# Comparativa de Abordajes de Cursos Introductorios de Programación

Mauricio R. Dávila [1,2]

1. Programa de Maestría en Ingeniería de Sistemas de Información.
Escuela de Posgrado, Facultad Regional de Buenos Aires. Universidad Tecnológica Nacional. Argentina.
2. Laboratorio de Investigación y Desarrollo en Espacios Virtuales de Trabajo de la Licenciatura en Sistemas y de la Maestría en Sistemas de Información del Departamento de Desarrollo Productivo y Tecnológico de la Universidad Nacional de Lanús. Argentina.
davilamr.80@gmail.com

***Resumen*— El pensamiento computacional se basa en resolver problemas haciendo uso de conceptos de informática, desarrollar esta habilidad es la tarea principal de las asignaturas de introducción a la programación en las carreras universitarias. En general, aunque se han creado lenguajes cada vez más cercanos al lenguaje humano, la programación no resulta ser una materia intuitiva ni de fácil comprensión y, por lo tanto, suele tratarse de una asignatura con una alta tasa de deserción.**

**En este contexto, se presenta una revisión de los métodos de enseñanza, el formato de ejercitación y el material utilizado en las materias iniciales de programación en la educación superior, tanto en universidades de los Estados Unidos como de la Argentina. A lo largo del trabajo se buscará determinar el estado de la cuestión, comparar los casos de las distintas universidades a fin de establecer qué aspectos de los distintos abordajes favorecen el proceso de la enseñanza y poder así realizar un esbozo de solución que apunte a disminuir los índices de deserción en las materias introductorias de programación.**

***Palabras clave*— deserción, enseñanza, programación.**

## I. Introducción

El objetivo del presente trabajo es elaborar un estado de la cuestión sobre los distintos abordajes en la enseñanza de la programación, particularmente en el nivel inicial universitario.

Se llevará adelante el estudio de los abordajes en la enseñanza de la programación que actualmente se encuentran vigentes en dieciséis universidades, ocho en los Estados Unidos y ocho en la Argentina. Se busca de esta manera determinar si existen aspectos comunes en las distintas propuestas.

El pensamiento computacional se basa en resolver problemas haciendo uso de conceptos de informática, desarrollar esta habilidad es la tarea principal de las asignaturas de introducción a la programación en las carreras universitarias. En general, aunque se han creado lenguajes cada vez más cercanos al lenguaje humano, la programación no resulta ser una materia intuitiva ni de fácil comprensión y, por lo tanto, con frecuencia tiene altas tasas de deserción. Según Byrne y Lyons [7] existe una gran cantidad de estudiantes que logran competencias en otros temas y que no logran alcanzar el éxito en las materias relacionadas a la programación. La preocupación por los cursos iniciales de programación en el nivel universitario ha llevado a muchos investigadores a analizar las causas que subyacen en esta problemática desde distintos enfoques.

Algunos autores abordan el tema analizando los problemas de comprensión de conceptos por parte de los estudiantes, Bayman y Mayer [4] examinaron conceptos erróneos relacionados con las sentencias de programas escritos en el lenguaje BASIC, encontrando que muchos estudiantes tenían comprensiones erróneas o falsas ideas. Spohrer y Soloway [30] examinaron la causa de los errores cometidos al momento de programar buscando determinar si éstos son producto de conceptos erróneos acerca de la semántica de los lenguajes de programación, llegaron a la conclusión que los errores son más propensos a surgir de falencias en la lectura y el análisis de las especificaciones. Brito y Sá-Soares [6] sostienen que los alumnos tienden a sobreestimar su nivel de aprendizaje y esta percepción equivocada los lleva a fracasar.

En el trabajo de Leone et al. [17] se identifican múltiples causales que pueden llevar a la permanencia o abandono de un alumno y los agrupa según los siguientes factores:

i. Factores personales: características individuales como competencias desarrolladas, experiencias previas, vocación, limitaciones, dificultades.

ii. Factores estructurales: se consideran diversos elementos del ambiente universitario que pueden tener una importante influencia, como por ejemplo, medios utilizados; servicios brindados; infraestructura; sistemas informáticos.

iii. Factores académicos: se refieren a la propuesta formativa e incluye tanto las actividades curriculares como las prácticas docentes, reglamentos o actividades extracurriculares.

iv. Factores sociales: hacen a la relación con los restantes actores, dado que a partir del ingreso el estudiante genera un nuevo mapa de vínculos y relaciones.

Muchas son las hipótesis que tratan de explicar el fenómeno de la deserción en los cursos iniciales de programación en el nivel universitario y por tratarse de una problemática compleja ninguna de estas interpretaciones agota el tema. Otros autores proponen soluciones de distinta índole para mejorar alguno de los causales de la deserción.

Una gran cantidad de estudios muestran que los entornos de aprendizaje basados en el contexto mejoran la participación de los estudiantes [5][26][32], éstos se tratan de herramientas que le permiten al estudiante escribir código y observar de inmediato la ejecución. De esta manera, los estudiantes llegan a entender los conceptos abstractos de programación. Ejemplos de entornos de aprendizaje basado en el contexto son: Scratch [26] y Karel [5]. Otra herramienta de aprendizaje basado en el contexto que ha ganado mucha atención últimamente es el uso de robots educativos; un ejemplo es el uso del robot Lego MindStorms [16].

Otra solución propuesta es la instrucción por pares, una técnica pedagógica para aumentar la participación en clase, en ésta los alumnos comienzan respondiendo a una pregunta de selección múltiple de manera individual y una vez emitido su

voto lo discuten en grupos para consensuar un voto grupal [24][25][29].

Otra alternativa propuesta es la realización de prácticas de programación de manera colaborativa [10], éstas difieren principalmente en la forma y en el número de participantes que se asigna a la actividad. Dos ejemplos de estas prácticas son, la programación por parejas en la cual se busca promover la cooperación entre programadores [21][27] y la colaboración en equipos como puede ser Coding Dojo Randori [28] , existe un problema a resolver y un grupo de alumnos a los que se asigna la tarea, dos integrantes del grupo comienzan realizando programación de a pares y en intervalos regulares de tiempo uno de los dos integrantes es reemplazado, esta técnica provoca que todos los participantes deben prestar suma atención ya que desconocen en qué momento les tocará tomar el mando y continuar programando.

Al igual que existen una gran número de hipótesis que intentan explicar el fenómeno de la deserción en los cursos iniciales de programación en el nivel universitario, existen un gran número de soluciones que se han propuesto. Disminuir los índices de deserción en las materias introductorias de programación es uno de los grandes desafíos de las instituciones de educación superior, este trabajo busca a través de la identificación y comparación de los distintos abordajes poder plantear posibles soluciones que permitan contrarrestar este fenómeno.

Se llevará adelante el estudio de los abordajes en la enseñanza de la programación que actualmente se encuentran vigentes en dieciséis universidades, ocho en los Estados Unidos y ocho en la Argentina. Se busca de esta manera determinar si existen aspectos comunes en las distintas propuestas.

El primer paso implicó decidir cuáles serían los aspectos a ser relevados de cada abordaje, para ello se tomó como referencia las recomendaciones del plan de estudios de la tecnología informática de año 2013 elaborado por ACM/IEEE-CS denominado “Joint Task Force on Computing Curricula” [1] y se elaboró un dispositivo usado para recopilar los datos tomando no sólo en consideración factores tales como las competencias que debe alcanzar un alumno en un curso inicial de informática sino también las metodologías de enseñanza, la utilización de aulas virtuales y el tipo de ejercitación. Por esta razón, el segundo paso fue realizar el relevamiento de la información utilizando la técnica de análisis documental para luego poder realizar la comparación de los distintos abordajes. Estos insumos facilitaron la elaboración de la hipótesis que son el punto de partida para futuras líneas de investigación.

## II. Estado de la Cuestión

Se presenta el estado de la cuestión sobre los temas abordados en las materias iniciales de programación en distintas casas de estudio tanto de los Estados Unidos como de la Argentina. El material incluido en este trabajo se obtuvo a través de la técnica de análisis documental la cual permitió la recopilación de información proveniente de: trabajos de investigación, programas analíticos de las distintas materias, literatura recomendada y material disponible en las aulas virtuales.

### *A. Instituciones de los Estados Unidos*

La educación superior en los Estados Unidos comienza luego de que el estudiante ha completado su educación preparatoria, doce años de educación (básica y secundaria), o han aprobado el Examen General de Desarrollo Educativo (GED por sus siglas en inglés).

Históricamente, un College era una subdivisión de una Universidad. Por ejemplo, la Universidad de California está dividida en diferentes colegios: colegio de artes y ciencias, de educación, de leyes, etc. Sin embargo, actualmente muchos colegios no forman parte de una universidad y también muchas universidades no tienen colegios.

En los Estados Unidos, un estudiante que asiste a un College o University está estudiando en el nivel de “Undergraduate” en búsqueda de obtener un “Bachelor´s Degree” que en general tienen cuatro años de duración. Esta graduación significa que el estudiante adquirió sólidos conocimientos (a "batch" of knowledge) en un campo de estudio en particular y obtuvo un conocimiento general en el resto.

Durante los primeros dos años de estudio, existe una variedad de clases en diferentes asignaturas, que se conocen comúnmente como prerrequisitos: literatura, ciencia, ciencias sociales, arte, historia, etc. Estas les permiten adquirir conocimientos generales en una variedad de asignaturas antes de concentrarse en un campo de estudio específico.

No obstante, los estudiantes pueden iniciar sus estudios en busca de un título de licenciatura en un community college, a través de esta institución muchos alumnos completan los primeros dos años de prerrequisitos, pues allí obtienen un título de transferencia denominado “Associate´s Degree”, los más comunes son el A.A. (Associate of Arts) y el A.S. (Associate of Science), y con éste título los alumnos pueden realizar la transferencia a una Universidad o College para obtener finalmente la licenciatura deseada.

Por otro lado, cabe destacar que los Estados Unidos no tienen un Ministerio de Educación central, cada institución de educación superior se encarga de su propio currículo académico. Sin embargo, existen entidades de acreditación privadas que funcionan a nivel estatal o regional, estas establecen estándares académicos y confieren reconocimientos de excelencia académica a las instituciones.

Los tipos de entidades de educación superior de los Estados Unidos que integran este trabajo son las siguientes:

- COLLEGE O UNIVERSIDAD ESTATAL: Es financiada y administrada por un gobierno estatal o local. Cada uno de los 50 estados de EE.UU. opera como mínimo una universidad estatal y posiblemente varios colleges estatales.
- COLLEGE O UNIVERSIDAD PRIVADA: Estas escuelas están administradas en forma privada, sin participación de una entidad de gobierno. Con frecuencia, las universidades y los colegios privados estadounidenses son más pequeños que las instituciones públicas.
- COMMUNITY COLLEGES: Son instituciones de dos años que otorgan títulos intermedios (transferibles) y también certificaciones.
- INSTITUTO DE TECNOLOGÍA: Es una institución que ofrece como mínimo cuatro años de estudios en ciencia y tecnología. Algunos tienen programas de posgrado, mientras que otros ofrecen cursos de corto plazo.

#### *1) Creighton University*

La Universidad Creighton es una institución privada, católica, de la Compañía de Jesús, ubicada en Omaha, Nebraska, Estados Unidos, fundada en 1878 como Creighton College por los Jesuitas.

Introduction to Programming, es el primer curso en la secuencia de materias de programación, ofrece una

introducción a la resolución de problemas mediante el empleo de la programación.

La información aquí expuesta se obtuvo analizando el documento Joint Task Force on Computing Curricula [1] como así también el sitio web de la materia [8], que corresponde a la cátedra de David Reed.

