Cursos Program.AR de introducción a la programación y su didáctica I y II

1. Sobre estos cursos

1.1. Resumen de los cursos

Estos cursos cubrirán una introducción a la programación y su didáctica, a la vez que algunos primeros elementos de algorítmica y estructuras de datos. Su objetivo es **proveer formación a docentes secundarios con título de Profesor de Informática u otros, que deseen tener un primer acercamiento tanto a la programación como a su enseñanza**.

La modalidad de los cursos es semipresencial y la carga horaria es de 70 horas presenciales cada uno. Con un funcionamiento dinámico, en las clases se realizarán permanentemente actividades que permitirán a los participantes incentivar a sus alumnos a que se animen a ser creadores de programas y no sólo usuarios de aplicaciones hechas por terceros.

La estrategia pedagógica está basada principalmente en el **aprendizaje basado en problemas** (Torp, 1998) y en las metodologías de enseñanza de programación propuestas por la Universidad Nacional de Quilmes (Martínez López, 2012), la Universidad Nacional de Córdoba[1], y el equipo de expertos de la Fundación Dr. Manuel Sadosky.

1.2. Características innovadoras

Dentro de la formación profesional docente es muy frecuente encontrar cursos cuyas propuestas resultan muy difíciles de llevar al aula. **Estos cursos buscan evitar esas dificultades mediante una serie de características innovadoras**:

- **Para aprobar el curso el docente debe aplicar lo aprendido con sus propios alumnos, debiendo dictar 8 horas de clase de programación**, 4 con apoyo de los docentes universitarios que imparten este curso, 4 de manera autónoma. De esta manera, las dudas, temores y dificultades que podrían desalentar la puesta en práctica de la innovación son trabajados como un elemento más de los cursos. Al finalizar los cursos, el docente cuenta con experiencia en la aplicación áulica de la innovación que acaba de aprender.
- Los docentes que deseen participar **deberán hacerlo de a pares**, junto con algún colega con quien compartan institución de trabajo. Está documentado que tener apoyo en la misma institución permite contar con un apoyo inmediato cuando surgen dudas o dificultades, lo que incrementa fuertemente las chances de que la innovación sea sostenible en la escuela.
- **Involucramiento de los directivos**. Para incrementar el apoyo institucional y minimizar las objeciones que pueden surgir de la propia institución cuando el docente desea introducir la innovación en la escuela, es clave involucrar a los directivos desde el primer momento. En este curso se los invita a participar de eventos tanto al comienzo como al final del curso.

[1] http://masmas.unc.edu.ar/

- Se generan "**comunidades de aprendizaje**", que son grupos que se reúnen en forma estable a lo largo del tiempo, lo que permite intercambiar experiencias y recursos, y contar con una red de apoyo que facilita enfrentar las dificultades que pudiesen surgir.

2. Fundamentación

La enseñanza de Ciencias de la Computación tiene un enorme potencial tanto para el desarrollo de la curiosidad y la creatividad en los chicos, como para desarrollar talento en el país, convirtiéndolo en un actor cada vez más relevante en la revolución de las Tecnologías de la Información y la Comunicación (TIC).

Las TIC abarcan todos los ámbitos de la experiencia humana. Están en todas partes y modifican los ámbitos de las actividades cotidianas: el trabajo, las formas de estudiar, las modalidades para comprar y vender, los trámites, el aprendizaje y el acceso a la salud. Este grupo de conocimientos y herramientas son directamente aplicados bajo la forma de sistemas de información y redes de comunicación, en mayor parte digitales, con el fin de satisfacer necesidades específicas de distintos usuarios. Dada su creciente ubicuidad y su relevancia social, es de vital importancia conocer los fundamentos básicos que los definen. "*Enseñar el pensamiento computacional no solamente podría inspirar a las generaciones futuras a entrar en el campo de las Ciencias de la Computación dada la aventura intelectual, sino que beneficiaría a la gente en todos los campos*"[2].

