados a hacer, muchos durante décadas. Hay que considerar además que también los estudiantes están acostumbrado a la metodología tradicional de escuchar las explicaciones de un docente. Romper una práctica tan arraigada requiere toma de conciencia de que no es efectiva, voluntad para cambiarla y conocimiento de alternativas.

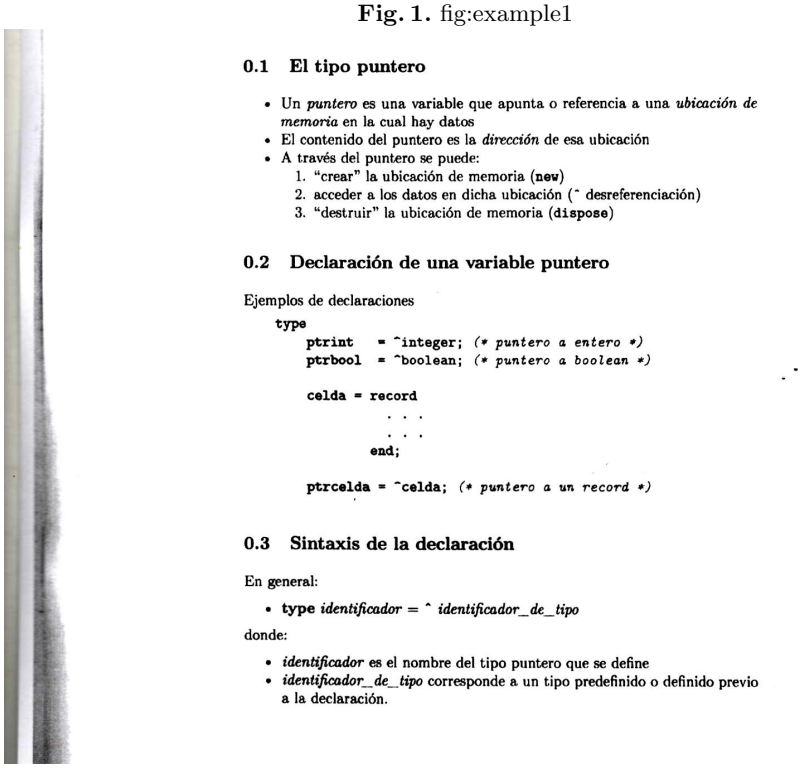
Si bien Brousseau trabaja en matemática, su teoría de las situaciones puede aplicarse a cualquier disciplina. Las situaciones que plantea son de dos tipos: a-didáctica y didáctica. La primera corresponde a lo que hemos descrito arriba: el estudiante trabaja sobre un tema, con material provisto por el profesor, pero sin intervención de éste. En la situación didáctica, el profesor establece el vínculo entre lo que se ha elaborado en la situación a-didáctica y los conceptos formales. El análisis de dicha elaboración le permite al profesor tener en cuenta las dificultades, los puntos débiles, los errores que aparezcan en el trabajo realizado. Por medio de nuevas preguntas el docente puede ayudar a corregirlos y superarlas y guiar al estudiante en la construcción de conocimiento conceptual y formal, a partir del conocimiento personalizado que construyó en la situación a-didáctica. La diferencia entre conocimiento conceptual y formal es que el concepto puede ser construido y expresado en lenguaje natural, por ejemplo, mientras que la formalización implica expresar el concepto en algún formalismo, es decir, transformarlo en conocimiento culturizado, que puede ser transmitido a otros.

3 El ejemplo de los estudiantes del curso DAED

La teoría de la situaciones de Brousseau es uno de los temas del curso DAED, que se introduce a través de trabajos en didáctica de la informática[10, 11]. El grupo de estudiantes que realizó su trabajo final basado en Brousseau tomó el tema "Memoria dinámica. Tipo puntero" del curso introductorio Programación1 de la carrera de Ingeniería en Computación, que ellos cursaron al inicio de sus estudios. El curso Programación1 es el primer curso de programación para la mayoría de los alumnos que ingresan a Ingeniería en Computación. El lenguaje que se utiliza es Pascal, creado en la década de los 70 para el aprendizaje de programación básica y que en nuestra opinión sigue siendo uno de los mejores lenguaje para ese objetivo (usamos una versión reciente: FreePascal). Tanto el Programa del curso como las slides a las que pertenecen las figuras 1 y 2 y los ejercicios complementarios de las figuras 3 y 4 están accesibles en https://www.fing.edu.uy/~darosa/#links.

