C AS A FIRST PROGRAMMING LANGUAGE:
Its suitability at tertiary and secondary level

BY DOUGLAS NEWLANDS AND JOY TEAGUE
Department of Computing and Mathematics
Deakin University
Geelong

Following the Joint Report by the IEEE and ACM on Computing Curricula and the DEET Report of the Discipline Review of Computing Studies and Information Sciences Education, the structure of our syllabi for computer science and information systems has been reviewed and, in particular, our first programming language has been changed. Pascal has been the favoured first programming language in universities for many years and more recently it has also become the language of choice in many later year computing courses in secondary schools. However, universities are now moving away from Pascal and teaching languages like C, Scheme and Miranda as the first language. At Deakin University the review included: external recommendations for first level tertiary courses; the purpose and content of the tertiary first year courses; the languages which might be taught in first year. In 1992 C was taught rather than Pascal. This change has prompted a complete review of: the method of presenting a first programming course; the state of preparation of incoming students; how students coped with the new course after the first year. This paper addresses each of these issues. Some observations and recommendations are made for both school and university based educators.

INTRODUCTION
The first computer language widely used for teaching was BASIC which was developed at Dartmouth College. It functioned as a simple approach to FORTRAN but remained in a university, military and commercial environment until the advent of the affordable microcomputer. As there were, initially, few software packages available, most microcomputers arrived with BASIC, enabling users to develop their own software. Thus BASIC made its way into the home and school environments as computers became affordable. However, BASIC programs frequently are monolithic and difficult to read and modify, so the ideas of structured programming (Böhm & Jacopini 1966) were embraced by universities and Pascal, designed by Wirth (Jensen & Wirth 1975) to exemplify these ideas, was adopted as the new mainstream teaching language throughout the seventies. Judging by the numbers of first year tertiary computing students with Pascal skills, it is only within the last five years that Pascal has seriously challenged BASIC in secondary schools. What reasons do universities have for moving away from Pascal? Does this herald a further change in teaching computing in a school environment?
WHAT SHOULD BE IN A FIRST YEAR COURSE?
The characterisation of tertiary-level computer science has changed from the earlier, somewhat loose and intuitive descriptions (Newell, Perlis & Simon 1967, Abelson & Sussman 1985), which can be difficult for outsiders to grasp, to the much more prescriptive definition produced in the IEEE/ACM joint report (Denning, Comer, Gries, Mulder, Tucker, Turner & Young 1988) on the 'core of computer science.' This prescriptive approach has the advantage of allowing a measure of the quality of a teaching offering by comparing it with the three major paradigms (theory, abstraction and design) and the nine sub-areas identified in the report (algorithms and data structures; architecture; AI and robotics; database and information retrieval; human-computer communication; numerical and symbolic computation; operating systems; programming languages; software methodology and engineering). Ideally, tertiary first year computer science courses provide an introduction to all of these subject areas, and the subsequent years of the syllabus will elaborate the basic concepts and distinctions. It is, therefore, crucially important to get the first year offerings correct or the rest of the course structure will be insecure and disjointed. The report emphasises the importance of laboratories so that students can see concrete examples of the theoretical ideas they meet in lectures. It also notes the importance of the first programming language as it is this which provides access to the structures and distinctions of the discipline. Our department set up a complete syllabus review, designed a syllabus which would reflect the above recommend-ations and, in 1992, began teaching the new first level course. The subsequent Australian Federal Government review of computer science (DEET 1992), encouraging tertiary computer science departments to reassess their teaching in the light of current social and economic needs, is seen as further justification for the changes which have been implemented. This review places a heavy emphasis on computing being taught with a 'design' or 'engineering approach' (ibid., recommendation 9.2). It also notes the 'relatively low level of English language ability and interpersonal skills' which 'appears rooted in inadequacies in secondary schooling in language and human communication' (ibid., summary p. 33). Ideally, this should be rectified in the secondary school system but, failing this, first year tertiary courses need to improve students' communication skills.

PASCAL AS THE FIRST LANGUAGE
For many years, Pascal has been the most popular language to teach to first year computer science students. Dey and Land (Dey & Land 1992) found that 74% of universities were teaching Pascal, with

BASIC, FORTRAN, Ada, Modula-2 and combinations of languages all scoring under 10%. C was presumably in the 1% 'other'. In secondary schools, it appears that BASIC is still the most popular language. Surveying our 1992 first year students we found that 70% had some programming experience and of these 90% had experience of BASIC, 60% of Pascal, 50% of Logo, 17% of assembler and 23% had exposure to 4GLs. Internally, several reasons were advanced for changing from Pascal. The most important of these were:

- Although Pascal was used in semesters 1 and 2 for teaching, subsequent semesters used C and the two week introductory course in C in semester 4 was not working well.
- From our experiences with subsequent programming languages, it was apparent that the concepts of structured programming were not being identified as separate from Pascal and consequently the expected transfer of ability to write structured code in other languages was not eventuating.
- In the information systems stream, it had been observed that three successive semesters using COBOL produced a marked maturity in students' programming skills and we were anxious to achieve this in the computer science stream.

