

Some experience in teaching an introductory programming course using Oberon

V. Mahnič

Faculty of Computer and Information Science, University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia

Abstract

In the academic year 1995/96 the content of the introductory programming course at the Faculty of Computer and Information Science of the University of Ljubljana was restructured in a fundamental way. The new content combines two programming paradigms: the procedural paradigm, and the object-oriented paradigm. Oberon was chosen as a teaching language because it makes possible the introduction of object-oriented concepts simply as extensions of well-known concepts of procedural programming. In this paper we describe the experience gained from teaching the course for the first time in 1995/96. The students were surveyed by a questionnaire at the end of the course, and the results are presented describing their evaluation of the modified course content, the choice of the teaching language, and the approach to object-oriented programming.

1 Introduction

All Computer Science students at the University of Ljubljana take the introductory programming course, Programming I, in their first semester. The course lasts 15 weeks, and comprises 45 hours of lectures and 45 hours of laboratory practice. Until 1995 the teaching language was Pascal, while the content of the course included the basic elements of Pascal (basic data types, expressions, statements), procedures, composite types, recursion, and dynamic data types.

Experience of the last several years has shown that most entering students had already been exposed to Pascal in secondary school and (in spite of the fact that they had not mastered it completely) they found Pascal as a teaching language redundant. On the other hand, it was noticed that students lacked the knowledge of the principles of “proper programming” characterized by readability, clear structure, modularity, etc.

So, in 1995/96 we restructured the content of the course in a fundamental way. The increased knowledge of entering students enabled us to reduce the amount of time devoted to basic programming constructs, and to pay more attention to program structure, program design principles (particularly, stepwise refinement), and the advantages of modular programming (hiding of details, abstract data structures, and abstract data types). Additionally, the introduction to object-oriented programming (OOP) became an important part of the course.

Oberon (Wirth [1], Reiser & Wirth [2]) was chosen as a teaching language because it has all the properties that are necessary for the teaching of good programming style (i.e. simple and precise syntax, strong typing, modular program structure supporting data abstraction), and because of its specific approach to OOP which enables a smooth transition from traditional procedural programming to OOP. The modified content of the course and the reasons for choosing Oberon were described in detail in Mahnič & Vilfan [3].

The modified course was taught for the first time in the academic year 1995/96. Students practised programming using different PC based implementations of Oberon developed at ETH in Zurich. The DOS version of Oberon System 3 V2.0 was installed in student laboratory because of hardware limitations (diskless IBM PC 80386 clones connected in a Novell network) while more sophisticated Windows implementations were used at their homes (e.g. Oberon System 3 for Windows or Oberon V4). In all these implementations the Oberon language is implemented as part of the Oberon system (Reiser [4]), i.e. the environment in which Oberon modules can be edited, compiled, debugged, and executed. Compared to Turbo Pascal which we used before, this environment proved to be new and sometimes unusual, it offered less convenient debugging facilities, and it was (particularly the DOS version) sometimes unstable.

At the end of the course the students were surveyed by a questionnaire with the aim of verifying whether our decision to modify the course content was right and whether the aims of modification were achieved. The sample size was 78, representing 48% of the students enrolled in the first year of Computer Science in 1995/96 (163). However, given that there are a lot of unsuccessful students who dropped out during the first year (only 92 students passed the exam, and only 73 enrolled for the second year), we estimate that the sample comprised those students who regularly attended lectures and fulfilled the course requirements.

The purpose of our paper is to discuss the results of the survey, and to describe the conclusions that will help us to improve the quality of study in forthcoming years. The answers to the questions will be presented in 4 sections: assessment of the previous knowledge (section 2), a general evaluation of the course content (section 3), evaluation of the Oberon programming language (section 4), and students' opinions about OOP (section 5).

Through questions and answers some dilemmas frequently encountered in the design of introductory programming courses will be described, e.g. the choice and the arrangement of the content, the choice of the teaching language, the transition from procedural programming to OOP etc.



2 Assessment of the previous knowledge

The aim of the first two questions was to verify our assumption that an increased knowledge of programming enables the restructuring of the course content.