La principal motivación de los estudiantes de DAED provino de las enormes dificultades que ellos tuvieron para la comprensión de este tema cuando se enfrentaron con él, a causa de la forma *indebida* en que tradicionalmente se presenta en los cursos introductorios de programación. Las figuras 1 y 2 muestran cómo el concepto de puntero se introduce de manera formal y cómo se usa para introducir el concepto de listas, siempre mediante definiciones formales. La experiencia de los estudiantes de DAED es que el tipo puntero y la estructura de listas presentados como se muestra en las figuras 1 y 2, no tienen sentido para quien está aprendiendo. Por ejemplo, la pregunta sobre la lista vacía y el último elemento de la lista en la figura 2 punto 1.2 tiene como objetivo introducir el concepto del puntero nulo en el punto 1.3, sin tener en cuenta que a priori los alumnos no tienen por qué pensar que ambas cosas están relacionadas y hasta pueden resultarles contradictorias (lista vacía y último elemento de una lista).

Fig. 1. fig:example1



0.1 El tipo puntero

- Un *puntero* es una variable que apunta o referencia a una *ubicación de memoria* en la cual hay datos
- El contenido del puntero es la *dirección* de esa ubicación
- A través del puntero se puede:
  1. "crear" la ubicación de memoria (**new**)
  2. acceder a los datos en dicha ubicación (^ desreferenciación)
  3. "destruir" la ubicación de memoria (**dispose**)

0.2 Declaración de una variable puntero

Ejemplos de declaraciones

```
type
    ptrint   = ^integer; (* puntero a entero *)
    ptrbool  = ^boolean; (* puntero a boolean *)

    celda = record
              . . .
              . . .
            end;

    ptrcelda = ^celda; (* puntero a un record *)
```

0.3 Sintaxis de la declaración

En general:

- **type** *identificador* = ^ *identificador_de_tipo*

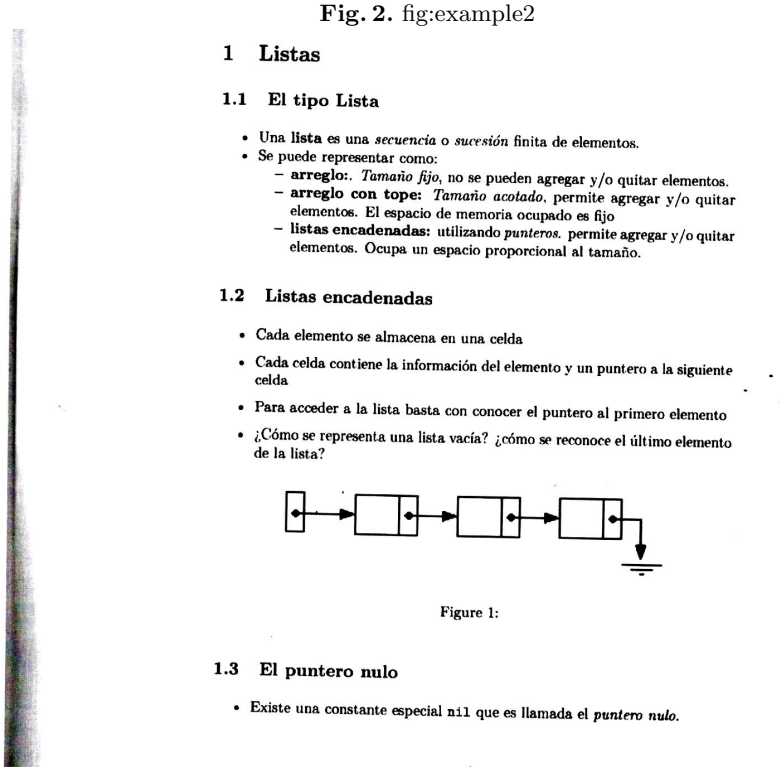
donde:

- *identificador* es el nombre del tipo puntero que se define
- *identificador_de_tipo* corresponde a un tipo predefinido o definido previo a la declaración.

Por otro lado los estudiantes del curso DAED observaron que una vez que se comienza a *operar* (insertar o eliminar elementos) con la estructura de lista enlazada (varias clases después), el tema cobra sentido y se abre la posibilidad de comprensión de listas y punteros. Esta observación es generada por los fundamentos teóricos de la teoría epistemológica de Jean Piaget, que son estudiados en el curso DAED. En efecto, a partir de una amplia y profunda base empírica, Piaget explica cómo y por qué la coordinación de las acciones (operaciones) de un individuo en interacción con un medio y la comprensión de los cambios que se imponen a ese medio, son la fuente de conocimiento, especialmente científico [5].

El trabajo final para el curso DAED consiste básicamente en responder a la pregunta ¿cómo introduciría tal tema usando los conceptos y principios visto en el curso? Estos estudiantes proponen una manera alternativa de introducir memoria dinámica y punteros, focalizándose en que el concepto de listas se tra-

Fig. 2. fig:example2



1 Listas

1.1 El tipo Lista

- Una **lista** es una *secuencia* o *sucesión* finita de elementos.
- Se puede representar como:
  - **arreglo:**. *Tamaño fijo*, no se pueden agregar y/o quitar elementos.
  - **arreglo con tope:** *Tamaño acotado*, permite agregar y/o quitar elementos. El espacio de memoria ocupado es fijo
  - **listas encadenadas:** utilizando *punteros*. permite agregar y/o quitar elementos. Ocupa un espacio proporcional al tamaño.