El curso inicia enseñando conceptos de algoritmia y resolución de problemas utilizando el lenguaje gráfico Scratch (desarrollado en el Media Lab dependiente del Instituto Tecnológico de Massachusetts -MIT- por un equipo dirigido por Mitchel Resnick).

Desde el año 2011, luego de impartir los conceptos de algoritmia con Scratch se utiliza como lenguaje Python, anteriormente se enseñaba en Java utilizando el paradigma orientado a objetos. Se consideró que un lenguaje orientado a objetos como Java no era adecuado para los principiantes y, por tanto, no era el lenguaje ideal para los estudiantes que toman dicho curso. Inicialmente el paradigma empleado es el imperativo para luego brindar los fundamentos del paradigma orientado a objetos.

El curso se dicta dos veces por semana durante dos horas, ambas jornadas en laboratorio, integrando teoría con práctica. Los estudiantes completan de seis a ocho tareas, que implican el diseño e implementación de programas en Python, las cuales también pueden incluir en algunos casos un informe escrito en el que se analiza el comportamiento del programa.

La materia cuenta con un aula virtual bien estructurada de acceso público, la cual se limita a compartir material (presentaciones utilizadas en clase y ejercicios) con los alumnos. Se recomienda como material de consulta el libro "Python Programming In Context" de Brad Miller y David Ranum.

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en: el manejo de referencias y aliasing, en el uso de listas enlazadas, en la utilización de ordenamiento y la falta de ejercitación referida a la refactorización de código.

*2) Grinnell College*

Grinnell College es una universidad privada de artes liberales en Grinnell, Iowa, Estados Unidos fundada en 1846 por un grupo de congregacionalistas, un movimiento que surgió de las iglesias protestantes inglesas a finales del siglo XVI y a principios del XVII.

Functional problem solving, es el primero de tres cursos existentes en la carrera, cada uno de ellos busca introducir al alumno a un nuevo paradigma de programación comenzando con el paradigma funcional. El curso explora mecanismos de representación y manipulación de imágenes.

La información aquí expuesta se obtuvo analizando el documento Joint Task Force on Computing Curricula [1] como así también el sitio web de la materia [13] que corresponde a la cátedra de Henry M. Walker.

Es prácticamente nula la utilización de algoritmos ya que la materia sólo cuenta con una breve explicación referida a cubrir temas basados en recursividad.

El paradigma de programación empleado es el funcional y se utiliza como lenguaje Scheme.

Todas las clases son dictadas en el laboratorio y se enseña en un formato de taller colaborativo. Las clases pueden comenzar o terminar con una exposición teórica abierta al debate pero se busca por sobre todo aprovechar el tiempo de clases para trabajar en las computadoras. La materia cuenta con cuatro clases semanales de cincuenta minutos cada una. Los estudiantes trabajan de a pares en la resolución de ejercicios y se busca que estas parejas cambien semana a semana.

Existe un aula virtual destinada a la materia, en ella se puede encontrar tanto material (modo texto) referido a la teoría como así también información pertinente a las fechas de la cursada.

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en: Se explica superficialmente el uso de algoritmos apuntando solo a lo referido a recursividad, no se aborda el tema de operaciones de entrada/salida ni el de manejo de archivos, no se trabaja con tipos abstractos de datos ni manejo dinámico de memoria, no se utiliza el concepto de listas enlazadas, no se practican técnicas de refactorización de código y no se utiliza un entorno de desarrollo ya que la materia basa sus prácticas en escribir scripts en el programa de manipulación de imágenes Gimp.

*3) Harvard University*

Es una universidad privada ubicada en Cambridge, Massachusetts, Estados Unidos, es la institución de enseñanza superior más antigua de los Estados Unidos, fue fundada en 1637 con el nombre de New College o the college at New Towne. Cambió el nombre a Harvard College el 13 de marzo de 1639, en recuerdo a su benefactor John Harvard.

Introduction to the intellectual enterprises of computer science and the art of programming, también conocido como CS50 enseña a los estudiantes a pensar algorítmicamente y resolver problemas de manera eficiente. Un rasgo a destacar del curso es la cantidad de lenguajes utilizados, se incluyen C, PHP, JavaScript, SQL, CSS y HTML.

Es importante mencionar que la cátedra a cargo del profesor David J. Malan lleva casi una década en la universidad de Harvard y que desde el año 2015 se comenzó a dictar de manera simultánea en la universidad de Yale. La única diferencia entre las dos clases es el nombre del curso, en Yale CS50 se conoce como CPSC100. La colaboración sin precedentes permite a los estudiantes de ambas escuelas ver vídeos en directo por streaming o archivados de las clases teóricas semanales, que tienen lugar sobre todo en Harvard y en ocasiones en Yale.

La información aquí expuesta se obtuvo analizando el documento [18] y de los contenidos del sitio web de la materia [15] que corresponde a la cátedra de David J. Malan.

En la semana inicial del curso se utiliza el lenguaje gráfico Scratch para impartir los conceptos elementales de algoritmia.

Una vez impartidos los conocimientos básicos de algoritmia empleando Scratch se comienza a trabajar utilizando el paradigma imperativo con el lenguaje C, se trabaja con éste en el período que transcurre desde la semana dos y la semana siete, luego se imparten conceptos referidos a la programación web utilizando como lenguaje PHP y por último JavaScript.

Este es un curso que se encuentra conformado por dos clases teóricas de una hora cada una por semana, es importante destacar que todas las conferencias son transmitidas en vivo en línea y quedan disponibles una vez finalizada la transmisión, inmediatamente después de finalizar. Adicionalmente los alumnos cuentan con un espacio denominado "Sections" de noventa minutos semanales, en él los colaboradores de la materia arman grupos reducidos de estudiantes a fin de poder ayudarlos con los temas abordados de esa semana. Por último existe un espacio dos veces por semana de cuatro horas donde los alumnos pueden asistir a trabajar con las guías prácticas y allí ser asistidos por los ayudantes. La materia cuenta con nueve guías de ejercicios cada una de ellas en dos modalidades de dificultad, semana a semana el estudiante puede elegir que guía resolver, en el caso de optar por la denominada "Hack Edition" podrá obtener puntos adicionales. La materia también

cuenta con un trabajo integrador el cual debe ser desarrollado aplicando tecnologías web.

La materia dispone de un aula virtual de acceso público la cual cuenta con las grabaciones de todas las clases teóricas, el material utilizado en las mismas, guías de ejercicios y literatura recomendada. Adicionalmente una vez expirada la fecha de entrega de cada guía de ejercicios los alumnos disponen un video denominado "autopsia", en estos videos encontraran una detallada descripción de la solución de los ejercicios de la guía en cuestión.

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en lo referido a refactorización de código.

*4) Harvey Mudd College*

Es una universidad de artes liberales, residencial y privada. Enfocada en la enseñanza de la ciencia, la ingeniería y las matemáticas, fundada en 1955 y situada en Claremont, California, Estados Unidos.

Introduction to Computer Science, todos los estudiantes de primer semestre en Harvey Mudd College toman este curso, el mismo no tiene requisitos previos y se ofrece en tres versiones distintas:

- Gold: pensada para los estudiantes que no tienen experiencia previa en programación.
- Black: es para los estudiantes con alguna experiencia.
- Green: es una versión pensada para los alumnos de Biología.

La información aquí expuesta se obtuvo analizando los documentos: Joint Task Force on Computing Curricula [1] como así también el sitio web de la materia [14] que corresponde a la cátedra de Zachary Dodds.

La materia imparte los conocimientos elementales de algoritmia en un lenguaje de programación simple denominado Picobot el cual controla un robot inspirado en el robot aspirador Roomba, este cuenta con un sitio web que oficia de entorno.

Una vez impartidos los conocimientos esenciales de algoritmia utilizando Picobot se comienza a trabajar con Python, en un enfoque multiparadigma el cual se denomina “Breadth-First”.

Este es un curso que se encuentra conformado por dos conferencias de setenta y cinco minutos por semana y un laboratorio opcional, pero que atrae a más del 90% de los estudiantes a una sesión suplementaria de dos horas a la semana. Las tareas contienen uno o más problemas "individuales" que cada estudiante debe completar por su cuenta, el resto de los problemas, es posible completarlos junto a otro estudiante.

La materia cuenta con un aula virtual de acceso público, la cual se limita a compartir material (presentaciones utilizadas en clase) con los alumnos y con un foro de acceso restringido solo a los alumnos matriculados en el sitio piazza.com .

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en que: no se aborda el tema de operaciones de entrada/salida ni el de manejo de archivos, no se utilizan referencias y aliasing (diferentes nombres simbólicos en el programa.), no se utiliza el concepto de listas enlazadas, no se practica técnicas de refactorización de código y no se hace hincapié en la documentación ni en las reglas de estilo.

*5) Massachusetts Institute of Technology*

El Instituto Tecnológico de Massachusetts (MIT por las iniciales de su nombre en idioma inglés) es una universidad privada localizada en Cambridge, Massachusetts, Estados Unidos. Fundada en 1861 en respuesta a la creciente industrialización de los Estados Unidos, utilizó el modelo de universidades politécnicas e hizo hincapié en la instrucción de laboratorio.

Introduction to Computer Science and Programming, se trata de una materia que no requiere conocimientos previos de programación, el objetivo de la misma es brindar las herramientas necesarias para permitir que el estudiante pueda resolver problemas empleando la programación.

La información aquí expuesta se obtuvo analizando el sitio web de la materia [22] que corresponde a la cátedra de John Guttag y Eric Grimson, y del libro propuesto como material de lectura “How to Think Like a Computer Scientist” de Nantais y Downey.

No se utilizan algoritmos, el curso comienza mostrando a los estudiantes el entorno de programación, presentando algunas sentencias y haciendo que desde el inicio se realicen pequeños scripts en Python.

Por tratarse de Python el lenguaje de programación elegido es posible comenzar a utilizarlo de manera imperativa para pasar poco a poco al paradigma de objetos.

Existen dos clases semanales de una hora cada una, adicionalmente los alumnos cuentan con una clase de una hora pensada para resolver dudas, allí tienen la oportunidad de hacer preguntas sobre el material de lectura o sobre el problema planteado para la semana. La materia cuenta con una guía de ejercicios que apunta a que cada alumno la resuelva de manera individual, muchos de estos ejercicios cuentan con “archivos de soporte” en ellos se encuentra el esqueleto del problema, en muchos casos con una exhaustiva explicación asociada a cada función, de manera que puede considerarse como una actividad de práctica semicontrolada ya que el alumno debe ceñirse al esqueleto del programa que recibe.

La materia dispone de un aula virtual de acceso público la cual cuenta con las grabaciones de todas las clases teóricas, el material utilizado en las mismas, guías de ejercicios y literatura recomendada.

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en que: no se explica el uso de algoritmos, no se aborda el tema de manejo de archivos, no se utiliza el concepto de listas enlazadas y no se practican técnicas de refactorización de código.

*6) Portland Community College*

Ubicada en Portland, Oregon, Estados Unidos es la institución más grande de educación superior en el estado, fue fundada en el marco de un programa de educación para adultos en el año 1959 y rebautizada como Portland Community College en el año 1961.

Java Programming I, en esta materia se enseñan conceptos de diseño, implementación y pruebas de software utilizando Java.

La información aquí expuesta se obtuvo analizando los documentos: Joint Task Force on Computing Curricula [1] como así también el sitio web de la materia [23] que corresponde a la cátedra de Cara Tang.

