UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Modelado didáctico de interacción humano-computadora

Informe de Proyecto de Grado presentado por

Eliana Rosselli, Santiago Correa, Cecilia Guayta

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Sylvia da Rosa
Federico Gómez Frois

Montevideo, 21 de agosto de 2023

Agradecimientos

En primer lugar queremos agradecer a nuestras familias y amigos, a todas las personas que nos han apoyado durante este largo camino. Su paciencia, aliento, y apoyo constantes han sido fundamentales para transitar este proceso.

Agradecemos a nuestros supervisores, Sylvia y Federico, por guiarnos a lo largo de todo el proyecto. Su conocimiento y experiencia en este campo fueron esenciales para el desarrollo de nuestro trabajo; sus valiosos consejos, comentarios y sugerencias nos han impulsado a mejorar constantemente.

Agradecemos a todos los participantes de las instancias de prueba, cuya contribución fue una parte clave del proyecto: Alejandra, Pilar, María Eugenia, Manuel, Nicolás, Clara, Rossina, Lucía, Diego, Sofía, Kariana, Juan, Guadalupe, Juan Pedro, Valentín, Noelia, Lill, Juan Andrés, Ana Inés, Rosina y Paulina. Agradecemos a cada uno de ustedes su voluntad de participar y compartir su tiempo y sus experiencias, que han enriquecido nuestra investigación y sin los cuales el proyecto no hubiera sido ni la mitad de interesante.

Más allá de esta breve lista, queremos agradecer a todas las personas que han sido parte de nuestro camino en esta carrera. Cada uno de ustedes sabe que sus aportes, grandes y pequeños, contribuyeron a que lleguemos hasta acá.

¡Muchas gracias!

Resumen

Una de las diferencias más importantes entre la comunicación entre humanos y la comunicación humano-computadora es el lenguaje: el lenguaje natural es ambiguo y dependiente del contexto, mientras que los lenguajes de programación tienen una sintaxis determinante, limitando lo expresado a una única interpretación. La interacción de un pensamiento humano con una computadora a través de un lenguaje de programación lo transforma en un pensamiento computacional; el pensamiento humano es capaz de resolver y razonar sobre problemas algorítmicos, mientras que el pensamiento computacional es necesario para programar una computadora para que esta los resuelva.

El objetivo general del proyecto es diseñar e implementar una herramienta de apoyo didáctico que obligue al estudiante a seguir una estrategia donde deba transformar su pensamiento en computacional para lograr un objetivo. La herramienta propuesta tiene como objetivo concreto fomentar el desarrollo del pensamiento computacional relacionado a los conceptos de variables y asignación.

Se diseñó e implementó una página web que contiene cuatro conjuntos de ejercicios, donde se trabaja el concepto de asignaciones de las forma `variable = valor` y `variable = expresión`. El diseño de la aplicación y los ejercicios se basa en la teoría epistemológica de Piaget, así como en el modelo didáctico *Hyibrid Interaction System* (HIS).

Posteriormente se realizaron dos instancias de pruebas con participantes de entre 15 y 25 años. El objetivo principal de estas instancias fue validar el diseño de los ejercicios, además de detectar errores y espacios de mejora tanto en la experiencia de usuario de la herramienta como en el diseño de los ejercicios. Al finalizar la primera instancia de pruebas, se realizaron modificaciones, para luego poder realizar una segunda instancia. Se observó que si bien no todos los participantes pudieron resolver todos los ejercicios sin ayuda, sí pudieron construir conocimientos sobre variables y asignación, incluso sin tener conocimientos previos de programación. Estas instancias evidenciaron que la herramienta cumple con el objetivo de brindar apoyo didáctico para el desarrollo del pensamiento computacional.

Palabras clave: Variable, Asignación, Construcción de conocimiento

Índice general

Capítulo 1

Introducción

Una de las diferencias más importantes entre la comunicación entre humanos y la comunicación humano-computadora es el lenguaje: el lenguaje natural es ambiguo y dependiente del contexto, mientras que los lenguajes de programación tienen una sintaxis determinante y reducen la capacidad de expresión (palabras claves, reglas estrictas, etc.), limitando lo expresado a una única interpretación. El lenguaje natural tiene una estrecha relación dialéctica con el pensamiento humano: "la relación entre pensamiento y palabra no es un hecho, sino un proceso, un continuo ir y venir del pensamiento a la palabra y de la palabra al pensamiento, y en él la relación entre pensamiento y palabra sufre cambios que pueden ser considerados como desarrollo en el sentido funcional. El pensamiento no se expresa simplemente en palabras, sino que existe a través de ellas" (Vygotsky, 1981, pág. 95). La interacción de un pensamiento humano con una computadora a través de un lenguaje de programación lo transforma en un pensamiento computacional; el pensamiento humano es capaz de resolver y razonar sobre problemas algorítmicos, mientras que el pensamiento computacional es necesario para programar una computadora para que esta los resuelva. El proyecto está enfocado en fomentar esa transformación del pensamiento humano en pensamiento computacional, mediante la interacción del estudiante con una computadora en una etapa informal del aprendizaje.

1.1. Objetivo

El objetivo general del proyecto es diseñar e implementar una herramienta de apoyo didáctico que obligue al estudiante a seguir una estrategia donde deba formar y/o transformar su pensamiento en computacional para lograr un objetivo. Al ejecutar un código o pseudocódigo (lenguaje formal), la herramienta debe mostrar el efecto generado por este; asimismo debe permitir que el estudiante corrija el texto y vuelva a ejecutarlo, para ver los nuevos efectos que las correcciones producirán.

La herramienta está pensada para ser usada como complemento a instancias

de aprendizaje formales, como por ejemplo una clase dictada por un docente, donde el pensamiento se va transformando en computacional en el contexto de uno o más lenguajes formales de programación.

La herramienta propuesta tiene como objetivo concreto fomentar el desarrollo del pensamiento computacional relacionado a los conceptos de variables y asignación. Estos son dos de los conceptos fundamentales de la programación imperativa, en los que se basa la construcción de conocimientos más avanzados. Para lograr el objetivo, se plantea un sistema de interacciones entre el estudiante y la herramienta, donde las acciones de uno afectan las acciones del otro. El estudiante interactúa con la herramienta, que se ve afectada, y estos cambios a su vez incitan nuevas acciones por parte del estudiante, constituyendo un ciclo de interacciones (Schulte y Budde, 2018). En este ciclo, es clave que los estudiantes puedan ver el efecto de sus acciones y así auto-corregir sus errores sin necesidad de intervención docente.

Los capítulos del informe se organizan de la siguiente manera: en el capítulo 2 se describe brevemente el marco teórico del proyecto, basado en principios de la teoría epistemológica de Jean Piaget. En el capítulo 3 presenta los ejercicios propuestos, agrupándolos en cuatro conjuntos; cada conjunto representa un aspecto crítico de los conceptos elegidos (variable y asignación), es decir aspectos que presentan dificultades para el aprendizaje de los conceptos. En el capítulo 4 se fundamenta el diseño de los ejercicios propuestos, relacionándolos con el marco teórico presentado en el capítulo 2. El diseño y la implementación de la herramienta se describen en general en el capítulo 5, siendo algunos detalles presentados en el Anexo A. En el capítulo 6 se describen las dos instancias de prueba realizadas con estudiantes, que permitieron introducir mejoras en la herramienta y las visualizaciones de los ejercicios. Por último, el capítulo 7 describe conclusiones del proyecto y oportunidades de trabajo futuro.

Capítulo 2

Marco teórico

Este capítulo describe brevemente el marco teórico utilizado para el proyecto, que es la base sobre la cuál se justifican los distintos aspectos del diseño de la herramienta y los ejercicios propuestos. Las fuentes principales son el modelo didáctico *Hybrid Interaction System* (HIS) (Schulte y Budde, 2018) y principios de la teoría epistemológica de Jean Piaget (Gomez, 2021; Gómez y da Rosa, 2023).

2.1. Hybrid Interaction System

El proyecto se basa en el modelo didáctico *Hybrid Interaction System* (HIS) (Schulte y Budde, 2018), que propone una guía didáctica que no parte de los aspectos teóricos de las ciencias de la computación, sino de las interacciones entre el humano y la computadora. Este modelo se basa en el concepto alemán *Bildung*: el concepto del aprendizaje como una transformación del individuo y sus percepciones. Se considera el aprendizaje como una transformación de la auto-percepción (cómo el individuo se percibe a sí mismo), pero también una transformación de cómo percibe al mundo y los objetos en este. Esto introduce la idea de educación como interacción: al interactuar con el mundo lo transformamos, tanto literalmente como mediante la transformación de nuestra percepción de este, y al mismo tiempo somos transformados.

Esta interacción como base del aprendizaje es la idea fundamental detrás del HIS. La parte central del modelo HIS es la interacción recíproca entre los dos actores (humano y computadora), representada por una secuencia de acciones alternadas: acción del humano seguida por acción de la computadora seguida por acción del humano y así sucesivamente. Un diagrama representativo del HIS puede verse en la figura 2.1.

Ambos actores están formando y siendo formados por el HIS en este proceso de interacción continua, donde el resultado de una acción retro-alimenta la siguiente. La computadora transforma al individuo, y a su vez el individuo transforma a la computadora: los roles de cada actor no pueden ser definidos sin

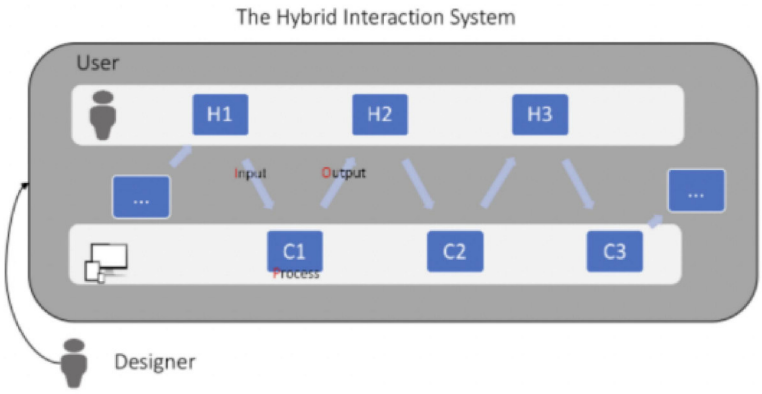

Figura 2.1: Hybrid Interaction System (HIS) definido en (Schulte y Budde, 2018)

el contexto de estas interacciones y del sistema del cual son parte. Para lograr esto, no se debe crear un sistema informático con procesos fijos, sino un contexto de interacción sobre el cual el humano y la computadora puedan actuar. Al interactuar con la computadora en este contexto, la persona puede hacerse preguntas como "¿Cuáles son los efectos de mis acciones?" y sacar conclusiones, construyendo nuevo conocimiento.

2.2. Teoría epistemológica de Piaget

Las interacciones dentro del HIS pueden ser analizadas en el marco de la teoría epistemológica de Piaget, que establece que "la fuente del conocimiento se remonta a las acciones del sujeto en interacción con objetos en su medio". Esta teoría explica el proceso de construcción del conocimiento mediante el pasaje del sujeto por una tríada de etapas. En primer lugar se tiene la etapa intra, donde el sujeto se concentra en objetos y elementos aislados; luego está la etapa inter, donde el sujeto tiene en cuenta las relaciones entre los objetos y sus transformaciones; por último, en la etapa trans el sujeto construye un sistema general, es decir estructuras generales que involucran tanto elementos generalizados como sus transformaciones, y donde se integran las construcciones de las etapas anteriores como casos particulares.

Piaget formuló la ley general de la cognición (ver la figura 2.2), que gobierna la relación entre el "saber hacer" y la conceptualización, construida en la interacción entre el sujeto y los objetos con los que tiene que trabajar para resolver problemas o realizar tareas. Esta relación es una relación dialéctica, donde a veces las acciones guían a los pensamientos y a veces los pensamientos guían a las acciones.

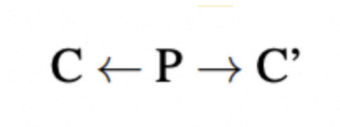

Figura 2.2: Ley general de la cognición

En el diagrama de la figura 2.2, P representa la periferia, es decir la reacción exterior e inmediata del sujeto al enfrentarse a los objetos para resolver un problema. Esta reacción está enfocada en alcanzar un objetivo y lograr resultados, sin conciencia de las acciones tomadas ni de las razones por las cuales se tuvo éxito (o se fracasó). Las flechas representan el mecanismo interno del proceso de pensamiento, mediante el cual el sujeto toma conciencia de la coordinación de sus acciones (C) y los cambios que causan en los objetos, además de las propiedades intrínsecas de estos (C'). El proceso de toma de conciencia gobernado por esta ley constituye un primer paso hacia la construcción de conocimiento.

Cuando consideramos la construcción de conocimiento computacional, es necesario tener en cuenta algunas particularidades propias de la computación. Un ejemplo de esto es un algoritmo de búsqueda lineal en una colección de elementos: un estudiante puede ser capaz de expresar en pseudocódigo un algoritmo correcto para resolver el problema, pero al escribirlo en un lenguaje formal no lograr ejecutar exitosamente el programa. Esto puede deberse a problemas que solo surgen cuando la computadora ejecuta el programa y no en el algoritmo en sí, un caso particular es cuando se intenta acceder a un índice del arreglo que se va fuera de rango (causando un error en tiempo de ejecución).

Para lograr programar un autómata que resuelve un problema, se debe comenzar por hacer que el estudiante reflexione sobre cómo hace él lo que quiere que haga el autómata. Es decir, primero tienen que poder resolver ellos mismos el problema, construyendo el algoritmo. Además, los estudiantes tienen que establecer una relación causal entre el algoritmo (acciones del estudiante sobre los objetos) y la ejecución del programa (acciones de la computadora sobre estados). No solo tienen que entender el algoritmo, sino que deben entender también las condiciones que hacen que la computadora ejecute exitosamente el programa. Entonces, podemos decir que la programación de un autómata comienza por reflexionar: "cómo hago yo, lo que quiero que haga el autómata" (ver figura 2.3).

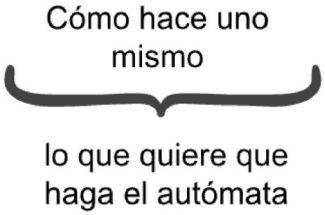

Figura 2.3: Reflexión inicial para programar un autómata (Gómez y da Rosa, 2023)

La relación causal entre ambas filas es la clave de la construcción de conocimiento sobre una máquina ejecutando un programa. Gómez y da Rosa (2023) definen análogamente la ley extendida de la cognición, partiendo de la ley de la cognición de Piaget. Esta ley extendida se muestra en la figura 2.4.

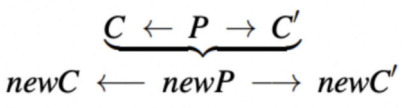

Figura 2.4: Ley extendida de la cognición (Gómez y da Rosa, 2023)

El conocimiento conceptual sobre el algoritmo (es decir el que se construye en las transiciones de P a C y C') constituye la nueva periferia (newP en el diagrama). La construcción de conocimiento sobre el programa se explica mediante las transiciones a los nuevos centros (newC y newC'), que representan la toma de conciencia sobre lo que sucede dentro de la computadora: la conceptualización de cómo la computadora ejecuta las instrucciones del programa (newC) y los cambios que estas imponen sobre las estructuras de datos usadas (newC'). Esta extensión de la ley de la cognición abarca tanto el pensamiento algorítmico, denotado por la primera fila del diagrama, como el pensamiento computacional, denotado por la segunda fila del diagrama, y la relación entre éstos, representada por la llave.

Piaget estableció que la construcción de conocimiento sobre métodos (en este caso algoritmos) y objetos (en este caso estructuras de datos) ocurre en la interacción entre C, P y C'. Análogamente, la ley extendida establece que la construcción de conocimiento sobre la ejecución de un programa, es decir la transformación a pensamiento computacional, ocurre en la interacción entre newC, newP y newC'. Además, para poder construir este conocimiento, es necesario ya haber construido el conocimiento sobre el algoritmo, tomando conciencia de sus acciones y sus efectos (transicionando desde P a C y C').

Piaget describe un instrumento cognitivo mediante el cual se dan estas transiciones desde la periferia hacia los centros, llamado abstracción reflexiva. En este proceso se proyectan las relaciones establecidas en el plano de la acción al plano del pensamiento, y además se introduce el entendimiento de las condiciones y motivaciones (da Rosa y Gomez, 2020). Esto es llamado por Piaget la "búsqueda de razones de éxito (o fracaso)": el sujeto experimenta el proceso de abstracción reflexiva motivado por entender cuál es el porqué de los efectos de sus acciones (ya sean exitosas o no). La visualización de la herramienta que se propone en este proyecto, donde el estudiante puede ver el efecto de sus acciones, se vuelve facilitadora en este proceso de "búsqueda de razones"; la visualización de las modificaciones que producen las instrucciones sobre los objetos computacionales cumple un rol análogo al de los objetos concretos que manipula el estudiante cuando resuelve un problema.

Capítulo 3

Ejercicios propuestos

En este capítulo se describen los ejercicios con los que la herramienta permite trabajar, agrupados en cuatro conjuntos que abarcan diferentes niveles de dificultad de la sentencia de asignación de variables, desde la comprensión de la relación posición-valor de una variable y la doble semántica, hasta la resolución de pequeños problemas generales. Los conjuntos son:

- Ejercicios básicos sobre asignación de variables
- Ejercicios sobre el orden de las instrucciones
- Ejercicios sobre intercambio de valores de variables
- Ejercicios de introducción a la formalización

Cada uno de estos conjuntos de ejercicios persigue un objetivo didáctico distinto. Para cada ejercicio se detalla el resultado esperado e intentos previstos de solución por parte de los estudiantes, incluyendo posibles errores. En la siguiente sección se describe brevemente cómo funciona la herramienta. Una descripción detallada del tutorial de la herramienta se encuentra en el Anexo A.3.

3.1. Estructura general de la herramienta

La estructura general de la herramienta se mantiene constante a lo largo de todos los conjuntos de ejercicios, y consiste en dos partes principales.

A la izquierda se tiene un panel de código, visible en la figura 3.1, para escribir las instrucciones y dos botones, uno para ejecutarlas y otro para reiniciar el ejercicio. Al apretar el botón de ejecución, siempre se ejecutan todas las sentencias una por una, mostrando en verde la instrucción que está siendo ejecutada. También hay un botón que permite agregar nuevas instrucciones (denotado por +) y botones (uno por instrucción) para borrar instrucciones (denotados por x). Cuando se presiona el botón de ejecución, los botones para agregar y borrar sentencias desaparecen hasta el final de la ejecución, donde vuelven a aparecer.

A la derecha se tiene un panel de visualización, visible en la figura 3.2, que muestra el resultado de la ejecución de las instrucciones (varía entre los distintos conjuntos de ejercicios). El panel de visualización permite al estudiante verificar si sus resultados son correctos o identificar y corregir sus errores. Los enunciados de los ejercicios se despliegan sobre ambos paneles.

