CURSO INTRODUCTORIO DE COMPUTACION CON MUY ESCASOS RECURSOS. UNA EXPERIENCIA

Juan V. Echague
Facultad de Ingeniería
Montevideo - *Uruguay*

RESUMEN

Se relata la experiencia de un curso introductorio a la informática en carreras universitarias de formación de profesionales en esa área, dictado con muy escasos recursos.

El curso se organiza alrededor de la programación como resolución de problemas. La atención se centra en la especificación, la programación imperativa y la interacción de estos dos elementos. Se utiliza como lenguaje de especificación la lógica (cláusulas de Horn), y como lenguaje de programación un subconjunto de Pascal.

Fue dictado durante el semestre de otoño de 1986 por el Departamento de Programación del Instituto de Computación (Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay).

1) Antecedentes.

La Universidad de la República ofrece dos carreras en informática: 'Ingeniero de Sistemas en Computación' y 'Analista Programador', ambas por medio de la Facultad de Ingeniería.

Fueron creadas en el año 1974, al inicio del período de intervención universitaria impuesta por el gobierno de facto. Los legítimos gobernantes de la Universidad no pudieron discutir ni decidir sobre los presupuestos básicos, los títulos ni las currícula de estas carreras desde su creación hasta el cese de la intervención, en 1984.

En 1985 la Comisión de Area de Computación eleva al Consejo de Facultad de Ingeniería una propuesta de modificaciones de contenidos de materias a fin de lograr 'una rápida actualización' de estos (CIn85, CAC85). Esa propuesta es aprobada.

Introducción a la Computación es una materia semestral, ubicada al inicio de las carreras en informática. En el año 1986 se dicta por vez primera como tarea del Departamento de Programación del Instituto de Computación, con el contenido propuesto y aprobado por el gobierno universitario legítimo.

La Comisión de Area de Computación (CAC85) recomienda para ella:

'Objetivos del curso:
- Introducir las nociones básicas de lógica.
- Introducir a la metodología de especificar y resolver problemas (de procesamiento de datos en particular).
- Introducir al diseño de algoritmos simples.
- Introducir un lenguaje de alto nivel (Pascal).'

2) Situación material del dictado del curso 1986.

La matrícula en el curso fue de 1248 estudiantes. Lo que mantiene la tendencia de matrícula numerosa y creciente presente desde hace varios años.

El curso fue dictado por 3 (tres) docentes, que sumaban una dedicación semanal de 110 (ciento diez) horas.

La relación tiempo-docente/estudiante es de 3.8 minutos semanales para este curso (el promedio del Departamento fue en ese semestre de 9 minutos).

3) El curso.

3.1) Los mecanismos usados para tomar las decisiones.

Las decisiones fundamentales en cuanto a contenido y forma del curso reflejan consensos alcanzados en reuniones y seminarios del Departamento de Programación.

En las reuniones participaron docentes del Departamento de Programación y estudiantes avanzados. Su objetivo fue la preparación de todos los cursos del semestre de otoño de 1986, tanto en su contenido como en su forma. Se desarrollaron a partir de mediados de febrero hasta el inicio de los cursos, a principios de mayo.

Los seminarios fueron dirigidos por docentes del Departamento de Programación, con asistencia de docentes y estudiantes avanzados. Cubrieron los temas cláusulas de Horn y metodología de resolución de problemas. Sus objetivos fueron: 1) profundización en el tema, 2) detección de los puntos de mayor dificultad para los estudiantes y 3) ensayo de distintas maneras de introducir los conceptos.

3.2) El contenido del curso.

El contenido del curso se organizó alrededor de las nociones de especificación y programación, articuladas en el marco del tema 'resolver problemas'. Cada una de estas nociones se enfrentó, como veremos mas adelante desde un marco conceptual distinto.

Esta dualidad es propia del estado de la discusión e investigación en el área de programación. Es ineludible enfrentarla hoy en el estudio de la programación. Hacerlo lo antes posible fue una de las decisiones de diseño mas profundas de este curso.

En los trabajos de preparación se revisaron, como acercamientos posibles a la especificación, técnicas funcionales, algebraicas, lógicas y de pre-condición/pos-condición, además de la opción por defecto de especificar problemas en lenguaje natural (L&Z77).

Se optó por la especificación en lógica, en particular, utilizando cláusulas de Horn. La lógica formal es ampliamente aceptada como lenguaje de especificación en la ciencia de la computación. Se adapta de manera ideal a la representación del conocimiento y a la descripción de problemas, independiente de la elección de un lenguaje de programación (Kow84).

Otra razón, para nada menor, fue la disponibilidad imediata de un texto sobre el tema: 'Logic for problem solving' (Kow79), que no reclama conocimientos previos de lógica ni programación.

Además del estudio de los temas de lógica y especificación, el manejo de cláusulas de Horn como herramienta de trabajo fue muy fértil al permitir ejemplificar metodologías ascendentes y descendentes (Kow79) y, mediante ejercicios, presentar las nociones de recurrencia, decidibilidad, gramáticas y lenguajes.