No se utilizan algoritmos, el concepto y propiedades de los algoritmos se abordan en un curso anterior, el curso utiliza un entorno de desarrollo Java diseñado específicamente para la enseñanza a un nivel introductorio, denominado BlueJ, este permite comenzar a interactuar con el concepto de objetos de manera gráfica. BlueJ ha sido diseñado e implementado por el equipo BlueJ en la Deakin University, Melbourne, Australia, y la University of Kent, en Canterbury, Reino Unido.

El paradigma empleado es de programación orientada a objetos y el lenguaje utilizado es Java. El enfoque del curso es el de "objetos primero" con el fin de concentrarse más en los conceptos y menos en los detalles sintácticos y prácticos necesarios para conseguir un programa en ejecución.

El curso está formado por cuarenta horas de clase teórica-prácticas y veinte horas de laboratorio opcionales. Todas las aulas están equipadas con una computadora en cada escritorio y el tiempo de clase se compone de dos conferencias y luego actividades en equipos. Los estudiantes suelen trabajar en los ejercicios y hacer preguntas relacionada con estos. A lo largo del curso se trabajan siete proyectos de programación, en seis de ellos los estudiantes añaden funcionalidad al código existente, que van desde la adición de un único método de una sola línea hasta añadir funcionalidad significativa y sólo en uno de los proyectos los estudiantes escriben todo el código desde cero.

El aula virtual de la materia, la cual no es de acceso público, cuenta con materiales propios de la cátedra que complementan el libro de texto y videos pregrabados que guían a los estudiantes en la resolución de ejercicios.

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en que: no se explica el concepto de recursividad, no se explican listas enlazadas, no se aborda el tema de búsquedas ni de ordenamiento y no se hace énfasis en la práctica de refactorización de código.

*7) Stanford University*

Es una universidad privada ubicada en Palo Alto, California, Estados Unidos en el corazón geográfico de Silicon Valley. La universidad abrió sus puertas oficialmente el 1 de octubre de 1890.

Programming Methodology, es un curso que fue diseñado para estudiantes de múltiples especialidades y no requiere conocimientos previos en programación.

La información aquí expuesta se obtuvo analizando el sitio web de la materia [31] que corresponde a la cátedra de Mehran Sahami.

El curso comienza utilizando Java de la manera más simple con Karel "el Robot", es una herramienta de aprendizaje que presenta los conceptos de una forma visual además de tener un nivel de abstracción bajo. Karel "el Robot" puede manipular sólo seis comandos básicos, mediante estas instrucciones le permiten al simulador moverse por su entorno e interactuar.

Una vez incorporados los conceptos básicos de programación con Karel, el lenguaje utilizado es Java y en él se trabajan los aspectos formales de programación empleando el paradigma de orientación a objetos.

El curso se dicta tres veces por semana durante una hora y el formato es netamente teórico, no cuenta con prácticas de laboratorio. Los alumnos pueden solicitar entrevistas con docentes de la clase que brindan apoyo a fin de poder evacuar dudas sobre los aspectos prácticos de la materia. A lo largo de la cursada existen siete trabajos prácticos los cuales son evaluados y sobre ellos es posible obtener una explicación detallada de los errores cometidos.

La materia dispone de un aula virtual de acceso público el cual cuenta con las grabaciones de todas las clases teóricas, el material utilizado en las mismas, guías de ejercicios y literatura recomendada.

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en que: No se explica el concepto de listas enlazadas ni de recursividad.

*8) Worcester Polytechnic Institute*

Es una universidad privada de investigación en Worcester, Massachusetts, Estados Unidos se centra en la instrucción e investigación de las artes técnicas y ciencias aplicadas. Fundada en 1865, Worcester Polytechnic Institute fue una de las primeras universidades de ingeniería y tecnología de los Estados Unidos.

Introduction to Program Design, es un curso pensado para estudiantes sin experiencia previa en programación, el curso está pensado para aprender a diseñar programas que resuelvan problemas.

La información aquí expuesta se obtuvo analizando el documento Joint Task Force on Computing Curricula [1]como así también el sitio web de la materia [41] que corresponde a la cátedra de Kathi Fisler y Glynis Hamel.

Prácticamente es nula la utilización de algoritmos, el curso introduce a la disciplina de la informática centrándose en la resolución de problemas desde la perspectiva de la programación funcional. El lenguaje utilizado en el curso es Racket, un lenguaje de programación de amplio espectro de la familia de Lisp y Scheme.

El curso cuenta con cuatro horas semanales de clases teórico prácticas y una clase de una hora pensada para resolver un ejercicio en el laboratorio. Se enfatiza en la metodología de desarrollo conocida como data-driven y test-driven en ella se le otorga al alumno un problema de programación y se pide completar en orden los siguientes pasos siguientes: (1) definir los tipos de datos, (2) escribir ejemplos de datos en cada tipo, (3) escribir el esqueleto de una función capaz de procesar los tipos de datos, (4) escribir el prototipo de la función indicando que recibe y que retorna, (5) escribir casos de prueba para esa función, (6) rellenar el esqueleto para el principal tipo de datos de entrada y (7) ejecutar los casos de prueba. Los estudiantes trabajan en una tarea de programación corta durante la clase de laboratorio cuya duración es de una hora y les es asignada una guía de ejercicios relacionada con los temas de programación abordados en las clases teóricas.

Existe un aula virtual destinada a la materia, en ella se puede encontrar tanto material (modo texto) referido a la teoría de la materia como así también información pertinente a las fechas de la cursada.

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en que: no se explica el uso de algoritmos, no se aborda el tema de operaciones de entrada/salida ni el de manejo de archivos y no se abordan temas referidos a la refactorización de código. Por otra parte queda un conjunto de temas a ser abordados en una materia denominada "CS2/Object-Oriented Design course" como ser: cadenas y procesamiento de las mismas, tipos abstractos de datos, manejo dinámico de memoria, listas enlazadas y búsquedas ordenamiento.

### *B. Instituciones de la Argentina*

La educación superior en la Argentina comienza luego que el estudiante ha completado su educación secundaria, es el segmento del sistema educativo que completa los trece años de educación obligatoria fijados por la Ley de Educación Nacional N° 26.206 o califiquen según el artículo 7º de la Ley de Educación Superior Nº 24.521, allí se establece que los mayores de 25 años que no reúnan la condición de completitud del ciclo secundario podrán ingresar a la educación superior siempre que demuestren, a través de las evaluaciones, que tienen preparación y/o experiencia laboral acorde con los estudios que se proponen iniciar, así como aptitudes y conocimientos suficientes para cursarlos satisfactoriamente.

A diferencia de lo que ocurre en Estados Unidos, en la Argentina existe un organismo encargado de evaluar y acreditar las carreras universitarias, La Comisión Nacional de Evaluación y Acreditación Universitaria (CONEAU) en un organismo público descentralizado que funciona en jurisdicción del Ministerio de Educación de la Nación. Fue creado con la finalidad de contribuir al mejoramiento de la educación universitaria. Su misión institucional es asegurar y mejorar la calidad de las carreras e instituciones universitarias que operan en el sistema universitario argentino por medio de actividades de evaluación y acreditación de la calidad de la educación superior.

Las casas de estudio analizadas en la Argentina son todas universidades de gestión pública, éstas son instituciones de educación superior las cuales pueden estar o no formadas por diversas facultades. Las facultades son las divisiones que tienen las universidades, según la rama del saber (Facultad de Economía, Facultad de Filosofía, Facultad de Ingeniería, etc.) o en el caso de la Universidad Tecnológica Nacional según regiones geográficas (Facultad Regional Buenos Aires, Facultad Regional Avellaneda, etc.). Como alternativas al modelo de organización por facultades, existen universidades que tienen una organización por departamento, por lo general las unidades académicas o departamentos (algunas universidades adoptan el nombre de "instituto") son responsables de coordinar una o más carreras y planes de estudio, asumiendo la gestión de las misiones propiamente universitarias: docencia, investigación, desarrollo, transferencia y extensión.

*1) Universidad de Buenos Aires*

Es una universidad nacional y pública de la República Argentina con sede en la Ciudad Autónoma de Buenos Aires. Fue fundada el 12 de agosto de 1821 y oficialmente inaugurada el 26 de agosto de 1821.

Algoritmos y Programación I. Es el primer curso de programación de la facultad de Ingeniería, en él se plantean los siguientes desafíos: enseñar una metodología para la resolución de problemas y poder implementarlas en un lenguaje formal de programación.

La información aquí expuesta se obtuvo analizando el documento "Algoritmos y Programación I - Aprendiendo a programar usando Python como herramienta" [40] como así también el sitio web de la materia [33] que corresponde a la cátedra de Rosita Wachenchauzer.

No se utilizan algoritmos, el curso comienza mostrando a los estudiantes el entorno de programación, presentando algunas sentencias y haciendo que desde el inicio se utilice Python desde la terminal.

El lenguaje utilizado para programar a lo largo del curso es Python y el paradigma es el imperativo, la cátedra considera que la elección del lenguaje de programación no es un tema menor, buscó un lenguaje que al mismo tiempo sea suficientemente expresivo, tenga una semántica clara y cuyas complicaciones sintácticas sean mínimas.

La materia cuenta con cuatro horas semanales de teoría y cuatro horas semanales de laboratorio. Los estudiantes completan cuatro trabajos prácticos, dos individuales y dos en grupos de dos alumnos, que implican el diseño e implementación de programas en Python, los cuales incluyen en algunos casos un informe escrito y una clara documentación del código.

La materia cuenta con un aula virtual la cual se limita a compartir material con los alumnos. Se recomienda como material de consulta un documento [40] elaborado por la cátedra.

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en la falta de ejercitación referida a la refactorización de código y por lo tanto en la comprensión de los mismos.

*2) Universidad Nacional de La Plata*

Es una universidad pública de la República Argentina, fundada en 1905 con sede en la ciudad de La Plata, Buenos Aires.

Algoritmos, Datos y Programas, es un curso que responde a un modelo clásico centrado en el aprendizaje de la expresión de algoritmos, la introducción a las estructuras de datos lineales y no lineales y a la incorporación de conceptos relacionados con modularización, análisis de eficiencia, abstracción y reutilización.

La información aquí expuesta se obtuvo analizando el documento "Aprenda Lenguaje ANSI C como si estuviera en Primero" de la Universidad de Navarra como así también el sitio web de la materia [36] que corresponde a la cátedra de Graciela Toccaceli.

Se comienza implementando algoritmos desde una notación gráfica, para ello se utilizan diagramas de flujo, los cuales son realizados en papel para luego introducir el concepto de pseudocódigo.

Una vez abordados los conceptos básicos empleando diagramas y pseudocódigo se explica cómo convertir estos últimos en lenguaje C ya que la enseñanza se encuadra dentro del paradigma imperativo mediante el uso de dicho lenguaje.

Las clases teóricas son semanales y de una hora y media de duración, en éstas se explican y/o presentan los temas que comprenden la materia. Las clases prácticas tienen la misma duración que las de teoría pero se realizan dos encuentros por semana en el laboratorio. Existen seis guías de ejercicios agrupados por temas y adicionalmente cuentan con guías de ejercicios extras las cuales no son de realización obligatoria. Al finalizar la cursada los alumnos podrán optar por la realización de un problema especial con el objeto de incrementar la nota, el mismo deberá ser realizado de manera individual, presentado y defendido.

Existe un sitio web que cumple la función de aula virtual, el mismo brinda información, contiene materiales teóricos, enlaces de interés y noticias.

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1]en que: no se abordan prácticas de trabajo colaborativo, ni reglas de documentación y estilo.

