NOVICE PROGRAMMING IN HIGH SCHOOLS: TEACHER PERCEPTIONS AND NEW DIRECTIONS

BY JOHN KING
James Cook University

JOHN FELTHAM
Ignatius Park College

DONNA NUCIFORA
Kirwan State High School

INTRODUCTION
Considerable interest has been generated of late as to the efficacy of teaching computer programming at the high school level. One recent comment in the Wiltshire Report (Cosgrove, 1994) in relation to the Years 11–12 Information Processing and Technology (IPT) subject in Queensland has gone so far as to suggest that the rate of development has made programming out of date for this level of student and that '...the emphasis might now move towards authoring systems and hypermedia rather than programming per se'. Given recent comments (Glynn, 1994) that the syllabus was designed by a forward looking informed committee and is '...still relevant twelve years after its beginnings', and given other recent concerns about the teaching of programming at this level, it was considered timely to review the research that has been undertaken in the area of novice programming and compare it to current teacher perceptions on the relevance of this topic in today's world.

While this paper specifically focuses on the perceptions of teachers of IPT to the teaching of algorithms and programming as one of five major topics in the subject, it should be of interest to teachers and researchers of novice programmers generally. Detailed background to the development of the IPT subject and recent assessment practices have been provided by Glynn (1994) and Dixon (1994).

METHODOLOGY
The approach selected to obtain teacher perceptions for the study was a descriptive exploratory survey. This was used to identify the different algorithmic languages selected in the algorithms and programming topic and to explore the issues involved in teaching these languages. A limited number of open-ended statements made it possible to further study the perceptions and opinions of the target population regarding the selection and use of programming languages.

According to Department of Education (Queensland) records, there were 126 schools (state and independent) which had, previous to 1994, offered the Board of Senior Secondary School Studies IPT subject at least once. A questionnaire was forwarded to these schools. Principals were informed that the survey form was only required to be completed by one current IPT teacher in the school. This was to avoid obtaining identical responses on some questions, as it was assumed that most schools would, in the main, adopt similar approaches and use similar resources for all classes taking the subject. Schools were also informed that the aims of the project were to investigate issues relating to the teaching of the algorithms and programming (A & P) topic within the Years 11–12 IPT subject. These issues included the current teacher thinking regarding the teaching of this topic, the range and type of resources being used, as well as the reasons for their use. Completed survey forms were returned by 71 of the schools. This represented a response rate of 56%. This paper concentrates on teacher perceptions of teaching the topic and a number of issues will be outlined in the following sections. These include such questions as:

- Does learning a programming language improve general problem-solving skills?
- Does training in programming provide valuable vocational opportunities for students?
- Does the algorithms and programming topic provide an important background for further studies in computer science?
- Should the study of IPT and hence algorithms and programming be restricted to students with a strong Maths/Science background?
VOCATIONAL OPPORTUNITIES

An often-quoted reason for teaching programming in schools is that programming is an important societal skill which will enhance student employment opportunities. However, an American study as far back as a decade ago (Goldstein & Fraser, 1985) suggested that ‘... relatively few workers need extensive education or training in computer-related skills most learn their skills in brief, on-the-job training’, with this ‘relatively few’ representing about two percent of the population. It is unlikely that this conclusion has changed over time, or differs significantly in the Australian context; given the trend to more user-friendly languages.

In a more recent Australian study of the information technology (IT) skill requirements of school leavers, Oliver and Newhouse (1993) investigated employer perceptions of the need for IT skills and compared these to the perceptions of students and teachers. While a large percentage of the jobs studied necessitated the use of computers, the skill level required tended to be primarily quite low, with menu-driven software use and single-key input being the order of the day. For in-house systems, on-the-job training was commonly available. While this study dealt with general IT skills, and with positions into which students could move di-
The topic provides valuable vocational opportunities

Figure 2 Teachers’ perceptions of the vocational importance of the algorithms and programming topic

Important background for further studies

Figure 3 Teachers’ perceptions regarding the usefulness of the AP topic as an important background for further studies in computer science

<table>
<thead>
<tr>
<th>Percentage Range</th>
<th>Response Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>20–22</td>
<td>4</td>
</tr>
<tr>
<td>23–25</td>
<td>15</td>
</tr>
<tr>
<td>26–28</td>
<td>15</td>
</tr>
<tr>
<td>29–31</td>
<td>29</td>
</tr>
<tr>
<td>32–35</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 1 Percentage of subject time allocated in teacher workshops to the algorithms and programming topic

