Computación en Ingeniería: La experiencia de pensar para solucionar problemas con algoritmos y programas en aula real y aula virtual.

Elizabeth Jiménez Rey

Departamento de Computación, Facultad de Ingeniería, UBA
Av. Paseo Colón 850, C1063ACV CABA, Argentina
ejimenezrey@yahoo.com.ar

Resumen. El aprendizaje de Computación en dos cursos en la Facultad de Ingeniería de la Universidad de Buenos Aires FIUBA atiende el conocimiento de la programación de computadoras y el rol de la enseñanza de la asignatura en la formación en ciencias básicas de los estudiantes y en el desarrollo de la capacidad de solucionar problemas en Ingeniería mediante algoritmos y programas. La principal dificultad de aprendizaje reside en solucionar problemas ingeniando algoritmos y codificándolos en un lenguaje de programación. La propuesta pedagógica sistematiza el análisis y diseño en la solución de problemas con la computadora mediante el uso de rutinas de pensamiento que visibilizan el pensamiento en aula real y aula virtual durante el proceso de descubrimiento de algoritmos. La tecnología provee el soporte indispensable para la externalización de los procesos de pensamiento de los estudiantes y la promoción de un aprendizaje activo, autónomo y colaborativo.

1 Introducción

La enseñanza de Computación [1], asignatura obligatoria para todas las ingenierías (excepto Civil, Electrónica e Informática) ha evolucionado en la FIUBA en sintonía con el desarrollo tecnológico y las demandas sociales en relación a la solución de problemas con la computadora, profundizándose la práctica de programación y el desarrollo de las clases teóricas en forma interrelacionada con las clases prácticas.

La Resolución 1232/01 del Ministerio de Educación del 20 de diciembre de 2001 incluye contenidos de Informática como ciencia básica en las carreras de ingeniería. El “Libro Rojo” del Consejo Federal de Decanos de Ingeniería de la República Argentina del 1 de junio de 2018 establece como descriptores de conocimientos básicos necesarios para las carreras de ingeniería a contenidos de Informática para Inge-

niería Civil y de Fundamentos de Programación para Ingenierías en Agrimensura, Alimentos, Electricista, Electrónica, Industrial, Mecánica, Química, Petróleo. Dentro de este marco normativo, el Programa Sintético en dos cursos de Computación en la FIUBA integra: Fundamentos de arquitectura y de funcionamiento de computadoras. Técnicas para representar y almacenar información en patrones binarios. Fundamentos de sistemas operativos y de tecnologías de comunicación. Algoritmia y Programación: resolución de problemas en algoritmia (Polya) y tipos de datos básicos y estructuras de control, subprogramas y archivos de texto.

Las principales causas de las dificultades con que se enfrentan los alumnos cuando deben programar computadoras son [2]: 1. Los alumnos no están habituados al uso de la lógica para resolver problemas. 2. La programación de algoritmos representa un nuevo paradigma de resolución de problemas que requiere representación mental del mundo real (abstracción), adaptación para tener una solución computable (sintaxis y semántica del lenguaje de programación) y criterio para elegir una alternativa eficiente de implementación (toma de decisión). En general, los alumnos que cursan Computación son estudiantes ingresantes que evidencian serias dificultades para realizar abstracciones, así como hábitos de aprendizaje mecánico y memorístico, y dificultades para analizar problemas y articular estrategias para resolverlos. El principal obstáculo del aprendizaje de la asignatura reside en solucionar problemas ingeniando algoritmos expresados en un lenguaje de programación para codificar un programa a ser ejecutado por la computadora.

Zapata Ros [3] sostiene que las competencias que se muestran como más eficaces en la codificación de un algoritmo son la parte más visible de una forma de pensar que es útil no sólo en el ámbito de las actividades cognitivas que intervienen en el desarrollo y en la creación de programas y de sistemas informáticos. Hay una forma específica de pensar, el pensamiento computacional, que propicia el análisis y la relación de ideas para la organización y la representación lógica de procedimientos y que favorece las competencias computacionales.

Jeannette Wing [4] utilizó el término “pensamiento computacional” para articular una visión en la que todos, no sólo los especialistas en Informática, pueden beneficiarse de pensar como un científico informático. Según Wing [5], de manera informal, el pensamiento computacional describe la actividad mental al formular un problema para admitir una solución computacional, se superpone con el pensamiento lógico y el pensamiento sistémico, incluye el pensamiento algorítmico y el pensamiento paralelo, que a su vez involucran otros tipos de procesos de pensamiento. Guillermo Simari [6] explicita que Wing describe esta forma de pensamiento como una combinación de las formas de pensamiento matemático e ingenieril. Wing [5] se refiere a una definición que utiliza junto a Jan Cuny (National Science Foundation) y Larry Snyder (University of Washington): “El pensamiento computacional

está integrado por los procesos mentales necesarios para formular un problema y encontrar su solución, de manera tal que ésta pueda efectivizarse por un conjunto de agentes con la capacidad de procesar información".