*3) Universidad Nacional de Lanús*

Es una universidad pública de la República Argentina fundada el 7 de junio de 1995 con sede central en la localidad bonaerense de Remedios de Escalada, en el partido de Lanús, Provincia de Buenos Aires.

Programación de computadoras, busca introducir al alumno en los conceptos básicos y fundamentales de los algoritmos y las estructuras de datos, brindándole las herramientas necesarias para el aprendizaje de cualquier lenguaje de programación.

La información aquí expuesta se obtuvo analizando el documento "Programa de la Materia Programación de computadoras" [35] que corresponde a la cátedra de Héctor A. Carballo.

La metodología empleada plantea la elaboración de un algoritmo que resuelve el problema planteado, utilizando para tal fin pseudocódigo en base a la nomenclatura UPSAM 2.0.

La enseñanza se encuadra dentro del paradigma imperativo mediante el uso del lenguaje C, como paso previo a la codificación en C se resuelve el problema planteado mediante el uso de pseudocódigo.

Las clases teóricas son semanales y de una hora y media de duración, en estas se explicarán y/o presentan los temas que comprenden la materia. Las clases prácticas tienen la misma duración que las de teoría pero se realizan dos encuentros por semana en el laboratorio. A lo largo de la cursada se plantean ejercicios de resolución individual y en casos menores se promueve el trabajo en equipo formando grupos de trabajo los cuales resuelven casos prácticos o monografías sobre alguno de los contenidos de la materia.

Existe un sitio web que brindar información y materiales a los alumnos, el mismo no es de acceso público.

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en que: no se aborda el tema corrección de programas y tampoco reglas de documentación y estilo.

*4) Universidad Nacional de Quilmes*

Es una universidad pública de la República Argentina con sede en la localidad de Bernal, en el partido bonaerense de Quilmes, Provincia de Buenos Aires, fue fundada en el año 1989.

Introducción a la Programación / Algoritmos y Programación, a diferencia de otros cursos iniciales que se enfocan demasiado en los elementos específicos de un lenguaje particular y no priorizan el entendimiento de las ideas subyacentes en las herramientas de programación que el lenguaje brinda, este curso presenta un enfoque guiado por la necesidad de focalizar el aprendizaje en el proceso de abstracción y en los conceptos fundamentales, transversales a todos los paradigmas y lenguajes.

La información aquí expuesta se obtuvo analizando los documentos: "El nombre verdadero de la programación" [19] y "Las bases conceptuales de la Programación: Una nueva forma de aprender a programar" [20] como así también el sitio web de la materia que corresponde a la cátedra de Eduardo Bonelli y Francisco Soulignac.

La cátedra creó un lenguaje denominado Gobstones orientado a personas que no tienen conocimientos previos en programación y desarrolló un entorno que permite ejecutarlo denominado PyGobstone.

La transición desde Gobstones a lenguajes usados en el mundo real es bastante simple, ya que su sintaxis fue pensada para ser similar a otros lenguajes imperativos convencionales (en particular C y Java).

La materia cuenta con tres horas semanales de teoría y seis horas semanales de laboratorio dividido en dos clases. Se trabaja sobre ejercicios guiados que se centran en evaluar la habilidad del estudiante para expresar soluciones de forma clara y su capacidad de distinguir en el proceso de programación cada concepto que le es impartido.

La materia dispone de un aula virtual de acceso público en la cual se encuentra: el material utilizado en las clases teóricas, las guías de ejercicios y la literatura recomendada.

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en que no se aborda: Operaciones de E/S, manejo de archivos, recursividad, manejo de cadenas de caracteres, utilización tipos abstractos de datos, manejo dinámico de memoria, aliasing, manejo de listas enlazadas, estrategias para la elección de la estructura de datos apropiada, comprensión programas, corrección del programas, refactorización simple y entornos de programación modernos.

*5) Universidad Nacional de Río Cuarto*

Es una universidad pública nacional de la República Argentina, con sede en la ciudad cordobesa de Río Cuarto, fundada el 1 de mayo de 1971.

Introducción a la Algorítmica y Programación, en esta materia se plantea la necesidad de independizarse del lenguaje de programación, definiendo un estándar llamado "pseudo-lenguaje", con un grado de generalidad que permita en un segundo paso la traducción de la solución algorítmica desarrollada a diferentes lenguajes de programación.

La información aquí expuesta se obtuvo analizando los documentos: "Enseñanza de la programación" [12] como así también el programa de la materia "Introducción a la Algorítmica y Programación" [37] que corresponde a la cátedra de Ariel Ferreira Szpiniak.

La metodología empleada se fundamenta en la elaboración de un algoritmo que resuelve el problema planteado, este punto es el paso previo al desarrollo del programa, se resuelve el problema de manera general, usando una notación algorítmica de alto nivel y sin necesidad de preocuparse por los detalles de programación propios de cada lenguaje, una vez obtenido el algoritmo se traduce a un lenguaje de alto nivel.

La enseñanza se encuadra dentro del paradigma imperativo mediante el uso de un lenguaje estructurado de carácter general, Pascal.

Las clases teóricas son semanales y de tres horas de duración. Las clases prácticas también son semanales y de cuatro horas de duración. También se cuenta con clases de consulta de dos horas de duración por semana y clases de implementación/repaso de dos horas de duración por semana. Para la actividad práctica se dispone de cuatro horas por semana reservadas exclusivamente para el acceso al laboratorio de informática fuera de los horarios de clase. Los alumnos desarrollan en grupos, de no más de tres integrantes, un proyecto integrador de los temas trabajados, de mediana complejidad. Dicho proyecto será evaluado al cierre cada cuatrimestre.

Existe un aula virtual disponible en la web para brindar información y materiales a los alumnos, contiene foros, materiales teóricos y prácticos, software, enlaces de interés, noticias, pizarrón, grupos, correo, y actividades (publicación, recepción de resoluciones, calificaciones y devoluciones).

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en que: no se abordan los siguientes temas: comprensión programas, corrección de programas, refactorización simple, y no se utiliza un entorno de programación moderno.

*6) Universidad Nacional de San Luis*

Es una universidad pública argentina; fundada el 10 de mayo de 1973, se organizó a partir de la división de la Universidad Nacional de Cuyo, sobre las dependencias que esta última tenía en la provincia de San Luis.

Programación I, introduce al alumno en los conceptos fundamentales de la programación imperativa, acompañando la enseñanza de los conceptos teóricos con prácticas realizadas en un lenguaje de programación.

La información aquí expuesta se obtuvo analizando el programa analítico, el material de estudio y el sitio web de la materia [39] que corresponde a la cátedra de Lorena Baigorria.

Los temas referidos a algoritmia son abordados por una materia predecesora "Resolución de problemas y algoritmos",

en la cual se comienza el proceso de enseñanza utilizando el Lenguaje TIMBA (Terribly Imbecile Machine for Boring Algorithms) que fuera ideado con fines educativos por un grupo de docentes de la UNSL, dirigido por el Ing. Hugo Ryckeboer. TIMBA es un pseudo-lenguaje simple que permite la definición de algoritmos basado en el paradigma de la programación estructurada. Otra herramienta utilizada en el curso es Dia Diagram Editor, es empleada como complemento y facilitador del proceso de diseño y definición del algoritmo final a través de la realización de diagramas de flujo simples.

La enseñanza se encuadra dentro del paradigma imperativo mediante el uso de un lenguaje algorítmico estructurado de carácter general, C.

Las clases teóricas son semanales y de tres horas de duración. Las clases prácticas son de tres horas de duración dos veces por semana. Los alumnos desarrollan de manera individual seis proyectos a lo largo de la cursada y un proyecto integrador de los temas trabajados, de mediana complejidad. Dicho proyecto será evaluado al cierre de la cursada.

Existe un aula virtual disponible en la web para brindar información y materiales a los alumnos. El aula contiene materiales teóricos y prácticos, software, enlaces de interés y noticias.

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en los siguientes temas: comprensión programas, corrección de programas y refactorización simple.

*7) Universidad Nacional del Sur*

Es una universidad pública de la República Argentina con sede central en la Ciudad de Bahía Blanca, al sur de la Provincia de Buenos Aires. Fue fundada el 5 de enero de 1956.

Resolución de problemas y algoritmos, tiene por objetivo principal que los alumnos adquieran la capacidad de desarrollar programas para resolver problemas de pequeña escala.

La información aquí expuesta se obtuvo analizando el programa de la materia como así también el sitio web de la misma [38] que corresponde a la cátedra de Alejandro García y Sergio Gómez.

Se plantea como objetivo fundamental la resolución de problemas de simple complejidad, para ello se plantea una metodología en tres etapas:

1. Adquirir habilidad en la detección de una situación de problema y en el planteo de los posibles caminos de solución mediante las técnicas generales de resolución de problemas.
2. Resolver los problemas dados en un lenguaje de diseño de algoritmos.
3. Transformar el algoritmo de programación escrito en un lenguaje de diseño a un programa escrito en el lenguaje elegido.

La enseñanza se encuadra dentro del paradigma imperativo mediante el uso de un lenguaje algorítmico estructurado de carácter general, Pascal.

La materia cuenta con cuatro horas semanales de teoría y cuatro horas semanales de laboratorio. El enfoque de la materia es esencialmente práctico, orientado a la resolución de problemas simples. Las clases teóricas son dictadas por el profesor; aunque se promueve la participación de los alumnos, la metodología de enseñanza es fundamentalmente expositiva. Los alumnos desarrollan de manera individual siete proyectos a lo largo de la cursada.

Existe un aula virtual disponible en la web para brindar información, el programa de la materia, los trabajos prácticos y cronograma de evaluación.

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en los siguientes temas: tipos abstractos de datos y manejo dinámico de memoria, referencias y aliasing (diferentes nombres simbólicos en el programa), listas enlazadas, estrategias para la elección de la estructura de datos apropiada, búsquedas, ordenamiento, comprensión programas, corrección de programas, refactorización simple y entornos de programación modernos.

*8) Universidad Nacional del Litoral*

Es una universidad pública nacional de la República Argentina, está emplazada en la provincia de Santa Fe, en la región Centro-Litoral del país, fue fundada el 17 de octubre de 1919.

Fundamentos de Programación, es una asignatura que corresponde al primer cuatrimestre del primer año de la carrera, tiene por propósito introducir a los estudiantes a los conceptos de resolución de problemas mediante algoritmos computacionales e implementación de programas mediante el uso de un lenguaje de alto nivel estándar.

La información aquí expuesta se obtuvo analizando el artículo "Intérprete para probar un programa escrito en pseudocódigo" [11] como así también el sitio web de la materia [34] que corresponde a la cátedra de Horacio Cesar Loyarte.

Se destaca el uso de un pseudocódigo en español en las 4 primeras unidades de la asignatura que apuntan a desarrollar en los estudiantes los conceptos de lógica de programación y el uso de ciertas estructuras básicas de control y de datos en forma independiente de un lenguaje de alto nivel. Esta tarea se lleva adelante empleando la herramienta PSeInt la cual proporciona un editor de pseudocódigo, un diagrama de flujo muy simple y un traductor con características diseñadas específicamente para el uso en el aula; brinda herramientas que ayudan a encontrar errores y comprender la lógica de los algoritmos.

La enseñanza se encuadra dentro del paradigma imperativo, se trabaja en un IDE denominado Zinajl, dirigido a las necesidades de aprendizaje y enseñanza, con características para facilitar la edición, depuración y prueba de programas en C ++.

Las clases teóricas se desarrollan semanalmente y tienen una duración de dos horas. Las clases prácticas, al igual que las teóricas, son semanales y tienen una duración de tres horas y media. Los alumnos desarrollan de manera individual diez guías de ejercicios a lo largo de la cursada.