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>HA or higher than average in Maths</td>
<td>26</td>
</tr>
<tr>
<td>Sound in English</td>
<td>26</td>
</tr>
<tr>
<td>No prerequisite</td>
<td>19</td>
</tr>
<tr>
<td>Sound in Maths</td>
<td>17</td>
</tr>
<tr>
<td>HA in English</td>
<td>15</td>
</tr>
<tr>
<td>HA in Science</td>
<td>4</td>
</tr>
<tr>
<td>Computer Studies (grades 8 to 10)</td>
<td>4</td>
</tr>
<tr>
<td>Other</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 2 Recommended prerequisites for IPT from grade 10

(HA = High Achievement)

in personal qualities such as good communication skills.

Figure 2 suggests that IPT teachers are currently ambivalent about the value of programming in relation to vocational opportunities. This is in spite of the emphasis in the subject on ‘... the processes of conceptualisation and formalisation, specification and design, rather than the production of code’ (Glynn, 1994); that is, an emphasis on general (problem-solving) skills rather than on specific or immediate vocational skills. This ambivalence may be further evidenced by the spread of time allocated by teachers in work programs for the algorithms and programming topic, from the minimum of 20%, through to 35% of the total time for the subject (Table 1).

It is clear that some teachers give significantly more time than others to this topic; but to what extent this relates to perceived vocational importance is not so obvious. It may well be, given the lack of need for programming skills of exiting high school students, that teachers perceptions of the usefulness of pre-tertiary programming are linked indirectly to vocational importance through their knowledge that students need tertiary programming experience to get employment as a programmer, and that their high school experience will help them in this regard.

BACKGROUND FOR FURTHER STUDIES

An interesting contradiction appears to exist between IPT teachers’ perceptions about programming as an important background for further studies in computer science compared to those of tertiary educators. In an interview with Professor Brian Harvey, Professor of Computer Science at the University of California, Berkeley (Harvey, 1992), it became clear that he was not concerned if students entered tertiary studies without computer experience. He was more concerned that students were ‘... feeling comfortable with the idea of formal notation, and thinking about manipulating symbols in a formal way and both being able to do it and liking to do it’. He was concerned that students enter computer science courses thinking that computer science is only about the study of programming languages. Newlands and Teague also saw that secondary school programming needed to be different from tertiary studies: ‘Depth of study and rigour are much less important at secondary level ... it is our belief that the replacement of much of the report writing by development work will provide a more exciting and stimulating course for enthusiastic students (p. 15).’

Figure 3 suggests that IPT teachers believe that the AP topic does have real advantages for their students who go on to tertiary studies. One explanation for this belief may relate to Pascal being the preferred language for this topic as well as for introductory computer science studies.

PREREQUISITES FOR NOVICE PROGRAMMING

A further contradiction appeared to exist between teachers perceptions regarding students’ need for a Maths/Science background (Figure 4) and the prerequisite recommendations of the schools for the IPT subject (Table 2). While teachers clearly saw little need for a strong Maths/Science background, the prerequisite recommendations of the schools suggested that a better than average past performance in Mathematics, and to a lesser extent English, was clearly desirable. This apparent difference in perception of the subject may have arisen as a result of teachers having interpreted the survey question in terms of equity of access; or alternatively, decision makers on subject prerequisites in schools may have a more elitist perception of the subject, a view which was not supported by the original syllabus committee.

The linkage by schools of the IPT subject to Mathematics may be one reason why there is a relative avoidance of the subject by girls. Participation rates of girls in IPT have oscillated
The importance of the 'right' language

The literature abounds with discussion of the advantages or otherwise of various programming languages for use with novice programmers. The discussion here will attempt to focus on some of the characteristics deemed important for teaching students at the pre-tertiary level. A comparison with reasons provided by teachers for the choice of the particular languages they use for the algorithms and programming topic in ITP will also be made.

Iffinger (1991) has argued for the use of Karel the Robot as an introductory language for ITP students because of the inherent pedagogical problems with the flexibility and power of Pascal. He argued that with Pascal, it is '...very easy to confuse students by introducing too many concepts in the early stages (p.11). He also argued that Pascal may place undue emphasis on the mathematical connotations associated with the topic. One wonders how difficult this is to avoid, given that computer laboratories are often associated, because of historical reasons, with Mathematics departments and because, as revealed by this survey, computing teachers in Queensland high schools are seven times more likely to teach Math/Science as their second teaching area than other subjects. Further to the link between programming and mathematical anxiety, Geya-May and Hazan-Seger (1993) observed that '...Logo seems to neutralise anxieties or inhibitions induced by previous lack of knowledge (i.e. in mathematics) or by limited exposure to the comparatively new technologies'.