Según coinciden distintos autores (véase, por ejemplo, Berners-Lee, 2013), "*Saber programación es la nueva brecha digital*", aún superando una primera brecha digital vinculada al acceso, otra se instalará en el futuro cercano: la de aquellos que tienen o no conocimientos de ciencias de la computación. "*Aunque las tecnologías individuales cambian día a día, están fundadas sobre conceptos y principios que han perdurado por décadas. Mucho tiempo después de que los estudiantes terminen la escuela y comiencen a trabajar –mucho después que las tecnologías que veían en la escuela sean obsoletas– los principios que aprendieron en Ciencias de la Computación todavía serán válidos.*"[3]

Si bien las Ciencias de la Computación son más que la programación de computadoras, ciertamente la programación es fundamental para estas ciencias. Desde lo educativo, la programación fomenta la creatividad, el pensamiento lógico, la precisión en la resolución de problemas, y permite desarrollar un aprendizaje y pensamiento requerido actualmente por otras materias en las escuelas. Es por esto que debería ser impartida en las escuelas, siendo uno de los puntos clave a ser tomados en cuenta para un aprendizaje significativo de lo que las Ciencias de la Computación tienen para ofrecer a los estudiantes[4].

Si bien se comenzará enseñando conceptos básicos de programación, también se realizarán actividades en las cuales los chicos deberán programar para resolverlas (en algunos casos incluso sin computadoras).

---

[2] Computational Thinking. Jeannette M. Wing, CACM, vol. 49, no.3 March 2006, pp. 33-35
[3] Michael Gove, Secretario de Estado para la Educación del Reino Unido, 2012.

[4] "Informe CC 2016. Una propuesta para refundar la enseñanza de la computación en las escuelas argentinas", 2012, Fundación Sadosky, Buenos Aires

Elegimos abordar los conceptos básicos de la programación (y no un lenguaje o software en particular) porque aprender computación no es sólo usar un software, sino que la computación se basa en conceptos básicos desarrollados mucho antes que los software específicos y que seguirán siendo relevantes cuando nuestros jóvenes de hoy estén en el mercado laboral (como la idea de funciones, variable, iteraciones, ciclo, bucle, autómata, robótica, bases de datos, redes, etc). Es por eso que elegimos plataformas educativas diseñadas especialmente para que los jóvenes aprendan conceptos básicos de computación, como Alice o Scratch.

A partir de la enseñanza de estrategias didácticas concretas para enseñar programación, abordamos la necesidad que tienen los docentes de conocer no sólo qué contenidos es necesario enseñar, sino también cómo acercar a los jóvenes a esos contenidos de una manera significativa. Buscamos, como dice Alliaud (2007), crear nuevas experiencias de aprendizaje en los docentes porque lo vivido, en términos de prácticas de la enseñanza, tiene más peso de aquello “que te dicen” teóricamente en un curso de formación. De esta forma, los docentes podrán llevar estas experiencias significativas a las aulas.

La evaluación consistirá en la resolución de algunos ejercicios, pero principalmente en el dictado de parte de lo aprendido a los propios alumnos, con supervisión y apoyo de los capacitadores universitarios, que ayudarán a cada docente a preparar su plan de clases. Esta propuesta se discute desde el principio, con la presencia de los directivos de la institución a la que el docente representa, de manera tal de que todos los actores están al tanto desde el comienzo y se facilita la llegada al aula. Así, al finalizar el curso el docente no tiene solamente unos conocimientos teóricos sino también la experiencia de haberlos ya impartido y la posibilidad de haber reflexionado sobre dicha experiencia con sus propios mentores.

Como resultado de este proyecto esperamos que los docentes cuenten con una capacitación adecuada para encarar un curso de enseñanza de Ciencias de la Computación. El proyecto asimismo incluye un conjunto de actividades para que el docente cuente en el marco de su curso y, por otra parte. Las transparencias y cuadernillos de actividades utilizadas en el curso quedarán como un activo a disposición de la provincia para su uso en futuros cursos de capacitación.

3. Curso I

3.1. Objetivos

- Comprender la importancia de enseñar Ciencias de la Computación para la formación de los ciudadanos del presente.
- Conceptualizar la noción de programa y mostrar que con éstos es posible representar ideas y resolver problemas.
- Entender que las computadoras sirven para ejecutar programas y realizan sólo lo que un programa indique.
- Incentivar a los alumnos a ser creadores de programas y no sólo usuarios de aplicaciones hechas por terceros.
- Ejecutar programas diseñados por los propios alumnos.
- Detectar y corregir errores de los programas propios y de los alumnos.

