Re armado de código en un primer curso de programación estructurada

Agustín Russell[1], Alejandro Miños[2]

[1] Instituto Superior Brazo Oriental. Dirección General de Enseñanza Técnica Profesional (Uruguay)
agustin.russell@docente.ceibal.edu.uy

[2] Instituto Normal de Enseñanza Técnica. Consejo de Formación en Educación. (Uruguay)
alejandromifa@gmail.com

Resumen. El primer curso de programación resulta de particular dificultad para los estudiantes, pues es necesario adquirir destrezas de modelado de soluciones y el uso de patrones. El estudiante debe comprender y poner en práctica la sintaxis para una serie de sentencias que son las que permiten construir programas; relacionando las sentencias de tal modo que permiten resolver la semántica de un problema dado. Este trabajo presenta una experiencia en la cual se modifican programas dados por el docente de modo que es necesario que el estudiante re ordene las sentencias o haga pequeñas modificaciones para generar una semántica correcta. Se observa que los estudiantes se ven motivados en la actividad, sobre todo aquellos que tienen menor rendimiento académico. Al mismo tiempo se percibe que los conceptos fuertes asociados a la actividad son comprendidos por el estudiantes, motivado esto tal vez por la reducción de la dificultad en la sintaxis.

Palabras clave: Didáctica de la informática. Programación. Estrategia.

1. Introducción

En Uruguay, luego de los tres años de formación post primaria, los adolescentes tienen la opción de realizar los cursos de bachillerato o bien una formación profesionalizante. Los primeros permiten el ingreso a cursos terciarios, pudiendo además formar al estudiante para el mercado laboral. Este tipo de cursos están bajo la órbita de la Dirección General de Educación Técnica Profesional, en los denominados bachilleratos tecnológicos [1]. Los bachilleratos tecnológicos tienen una duración de tres años, con un eje equivalente (común a todas las especialidades de bachilleratos) y un segundo eje específico, propio de la especialidad.

En el caso de los bachilleratos de informática, y dentro del eje específico, destacan los ejes de mantenimiento, videojuegos y desarrollo [2]. El primer eje se asocia a la administración de sistemas operativos y mantenimiento de hardware, el segundo a

estudiar las bases del desarrollo de videojuegos, en tanto que el tercero se orienta al estudio de la programación, bases de datos, fundamentos de ingeniería de software.

En el caso de las asignaturas Programación I y II se observa que las primeras tienden a resultar difíciles para los estudiantes, con relativos altos grados de reprobación. Lo anterior no solo está dado por la poca cantidad de horas semanales de estudio, sino que además por la carga cognitiva asociada a la resolución de problemas [3]. La importancia de la programación es tal que la misma es estructurante del currículo, teniendo su incidencia en casi todas las áreas de disciplina [4]. Por tanto, el estudio de la programación y la adquisición de las técnicas adecuadas permite no solo la construcción de algoritmos en un lenguaje dado, sino que además está en relación con una mejor comprensión de la propia disciplina.

2. Enseñanza de la informática

Múltiples desafíos debe enfrentar el docente de un primer curso de programación (objeto de estudio del presente trabajo) a la hora de planificar sus actividades.

El proceso de abstracción y modelado propio de la programación resulta complejo para el estudiante, quien debe dominar una serie de técnicas al tiempo que evaluar la correcta aplicación de cada una de ellas cuando existen muchas disponibles [5]. Si bien en un primer curso de programación la abstracción puede ser reducida a la elección de variables o a la elección de las sentencias adecuadas, no menos cierto es que la actividad no es trivial para un estudiante que se enfrenta por primera vez al hecho de programar.

La fuerte tendencia a la concretización que evidencian los estudiantes dificulta el proceso de abstracción y la resolución de problemas cuando no se tiene la posibilidad de interactuar con el propio objeto de estudio [6]. Esta puede ser subsanada parcialmente con la construcción de actividades en las cuales se opere con entradas que son procesadas y generan salidas que son relevantes y útiles para el estudiante. Sin embargo, antes de generar este tipo de actividad el estudiantes debe tomar distancia de los casos particulares y resultados, es decir, reducir el grado de concretización de la solución.

Al mismo tiempo el docente debe procurar que el estudiante aprenda una serie de patrones de trabajo como aquellos conocidos por el programador experto [7]. En un primer curso de programación, los patrones y el modelado de la solución se asocian principalmente con la identificación de variables, iteraciones o condiciones [8]. Sin embargo, describir una serie de estrategias no es sinónimo de aprendizaje de las técnicas de programación, debiendo ser posible también la puesta en práctica de las mismas en la resolución de problemas concretos [9]. Al mismo tiempo es importante que el estudiante comprenda además de la sintaxis de una sentencia, su semántica y utilidad [10].