Some authors have argued that the language is less of an issue than the approach used. For example, Marchionini (1985) expounded principles which he believed should apply for the teaching of any programming language. His 'developmental approach' required the need to stress concepts rather than vocabulary and syntax; provide motivational, relevant examples and activities; proceed from concrete to pictorial to abstract according to age and previous experience; and, use a sequence of increasingly complex activities which build on and extend previously learned concepts. More recently, Tolhurst (1993) has considered several other instructional strategies in the teaching of programming. Harvey (1992) also stressed the need for students at high school level who are interested in programming to undertake programming tasks which involve 'serious work' and be involved in solving 'real problems' in the same way that students involved in publishing school newspapers can.

How instructional practices influence what students learn in Pascal programming classes was further investigated by Husic, Linn and Sloane (1989). Programming proficiency was found to vary as a function of instructional practices and class level with introductory students performing better with direct instruction and advanced students benefiting from less guidance and more autonomy. Results of another study by Oliver and Malone (1993) also indicated that the instructional format can have a significant influence on the students' understanding of programming constructs and algorithms. A significant advantage was achieved in those classrooms where '...students were exposed to instructional formats involving descriptions of program semantics combined with practice with tasks of a conceptual nature'.

A more recent version of BASIC, namely Visual BASIC has been described by Nielsen (1994) as '...a terrific bonus for teachers', and as a language which allows students to create stimulating and dynamic applications for a wide range of audiences' (p.27). These characteristics arise from the inclusion of graphical design tools and a simplified high-level language, with an emphasis on '...feedback and debugging tools which quickly take you from an idea to a solution'. This description seems to fit well with the requirements of Marchionini (1985) and Harvey (1992) described above, in that the language can be used by students to undertake real tasks and obtain system-building solutions without undue time and energy required for coding and debugging. The description and use of the program provided by Nielsen also appears to go some way to support the comments of Cosgrove (1994) in the 'Wiltshire Report', by placing an emphasis on multimedia programming and reduced programming effort. While other languages such as HyperTalk also allow relatively easy incorporation of multimedia features, Nielsen claims that Visual BASIC accomplishes this task with far less energy.

Figure 5 indicates that teachers in the survey were rather equivocal about the notion of incorporating sound and graphics in programming activities.

While only one teacher indicated the use of Visual BASIC, and five teachers indicated the use of HyperTalk as their main teaching language, the recent advent of the Windows environment which Visual BASIC requires, and the growing acceptance of the Macintosh graphical user interface in Queensland high schools may accelerate the trend to a greater use of multimedia in programming. One also suspects that some primary schools are leading the way here at a less complex level, as they did in the introduction of Logo and LogoWriter.

Another new language direction which a few teachers in the survey
suggested they may move towards the programming language C. Newlands and Teague (1993) outlined some reasons which influenced their decision to change to C at the first year tertiary level. Some of these, such as a clear preference for C programmers in the market, and that C has replaced BASIC as the programming language for serious home programmers, may well influence secondary teachers to consider C as a replacement for Pascal. However, Newlands and Teague caution that "... 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'.

Given the diversity of programming languages and the diversity of opinion as to what programming language best suits novice programmers, it was somewhat surprising to find that teachers contributing to the survey displayed amazing conformity as to the main programming language used in the algorithms and programming topic (Table 3).

A number of reasons were provided for using Pascal as the main language in this topic. The more frequently mentioned of these in order of popularity were that:

- It teaches structured programming and is a good teaching language.
- I am experienced in using it and/or the language was taught to me.
- It is well accepted at the tertiary level and by colleagues.
- It was already in use at the school, the school has a site licence, and other cost reasons.
- It is easy to use.
- Useful accompanying resources are available.
- It maintains similarity to professional languages such as C and 4GLs.
- It contains everything to do the job.
- It is consistent with the syllabus objectives.

Apart from historical factors, tertiary institutions continue to have a strong influence in the justification of computer language selection in high schools. Two of the three most popular reasons listed above relate to this influence in different ways. The first relates to what tertiary institutions teach first-year introductory computer science students. This tends to determine what high school teachers teach because of their perception that students will find life easier when they go on to further studies. The second influence relates to what the teachers themselves were taught in their preservice or in-service training (most commonly a Graduate Diploma of Computer Education). Teaching what they were taught has obvious benefits in terms of preparation load. Together, these influences are making it difficult for alternative approaches to gain a foothold.