- Planificar la solución a un problema de programación como la división en subproblemas, e identificar a estos.
- Comprender el desarrollo de la capacidad de abstracción que permiten las nociones de estado de un programa, identificación de patrones, repetición fija o condicional y parámetros.
- Trabajar conceptos relacionados con las Ciencias de la Computación para desarrollar habilidades de pensamiento.
- Diseñar propuestas áulicas creativas aplicando los conocimientos trabajados.
- Evaluar los aprendizajes de los estudiantes.

3.2. Contenidos

- Ciencias de la Computación. Informática. TIC. Diferencias. Importancia de enseñar Ciencias de la Computación.
- Comandos (acciones) y valores (datos). Procedimientos. Programas. Noción de programa y autómata.
- División en subtareas. Repeticiones simples. Alternativas condicionales. Repeticiones condicionales. Parámetros.
- Procedimiento de síntesis del objetivo de un programa. Procesamiento de estructuras lineales.
- Uso de las herramientas Alice, Scratch y Lightbot.
- Programación básica de robots educativos: movimientos, lectura de sensores y prendido y apagado de actuadores.
- Identificación de patrones, repetición fija o condicional y parámetros.
- Resolución de problemas. Modos de abordaje: un mismo problema puede ser resuelto por distintos programas. Entre dos programas que resuelven un mismo problema, uno puede comunicar la solución en forma más sintética y clara que otro.
- Diseño de actividades de programación típicas sobre estructuras lineales, incluyendo la selección de operaciones primitivas del sistema de cómputo elemental. Procedimientos creativos.
- Metodología para la corrección de errores del programa analizando la diferencia anterior.
- Planificación de la solución a un problema de programación. Identificación de subproblemas.
- Adaptación de contenidos a distintas habilidades y conocimientos previos de los estudiantes.
- Formas de evaluar los aprendizajes. Instrumentos de evaluación.

4. Curso II

4.1. Objetivos

- Aprender a realizar programas sencillo que procesen datos y/o computen información.
- Abordar problemas tecnológicos/informáticos relacionados a la experiencia cotidiana.
- Dotar al docente cursante de conocimientos que le den una base sólida para ayudar a sus alumnos en proyectos escolares de programación.

- Resolver desafíos de programación no triviales, entendidos como aquellos en que su estrategia de solución no es obvio o directa.
- Dotar al docente cursante de experiencia práctica en varias herramientas de programación escolar.
- Detectar y corregir errores de los programas propios y de los alumnos.
- Planificar la solución a un problema de programación como la división en subproblemas, e identificar a estos.
- Comprender el desarrollo de la capacidad de abstracción que permiten las nociones de estado de un programa, identificación de patrones, repetición fija o condicional y parámetros.
- Diseñar propuestas áulicas creativas aplicando los conocimientos trabajados.
- Evaluar los aprendizajes de los estudiantes.

4.2. Contenidos

- Noción de sensor.
- Noción de función.
- Variables, concepto y usos específicos (eg, acumuladores).
- Introducción a los datos y las estructuras de datos.
- Introducción a la noción de representación.
- Manejo de secuencias.
- Manejo de tuplas.
- Algoritmos sencillos sobre secuencias.
- Recorridos sobre secuencias para implementar filtros (totalizaciones, selecciones).
- Algoritmos simples de ordenamiento.
- Búsqueda secuencial y búsqueda binaria.
- Uso de las herramientas AppInventor y Gobstones Web.
- Trabajo con secuencias didácticas "unplugged".
- Resolución de problemas. Modos de abordaje: un mismo problema puede ser resuelto por distintos programas. Entre dos programas que resuelven un mismo problema, uno puede comunicar la solución en forma más sintética y clara que otro.
- Metodología para la corrección de errores del programa analizando la diferencia anterior.
- Planificación de la solución a un problema de programación. Identificación de subproblemas.
- Adaptación de contenidos a distintas habilidades y conocimientos previos de los estudiantes.
- Formas de evaluar los aprendizajes. Instrumentos de evaluación.

5. Aspectos metodológicos

Ambos cursos constarán de instancias presenciales y no presenciales.

Las instancias presenciales se llevarán a cabo en la sede de la universidad participante. Los docentes trabajarán de manera dinámica en las clases, y durante las horas de clase realizarán

más de 40 actividades que luego podrán implementar directamente en sus aulas. Se trata de actividades que se resuelven mediante la programación, algunas incluso sin computadoras.

La estrategia pedagógica estará basada principalmente en el aprendizaje basado en problemas (Torp, 1998) y en las metodologías de enseñanza de programación propuestas por la Universidad Nacional de Quilmes (Martínez López, 2012) y la Universidad Nacional de Córdoba[5].