Question 1: *Which programming language(s) did you learn before enrolment? If you have learnt more than one language, list all of them.*

Pascal	74	Paradox (PAL)	10
Basic	30	dBaseIV (SQL)	6
C or C++	26	Delphi	3
Assembler	14	Logo	1
Prolog	13	none	2

Question 2: *Estimate your knowledge of Pascal using grades from 1 (insufficient) to 5 (excellent):*

The answers are presented as average grades for each domain separately:

a) simple programs with repetitive and conditional statements	4.38
b) procedures, parameters, local and global variables	3.85
c) recursion	2.86
d) arrays	3.59
e) records	3.45
f) files	3.10
g) sets	2.45
h) pointers and dynamic data structures	2.58
i) object-oriented programming in Turbo Pascal	1.83

Comment

The answers confirm the assumption that the great majority of students had used Pascal before their enrolment at the university, while some of them were also exposed to other programming languages. There were only 4 students that had not used Pascal before: two of them used Basic, while the other two had no programming experience at all. Surprisingly, quite a lot of students reported experience with other programming languages. However, this does not mean that they had mastered them completely, but they acquired some knowledge during special summer courses which are organized in Slovenia to popularize computing in secondary schools.

Although almost all students had used Pascal before attending our course, they admit that their knowledge of Pascal is insufficient in the domains of recursion, sets, pointers, and dynamic data structures. Therefore, these topics deserve more attention in the introductory programming course. During examinations it was also found that many students had not mastered the concepts of parameter passing, although they graded their previous knowledge of this domain as satisfactory (3.85).

3 General evaluation of course content

In the design of the introductory programming course we decided to combine two programming paradigms: the procedural paradigm, and the object-oriented paradigm. Oberon was chosen as a teaching language to enable a smooth transition from procedural programming to object-oriented programming. In accordance with this decision the course consists of three parts (Mahnič & Vilfan [3]): the first is devoted to the basics of the Oberon language and the Oberon operating system. On the assumption that students are already familiar with Pascal it lasts only 3 weeks. The second part, lasting 6 weeks, deals with procedural programming, while the third part, also lasting 6 weeks, covers object-oriented programming.

The purpose of this group of questions was to discover students' opinions about course content with an emphasis on the level of difficulty, the usefulness of the course, and the quality of lectures and laboratory practice.

Question 3: *How difficult is the course?*

- | | |
|--|----|
| a) the course is too easy | 3 |
| b) the level of difficulty is just right | 60 |
| c) the course is too difficult | 14 |

Question 4: *How have you found the usefulness of the course?*

- | | |
|--|----|
| a) the course is useful and interesting | 24 |
| b) the course is useful | 36 |
| c) the course is not useful | 7 |
| d) the course not useful and uninteresting | 10 |

Question 5: *How have you found the quality of lectures and laboratory practice (compared to other courses)?*

- | | |
|---------------------------|----|
| a) worse | 12 |
| b) approximately the same | 51 |
| c) better | 14 |

Question 6: *Which topics deserve more attention? (the students could choose up to 3 answers)*

- | | |
|--|----|
| a) basic programming constructs | 6 |
| b) program statements (e.g. assignment, WHILE, IF ...) | 7 |
| c) program syntax | 10 |
| d) the Oberon operating system (description of programming environment) | 2 |
| e) procedures, parameters, local and global variables | 11 |
| f) examples of simple programs suitable for beginners | 27 |
| g) examples of complex programs illustrating the modular structure and program design principles | 27 |
| h) sets | 5 |
| i) arrays | 1 |
| j) records | 2 |

k) pointers and dynamic data structures	25
l) object-oriented programming	41
m) recursion	27
n) files	32

Comment

We were pleasantly surprised by the answers to question 3, indicating that we had chosen the right level of difficulty. Given that the previous knowledge of new students differs widely we did not expect such uniformity of responses.

The answers to question 4 are also very encouraging. Most students feel that the course is useful or useful and interesting. The course was found not to be useful (and uninteresting) by students who did not agree with the use of Oberon as the teaching language.