1.2 Listas encadenadas

- Cada elemento se almacena en una celda
- Cada celda contiene la información del elemento y un puntero a la siguiente celda
- Para acceder a la lista basta con conocer el puntero al primero elemento
- ¿Cómo se representa una lista vacía? ¿cómo se reconoce el último elemento de la lista?

Figure 1:

1.3 El puntero nulo

- Existe una constante especial `nil` que es llamada el *puntero nulo*.

baje *antes y sea usado para introducir* el concepto de puntero. Su fundamentación se basa en el hecho de que, en general, todas las personas tienen la noción de 'lista encadenada', a través de diferentes objetos que conocen, por ejemplo, una cadena de eslabones, los vagones de un tren, un collar, etc. Por otro lado, los alumnos de Programación1, conocen las estructuras de array y array con tope, que ya se han visto en el curso. Basados en la teoría de las situaciones y en los fundamentos epistemológicos que estudiaron en el curso DAED, el punto central de los autores de esta propuesta, es vincular 'lo que el estudiante ya sabe' con los conceptos formales. Para ello, proponen comenzar por actividades y preguntas donde el estudiante por un lado, opere con una lista concreta, insertando y/o eliminando elementos, y por otro, reflexione sobre cómo lo ha logrado y qué cambios produjo en la lista dada.

Los docentes de Programación1 que somos a su vez docentes de DAED, consideramos que esta propuesta aportaba una perspectiva útil y novedosa para el

tema en cuestión, y sobretodo lo hacía con fundamentos teóricos sólidos. Para la siguiente edición del curso Programación1 elaboramos ejercicios complementarios para el tema "Memoria Dinámica. Tipo puntero", inspirados en el trabajo de los estudiantes DAED (algunos fueron incluidas sin cambios), que se muestran en las figuras 3, 4 y 6.

Fig. 3.

Ejercicio 1

Dadas las siguientes declaraciones para un arreglo de enteros:

```
CONST TAM = 5;
TYPE Arreglo = ARRAY [1..TAM] OF Integer;
```

Suponga que se tiene el siguiente arreglo de enteros, del tipo `Arreglo` declarado. ¿Sería posible agregarle un nuevo elemento? Explique su respuesta.

1	2	3	4	5
8	2	9	6	3

Ejercicio 2

Dadas las siguientes declaraciones para un arreglo con tope de enteros:

```
CONST MAX = 8;
TYPE ArregloTope = RECORD
                     elems : ARRAY [1..MAX] OF Integer;
                       tope : 0..MAX;
                   END;
```

Suponga que se tiene el siguiente arreglo con tope de enteros, del tipo `ArregloTope` declarado. ¿Sería posible agregarle un nuevo elemento? Explique su respuesta.

1	2	3	4	5	6	7	8
8	2	9	6	3			

(tope = 5)

Ejercicio 3

Considere el tipo `ArregloTope` del segundo ejercicio. Suponga que se tiene el siguiente arreglo con tope **ordenado** y se desea insertar el valor 7, **manteniendo** el orden de los elementos. ¿Sería posible hacerlo? Explique su respuesta. En caso afirmativo, utilice goma y lápiz para ilustrar cómo quedaría el arreglo con tope.

1	2	3	4	5	6	7	8
4	5	6	8	10			

(tope = 5)

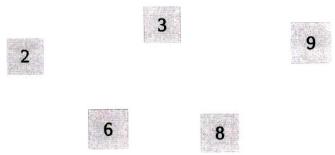
Fig. 4. fig:example9

Ejercicio 4

Si en el ejercicio anterior el elemento que se desea insertar es el 3, ¿cuántos elementos deben ser re ubicados?

Ejercicio 5

Dada la siguiente figura:



a) Utilice flechas para conectar las cajas entre sí, de modo tal que, siguiendo las flechas, los elementos queden en orden ascendente.

b) Agregue elementos con los valores 1 y 5, de modo que el orden se mantenga.

c) Agregue un nuevo elemento de valor 10, de modo que el orden se mantenga.

d) Elimine el elemento con el valor 1, marcando cómo quedan las flechas al hacerlo. Haga lo mismo con el elemento 5 y el 10.

e) ¿Cuál es la flecha que representa toda la estructura?

f) ¿Qué elementos componen a cada uno de los eslabones de la estructura?