Para enseñar a solucionar problemas con la computadora se plantea la dicotomía de solamente enseñar a programar con dificultad progresiva o también favorecer el pensamiento algorítmico y computacional [3]. Esta propuesta adhiere al segundo planteo porque incluye al primero y centra la mirada del estudiante de ingeniería en la búsqueda de la solución de un problema con la computadora (competencia general) y la descentra de la reducción de la búsqueda a la implementación de una secuencia de instrucciones para desarrollar un programa (competencia específica) sin una comprensión auténtica del problema que se está resolviendo [7]. Para Seymourt Papert [8], programar es esencialmente crear con tecnología, es construir con una computadora. Y más allá de ofrecer instrucciones precisas, la construcción es la representación de ciertas ideas a través de un lenguaje, es decir, expresar una forma de resolver un problema (algoritmo) a partir de la creación de un artefacto (programa). La Algoritmia y la Programación no se aíslan sino que se distinguen y se conjugan.

El aprendizaje de Computación atiende no solamente el conocimiento de los alumnos de la programación de computadoras sino también el rol que cumple la enseñanza de la asignatura en la formación en ciencias básicas de los estudiantes y en el desarrollo de la capacidad de resolver problemas en Ingeniería mediante algoritmos y programas que permitan ejecutarlos en la computadora [9]. La tecnología como recurso material y el conocimiento tecnológico como recurso intelectual, son los medios de los que dispone el ingeniero para la solución de los problemas [10].

2 Planteo del Problema

Enfrentados al enunciado de un problema a resolver se observa que los estudiantes, en general, piensan de diferentes maneras que no resultan eficientes [11]. A veces de forma inconciente y automática como en la vida cotidiana, otras de un modo impulsivo y reactivo que los lleva a conclusiones precipitadas. En ocasiones, piensan de manera intuitiva, sin ninguna premeditación, o distraída, sin prestar atención, sin comprobar la corrección de una afirmación antes de actuar en consecuencia. Por ejemplo: responden a una pregunta sin reflexionar previamente sobre la respuesta, intentan desarrollar un programa sin analizar en profundidad el problema que resuelve, plantean una solución algorítmica ingenua que resuelve un problema para un caso particular sin advertir que un algoritmo representa una solución general que puede aplicarse a diversas situaciones, consideran válida una solución algorítmica sin realizar una prueba de escritorio, eligen una herramienta de programación para

implementar la solución a un subproblema sin detectar cuál es la naturaleza del problema y la estructura apropiada para codificarla, usan programas desarrollados con anterioridad sin evaluar la pertinencia de su aplicación para resolver la situación problemática en cuestión, etc.

Es necesario enseñar a pensar de forma eficiente. El pensamiento eficiente [11] "se refiere a la aplicación competente y estratégica de destrezas de pensamiento y hábitos de la mente productivos que nos permiten llevar a cabo actos meditados de pensamientos, como tomar decisiones, argumentar y otras acciones analíticas, creativas o críticas".

3 Propuesta Metodológica

La estrategia pedagógica utilizada para el desarrollo de contenidos de Computación se sintetiza en la experiencia de pensar para crear juntos. Se ha explorado esta línea de acción que integra enseñar a pensar y enseñar contenidos en el aula real desde el año 2007 y además, en el aula virtual desde el año 2012. Esta propuesta busca intensificar, profundizar y expandir las acciones emprendidas para mejorar la calidad de la estrategia pedagógica a través de una práctica educativa activa y reflexiva focalizada en hacer visible el pensamiento de los alumnos de una manera sistematizada y formalizada para valorar el impacto de la metacognición en el aprendizaje de la algoritmia y la programación en el primer y segundo cuatrimestres del año 2019. La propuesta metodológica focaliza la acción educativa en tres ejes:

3.1 Pensamiento computacional *(Problem Solving)*

El pensamiento computacional requiere entender las capacidades y limitaciones de una computadora, y ser capaz de expresar la solución de un problema de forma tal que una computadora lo pueda resolver [8]. Se formaliza en el ámbito de la construcción de programas con la aplicación del Modelo prescriptivo de Solución de Problemas de Polya que distingue cuatro fases en la solución de problemas con la computadora a través del planteo de una lista de preguntas y sugerencias relacionadas con las nociones fundamentales y las operaciones mentales asociadas a la implementación de cada fase para ayudar a los alumnos a resolver los problemas con programas. Es decir, cuando en el proceso de enseñanza se favorece el desarrollo de este pensamiento específico mediante el análisis del problema y el diseño de algoritmos (Fases Análisis y Diseño) durante el proceso evolutivo de construcción de pro-

gramas como una competencia clave de aprendizaje en la formación de los estudiantes de ingeniería y no se reduce el aprendizaje al hallazgo de una secuencia de instrucciones de programa (Fases Codificación y Evaluación) para solucionar problemas de dificultad progresiva.

3.2 Pensamiento visible *(Visible Thinking)*

El pensamiento visible se focaliza en tres prácticas básicas [12]: las rutinas de pensamiento, la documentación del pensamiento del estudiante y la práctica profesional reflexiva. Una rutina de pensamiento promueve movimientos de pensamiento específicos e implica un proceso que al ponerse en práctica permite que el pensamiento de los estudiantes se haga visible a medida que expresan sus ideas, debaten y reflexionan en torno a ellas. Para comprender cómo operan en el aula las rutinas de pensamiento y cómo se pueden utilizar y crear las propias se las debe considerar desde tres perspectivas: como herramientas, como estructuras y como patrones de comportamiento [13].

3.3 Mediación tecnológica *(TIC)*

Se proporciona un modo de enseñar mediado por la tecnología para ayudar a los alumnos a desarrollar formas más eficientes de utilizar la mente y a mejorar el aprendizaje perfeccionando los procesos de pensamiento grupal e individual. La capacidad de las TIC para transformar y mejorar las prácticas pedagógicas está estrechamente relacionada con la manera como estas tecnologías son realmente utilizadas por los profesores y los estudiantes en las situaciones particulares de enseñanza y aprendizaje y la manera como se insertan en el desarrollo de la actividad conjunta que despliegan profesores y estudiantes en estas situaciones [14].

4 Desarrollo de la Experiencia

4.1 Aula real

El profesor responsable en la segunda y tercera clase del primer módulo de la asignatura enuncia un problema para introducir el conocimiento de una nueva herramienta de programación, selectiva en segunda clase y repetitiva en tercera clase, y

despliega rutinas de pensamiento [13] para cuestionar, escuchar y responder las preguntas de los estudiantes con nuevas preguntas más intrigantes para movilizar el pensamiento y promover la habilidad de descubrir un algoritmo para construir el nuevo conocimiento.

Para enseñar estructuras de control básicas (selectivas /repetitivas) utiliza principalmente, entre otras, la rutina *Conectar-Ampliar-Desafiar* porque ofrece una estructura en la que el pensamiento de los estudiantes inspirado por una nueva situación problemática se puede hacer visible. Se interpela a los estudiantes y a través de preguntas, se provoca la escucha activa en lugar del oír pasivo y se los prepara para que tomen conciencia de la nueva experiencia de aprendizaje estableciendo conexiones (*Conectar*) entre el conocimiento nuevo y el conocimiento previo, identificando nuevas ideas para pensar en nuevas direcciones (*Ampliar*) y buscando cómo las nuevas ideas desafían a pensar en nuevas maneras de solucionar problemas o a cuestionar suposiciones (*Desafiar*).

Por ejemplo, en clase anterior, primera del primer módulo, se resolvió el problema de calcular el jornal de un operario a partir del costo horario de trabajo y la cantidad de horas trabajadas con *estructuras de control secuenciales* (entrada, asignación, salida), conocimiento previo. Y, en clase actual, segunda del primer módulo, se resuelve el problema de calcular el jornal de un operario a partir de su turno de trabajo, diurno (código d/D) o nocturno (código n/N), de sus respectivos costos horarios de trabajo diurno y nocturno y de la cantidad de horas trabajadas *con estructuras de control secuenciales y selectivas simples*, conocimiento nuevo. Entonces, el profesor responsable enuncia un nuevo problema: Los operarios de una fábrica trabajan en dos turnos, diurno (d/D) o nocturno (n/N), siendo los costos horarios diurnos y nocturnos de 73,45 pesos y 110,25 pesos, respectivamente. Desarrollar un programa en lenguaje Python que solicite al usuario el ingreso desde el teclado del valor del turno de trabajo del operario y de la cantidad de horas trabajadas, y muestre por pantalla el valor del jornal del operario. Considerar que el usuario no comete error de tipeo en el ingreso de los datos. Y entrega a los alumnos *una hoja de registro.*