Los alumnos disponen de un aula virtual donde podrán acceder al material de la cátedra, noticias, foros, sitios de interés, material adicional, cartelera de avisos, etc.

Los contenidos de la materia difieren con respecto a los enunciados en el documento Joint Task Force on Computing Curricula [1] en los siguientes temas: Archivos, recursividad, tipos abstractos de datos y manejo dinámico de memoria, listas enlazadas, búsquedas, ordenamiento, comprensión de programas, corrección de programas, refactorización simple, documentación y reglas de estilo.

## III. Comparación de Abordajes

La utilización del método comparativo al igual que cualquier otro método de análisis empírico requiere una serie de decisiones previas referidas al diseño de investigación. Se considera que la comparación es el instrumento apropiado en situaciones en las que el número de casos bajo estudio es

demasiado pequeño para permitir la utilización del análisis estadístico.

En la sección anterior, han sido definidos los criterios con los cuales son impartidos los cursos iniciales de programación en diversas casas de estudio, en esta se presenta el instrumento desarrollado para llevar adelante la comparación (A) y una descripción comparativa de los casos analizados (B).

### *A. Instrumento utilizado*

El dispositivo usado para recopilar los datos fue diseñado ad hoc para esta experiencia, tomando en consideración factores tales como las competencias que debe alcanzar un alumno en un curso inicial de programación, las metodologías de enseñanza, la utilización de aulas virtuales, el tipo de ejercitación, la experiencia docente de colegas y la propia, entre otras cuestiones. Para abordar el problema de análisis del dispositivo de enseñanza de Programación en un estadio inicial hemos agrupado los aspectos en tres grupos: contenidos (sección 1), didáctica, contenidos (sección 2), y herramientas (sección 3). En la tabla 1 se sintetiza el protocolo propuesto.

#### *1) Contenidos*

Los aspectos referidos a contenidos los hemos tomado del trabajo propuesto por la "Association for Computing Machinery" (ACM) en conjunto con el "Institute of Electrical and Electronics Engineers" (IEEE) [1] por considerarlo un referente internacional para los currículos de computación. En él se sugieren los contenidos y la distribución de los mismos en cursos, estructurados por área de conocimiento y nivel de enseñanza. En el apartado Software Development Fundamentals se centra en el proceso de desarrollo de software, la identificación de los conceptos y las habilidades necesarias que debe abarcar en un curso de programación inicial, los conceptos se subdividen en: algoritmos y diseño, conceptos fundamentales de programación, estructuras de datos fundamentales y métodos de desarrollo.

#### *2) Didáctica*

Los aspectos referidos a didáctica se elaboraron partiendo de la premisa que para poder utilizar las computadoras en la resolución de problemas de manera efectiva, los estudiantes no solo deben alcanzar las competencias en la lectura y escritura de programas sino, también en el análisis y diseño de soluciones. Nos pareció pertinente determinar el formato de las clases, el modo de ejercitación y si la estrategia de enseñanza responde o no a una metodología tradicional (un modelo por imitación, en donde el instructor pro-pone un problema y desarrolla e implementa su solución, esperando que el estudiante lleve este desarrollo a su propio contexto).

#### *3) Herramientas*

Los aspectos referidos a herramientas se elaboraron teniendo en cuenta lo expresado por [12] en relación a que no habría un consenso para enseñar programación, hay métodos de enseñanza que se fundamentan a partir de un paradigma de programación en particular (funcional, imperativo, imperativo con el aporte de la teoría de objetos) y dentro de ese paradigma se utilizan varios enfoques: enseñar a programar sobre un lenguaje de programación en particular o emplear un lenguaje algorítmico general. En lo referido al lenguaje no solo nos pareció relevante determinar cuál es el utilizado y dentro de que paradigma sino también sobre que herramienta se trabaja. Con respecto al uso del campus virtual se pretende determinar, por un lado su utilización y, por otro su alcance, esto es, si se utiliza como soporte de un curso a distancia o como apoyo a la presencialidad.

### *B. Comparativa de los Casos Relevados*

En la actualidad, coexisten muchas tendencias en lo que respecta al enfoque didáctico para las materias introductorias de programación, pudiéndose verificar que no hay un consenso en los métodos a utilizar. En efecto, hemos detectado a partir de los casos enunciados anteriormente que los distintos enfoques pueden variar en los siguientes aspectos:

- Utilización de algoritmos
- Paradigma de programación empleado
- Metodología de ejercitación
- Formato de las clases
- Utilización del aula virtual
- Contenidos de las materias

TABLA I. PROTOCOLO DE ANÁLISIS PARA CURSOS INICIALES DE PROGRAMACIÓN

| | | |
|---|---|---|
| **Contenido** | **Algoritmos y Diseño** | El concepto y propiedades de los algoritmos |
| | | El papel de los algoritmos en el proceso de resolución de problemas |
| | | Las estrategias de resolución de problemas |
| | | Conceptos de diseño fundamentales |
| | **Conceptos Fundamentales de Programación** | Sintaxis básica y la semántica de un lenguaje de alto nivel |
| | | Variables y tipos de datos primitivos |
| | | Expresiones y asignaciones |
| | | Operaciones de E / S |
| | | Archivos de E / S |
| | | Estructuras de control condicionales e iterativas |
| | | Funciones y pasaje de parámetros |
| | | El concepto de recursividad |
| | **Estructuras de datos fundamentales** | Listas |
| | | Cadenas y procesamiento de cadenas |
| | | Tipos abstractos de datos y manejo dinámico de memoria |
| | | Referencias y aliasing |
| | | Las listas enlazadas |
| | | Estrategias para la elección de la estructura de datos apropiada |
| | | Búsquedas |
| | | Ordenamiento |
| | **Métodos de Desarrollo** | Comprensión Programas |
| | | Corrección del Programas |
| | | Refactorización simple |
| | | Entornos de programación modernos |
| | | Estrategias de depuración |
| | | Documentación y reglas de estilo |
| **Didáctica** | **Formato clases** | ¿Se dictan clases netamente teóricas? |
| | | ¿Se dictan clases prácticas? |
| | **Ejercitación** | ¿Se realizan ejercicios de manera colaborativa? |
| | | ¿Se realizan ejercicios desde cero? |
| | | ¿Se realizan ejercicios a completar? |
| | **Modo Enseñanza** | ¿Se enseña desde el ejemplo? |
| | | ¿Se enseña guiado por el problema? |
| **Herramientas** | **Lenguaje** | ¿En qué paradigma se programa? |
| | | ¿En qué lenguaje se programa? |
| | | ¿En qué entorno se programa? |
| | **Algoritmia** | ¿Cuál es el entorno? |
| | | ¿Qué leguaje utiliza? |
| | **Campus virtual** | ¿Cuenta con campus la materia? |
| | | ¿El campus se limita a compartir material? |
| | | ¿Brinda el campus herramientas de autoevaluación? |

*1) Utilización de algoritmos*

Según Moroni y Señas [46] la complejidad de los programas que se desarrollan en la actualidad produce la necesidad de iniciar a los alumnos en un camino que los conduzca a utilizar técnicas de programación efectivas, considerando importante para ello poner énfasis en el diseño previo y en la utilización de los algoritmos como recursos esquemáticos para plasmar el modelo de resolución de un problema.

En los Estados Unidos se detectó que en la mitad de los casos relevados se emplea algoritmia y en todos los casos en los que se opta por emplearla se utiliza un lenguaje específico del dominio (LED).

En la Argentina esta técnica con sus distintas variantes (pseudocódigo, diagramas y lenguajes específicos del dominio) se emplea en casi la totalidad de los casos relevados (siete de los ocho relevados), existiendo una marcada preponderancia de la utilización de pseudocódigo ya sea como única herramienta o combinado con diagramas de flujo (Figura 1).

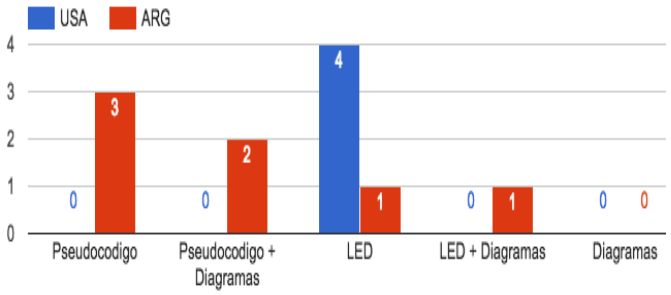



Fig. 1. Modelo elegido para describir algoritmos.

*a) Utilización de Pseudocódigo*

El pseudocódigo es una descripción de alto nivel de un algoritmo que emplea una mezcla de lenguaje natural con algunas convenciones sintácticas propias de lenguajes de programación.

En ninguno de los casos relevados en Estados Unidos se utilizan pseudocódigos a diferencia de lo que ocurre en la Argentina donde la utilización de pseudocódigo se encuentra muy extendida (cinco de los ocho casos analizados), en más de la mitad de los casos dicha técnica se utiliza a lo largo de toda la cursada, esto implica realizar el pseudocódigo del ejercicio antes de pasar a la etapa de codificación en el lenguaje formal empleado. En dos universidades en las que se emplea pseudocódigo se lo utiliza en combinación con diagramas de flujo.

*b) Utilización de Diagramas*

Los diagramas son descripciones gráficas de algoritmos que usan símbolos conectados con flechas para indicar la secuencia de instrucciones.

En ninguno de los casos relevados en Estados Unidos se utilizan diagramas y en la Argentina donde si se usa este método (tres de los ocho casos relevados), hemos detectado que no son utilizados como único elemento que permita describir un algoritmo, pues en todos los casos se utiliza una combinación de diagramas con pseudocódigo o con un lenguaje específico del dominio.

*c) Utilización de lenguajes específicos del dominio*

Los lenguajes específicos del dominio son lenguajes de programación dedicados a resolver un problema en particular, representar un problema específico y proveer una técnica para solucionar una situación particular. Estos lenguajes son normalmente utilizados en entornos de aprendizaje basados en el contexto [5][26][32], éstos se tratan de herramientas que le permiten al estudiante escribir código y observar de inmediato la ejecución.

En los casos relevados de las universidades de los Estados Unidos los lenguajes específicos del dominio son la única técnica empleada para la enseñanza de algoritmia y se utiliza en cuatro de las ocho universidades relevadas. En el relevamiento se detectó que en todos los casos se cuentan con herramientas informáticas para poder trabajar con dichos lenguajes.

En la Argentina solo es utilizado en dos casos, en uno de ellos se emplea un lenguaje denominado Gobstones el cual cuenta con un entorno de trabajo y en el otro se emplea un lenguaje que se denomina TIMBA, el cual es un lenguaje que a través de un personaje que sólo comprende una cantidad pequeña de órdenes, manipula pilas de cartas permitiendo la construcción de algoritmos utilizando las tres estructuras básicas (secuencial, condicional e iterativa) lo interesante de este caso es que solo se emplea una baraja de naipes, papel y lápiz (Figura 2).

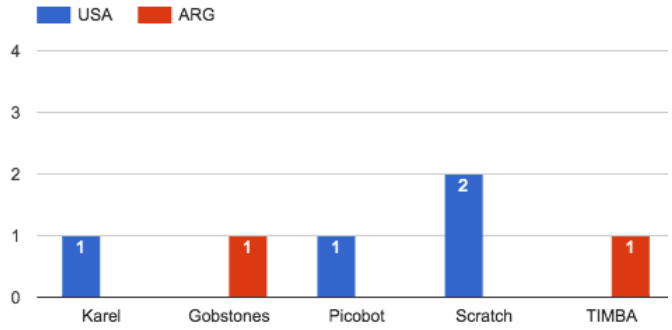



Fig. 2. Lenguajes específicos del dominio.