Los ejercicios se reparten en las clases de Programación1 y los alumnos deben trabajar en ellos individualmente o en grupo, (en una situación a-didáctica). Las clases en las que se trabaja con los ejercicios complementarios poseen una dinámica opuesta a la tradicional: no hay exposición del docente, que se limita a responder eventuales dudas, y son los estudiantes los que desarrollan actividades. Al finalizar, los alumnos entregan lo que han elaborado para las distintas preguntas, y ese material es utilizado posteriormente para analizar el proceso de construcción de conocimiento sobre el tema. Ejemplos del análisis se presentan en la siguiente sección.

4 Análisis de algunas respuestas

Las respuestas de los estudiantes proveen información valiosa para elaborar la introducción formal de los conceptos y nuevos ejercicios de implementación en Pascal, que se trabajan de la misma manera. El rendimiento (si responden correctamente o no) no es un punto prioritario ya que el objetivo es conocer cómo resuelven los problemas planteados; qué errores cometen; cuáles son las dificultades, como insumos para las instancias que vendrán luego (situaciones didácticas), en las que se construye el vínculo con los conceptos formales. Por eso, se pide que entreguen las respuestas a los ejercicios por escrito, y se aclara que las mismas no serán evaluadas de modo que sumen o resten puntos para la aprobación del curso. Eso los libera de la presión que significa pensar "¿cuál será la respuesta que quiere el profesor?", que sumado a la discusión con el grupo de compañeros, contribuye a crear un ambiente de trabajo altamente productivo. La respuesta de los estudiantes de Programación1 desde el punto de vista de la motivación es muy positiva; en general se quedan en clase a pesar de que no se dicta ningún contenido, lo que no suele ocurrir cuando se les propone trabajar en los prácticos del curso cuyos ejercicios son tomados de textos tradicionales.

El factor decisivo sin embargo, es que los ejercicios se elaboran con el foco puesto en los conocimientos que el estudiante ya tiene sobre el tema de modo que pueda poner en funcionamiento sus estructuras cognitivas y construir nuevos conocimientos a través de la reflexión acerca de los problemas planteados.

En los primeros ejercicios se introduce la noción de lista dinámica en relación con la estructura de arreglo que se ha visto anteriormente. Las preguntas del ejercico 5 en la figura 4, utilizan una lista concreta para que el alumno opere sobre la misma, agregando o eliminando elementos, considerando que cada uno de ellos consta de un valor y una flecha. Por ejemplo, al tener que agregar un elemento a una lista como se pide en la pregunta b del ejercicio 5, deberá crear 'una caja' y colocar flechas para insertarla. Estará creando un elemento del tipo puntero, y aunque aún no lo sepa en esos términos, lo sabe de manera *instrumental y conceptual*. Como bien expresan los estudiantes de DAED en su trabajo: *un puntero por sí sólo no tiene sentido, sin embargo como eslabón de una cadena se comprende inmediatamente.*

En la figura 5 se muestra el trabajo de un grupo de estudiantes (todos los trabajos han sido similares). El análisis de las respuestas evidencia las dificultades que el concepto de listas entraña para ellos, y permite tomarlas en cuenta antes de la formalización. De ese modo se impulsa la construcción conceptual usando una representación cercana al estudiante, sin agregar la dificultad extra que significa aprender la sintaxis de un formalismo como un lenguaje de programación. En las figuras 7 y 8 se muestran ejemplos de implementación en Pascal del tipo lista y de la operación de insertar un elemento en una lista dada.