Figura 3.1: Panel de código

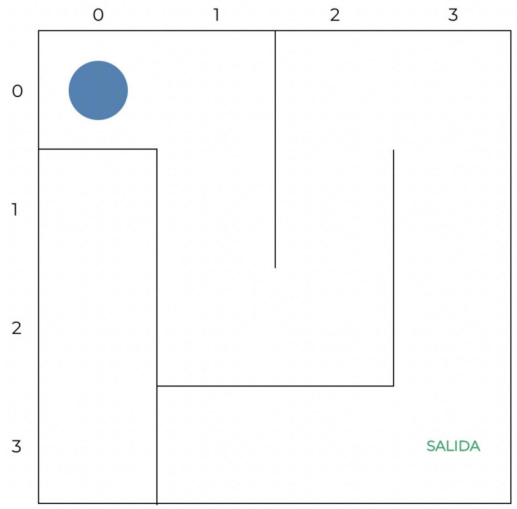

Figura 3.2: Panel de visualización (para ejercicios del segundo conjunto)

3.2. Ejercicios básicos sobre asignación de variables

El primer conjunto de ejercicios busca que los estudiantes se familiaricen con la sintaxis de la sentencia de asignación usada en la herramienta, y que reflexionen sobre el rol de una expresión a la izquierda o a la derecha del operador de asignación (=). En la figura 3.3 se muestran los paneles de código y de visualización para el primer ejercicio del conjunto. En el panel de código tenemos la asignación del valor 2 a una variable llamada posicion (que sería un índice de variable, por ejemplo si consideramos a la línea como un arreglo), y en el panel de visualización la situación de una pelota (que sería el valor de la variable cuyo índice es posicion) antes de la ejecución de la instrucción. Los valores que puede tomar la variable posicion están indicados por los índices en la recta.

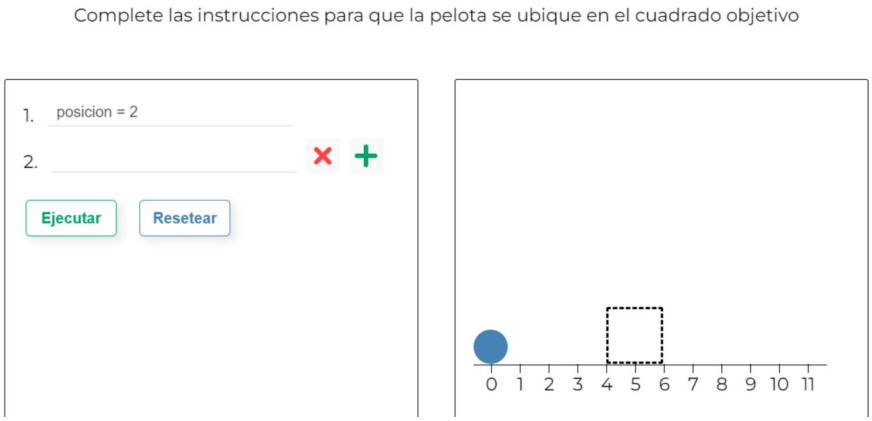

Figura 3.3: Paneles iniciales en el ejercicio 1

Todos los ejercicios de este conjunto tienen el mismo formato, donde el estudiante debe lograr que la "pelota" se ubique en un cuadrado determinado. En todos los casos, el problema puede ser resuelto con una única instrucción de asignación de la forma `variable = expresión`. En los primeros ejercicios, la solución toma la forma particular de `variable = valor`, mientras que en los ejercicios finales toma la forma de `variable = variable + valor`.

3.2.1. Ejercicio 1

Para este primer ejercicio se decidió darles a los estudiantes una primera instrucción de ejemplo, tanto de sintaxis como de resultado de ejecución. Al ejecutar esta instrucción, se visualiza cómo la pelota se mueve a la posición 2, como se muestra en la figura 3.4. Se pide que escriba una instrucción que asigne un valor a posicion para lograr el resultado. La respuesta correcta es `posicion = 5`.

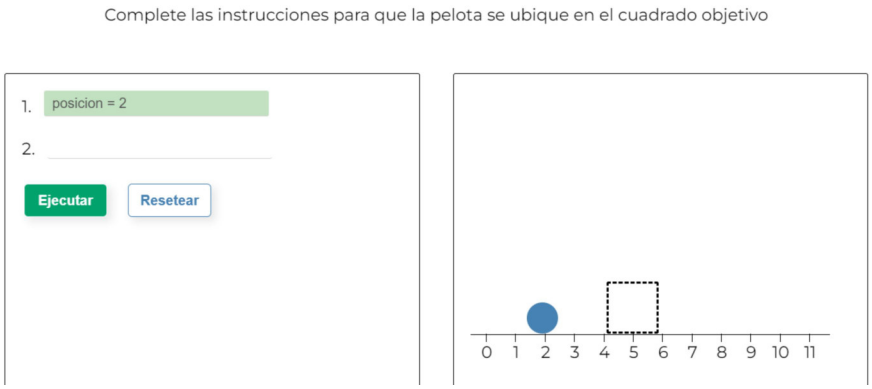

Figura 3.4: Paneles una vez que se ejecuta la instrucción dada.

Por un lado, el ejercicio permite que los estudiantes comiencen a familiarizarse con la herramienta. Por otro, el ejercicio es una primera experiencia del estudiante con las variables computacionales: un comportamiento esperado es que el estudiante escriba la instrucción `posicion = 3`, que al ser ejecutada mostrará que la pelota se mueve a la posición 3 y no a la 5 como se pide. Esto evidencia que el estudiante piensa en "¿cuánto falta para que la pelota llegue al 5?" y no en "¿cuál es el índice que corresponde al lugar que se pide?". El objetivo es que el estudiante reflexione sobre ello, y reasigne la variable posicion mediante la instrucción `posicion = 5`.

Además, se espera que los docentes elaboren otros recursos didácticos que generen en el estudiante la necesidad de experimentar con distintos valores a modo de explorar la herramienta, llegando a probar índices no válidos, como por ejemplo `posicion = 12` (ya que queda por fuera de los valores visibles en la recta, que oficia de límite de los valores, por ejemplo índice máximo de un arreglo) o valores no válidos, como `posicion = cuadrado`. Estos casos serán contemplados por la herramienta presentando distintos mensajes de error: "Instrucción inválida", "El valor de posición se va fuera de rango. Prueba un valor más pequeño.", entre otros.

La importancia de este ejercicio está en que permite relacionar el valor asignado con el acceso directo al lugar, diferenciándolo de la suma de las distancias, previendo que los estudiantes pueden estar influenciados por ideas preconcebidas.

3.2.2. Ejercicio 2

Este ejercicio intenta que el estudiante se enfrente a un problema similar al que ya resolvió en el ejercicio 1, con las diferencias de que no se da una instrucción inicial y que el cuadrado se ubica en la posición 10. El objetivo es consolidar mediante repetición la idea de que posicion es una variable que indica "un lugar" y no una distancia.

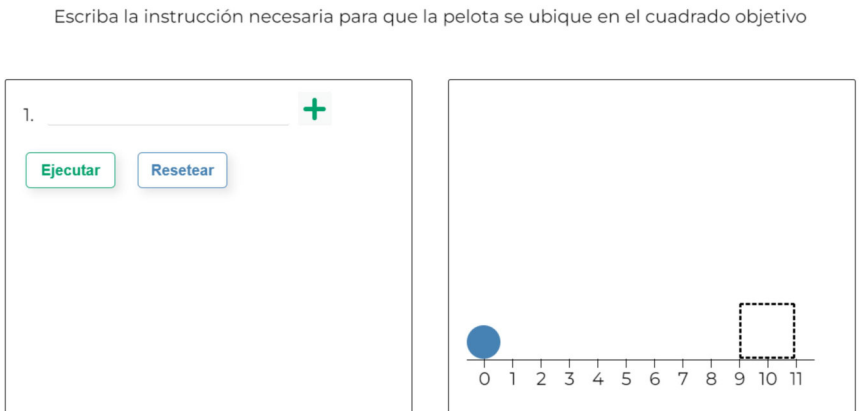

Figura 3.5: Paneles iniciales en el ejercicio 2

La solución correcta es `posicion = 10`. No se esperan errores particulares por parte de los estudiantes, aunque de ser necesario pueden intentar varias veces, ejecutando sus instrucciones para ver el resultado.

3.2.3. Ejercicio 3

En este ejercicio se continua trabajando el mismo objetivo que en el ejercicio anterior. También se realiza una variante, reemplazando las marcas en la línea que denotaban las posiciones (índices) por la distancia entre la pelota y el cuadrado objetivo, indicada con una llave (ver figura 3.6).

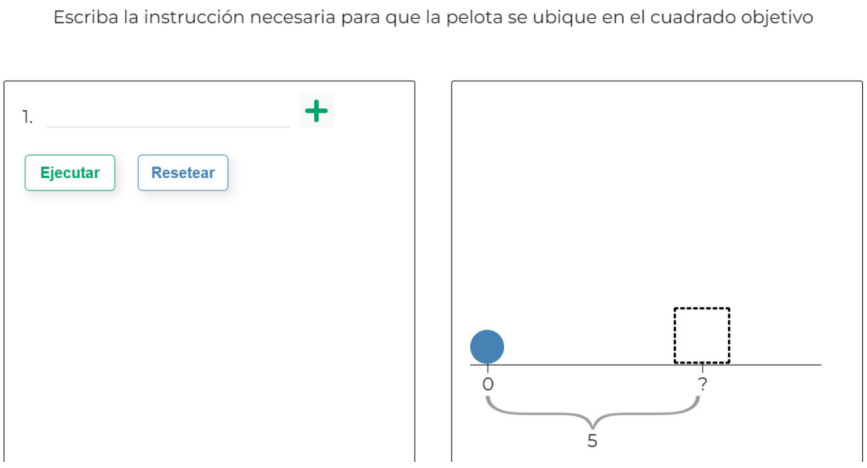

Figura 3.6: Paneles iniciales en el ejercicio 3

Se puede observar que la posición inicial de la pelota sigue siendo conocida,

pudiendo el estudiante conocer el valor exacto de la posición del cuadrado. Entre las soluciones posibles se tienen `posicion = 5` y `posicion = posicion + 5`. La primera es la solución esperada, ya que los estudiantes han asociado en los ejercicios anteriores la posición de la pelota con un índice de la recta; es entonces esperable que resulte natural asociar el índice a la distancia, considerando que la posición inicial es 0.

3.2.4. Ejercicio 4

El objetivo de este ejercicio es comenzar a dar variantes de la sentencia de asignación vista hasta el momento (`variable = valor`), para a partir de los siguientes ejercicios, llegar a una versión más general (`variable = expresión`). Para esto, se reemplaza la distancia del ejercicio anterior por dos distancias que deben ser sumadas para obtener la distancia total, y así poder determinar el índice correcto que debe asignarse a la variable posicion. Los paneles iniciales para este ejercicio se muestran en la figura 3.7.

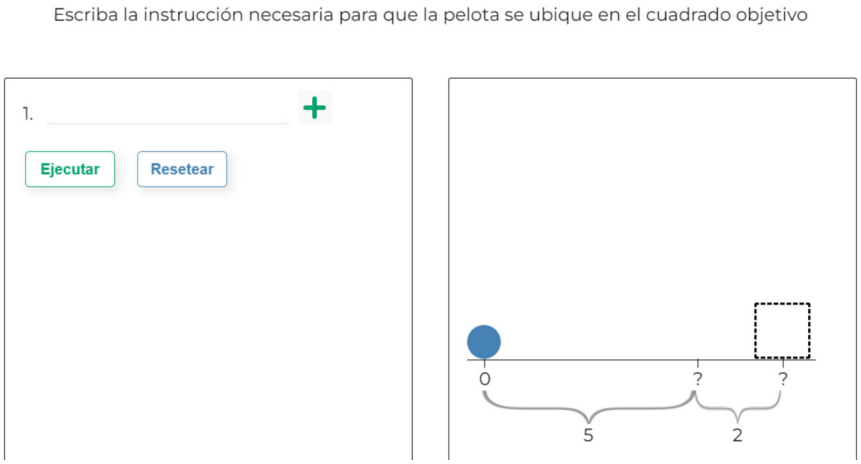

Figura 3.7: Paneles iniciales en el ejercicio 4

En este ejercicio existen diferentes soluciones con distinto nivel de generalidad. En una de las soluciones predomina lo trabajado en los ejercicios anteriores (`posición = 7`), mientras que en otras predomina lo visual (por ejemplo, resolviendo el ejercicio con una única sentencia `posicion = posicion + 7`, o dos sentencias separadas, `posicion = posicion + 5; posicion = posicion + 2`).

Un error posible es que el estudiante intente responder con dos sentencias independientes, como `posicion = 5; posicion = 2`, olvidándose de que la variable representa un índice (nuevamente interpretando posicion como la distancia que quiere que se mueva la pelota). En este caso, la herramienta responderá mostrando en el panel derecho a la pelota moverse hasta el valor asignado, pero

no llegaría a moverse hasta el cuadrado, frenando antes y retornando a la posición inicial, como se muestra en la figura 3.8. Un caso así evidenciaría que no se lograron los objetivos de los ejercicios anteriores y es necesario volver a ellos.

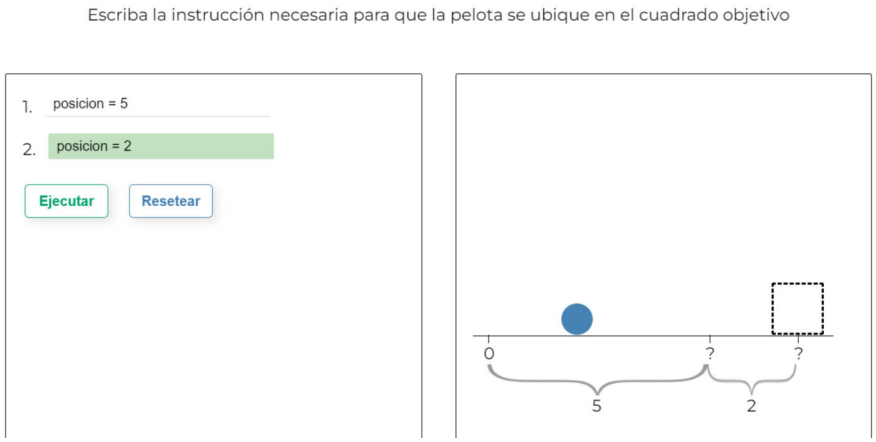

Figura 3.8: Posible error para el ejercicio 4

3.2.5. Ejercicio 5

El objetivo de este ejercicio y el siguiente es que el estudiante pueda distinguir el acceso directo a un índice, dado por las asignaciones de la forma `posicion = valor`, del acceso relativo a un índice anterior, obtenido de la distancia entre este y el objetivo. Para introducir esto, este ejercicio modifica la posición inicial de la pelota (ya no es 0), aunque esta sigue teniendo un valor conocido. Con este cambio, se espera comenzar a introducir la idea de que el valor inicial no siempre es el mismo (y en futuros ejercicios, ni siquiera será conocido). El ejercicio se muestra en la figura 3.9.

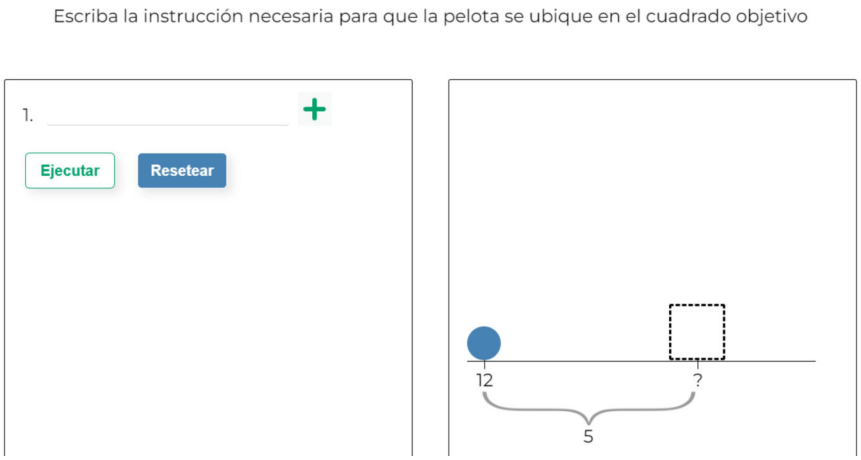

Figura 3.9: Paneles iniciales en el ejercicio 5

En este caso, el estudiante experimenta que el valor de la expresión de la derecha (posición inicial + distancia) da el valor que se asigna al índice (posicion) del objetivo. Se tienen dos tipos de soluciones: en una predomina lo trabajado en los ejercicios anteriores (`posicion = 17`, acceso directo) y en otra predomina lo visual (`posicion = posicion + 5`, acceso relativo). Se espera que la mayoría de los estudiantes resuelvan el ejercicio mediante un acceso directo, que fue lo trabajado hasta el momento.

Un error esperado es la instrucción `posicion = 5`, donde el estudiante no logró los objetivos de los ejercicios anteriores, manteniendo su atención en la distancia y no en la posición (índice). Esto se evidencia en el hecho de que el estudiante no logra relacionar la posición inicial (que en este caso ya no es 0) con la posición final y usar el dato de la distancia para calcular este último.

3.2.6. Ejercicio 6

En este ejercicio se introduce uno de los conceptos más importantes de la sentencia de asignación. Utilizando una posición inicial desconocida, se obliga al estudiante a manejar el incremento de la variable basándose en su valor anterior, y no en el valor absoluto deseado. El objetivo del ejercicio radica en construir el concepto de las instrucciones de la forma `posicion = posicion + valor`, tanto sintácticamente, para lo cual una instrucción es dada, como semánticamente, para lo cual el estudiante necesita escribir una sentencia de esa forma (o cometer errores). El ejercicio se muestra en la figura 3.10.

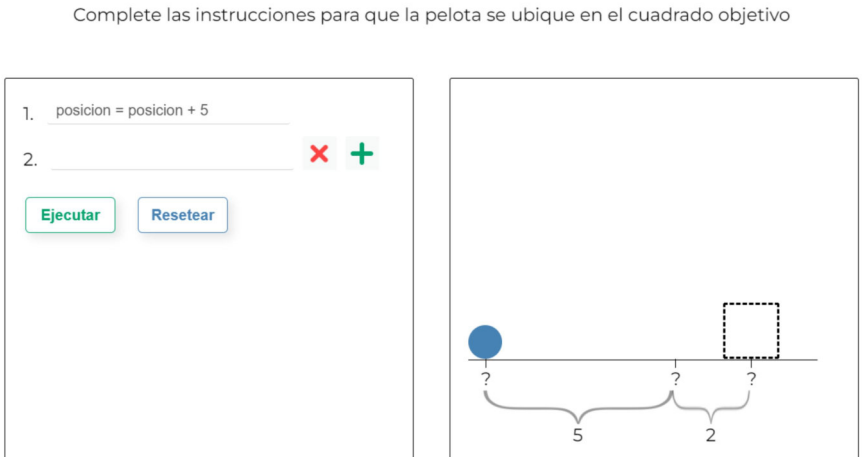

Figura 3.10: Paneles iniciales en el ejercicio 6

La solución correcta es agregar la instrucción `posicion = posicion + 2`. Se esperan errores como `posicion = 7` (no tiene en cuenta que la posición inicial no es conocida) y `posicion = posicion + 7` (no tiene en cuenta que la primera sentencia modifica el valor de la posición). Otro error posible es que el estudiante, luego de ejecutar la primera sentencia, escriba `posicion = 2`, demostrando la necesidad de volver a los ejercicios anteriores (ya que no se habrían cumplido sus objetivos).