Los estudiantes en pequeños grupos de trabajo (2 o 3 integrantes) piensan juntos para comprender el problema, es decir, realizan una *lectura comprensiva* del enunciado (¿qué problema se debe resolver?) y una *lectura de rastreo* del enunciado (¿cuáles son los recursos necesarios para resolver el problema?): identifican datos conocidos (propios del problema) o a conocer (a ser proporcionados por el usuario) iniciales o de entrada, datos desconocidos intermedios y finales (a ser calculados por el programa) o de salida y clasifican los recursos en constantes y variables. En este momento de la clase completan la *hoja de registro* como se ejemplifica en la Tabla 1. La tabla organiza el análisis del problema (aislar sus elementos y clasificarlos se-

gún sus características) y orienta y visibiliza el pensamiento (mediante descripciones y *preguntas guía* que proporcionan distintas perspectivas de análisis para la elaboración de la respuesta).

En un momento posterior, el profesor responsable presenta a los alumnos en una pizarra del aula, identificada como "Un Lugar para Algoritmiar", una tabla clasificada en Análisis del Problema (ejemplo en Tabla 2) y Diseño de la Solución (ejemplo en Tabla 3) y en otra pizarra, identificada como "Un Lugar para Programar", un texto que representa el Modelo de Programa Tipo en Python (ejemplo en Tabla 4). La *tabla* y el *modelo* constituyen dispositivos didácticos para sistematizar el pensamiento en el proceso de búsqueda de un algoritmo y su representación en forma de programa. El profesor responsable y los estudiantes emprenden un proceso de exploración para construir una buena solución del problema planteado y lograr la apropiación del nuevo conocimiento desde el conocimiento previo.

Los dispositivos didácticos, *tabla* y *modelo*, ayudan a los estudiantes a pensar (*How To Think It*) para enfrentar la incertidumbre y la perplejidad que se presentan cuando deben abordar la solución de un problema con la computadora. Cada columna de la tabla define una acción a desarrollar.

Los diferentes grupos de trabajo comparten sus respuestas volcadas en las hojas de registro y todos los participantes del curso piensan juntos y con el profesor responsable en una acción conjunta que suscita ideas más pensantes para encaminarse hacia el encuentro de la solución del problema. Se registra el proceso de indagación en cada fila y se documenta y visibiliza el pensamiento mediante la captura de las respuestas de los estudiantes.

En Tabla 2: Columna **Número de caso**. ¿Cuántos casos son representativos del problema? Se numera cada valor elegido entre todos los posibles de ser ingresados por el usuario durante la ejecución del programa para ser analizado. Se considera valor *representativo* a cada valor que requerirá del programador la comunicación al procesador de un conjunto de órdenes distintas a ejecutar para obtener una solución general. Se identifican 2 casos. Columna **Estado Inicial**. ¿Cuáles serían los valores *representativos* del problema entre los valores posibles (*recursos de entrada*) que podría ingresar el usuario? Se considera el enunciado del problema para seleccionar los valores necesarios que representen diferentes situaciones del problema y ayuden a pensar cómo proceder para encontrar una solución general. Por ejemplo, para caso 1 se elige código de turno D y 10 horas trabajadas y para caso 2, código de turno N y 10 horas trabajadas. Columna **Estado Final**. ¿Qué resultado (valor de *recurso de salida*) se espera obtener para cada valor representativo del problema? Se debe poder determinar de inmediato, es decir, el programador debe conocer de antemano el resultado esperado asociado al valor representativo elegido o debe recurrir a una re-

presentación del problema real para verificar/corroborar/establecer el resultado esperado. Para caso 1, 734,5 pesos y para caso 2, 1102,5 pesos.

Tabla 1. Hoja de Registro.

Problema	**Integrantes**	**Curso**	**Clase**	**Fecha**
	1.	**N°**	**N°**	
	2			
	3.			

¿Qué problema se debe resolver?

Preguntas Guía: ¿cuál es la incógnita del problema? o ¿qué solicita el enunciado informar por pantalla? o ¿qué resultado(s) se debe(n) obtener? o ¿cuál es el objetivo del programa?

"Calcular el jornal de cada operario (nocturno y diurno)"

¿Cuáles son los recursos (objetos de programación a ser manipulados por las instrucciones) **necesarios para resolver el problema?**

Datos conocidos

Pregunta Guía: ¿cuáles son los datos conocidos por el programador, es decir, propios del problema y a ser proporcionados por el usuario?