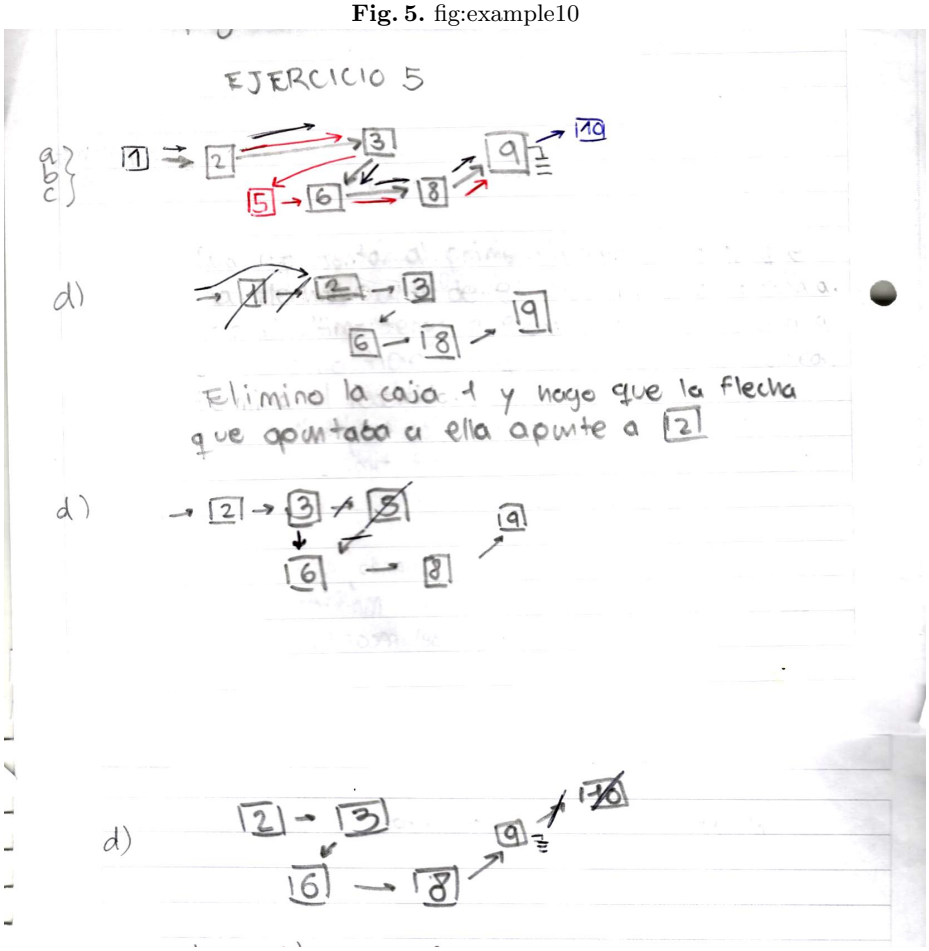
Las respuestas que se analizan (figura 5) corresponden a las preguntas del ejercicio 5 (figura 4). La primera tarea es enganchar los elementos a partir del 2 (a), lo cual el estudiante realiza correctamente, pero sin indicar ninguna flecha hacia el primer elemento (el 2). La flecha hacia el primer elemento aparece en la segunda tarea (b), cuando debe agregar el elemento 1 en el lugar correcto para

conservar el orden, es decir, antes del 2. Para ello, dibuja correctamente una flecha saliendo del elemento 1 y llegando al 2 (figura 5, primer dibujo). Omite poner una flecha llegando al elemento 1. Sin embargo cuando debe *eliminar* el elemento 1, aparece la flecha que lo indica como primer elemento (figura 5, segundo dibujo) y el estudiante escribe 'Elimino la caja 1 y hago que la flecha que apuntaba a ella apunte a 2'. En el tercer dibujo, donde se pide eliminar el 5, (lo cual realiza correctamente, tachando la caja que contiene al 5 y dibujando una flecha entre la caja que contiene al 3 y la que contiene al 6), la flecha apuntando al primer elemento *permanece*. Sin embargo en el último dibujo que se centra en el último elemento de la lista, vuelve a omitir la flecha al primer elemento. El conocimiento del estudiante de la lista como estructura no está consolidado y no puede prestar atención al primer elemento cuando está trabajando con el último[6]. Con respecto al puntero a NIL como siguiente puntero del último elemento de la lista, observar que en las operaciones de agregar y eliminar el elemento 10 (primer y último dibujo de la figura 5 respectivamente) el estudiante lo agrega al elemento 9 al ser o quedar éste como último, pero no lo indica como siguiente del último elemento que es el 10 (en ninguna de sus listas). [1]

Este proceso dialéctico entre lo que el alumno hace bien y los errores que comete, es normal en la construcción de conceptos. El ejercicio 5 (figura 4) tiene el objetivo de generar *la necesidad* de realizar una acción (operación) que debe sentir el estudiante al resolver el problema. Por ejemplo, las partes b, c y d del ejercicio 5 plantean la necesidad de que todos los elementos tengan una flecha de salida, incluso el último que la necesita si hay que agregar uno más, y la pregunta e pone de manifiesto que es necesaria una flecha para toda la estructura, que es la que llega al primer elemento. La pregunta f tiene el objetivo de inicitar a la reflexión sobre cómo se construyeron los elementos de las listas para preparar la introducción del tipo puntero y su implementación en Pascal (figuras 7 y 8). Esta idea, opuesta a la metodología tradicional de tratar el tema, constituye el aporte innovador de los estudiantes DAED.

[1] Tener en cuenta que los estudiantes usan la misma figura para varias tareas, de ahí que haya más de una flecha entre cajas en algunos dibujos.