Por otro lado debe considerarse que el docente debe planificar una serie de actividades en las cuales se trabajen los conceptos fuertes del curso sin que ellos repercuta en una sobrecarga cognitiva [11], como cuando se hace abuso de ejemplos y conceptos matemáticos en la construcción de algoritmos. En ocasiones el docente puede hacer énfasis en la resolución de programas que si bien permiten aprender

conceptos fuertes de programación, pero que sin embargo generan una dificultad extra al estudiante.

Vistas las dificultades en el aprendizaje de la programación, es que el docente se encuentra frente a la necesidad de generar actividades que resulten motivantes para el alumno, que sean relevantes para el mismo y logren captar su atención. Al mismo tiempo es necesario que en la planificación de las actividades el estudiante reciba la adecuada retroalimentación de sus logros y resultados, transformando así a cada actividad en parte de un verdadero proceso de evaluación formativo [12].

3. Re armando código

En el presente trabajo se analiza una estrategia áulica en la cual el docente provee una serie de sentencias, desordenadas o incompletas, que resuelven un problema pequeño. Mientras algunas sentencias están completas, en el sentido que no requieren modificación alguna para su correcto funcionamiento, otras necesitan ser complementadas con la inclusión de un operador lógico. Evidenciado el hecho que los nóveles programadores presentan dificultad en la relación semántica entre las distintas sentencias [13], se entiende que este tipo de actividad busca que el estudiante no analice solamente unidades independientes, sino que las organice en programa semánticamente adecuado. Esta actividad parte del supuesto que ya se han trabajado en clase todos los contenidos incluidos en ella, teniendo por tanto la intención de fortalecer los saberes obtenidos.

Entendiendo importante realizar actividades en las cuales se fomente el aprendizaje colaborativo, es que se organizan los estudiantes en equipos, donde cada integrante del grupo aporta y comparte sus experiencias y conocimientos con el resto de los compañeros. Presentada la consigna los estudiantes conforman grupos y realizan la actividad; algunos alumnos con altos rendimientos académicos pueden trabajar solos. Durante el trabajo en equipo y una vez finalizada la actividad el docente revisa la solución y analiza los resultados obtenidos. En todos los casos se hace una puesta en común en donde se ordenan bloques, se completan las incógnitas y / o se descubren los errores existentes. Se genera un espacio de debate, preguntas, y análisis respecto a la solución planteada por el docente y las que surgieron en trabajos previos.

El docente actúa como facilitador y guía, realizando preguntas orientativas ayudando a que los alumnos identifiquen y reconozcan la importancia de sentencias, analicen el uso de una iteración o cuántas veces se realiza una acción, entre otras. De este modo se busca que el estudiante no solo resuelva el problema, sino que además reflexione sobre la propia construcción de la solución y analizando los motivos de sus errores.

3.1 Primera actividad: re armado de código con incógnitas.

La consigna es la siguiente: “ordena los bloques de código para solicitar edades por teclado hasta que se ingrese una edad negativa. En ese momento, la solución deberá indicar la edad mayor ingresada.”

Algoritmo 1. Cada numeración indica un grupo de sentencias en desorden dadas a los estudiantes.

```
1) System.out.println("La edad mayor es " + maximo);
2) Scanner dato = new Scanner(System.in);
   int maximo = 0;
   int edad = 0;
3) System.out.println("Ingrese edad: ");
4) if (edad __ maximo) {
   }
5) maximo = edad;
6) do {
7) System.out.println("La edad mayor es " + maximo);
8) edad = dato.nextInt();
9) } while (edad ___ 0);
```

Algunas de las preguntas formuladas a los alumnos fueron: ¿por qué se utiliza do-while en lugar de while?; ¿qué pasa cuando la edad ingresada es mayor al máximo? ¿cuál pasa a ser el máximo?; ¿qué pasa cuando ingresas un número negativo? ¿Se cierra el programa?; ¿El ingreso de la edad lo hacen una sola vez o tienen que iterar?. Algunos errores o respuestas típicas de los estudiantes es que ignoran la iteración de la solicitud de la edad, desconocen el uso correcto de do-while, el programa entra en bucle infinito o la edad máxima no se guarda realmente en tiempo de ejecución.