Increasingly, students entering the course had previous Pascal experience and were disdainful of repeating a language which they considered they knew, although their exam results did not necessarily confirm their knowledge.

As a corollary to the previous point, many students new to Pascal became dispirited as they observed others 'breathing through' the course when they, themselves, were struggling to understand the concepts.

THE CHOICE OF A NEW LANGUAGE
A number of alternative teaching languages were considered but only C and Pascal survived first consideration. Some of the reasons for the eventual decision to change to C are listed below:

- Market demand for computer science graduates shows a clear preference for C/Unix skills.
- There is some evidence (at last?) of a decline in the popularity of COBOL and a corresponding increase in demand for C skills.
- Students with computers at home want to learn C as C has replaced BASIC as the programming language for serious home programmers.
- It will remove some of the disadvantage felt by students who are present have no Pascal skills; very few first year students have previously learned C.
- Students are aware of the professional market place require-ments and surveys of student opinion (Newlands 1992) show a demand for C rather than Pascal.

As might be expected, arguments were raised against teaching C as the first language. Answers are provided below to the most persuasive of these:

- C is only a systems programming language. This commonly-held view was rebutted only by presenting the views of authors who describe C as general purpose, flexible, widely available and machine-independent, suitable for developing applications, suitable for teaching data structures and widely used, being the standard development language for personal computers. (Ammeraal 1988; Barclay 1989; Elishon 1987; Gottfried 1990; Hutchison & Just 1988; Kassab 1989; Kelley & Pohl 1984; Kernighan & Ritchie 1978; Tenenbaum, Langsam & Augenstein 1990).
C is not a 'safe' language. The usual example is array bounds checking but Pascal does not always catch this either.

The C compiler error messages are hard to interpret. This can be overcome with adequate expert support in teaching laboratories.

There is no good ANSI C compiler available. The GNU C compiler, available at no cost, is fully ANSI compatible.

C is too permissive and does not enforce any particular programming style. Although Pascal certainly supports Wirth's ideas on structured programming, these concepts are largely independent of language choice.

The somewhat strident attack on C by Mody (Mody 1991) was felt to raise issues which are beyond the scope of first year courses.

STUDENT BACKGROUNDS
While 70% of the 1992 students had some knowledge of a programming language, less than half (41%) had completed year 12 Computer Science. Of the students who had previous programming experience, most (72%) had had 300 hours or less (typically 100 to 200 hours) and 10% had between 300 and 1000 hours. Thus, only the remaining 18% of the students approached the 'thousands of hours' which are deemed necessary to acquire programming expertise (Kurland, Mawby & Cahir 1984). Of this group, only one student, a professional programmer, achieved a high distinction in the introductory computer science unit at the end of the first semester. 'Hackers' do not, necessarily, use good programming techniques nor understand well the fundamentals of computer science theory.

When teaching first year students, it quickly becomes apparent that most students, whether they have previous programming experience or not, have difficulties writing program structures. Most have only a very hazy understanding of the principles of programming and program development. This can be attributed both to the limited formal tuition which they have received and to their lack of practical experience. Providing a solid foundation for a computer science course is the challenging task facing the first year computer science lecturer.

THE NEW APPROACH
Having considered the arguments for and against C, it was felt there was a strong case for moving to C but that it should be tempered by teaching algorithm construction separately from the programming language. Unless the inability to construct solutions to problems can be remedied, teaching syntax and program coding is pointless. No amount of improperly functioning code, friendly user-interfaces, or clear and concise documentation can compensate for a faulty solution.

Structured programming concepts and algorithm construction are taught prior to the introduction of any computer language at the beginning of the semester. The Structured English used for the first five weeks of the thirteen week semester prepares the students for the introduction of any computer language, or even for shell scripts. In this period, students are introduced to control structures in the language with which they are most familiar (English). Even concepts like coupling and cohesion of modules can thus be discussed outside the framework of a computer language. The second five weeks of the semester are spent introducing C, always comparing and contrasting material with the earlier Structured English where appropriate. This approach allows a C program to be introduced as a collection of functions, some obtained from standard libraries and some user-written, resulting in a natural, holistic approach, rather than a piecemeal approach which combines the introduction of background concepts and the C language.