Las clases se dictarán en dos modalidades distintas:

- Cuando se desee favorecer la presencia de docentes de ciudades lejanas el curso se dictará en dos jornadas de 7 hs a realizarse un viernes y un sábado al mes.
- Cuando se desee optar por un modo de cursada menos compacto, se realizarán dos encuentros semanales de dos horas.

En todos los casos se habilitará un aula virtual para seguimiento y tutoría, de manera que los docentes puedan realizar consultas a los capacitadores relacionadas con las actividades no presenciales. También podrán tener acceso a bibliografía y subir los trabajos para ser evaluados.

La carga total de cada curso será de 100 hs reloj, de las cuales 70 hs reloj serán destinadas a las presenciales (10 clases de 7 hs reloj en una modalidad, 30 clases de 2 hs, más 2 clases de 4 hs en la otra) y 30 hs reloj a los no presenciales.

Los docentes que deseen participar deberán hacerlo de a pares, junto con algún colega con quien compartan institución de trabajo. Está documentado que tener apoyo en la misma institución permite contar con un apoyo inmediato cuando surgen dudas o dificultades, lo que incrementa fuertemente las chances de que la innovación sea sostenible en la escuela.

Para incrementar el apoyo institucional y minimizar las objeciones que pueden surgir de la propia institución cuando el docente desea introducir la innovación en la escuela, es clave involucrar a los directivos desde el primer momento. En este curso se los invita a participar de jornadas institucionales al comienzo y al final del curso.

Para aprobar cada curso el docente debe aplicar lo aprendido con sus propios alumnos, debiendo dictar 8 horas de clase de programación, 4 con apoyo de los docentes universitarios que imparten este curso, 4 de manera autónoma. De esta manera, las dudas, temores y dificultades que podrían desalentar la puesta en práctica de la innovación son trabajados como un elemento más del curso. Al final del curso, el docente ya tiene experiencia en la aplicación áulica de la innovación que acaba de aprender.

Se generan "comunidades de aprendizaje", que son grupos que se reúnen en forma estable a lo largo del tiempo, lo que permite intercambiar experiencias y recursos, y contar con una red de apoyo que facilita enfrentar las dificultades que pudiesen surgir.

[5] http://masmas.unc.edu.ar/

6. Aspectos organizativos

6.1. Destinatarios

Los cursos están dirigido a docentes de nivel primario y profesores del nivel secundario de todas las disciplinas. Se dará prioridad a aquellos cuyo título de base esté relacionado con la Tecnología, la Informática y/o la Matemática. También tendrán prioridad quienes estén a cargo del dictado de horas de materias tecnológicas.

6.2. Cronograma tentativo

El curso dura cuatro meses que pueden organizarse de dos maneras alternativas:

1. Un viernes y un sábado al mes de 7 hs cada día (esquema "mensual").
2. Dos días a la semana, dos horas por día (esquema "semanal").

A modo de ejemplo se presenta un cronograma posible para el curso I:

	Tema	Observaciones	Horas	Nro clase esquema semanal	Nro clase esquema mensual
Evento 1	Presentación de Program.AR y el curso		1,5	1	1
	Noción de autómata (+ejercicio)		0,5	1	1
	Juego con robots		2	1	1
Subtareas	Presentación herramienta de programación y retomar idea de autómata. División en subtareas		2	2	1
	Lightbot		2	3	1
Repetición	Repetición fija (Alice + unplugged)		2	4	2
Ejercitación	Retomar división subtareas (problemas más complejos)	Recorrido de una estructura	6	5 - 7	2
Condicional	Condicional	Decisiones frente a diferentes escenarios iniciales	3	8	3

	Condicional + Repetición	Recorrido de una estructuras iniciales desconocidas	4	9 - 10	3
Repetición condicional	Repetición condicional	Recorrido de una estructura de longitud desconocida o procedimientos de búsqueda	3	11 y 12	4
Evento 2	Mostrar estudiantes + charla motivadora para todos		4	13	4
Ejercitación	Combinaciones de los anteriores	Recorrido de una estructura de longitud desconocida o procedimientos de búsqueda	5	14 y 15	5
Parametriza-ción	Expresiones / lectura variables	Saber que existe una variable, leerla	2	16	5
	Parametrización (Intro canciones)		4	17 y 18	6
	Parámetros múltiples		2	19	6
	Parametrización con repetición		6	20 - 22	7
Variables	Manejo de variables aisladas		2	23	8
	Manejo de variables en recorridos	contadores, acumuladores, etc	4	24 y 25	8
Interactividad	Intro Interactividad		2	26	9
	Intro a Lógica proposicional aplicada		2	27	9
	Diseño juegos		6	28 -30	9
Cierre	Cierre interno		1	31	10
Evento 3	Cierre		4	32	10