3.2.7. Ejercicio 7

Este ejercicio plantea lo mismo que el anterior, donde la posición inicial de la pelota no es conocida y solo se conoce la distancia entre ella y el cuadrado objetivo (ver figura 3.11). La diferencia con el ejercicio 6 es que no hay ninguna instrucción previamente dada. El objetivo es que el estudiante experimente que sin importar cual sea la posición inicial, puede usar su valor y la distancia dada para calcular la posición final. Este ejercicio evidencia la construcción de conocimiento sobre el incremento de una variable, donde el valor se modifica de forma relativa a su valor anterior.

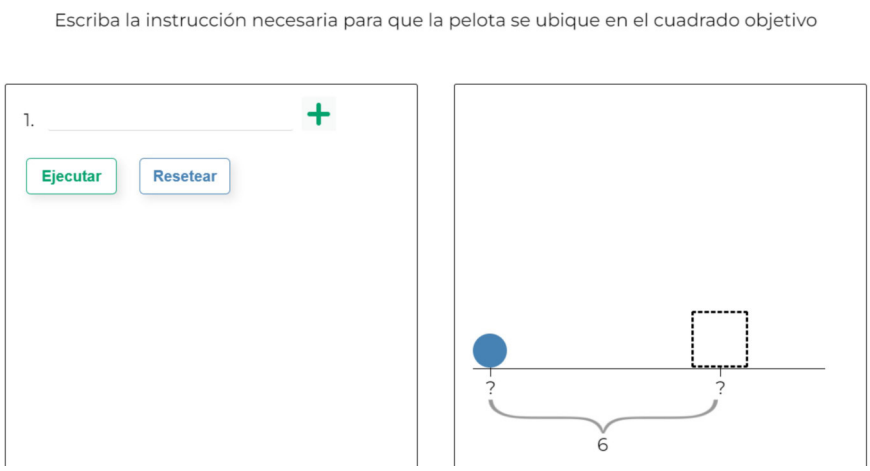

Figura 3.11: Paneles iniciales en el ejercicio 7

Se espera que el estudiante aplique lo visto en el ejercicio anterior, resolviendo lo pedido mediante la instrucción `posicion = posicion + 6`.

3.3. Ejercicios sobre el orden de las instrucciones

El objetivo de este segundo conjunto de ejercicios es introducir la importancia del orden de las sentencias, logrando que el estudiante observe que ejecutar las mismas sentencias en distinto orden produce distintos resultados. Se plantea un escenario similar al de los ejercicios anteriores, donde el objetivo es lograr que una pelota se ubique en un lugar deseado, pero siendo ahora los movimientos dentro de una matriz de dos dimensiones. La pelota se encuentra en un laberinto, y su posición es dada por dos variables numéricas: fila, representando el índice de la fila en la matriz, y columna, representando el índice de la columna.

Se espera que el estudiante ya haya culminado el primer bloque de ejercicios, habiendo experimentado con los conceptos de variable y asignación. En este conjunto de ejercicios, se introduce la necesidad de ordenar las sentencias de forma correcta mediante el uso de paredes dentro del laberinto. Para llegar de un punto del laberinto a otro, hay secuencias de sentencias (es decir, caminos dentro del laberinto) que son válidas, y secuencias que no, ya que la pelota se topa con paredes que le impiden el paso. En caso de que intente atravesar una pared, se realiza una animación sobre la pelota de forma que esta se mueva hasta la pared y "rebote", volviendo a su posición anterior. En la figura 3.12 se muestra un ejemplo de este tipo de ejercicios.

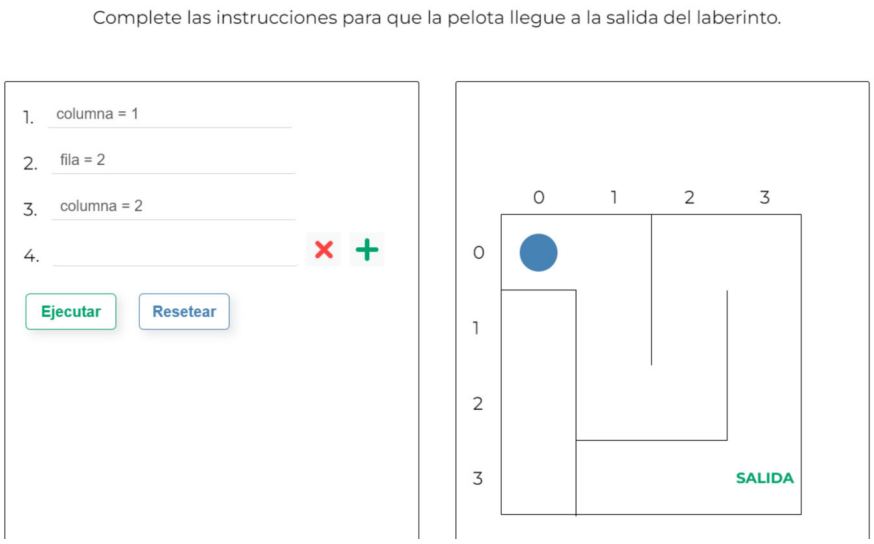

Figura 3.12: Paneles iniciales en el ejercicio 1 del conjunto 2

Todos los ejercicios se resuelven con sentencias de la forma `variable = valor` que se ejecutan de forma ordenada, donde un cambio en el orden produce otro resultado. A diferencia del conjunto de ejercicios anterior, se trabaja con dos variables independientes, y se debe considerar el orden en el que se escriben las sentencias. Es importante observar que para resolver el ejercicio correctamente, no se debe redefinir la posición de la pelota en cada paso como un par de coordenadas. Las variables tienen "memoria" (un aspecto computacional muy importante), con lo que al ejecutar las instrucciones que se muestran en el panel de la figura 3.12, la pelota se moverá por las siguientes posiciones: (0,1), (2,1), (2,2). Cada instrucción modifica una sola componente de la posición, mientras que la otra mantiene su valor actual. En cada momento, el par de coordenadas que representa la posición de la pelota está dado por los valores de las variables (fila, columna). Se observa que una única instrucción puede causar que la pelota se mueva una distancia mayor a 1. Por ejemplo, si agregamos la instrucción `fila = 0` a las instrucciones visibles en la figura 3.12, al ejecutarla la pelota se moverá desde la posición (2,2) hacia la posición (0,2).

3.3.1. Ejercicio 1

En el primer ejercicio se introducen dos variables que representan la fila y la columna correspondientes a la posición de la pelota. Además se dan tres instrucciones para que el estudiante las ejecute y visualice el resultado de la ejecución, como se muestra en la figura 3.12. El ejercicio se resuelve dando al menos otras tres instrucciones en el orden correcto (por ejemplo: `fila = 0`, `columna = 3`, y `fila = 3`).

Un error esperado es que los estudiantes no escriban las sentencias en el orden correcto, escribiendo directamente el valor de fila o columna al que quieren llegar (por ejemplo escribiendo `columna = 3` como cuarta instrucción). Se espera que la visualización los ayude a comprender la necesidad de las instrucciones intermedias.

Otro error esperado es que los estudiantes inicialmente piensen que el valor asignado a alguna de las variables es la distancia a moverse, como por ejemplo escribiendo `fila = 2` para intentar mover a la pelota 2 filas hacia arriba (o hacia abajo).

Además, si los estudiantes prueban distintos valores a modo de explorar la herramienta, como por ejemplo `fila = hola`, se mostrará un mensaje de error; si prueban "teletransportar" a la pelota a cierta ubicación (ignorando las paredes), como por ejemplo `fila = 4; columna = 4`, la pelota encontrará su avance bloqueado por una pared.

3.3.2. Ejercicio 2

Se presenta un ejercicio con similitudes (misma consigna y enunciado) y diferencias con el anterior (topología del laberinto y posición de la salida). En este caso no se le da al estudiante ninguna instrucción ya escrita, ya que se espera que pueda consolidar el conocimiento construido en el ejercicio anterior. El ejercicio puede verse en la figura 3.13.

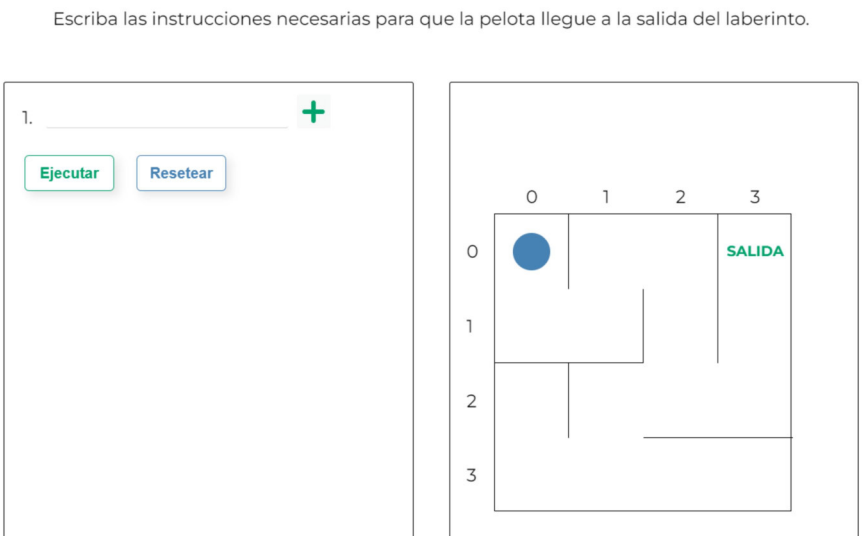

Figura 3.13: Paneles iniciales en el ejercicio 2

La solución esperada es el conjunto de instrucciones que lleven la pelota de la posición (0,0) a la posición (0, 3) sin atravesar las paredes. Una posible

solución es: `fila=1; columna=1; fila=0; columna=2; fila=2; columna=3; fila=0;`

Pueden existir otras soluciones, si el estudiante realiza pasos intermedios como "desandar sus pasos" (moviendo la pelota para atrás y para adelante), o se mueve de a menos pasos (por ejemplo de a una celda a la vez), o incluso se "encierra" en algún lugar sin salida.

Se esperan los mismos errores por parte de los estudiantes que en el ejercicio 1. El objetivo es que el estudiante trabaje con un número mayor de instrucciones y adquiera destreza en el manejo de las mismas.

3.3.3. Ejercicio 3

El ejercicio 3 presenta la misma consigna que el ejercicio anterior, variando únicamente la topología del laberinto y ubicación de la salida. En este caso, la topología del laberinto admite dos caminos posibles para llegar desde la posición inicial a la salida, siendo uno más largo (en términos de cantidad mínima de sentencias necesarias para recorrer ese camino) que el otro. El objetivo del ejercicio es consolidar los conceptos relacionados con el orden de las instrucciones. En la figura 3.14 se muestra en el ejercicio.

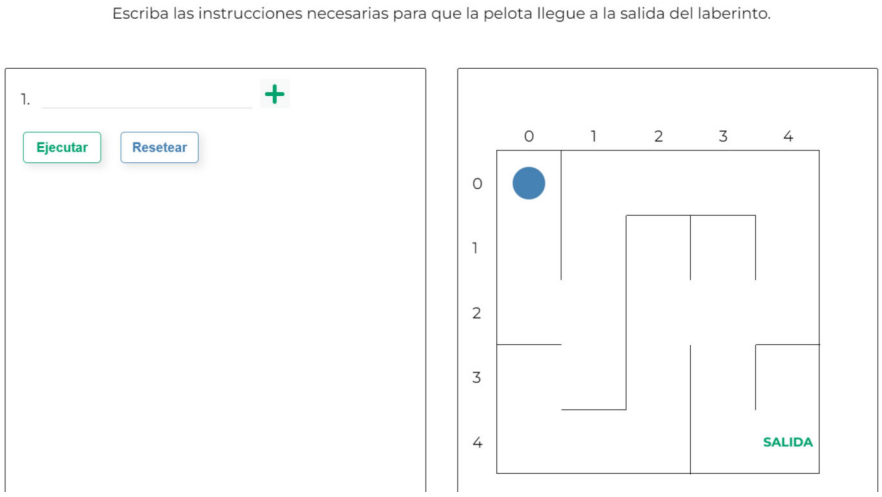

Figura 3.14: Paneles iniciales en el ejercicio 3

3.4. Ejercicios sobre intercambio de valores de variables

El objetivo de este tercer conjunto de ejercicios es aportar a la comprensión del intercambio de valores entre dos variables. Considerando que el estudiante ya realizó satisfactoriamente los dos conjuntos de ejercicios anteriores, se espera que

los primeros tres ejercicios de este conjunto no presenten mayores dificultades. Para facilitar la comprensión de los aspectos computacionales del intercambio mediante una visualización adecuada, se representa a las variables por medio de etiquetas, como se ve en la figura 3.15. A diferencia de los conjuntos anteriores, en este conjunto de ejercicios el contenido de las variables es de tipo string en lugar de numérico, introduciendo el aspecto computacional de que las variables pueden ser de diferentes tipos.

Figura 3.15: Ejemplo de ejercicio del tercer conjunto

Los ejercicios consisten en "escribir" oraciones en las etiquetas. Se resuelven mediante sentencias de la forma `variable = valor` y `variable = expresión`, donde ahora valor y expresión son secuencias de caracteres escritas entre comillas (es decir del tipo string). El objetivo detrás de tener etiquetas cumpliendo el rol de variable es poder formalizar el concepto de la "doble semántica" de una variable en relación al operador de asignación: el significado de la variable cambia según si está a la izquierda del operador de asignación (variable) o a la derecha (valor). La construcción de este concepto se evidencia de forma clara en el ejemplo concreto del intercambio de dos variables.

Al ejecutarse una instrucción que asigne un valor a una de estas etiquetas, la visualización mostrará cómo se sobreescribe el valor que anteriormente estaba en esa etiqueta (ver figura 3.16). Al ejecutarse una instrucción que asigne a una etiqueta el valor de otra, por ejemplo `etiqueta1 = etiqueta2`), la visualización mostrará cómo el valor de la etiqueta de la derecha del operador de asignación (etiqueta2 en el ejemplo) se copia hacia la etiqueta de la izquierda (etiqueta1 en el ejemplo); esta copia se visualiza mediante el valor de la etiqueta2 moviéndose hacia la etiqueta1 y reemplazando el valor que anteriormente estaba en esta etiqueta.

Figura 3.16: Ejemplo de sentencia que "escribe" texto en la etiqueta

3.4.1. Ejercicio 1

En este primer ejercicio el objetivo es que los estudiantes se familiaricen con la representación de variables de tipo string por medio de etiquetas en las cuales se escriben frases. Para esto se brinda una sentencia ya escrita, introduciendo la sintaxis de este nuevo tipo de variable (utilizando comillas para el valor), y se pide que escriban en la etiqueta2 la frase "Hola Mundo!" (ver figura 3.17).

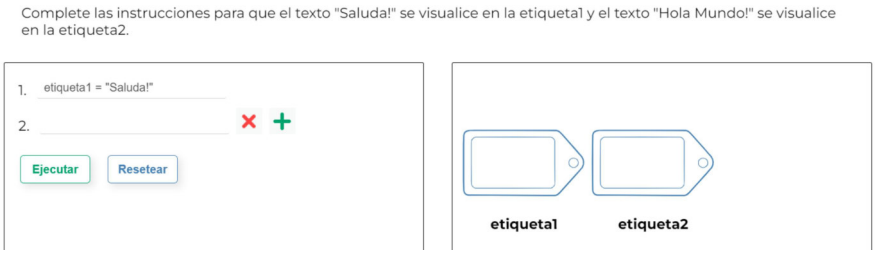

Figura 3.17: Paneles iniciales en el ejercicio 1

La solución esperada es `etiqueta2 = "Hola Mundo!"` y no se espera ningún error conceptual. Se esperan errores en el uso de la nueva sintaxis, por ejemplo omitiendo las comillas; en este caso se muestra un mensaje de error como el que puede verse en la figura 3.18.

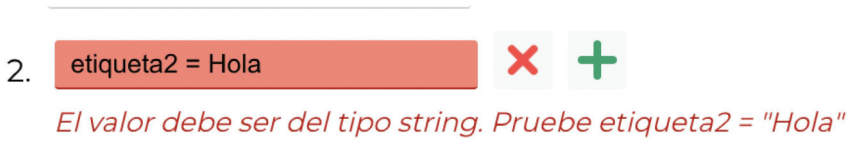

Figura 3.18: Error al omitir las comillas

3.4.2. Ejercicio 2

Al igual que en el ejercicio anterior, el siguiente tiene el mismo objetivo y no se espera ningún error en particular (ver figura 3.19). El estudiante debe escribir una sentencia inicial y agregar otra, sin que se le brinde una instrucción de ejemplo.

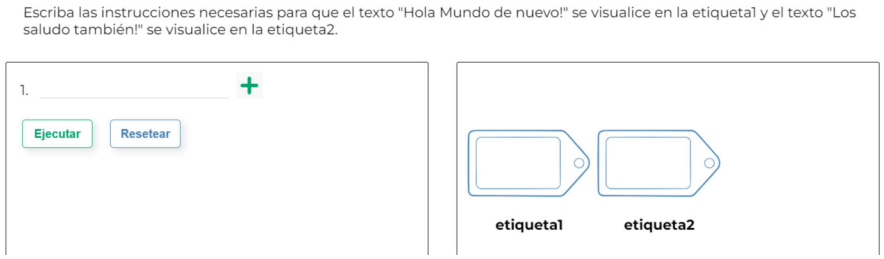

Figura 3.19: Paneles iniciales en el ejercicio 2

3.4.3. Ejercicio 3

El objetivo de este ejercicio es preparar al estudiante para la introducción de sentencias de la forma `variable = variable` en el siguiente ejercicio. Para esto se le pide al estudiante que escriba las instrucciones necesarias para que el valor de una de las etiquetas sea igual a la otra. El ejercicio se muestra en la figura 3.20.

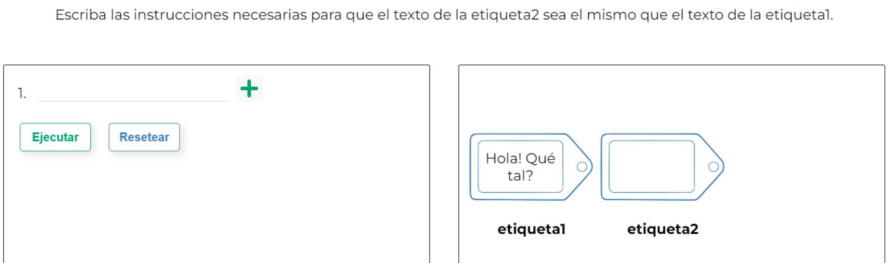

Figura 3.20: Paneles iniciales en el ejercicio 3

Se consideran dos soluciones posibles: `etiqueta2 = "Hola! Qué tal?"` y `etiqueta2 = etiqueta1`, de las cuales se espera que la primera opción sea la más favorecida por el estudiante.