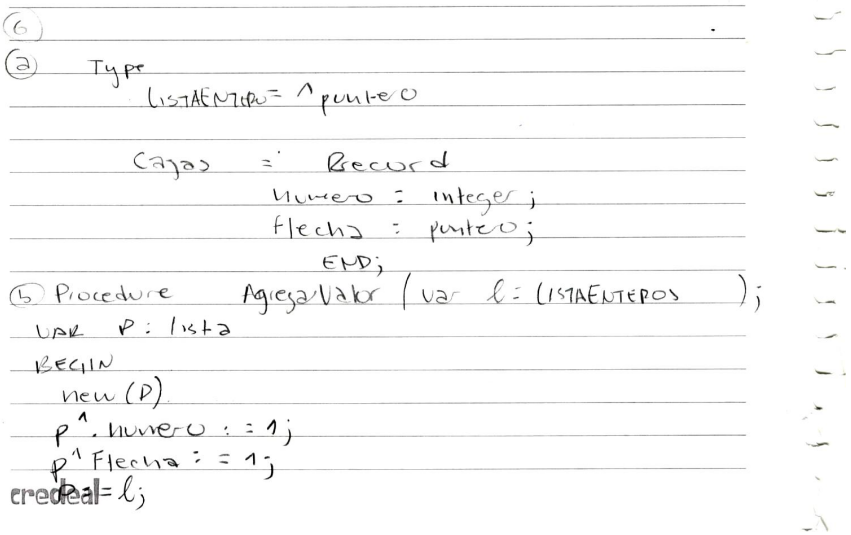
Fig. 5. fig:example10



4.1 Solución al problema general: el programa

Tal como se plantea en la teoría de Piaget existe una relación dialéctica entre la acción y el pensamiento formal [3, 4], entre la construcción de los conceptos y su formalización. Hay varios aspectos a tener en cuenta aquí: por un lado, la formalización significa utilizar un formalismo para describir el problema y su solución, por ejemplo un lenguaje de programación, que debe ser introducido por el docente, en una situación didáctica. Por otro lado, cabe preguntarse ¿cuál problema/solución expresamos en el formalismo? Observar por ejemplo en el

Fig. 6. fig:example12



ejercico 5 en figura 4 y sus soluciones en figura 5, que en la acción se han insertado los elementos 1, 5 y 10 en una lista dada, lo que significa conceptualizar el método empleado para solucionar tres problemas, (de hecho una de las características de una situación a-didáctica es que se actúa sobre elementos concretos).

Si bien es posible formalizar cada uno de esos problemas, como lo hacen los estudiantes al describir sus soluciones gráfica y textualmente (ver texto en punto d) figura 5), el pasaje a un nivel de mayor conocimiento se da al pasar al plano (mental) de la conceptualización, el problema y *una solución general* que abarca las instancias particulares. De hecho, desde una perspectiva didáctica, el conocimiento formal consiste en lograr expresar en el formalismo soluciones y problemas generales. El pasaje al caso general implica despegarse de los elementos concretos y transformar las acciones en operaciones que actúan sobre elementos cualesquiera, (en el plano conceptual (mental)). Los instrumentos cognitivos que hacen posible esa transformación son la abstracción reflexiva y la generalización constructiva, en un proceso descrito por Piaget y colaboradores en [3, 4]. En especial, en [6] Benjamin Matalon publicó un capítulo denominado *Recherches sur le nombre quelconque* (Investigaciones sobre el número cualquiera), en el cual analiza la relación entre el concepto de elemento genérico y el razonamiento por inducción, trabajando sobre la serie de los números naturales. Entre otras cosas, Matalon

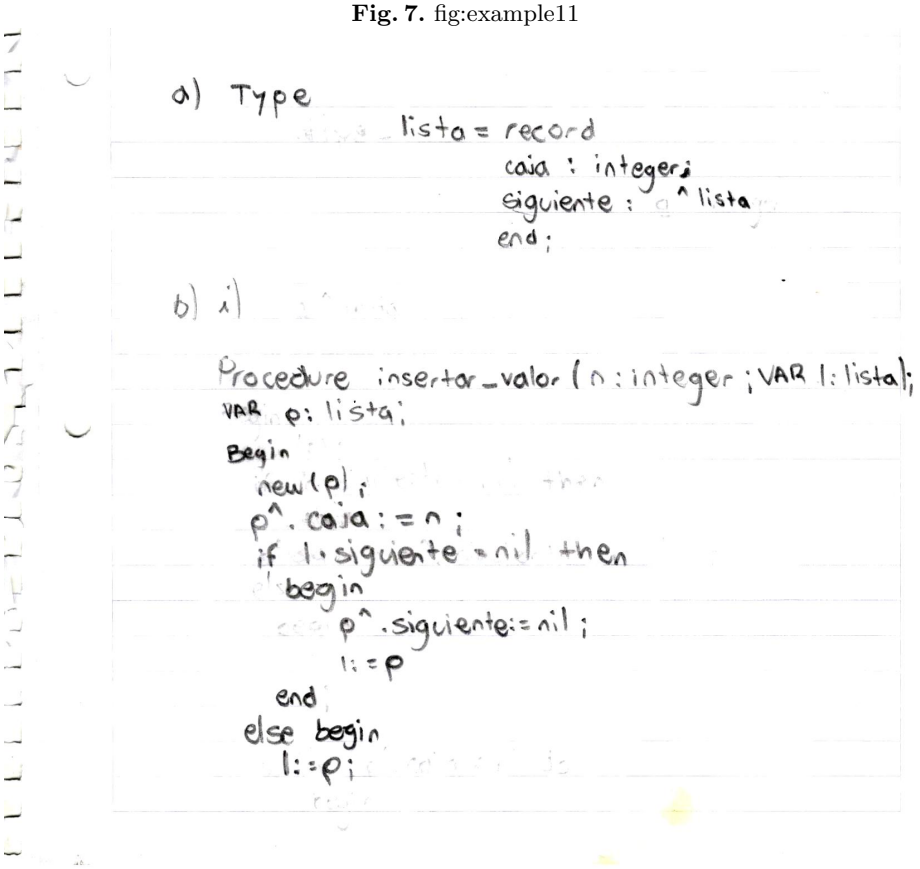
concluye que la construcción del concepto de elemento genérico se apoya en la construcción de la *acción genérica*, es decir la acción repetida para construir el elemento genérico, que en el caso de la serie de números naturales es el pasaje de n a $n + 1$. Nosotros interpretamos que cuando el formalismo es un lenguaje de programación, la acción genérica que menciona Matalon es la versión automatizada del algoritmo, es decir un programa que soluciona un problema para *cualquier caso* y que actúa sobre estructuras abstractas (cualesquiera).