A pesar de no existir unicidad de criterio sobre qué lenguaje específico del dominio aplicar, como puede observarse en la figura II, en los casos de Karel, Picobot, Scratch y Gobstones, se trabaja en un entorno gráfico simple, apuntado a la comprensión de los conceptos básicos de la algoritmia.

*2) Paradigma de programación empleado*

Ferreira Szpiniak y Rojo [12] expresan que no hay un consenso para enseñar programación, hay métodos de enseñanza que se fundamentan a partir de un paradigma de programación en particular (funcional, imperativo, imperativo con el aporte de la teoría de objetos) y dentro de ese paradigma se utilizan varios enfoques: enseñar a programar sobre un lenguaje de programación en particular o emplear un lenguaje algorítmico general.

En los casos relevados de las universidades de los Estados Unidos se detectó que existe una marcada tendencia en la utilización del paradigma orientado a objetos, ya sea en los dos casos donde se utiliza a éste como único paradigma de la cursada como en aquellos casos donde se trabaja con múltiples paradigmas y en todos los casos el orientado a objetos forma parte. Es importante destacar que en todos los casos donde se opta por trabajar con un enfoque multi paradigma el lenguaje empleado es Python (Figura 3).

En la Argentina es unánime la utilización del paradigma imperativo, a pesar de ser abordado con diversos lenguajes de programación como se aprecia en la (figura 4) vale la pena

destacar que emerge como el más popular el lenguaje C (utilizado en tres de los ocho casos).

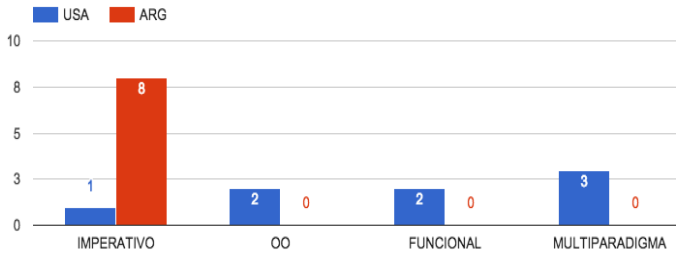



Fig. 3. Paradigma de programación empleado.

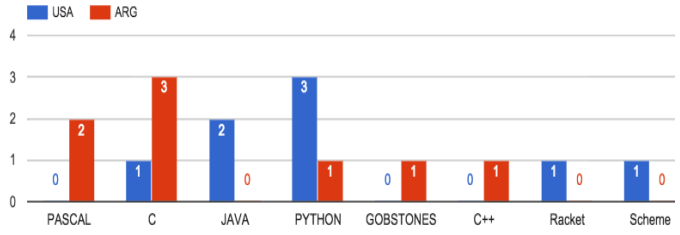



Fig. 4. Lenguajes Utilizados.

Hemos detectado una marcada tendencia tanto en las universidades de los Estados Unidos como en las de Argentina, a enseñar los conocimientos en entornos de trabajo modernos que en su mayoría permiten ser utilizados en múltiples sistemas operativos (Figura 5).

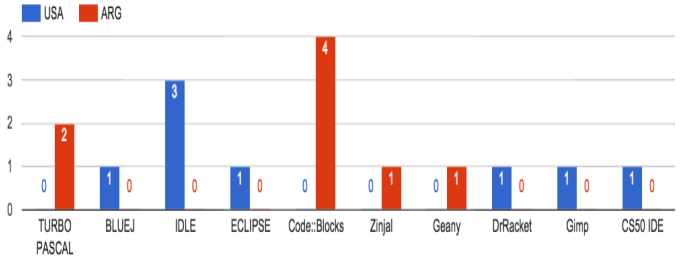



Fig. 5. Entornos de Desarrollo.

El CS50 IDE, utilizado tanto por Harvard como por Yale, es un entorno de desarrollo integrado basado en la nube que cuenta con todas las características de un IDE convencional pero a diferencia de estos se utilizada desde un navegador web, permitiendo a los alumnos poder trabajar desde cualquier dispositivo.

*3) Metodología de Ejercitación.*

(El común denominador sobre los casos relevados en la Argentina es la metodología de ejercitación desde cero, esto implica tomar como punto de partida de cada una de las prácticas un nuevo programa.

En los Estados Unidos a pesar de ser la tendencia preponderante las prácticas desde cero se detectaron tres casos en los cuales no se utiliza como la única metodología de ejercitación, en estos casos se alterna con ejercicios a completar por parte de los alumnos.

Se destaca entre los que optaron por incorporar otra metodología de ejercitación, el Portland Community College de los Estados Unidos, allí a lo largo de la cursada se desarrollan siete proyectos de programación, en seis de los cuales los estudiantes añaden funcionalidad al código existente y tan solo en el último proyecto los estudiantes escriben todo el código desde cero. Otro de los casos en los que se promueven las prácticas semi-libres es en el MIT allí los alumnos cuentan con una guía de ejercicios con "archivos de soporte" y en ellos se encuentra el esqueleto del problema, en muchos casos con una exhaustiva explicación asociada a cada función (Figura 6). Otro aspecto a destacar es que una vez adquiridos los conocimientos mínimos de programación son muy pocos los casos en los que se desarrollan ejercicios de manera colaborativa, en los Estados Unidos solo en dos casos y en la Argentina en otros dos.

*4) Formato de las Clases.*

Notamos que tanto en los Estados Unidos como en la Argentina sigue predominando el modelo en el cual se dictan clases netamente teóricas para luego dar lugar a las prácticas en el laboratorio, tan solo en tres casos en los Estados Unidos y en un caso en la Argentina se trabaja íntegramente con clases teórico prácticas (Figura 7).

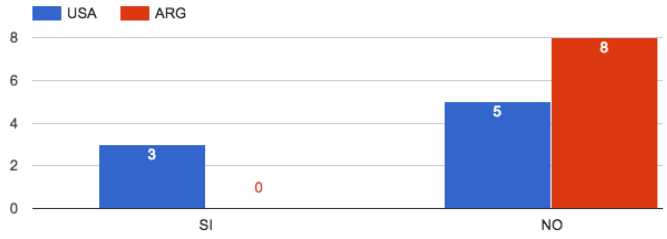



Fig. 6. Realización de prácticas semi-libres.

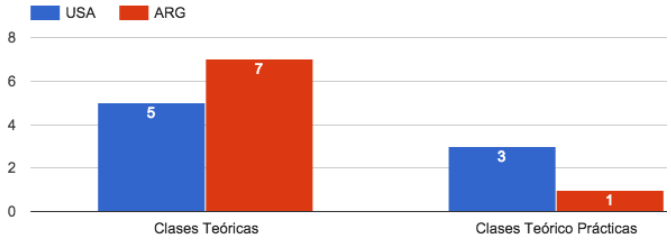



Fig. 7. Formato de las Clases.

Tanto en las universidades de los Estados Unidos como en las de Argentina el modo predominante a la hora de impartir conocimiento es desde el ejemplo, mediante éste método el docente explica el marco teórico y refuerza con ejemplos prácticos para luego permitir ejercitar a los alumnos.

*5) Utilización del Aula Virtual.*

Graham [45] presenta el aprendizaje híbrido como la convergencia de dos ambientes de aprendizaje arquetípicos. Por un lado, están los tradicionales ambientes de aprendizaje cara a cara que han sido usados durante siglos, por otro, se tienen los ambientes de aprendizaje distribuidos que han empezado a crecer y a expandirse de manera exponencial a la par de la expansión de las posibilidades tecnológicas de comunicación e interacción distribuida. Expresa Graham que en el pasado estos dos ambientes de aprendizaje han permanecido ampliamente separados porque constituyen diferentes combinaciones de métodos y medios y se han dirigido a audiencias diferentes.

Se observa en base a los datos relevados que en las universidades de la Argentina existe un aula virtual asociada a la materia pero que en la mayoría de los casos se limita solo a compartir información, ejercicios y material de lectura. Son pocos los que cuentan con alguna herramienta que permita intercambiar opiniones a los alumnos.

En dos de los casos relevados de las universidades de los Estados Unidos, se detectan tres situaciones bien diferenciadas:

a. Los casos que se limitan solo compartir información, ejercicios y material de lectura.

b. Los casos del Massachusetts Institute of Technology y el de Stanford University, en ambas casas de estudio notamos que se busca extender el aula y permitirle al alumno acceder no solo al material teórico sino también a las grabaciones de las clases.
c. El caso de Harvard University, se incorpora a los videos de las clases teóricas tres agregados:
   - Streaming, todas las clases teóricas de Harvard y Yale son transmitidas en vivo y luego quedan a disposición de los alumnos.
   - Walkthroughs, vídeos a través de los cuales los responsables del curso indican a los alumnos por dónde empezar y cómo abordar un desafío. Se espera que el alumno vea estos tutoriales antes de hacer preguntas sobre el problema en horario de oficina o por medio del foro.
   - Postmortems, disponibles después de las fechas límite de entrega de las guías de ejercicio, a través de estos videos el responsable del curso muestra soluciones reales a los problemas planteados en la guía.

*6) Utilización del Aula Virtual.*

A efectos de visualizar cuánto de lo propuesto en Joint Task Force on Computing Curricula [1] es abordado, se elaboró un índice que indica el porcentaje de los temas cubiertos en cada universidad (figura VIII).

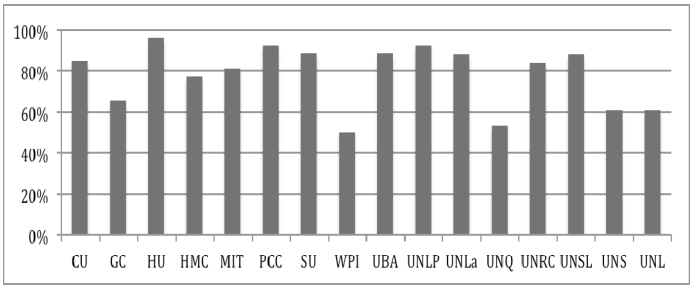



Fig. 8. Índice de cumplimiento de los contenidos.

*a) Algoritmos y Diseño.*

Sobre este punto vale la pena destacar que solo en la mitad de los casos relevados en las universidades de los Estados Unidos se utilizan algoritmos a diferencia de lo que ocurre en la Argentina donde estos siguen teniendo un rol protagónico en las materias introductorias.

*b) Conceptos Fundamentales de Programación.*

En las universidades de los Estados Unidos notamos un bajo índice de cobertura en las operaciones de entrada salida y en el manejo de archivos, el resto los temas planteados en este apartado tienen una buena cobertura en ambos países.

*c) Estructuras de Datos.*

En las universidades de los Estados Unidos es bajo el índice de cobertura en el tema listas enlazadas y el tema de referencias y alising.

*d) Métodos de Desarrollo.*

Tanto en las universidades de los Estados Unidos como en las de Argentina es bajo el índice de cobertura en el tema refactorización simple, entendiendo a éste como un cambio hecho a la estructura interna de un software para hacerlo fácil de entender y mantener sin modificar el comportamiento observable. En la Argentina también se observa una falencia en el abordaje de técnicas que permitan abordar la comprensión de programas de terceros y la posterior corrección de los mismos.

## IV. Anexos

En los anexos se presentan los resultados del relevamiento de información con el protocolo propuesto, el Anexo I corresponde a los resultados de las universidades de los Estados Unidos y el Anexo II a las universidades de la Argentina.