The use of the laboratory as outlined in the IEEE/ACM report was adopted to provide students with concrete examples of theoretical work from lectures. When students reach the laboratory, they have been introduced to the material in lectures, read it in the textbook, completed pencil and paper exercises on it in tutorials and finally they are ready to acquire the practical programming experience. Working in groups allows students to interact with others in their group and thus to avoid the temptation to resort to plagiarism to which struggling students, supposedly working individually, might otherwise resort. On a more positive note they have practical experience of dividing up work and defining interfaces between software blocks. Also laboratories remove the stress associated with individual assignments with a deadline. Assessment in the well-supervised laboratories is made towards the end of the laboratory session and the group are questioned closely enabling us to establish that all members of a group understand all aspects of their joint work.

OUTCOMES FROM THE NEW COURSE
The outcomes of the 1992 (C) and 1991 (Pascal) classes were compared in some detail (Newlands 1992/1993). The results showed an apparent decline in student performance at the end of the first semester. Some decline had been expected as there were environmental reasons for expecting a worse performance from the 1992 class. Previous year groups had always included some students with Pascal experience and their help was sought by other students; this was not the case with C. Similarly, previous year groups could approach senior students in the terminal room for help and again this was less readily available because the second year students did not yet have any C experience. Lastly, this was a new course being taught for the first time. Examination of the completion/withdrawal numbers showed they were very similar for both years and, although the examination results were slightly below the previous year, it
C as a First Programming Language

was not clear whether this was due to normal variation between classes or reflected a difficulty with C after only one semester of exposure to it. This result was slightly disappointing in that we might have expected the laboratory classes to improve the retention rates and to compensate for inexperience in teaching the new language, especially given the positive comments from the laboratory supervisors and from the students themselves. Nonetheless, the laboratory-based teaching was continued into the second semester data structures course which also is taught using C. At the end of that semester, the students were surveyed again and it was found that not only was the completion/withdrawal rate the same for both year groups but also that the examination performances of both groups were not significantly different. The first semester results for the 1992 class were 13% lower than the 1991 class but by the end of second semester they were indistinguishable.

As the course material is basic data structures and was presented identically in both years, except for the implementation language, the most obvious explanation for this improvement is that the benefit of the laboratory component of the course is off-setting the "teething problems" of teaching a new language for the first time. It also possible that it takes longer to become a proficient programmer in C than Pascal and that the improved second semester results reflect this. Clearly this study needs to be repeated in the future to examine what happens when the changeover itself is no longer affecting outcomes.

Conclusions

Although there can be quite pressing reasons for changing from Pascal in a tertiary teaching environment, C is only one of a number of suitable programming languages. From a secondary point of view, the arguments in favour of Pascal, or a similar well-structured language, rather than BASIC, continue to be very strong and the reasons why tertiary institutions might leave Pascal are essentially irrelevant to secondary teaching. Our first year students' most pressing problem is algorithm construction. Some observations might be made about the choice of language at secondary level as a preparation for tertiary studies:

- the use of non-procedural languages (e.g. dBaseIV), does not lead to problem solving skills (c.f. earlier remark that the first language should introduce the concepts and distinctions of the discipline);
- the use of unstructured languages like BASIC does not intrinsically help one to develop top-down planning techniques since the program structure does not reflect the planning structure nor does it allow detail hiding which is the most basic technique for handling complexity;
- since the students already have difficulty handling top-down planning, options like using FORTH, which is clearly a bottom-up language, provide no assistance at all.

On the mode of presentation, a vast improvement in the students' attitudes and work habits was obtained in the 1992 class as a result of the laboratory-oriented teaching, and through the strong links between the lectures, tutorials and laboratories. It is difficult to overstate the good feedback obtained from those students, senior students who observed the laboratory classes and the laboratory supervisors. Secondary teachers will not be at any disadvantage here because they almost always teach in a laboratory environment.

As regards the syllabus, tertiary departments can be expected to keep reasonably close to the recommendations in the Joint Report (Denning et al. 1988) but secondary teachers have a much wider choice within the syllabus, limited only by facilities and their expertise. However, secondary courses are not expected to produce computer professionals; they should be introducing students to the exciting field of computing and, hopefully, to as many aspects as possible. Depth of study and rigour are much less important at secondary level. The descriptions of work to be completed in the Common Assessment Tasks (Victorian Curriculum Assessment Board 1992) of the Victorian Information Technology subjects make depressing reading, with the emphasis on reports to be written rather than on exciting developmental work. While it is desirable for some planning documentation to be produced (since any worthwhile project is going to need some planning), it is our belief that the replacement of much of the report writing by developmental work will provide a more exciting and stimulating course for enthusiastic students.

References


Victorian Curriculum Assessment Board 1992, Revised Descriptions of the Common Assessment Tasks for 1993, VCAB.