La relación dialéctica entre conceptualización y formalización puede apreciarse en las implementaciones que los estudiantes realizan para responder los ejercicios en los que se pide escribir subprogramas para las operaciones ya trabajadas.. Observar por ejemplo en la figura 7, que el estudiante implementa un procedimiento para insertar el 1 en la lista (asigna 1 al valor y al puntero) y posiblemente se plantee otro para el 5 y otro para el 10, tal como operó para realizar las tareas en el dibujo de la figura 5. Esto revela que se encuentra apegado a las acciones con elementos concretos, mientras que el estudiante que realiza la implementación de la figura 8, considera un elemento cualquiera a insertar (le llama n) pero no itera sobre la lista dada para encontrar su lugar. Estas implementaciones incompletas permiten ir construyendo paso a paso el algoritmo general hasta obtener una primera versión que el estudiante considere correcta. Por ejemplo, ejecutando su implementación a mano con casos concretos, puede descubrir la necesidad de iterar sobre la lista para encontrar el lugar del elemento a insertar. La etapa de la ejecución del programa constituye una vuelta al plano de la acción, pero ahora no del sujeto sino ejecutada por la computadora. La interacción entre el programa (computadora) y el algoritmo (la persona humana) genera conocimiento sobre la solución al problema general y las estructuras de datos sobre las que se aplica. En esa interacción aparecen elementos propios de la ciencia de la computación en el proceso de construcción de conocimiento, que hemos descrito en [12].

Otro ejemplo de la influencia de la formalización en la acción es que la introducción de la lista vacía se puede hacer naturalmente a partir del procedimiento en Pascal para insertar un valor en una lista dada. La introducción de la lista vacía en un nivel instrumental es poco natural para los estudiantes, como hemos visto en estudios anteriores[7–9, 13]. Basándonos en resultados de esos estudios, sabemos que una vez que el alumno ha implementado su algoritmo sin considerar la lista vacía, la necesidad de hacerlo surge inmediatamente cuando se produce un error al ejecutar el programa o aplicar el algoritmo a mano. En este caso, ¿cómo representar la lista vacía con las cajas? Cuando el estudiante implementa su procedimiento en el algoritmo insertar_valor de la figura 8, se ve que no considera el caso en que la lista l pasada como parámetro, pueda ser vacía. Ello producirá un error en la instrucción 'if ...' y necesitará saber por qué se produce y cómo corregirlo. Esa necesidad permite introducir la lista vacía como elemento que soluciona un problema, análogamente a cómo se introdujo el 0 en la historia de los números[2].

[2] La palabra 'cero' proviene de la traducción de su nombre en sánscrito shunya (vacío) al árabe.

Fig. 7. fig:example11



```
a) Type
        lista = record
                  caja : integer;
                  siguiente : ^lista
                end;

b) i)

Procedure insertar_valor (n: integer; VAR l: lista);
VAR p: lista;
Begin
  new(p);
  p^.caja := n;
  if l.siguiente = nil then
    begin
      p^.siguiente := nil;
      l := p
    end
  else begin
    l := p;
```

5 Conclusiones

Los estudiantes del curso DAED elaboraron una propuesta para que la estructura de lista se utilice para introducir el tipo puntero (y no al revés como se hace en el material tradicional presentado en las figuras 1 y 2), aplicando principios teóricos del curso DAED por los que fundamentan que las *operaciones de agregar y/o eliminar* elementos de la lista resultan imprescindibles para comprender tanto la estructura como tal como los elementos de la misma de *tipo puntero*.