## V. Conclusiones

En este apartado se presenta un resumen de los resultados del trabajo (sección A) y las futuras líneas de investigación surgidas a partir de él (sección B).

### *A. Resumen de los Resultados del Trabajo*

En la primera parte de este trabajo se expuso la problemática, dejando expresado que se busca disminuir los índices de deserción en las materias introductorias de programación. Destacando que este trabajo busca a través de la identificación y comparación de los distintos abordajes poder plantear posibles soluciones que permitan contrarrestar este fenómeno.

En la segunda parte de este trabajo realizamos una breve introducción de sistemas de educación superior tanto en los Estados Unidos como en la Argentina y una pormenorizada descripción del abordaje que le da cada universidad a la enseñanza de la programación, allí se reflejaron los datos recopilados en la etapa de investigación documental.

En la tercera parte se realizó una comparación de los diferentes abordajes, para llevar esta tarea adelante se agrupó el análisis en diferentes bloques temáticos como ser: la utilización de algoritmos, el paradigma de programación empleado, la metodología de ejercitación, el formato de las clases, la utilización del aula virtual y los contenidos de las materias. Esto permitió en cada bloque encontrar aspectos comunes y elaborar un índice que permitió establecer el grado de cumplimiento de cada universidad en lo que respecta al contenido curricular expresado en el documento Joint Task Force on Computing Curricula 2013 [1].

### *B. Futuras Líneas de Investigación*

Según Undurraga y Araya [42] el aprendizaje de los adultos es mayor cuando se usan procesos cognitivos divergentes y cuando procesan el material de aprendizaje por medio de diversas estrategias. Pero además, las competencias tienen características específicas que demandan estrategias didácticas particulares para la formación de cada una de ellas. Los adultos en situación de aprendizaje deben usar diversos canales de exploración y puesta en práctica para promover el desarrollo de dichas competencias. De acuerdo con Le Boterf [43], actuar con competencia remite a proceder de manera pertinente en un contexto particular, eligiendo y movilizando un equipamiento doble de recursos: recursos personales (conocimientos, saber hacer, cualidades, cultura, recursos emocionales) y recursos externos (banco de datos, redes documentales, redes de experiencia especializada, entre otras).

En el trabajo de Mariño [44] se destaca que vivimos en la sociedad de la información, en un mundo globalizado donde los avances tecnológicos han originado profundas transformaciones y en este punto se plantea: ¿Por qué creemos que la educación puede sobrevivir sin hacer ninguna de esas transformaciones y sin utilizar ninguna de las herramientas tecnológicas, que la gente usa para desarrollar

transformaciones en todos los demás sistemas?. Claro está que el abordaje de la enseñanza no puede ser el mismo que se utilizaba hace cinco o diez años, ya que el ritmo de los cambios en la disciplina informática es vertiginoso, por lo tanto se considera que las clases deben ser más dinámicas, guiadas por el docente, pero en donde los estudiantes participen activamente y que las TIC utilizadas correctamente y no meramente como un medio que solo persigue el fin de compartir material, pueden generar grandes beneficios en la educación presencial.

Tanto en las universidades de los Estados Unidos como en las de Argentina se identifican un gran número de casos que presentan dificultades para innovar en propuestas de enseñanza y que tienden a utilizar las aulas como espacios de aprendizaje individual, regidos por extensos programas de estudios que centran el foco en la transmisión de información.

En vista de lo expuesto anteriormente se cree que es justificada la elaboración de un proceso sistémico que de soporte al desarrollo de un dispositivo de enseñanza, dicho proceso debe ser capaz de transformar una pieza de teoría en un conjunto de artefactos tecnológicos que ayuden a la educación del alumno y le permitan desarrollar un entrenamiento de la práctica profesional. El proceso deberá permitir la trazabilidad de dichos artefactos como así también la elaboración de métricas que permitan evaluar la efectividad de los mismos. Dicho proceso deberá considerar en una propuesta educativa diversos recursos, los propios de la educación presencial, más aquellos de la educación a distancia, de manera que dicha combinación apunte a lograr un aprendizaje significativo por parte de los alumnos.

## REFERENCIAS


[1] ACM/IEEE-CS Joint Task Force on Computing Curricula. (2013). Computer Science Curricula 2013. ACM Press and IEEE Computer Society Press.

[2] Arellano, N., Rosas, M. V., Zuñiga, M. E., Fernández, J., y Guerrero, R. (2014). Una experiencia en la enseñanza de la programación para la permanencia de los alumnos de Ingeniería Electrónica.

[3] Barg, M., Fekete, A., Greening, T., Hollands, O., Kay, J., Kingston, J. H. y Crawford, K., Problem based learning for foundation computer science courses. Computer Science Education, 2000

[4] Bayman, P. and Mayer, R. E. 1983. A diagnosis of beginning programmers' misconceptions of BASIC programming statements. Commun. ACM 26, 9 (Sep. 1983), 677-679.

[5] Becker, B. W. (2001, February). Teaching CS1 with karel the robot in Java. InACM SIGCSE Bulletin (Vol. 33, No. 1, pp. 50-54). ACM.

[6] Brito, M., y de Sá-Soares, F. 2014. Assessment frequency in introductory computer programming disciplines. Computers in Human Behavior, 30, 623-628.

[7] Byrne, P., y Lyons, G. 2001. The effect of student attributes on success in programming. In ACM SIGCSE Bulletin (Vol. 33, No. 3, pp. 49-52). ACM.

[8] CU, 2016. Introduction to Programming. Creighton University. http://dave-reed.com/ csc221.F13. Pagina vigente al 28/03/2016

[9] de Jalón, J. G., RODRIGUEZ, J., SARRIEGUI, J., y BRAZALES, A. (1998). Aprenda Lenguaje ANSI C como si estuviera en Primero. Industri Injineruen Goimailako Eskola, Nafarroako Unibertsitatea, San Sebastián.

[10] Estácio, B., Oliveira, R., Marczak, S., Kalinowski, M., Garcia, A., Prikladnicki, R., y Lucena, C.(2015) Evaluating Collaborative Practices in Acquiring Programming Skills: Findings of a Controlled Experiment.

[11] Evangelista, F. y Novarra, P.(2014).Intérprete para probar un programa escrito en pseudocódigo. Industrial Data, 17(1), 101-109.

[12] Ferreira Szpiniak, A., y Rojo, G. A. (2006). Enseñanza de la programación. TE y ET.

[13] Grinnell College GC, 2016. Functional problem solving. Grinnell College. http:// www.cs.grinnell.edu/~davisjan/csc/ 151/2013F/. Pagina vigente al 28/03/2016

[14] HMC, 2016. Introduction to Computer Science. Harvey Mudd College. https://ww w.cs.hmc.edu/twiki/bin/view/CS5. Pagina vigente al 28/03/2016

[15] HU,2016. Introduction to the intellectual enterprises of computer science and the art of programming. Harvard University. https://cs50.harvard. edu/ Pagina vigente al 28/03/2016

[16] Klassner, F., y Anderson, S. D. (2003). Lego MindStorms: Not just for K-12 anymore. IEEE Robotics y Automation Magazine, 10(2), 12-18.

[17] Leone, L., Veizaga, K., Conforte, J., y Zanazzi, J. L. 2014. Modelos para explicar el desgranamiento en una carrera de Ingeniería. En Actas de las XLIII Jornadas Argentinas de Informática e Investigación Operativa (43JAIIO)-XII Simposio Argentino de Investigación Operativa (SIO)(Buenos Aires, 2014).

[18] Malan, D. J. (2010, March). Reinventing CS50. In Proceedings of the 41st ACM technical symposium on Computer science education (pp. 152-156). ACM.

[19] Martínez López, P., Bonelli, E., Sawady, F., de Terramar, U, y Le Guin, U. (2012). El nombre verdadero de la programación. En Simposio sobre la Sociedad de la Información, SSI 2012

[20] Martínez López, E. (2013). Las bases conceptuales de la programación : una nueva forma de aprender a programar. 1ra ed. - La Plata : el autor, 2013. EBook.

[21] McDowell, C., Werner, L., Bullock, H. E., y Fernald, J. (2006). Pair programming improves student retention, confidence, and program quality. Communications of the ACM, 49(8), 90-95.

[22] MIT, 2016. Introduction to Computer Science and Programming. Massachusetts Institute of Technology. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science. Pagina vigente al 28/03/2016.

[23] PCC, 2016. Java Programming I. Portland Community College. http://www.pcc.edu/ ccog/default.cfm?fa=ccogysubject=CISycourse=13. Pagina vigente al 28/03/2016

[24] Porter, L., y Simon, B. (2013, March). Retaining nearly one-third more majors with a trio of instructional best practices in CS1. In Proceeding of the 44th ACM technical symposium on Computer science education (pp. 165-170). ACM.

[25] Porter, L., Bailey Lee, C., Simon, B., Cutts, Q., y Zingaro, D. (2011, June). Experience report: a multi-classroom report on the value of peer instruction. InProceedings of the 16th annual joint conference on Innovation and technology in computer science education (pp. 138-142). ACM.

[26] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... y Kafai, Y. (2009). Scratch: programming for all.Communications of the ACM, 52(11), 60-67.

[27] Rimington, K. B. (2010). Expanding the Horizons of Educational Pair Programming: A Methodological Review of Pair Programming in Computer Science Education Research.

[28] Rooksby, J., Hunt, J., y Wang, X. (2014). The theory and practice of randori coding dojos. In Agile Processes in Software Engineering and Extreme Programming (pp. 251-259). Springer International Publishing.

[29] Simon, B., Kohanfars, M., Lee, J., Tamayo, K., y Cutts, Q. (2010, March). Experience report: peer instruction in introductory computing. In Proceedings of the 41st ACM

technical symposium on Computer science education (pp. 341-345). ACM.

[30] Spohrer , J. C., y Soloway, E. (1986, April). Alternatives to construct-based programming misconceptions. In ACM SIGCHI Bulletin (Vol. 17, No. 4, pp. 183-191). ACM.

[31] SU, 2016. Programming Methodology. Stanford University. http://web.stanford. edu/class/cs106a/. Pagina vigente al 28/03/ 2016

[32] Sung, K., Panitz, M., Wallace, S., Anderson, R., y Nordlinger, J. (2008, March). Game-themed programming assignments: the faculty perspective. In ACM SIGCSE Bulletin (Vol. 40, No. 1, pp. 300-304). ACM.

[33] UBA, 2016. Algoritmos y Programación 1. Universidad de Buenos Aires. http://www.algoritmos7540-rw.tk. Pagina vigente al 25/03/2016

[34] UNL, 2016. Fundamentos de Programación. Universidad Nacional del Litoral http://fich.unl.edu.ar/planificaciones/planificacion.php?id=325yanio=2015ycarrera=3. Pagina vigente al 25/03/2016.

[35] UNLa, 2016. Programa de la Materia Programación de computadoras. Universidad Nacional de Lanús. http://sistemas.unla.edu.ar/sistemas/sls/ls-1-programacion-de-computadoras/pdf/Programa-Programacion-de-Computadora.pdf. Pagina vigente al 25/03/2016.

[36] UNLP, 2016. Algoritmos, Datos y Programas. Universidad Nacional de La Plata. http://www.ing.unlp.edu.ar/progalg/ . Pagina vigente al 25/03/2016.

[37] UNRC, 2016. Introducción a la Algorítmica y Programación. Universidad Nacional de Rio Cuarto. http://dc.exa.unrc.edu.ar/principal/node/24. Pagina vigente al 25/03/2016.