As the researchers were aware that some teachers used an introductory language, that is, a second language as an introduction to the main language, it was considered useful to ascertain the extent of this approach, the type of introductory language selected for use, if any, and the reasons for using, or not using, more than one language. Some of this information is set out in Table 4.

The responses here were somewhat more varied, with a significant minority using no introductory language. The presence of TurboPascal on this list may have come about through some respondents' confusion as to what was meant by an introductory language in the survey. Karel the Robot was preferred mainly for its ease of use, its motivational/functional characteristics and its easy introduction to modular programming among other reasons. No particular reason stood out with RoboPascal, but one might assume that, as it was written as an introduction to Pascal via robot programming (McGilvray, 1993), its justification goes without saying. Logo was favoured for its visual/graphic interface, its instant feedback, its simple starting characteristics and easy introduction to modular programming.

Some unsolicited reasons proffered for not using an introductory language for this type of student included:

- Too much wasted time.
- Emphasis should be on algorithms, not programming.
- Students prefer to get on with the real stuff.
- Learning one language well carries over to other environments.
- There's not enough time to learn two or three environments.

It appears from these comments and from Table 4 that teacher decision making in relation to using an introductory language is influenced by the tension between the practical consideration of time availability, which may vary from a minimum of 44 hours to a maximum of 77 hours, and the pedagogical advantage of providing students with a more gradual introduction to the more abstract concepts of the main language taught.

Teachers feelings as to the importance of suitable language selection are displayed as Figure 6. While a significant majority indicated that language selection is important, a similar number agreed that teaching algorithm construction was more important than programming (Figure 7).

The survey also confirmed that considerable diversity existed in the algorithm design methodologies applied in teaching the algorithms and programming topic. This is indicated in Table 4.

Structured Design Charts (SDCs) were preferred mainly because of similarities to DELTA, a commonly used computer program developed for the subject in Queensland which enables the construction of algorithms which can be automatically translated into several computer languages (Gordon, 1993). However, SDCs were also selected for reasons such as: ease of use, mentioned in the text, good for top down design — among others. Pseudocode was seen by its supporters to transfer easily to code, to be easy to use and to be closest to the natural
language. Experience in use also featured in most selections from Table 4. A few teachers suggested that there was no best methodology and preferred a variety of approaches.

Concluding remarks
This study has highlighted the variation in teacher perceptions about a number of issues in the teaching of novice programmers at the pre-tertiary level. Further, a contradiction has been shown to exist between teacher perceptions of background required for the subject and school perceptions in terms of prerequisites recommended. Some evidence from the literature review that secondary and tertiary educators differ in their perceptions as to the importance of pre-tertiary programming experiences is also evident.

The survey has also exposed the high level of conformity in the use of Pascal as the main programming language in the algorithms and programming topic. Tertiary institutions continue to exert a strong influence, both in terms of the languages teachers learn, and in terms of the future languages to which exiting high school students will be exposed when they enter tertiary institutions. It is fitting here to quote Newlands and Teague, (1993) regarding the high school/tertiary interface when they say that '...tertiary departments can be expected to keep reasonably close to 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'. It is argued that the introduction of the Windows environment and the greater acceptance of the Macintosh graphical user interface in the upper high school may help to break this cycle of dependence and encourage teachers to move towards multimedia programming applications. These may allow students to work on real problems related to their own learning with a greater degree of productivity and with less effort.

New languages under development may provide other choices for high-school teachers. For example, Boxer, the new computational medium which grew out of Logo (DiSessa, Abelson & Ploeger, 1991; Roschelle, 1993), has been designed within an easily learned framework so that non-computer specialists can use a wide range of functions from hypertext processing to interactive graphics, to databases and programming. Alternately, the focus may move further towards DELTA-type tools which let students concentrate on the problem-solving aspects of algorithm construction, while allowing the machine to generate the appropriate code.

It will be interesting to see to what extent these alternative directions, together with the more recent emphasis on multimedia and hypermedia, will de-emphasise programming or transform it into a cross-curriculum activity in high schools, rather than see it remain as a 'specialised pre-tertiary activity which is currently conceptualised by some to be.

Acknowledgement
The authors wish to acknowledge the support provided for this work by the joint research program funded by the School of Education, James Cook University, Townsville and the Northern Regional Office of the Department of Education, Queensland.

REFERENCES


