XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa
Buenos Aires, Setiembre 1988.

LUBO-1: UN MODELO DE MAQUINA ABSTRACTA PARA UN PRIMER CURSO UNIVERSITARIO DE PROGRAMACION.

Patricia Pesado. (*)
Maria C. Madoz. (*)
Lia H. Molinari. (*)
Armando E. De Giusti. (**)

(*) Profesora Adjunta del Dpto. de Informática de la Facultad de Ciencias Exactas de la UNLP.

(**) Prof. Titular de la Fac. de Cs. Exactas de la UNLP.
Investigador Independiente del CONICET.

Laboratorio de Investigación y Desarrollo en Informática (LIDI)
Facultad de Ciencias Exactas - Univ. Nac. de La Plata -
50 y 115 - 1er. Piso (1900) La Plata -

RESUMEN:

Se presenta un anàlisis del modelo de màquina abstracta adoptado en 1988 para el curso de Programaciòn de Computadoras (1er. Año de la Licenciatura en Informàtica de la UNLP), que ha sido una evoluciòn de experiencias anteriores realizadas en el Dpto. de Informàtica de la UNLP.-
El aporte màs novedoso es la combinaciòn de primitivas de movimiento (o graficaciòn) con las estructuras de control de un lenguaje procedural "Pascal-like", en un contexto sintàctico preciso.-

INTRODUCCION:

Los contenidos, la metodologia y las herramientas didàcticas utilizables en un primer curso de Programaciòn han sido motivo de numerosos trabajos y experiencias (Ref. 1,2,3,4,5).-
De todos modos una conclusiòn bastante generalizada es la utilidad que se puede obtener del empleo de un modelo abstracto de màquina con un conjunto de instrucciones reducido que pueda extenderse gradualmente. Conceptualmente interesa desprender al alumno de cualquier aspecto "de implementaciòn", tanto en la arquitectura como en el lenguaje, de modo de expresar operaciones abstractas sobre objetos simbòlicos en las que el aspecto primordial sea la correcciòn algoritmica. (Ref. 6,7).-

Enfoques clàsicos y muy conocidos en este sentido han sido los modelos del LOGO (Ref. 8, 9), del robot KAREL (Ref. 10) y de la calculadora evolutiva (Ref. 11). Universidades muy importantes en el mundo los han utilizado en cursos iniciales de programaciòn.-

Sin embargo, desde el punto de vista de la formaciòn esperada para un alumno de un curso de Programaciòn existe un importante "gap" entre el tipo de problemas soluble en el lenguaje reducido de estas màquinas abstractas y la implementaciòn posterior de soluciones en un lenguaje "profesional" tal como PASCAL, C o PROLOG.-

En la carrera de Informàtica de la UNLP, aùn antes de la creaciòn de la Licenciatura en Informàtica (1985), se han realizado una serie de experiencias basadas en diferentes modelos de aprendizaje para el alumno de un primer curso de Programaciòn que oscilaron entre dos extremos:

** El planteo convencional directo "Algoritmo - Lenguaje de Implementaciòn", que incluyò experiencias con FORTRAN, PL/1 y PASCAL.-

** La utilizaciòn de "Màquina abstracta + Seudocòdigo", que no incluìa instrucciones "ejecutables".-

Las mayores criticas al primer enfoque fueron, lògicamente, la distorsiòn en el aprendizaje de los alumnos por el esfuerzo en manejar los detalles del lenguaje involucrado, especialmente en el tratamiento de estructuras de datos.-

Recìprocamente el segundo enfoque, si bien correcto desde el punto de vista de la formaciòn algoritmica, era de difìcil coordinaciòn con los cursos posteriores y ademàs quitaba el aspecto de motivaciòn generado por la realizaciòn de programas "ejecutables" por los alumnos.-

Nuestro trabajo consistió en analizar críticamente las experiencias previas y proponer un modelo, aplicado a partir de 1988, que parte de una máquina abstracta con un lenguaje gradual que evoluciona hacia un PASCAL+ de modo que el alumno culmina el curso realizando programas "reales" en PASCAL sobre una arquitectura tipo PC.-

Un aspecto importante en este planteo fue "extender" los tipos de datos standard del Turbo Pascal de modo que el alumno en una primer etapa manejara naturalmente las estructuras de pilas y colas sin entrar en su implementación utilizando otro tipo primitivo. (Ref. 12).-