En primer lugar, cabe señalar que una propuesta basada en sólidos principios teóricos tanto epistemológicos como didácticos, cuenta con ellos para su legitimidad. Es decir, sabemos que esta manera de introducir listas y punteros es

válida y efectiva, porque tenemos teorías que la sustentan[2]. Este caso cuenta con el valor agregado de que han sido los estudiantes del curso DAED que la han diseñado, aportando con su experiencia una visión cercana a aquellos a quienes la propuesta está dirigida.

Los docentes que dictamos el curso DAED también participamos del dictado del curso de Programación1 y por eso pudimos utilizar esta propuesta del DAED para agregar los ejercicios complementarios siguiendo su enfoque. No ha sido posible hasta el momento hacer un cambio más profundo (por ejemplo, modificar el material teórico de las figuras 1 y 2), ya que el equipo docente de Programación1 consta de doce docentes (son casi mil alumnos) y los cambios deben ser concensuados. Más de la mitad de los docentes del equipo participa dictando sus clases de manera tradicional conociendo muy poco de temas didácticos (lo que es bastante usual en las universidades).

Otros estudiantes del curso DAED realizaron asimismo trabajos muy interesantes sobre temas de otros cursos, tanto universitarios como de enseñanza media y terciaria, mostrando que la experiencia es aplicable a otros contenidos. Algunos participantes del curso DAED que son a la vez docentes, sobretodo de enseñanza media, aportaron puntualmente en el diseño o la modificación de materiales y metodología en la introducción de algunos temas. Sin embargo, la propuesta no es institucional con lo cual se dificulta su implementación de forma estable. Los cambios educativos a nivel institucional son muy difíciles de lograr ya que en su implementación intervienen factores ajenos al carácter académico de una propuesta.

Este artículo tiene el objetivo de compartir y difundir la experiencia, resaltando especialmente la importancia de brindar formación didáctica a estudiantes de informática que a través del estudio de temas didácticos pueden llegar a desubrir una vocación docente inesperada. Muchos de ellos son incluso docentes ayudantes o asistentes en las carreras de grado, que pueden vislumbrar más allá de la práctica concreta al tomar contacto con la teoría. Como hemos descrito, se vuelven capaces de diseñar contenidos desde una perspectiva que les permite poner el foco en las dificultades que entraña el estudio de los temas por primera vez, cosa que muchas veces tanto los libros de texto como los materiales elaborados por los docentes olvidan.

6 Agradecimientos

Agradecemos a los estudiantes del curso DAED autores de la propuesta Joaquín Cardozo y Matías de Horta.

References

1. Brousseau, G.: Theory of didactical situations in mathematics. Dordrecht: Kluwer (1997)
2. Christian Holmboe, L.M., E.George, C.: Research Agenda for Computer Science Education. In G.Kadoda (Ed). Proc. PPIG 13, pp 207-223 (2001)

3. Piaget, J.: La Prise de Conscience. Presses Universitaires de France (1964)
4. Piaget, J.: Success and Understanding. Harvard University Press (1974)
5. Piaget, J.: Genetic Epistemology, a series of lectures delivered by Piaget at Columbia University, translated by Eleanor Duckworth. Columbia University Press (1977)
6. Piaget, J., coll.: La Formation des Raisonnements Recurrentiels. Presses Universitaires de France (1963)
7. da Rosa, S.: Designing Algorithms in High School Mathematics. Lecture Notes in Computer Science, vol. 3294, Springer-Verlag (2004)
8. da Rosa, S.: The Learning of Recursive Algorithms from a Psychogenetic Perspective. Proceedings of the 19th Annual Psychology of Programming Interest Group Workshop, Joensuu, Finland pp. 201–215 (2007)
9. da Rosa, S.: The Construction of the Concept of Binary Search Algorithm. Proceedings of the 22th Annual Psychology of Programming Interest Group Workshop, Madrid, Spain pp. 100–111 (2010)
10. da Rosa, S.: The construction of knowledge of basic algorithms and data structures by novice learners. Proceedings of the 26th Annual Psychology of Programming Interest Group Workshop, Bournemouth, UK (2015)
11. da Rosa, S.: Preconceptions of novice learners about program execution. Proceedings of the 27th Annual Psychology of Programming Interest Group Workshop, Cambridge, UK (2016)
12. da Rosa, S.: Piaget and Computational Thinking. CSERC '18: Proceedings of the 7th Computer Science Education Research Conference pp. 44–50 (2018)
13. da Rosa, S.: Students teach a computer how to play a game. LNCS of The 11th International Conference on Informatics in Schools ISSEP 2018 (2018)
14. Winslöw, C.: Didactics of Mathematics - The French Way. Texts from a Nordic Ph.D.-Course at the University of Copenhagen. Center for Naturfagenes Didaktik, University of Copenhagen (2005), http://www.ind.ku.dk/publikationer/inds_skriftserie/2005maj4F/4F.pdf