3.4.4. Ejercicio 4

Este ejercicio tiene como objetivo que el estudiante experimente el hecho de que una variable es un espacio donde hay un valor, y una vez que se modifica

no se puede obtener el valor anterior. La idea de este ejercicio es mostrar qué ocurre cuando a una etiqueta se la "sobrescribe" con un nuevo valor.

Se provee una instrucción ya dada con el formato `variable1 = variable2` para poder mostrar la visualización de dicha instrucción (donde el valor que está en variable2 se copia a variable1, sobrescribiendo el valor previo de esta), dos etiquetas que ya cuentan con valores y una tercera etiqueta vacía. El ejercicio puede verse en la figura 3.21.

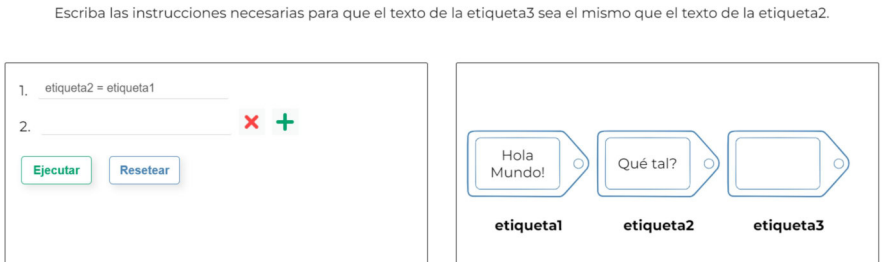

Figura 3.21: Paneles iniciales en el ejercicio 4

Hay tres soluciones posibles (`etiqueta3 = "Hola Mundo!"`, `etiqueta3 = etiqueta2` o `etiqueta3 = etiqueta1`), y se espera que la primera opción sea la más favorecida por el estudiante.

Un error esperado es que intenten la instrucción `etiqueta3 = "Qué tal?"`. En ese caso, al ejecutar las instrucciones verán cómo el valor que inicialmente estaba en la etiqueta2 ("Qué tal?") es sobrescrito al ejecutarse la primera sentencia, entendiendo así que deben lograr que el valor de etiqueta3 sea "Hola Mundo!".

Al ver esto, puede suceder que el estudiante simplemente agregue la instrucción `etiqueta3 = "Hola Mundo!"` sin razonar sobre por qué esta es la solución correcta. Este problema se abordará a partir del siguiente ejercicio.

3.4.5. Ejercicio 5

Para abordar el problema planteado en el ejercicio anterior, se introducen en este ejercicio etiquetas "tapadas", donde el estudiante no puede ver lo que está escrito. La consigna es la misma que en el ejercicio anterior, donde deben dejar dos variables con el mismo valor ("copiando" de una a otra), pero la diferencia radica en que el valor inicial de la primera variable es desconocido (está "tapado").

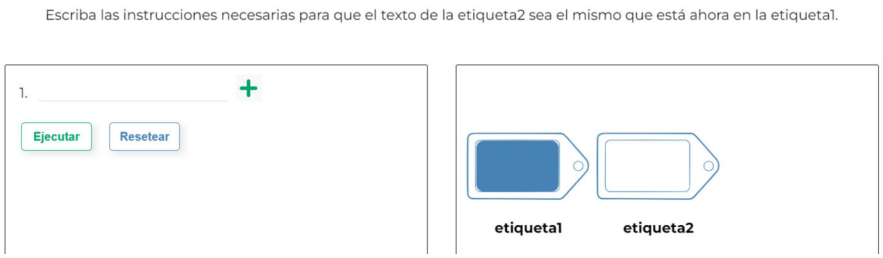

Figura 3.22: Paneles iniciales en el ejercicio 5

A diferencia del ejercicio 3, solamente hay una solución correcta (`etiqueta2 = etiqueta1`). Para evitar que el estudiante simplemente se fije cual es el texto luego de una ejecución y escriba `etiqueta2 = "el texto que vio"` (intentando lo mismo que en el ejercicio anterior), el texto "escrito" en la etiqueta1 cambia después de cada ejecución (se elige un texto aleatoriamente de entre 10 opciones).

Este ejercicio es clave para construir el conocimiento sobre la "doble semántica" de la variable, es decir la diferencia entre utilizar la variable a la izquierda del operador de asignación (donde el rol de la variable es de contenedor) y utilizarla a la derecha del mismo (donde el rol de la variable es de contenido). El ejercicio obliga al estudiante a enfrentarse a este obstáculo, y el objetivo de la visualización es ayudar a superarlo y construir este conocimiento.

3.4.6. Ejercicio 6

En este ejercicio, la consigna es la misma que en el anterior, pero se agrega una tercera variable (ver figura 3.23).

Figura 3.23: Paneles iniciales en el ejercicio 6

3.4.7. Ejercicio 7

Se plantea lo mismo que en el ejercicio anterior, pero teniendo en cuenta la diferencia de que no se ve lo que está en las etiquetas 1 y 2 (ver figura 3.24). Este ejercicio es el paso previo al ejercicio de intercambio de variables, con lo que se

establece el mismo escenario (tres etiquetas, dos "tapadas"), pero pidiendo solo que se igualen dos etiquetas.

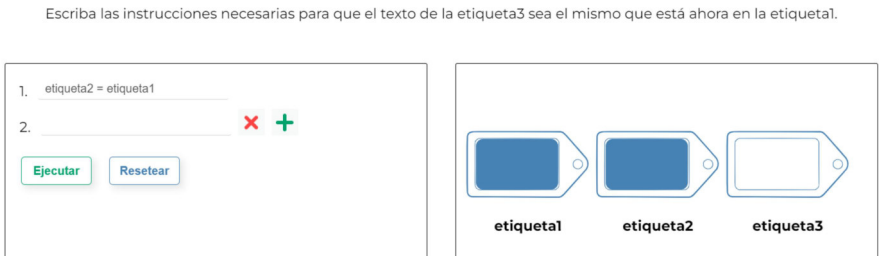

Figura 3.24: Paneles iniciales en el ejercicio 7

De la misma manera que antes, hay dos soluciones posibles (`etiqueta3 = etiqueta1` y `etiqueta3 = etiqueta2`) ya que los textos en dichas etiquetas cambian con cada ejecución de las instrucciones.

3.4.8. Ejercicio 8

Con el último ejercicio del conjunto se plantea el problema típico de intercambiar los valores entre dos variables (ver figura 3.25).

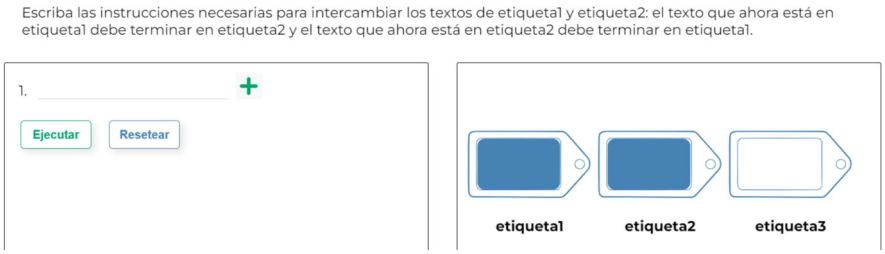

Figura 3.25: Paneles iniciales en el ejercicio 8

Las soluciones posibles requieren utilizar la tercera etiqueta como variable auxiliar para temporalmente "recordar" uno de los valores. El error esperado es que el estudiante no se de cuenta de la necesidad de usar esta tercera variable, escribiendo solamente dos instrucciones (`etiqueta1 = etiqueta2; etiqueta2 = etiqueta1`, o viceversa).

3.5. Ejercicios de introducción a la formalización

En este conjunto de ejercicios se utilizan símbolos abstractos (x, y, z, etc) para los nombres de las variables, a modo de ayudar al estudiante a despegarse

de los casos concretos (caja, laberinto, etiqueta), como un primer paso hacia la formalización de la sentencia de asignación. Se trabaja sobre ejercicios similares a los conjuntos de ejercicios anteriores.

En todos los ejercicios se representa cada variable mediante un cuadrado con nombre, donde dentro se muestra el valor que tiene la variable. El color del cuadrado se corresponde al tipo de la variable (ver figura 3.26).

3.5.1. Ejercicio 1

Como en los casos anteriores, se comienza con un ejercicio que ya tiene una instrucción provista, a modo de introducir la sintaxis de las instrucciones. En este caso el ejercicio es simple, pidiendo únicamente la asignación de valores a variables. Se introduce implícitamente el concepto de tipo (teniendo dos variables numéricas y una variable de tipo string), denotando cada tipo de variable con un color distinto (azul para las variables numéricas y violeta para los strings).

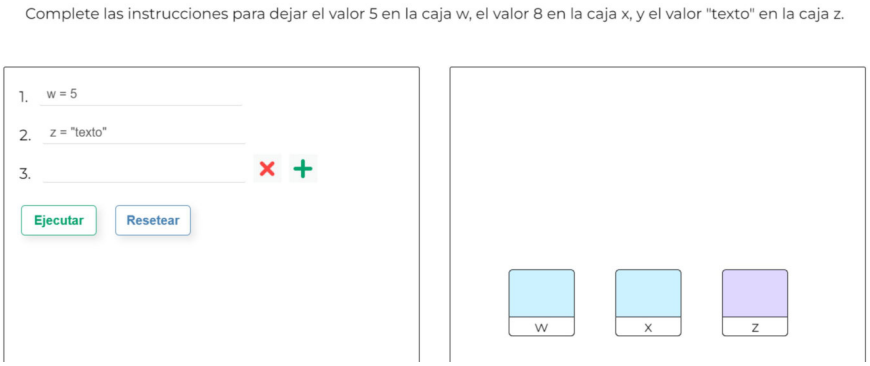

Figura 3.26: Paneles iniciales en el ejercicio 1

La solución correcta es la asignación `x = 8`. Si los estudiantes intentan asignar un número a una variable de tipo string, se mostrará un mensaje de error que indica que el tipo que se le está asignando no es válido, como se muestra en la figura 3.27. Lo mismo sucede si intentan asignar un string a una variable numérica.

Figura 3.27: Mensaje de error de tipo

3.5.2. Ejercicio 2

Este ejercicio plantea una consigna igual al del anterior, pero no se da ninguna instrucción ya provista. Se tiene una variable numérica y otra de tipo string, manteniendo la relación de los colores de las variables con su tipo. El ejercicio se muestra en la figura 3.28.

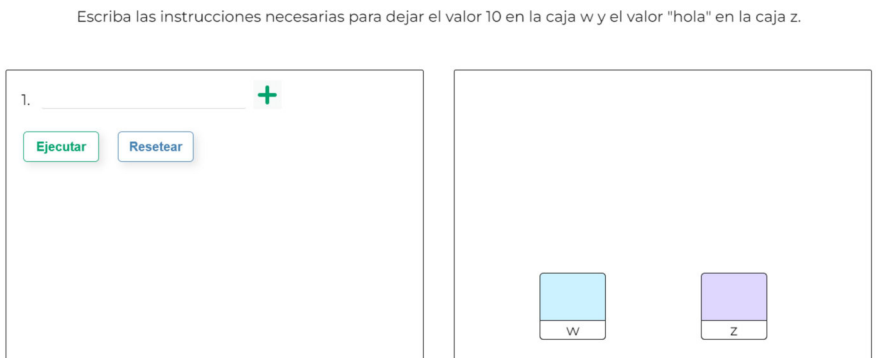

Figura 3.28: Paneles iniciales en el ejercicio 2

La solución correcta es la secuencia de asignaciones `w = 10` y `z = "hola"` (en cualquier orden).

3.5.3. Ejercicio 3

Este ejercicio tiene como objetivo trabajar con la idea del incremento de una variable, introducida en el primer conjunto de ejercicios. En este caso no se muestran los valores iniciales de x y w al estudiante; solo al ejecutar se revelan los valores (y cambian de ejecución en ejecución). Se pide a los estudiantes que incrementen el valor de dos variables, con una instrucción ya provista, a modo de recordar lo trabajado en el primer conjunto de ejercicios (ver 3.29).

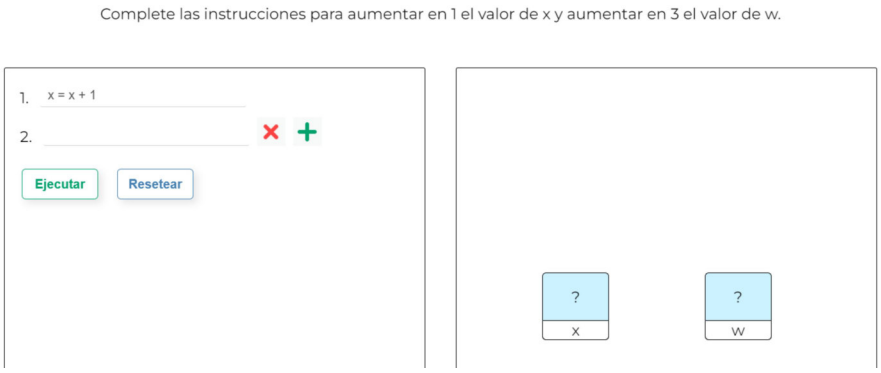

Figura 3.29: Paneles iniciales en el ejercicio 3

Se espera que los estudiantes puedan generalizar el conocimiento construido en el primer conjunto de ejercicios, donde incrementaron la posición de la pelota, para poder modificar el valor de las variables de forma de cumplir con lo pedido.

3.5.4. Ejercicio 4

Este ejercicio formaliza algunos de los conceptos vistos en los ejercicios del tercer conjunto. La idea de este ejercicio es trabajar sobre las asignaciones de la forma `variable = variable`. En este caso, el valor inicial de x es desconocido, y tienen que dejar en la variable z el mismo valor que en x (ver 3.30).

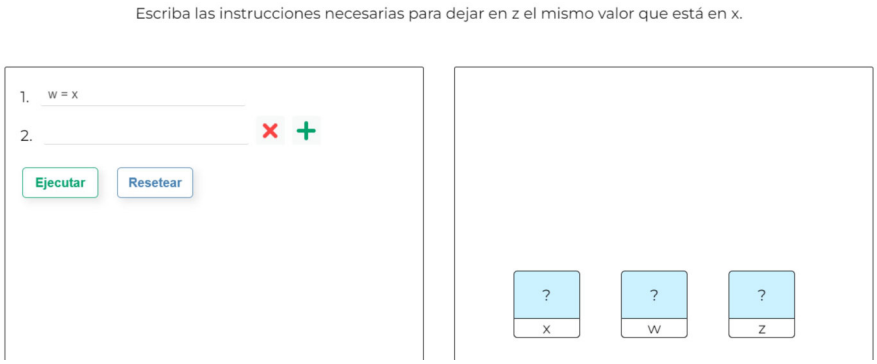

Figura 3.30: Paneles iniciales en el ejercicio 4

Al ejecutar la sentencia dada (`w = x`), verán el valor de x copiarse en w. El valor de x cambiará cada vez que se ejecute, para evitar que los estudiantes intenten realizar el ejercicio con la asignación `z = valor_de_x` (es decir, con una sentencia de la forma `variable = valor`). Este es uno de los posibles errores; luego de un par de ejecuciones pueden darse cuenta que el valor de x cambia (es

desconocido) y que esa estrategia no les funcionará. Se espera que rápidamente se adapten a la solución esperada, habiendo ya resuelto los ejercicios de las etiquetas.

3.5.5. Ejercicio 5

Este ejercicio es análogo al último de las etiquetas, donde se plantea el intercambio de dos variables. Se provee una tercera variable auxiliar (necesaria para el intercambio), pero no se le dice al estudiante su propósito. El ejercicio se muestra en la figura 3.31.

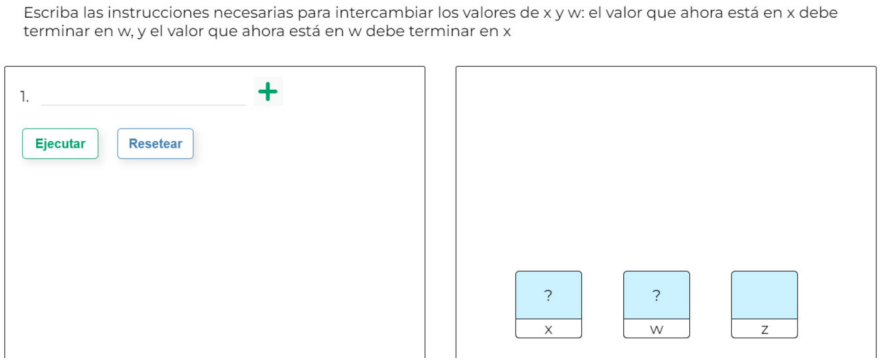

Figura 3.31: Paneles iniciales en el ejercicio 5

Se esperan errores del mismo tipo que en el último ejercicio del tercer conjunto, como por ejemplo `x = w; w = x`. Se espera que mediante la ejecución de las instrucciones, el estudiante pueda ver su error, y que pueda generalizar lo visto en el ejercicio del tercer conjunto para resolver correctamente el actual.

Capítulo 4

Diseño de los ejercicios

En este capítulo se describen las ideas y justificaciones detrás del diseño de cada conjunto de ejercicios, relacionándolas con el marco teórico presentado en el capítulo 2.

4.1. Pautas generales del diseño

El objetivo general de los ejercicios planteados por la herramienta es que los estudiantes construyan conocimiento sobre las variables y la sentencia de asignación. El objetivo concreto es que los estudiantes resuelvan exitosamente la mayoría de los ejercicios, evidenciando avances en el proceso de construcción de conocimiento sobre variables computacionales y asignación. En este proceso entendemos que el mayor nivel se alcanza resolviendo el ejercicio de intercambio entre dos variables.

Se asume que los estudiantes no tienen ningún conocimiento previo sobre variables computacionales, pero que sí lo tienen sobre variables matemáticas. Es importante notar que estas son inherentemente distintas a las variables computacionales, ya que representan un valor o elemento desconocido o arbitrario pero que una vez definido siempre es el mismo, mientras que las variables computacionales representan un espacio en la memoria de la computadora, cuyo valor puede ir cambiando. Esta diferencia determina diferencias semánticas tanto en los símbolos utilizados para la operación como en el rol del nombre de las variables.

En efecto, en matemática el signo de `=` se usa implícitamente con dos semánticas distintas. Por un lado, se utiliza para significar la igualdad entre una variable y su valor, por ejemplo en la frase `Si x = 0 entonces ...`, es decir en una expresión booleana que evalúa a verdadero o falso. Por otro lado, también se utiliza para darle un valor a una variable matemática, por ejemplo en la frase `Sea x = 3, hallar P(x) ...`, donde se tiene una especie de asignación, pero sin memoria: el valor de x no cambiará. A diferencia de la matemática, en computación el símbolo de asignación está relacionado con un *cambio*. Varios autores han comprobado que la influencia de la noción matemática de variable actúa como idea

preconcebida en el caso de la asignación computacional, por ejemplo Dehnadi y Bornat (2008) y da Rosa (2016).