LOS OBJETIVOS DE UN PRIMER CURSO DE PROGRAMACION Y SU RELACION CON EL MODELO DE MAQUINA ABSTRACTA:

Una rápida caracterización de los objetivos de un primer curso de Programacion relaciona los mismos con dos contenidos: algorítmica y estructuras de datos.-

El alumno debe aprender las estructuras de control que caracterizan la programación estructurada y su utilización a fin de resolver algoritmos de diferente tipo y complejidad.-

Simultáneamente el alumno incorpora gradualmente las estructuras de datos: datos primitivos (numéricos y lógicos); conjuntos; registros; pilas; colas; arreglos; archivos; listas y por fin una generalización al concepto de tipo de dato abstracto. (Ref. 13).-

Un aspecto importantísimo dentro de los objetivos es el desarrollo de una metodología top-down en el planteo y análisis previo de los problemas que se expresaran algorítmicamente. Interesa especialmente tratar los problemas de modo jerárquico, modularizando en bloques funcionales y tratando de que el tratamiento algorítmico de cada módulo se haga por refinamientos sucesivos.-

Más allá de las justificaciones de simplicidad o fácil comprensión de "la máquina" que pueden dar ejemplos como la tortuga o el robot, este tipo de máquinas abstractas brinda dos características de especial interés en relación con los objetivos mencionados anteriormente:

** El "tipo" de problemas gráficos (dibujos - recorridos) son naturalmente modularizables y el alumno incorpora rápidamente una metodología de trabajo como la deseada.-

** Se puede introducir también naturalmente el concepto de "tipo de datos" asociado con "objeto": por ejemplo un robot como Karel recoge flores y con ellas puede realizar un conjunto de operaciones (depositar, recoger, guardar en su bolsa) y un conjunto de verificaciones (hay flor?, color=, bolsa llena?) que resultan asociadas sin esfuerzo al tipo de datos en cuestión.-

Con este análisis previo, es interesante hacer algunos comentarios sobre el contexto de trabajo y luego detallar las características de LUBO-1.-

EL CONTEXTO:

A- Los alumnos:

Los alumnos que comienzan un curso de programaciòn en la Universidad, constituyen un grupo heterogèneo en conocimientos sobre el tema. Podrìamos hacer una somera clasificaciòn en:

** los que no poseen siquiera conceptos mìnimos de programaciòn.

** los que han realizado un curso sobre algùn lenguaje, incluso algunos alumnos de este grupo poseen experiencia laboral en el tema.

** los que tienen experiencia como "operadores" (entrada de datos, manejo de menùes, juegos).

Los del primer grupo, son alumnos reticentes a la discusiòn de conceptos bàsicos de Informàtica. Habitualmente su visiòn de lo que significa una carrera en esta àrea està muy distorsionada.-
Sin embargo, superada una primer "adaptaciòn"aceptan màs fàcilmente los modelos de abstracciòn, sin impaciencia. Son màs proclives a la especificaciòn, al "porquè" màs que al "còmo".

El segundo grupo es, quizà, el màs conflictivo. Se les ha enseñado a codificar. Es muy difícil que hayan desarrolado un buen estilo de programaciòn con todo lo que ello implica: anàlisis, documentaciòn, uso de estructuras de control adecuadas al algoritmo a utilizar, criterios de correcciòn, etc.-
Y ademàs... estàn seguros de "saber suficiente" del tema.-

El tercer grupo logra una ràpida visualizaciòn en lo que respecta a la importancia en la presentaciòn de un sistema. Conoce de las incompatibilidades surgidas por una especificaciòn de requerimientos ambigua. La confianza con que el operador ejecuta un programa està estrechamente relacionada con una comunicaciòn clara y precisa entre el usuario y el sistema.-
Sin embargo, se trata de alumnos "ansiosos" que creen que toda la Informàtica termina en "un hombre y su terminal". Es muy difícil desarrollar en ellos una valorizaciòn de la abstracciòn y de los criterios generales de los que se derivan aplicaciones.-

Un complemento poco beneficioso es el escaso conocimiento general de los alumnos sobre Lògica y en general sobre Matemàticas: la mayoría de los alumnos (y lamentablemente algunos docentes de ambas disciplinas tambièn...) conciben dos mundos: "el de las matemàticas y el de la computaciòn"; incluso los enfrentan.-
Una de las tareas màs arduas en la enseñanza de programaciòn es la revalorizaciòn de los fundamentos lògicos y matemàticos buscando una formalizaciòn del anàlisis y resoluciòn de problemas.-