7. Evaluación

La evaluación consistirá en la resolución de algunos ejercicios, pero principalmente en el dictado de parte de lo aprendido a los propios alumnos, con supervisión y apoyo de los capacitadores

universitarios, que ayudarán a cada docente a preparar su plan de clases.

8. Ficha técnica

8.1. Curso I

Curso: “Programación y su didáctica - método Program.AR”
Modalidad: Semipresencial
Carga horaria: 100 hs reloj totales, 72 hs reloj presenciales.
Evaluación: con presentación de trabajos y dictado de clases.
Requisito de asistencia: 80%
Lugares de dictado:
Universidades de la Provincia de Buenos Aires y de la Ciudad Autónoma de Buenos Aires que resulten seleccionadas mediante convocatorias públicas aprobadas por disposiciones del Ministerio de Ciencia, Tecnología e Innovación Productiva de la Nación en el marco de la Iniciativa Program.AR.
Cupo: 70 inscriptos.
Destinatarios: Los cursos están dirigido a docentes de nivel primario y profesores del nivel secundario de todas las disciplinas. Se dará prioridad a aquellos cuyo título de base esté relacionado con la Tecnología, la Informática y/o la Matemática. También tendrán prioridad quienes estén a cargo del dictado de horas de materias tecnológicas.
Resumen: Este curso busca capacitar docentes secundarios en la enseñanza de las Ciencias de la Computación En particular, se busca que los docentes adquieran conocimientos básicos de programación, así como estrategias adecuadas para la enseñanza a alumnos del nivel mencionado. Con una modalidad dinámica, en las clases se realizarán permanentemente actividades que les permitirán incentivar a sus alumnos a que se animen a ser creadores de programas y no sólo usuarios de aplicaciones hechas por terceros.

8.2. Curso II

Curso: “Programación y su didáctica - método Program.AR”
Modalidad: Semipresencial
Carga horaria: 100 hs reloj totales, 72 hs reloj presenciales.
Evaluación: con presentación de trabajos y dictado de clases.
Requisito de asistencia: 80%
Lugares de dictado:
Universidades de la Provincia de Buenos Aires y de la Ciudad Autónoma de Buenos Aires que resulten seleccionadas mediante convocatorias públicas aprobadas por disposiciones del Ministerio de Ciencia, Tecnología e Innovación Productiva de la Nación en el marco de la Iniciativa Program.AR.
Cupo: 70 inscriptos.
Destinatarios: Los cursos están dirigido a docentes de nivel primario y profesores del nivel secundario de todas las disciplinas que posean conocimientos básicos de programación. Se dará prioridad a aquellos cuyo título de base esté relacionado con la Tecnología, la Informática y/o la

Matemática. También tendrán prioridad quienes estén a cargo del dictado de horas de materias tecnológicas.

Resumen: Este curso busca capacitar docentes secundarios en la enseñanza de las Ciencias de la Computación En particular, se busca que los docentes adquieran conocimientos básicos de programación, así como estrategias adecuadas para la enseñanza a alumnos del nivel mencionado. Con una modalidad dinámica, en las clases se realizarán permanentemente actividades que les permitirán incentivar a sus alumnos a que se animen a ser creadores de programas y no sólo usuarios de aplicaciones hechas por terceros.

9. Sobre Program.AR y las Ciencias de la Computación

9.1. Sobre Program.AR

Program.AR es una iniciativa que trabaja en pos de contribuir a la enseñanza y el aprendizaje de las Ciencias de la Computación en las escuelas argentinas. Esta iniciativa incluye múltiples aspectos relacionados con la difusión y popularización de la disciplina, la generación de contenidos escolares la formación docente. Puede encontrarse más información sobre Program.AR en su sitio web: http://program.ar

Dado que se trata de un curso de programación, es importante entender la relación entre ésta y las Ciencias de la Computación en general.