Por otro lado, se observan dos semánticas distintas para el nombre de una variable en la sentencia de asignación. Cuando el nombre de la variable se encuentra a la izquierda del operador de asignación, está referenciando el espacio en memoria representado por la variable. En cambio, cuando el nombre de la variable se encuentra a la derecha del operador de asignación, está referenciando el valor que ocupará el espacio en memoria representado por la variable. Esta doble semántica de los nombres de las variables en la sentencia de asignación es frecuentemente dejada implícita, y resulta un impedimento a la hora de realizar ejercicios clásicos como el de intercambio de dos variables.

A continuación se detallan los objetivos particulares de cada conjunto de ejercicios, y qué decisiones se tomaron en cuanto al diseño de los ejercicios y su visualización para lograr los objetivos propuestos.

4.2. Primer conjunto de ejercicios

El primer conjunto de ejercicios busca introducir la sentencia de asignación, construyendo conocimiento sobre asignaciones simples (de la forma `variable = valor`) y sobre incrementos sobre la variable (asignaciones de la forma `variable = variable + valor`). Se parte del conocimiento ya construido por los estudiantes sobre distancias en una recta numérica, haciendo que la posición de una pelota tome el rol de variable. Se plantea una asociación entre el valor de la variable posición y la posición (índice) de una pelota en una recta.

Los objetivos concretos de este conjunto de ejercicios son dos. En primer lugar, se espera que el estudiante pueda realizar asignaciones simples que no contengan variables a la derecha del operador de asignación. En segundo lugar, se busca que los estudiantes comiencen a construir conocimiento sobre la "doble semántica" de una variable en relación al operador de asignación, previamente mencionada en 4.1.

El conjunto comienza continuando la línea del ejercicio propuesto en el tutorial, de manera tal que los estudiantes se familiaricen con la idea de la herramienta: escriben instrucciones en el panel de la izquierda que, al ejecutarlas secuencialmente, muestran su efecto en el panel de la derecha. Para esto están los primeros dos ejercicios.

A partir del tercer ejercicio se introduce un cambio en la visualización. Ya no se muestra mediante marcas cada punto en la recta, si no que se muestran distancias entre puntos. El objetivo de esta modificación es fomentar el cambio del razonamiento sobre la acción a realizar, pasando de un cambio absoluto en la posición a un cambio relativo a su posición actual.

Si bien el cuarto ejercicio puede resolverse realizando la asignación de forma absoluta (`posicion = 7`), comienza a introducirse la idea de posiciones relativas representadas con distancias sobre la recta. El ejercicio 5 introduce una posición inicial diferente (12 en vez de 0), como paso previo a una posición inicial desconocida como se plantea en los ejercicios 6 y 7.

Con estas dos ideas en mente, el ejercicio 6 es igual al ejercicio 4 pero desconociendo todas las posiciones: la inicial, la intermedia y la final. Por lo tanto no hay manera de inferir directamente un cambio de la posición de forma absoluta. Recordemos la figura 2.4; nuestro objetivo es llevar a los estudiantes desde la periferia newP hacia los centros newC y newC'. Al no poder inferir directamente la posición absoluta, los estudiantes son desafiados a encontrar una transformación de sus acciones (C) sobre la posición de la pelota para que el autómata sepa qué hacer (newC). Ya no puede el estudiante hacer el trabajo de calcular la posición final por el autómata; debe comprender que la posición no es un lugar absoluto en la recta (C') sino que es el valor actual de donde se encuentra la pelota (newC'). El ejercicio debe resolverse utilizando una asignación de la forma `variable = variable + valor`. Logrando el objetivo propuesto, se construye conocimiento sobre las propiedades intrínsecas de las variables (newC'); en este caso la capacidad de guardar datos. El último ejercicio del conjunto plantea la misma problemática sin una instrucción dada para finalizar el conjunto de ejercicios.

Al finalizar los ejercicios de este conjunto, se espera que los estudiantes hayan comenzado a construir conocimiento instrumental sobre las diferencias entre el rol que juega una variable a la izquierda del operador de asignación (rol de "espacio en memoria") y a la derecha del operador de asignación (rol de "valor").

4.3. Segundo conjunto de ejercicios

El segundo conjunto de ejercicios busca que los estudiantes construyan conocimiento de conceptos clave sobre la importancia del orden de ejecución de las instrucciones. Este conocimiento será usado en el objetivo final del tercer conjunto de ejercicios: resolver el problema de intercambio de dos variables.

Se mantiene a la entidad de la pelota igual que en el conjunto anterior, pero esta se traslada a una matriz bidimensional, aumentando la complejidad del ejercicio, ya que utiliza dos variables (una para representar la fila actual de la pelota, y otra para representar la columna). El objetivo de este cambio es mostrar la importancia del orden de las instrucciones y que el mismo problema puede tener varias soluciones, representadas por distintos caminos en el laberinto.

El primer ejercicio tiene 3 instrucciones dadas para mostrar el uso de las dos variables (fila y columna) y solamente tiene un camino posible para resolverlo. Lo mismo ocurre con el segundo ejercicio. Estos ejercicios buscan que los estudiantes razonen correctamente cuando deben modificar una u otra variable, evitando chocar contra las paredes del laberinto.

El último ejercicio admite dos caminos posibles para resolverlo, pudiendo los estudiantes decidir cuál camino usar. En este conjunto se plantea una cantidad menor de ejercicios que en el conjunto anterior, dado que el estudiante ya cuenta con cierto manejo de la herramienta. Por otro lado creemos que son ejercicios suficientes para ayudar a la comprensión del orden de las instrucciones.

4.4. Tercer conjunto de ejercicios

El tercer conjunto de ejercicios busca que los estudiantes, a través del conocimiento construido en los dos conjuntos anteriores, puedan resolver el problema de intercambio de dos variables. A su vez, evidencia por primera vez el concepto de tipo en las variables, ya que a diferencia de los conjuntos anteriores, las variables almacenan valores de tipo string.

Al igual que en el primer conjunto de ejercicios, se comienza con ejercicios introductorios que repasan el conocimiento construido anteriormente (del 1 al 3 inclusive) buscando la familiarización con la visualización y con la sintaxis de la asignación de strings (utilizando las comillas).

El ejercicio 4 es el primer ejercicio del conjunto donde, a través de la visualización, se busca reducir la distancia entre cómo el estudiante razona sobre lo que quiere hacer y cómo le dice al autómata que lo haga (recordemos la figura 2.4). La ejecución de la instrucción ya provista (ver figura 3.21) evidencia la diferencia entre las sentencias vistas hasta el momento (asignaciones de la forma `variable = valor`) y la nueva sentencia (`variable1 = variable2`). La visualización muestra como el valor "sale" de la etiqueta1 (es copiado) y "sobreescribe" el valor que está en la etiqueta2.

El diseño de este ejercicio y sus animaciones correspondientes se realizó con dos enfoques distintos. Por un lado, la visualización cuando la asignación es de la forma `variable = valor` es tal que se ve cómo se "borra" el valor anterior y se escribe el nuevo. Por otro lado, cuando la asignación es de la forma `variable1 = variable2`, la visualización muestra cómo se "lee" el valor de la variable2 y es ese mismo valor que va hacia la variable1 y se "escribe" en ella. De esta forma se establece una analogía entre las etiquetas y las variables computacionales, de manera que puedan asociar las propiedades intrínsecas de las etiquetas (C') a las propiedades intrínsecas de las variables (newC').

Observemos que los estudiantes pueden resolver el ejercicio 4 asignando directamente el valor a la variable (`etiqueta2 = "Hola Mundo!"`). Para obligar al estudiante a realizar asignaciones de la forma `variable = variable` se introduce a partir del ejercicio 5 una modificación a la visualización: algunas etiquetas se encuentran "tapadas", siendo su valor desconocido. De esta forma, se motiva al estudiante a pasar del plano de "cómo lo hago yo" (un humano que puede leer los valores y simplemente intercambiarlos en su mente) al plano de "cómo lo hace un autómata" (que no tiene noción semántica del significado de "intercambio"). Al enfrentarse al problema del intercambio de variables, deben lograr comprender que al no poder "leer" lo que está escrito en la etiqueta, no se puede resolver el problema sin "escribirlo" en otro lado. Se puede ver la necesidad del uso de una variable auxiliar en el problema de intercambio de variables como una variable para que el autómata pueda "recordar momentáneamente" un valor, de la misma manera que si una persona lee dos etiquetas, puede recordar qué está escrito en ambas e intercambiar los mensajes correspondientes.

4.5. Cuarto conjunto de ejercicios

El cuarto conjunto de ejercicios busca que los estudiantes consoliden la construcción de conocimiento realizado en los anteriores ejercicios sin una visualización explícita. En este caso, la visualización se asemeja más a la realidad: una variable es un espacio en la memoria de la computadora (en lugar de la dirección de memoria se utiliza el nombre de la variable para evitar complejidades innecesarias en esta etapa) que contiene un valor (visible en la visualización como el valor dentro del lugar). También queda explícito, mediante un color, que la variable tiene un tipo asociado.

Los primeros dos ejercicios son introductorios a la nueva visualización, pidiendo en uno y el otro asignaciones de distinto tipo: en el primero solamente se debe asignar un valor numérico y en el segundo uno numérico y otro de tipo string.

El tercer ejercicio, que pide incrementar el valor de una variable, se resuelve mediante una asignación de la forma `variable = variable + valor` al igual que en el primer conjunto de ejercicios. A diferencia de los conjuntos de ejercicios anteriores, no hay un razonamiento análogo a realizar por parte del estudiante (como llevar la pelota al objetivo o que en la etiqueta esté escrito "Hola") sobre una realidad planteada, si no que la visualización es en sí el estado actual en el que se encuentran las variables al momento de la ejecución. Ocurre lo mismo con el cuarto ejercicio, que plantea únicamente asignar a una variable el valor que tiene otra sin poder verlo, basándose en lo realizado en el tercer conjunto de ejercicios.

Por último, el quinto ejercicio es el problema de intercambio de valor entre dos variables. Partiendo del conocimiento construido en el tercer conjunto de ejercicios, se espera que los estudiantes hayan progresado en el pasaje a la etapa *inter*, siendo este ejercicio un primer acercamiento a la etapa *trans* de formalización (ver 2.2 para más detalle sobre las etapas). Se espera que el estudiante sea capaz de resolver el problema transponiendo lo aprendido en el plano de los objetos (las etiquetas) al plano de los conceptos (las variables). Al lograr resolver el problema, el estudiante está unificando el conocimiento construido sobre el orden de las instrucciones, las propiedades intrínsecas de las variables y los efectos de las acciones que se realizan sobre estas.

Capítulo 5

Diseño e implementación

En este capítulo se detallan las decisiones del diseño e implementación de la herramienta y sus distintos componentes así como su vinculación con los requerimientos del proyecto.

5.1. Diseño

5.1.1. Visualización

Como se explicó en la sección 1.1, la herramienta debe permitir la visualización de los efectos de la ejecución de las sentencias. Para esto, se divide la pantalla en dos paneles: el panel de instrucciones a la izquierda y el panel de visualización a la derecha. Al apretar el botón de ejecutar, las sentencias se irán ejecutando secuencialmente, causando efectos en la visualización. Es importante que el estudiante pueda ver el efecto de cada sentencia ejecutada; si bien un programa puede verse como una secuencia de sentencias, es necesario entender cada sentencia por si misma, y no solo concentrarse en el estado final luego de ejecutar todo el programa. Para esto, se dejan unos segundos de tiempo entre la ejecución de una sentencia y la siguiente, de forma que el estudiante pueda ver cada cambio individualmente (y no solo el estado final).

Además, la herramienta debe ser capaz de mostrar mensajes de error frente a entradas inesperadas. Se contemplaron diversas entradas incorrectas y se diseñaron mensajes de error para mostrar en cada caso. Un ejemplo de mensaje de error puede verse en la figura 5.1, donde la entrada es simplemente `5`, en vez de `posicion = 5`.

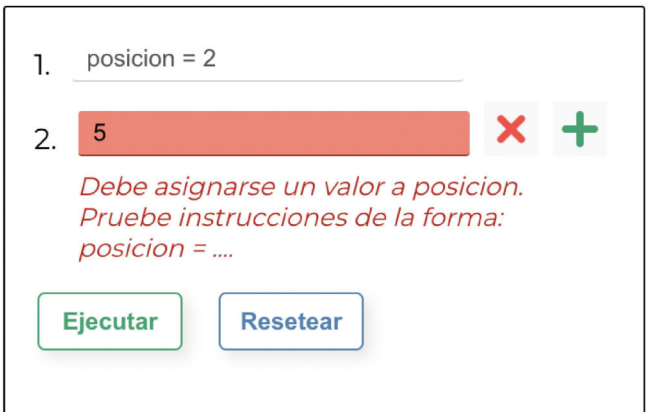

Figura 5.1: Ejemplo de mensaje de error

5.1.2. Lenguaje de las sentencias

Se definió un lenguaje sencillo para las sentencias que deben ser escritas por los estudiantes. Se optó por simplificar los aspectos de sintaxis, por ejemplo no exigiendo el uso de punto y coma luego de una sentencia, para que los estudiantes puedan concentrarse en la semántica asociada a cada instrucción.

Se decidió que el lenguaje sea fuertemente tipado, ya que entendemos que el tipo de una variable es una característica fundamental de la misma y que es necesario construir este conocimiento en una etapa inicial del aprendizaje. En el lenguaje definido por la herramienta, el tipo inicial de una variable no cambia, y una variable no puede ser asignada a otra de tipo distinto. El tipo de cada variable es dado por el contexto del ejercicio: en los primeros dos conjuntos todas las variables son numéricas, en el tercer conjunto todas son de tipo string, y en el último conjunto se tienen variables de ambos tipos (cada tipo señalizado con un color). En la figura 5.2 podemos ver un ejemplo de error de tipado.

Figura 5.2: Mensaje de error de tipado

Las sentencias válidas permitidas por la herramienta son las siguientes :

- Sentencias de la forma `variable = valor`, donde `valor` puede ser un número o un string. Para variables de tipo string, los valores deben ser representados entre comillas dobles (ej `etiqueta1 = "hola"`).
- Sentencias de la forma `variable1 = variable2`, donde las variables disponibles están pre-determinadas por cada ejercicio. Ambas variables deben

ser del mismo tipo para que esta sentencia sea válida.

- Sentencias de la forma `variable1 = variable2 <op> valor` o `variable1 = valor <op> variable2`, donde `variable1` y `variable2` son variables numéricas y el operador es + o -.

5.1.3. Arquitectura

Uno de los requerimientos de la solución es que pueda ser usada por estudiantes tanto dentro del salón de clase como fuera de este; la herramienta está pensada como complemento a instancias formales de aprendizaje, pudiendo el docente utilizarla como parte de la clase o como tarea domiciliaria. Por esta razón se optó por implementar la herramienta (y los ejercicios propuestos en ella) como una página web, que permite que cualquier persona con acceso a una computadora y a Internet pueda acceder a la herramienta sin necesidad de instalar ningún programa.

Se diseñó una arquitectura únicamente con tecnologías del lado del cliente, sin tener un servidor de backend o una base de datos. Esta decisión simplificó significativamente la implementación, permitiendo concentrar los esfuerzos en el diseño de ejercicios que estuvieran didácticamente alineados con los objetivos del proyecto. Como se tiene solo el cliente web, el diagrama de despliegue de la aplicación representado en la figura 5.3 es extremadamente simple. La página web puede ser accedida en este enlace.

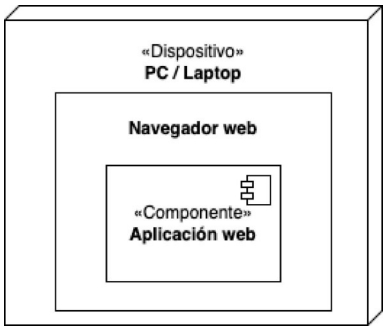

Figura 5.3: Diagrama de despliegue

El cliente web se implementó utilizando el lenguaje de programación JavaScript (*JavaScript*, s.f.) y la librería React (*React*, s.f.), que permite crear interfaces de usuario. Esta es una librería basada en componentes altamente reutilizables que interactúan entre sí para formar estructuras complejas. Los componentes principales de la aplicación se describen a continuación, y un diagrama representativo se muestra en la figura 5.4. Para mayor visibilidad, se omiten los componentes de los conjuntos 2 y 3, siendo estos análogos a los representados. El objetivo detrás de este diseño es que la herramienta sea extensible, siendo fácil reutilizar los componentes existentes para crear nuevos conjuntos de ejercicios. Lo único que se debería crear desde cero para crear un nuevo conjunto

de ejercicios es la visualización del mismo y su especificación en formato JSON, ya que esto es específico a cada conjunto (ver 5.2.1 para más detalle).

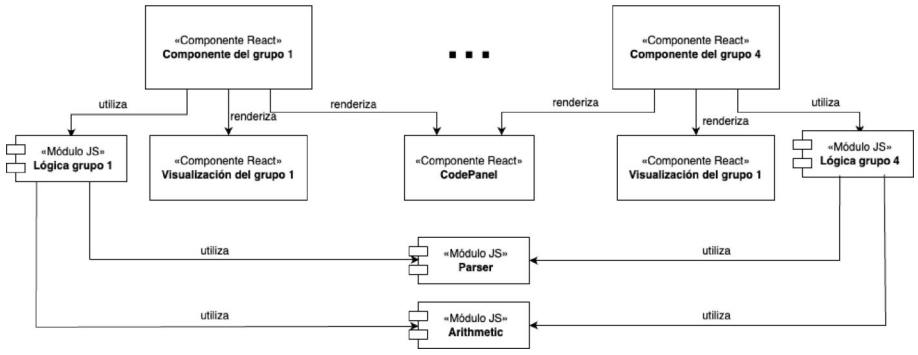

Figura 5.4: Diagrama de componentes

Se definió un componente reutilizable llamado *CodePanel*, encargado de la interfaz y lógica del panel de código. La responsabilidad de este componente es renderizar el panel de código y mostrar las animaciones en este, por ejemplo marcando en verde la linea que está ejecutando actualmente. Además, el componente tiene como argumento una función *callback* a la cual llama al ejecutar cada sentencia. De esta forma, cada conjunto de ejercicios puede definir la lógica y validaciones necesarias, abstrayendo al *CodePanel* de esta responsabilidad.

Se definieron dos módulos JavaScript que son utilizados en todos los conjuntos de ejercicios: el módulo *Parser* encargado del *parsing* de las sentencias ingresadas por el usuario y el módulo *Arithmetic* encargado de calcular los cambios en los valores de las variables luego de ejecutar una sentencia.