B- Los requerimientos de los cursos posteriores:

Una breve síntesis de las expectativas de los cursos posteriores al de Programación de Computadoras puede ser la siguiente:

** Que el alumno maneje adecuadamente las estructuras de control de un lenguaje tipo PASCAL.-

** Que conozca la noción de tipo de datos y tenga capacidad de abstracción de nuevos tipos, tratados como objetos y más allá de los detalles particulares de implementación.-

** Que haya desarrollado una metodología mínima de análisis de problemas top-down y que sepa aplicar refinamientos sucesivos ante un problema relativamente complejo.-

** Que tenga nociones conceptuales de corrección y que relacione claramente el proceso de especificación y documentación con el de prueba de corrección.-

** Que entienda los parámetros de medida de eficiencia de un algoritmo y pueda comparar alternativas algorítmicas.-

Otro aspecto importante a tener en cuenta es que un alto porcentaje de los alumnos opta por la salida intermedia (tres años), lo que significa que al menos debe superar primer año desarrollando aptitudes para el análisis y resolución de problemas "in the small" en un lenguaje ejecutable.-

Con este marco, las orientaciones lógicas fueron:

1- Adoptar una máquina abstracta con capacidad de recorrido o graficación, pero con un lenguaje orientado a PASCAL.-

2- Desarrollar rápidamente la evolución de la máquina hacia la resolución de problemas más abstractos que no requieran el empleo de las primitivas gráficas o de recorrido.-

3- Incorporar al Pascal a utilizar por los alumnos (Turbo Pascal sobre microcomputadora) una serie de bibliotecas desarrolladas por los auxiliares de la cátedra, a fin de reducir al mínimo los problemas "de implementación" respecto de estructuras de datos clásicas como pilas y colas.-

ANALISIS DE ALTERNATIVAS POSIBLES:

A- Tortuga/Lenguaje LOGO:

A nuestro juicio, las caracteristicas màs destacables del lenguaje LOGO estàn fundadas en los siguientes aspectos:

* Permite resolver problemas de distinta complejidad.
* Fomenta la modularidad.
* Permite estructuraciòn.
* Maneja naturalmente la recursividad.

Naturalmente el alumno de LOGO se inclina a escribir procedimientos asociados univocamente a las componentes funcionales del problema en cuestiòn. Esto implica evidentemente un mètodo de desarrollo top-down para el anàlisis y el diseño del programa. Por lo tanto la soluciòn de un problema se logra fàcilmente por la combinaciòn de los procedimientos desarrollados obteniendo como resultado un programa ampliamente modular.-

Por otro lado es inmediato advertir la posibilidad de resolver desde problemas muy simples hasta problemas muy complicados, introduciendo gradualmente herramientas que permiten atacar un primer nivel de manejo de la tortuga, luego el manejo de actores, siguiendo con la incorporaciòn de elementos adicionales (sonido, tiempo, etc.). Uno de los puntos de motivaciòn màs importantes para el uso de la tortuga es la ràpida visualizaciòn del resultado obtenido al correr los procedimientos desarrollados.-

Se ve claramente entonces que es posible comenzar con un subconjunto de posibilidades reducidas para iniciar los primeros pasos y que ese subconjunto es fàcilmente utilizable.-

Respecto a la construcciòn de los algoritmos es posible lograr una estructuraciòn màs o menos rudimentaria utilizando bloques del tipo IF THEN ELSE y DO WHILE.-

Quizàs en el tratamiento de la recursividad se encuentren las mayores bondades del lenguaje porque permite el llamado reiterado a un procedimiento con paso de parâmetros, evitando cualquier complicaciòn de implementaciòn para el usuario.-

Hasta aqui hemos marcado cuatro aspectos que representan las mayores cualidades del lenguaje, pero no debemos olvidar que existen algunos usos desaconsejados para resolver con LOGO:

- No permite manejar operaciones complejas con datos compuestos.
- No tiene un buen manejo de archivos.
- No facilita las operaciones de E/S masiva de datos.
- Los mensajes de error son demasiado genèricos y dificultan la autocorrecciòn del principiante.
- No permite una buena autodocumentaciòn porque no respeta la indentaciòn y los comentarios "en linea" estàn restringidos.
- La falta de indentaciòn provoca que en el còdigo, no sean legibles las diferencias de nivel por anidamientos sucesivos.