Algunas técnicas de resolución de problemas ('path-finding' y 'and-or trees') tienen una representación muy directa en cláusulas de Horn (Kow79), por lo que este tema se enfrentó con este mismo esquema conceptual.

En cuanto a programación el objetivo planteado fue el estudio de bases conceptuales de algún paradigma de los lenguajes de programación, prestando principal atención a los aspectos 'semánticos', dejando en un lugar secundario los 'sintácticos'.

En la preparación del curso se revisaron como opciones los paradigmas de lenguages lógicos, funcionales e imperativos. Los argumentos a favor de los dos primeros se centraron en la claridad y el desarrollo alcanzados en ellos (Fur86), y la convicción que encierran, en alguna forma, los lenguajes de programación del futuro cercano.

Sin embargo se optó finalmente por estudiar los fundamentos de los lenguajes imperativos, entendiendo estos como aquellos cuya característica central es permitir la creación de variables (Hor84). Dos razones poderosas para esta decisión fueron la familiaridad del cuerpo docente con lenguajes de este tipo y su amplia difusión en el medio.

En el curso se estudió una máquina teórica, de arquitectura Von Neumann. Luego se introdujeron nociones de lenguajes de alto nivel imperativos (variable, asignación, tipo, secuencia, selección e iteración) como modelaciones, de 'mas alto nivel' del lenguaje absoluto de esa máquina. Se tomaron pequeños ejemplos de media docena de lenguajes de alto nivel, poniendo acento en las identidades 'semánticas' existentes detrás de las diferencias 'sintácticas'.

Se postergo tanto como fue posible, la introducción de la sintaxis de un lenguaje de programación en particular. Con esto buscamos minimizar el 'efecto de lengua materna' que tiene el primer lenguaje de programación que se aprende.

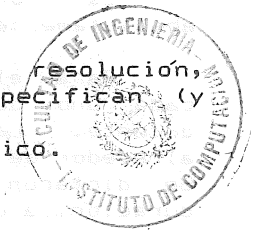
El lenguaje de programación utilizado fue una versión reducida de Pascal (J&W75). Soporta unicamente variables enteras, lógicas y de caracteres, sentencias de asignación, de selección (solamente 'if'), de iteración (solamente 'while' y 'repeat') y entrada/salida sobre archivos 'standard' (como sentencias). No se incluye reales, tipos estructurados ni subprogramas. En una sesión de alrededor de una hora se explicó el manejo de la cartilla sintáctica y se señalaron las correspondencias con la 'semántica' ya vista.

El difícil acople de los dos esquemas conceptuales fue también analizado en el curso teórico en varias oportunidades. Allí se pasó revista al proceso de resolución de problemas utilizando una computadora, los orígenes históricos de las nociones de especificación y programación imperativa, los 'distintos niveles de abstracción', su influencia en el costo de la actividad humana de programar y las posibles alternativas de futuro.

Podemos resumir el contenido del curso en el siguiente esquema:

Parte 1. Especificación.

Orientada a los problemas y las técnicas de resolución, con un 'alto nivel de abstracción'. Se especifican (y resuelven) problemas con lógica de cláusulas.
Ocupa el 60% del teórico y el 85% del práctico.



Parte 2. Programación.

Se presenta como actividad de resolución (posiblemente automatizable en parte) de problemas. Se discute el papel de la especificación y de un agente (quizas mecánico) que puede realizar acciones. Se presentan también las nociones de procedimiento y algoritmo.
Ocupa el 4% del teórico.

Parte 3. La máquina de Von Neumann.

Se estudia un modelo simplificado de computador con arquitectura Von Neumann. Memoria, unidad de proceso y conjunto de instrucciones. Es el punto de 'menor nivel de abstracción' del curso. Se trabaja alrededor de algunos problemas, comparando su especificación y un programa (en lenguaje absoluto de esa máquina) para resolverlo.
Ocupa el 9% del teórico.

Parte 4. Los lenguajes imperativos.

Se introducen como una forma mas cómoda de trabajo en una máquina Von Neumman, que permiten subir en el 'nivel de abstracción'. Se tratan las nociones de lenguajes de alto nivel, compilador, variable, expresión, asignación, tipo, secuencia, selección e iteración, con ejemplos en varios lenguajes de programación. El acento es puesto en las nociones de variable y asignación.
Ocupa el 18% del teórico.

Parte 5. Un lenguaje imperativo minimo.

Se introduce la sintaxis de un subconjunto reducido de Pascal, haciendo referencia a la semántica vista en la parte 4. Se muestran ejemplos.
Ocupa un 9% del teórico y un 15% del práctico.

3.3) La forma del curso.

La forma de dictado del curso estuvo determinada absolutamente por las condiciones materiales.

Para el dictado de teórico se trabajó en tres grupos, cada uno de alrededor de 400 estudiantes a cargo de un docente. La asistencia real fue (sumando los grupos) de alrededor de 600 estudiantes, estable a lo largo del curso. Se dictaron para cada grupo 30 clases de 2 horas de duración, a un ritmo de 3 clases semanales.