El módulo *Parser* cuenta con una función *parseInput* que tiene como argumentos una sentencia ingresada por el usuario y una lista de variables (cada una con su nombre y su tipo). Utilizando expresiones regulares, la función determina si la sentencia ingresada es sintácticamente válida, además de realizar chequeos de tipos. En caso de instrucciones inválidas o con errores de tipado, la función devuelve un mensaje de error que es luego mostrado al usuario; en caso de sentencias válidas, devuelve un diccionario con el tipo de la sentencia (en 5.1.2 se describen los tres tipos posibles) y los grupos reconocidos por la expresión regular que reconoció la sentencia. Cada conjunto de ejercicios utiliza esta función *parseInput*, pasándole como parámetro cada sentencia ingresada y las variables permitidas para el conjunto (por ejemplo, para el primero se tiene una única variable posicion, para el segundo se tienen dos variables, fila y columna, etc). El *parser* se describe con más detalle en 5.1.4.

El módulo de *Arithmetic* define una función *calculateNewValue* cuyos argumentos son la salida de la función *parseInput* y un diccionario de información de variables, donde las claves del diccionario son los nombres de las variables, y los valores son diccionarios que contienen el tipo de la variable y su valor actual.

Esta función calcula el nuevo valor de la variable actualizada por la sentencia[1], devolviendo un diccionario que contiene la variable actualizada y su nuevo valor. Por ejemplo, si la sentencia ingresada fue `x = w + 2` y el valor actual de `w` es 5, la función devuelve un diccionario donde la variable actualizada es `x` y su valor es 7.

Se define un componente React de visualización para cada conjunto de ejercicios, encargado de la interfaz del panel de visualización de ese conjunto. Estos componentes no son reutilizables, ya que son específicos a cada conjunto de ejercicios, y la visualización varía significativamente de un conjunto a otro. También se tiene un módulo de lógica para cada conjunto, encargado de llamar a los módulos *Parser* y *Arithmetic* con los valores correctos, y de utilizar la salida de la función de *parsing* como entrada de la función *calculateNewValue*.

Por último, cada conjunto de ejercicios tiene su propio componente, responsable de mantener la lógica y el estado asociado al conjunto. Estos componentes obtienen el id del ejercicio actual de la URL actual (por ejemplo en la URL `/primer-grupo/2`, el id del ejercicio es 2) y utilizan ese id para obtener la información específica del ejercicio (ver sección 5.2.1 para más detalle). Estos componentes son los responsables de llamar a los módulos de lógica y de utilizar el componente de *CodePanel* (genérico) y el de visualización (específico a cada conjunto).

5.1.4. Parser

Se diseñó un módulo *Parser* responsable de realizar el *parsing* de las sentencias ingresadas por el usuario, determinando para cada sentencia si es válida o no para el lenguaje definido en 5.1.2. A continuación se detalla el funcionamiento de este módulo.

La figura 5.5 representa el flujo que realiza el *parser* para determinar si una entrada es válida o no, además de las distintas salidas y errores para cada caso.

Para determinar si una sentencia es válida o no, se utiliza una secuencia de expresiones regulares y evaluaciones condicionales para concluir si se debe retornar un mensaje de error o el diccionario con los datos correspondientes. Las expresiones regulares y evaluaciones condicionales corresponden a los círculos y rombos respectivamente. Ambos derivan en una respuesta booleana; si o no. Eventualmente se llega a un estado final, representado por un cuadrado, o a un mensaje de error, representado por texto plano.

En cada mensaje de error contemplado se adjunta en azul un ejemplo de sentencia que haya derivado a ese estado final. Si se trata de una variable en la sentencia y el tipo de la misma afecta al flujo, se señala mediante S o N el tipo de la variable, siendo S correspondiente a *string* y N correspondiente a *number*.

[1] Observar que por los tipos de sentencias permitidos, siempre se actualiza una única variable por sentencia.

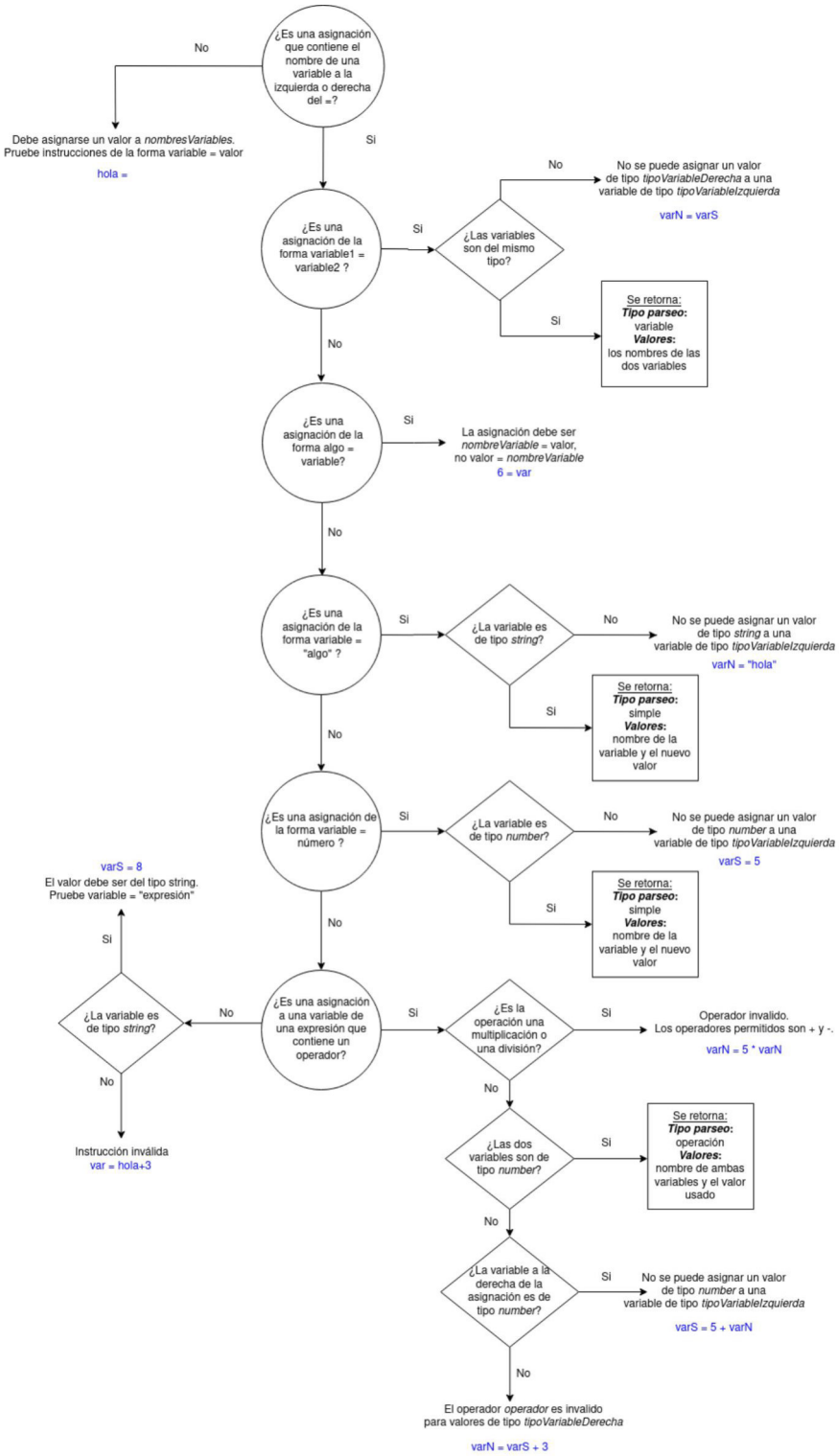

Figura 5.5: Diagrama del parser

5.2. Implementación

5.2.1. Ejercicios

Los conjuntos de ejercicios están guardados en archivos de formato JSON que están incluidos en los archivos de la aplicación. Esto también permite que sea muy fácil agregar ejercicios nuevos a los conjuntos existentes; simplemente se deben definir los parámetros del nuevo ejercicio y agregar a este como un nuevo ítem del JSON. De esta forma, cada conjunto de ejercicios es altamente extensible, pudiéndose agregar más ejercicios sin realizar nuevas implementaciones. El formato de estos archivos JSON se describe en más detalle en A.1.

5.2.2. Página de inicio

Se diseñó una página de inicio simple que muestra el listado de conjuntos de ejercicios. Los conjuntos ya realizados se muestran con un *tick* verde; el progreso (es decir, qué conjuntos ya fueron completados) del estudiante es guardado en el almacenamiento local del navegador, de forma que si se cierra la página y se vuelve a abrir, el progreso será mantenido. Además se provee un botón de *Reset* que permite reiniciar los conjuntos, marcando todos como no completados aún. Una captura de la página de inicio puede verse en la figura 5.6; en este caso el tutorial y el primer conjunto de ejercicios ya fueron completados.

Figura 5.6: Página de inicio

5.2.3. Tutorial

Como un ejercicio inicial, previo a los conjuntos de ejercicios, se incluyó un tutorial de la plataforma. El objetivo del tutorial no es explicar los ejercicios, sino demostrar el uso de las distintas partes de la herramienta. La motivación del tutorial es que en los ejercicios se minimicen los errores por causa de no comprender el uso de la herramienta, para así poder incentivar al descubrimiento de errores conceptuales. El uso de un tutorial también pretende disminuir la necesidad de intervención docente para explicar el funcionamiento de la herramienta.

El tutorial está dividido en varios pasos que el estudiante tiene que atravesar. En cada paso, toda la pantalla está oscurecida excepto por la parte de la herramienta que se está enfocando (ver figura 5.7). Los pasos del tutorial son detallados en A.3.

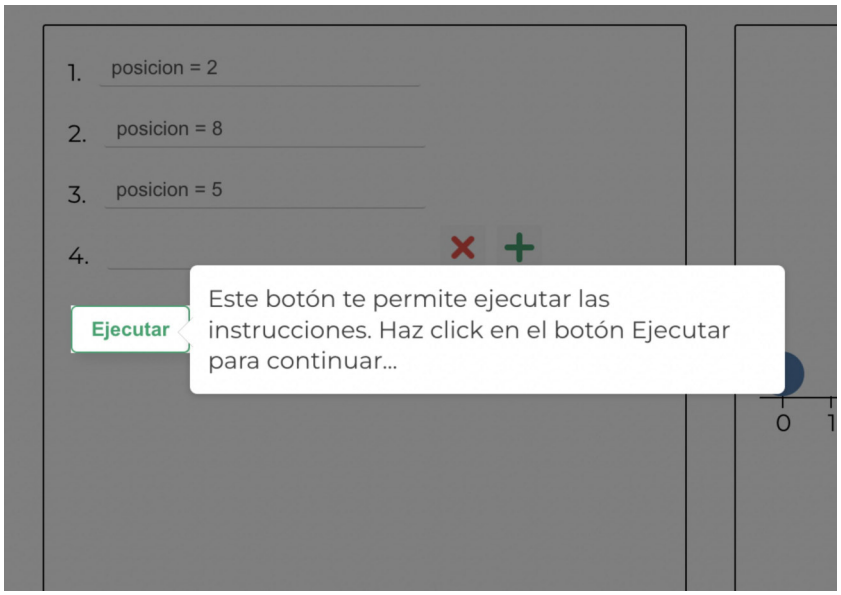

Figura 5.7: Ejemplo de un paso del tutorial

Capítulo 6

Pruebas

Una vez finalizada la implementación de todos los conjuntos de ejercicios, se realizaron algunas instancias de prueba con el objetivo de validar lo realizado y detectar errores y espacios de mejora en la herramienta. El objetivo principal de estas instancias de prueba fue validar si los errores esperados en cada ejercicio son realmente los errores que los estudiantes cometen, y que las visualizaciones propuestas por la herramienta realmente los llevan a corregir sus errores y lograr los objetivos propuestos. Además, se aprovecharon las instancias para detectar errores y espacios de mejora tanto en la experiencia de usuario de la herramienta como en el diseño de los ejercicios y sus visualizaciones.

Para llevar a cabo estas instancias se contó con la participación de 12 adolescentes y adultos jóvenes, de entre 15 y 25 años de edad. Cada instancia de prueba consistía en dos partes: en primer lugar, el participante respondía algunas preguntas sobre su nivel educativo y sus conocimientos previos de programación, y en segundo lugar realizaba todos los ejercicios propuestos por la herramienta, mientras alguno de los integrantes del equipo del proyecto tomaba notas y apuntes sobre lo realizado. Si en algún momento el participante no podía corregir un error, se le brindaban ayudas para que lo lograra y pudiera continuar con el siguiente ejercicio. Se documentaron las ayudas brindadas, ya que estas evidencian casos en donde la herramienta por si sola no fue suficiente para darle al participante las estrategias necesarias para resolver el problema, necesitando intervención de alguien en el rol de docente.

Se eligieron personas jóvenes porque este es el público objetivo de la herramienta: adolescentes cursando educación secundaria y adultos jóvenes empezando sus carreras terciarias o realizando cursos de programación. Además, la mayoría de los participantes no tenían conocimientos de programación. Más del 50 % respondió "no" a la pregunta "¿Conocés lo que es la programación?" y solo 16 % respondió "sí" a la pregunta "¿Alguna vez programaste?". Los dos participantes que respondieron "sí" a esta pregunta habían programado en Scratch.

A continuación se detallan resultados y conclusiones de las pruebas realizadas para cada conjunto de ejercicios. Luego de realizar estas pruebas, se hicieron cambios en el diseño de los ejercicios y su visualización, para intentar resolver

algunos de los problemas evidenciados en estas instancias. Finalmente, se realizó una segunda instancia de pruebas con nuevos participantes, para evaluar si los cambios afectaban positivamente en su desempeño.

6.1. Primer conjunto

La mayoría de los participantes lograron alcanzar los objetivos del primer conjunto exitosamente. Más del 50 % resolvieron el primer ejercicio del conjunto sin errores, mientras que el resto cometieron algunos de los errores esperados y fueron capaces de corregirse gracias a la visualización de la herramienta. El 100 % realizó el segundo ejercicio sin errores, pudiendo aplicar lo realizado en el ejercicio anterior. En el tercer ejercicio solo 25 % de los participantes realizaron errores; en todos los casos fue debido a no interpretar el número denotado por la llave como la distancia entre la pelota y el cuadrado objetivo, que era un error esperado. Ningún participante cometió errores en el ejercicio 4, y solo un participante realizó un error en el ejercicio 5, pudiendo corregirlo sin ayuda. Como era esperado, los ejercicios 6 y 7 resultaron más difíciles. En el ejercicio 6, solo 2 participantes no realizaron errores; los errores realizados por la mayoría de los participantes eran errores esperados, y pudieron corregirlos sin ayuda. Un 33 % de los participantes pudo resolver el ejercicio 7 sin errores, demostrando que construyeron conocimiento en base al ejercicio 6; de los 8 participantes restantes, 6 pudieron corregir sus errores utilizando únicamente la herramienta, mientras que 2 precisaron ayuda.

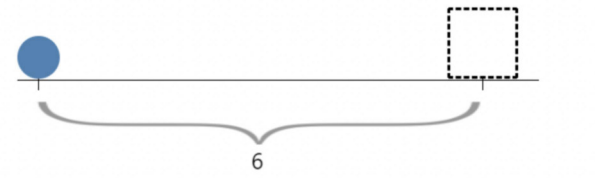

Figura 6.1: Visualización del ejercicio 7 previo a las pruebas

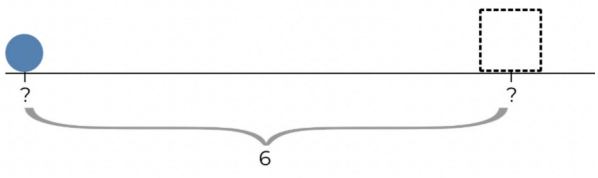

Figura 6.2: Visualización del ejercicio 7 luego de las pruebas

Las pruebas validaron la visualización del primer conjunto de ejercicios, así como también los errores que se espera que los estudiantes realicen. Se observó que a muchos participantes les costó darse cuenta cuando la posición inicial pasaba de ser conocida a desconocida; gracias a esto, se modificó la visualización del ejercicio para denotar las posiciones desconocidas con un signo de interrogación ("?"). En la figura 6.1 se muestra un ejemplo de la visualización previo a la realización de las pruebas, mientras que en la figura 6.2 se muestra la visualización luego de los cambios realizados.

6.2. Segundo conjunto

El segundo conjunto de ejercicios es el más corto y los participantes lo realizaron rápidamente. En el primer ejercicio del conjunto, 25 % de los participantes tuvieron errores, pudiendo corregirlos con la visualización. En los dos siguientes ejercicios solo 2 de los participantes cometieron errores, también siendo capaces de corregirlos sin ayuda.

6.3. Tercer conjunto

El tercer conjunto de ejercicios fue donde los participantes presentaron las mayores dificultades. En los primeros dos ejercicios, el 75 % de los participantes no realizó ningún error, mientras que en el ejercicio 3 este número fue un 58 %. La mayoría de los errores cometidos por los participantes eran los esperados, como por ejemplo no poner comillas alrededor del valor de string, y pudieron ser corregidos gracias a la visualización y los mensajes de error. Un error no esperado fue la dificultad de los participantes para escribir el *string* pedido, ya que en varios casos ignoraban mayúsculas, tildes, y signos de puntación. Luego de observar esto, se decidió modificar los ejercicios para que aceptaran como solución correcta *strings* que fueran iguales al pedido, pero ignorando diferencias en mayúsculas, tildes, y signos de puntuación, ya que si bien estos son aspectos computacionales importantes, no se consideran relevantes al objetivo del ejercicio.

Más de un 50 % de los participantes resolvió el ejercicio 4 sin errores (el ejercicio puede verse en la figura 3.21). De los 5 participantes que cometieron errores, 4 asignaron el valor inicial de etiqueta2 a etiqueta3, en lugar del nuevo valor. Todos los participantes pudieron auto-corregirse utilizando la visualización de la herramienta como guía. Como se explicó en 3.4.4, este ejercicio tiene 3 soluciones posibles; solo uno de los participantes llegó a la solución `etiqueta3 = "Qué tal?"`, 3 participantes llegaron a la solución `etiqueta3 = etiqueta1`, y el resto llegó a la solución `etiqueta3 = etiqueta2`. Esto evidencia la construcción de conocimiento instrumental de los participantes en lo que refiere a las asignaciones de la forma `variable = variable`.

Un 50 % de los participantes realizó el ejercicio 5 sin errores. Al ver los errores cometidos, se observó que todos intentaban asignar un valor explícito a

etiqueta1, y luego el mismo valor a etiqueta2. Viendo esto, se observó que las instrucciones del ejercicio no estaban bien redactadas, dando lugar a ambigüedad en el objetivo del ejercicio; luego de las pruebas se corrigió la redacción para que el objetivo quedara más claro. El ejercicio 6 fue resuelto por 80 % de los participantes sin errores, y el 20 % restante pudo corregirlos utilizando la visualización de la herramienta. El ejercicio 7 es el ejercicio previo al intercambio de variables, y fue resuelto sin errores por todos los participantes.