Por otra parte las implementaciones de LOGO son bastante dependientes de la màquina que lo soporta (sobre todo en el manejo gràfico de actores), por lo que es difícil tener un "standard" suficientemente general.-

El punto crítico en la decisiòn de utilizar el LOGO como lenguaje introductorio es el gran cambio al que debe someterse el alumno para pasar a un lenguaje de mayor potencialidad.-

Ya que obviamente el LOGO puede ser utilizado sòlo para que los alumnos hagan sus primeras armas y no de una manera profesional, es necesario crear la transiciòn y dada la estructura del LOGO (derivado del LISP), es sumamente difícil "enganchar" con cursos basados en lenguajes procedurales tipo PASCAL.-

Resumiendo, sin negar las cualidades didàcticas, pensamos que esmàs útil volcarse a una herramienta tipo màquina abstracta pero con màs puntos de conecciòn con un lenguaje "a la PASCAL".-

Robot/Seudocòdigo:

La alternativa natural en esta instancia es utilizar una màquina abstracta tipo ROBOT, con un lenguaje de expresiòn de algoritmos màs potente basado en un seudocòdigo o en un lenguaje de especificaciòn tipo PDL (Ref. 14).-

Las ventajas màs significativas de esta soluciòn es que, al no tener un compilador al que ajustar la sintaxis, puede hacerse un tratamiento muy general de las estructuras de control y tambièn de las de datos, haciendo coincidir la especificaciòn con la codificaciòn.-

Sin embargo estas mismas ventajas constituyeron dificultades recogidas de las experiencias en 1986 y 1987 en Programaciòn:

** Un curso de 600 alumnos tiene unos 40 docentes auxiliares. El grado de libertad que da la utilizaciòn de un seudocòdigo sin una sintaxis precisa que se valide a través de un compilador produjo numerosas "expresiones alternativas" generadas por los diferentes docentes.
Un caso especialmente conflictivo fue el tratamiento de problemas con tipos de datos como pilas, colas y listas.-

** El seudocòdigo permite la creaciòn un tanto libre de "tipos standard", para los que valen todas las operaciones pre-existentes, sin restricciones naturales. Esto nos aleja del concepto deseado de objeto, con un TIPO definido por el usuario y un conjunto de operaciones validas y condiciones verificables. Sin ser completa, el empleo del concepto de TIPO y el encapsulamiento que pueden dar las unidades en las versiones màs avanzadas del PASCAL, llevan al alumno a una mejor abstracciòn de datos.-

** La expresiòn algorítmica en seudocòdigo no ejecutable sobre una màquina real mantiene el "gap" que va desde el anàlisis a la puesta a punto de un programa y no colabora con el desarrollo de una idea unificada de lo que es el ciclo de vida del software. Màs aùn: se pierde el fuerte efecto motivador del trabajo interactivo, que incluye etapas muy interesantes como el manejo de un editor orientado a la sintaxis del lenguaje; la detecciòn de errores sintàcticos y semànticos e incluso la re-adaptaciòn de un mòdulo previamente declarado "correcto".-

Calculadora/ Calculadora Programable:

Un modelo muy interesante por su simplicidad y su inmediata comprensiòn por el alumno es el de "Calculadora" que gradualmente incrementa su capacidad desde las 4 operaciones y entrada/salida a la programaciòn de estructuras de control.-

Sin embargo, fue descartado por:

** Existe una contradicciòn entre el modelo propuesto, que incorpore estructuras de control "a la PASCAL" y las calculadoras programables realmente existentes, que en general incorporan primitivas "a la BASIC".-

** Es muy difícil salir de los ejemplos de cálculo matemàtico y especialmente numèrico, encontrándonos con la barrera de los conocimientos previos de los alumnos.-

** Los tipos de datos "no numèricos", y màs aùn los estructurados resultan poco naturales en el marco de las aplicaciones de una calculadora. Es así que superada la etapa de los datos primitivos, resulta muy difícil encontrar los ejemplos "de motivaciòn" que justifique la creaciòn o incorporaciòn de nuevos tipos y màs aùn de nuevas operaciones y verificaciones.-