[38] UNS, 2016. Resolución de problemas y algoritmos. Universidad Nacional del Sur. http://cs.uns.edu.ar/materias/rpa/. Pagina vigente al 25/03/2016.

[39] UNSL, 2016. Programación I. Universidad Nacional de San Luis. http://proguno.unsl.edu.ar. Pagina vigente al 25/03/2016.

[40] Wachenchauzer,R., Manterola, M., Curia, M., Medrano,M. y Paez,N. (2012) Algoritmos y Programación I - Aprendiendo a programar usando Python como herramienta. Recuperado el 17 Diciembre 2015, de http://www.algoritmos7540-rw.tk/home/apunte_7540.pdf

[41] WPI, 2016. Introduction to Program Design. Worcester Polytechnic Institute. http://web.cs.wpi.edu/~cs1101/a15/. Pagina vigente al 28/03/ 2016.

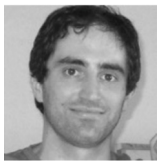
[42] Undurraga, C., & Araya, C. (2001). El estado de la enseñanza de la formación en gestión y política educativa en Chile. op. cit.

[43] Le Boterf, G. (2000). Construire les compétences individuelles et collectives. Ed. d'Organisation.

[44] Mariño, J. C. G. (2006). B-Learning utilizando software libre, una alternativa viable en Educación Superior. Revista complutense de Educación, 17(1), 121-133.

[45] Graham, C. R. (2006). Blended learning systems. The handbook of blended learning, 3-21.

[46] Moroni, N., & Señas, P. (2005). Estrategias para la enseñanza de la programación. In I Jornadas de Educación en Informática y TICs en Argentina.)



**Mauricio R. Dávila** Es Licenciado en Informática por la Universidad Atlántida Argentina. Es Candidato del Programa de Magister en Ingeniería de Sistemas de Información de la Escuela de Postgrado de la Facultad Regional Buenos Aires de la Universidad Tecnológica Nacional. Es Coordinador Académico de la Tecnicatura Superior en Programación y de la Tecnicatura Superior en Sistemas Informáticos de la Facultad Regional Avellaneda de la Universidad Tecnológica Nacional. Es Docente de las asignaturas Programación I y Laboratorio de Computación I de la Tecnicatura Superior en Programación de la Facultad Regional Avellaneda de la Universidad Tecnológica Nacional. Es Investigador Tesista del Laboratorio de Investigación y Desarrollo en Espacios Virtuales de Trabajo de la Licenciatura en Sistemas y de la Maestría en Sistemas de Información del Departamento de Desarrollo Productivo y Tecnológico de la Universidad Nacional de Lanús.

**ANEXO I**. RESULTADOS DEL RELEVAMIENTO DE INFORMACIÓN DE LAS UNIVERSIDADES DE LOS ESTADOS UNIDOS

| | | | CU | GC | HU | HMC | MIT | PCC | SU | WPI |
|---|---|---|---|---|---|---|---|---|---|---|
| **Contenido** | **Algoritmos y Diseño** | El concepto y propiedades de los algoritmos | SI | NO | SI | SI | NO | NO | SI | NO |
| | | El papel de los algoritmos en el proceso de resolución de problemas | SI | NO | SI | SI | NO | NO | SI | NO |
| | | Las estrategias de resolución de problemas | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Conceptos de diseño | SI | SI | SI | SI | SI | SI | SI | SI |
| | **Conceptos Fundamentales de Programación** | Sintaxis básica y la semántica de un lenguaje de alto nivel | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Variables y tipos de datos primitivos | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Expresiones y asignaciones | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Operaciones de E / S | SI | NO | SI | NO | SI | SI | SI | NO |
| | | Archivos de E / S | SI | NO | SI | NO | NO | SI | SI | NO |
| | | Estructuras de control condicionales e iterativas | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Funciones y pasaje de parámetros | SI | SI | SI | SI | SI | SI | SI | SI |
| | | El concepto de recursividad | SI | SI | SI | SI | SI | SI | NO | SI |
| | **Estructuras de datos** | Listas | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Cadenas y procesamiento de cadenas | SI | SI | SI | SI | SI | SI | SI | NO |
| | | Tipos abstractos de datos y manejo dinámico de memoria | SI | NO | SI | SI | SI | SI | SI | NO |
| | | Referencias y aliasing | NO | NO | SI | NO | SI | SI | SI | NO |
| | | Las listas enlazadas | NO | NO | SI | NO | NO | SI | NO | SI |
| | | Estrategias para la elección de la estructura de datos apropiada | SI | SI | SI | SI | SI | SI | SI | NO |
| | | Búsquedas | SI | SI | SI | SI | SI | SI | SI | NO |
| | | Ordenamiento | NO | SI | SI | SI | SI | SI | SI | NO |
| | **Métodos de Desarrollo** | Comprensión Programas | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Corrección del Programas | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Refactorización simple | NO | NO | NO | NO | NO | SI | NO | NO |
| | | Entornos de programación modernos | SI | NO | SI | SI | SI | SI | SI | NO |
| | | Estrategias de depuración | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Documentación y reglas de estilo | SI | SI | SI | NO | SI | SI | SI | NO |
| **Didáctica** | **Formato clases** | ¿Se dictan clases netamente teóricas? | NO | NO | SI | SI | SI | NO | SI | SI |
| | | ¿Se dictan clases prácticas? | SI | SI | SI | SI | NO | SI | SI | SI |
| | **Ejercitación** | ¿Se realizan ejercicios de manera colaborativa? | NO | SI | SI | NO | NO | NO | NO | SI |
| | | ¿Se realizan ejercicios desde cero? | SI | SI | SI | SI | SI | SI | SI | SI |
| | | ¿Se realizan ejercicios a completar? | NO | NO | SI | NO | SI | SI | NO | NO |
| | **Modo Enseñanza** | ¿Se enseña desde el ejemplo? | SI | SI | SI | SI | SI | SI | SI | SI |
| | | ¿Se enseña guiado por el problema? | SI | NO | NO | NO | NO | NO | NO | NO |
| **Herramientas** | **Lenguaje** | ¿En qué paradigma se programa? | Mixto | Funcional | Imperativo | Mixto | Mixto | OO | OO | Funcional |
| | | ¿En qué lenguaje se programa? | Python | Scheme | C/PHP/JS | Python | Python | Java | Java | Racket |
| | | ¿En qué entorno se programa? | IDLE | Gimp | CS50 IDE | IDLE | IDLE | BLUEJ | ECLIPSE | DrRacket |
| | **Algo-ritmia** | ¿Cual es el entorno? | Scratch | - | Scratch | Picobot | - | - | Karel | - |
| | | ¿Que leguaje utiliza? | Scratch | - | Scratch | Picobot | - | - | Java | - |
| | **Aula virtual** | ¿Cuenta con aula virtual la materia? | SI | SI | SI | SI | SI | SI | SI | SI |
| | | ¿El aula virtual se limita a compartir material? | SI | SI | NO | SI | SI | SI | SI | SI |
| | | ¿Brinda el aula virtual herramientas de autoevaluación? | NO | NO | SI | NO | NO | NO | NO | NO |

**ANEXO II**. RESULTADOS DEL RELEVAMIENTO DE INFORMACIÓN DE LAS UNIVERSIDADES DE LA ARGENTINA

| | | | UBA | UNLP | UNLa | UNQ | UNRC | UNSL | UNS | UNL |
|---|---|---|---|---|---|---|---|---|---|---|
| Contenido | Algoritmos y Diseño | El concepto y propiedades de los algoritmos | SI | SI | SI | SI | SI | SI | SI | SI |
| | | El papel de los algoritmos en el proceso de resolución de problemas | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Las estrategias de resolución de problemas | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Conceptos de diseño | SI | SI | SI | SI | SI | SI | SI | SI |
| | Conceptos Fundamentales de Programación | Sintaxis básica y la semántica de un lenguaje de alto nivel | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Variables y tipos de datos primitivos | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Expresiones y asignaciones | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Operaciones de E / S | SI | SI | SI | NO | SI | SI | SI | SI |
| | | Archivos de E / S | SI | SI | SI | NO | SI | SI | SI | NO |
| | | Estructuras de control condicionales e iterativas | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Funciones y pasaje de parámetros | SI | SI | SI | SI | SI | SI | SI | SI |
| | | El concepto de recursividad | SI | SI | SI | NO | SI | SI | SI | NO |
| | Estructuras de datos | Listas | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Cadenas y procesamiento de cadenas | SI | SI | SI | NO | SI | SI | SI | SI |
| | | Tipos abstractos de datos y manejo dinámico de memoria | SI | SI | SI | NO | SI | SI | NO | NO |
| | | Referencias y aliasing | SI | SI | SI | NO | SI | SI | NO | SI |
| | | Las listas enlazadas | SI | SI | SI | NO | SI | SI | NO | NO |
| | | Estrategias para la elección de la estructura de datos apropiada | SI | SI | SI | NO | SI | SI | NO | SI |
| | | Búsquedas | SI | SI | SI | SI | SI | SI | NO | NO |
| | | Ordenamiento | SI | SI | SI | SI | SI | SI | NO | NO |
| | Métodos de Desarrollo | Comprensión Programas | NO | SI | SI | NO | NO | NO | NO | NO |
| | | Corrección del Programas | NO | SI | NO | NO | NO | NO | NO | NO |
| | | Refactorización simple | NO | NO | NO | NO | NO | NO | NO | NO |
| | | Entornos de programación modernos | SI | SI | SI | NO | NO | SI | NO | SI |
| | | Estrategias de depuración | SI | SI | SI | SI | SI | SI | SI | SI |
| | | Documentación y reglas de estilo | SI | NO | NO | SI | SI | SI | SI | NO |
| Didáctica | Formato clases | ¿Se dictan clases netamente teóricas? | SI | SI | NO | SI | SI | SI | SI | SI |
| | | ¿Se dictan clases prácticas? | SI | SI | SI | SI | SI | SI | SI | SI |
| | Ejercitación | ¿Se realizan ejercicios de manera colaborativa? | SI | NO | NO | NO | SI | NO | NO | NO |
| | | ¿Se realizan ejercicios desde cero? | SI | SI | SI | SI | SI | SI | SI | SI |
| | | ¿Se realizan ejercicios a completar? | NO | NO | NO | NO | NO | NO | NO | NO |
| | Modo Enseñanza | ¿Se enseña desde el ejemplo? | SI | SI | SI | SI | SI | SI | SI | SI |
| | | ¿Se enseña guiado por el problema? | NO | NO | NO | NO | NO | NO | NO | NO |
| Herramientas | Lenguaje | ¿En qué paradigma se programa? | Imperativo | Imperativo | Imperativo | Imperativo | Imperativo | Imperativo | Imperativo | Imperativo |
| | | ¿En qué lenguaje se programa? | Python | C | C | Gobstones | PASCAL | C | PASCAL | C++ |
| | | ¿En qué entorno se programa? | Geany | Code::Blocks | DEV C++ / Code::Blocks | Code::Blocks | TURBO PASCAL | Code::Blocks | TURBO PASCAL | Zinjal |
| | Algoritmia | ¿Cual es el entorno? | - | - | - | - | - | - | - | - |
| | | ¿Que leguaje utiliza? | - | - | - | Gobstones | - | Timba | - | - |
| | Aula virtual | ¿Cuenta con aula virtual la materia? | SI | SI | SI | SI | SI | SI | SI | SI |
| | | ¿El aula virtual se limita a compartir material? | SI | SI | SI | SI | SI | SI | SI | SI |
| | | ¿Brinda el aula virtual herramientas de autoevaluación? | NO | NO | NO | NO | NO | NO | NO | NO |