El último ejercicio del conjunto plantea el clásico problema de intercambio de variables. Este ejercicio resultó notoriamente más difícil que los anteriores: tan solo un 15 % de los participantes llegó a una solución sin errores. De los participantes que realizaron errores, la gran mayoría precisó ayuda para resolver el ejercicio. Esto demostró que la visualización propuesta no estaba siendo suficiente para construir el conocimiento necesario sobre el intercambio de variables, y que esta debía ser modificada. La modificación surgió de observar cómo uno de los participantes lograba resolver el ejercicio: para razonar sobre las asignaciones, utilizaba movimientos con los dedos, moviendo un dedo de una etiqueta a otra para simbolizar el valor siendo copiado de una etiqueta a otra. Se utilizó esta idea para modificar la visualización de las etiquetas: en las asignaciones de la forma `variable1 = variable2`, se agregó una animación que muestra el valor de variable2 moviéndose hacia la variable1, de forma análoga a como el participante realizaba con sus dedos. Las modificaciones propuestas en este conjunto de ejercicios para la visualización buscan ayudar al estudiante a tomar conciencia de que lo que él hace con sus dedos, también lo hace la computadora con los elementos de los cuales dispone. De esta manera se induce a la transformación de una acción en un concepto por medio de la abstracción reflexiva (ver pág. 7 de cap. 2).

6.4. Cuarto conjunto

Para las pruebas del último conjunto se contó con la participación de 11 personas, ya que una no tuvo el tiempo de realizar estos ejercicios. De estos participantes, ninguno cometió errores en los ejercicios 1 y 3, y solo uno cometió (y corrigió) un error en el segundo. En el ejercicio 4, hubo un único participante que realizó errores, pudiendo el resto resolver el ejercicio en su primer intento. El último ejercicio del conjunto, que introduce la formalización del intercambio de variables, fue nuevamente el más difícil: 5 participantes realizaron errores, y solo uno de ellos pudo corregirse sin ayuda. Todos los participantes que precisaron ayuda para resolver este ejercicio también habían tenido errores en el ejercicio 8 del tercer conjunto. Esto demostró que los participantes no estaban construyendo el conocimiento conceptual requerido sobre el intercambio de variables, y entonces tampoco podían formalizarlo en este ejercicio. Se espera que al realizar los cambios propuestos en la visualización de las etiquetas, los participantes puedan construir ese conocimiento conceptual, y puedan así formalizarlo en este ejercicio.

6.5. Segunda instancia de pruebas

Luego de la primera instancia de pruebas, se realizaron las modificaciones ya mencionadas en los ejercicios, para luego poder realizar una segunda instancia. En la segunda instancia se contó con 9 participantes de entre 15 y 25 años. Un 88 % de los participantes respondieron "Sí" a la pregunta "¿Conocés lo que es la programación?", y un 44 % habían programado antes (mayormente en lenguajes como Scratch). Los participantes realizaron todos los conjuntos de ejercicios. A continuación, se detallan los resultados para el primer y el tercer conjunto, siendo estos los conjuntos con cambios en su visualización.

En esta segunda instancia, un 77 % de los participantes pudieron resolver el ejercicio 6 del primer conjunto sin ayuda, donde se había realizado la modificación en la visualización que se muestra en las figuras 6.1 y 6.2. Los errores realizados fueron los esperados, mayormente `posicion = 7` y `posicion = posicion + 7`; 7 de los participantes pudieron corregir sus errores utilizando únicamente la visualización, mientras que 2 precisaron ayuda.

El tercer conjunto fue el que tuvo modificaciones más significativas, ya que se cambió la animación de las etiquetas para las asignaciones de la forma `variable = variable`. En los primeros dos ejercicios, no hubo resultados inesperados; los participantes pudieron resolverlos todos sin errores, exceptuando a un participante en el primer ejercicio que tuvo un error al escribir el texto de la etiqueta (y que rápidamente corrigió). En el ejercicio 3 de este conjunto se tuvo un resultado inesperado: al pedirles que igualaran dos etiquetas, más del 50 % de los participantes resolvieron el ejercicio con la sentencia `etiqueta2 = etiqueta1` a pesar de no haber visto ninguna sentencia de esta forma. Si bien esta resolución es correcta, la solución que esperábamos con más frecuencia es `etiqueta2 = "Hola! Qué tal?"`.

Un 66 % de los participantes resolvió el ejercicio 4 sin errores, mientras que el 33 % restante pudo corregir sus errores utilizando la visualización como guía. En el ejercicio 5 solo un participante cometió errores; a modo de ayuda, se le instruyó que volviera al ejercicio anterior y lo resolviera de nuevo, luego de lo cual pudo resolver el ejercicio 5 sin problema. Se recuerda que en la instancia de pruebas anteriores, solo un 50 % habían resuelto el ejercicio sin errores, en lo que había sido atribuido a una ambigüedad en la descripción del ejercicio. Entendemos que estos resultados demuestran que el cambio en la descripción del ejercicio impactó positivamente en el número de participantes que lo resolvieron correctamente. El 100 % de los participantes realizó el ejercicio 6 sin errores, y solo un participante cometió errores en el ejercicio 7.

En el último ejercicio del conjunto se vieron resultados positivos con respecto al cambio de visualización. Un 22 % de los participantes realizó el ejercicio sin errores, pero lo interesante fue ver cómo corrigieron sus errores los participantes que sí los cometieron. En todos los casos el error principal era no reconocer la necesidad de utilizar la variable auxiliar. 4 de los 9 participantes realizaron movimientos con los dedos (yendo de una etiqueta a otra) para poder razonar el ejercicio, sugiriendo que se logró lo que se buscaba con la nueva visualización. Esto sugiere que la nueva visualización es más intuitiva como forma

de representar las asignaciones `variable = variable`, permitiendo reforzar la construcción del concepto de copia. Si bien un 44 % de los participantes precisó ayuda para resolver el ejercicio, notamos que precisaron menos ayuda que los participantes de la primera instancia, y que para ayudarlos podíamos hacer preguntas que incitaran a la reflexión basándose en la visualización (por ejemplo "¿Qué pasa en la pantalla cuando pongo etiqueta2 = etiqueta1?"). El hecho de que varios de los participantes hayan necesitado ayuda para resolver el ejercicio de intercambio de variables puede ser evidencia de la necesidad de una tercera iteración sobre este conjunto de ejercicios.

Al formalizar este conocimiento en el ejercicio 5 del cuarto conjunto, se observa que solo uno de los participantes requirió ayuda, mientras que el resto pudo resolver el ejercicio sin errores, evidenciando que se logró construir el conocimiento sobre intercambio de variables.

En la figura 6.3 se muestra una comparación de resultados entre la primera y la segunda instancia de pruebas. Se consideran los ejercicios en los cuales los participantes habían precisado ayuda durante la primera instancia, para evaluar si las modificaciones realizadas disminuyeron la necesidad de ayuda por parte de los entrevistadores. Si bien estas pruebas no son exhaustivas, entendemos que los cambios introducidos mejoraron la visualización de la herramienta, facilitando la construcción de conocimiento computacional por parte de los participantes.

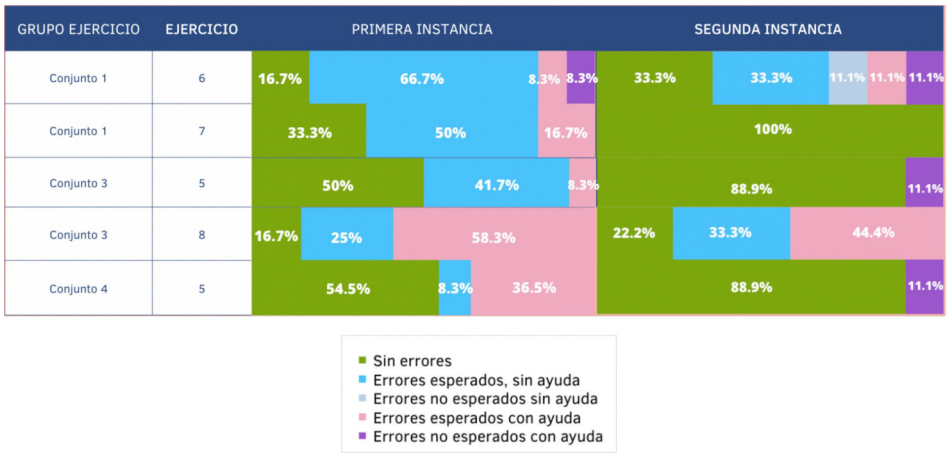

Figura 6.3: Comparación de resultados entre instancias

Capítulo 7

Conclusiones

En este capítulo se describen las conclusiones del proyecto. Se evalúan los resultados alcanzados con respecto a los objetivos propuestos, además de describir ideas de trabajo futuro.

7.1. Conclusiones del proyecto

El objetivo del proyecto era diseñar e implementar una herramienta didáctica que obligara al estudiante a transformar su pensamiento en pensamiento computacional para lograr un objetivo (1.1). El objetivo concreto de la herramienta debía ser fomentar este pensamiento computacional aplicado a los conceptos de variable y asignación, fundamentales en la programación imperativa.

Se diseñaron cuatro conjuntos de ejercicios incrementales, donde cada conjunto tiene como objetivo la construcción de nuevo conocimiento partiendo del conocimiento construido en el conjunto anterior. Se trabajaron lo conceptos de incremento de variable, orden de las sentencias de un programa, intercambio de variables y asignaciones de la forma `variable = variable` o `variable = expresion` donde `expresion` es de la forma `variable + valor`.

Se realizaron dos instancias de pruebas con participantes de entre 15 y 25 años. En la primera instancia, se evidenció la construcción de conocimiento sobre variable y asignación por parte de los participantes, pero la mayoría no alcanzó el nivel máximo (representado por el ejercicio de intercambio de variables) sin ayuda. A raíz de esto se realizaron modificaciones en las visualizaciones de dos conjuntos de ejercicios, para facilitar la conceptualización que estaba faltando. Luego de implementar estas modificaciones, se realizó una segunda instancia de pruebas con nuevos participantes. En esta segunda instancia se notaron cambios positivos, pudiendo los participantes resolver más ejercicios sin ayuda. Si bien varios de los participantes necesitaron ayuda para resolver el ejercicio de intercambio de variables, la visualización facilitó tanto la resolución del ejercicio como la ayuda que fue necesaria por parte de los integrantes del equipo del proyecto.

Las instancias de prueba resultaron fundamentales para validar el diseño de los ejercicios y de la aplicación, y a su vez permitieron validar que los errores cometidos por los participantes eran en su mayoría los errores esperados. Fue muy interesante ver cómo personas sin conocimientos de programación lograron dar los primeros pasos hacia la construcción de conocimiento sobre variables y asignación en un corto tiempo, evidenciando que se alcanzaron los objetivos propuestos.

7.2. Trabajo futuro

Desde el punto de vista teórico, sería interesante realizar un análisis más profundo de los resultados de las pruebas, tanto desde el punto de vista de la construcción del concepto de asignación como desde el punto de vista del impacto de algunas estrategias usadas en el diseño de la herramienta en el proceso de dicha construcción. En el primer caso se podrían comparar resultados de otras investigaciones realizadas sobre este concepto y en el segundo caso se podrían justificar algunas decisiones tomadas intuitivamente con teorías didácticas establecidas, como por ejemplo la teoría de las variaciones, lo cual estaba fuera del alcance de este proyecto.

Las instancias de pruebas evidenciaron que la herramienta cumple con el objetivo de brindar apoyo didáctico para el desarrollo del pensamiento computacional. Por lo tanto, la expansión de la misma con conjuntos de ejercicios que aborden tanto el concepto de variable y asignación desde otros enfoques como otros conceptos básicos de programación es una línea de trabajo futuro que queda naturalmente abierta. Podrían diseñarse e implementarse ejercicios para trabajar sobre conceptos fundamentales como selección e iteración, evaluación de operaciones booleanas, entre otros.

Desde un punto de vista de la implementación de la herramienta, se podría agregar un *backend* para tener funcionalidades de autenticación y sesiones. De esa forma, los estudiantes podrían autenticarse en el sistema y que este almacene su progreso; se recuerda que actualmente el progreso del estudiante se guarda en el almacenamiento del navegador, con lo que si el estudiante accede a la página desde otra computadora, no ve su progreso actual. Se podría expandir sobre esta idea y agregar también un registro de algunos de los errores cometidos por cada estudiante. También se podría incluir usuarios de tipo “Docente” que puedan ver el progreso de todos los estudiantes desde un panel de administración y agregar nuevos ejercicios a los conjuntos existentes a partir de un formulario que complete el JSON (sin necesidad de que ellos conozcan la “estructura” a seguir).

Referencias

da Rosa, S. (2016). Preconceptions of novice learners about program execution. En *Proceedings of the 27th annual psychology of programming interest group workshop, cambridge, uk.*

da Rosa, S., y Gomez, F. (2020, 04). A research model in didactics of programming. *CLEI Electronic Journal*, *23*. doi: 10.19153/cleiej.23.1.5

Dehnadi, S., y Bornat, R. (2008, 07). The camel has two humps. *School of Computing, Middlesex University*, 11. (http://www.eis.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf)

Gomez, F. (2021). *Un modelo de investigación en didáctica de la programación* (Tesis de maestría). Facultad de Ingeniería, Universidad de la República.

Gómez, F., y da Rosa, S. (2023, 03). The construction of knowledge about programs. En *Ppig 2022 - 33rd annual workshop.*

Javascript. (s.f.). Descargado de https://developer.mozilla.org/en-US/docs/Web/JavaScript (Accedido: 2023-07-02)

Jest. (s.f.). Descargado de https://jestjs.io/ (Accedido: 2023-07-16)

React. (s.f.). Descargado de https://react.dev/ (Accedido: 2023-07-02)

Schulte, C., y Budde, L. (2018, 11). A framework for computing education: Hybrid interaction system: The need for a bigger picture in computing education. En *Koli calling '18: Proceedings of the 18th koli calling international conference on computing education research* (p. 1-10). doi: 10.1145/3279720.3279733

ShepherdJS. (s.f.). Descargado de https://shepherdjs.dev/ (Accedido: 2023-06-30)

Vygotsky, L. S. (1981). Capítulo vii. En M. M. Rotger (Traduc.), *Pensamiento y lenguaje.* MIT.

Anexo A

Implementación

A.1. Ejercicios en formato JSON

Cada conjunto de ejercicios es representado mediante un objeto en formato JSON, definido en un archivo JavaScript. Para cada conjunto, el objeto JSON que lo representa tiene como clave los números de los ejercicios (1,2,3, etc) y como valor otro objeto JSON que representa al ejercicio. El formato de este otro objeto depende del conjunto de ejercicios, ya que la información relevante es altamente dependiente en la semántica del ejercicio en sí. A continuación se explican los formatos utilizados para cada conjunto de ejercicios.

A.1.1. Primer conjunto

Los ejercicios de este conjunto están representados por un objeto con las siguientes propiedades:

- `description` - Descripción del ejercicio, mostrada al estudiante en la parte superior de la pantalla.
- `givenInstructions` - Arreglo de strings que representa las instrucciones ya dadas en el ejercicio. Si es vacío, no se provee ninguna instrucción dada.
- `initialPosition` - Representa la posición inicial de la pelota
- `offset` - Posición en la cual "empieza" la línea donde se encuentra la pelota, es decir la primera posición que sería visible en la línea.
- `lineLength` - Largo total visible de la línea
- `goal` - Posición en la línea en la que está el cuadrado objetivo
- `marks` - (opcional) Arreglo de objetos de la forma

```
{
markNumber: number,
markLabel: boolean
}
```

Si se provee este arreglo, se omitirán todas las marcas de índices en la línea excepto las especificadas en el arreglo, y se agregaran llaves indicando la distancia entre ellas. La propiedad `markLabel` indica si se debe mostrar el índice correspondiente a esa marca, o si se deberá dejar como una incógnita. Esto permite definir ejercicios en donde la posición inicial es desconocida.

En el listado A.1 se muestra la definición del primer ejercicio del conjunto, mientras que en el listado A.2 se muestra la definición del último ejercicio del conjunto.

Listado A.1: Definición del primer ejercicio del conjunto 1

```
{
    description: "Complete las instrucciones para que
      la pelota se ubique en el cuadrado objetivo",
    initialPosition: 0,
    offset: 0,
    goal: 5,
    lineLength: 12,
    givenInstructions: ["posicion = 2"],
}
```

La razón por la cuál se permiten valores con decimales (no enteros) para `initialPosition` `offset`, `goal` es para que al utilizar posiciones iniciales desconocidas, el estudiante no pueda adivinar cuál es la posición del cuadrado objetivo mediante prueba y error. Como la sintaxis de las instrucciones válidas solo permite asignar valores enteros a posicion, nunca podrán insertar directamente el valor. Esto obliga al estudiante a asignar un valor a posición definido como un incremento sobre el valor actual (desconocido).

A.1.2. Segundo conjunto

Los ejercicios del segundo conjunto están representados por objetos con las siguientes propiedades:

- `description` - Descripción del ejercicio, mostrada al estudiante en la parte superior de la pantalla.
- `givenInstructions` - Arreglo de strings que representa las instrucciones ya dadas en el ejercicio. Si es vacío, no se provee ninguna instrucción dada.
- `goal` - Objeto con dos claves `row` y `column` que representan la fila y columna donde debe estar la salida del laberinto.

Listado A.2: Definición del último ejercicio del conjunto 1

```
{
    description:
      "Escriba la instruccion necesaria para que
      la pelota se ubique en el cuadrado objetivo",
    initialPosition: 4.71,
    lineLength: 9,
    offset: 4.71,
    goal: 10.71,
    givenInstructions: [],
    marks: [
      {
        markNumber: 0,
        markLabel: false,
      },
      {
        markNumber: 6,
        markLabel: false,
      },
    ],
  },
```

- `cells` - Objeto en formato JSON que representa el laberinto del ejercicio. Su formato se explica a continuación.

La propiedad `cells` toma como valor un arreglo de arreglos que representa una matriz de tamaño n x n, donde n es la dimensión del laberinto. En las celdas de la matriz (que representan las celdas del laberinto), se guarda un objeto con dos propiedades booleanas `right` y `bottom`. La propiedad `right` es `true` si existe una pared en el borde derecho de la celda, mientras que `bottom` es `true` si existe una pared en el borde inferior de la celda. Se asume que el laberinto tiene un perímetro de paredes, con lo cual no es necesario incluir las paredes derechas de las celdas en la última columna ni las paredes inferiores de las celdas en la última fila. En el listado A.3 se muestra la definición del primer ejercicio del conjunto y en el listado A.4 se muestra la definición del laberinto para este ejercicio.