Este anàlisis de alternativas nos llevò a la conclusiòn de que el camino correcto era profundizar las ideas del Robot + Seudocòdigo, acercando dicho seudocòdigo lo màs posible a PASCAL (si era necesario modificando aspectos del PASCAL a utilizar por los alumnos) y además llegamos a la convicciòn de que el proceso de generaciòn y ejecuciòn real de programas por los alumnos sobre màquinas convencionales (en lo posible interactivas) incrementaría su motivaciòn y por ende mejoraría los resultados generales del curso.-

Así nace la idea de especificar LUBO-1, un robot con dos niveles de instrucciones: las de movimiento y tratamiento de objetos del mundo real y las de procesamiento convencional "a la PASCAL".-

UN BREVE ANALISIS DE LUBO-1:

Basado en las ideas de varios docentes del Dto. de Informàtica, se dearrollò una especificaciòn de màquina abstracta, que hemos llamado LUBO-1.-

Conceptualmente, el alumno dispone de una calculadora + movimientos + comunicaciòn con el mundo que rodea al robot.-
Se dispone de un conjunto de instrucciones que permiten cinco capacidades bàsicas en una primer fase:

* moverse.
* orientarse.
* reconocer objetos, recogerlos, depositarlos.
* realizar càlculos elementales.
* leer e informar.

El alumno resolverà problemas:

- usando instrucciones simples.
- construyendo nuevas instrucciones.
- manejando señales de control.
- ordenando càlculos.

LUBO-1 se mueve en un contexto definido como una ciudad de 100 calles horizontales y 100 avenidas verticales.
Cada paso del LUBO-1 equivale a una cuadra.
Establecido en un punto de la ciudad LUBO-1 puede ejecutar sobre el tipo de objetos que reconoce funciones tales como recogerlos, depositarlos y contarlos.
LUBO-1 tambièn cuenta con un reloj que sirve como elemento de control temporal para las acciones.

Por lo tanto, existen varios niveles de problemas:
- recorridos.
- recorridos con obstàculos.
- càlculos simples.
- càlculos con informes.
- recorridos con càlculos e informes.

A su vez se propicia la tècnica de descomposiciòn del problema en problemas màs simples mediante el fàcil manejo de mòdulos bien encapsulables.
Si bien la definiciòn primitiva del lenguaje se ciñe a un conjunto reducido de instrucciones para lograr un uso adecuado de las primeras herramientas, se agregan gradualmente nuevas capacidades que se incorporan al conjunto de definiciòn, permitiendo aumentar el alcance de los problemas a resolver.-

En esta primer fase, constituida por el curso de Ingreso y las primeras 4 semanas de Programaciòn de Computadoras, el alumno aprende a utilizar las estructuras de control (decisiòn, selecciòn, repetitivas e iterativas) y simultàneamente desarrolla una tècnica de descomposiciòn funcional de los problemas planteados.-
En todo momento se respeta una sintaxis PASCAL, incluso con la estructura de Programa con su secciòn de encabezamiento la de declaraciones y la de implementaciòn de los algoritmos.-

Una característica importante de esta primer fase es el manejo de dos tipos de datos: numéricos enteros y booleanos, conjuntamente con "objetos" como claveles y caramelos que LUBO-1 puede transportar en su "bolsa" (almacenamiento auxiliar) y sobre los que hay un conjunto limitado y específico de primitivas y tests válidos.-

La ciudad constituye un interesante contexto pues obliga a desarrollar algoritmos de recorrido que tengan en cuenta "obstáculos" (nuevamente condiciones verificables mediante primitivas relacionadas con una variable boolean) y naturalmente introduce al alumno en los conceptos de corrección y de eficiencia.-

Nótese también que los modelos "de recorrido" pueden adquirir una complejidad igual o mayor que la obtenible con los actores del LOGO al introducirnos en juegos tales como los encierros entre obstáculos o los laberintos en los que se puede trabajar con el robot.-

También resulta natural la introducción de la recursividad, y sobre todo el concepto de "informe" que establece el sentido de las instrucciones de E/S que caracterizan a todos los lenguajes de programación. En este punto un "robot" que hace recorridos es más natural que lea,calcule e informe que un "lapiz que dibuja".-

Una vez superada la primer fase, la incorporación de más instrucciones y sobre todo nuevos tipos de datos estructurados va marcando una reducción en la necesidad del soporte de aprendizaje que constituye la máquina abstracta; sin embargo muchas explicaciones introductorias se hacen utilizando el modelo de LUBO-1:

** Por ejemplo la introducción al tipo de datos COLA, creando ejemplos con robots que tienen que acceder a un mismo recurso en un punto de la ciudad.-

** El planteo de los algoritmos clásicos de manejo de caracteres, en los que se convierte a LUBO-1 en una máquina de procesamiento de caracteres (y posteriormente de tratamiento de textos).-

** La asociación de los conceptos de posición en X y en Y (calle-avenida de la ciudad de LUBO-1) con los índices de arreglos y matrices al introducir estos tipos de datos.-

ALGUNOS RESULTADOS:

Si bien es prematuro dar conclusiones a la experiencia con LUBO-1 realizada en 1988, el criterio general es que los alumnos han respondido de un modo superior en cuanto a interés y motivación: el horizonte que tienen al trabajar en un lenguaje "profesional" ha colaborado para obtener una mayor participación en los primeros meses de su formación universitaria.-

Claramente los auxiliares docentes (por otra parte en general alumnos avanzados con muy buen dominio de PASCAL) han homogeneizado las respuestas sobre sintaxis y semántica de las instrucciones y esto también contribuye en la perfomance de los alumnos.-

El riesgo siempre latente es no caer en convertir un curso inicial de Programaciòn en otro "de lenguaje PASCAL"; por ello tambièn es importante trabajar con una màquina abstracta que brinde prestaciones adicionales, sobre las cuales se pueden plantear aspectos algoritmicos generales que excedan las prestaciones del lenguaje de referencia (por ejemplo modelizaciòn o simulaciòn de fenòmenos de tiempo real utilizando la variable "tiempo" de que dispone LUBO-1).-

CONCLUSIONES:

Se ha presentado una experiencia de definiciòn de màquina abstracta y lenguaje para un primer curso de Programaciòn, adecuado al contexto particulàr de la Licenciatura en Informàtica de la UNLP, que es la consecuencia natural de una serie de experiencias realizadas en el mismo àmbito en los ùltimos años.-

Creemos que el tema del enfoque y las herramientas para el aprendizaje por los alumnos en un curso de este nivel sigue siendo un tema abierto de discusiòn en nuestro paìs y en el mundo, con lo que este trabajo no intenta màs que hacer un pequeño aporte al tema.-

BIBLIOGRAFIA:

1 - SECYT.
"Formaciòn de recursos humanos en Informàtica"
Documento SID num. 8, 1985.-

2 - ACM recommended curricula for Computer Science and Information Processing programas in colleges and universities. 1968-1981.
ACM Press, 1982.-

3 - J. B. ROGERS.
" Teaching beginners to program: some cognitive considerations"
Proc. The Computer: Extension of the human mind.
ERIC 1982.

4 - E. TORRERO.
"Education:The challenges are classic"
IEEE Spectrum, Nov. 1984.-

5 - DE GIUSTI, ROSSI, DIAZ.
"Educaciòn Informàtica: lenguajes de programaciòn y docentes"
Anales del Simposio Internacional sobre Informàtica y Educaciòn.
Tucumàn, Mayo 1984.-

6 - R. HOLT y otros.
"SP/K: A system for teaching computer programming"
Communication of the ACM. May 1977.-

7 - M. MULDER, J. DALPHIN.
"Computer science program requirements and accreditation".
COMPUTER IEEE. Vol 17, num. 4, Apr 1984.-

8 - S. PAPERT.
"Desafìo a la mente".
Ed. Galàpago, Bs. As. 1981.-

9 - DE GIUSTI A.
"El LOGO como lenguaje en educaciòn. Anàlisis crìtico".
Anales de la X Conferencia Latinoamericana de Informàtica.
Valparaiso (Chile) 1984.-

10- R. E. PATTIS.
"Karel The Robot. A gentle introduction to the art of programming with Pascal". Wiley 1981.

11- S. J. GARLAND.
"Introduction to computer science with applications in Pascal".
Addison Wesley, 1986.

12- Cátedra Programación de Computadoras 1988.
"Extensiones de Turbo Pascal para el manejo de estructuras de datos dinámicas"
Informe Técnico interno. La Plata, 1988.-

13- Programa del curso de Programación de Computadoras.
Departamento de Informática. UNLP. 1988.-

14- D. GRIES.
"The science of programming".
Springer Verlag, 1981.