Construida la solución se pide a los alumnos que realicen la corrida a mano del programa, analizando cómo las variables cambian durante el tiempo de ejecución. Si bien es cierto que la computadora es un elemento clave desde la didáctica de la informática [8] y su no uso es una debilidad, la ausencia de errores sintácticos en esta actividad reduce la dificultad asociada a su ausencia, permitiendo que el alumno centre su esfuerzo en la semántica del problema. Posteriormente se solicita a los estudiantes a probar el programa en la computadora.

3.2 Segunda actividad: re armado de código con descubrimiento de errores.

Está vez, la experiencia surge a partir de hacer una devolución de un programa ya realizado como tarea domiciliaria.

Esta actividad solicita realizar un menú donde se ingresarán notas y se mostrará un informe de calificaciones de suficiencia, insuficiencia y promedio. Es un problema conocido para los estudiantes. Al menú se puede acceder con las credenciales correctas, es decir se solicita una validación de usuario y contraseña predeterminada. La solución correcta implica el uso de operadores lógicos y aritméticos, condicional con alternativa, estructura repetitiva do-while, estructura de control switch, variables acumulativas y contadoras.

Este programa tiene una extensión de unas 50 líneas, siendo relativamente complejo para el momento en el cual se desarrolla el curso. Se intenta mantener el nivel de abstracción adecuado y que las porciones del código funcionen como entidades que solucionan subproblemas del ejercicio a resolver. Es por esto, que las porciones de código a ordenar se vuelven más grandes. Re armar el código con mayor

cantidad de líneas puede llevar a una sobrecarga cognitiva [11] y complejizar de tal modo el ejercicio que dificulta innecesariamente la actividad. Por el mismo motivo, el código se brinda indentado reforzando la necesidad de legibilidad adecuada de un algoritmo.

Algoritmo 2. Ejemplo de una de las porciones de código que resuelve un subproblema.

```
1)  do {
2)          System.out.println("Ingrese nota: ");
3)          nota = dato.nextInt();
4)          if (nota >= 1 || nota <= 12) {
5)                              cantidad++;
6)                              total = nota + total;
7)                              if (nota >= 8 && nota <= 12) {
8)                                      reprobados++;
9)                              } else {
10)                                     aprobados++;
11)                             }
12)         }
13) } while (nota >= 1 && nota <= 12);
```

Se pueden observar algunos errores, los cuales sean identificados por parte del equipo de trabajo. En primer lugar en la línea 4, es necesario usar el operador lógico AND (&&) en vez del operador OR (||) ya que la calificación válida es del 1 al 12. Otra opción válida posible es que estos operadores sean planteados como incógnita y se pregunte a los estudiantes cuál es el correcto, aunque en ese momento del curso la primera opción se considera más adecuada. Por otro lado las variables que cuentan la cantidad de estudiantes aprobados y reprobados están invertidas de modo que se logre identificar el uso y funcionalidad de cada una. Finalmente, si bien no se considera un error, la condición del do-while se plantea de forma diferente a la condición del if a propósito. En particular, nota >= 1 && nota < 13 es lo mismo que nota >= 1 && nota <= 12 debido a que nota es una variable entera. Sin embargo, su incidencia en el algoritmo es diferente.

Podemos observar que los errores mencionados tiene relación con la semántica del problema a resolver (como ser en la línea 4). Este tipo de error busca que los estudiantes no se centren exclusivamente en la sintaxis, sino que además procuren entender el problema. La reflexión de lo mencionado permite que el estudiante analice cómo crear las condiciones, y justifique el uso de estas dentro de las estructuras, siendo esto una de las mayores dificultades a las cuales se enfrentan los futuros programadores.

Hemos observado que en principio los estudiantes centran su análisis en la sintaxis, en identificar el correcto uso del el punto y coma, las llaves o los paréntesis. Es necesario analizar con los estudiantes la unidad del problema, la intención del programa y de quien lo programó para que este tipo de error se identifique. Al mismo tiempo, como forma de guía se indica a los estudiantes la cantidad de errores existentes, para que sea tomado como referencia del grado de avance de la tarea. En otras porciones de código, se agregan más errores para trabajar con las características

adecuadas de un menú como el uso de break o default. Se realiza una puesta en común del orden adecuado del algoritmo y los errores existentes. Así se da espacio a la autoevaluación, intercambio, y análisis de los contenidos ya presentados con anterioridad.