Listado A.4: Definición del laberinto del primer ejercicio del conjunto 2

```
const CELDAS_PRIMER_EJERCICIO = [
  // row 1
  [
    { right: false, bottom: true },
    { right: true, bottom: false },
    { right: false, bottom: false },
    { right: false, bottom: false },
  ],
```

Listado A.3: Definición del primer ejercicio del conjunto 2

```
{
    description:
      "Complete las instrucciones para que la
      pelota llegue a la salida del laberinto.",
    cells: CELDAS_PRIMER_EJERCICIO,  // (ver listado 4)
    goal: {
      row: 3,
      column: 3,
    },
    givenInstructions:
      ["columna = 1", "fila = 2", "columna = 2"],
  },
```

```
  // row 2
  [
    { right: true, bottom: false },
    { right: true, bottom: false },
    { right: true, bottom: false },
    { right: false, bottom: false },
  ],
  // row 3
  [
    { right: true, bottom: false },
    { right: false, bottom: true },
    { right: true, bottom: true },
    { right: false, bottom: false },
  ],
  // row 4
  [
    { right: true, bottom: false },
    { right: false, bottom: false },
    { right: false, bottom: false },
    { right: false, bottom: false },
  ],
];
```

A.1.3. Tercer conjunto

Los ejercicios del tercer conjunto están representados por objetos con las siguientes propiedades:

- `description` - Descripción del ejercicio, mostrada al estudiante en la parte superior de la pantalla
- `givenInstructions` - Arreglo de strings que representa las instrucciones ya dadas en el ejercicio. Si es vacío, no se provee ninguna instrucción dada.

- `tags` - Objeto que tiene como claves los nombres de las etiquetas admitidas en el ejercicio, y como valores objetos con las siguientes propiedades:
  - `initialValue` - Valor inicial de la etiqueta; puede ser vacío.
  - `finalValue` - (opcional) Valor final que debe tener la etiqueta para cumplir el objetivo del ejercicio (si no está, no se espera ningún valor final en particular).
  - `hidden` - (opcional) Valor booleano que representa si la etiqueta está escondida o no (si no se incluye, se asume que la etiqueta es visible).
  - `random` - (opcional) Valor booleano que determina si el valor inicial de la etiqueta es dado por `initialValue` o si debe ser randomizado (cuando es `true`, el valor inicial es randomizado).
- `conditions` - (opcional) Condiciones que deben cumplirse para haber logrado el objetivo del ejercicio. Es un arreglo de condiciones (objetos) que tienen las siguientes propiedades:
  - `type` - Tipo de la condición; puede ser o `CONDITION_TYPES.equality` para representar que dos variables deben ser iguales, o `CONDITION_TYPES.exchange` para representar que dos variables deben intercambiar sus valores.
  - `variable1` - Nombre de la primera variable involucrada en la condición.
  - `variable2` - Nombre de la segunda variable involucrada en la condición.

En los listados A.5 y A.6 se muestran las definiciones del primer y último ejercicio del conjunto, respectivamente.

Listado A.5: Definición del primer ejercicio del conjunto 3

```
{
    description:
      'Complete las instrucciones para que el texto
      "Saluda!" se visualice en la etiqueta1 y el
      texto "Hola Mundo!" se visualice en la etiqueta2.',
    tags: {
      etiqueta1: {
        initialValue: "",
        finalValue: "Saluda!"
      },
      etiqueta2: {
        initialValue: "",
        finalValue: "Hola Mundo!"
      }
    },
    givenInstructions: ['etiqueta1 = "Saluda!"']
}
```

Listado A.6: Definición del último ejercicio del conjunto 3

```
{
    description:
      "Escriba las instrucciones necesarias para intercambiar
      los textos de etiqueta1 y etiqueta2: el texto que ahora
      esta en etiqueta1 debe terminar en etiqueta2 y el texto
      que ahora esta en etiqueta2 debe terminar en etiqueta1.",
    tags: {
      etiqueta1: {
        hidden: true,
        random: true
      },
      etiqueta2: {
        hidden: true,
        random: true
      },
      etiqueta3: {
        initialValue: ""
      }
    },
    conditions: [
      {
        variable1: "etiqueta1",
        variable2: "etiqueta2",
        type: CONDITION_TYPES.exchange
      }
    ],
    givenInstructions: []
}
```

A.1.4. Cuarto conjunto

- `description` - Descripción del ejercicio, mostrada al estudiante en la parte superior de la pantalla.
- `givenInstructions` - Arreglo de strings que representa las instrucciones ya dadas en el ejercicio. Si es vacío, no se provee ninguna instrucción dada.
- `variables` - Arreglo de objetos que representan las variables del ejercicio. Las variables son objetos con las siguientes propiedades:
  - `name` - String que representa el nombre de la variable
  - `type` - String que representa el tipo de la variable; puede ser o `VARIABLE_TYPES.number` o `VARIABLE_TYPES.string`
  - `initialValue` - Valor inicial de la variable; puede ser vacío.
  - `random` - (opcional) Booleano que representa si el valor inicial de la variable debe ser aleatorio
- `conditions` - (opcional) Condiciones que deben cumplirse para haber logrado el objetivo del ejercicio. Es un arreglo de condiciones (objetos) con el mismo formato que en el tercer conjunto (ver A.1.3).
- `finalValues` - Objeto que tiene como claves los valores de las variables, y como valores el valor final de la variable (para que se cumpla el objetivo del ejercicio).

En los listados A.7 y A.8 se muestran las definiciones del primer y último ejercicio del conjunto, respectivamente.

Listado A.7: Definición del primer ejercicio del conjunto 4

```
{
    description:
      'Complete las instrucciones para dejar el
      valor 5 en la variable w, el valor 8 en la
      variable x, y el valor "texto" en la variable z.',
    givenInstructions: ["w = 5", 'z = "texto"'],
    variables: [
      { name: "w", type: VARIABLE_TYPES.number },
      { name: "x", type: VARIABLE_TYPES.number },
      { name: "z", type: VARIABLE_TYPES.string },
    ],
    finalValues: {
      w: 5,
      x: 8,
      z: "texto",
    },
}
```

Listado A.8: Definición del último ejercicio del conjunto 4

```
{
    description:
      "Escriba las instrucciones necesarias para
      intercambiar los valores de x y w: el valor
      que ahora esta en x debe terminar en w, y el
      valor que ahora esta en w debe terminar en x",
    givenInstructions: [],
    variables: [
      {
        name: "x",
        type: VARIABLE_TYPES.number,
        random: true
      },
      {
        name: "w",
        type: VARIABLE_TYPES.number,
        random: true
      },
      {
        name: "z",
        type: VARIABLE_TYPES.number
      },
    ],
    finalValues: {},
    conditions: [
      {
        variable1: "x",
        variable2: "w",
        type: CONDITION_TYPES.exchange,
      },
    ],
  },
```

A.2. Pruebas unitarias

Se realizaron pruebas unitarias para los módulos *Parser* y *Arithmetic*, ya que estos contienen gran parte de la lógica de la aplicación. Las pruebas utilizan Jest (*Jest*, s.f.), un *framework* de pruebas de JavaScript.

Las pruebas del módulo *Parser* prueban que para distintas entradas, el resultado de la función de *parsing* sea el esperado. Se prueban tanto entradas válidas como entradas inválidas, en este último caso verificando que el mensaje de error es el esperado. Las pruebas del módulo *Arithmetic* prueban que la función *calculateNewValue* devuelva la variable y el valor esperado para distintas entradas.

Para ambos módulos, las pruebas unitarias registran un 100 % de cubrimiento en cuanto a sentencias, bifurcaciones, funciones y líneas (como se muestra en A.1).

src/utils	88.69	100	88.46	88.81
arithmetic.js	100	100	100	100
parser.js	100	100	100	100

Figura A.1: Cubrimiento en *Parser* y *Arithmetic*

A.3. Implementación del tutorial

A continuación se detalla el funcionamiento del módulo *Tutorial*. La implementación de este modulo se basa en el uso de la librería ShepherdJS (*ShepherdJS*, s.f.), que permite crear tutoriales utilizando modales en una página web. La librería permite definir distintos pasos de un tutorial, y en cada paso bloquear toda la pantalla excepto una sección determinada. De esta forma, se puede guiar al estudiante por los distintos componentes de la aplicación, para que pueda entender la funcionalidad de cada uno.

El tutorial comienza resaltando la descripción del ejercicio que plantea el objetivo a lograr, como se muestra en la figura A.2.

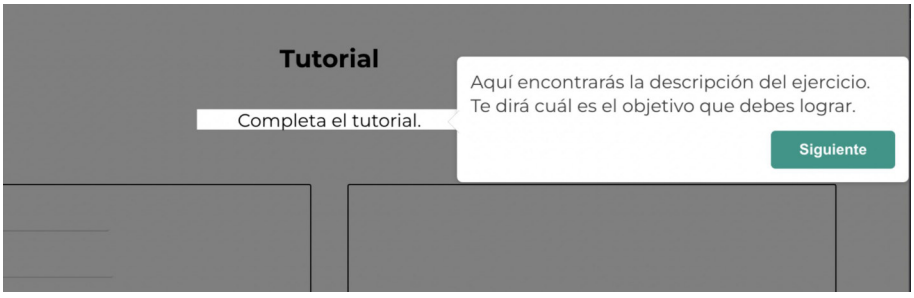

Figura A.2: Paso 1

El segundo paso, que se muestra en la figura A.3, resalta la primera instrucción dada, introduciendo el concepto de instrucción.

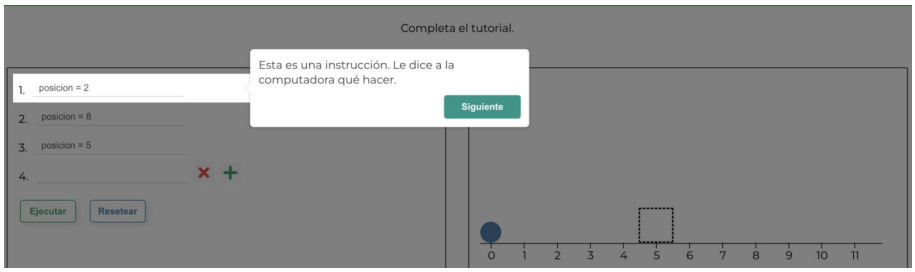

Figura A.3: Paso 2

Como tercer paso se resalta el panel de visualización, donde los estudiantes podrán ver los efectos de la ejecución de cada instrucción. Este paso puede verse en la figura A.4.

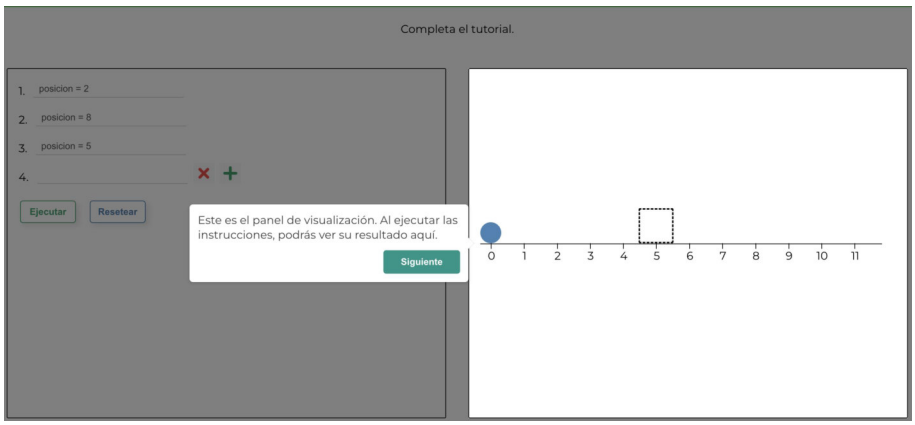

Figura A.4: Paso 3

El flujo explicado paso a paso en los siguientes pasos del tutorial es lo que los estudiantes realizarán en los ejercicios y que tomará tan solo unos pocos segundos. Por ser un proceso tan rápido en los ejercicios, fue importante incluirlo en el tutorial de manera lenta y explicada, para evitar confusiones más adelante.

Se resalta el botón de *Ejecutar*, como se ve en la figura A.5.

En el siguiente paso, el tutorial continúa con lo que ocurre a partir de presionar el botón. Las instrucciones se ejecutan secuencialmente, con lo que la primera instrucción va a ejecutarse; esto se denota subrayándola en verde, como se muestra en la figura A.6. El estudiante debe apretar el botón de *Siguiente* para que la instrucción efectivamente se ejecute.

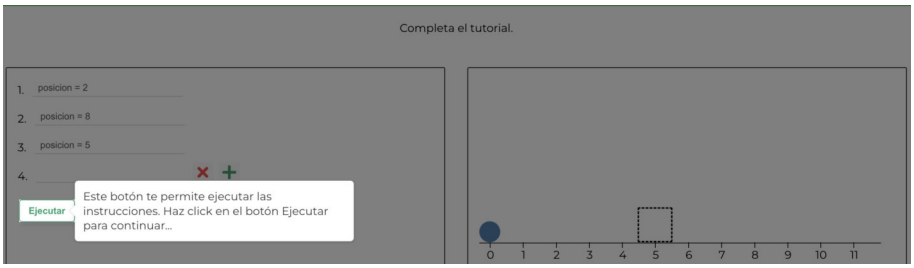

Figura A.5: Paso 4

Figura A.6: Paso 5

Al ejecutarse la primera instrucción, se ve el efecto de la ejecución en el panel de visualización, que es donde se detiene el paso 6 del tutorial (ver figura A.7).

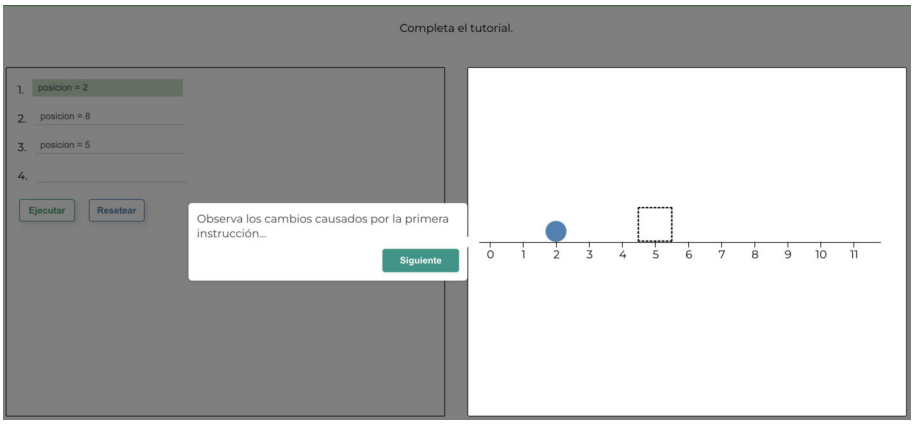

Figura A.7: Paso 6

Se repite el mismo flujo para el resto de las instrucciones, con el motivo de mostrar la ejecución de múltiples instrucciones y destacar el hecho de que esta ocurre de manera secuencial. En cada paso, se subraya la instrucción que será

ejecutada, y luego el siguiente paso muestra el resultado de la ejecución. Esto puede verse en las figuras A.8, A.9, A.10 y A.11.

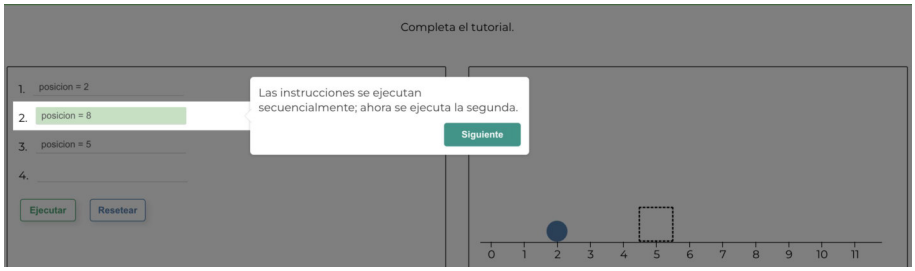

Figura A.8: Paso 7

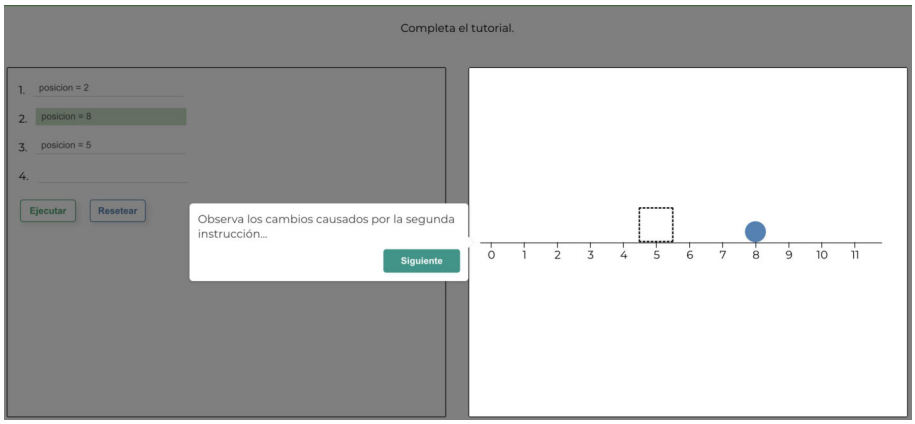

Figura A.9: Paso 8

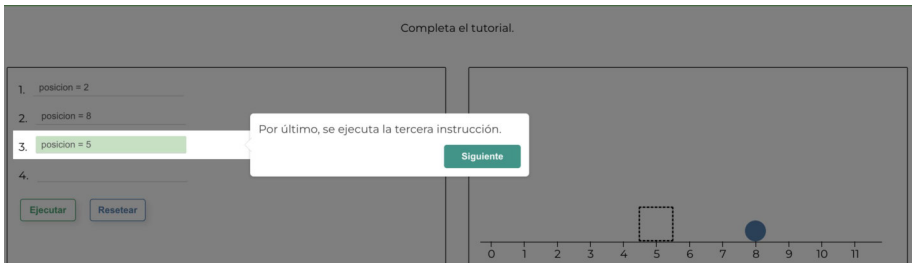

Figura A.10: Paso 9

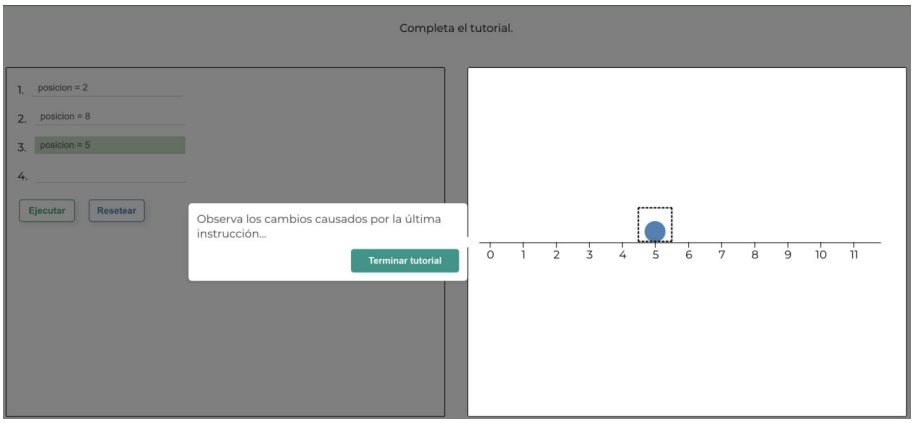

Figura A.11: Paso 10