"Valor de hora de trabajo diurno, valor de hora de trabajo nocturno, cantidad de horas trabajo/día, código del turno"

Datos desconocidos

Pregunta Guía: ¿cuáles son los datos desconocidos por el programador, es decir, a ser calculados por el programa?

"Valor del jornal del operario"

Constantes (datos que no pueden cambiar durante la ejecución de un programa o en futuras ejecuciones pero pueden cambiar sus valores con el transcurso del tiempo)

"Valores de costo por hora nocturno y diurno"

Variables (datos que pueden cambiar durante la ejecución de un programa o en futuras ejecuciones)

"Cantidad de horas trabajadas, código del turno, valor del jornal"

Columna **Estado Intermedio de Transformación**. ¿Qué procedimiento producirá los estados de transformación de los datos (valores de *recursos de entrada*) en resultados (valores de *recursos de salida*) en correspondencia con cada caso *representativo*? El estado final tracciona desde el estado inicial la solución del problema y el

estado intermedio de transformación resuelve la tensión entre el estado inicial y el estado final. Se ejemplifica para cada caso.

Tabla 2. Análisis del Problema.

ANÁLISIS del Problema			
Número de Caso	**Estado Inicial**	**Estado Final**	**Estado Intermedio de Transformación**
#1	código D horas 10	734,5 pesos	734,5 = 10 x 73,45
#2	código N horas 10	1102,5 pesos	1102,5 = 10 x 110,25

En Tabla 3: Columna **Estado Genérico de Adaptación**. ¿Cómo se pueden representar las operaciones aritméticas expresadas con *valores* en la columna Estado Intermedio de Transformación en función a los recursos del problema (datos conocidos, a definir y desconocidos) expresados con *identificadores*? Se observa el procedimiento matemático implementado y se expresan los valores numéricos involucrados en forma de datos para cada caso representativo. Se detecta la relación entre los datos que gobiernan los diferentes procedimientos para iniciar el proceso de generalización hacia el hallazgo del algoritmo. Se ejemplifica para cada caso. Columna **Descomposición del Problema en Subproblemas**. ¿Cuál sería una secuencia lógica de subproblemas para obtener el resultado esperado cualesquiera sean los valores (posibles) proporcionados por el usuario? Se observan los procedimientos obtenidos en la columna Estado Genérico de Adaptación y se escudriña cuál es la relación entre los datos de entrada que condiciona la comunicación al procesador de un conjunto de órdenes distintas a ejecutar para avanzar hacia la adaptación computacional y obtener una solución general al problema planteado. Columna **Naturaleza de los Subproblemas**. ¿Cuál es la naturaleza de los subproblemas que constituyen el algoritmo? En general, se debe detectar si la naturaleza del subproblema es secuencial, selectiva o repetitiva, establecer cuál es la condición (simple o compuesta) que define una selección (simple o múltiple) o controla una repetición, establecer qué acciones se deben ejecutar dentro de las cláusulas selectivas (verdadero o falso) o en el cuerpo del ciclo indefinido. En este caso, se detectan dos opciones gobernadas por los valores posibles del código de turno que determinan uno u otro resultado. Se introduce el nuevo conocimiento, naturaleza del problema selectiva simple y surge la necesidad de uso de una nueva estructura de control. Columna **Primitivas de Programación**. ¿Con qué tipo de estructuras de control básicas se implementará el al-

goritmo? Se debe identificar dentro de las estructuras de control secuenciales a sentencias de *lectura o entrada* (input), de *asignación* (=) y de *escritura o salida* (print); de las selectivas, *simples* (if <condición>…else…) o *múltiples* (if <condición>…elif…else…) y de repetición indefinida (while <condición>…). En este momento de la clase, se trabaja con los estudiantes en la pizarra "Un Lugar para Programar" y se completa el texto del programa en el Modelo de Programa Tipo (ejemplo en Tabla 4). Se produce la transformación del algoritmo diseñado en la pizarra "Un Lugar para Algoritmiar" en programa codificando los enunciados algorítmicos a través de sentencias en lenguaje Python.

A continuación en la pizarra "Un Lugar para Programar" que contiene el texto del Modelo de Programa Tipo diseñado por el profesor responsable se completan las secciones declarativa y algorítmica (en las que se definen el objetivo y los recursos del programa), se codifica el algoritmo (los enunciados algorítmicos se convierten a instrucciones en lenguaje Python) y se diseñan los juegos de datos de prueba del programa en la sección evaluativa. Se ejecuta una versión del programa (previamente escrita por el docente en el entorno de desarrollo y aprendizaje IDLE Python) con los juegos de datos de prueba diseñados para completar el proceso de creación de un programa.