9.2. Ciencias de la Computación y Programación

Las Ciencias de la Computación son las disciplinas que estudian la programación de computadoras, el funcionamiento de las mismas y las telecomunicaciones. Algunos de sus saberes troncales son:

- Los necesarios para poder formular soluciones efectivas y sistemáticas a diversos tipos de problemas. Por ejemplo: pensemos en un GPS. ¿Cuál camino debe sugerir a un usuario, entre todos los posibles, en un momento determinado y teniendo en cuenta las condiciones de tránsito? A esta área de la Computación se la conoce como **algorítmica**.
- La **programación**. Es decir, los conocimientos necesarios para poder volcar esas soluciones algorítmicas a los diversos lenguajes que utilizan las computadoras.
- Cómo almacenar la información de manera que pueda ser recuperada más adelante y que se pueda buscar velozmente un dato entre miles o millones de otros, como hacen por ejemplo los buscadores de Internet. Estos saberes se agrupan en dos áreas temáticas: **estructuras de datos y bases de datos**.
- Los **fundamentos teóricos** que marcan las diferencias entre los distintos lenguajes, sus posibilidades e imposibilidades, ventajas y desventajas.
- Las **arquitecturas de computadoras**. Nos referimos al entendimiento de los componentes que definen los distintos tipos de computadoras. También al entendimiento de cómo estos componentes se construyen a partir de la combinación de manipulaciones sencillas de voltaje eléctrico.
- Las **redes de computadoras**. Es decir, la forma en que las computadoras intercambian información permitiendo el funcionamiento de Internet y todas las aplicaciones que

funcionan gracias a Internet, como la web, la mensajería instantánea, los juegos en línea, las transmisiones de audio y video, etc.

10. Bibliografía

- Berners-Lee, T. (2013). *Saber programación es la nueva brecha digital, según Berners-Lee*. CIO. Lima. Retrieved from http://www.cioperu.pe/articulo/12237/saber-programacion-es-la-nueva-brecha-digital-segun-bernerslee/
- Torp, L., & Sage, S. (1998). *El Aprendizaje Basado en Problemas*. (E. Litwin, Ed.). Buenos Aires: Amorrortu.
- Bell, T., Witten, E., Fellows, M. (2008). *Computer Science Unplugged: Un programa de extensión para niños de escuela primaria*.
- Busaniche, B. (2007). *Alfabetización digital: las fronteras del aprendizaje y el control de la información*. In R. C. y D. Levis (Ed.), . Capital Federal: Prometeo.
- Dann, W., Cooper S. & Pausch, R. (2011). *Learning to Program with Alice*. Prentice Hall
- Levis, D. (2007). *Enseñar y Aprender con informática/ Enseñar y aprender informática. Medios Informáticos en la escuela Argentina*. In R. Cabello & D. Levis (Eds.), Medios Informáticos en la Educación a principios del siglo XXI (pp. 21–50). Buenos Aires-Argentina: Prometeo.
- Martínez López, P. E., Bonelli, E. A. & Sawady O'Connor, F.A. (2012). *El nombre verdadero de la programación. Una concepción de la enseñanza de la programación para la sociedad de la información*. Anales del 10mo Simposio de la Sociedad de la Información (SSI'12), dentro de las 41ras Jornadas Argentinas de Informática (JAIIO '12), 1–23. ISSN 1850-2830.
- Martínez López, P. E. (2013). *Las bases conceptuales de la programación. Una nueva forma de aprender a programar*. Creative Commons.
- Ministerio de Educación, Ciencia y Tecnología de La Nación, 2007, "*Informe y Recomendaciones de la Comisión Nacional para el Mejoramiento de la Enseñanza de las Ciencias Naturales y la Matemática*", Buenos Aires.
- Ministerio de Educación, Ciencia y Tecnología (2005), "*Núcleos de Aprendizajes Prioritarios de Matemática", para el Segundo Ciclo del Nivel Secundario*", Buenos Aires.
- Torp, L., & Sage, S. (1998). *El Aprendizaje Basado en Problemas*. (E. Litwin, Ed.). Buenos Aires: Amorrortu.
- Sawady O'Connor, F.A. & Factorovich, P. (2013). *Cuaderno Teórico de Scratch*. Fundación Sadosky
- Wing, J.M. (2006). *Computational thinking*. Communications of the ACM, 49(3), 33–35.