4. Conclusiones

La técnica de re armado de código parece fomentar la abstracción ya que el estudiante debe olvidarse de los problemas asociados al entorno de desarrollo. Al mismo tiempo la sintaxis no resulta un problema, pues la misma no presenta errores. De este modo se logra que el estudiante se concentre en cómo resolver el problema planteado. Si bien es cierto que en ocasiones las sentencias están incompletas (y serían inadecuadas para ser usadas en el IDE), la guía que se da el indicar su uso resulta en una fortaleza y no genera dificultades mayores a no ser por la identificación de la sentencia. Al ser un curso de introducción a la programación este es un detalle muy significativo ya que muchas frustraciones generadas en los alumnos viene dada por la dificultad de manejar el IDE, su sintaxis y su idioma no nativo.

Es así que la actividad permite la solución de problemas concretos, o coadyuva a ello. A su vez, el estudiante más experimentado dialoga, debate y construye a la par con el estudiante con mayores dificultades. Estos últimos se notan motivados y activos en el ordenamiento de los bloques de código, pareciendo lograrse la motivación por el logro. En efecto, este grupo de estudiantes perciben que son parte de la construcción de una solución que efectivamente funciona, y que si bien el docente ha participado en ella, son ellos quienes han logrado que dicha solución resuelva el problema planteado. Los estudiantes declaran que la dificultad que tienen al usar un entorno de desarrollo no la tienen al re armar los bloques de código. Se muestran competentes al descubrir y entender los motivos de los errores presentados.

Esta estrategia no garantiza la aplicación de código en el entorno de programación, ni se genera un espacio de análisis y diseño de como resolver problema ya que la solución es limitada a una sola respuesta correcta. Sin embargo, es cierto que genera un esfuerzo cognitivo extra al ser necesario relacionar o interpretar las sentencias dadas, de modo de mantener la semántica solicitada.

Aunque no se ha sistematizado ni cuantificado los resultados, en la población estudiada, la técnica parece lograr buenos resultados. En efecto, mediante la discusión sobre cómo la ordenación del código, resolución de incógnitas y descubrimiento de errores, podemos ver que el estudiante logra resolver la consigna, construyendo colectivamente una solución.

5. Referencias

[1] <<Educación Media Tecnológica. ,>>[En línea]. Available: https://www.utu.edu.uy/media-superior/educacion-media-tecnologica. [Último acceso: 12 09 2022].

[2] <<Educación Media Tecnológica Informática énfasis Desarrollo y Soporte, Desarrollo Web, Videojuegos,>>. [En línea]. Available: https://www.utu.edu.uy/educacion-media-tecnologica-informatica-enfasis-desarrollo-y-soporte-desarrollo-web-videojuegos. [Último acceso: 12 09 2022].

[3] Y. Liao y G. Bright, <<Effects of computer programming on cognitive outcomes: A meta-analysis. Journal of educational computing research,>> 1991, 7(3), 251-268. doi:10.2190/e53g-hh8k-ajrr-k69m

[4] P. Denning, D. Comer, D. Gries, M. Mulder, A. Tucker, A. Turner y P. Young, <<Computing as a discipline. Communications of the ACM,>> 32(1), 9 – 23.

[5] A. Aho, J. Hopcroft y J. Ullman, <<Estructuras de datos y algoritmos,>> México: Adisson Wesley Longman, 1998.

[6] O. Meerbaum-Salant, M. Armoni y M Ben-Ari, <<Learning computer science concepts with scratch. Computer Science Education,>> 23(3), p. 239-264, 2013.

[7] A. Robins., J. Rountree y N. Rountree, <<Learning and Teaching Programming: A Review and Discussion. Computer Science Education,>> 13(2), 137–172. doi:10.1076/csed.13.2.137.14200, 2003.

[8] A. Miños, <<Elementos estructurantes de la Didáctica de la Informática. Virtualidad, Educación y Ciencia,>> vol. 8, n° 14, p. 100 – 110, 2017.

[9] S. P. Davies, <<Models and theories of programming strategy. International Journal of Man-Machine Studies,>> 39(2), 237–267. doi:10.1006/imms.1993.1061, 1993.

[10] B. Xie, G. L. Nelson y A. J. Ko, <<An Explicit Strategy to Scaffold Novice Program Tracing,>> En Proceedings of the 49th ACM Technical Symposium on Computer Science Education - SIGCSE '18. doi:10.1145/3159450.3159527, 2018.

[11] J. Astolfi, <<El error, un medio para enseñar,>> Sevilla: Diada Editora, 1999.

[12] E. Fiore, y J. Leymonié, <<Didáctica práctica para Enseñanza Media y Superior,>> Montevideo: Editorial Grupo Magro, 2007.

[13] L.E. Winslow, <<Programming pedagogy – A psychological overview,>> SIGCSE Bulletin, 28, 17–22, 1996.