4.1 Aula virtual

En correspondencia con la segunda clase presencial, durante la semana, los estudiantes en pequeños grupos de 2 o 3 integrantes trabajan en talleres propios habilitados en Google Drive, "Un Lugar para Algoritmiar", para resolver juntos problemas propuestos como actividades formativas grupales de entrega obligatoria para el logro de la habilidad de descubrir algoritmos, en este ejemplo, con secuencias de subproblemas de naturaleza *secuencial* y selectiva *simple*. El aula real se expande en el aula virtual y se proporciona así la plataforma necesaria para hacer visible el pensamiento de los estudiantes. En el taller de cada grupo de trabajo, el profesor responsable enuncia un nuevo problema, como por ejemplo: Desarrollar un programa en lenguaje Python que solicite al usuario el ingreso desde el teclado de los valores de dos fechas distintas, antigua (a1, m1) y reciente (a2, m2), expresadas en años y meses y calcule la diferencia de fechas (difa, difm), expresada en años y meses. Informar por pantalla el valor de la diferencia de fechas (no se visualizarán valores nulos). Considerar que los años y los meses son unidades de tiempo atómicas (indivisibles) y que el usuario no comete error de tipeo en el ingreso de los datos. Y presenta tablas a los estudiantes para sistematizar las fases de análisis (comprender el

problema) y diseño (definir recursos y descubrir algoritmo), es decir, ofrece una estructura que orienta y hace visible el pensamiento y establece un patrón de comportamiento para la rutina *Conectar-Ampliar-Desafiar*. Esta rutina ayuda a los estudiantes a conectar las ideas y tomar conciencia de los enigmas que se pueden hacer visible al leer, observar, cuestionar, escribir, atender, representar, dialogar, participar, interactuar, comunicar, construir, experimentar el pensar. Promueve el aprendizaje activo porque está diseñada para facilitar el procesamiento activo de nueva información.

Cada integrante del grupo elige un color para identificar sus aportes. En una acción conjunta de responsabilidad grupal los alumnos completan las tablas que se presentan. Se produce el descubrimiento de un algoritmo-solución por sucesivas devoluciones del docente y depuraciones de los alumnos en una trama educativa compuesta por procesos didácticos, comunicativos y colaborativos. Las contribuciones de los estudiantes permiten visibilizar el pensamiento individual y grupal (pensar para aprender) y las intervenciones del profesor responsable orientan, cuestionan, encauzan, movilizan el pensamiento para el logro de un algoritmo-solución del problema (aprender a pensar algorítmicamente). Se visualiza en cada taller una trama cromática de pensamiento conjunto (cada alumno consigo mismo, alumno con sus pares, docente con alumnos). Se ejemplifica la sistematización del trabajo en aula virtual en las Tablas 5, 6, 7 y 8.

Tabla 3. Diseño de la Solución.

DISEÑO de la Solución			
Estado Genérico de Adaptación	**Descomposición del Problema en Subproblemas**	**Naturaleza de los Subproblemas**	**Primitivas de Programación**
#1 jornal=horas x hora_diurna (procedimiento para turno diurno)	# obtención de código de turno y cantidad de horas trabajadas	# secuencial	# sentencia de entrada (input)
#2 jornal=horas x hora_nocturna (procedimiento para turno nocturno)	# cálculo del jornal del operario según turno de trabajo Si el turno es diurno	# selectiva simple	# sentencia de selección simple (if <condición>: …

	jornal = horas x hora_diurna en caso contrario jornal = horas x hora_nocturna		else: …)
	#exhibición valor del jornal	# secuencial	# sentencia de salida (print)

Tabla 4. Modelo de Programa Tipo.

```
#SECCIÓN DECLARATIVA (Definición de Recursos)
#SECCIÓN ALGORÍTMICA (Desarrollo de Solución)
#Nombre del programa
#Síntesis del enunciado (qué problema se resuelve)
#Autor, fecha, versión
#1 PRÓLOGO
#1.1 Presentación
#1.1.1 Título
#1.1.2 Bienvenida
#1.1.3 Objetivo
#1.2 Obtención de recursos
#1.2.1 Solicitud e ingreso de recursos desde teclado
#1.2.2 Definición de recursos (conocidos o intermedios)
#2 RESOLUCIÓN
#2.1 Cálculo de la solución del problema
# (Descomposición del problema en subproblemas)
# 3 EPÍLOGO
# 3.1 Muestra de solución del problema por pantalla
# 3.2 Finalización
# 3.2.1 Despedida
# 3.2.2 Agradecimiento
# 3.2.3 Pausa para lectura de resultados
# SECCIÓN EVALUATIVA
```