Los cursos prácticos estuvieron a cargo del gremio estudiantil (Centro de Estudiantes de Ingeniería). Se organizaron asi 33 grupos de práctico (llamados 'grupos de auto-estudio' por razones históricas) que se reunían 2 horas semanales a cargo de 135 estudiantes-colaboradores. Es muy difícil obtener datos sobre su funcionamiento, podemos evaluar que globalmente 400 estudiantes participaron.

Los ejercicios practicos (42 a lo largo de todo el curso) fueron propuestos por el Departamento como material de trabajo para esos grupos. Fueron planteados en sesiones de trabajo de estudiantes-colaboradores asistidos por un docente con una semana de antelación a su presentación en los 'grupos de auto-estudio'. Una resolución modelo era distribuida luego a los estudiantes-colaboradores.

Todos los ejercicios prácticos fueron 'de escritorio'. No hubo en ningún momento acceso de los estudiantes a computadoras.

Como bibliografía se empleó el texto de Kowalski ya citado (los cuatro primeros capítulos) para la primera parte del curso. Y un folleto de apuntes editado por la Facultad de Ingeniería (CET86) para el resto.

El curso fue completamente libre. No se controló asistencia ni se tomaron pruebas parciales.

5) Una primera evaluación.

A la fecha de escribir este trabajo no se conocen los resultados del primer examen de evaluación de este curso. Por lo tanto, solo pueden hacerse consideraciones subjetivas sobre esta experiencia.

Se cumplió globalmente lo propuesto en cuanto contenido del curso. La respuesta estudiantil ante esto fue variada, aunque podemos describir algunas pautas recurrentes.

La presentación de la lógica como herramienta de especificación de problemas fue aceptada con naturalidad por parte de los estudiantes, accediendo sin mayor dificultad a nociones de recursión, unificación, etc. El tema fue considerado globalmente 'fácil'. Hubo cierto desconcierto atribuible a la inadecuación del tema a las fantasías de los estudiantes sobre cuales son los temas 'importantes' en computación.

El tema de la programación imperativa y de un lenguaje de programación imperativo fue considerado por los estudiantes muy árido. Se lo valoraba como un tema muy importante. Muchos estudiantes complementaron el curso en el conocimiento del Pascal a traves de libros y manuales.

La disponibilidad de un texto para la primera parte fue muy importante. Tradicionalmente, en la carrera no existían 'textos del curso'. Eso, unido a que solo existe en versión inglesa, demoró el acceso de los estudiantes a él. Sin embargo, ante la inminencia del examen, el aprovechamiento del libro creció.

El tema programación imperativa tuvo varios problemas: no se dispuso de un texto adecuado, no se elaboraron prácticos sobre el tema ni se presentaron técnicas de resolución de problemas mediante algoritmos. Posiblemente el primero de los problemas fue el origen de los otros.

Fue claramente perceptible, sobre el fin del curso, el alto desgaste en los docentes.

6) Conclusiones.

En lo referente a cursos introductorios para carreras en informática, dictados en situaciones de muy escasos recursos, hay algunos elementos que esta experiencia contribuye a apoyar, sin ser concluyente al respecto.

Sería posible incluir herramientas de alto nivel de abstracción, con un aprovechamientos satisfactorio. Esto permitiría introducir facilmente conceptos fundamentales.

Sería posible incluir algunos elementos de la discusión académica en temas básicos del momento. Esto parece una posición necesaria en un terreno en que los cambios se procesan tan rapidamente.

Bibliografía.

CAC85. Resumen de cambios propuestos al plan de estudios de Ingeniería de Sistemas en Computación y Analista Programador. Comisión de Area de Computación, Facultad de Ingeniería. Julio de 1985.

CIn85. Informe de la Comisión de Informática al Consejo Interino de la Facultad de Ingeniería. Febrero de 1985.

CET86. Apuntes sobre programación imperativa. Crispino G., Echague J., Tasistro A. 1986. Publicación interna de la Facultad de Ingeniería.

Fur86. Paradigmas de lingugens de computaçao. Furtado. 1986. Editora da Unicamp.

Hor84. Fundamentals of Programming Languages (2nd edition). Horowitz, Ellis. 1984. Springer-Verlag Berlin Heidelberg New York Tokyo.

Kow79. Logic for problem solving. Kowalski, R. A. 1979. New York, Amsterdam: Elsevier.

Kow84. The relation between logic programming and logic specification. Kowalski, R. A. 1984. En 'Mathematical Logic and Programming Languages', editado por C.A.R. Hoare y J.C.Shepherdson, Prentice/Hall International.

J&W75. Pascal. User manual and Report. Jensen, Kathleen y Wirth, Niklaus.

L&Z77. An Introduction to formal specifications of data abstractions. Liskov, Barbara & Zilles, Stephen. 1977. En 'Current trends in programming methodology', volumen 1 'Software Specification and Design' editado por R. T. Yeh. Prentice - Hall Inc.