(Diseño de datos de prueba)
Juego Nº Recursos (valores)
1 …
…

5 Reflexiones Preliminares

Las fases de análisis y diseño, algorítmicas en el proceso de construcción de programas, se sistematizan en las tablas que ofrecen una organización para que los estudiantes externalicen sus procesos de pensamiento y puedan tener mayor dominio sobre ellos. La columna Estado Intermedio *conecta* con las anteriores (Estado Inicial y Estado Final), la columna Estado Genérico se resuelve en base a la columna Estado Intermedio y *amplía* el pensamiento para iniciar la adaptación algorítmica del procedimiento matemático y en la columna Descomposición del Problema se *desafía* al pensamiento para organizar la secuencia de subproblemas que constituye el algoritmo-solución. La rutina de pensamiento *Conectar-Ampliar-Desafiar* está embebida en las tablas que estructuran el pensamiento computacional.

Se encuestó a 23 estudiantes participantes en relación a cómo influyó en el descubrimiento de un algoritmo el hacer visible el pensamiento. Las respuestas obtenidas fueron: “Ayudó a representar el problema para validar el resultado y poder ingeniar los estados genéricos de adaptación” (34,8%); “Posibilitó el hacer conciente la forma de pensar el problema y favoreció la descomposición del problema en subproblemas” (56,5%); “Sirvió para abordar la búsqueda de otras soluciones con mayor facilidad y confianza” (39,1%); “Permitió codificar un algoritmo de mejor calidad” (26,1%).

La herramienta tecnológica Google Drive soporta el pensamiento visible y el pensamiento computacional que se despliega en el descubrimiento de un algoritmo solución y provee una plataforma estable y flexible que potencia la actividad conjunta desplegada por el profesor responsable y los estudiantes en torno al contenido de aprendizaje en el aula virtual e impacta sobre el control de la calidad de diseño del algoritmo en el proceso de construcción de un programa.

Tabla 5. Análisis del Problema (COMPRENDER el problema).

¿Qué problema se debe resolver? (considerar la computadora como herramienta que se utiliza para modelizar, es decir, representar una situación del mundo real en forma de datos a procesar para generar resultados) *Preguntas Guía:* ¿cuál es la incógnita del problema? o ¿qué solicita el enunciado

informar por pantalla? o ¿qué resultado(s) se debe(n) obtener? o ¿cuál es el objetivo del programa?
"Calcular la diferencia entre dos fechas, una antigua y otra reciente (en meses y años)"

Tabla 6. Diseño de la Solución (IDEAR un plan: IDENTIFICAR recursos).

¿Cuáles son los recursos (objetos de programación a ser manipulados por las instrucciones) **necesarios para resolver el problema?**
Datos conocidos
Pregunta Guía: ¿cuáles son los datos conocidos por el programador, es decir, propios del problema y/o a ser proporcionados por el usuario?
"a1, m1, a2, m2"
Datos desconocidos
Pregunta Guía: ¿cuáles son los datos desconocidos por el programador, es decir, a ser calculados por el programa?
"La diferencia en meses y años entre las dos fechas"
Constantes (datos que no pueden cambiar durante la ejecución de un programa o en futuras ejecuciones pero pueden cambiar sus valores con el transcurso del tiempo)
"No hay constantes "
Variables (datos que pueden cambiar durante la ejecución de un programa o en futuras ejecuciones)
"a1, m1, a2, m2, difa, difm"

Tabla 7. Diseño de la Solución (IDEAR un plan: DESCUBRIR el algoritmo).

ANÁLISIS del Problema (trabajar con valores de los datos para solucionar el problema en el mundo real).

Número de Caso	**Estado Inicial**	**Estado Final**	**Estado Intermedio de Transformación**
#1	4/2010 8/2018	8 años 4 meses	8 = 2018 - 2010 4 = 8 - 4
#2	8/2010 4/2018	7 años 8 meses	7 = 2018 - 2010 - 1 8 = (12 - 8) + 4

Tabla 8. Diseño de la Solución (IDEAR un plan: DESCUBRIR el algoritmo).

DISEÑO de la Solución (representar los valores de los datos con identificadores de recursos, transformar los casos representativos específicos en casos genéricos inclusivos, organizar la secuencia de subproblemas que produce la adaptación computacional).

Estado Genérico de Adaptación	**Descomposición del Problema en Subproblemas**	**Naturaleza de los Subproblemas**	**Primitivas de Programación en Python**
#1 difa = a2 - a1 difm = m2 - m1 m1 <= m2	#obtención fechas antigua y reciente desde teclado #cálculo diferencia de fechas	#secuencial #selectiva simple	#sentencia de entrada (input) #sentencia de selección simple (if <condición>: … else: …)
#2 difa = a2 - a1 - 1 difm = m2 - m1+12 m1>m2	Si m1<=m2 difa=a2-a1 difm=m2-m1 en caso contrario difa=a2-a1-1 difm=m2-m1+12		
	#exhibición diferencia de fechas por pantalla	#secuencial	#sentencia de salida (print)

6 Trabajos Futuros

Se relató la experiencia educativa realizada en el módulo 1 durante el primer cuatrimestre en los cursos 6 y 9 de Computación. Se analizarán los datos recolectados a través de distintas técnicas (observación, entrevistas guiadas, registros en aula presencial y aula virtual, audios de clases, encuestas, fotografías de pizarras) en el proceso de enseñanza y aprendizaje de los módulos 1 y 2 para revisar, ajustar y mejorar la versión preliminar de la estrategia tecnopedagógica y se implementará la versión definitiva en el marco de un trabajo de investigación-acción cuyo propósito es desa-

rrollar competencias en algoritmia y programación como formación básica de los estudiantes de Ingeniería en la FIUBA.

7 Referencias

1. Grossi, M.D., Jiménez Rey, E., Servetto, A. y Perichinsky, G. (2003): Enseñanza de Computación en Carreras de Ingeniería. En IX Congreso Argentino de Ciencias de la Computación, 1252-1263, La Plata, Argentina.
2. Bruno, O. (2005): Análisis de la percepción de los alumnos y de los docentes para la incorporación de un sistema tutor inteligente como facilitador del aprendizaje de algoritmia. Revista de Informática Educativa y Medios Audiovisuales, 2 (4), 1-31.
3. Zapata Ros, M. (2015): Pensamiento Computacional: Una nueva alfabetización digital. Revista de Educación a Distancia, 46(4).
4. Wing, J. (2006): Computational Thinking. Comunications of the ACM, 49 (3), 33-35.
5. Wing, J. (2010): Computational Thinking: What and Why? Magazine of Carnegie Mellon University's School of Computer Science.
6. Simari, G. (2013): Los fundamentos computacionales como parte de las ciencias básicas en las terminales de la disciplina Informática. En VIII Congreso de Tecnología en Educación y Educación en Tecnología, Santiago del Estero, Argentina.
7. Morin, E. (1999): Los siete saberes necesarios para la educación del futuro, París: UNESCO.
8. Martínez, C. y Echeveste, E. (2017): Aprender a programar para integrar(nos). Córdoba: UEPC, ICIEC.
9. Jiménez Rey, E., Calvo, P. y Servetto, A. (2018): Sistematización del análisis como estrategia didáctica para el diseño de algoritmos en la formación básica de los ingenieros. III Congreso Internacional de Enseñanza de las Ciencias Básicas, UTN, Concordia.
10. Lerch, C. y de Vedia, L. (2014): El conocimiento tecnológico y el conocimiento ingenieril en la formación del ingeniero para un mundo cambiante. En La educación del ingeniero para un mundo cambiante, 49-78, 6, ANCEFN. CABA: Luis de Vedia Editor.
11. Swartz, R., Costa, A., Beyer, B., Reagan, R. y Kallick, B. (2008): La importancia del pensamiento eficaz. En El aprendizaje basado en el pensamiento. Cómo desarrollar en los alumnos las competencias del siglo XXI (pp. 7-47). Nueva York: Ediciones SM.
12. Proyecto Cero de la Escuela de Educación de Harvard (s. f.): Recuperado de: http://www.pz.harvard.edu/
13. Ritchhart, R., Church, M. y Morrison, K. (2014): Hacer visible el pensamiento. Ciudad Autónoma de Buenos Aires: Paidós.
14. Coll, C., Onrubia, J. y Mauri, T. (2007): Tecnología y prácticas pedagógicas: las TIC como instrumentos de mediación de la actividad conjunta de profesores y estudiantes. Anuario de Psicología, vol. 38, n° 3, 377-400. Facultat de Psicología. Universitat de Barcelona.