The responses to question 5 show that lectures and laboratory practice are of approximately the same quality as lectures in the other courses. The score is not bad, considering that the restructured course was running for the first time, and that the detailed structure and content of particular lectures was often defined just a day or two before the lecture took place. For this reason, there was not enough time occasionally to prepare aids (overhead slides, more sample programs) that would have given the lectures professional polish. However, we are striving for a better quality of lectures in the forthcoming year. Using last year's experience we have defined the detailed content of particular lectures, prepared visual aids, additional practical examples, and the corresponding course-books in Slovene language.

Responses to the question 6 can be summarized as follows:

- OOP is a promising new programming technology that attracts students. Therefore, it is not surprising that most of them want to devote more attention to this topic.
- The relatively high number of responses requesting more detailed treatment of file structures and recursion is a consequence of the fact that only a brief discussion of these two topics was given, like in Reiser & Wirth [2].
- There is a relatively strong demand for additional practical examples. This is quite natural, since practical examples increase the comprehensibility of the material covered. The number of students who want more simple examples appropriate to beginners is equal to the number of students who want more complex programs illustrating the principles of program design, modularity, and data abstraction. This duality reflects the different needs of particular students as a consequence of differences in their previous knowledge.
- Among other topics, pointers and dynamic data structures stand out. This can be explained partly by insufficient previous knowledge (see question 2), and partly by the fact that the knowledge of pointers



and dynamic data structures is a prerequisite for the understanding of OOP in Oberon.

Although we agree in principle with students' proposals, it seems impossible to take into account all of them because of time constraints, which allow only 45 hours for lectures and 45 hours for laboratory practice. In the forthcoming year we are planning to increase slightly the portion of the course devoted to OOP, and, first of all, give more practical examples.

4 Evaluation of the Oberon programming language

Regarding the choice of the teaching language, we intended the course to be more interesting by replacing Pascal with a new programming language. Our choice was Oberon because, besides supporting a smooth transition from procedural programming to OOP, it satisfied the software quality criterion (Burgess & Jones [5]): the programming language should support the elements of good programming style that we wish to impart to students. In that case they are more likely to preserve that style in other languages that may be more prevalent, but less appropriate for an introductory course.

Some students opposed Oberon, arguing that a widely used language would have been a better choice. They also complained of the Oberon environment. Thus, the purpose of the third group of questions was to obtain students' opinions about factors affecting the choice of programming language, as well as their evaluation of Oberon as a teaching language for the introductory programming course.

Question 7: *Is Pascal appropriate for the introductory programming course or (assuming that you learnt Pascal in the secondary school) would it be better to teach another programming language?*

- | | |
|--|----|
| a) I propose the use of Pascal | 36 |
| b) it is necessary to teach another language | 41 |
| c) undecided | 1 |

Question 8: *Which factor should prevail when choosing the teaching language for an introductory programming course?*

- | | |
|---|----|
| a) an emphasis on the principles of "proper programming"
(readability, clear structure, modularity etc.) | 42 |
| b) commercial use in practice | 36 |

Question 9: *Which programming language would you choose for teaching the introductory programming course?*

- | | | | |
|-------------|----|--------------|----|
| a) Pascal | 29 | d) Delphi | 12 |
| b) Oberon | 17 | e) Smalltalk | 1 |
| c) C or C++ | 13 | f) undecided | 13 |

Question 10: *Is Oberon suitable as the teaching language for the introductory programming course?*



a) yes	44
b) no	33
c) undecided	1

Question 11: *Which feature(s) of Oberon do you like most? (you can choose more than one response)*

a) the environment (the Oberon operating system)	11
b) simple and precise syntax	29
c) modular design of programs	43
d) strong typing	12
e) definition of structured data types using simpler data types as components of more complex structures	25
f) use of pointers and dynamic data structures	25
g) object-oriented programming	38
h) writing of clear and understandable programs	31

Question 12: *Which are the most troublesome features of Oberon? (you can choose more than one response)*

a) environment (the Oberon operating system)	59
b) insufficient commercial use	57
c) (too) simple syntax	1
d) modular design of programs	9
e) strong typing	19
f) improper approach to OOP	16

Comment

Question 7 was intended to verify the assumption that most students see using Pascal as a teaching language as redundant. Results confirm that most students really want to learn another language, but the majority are not as explicit as we expected based on the experience of previous years.

The situation is similar with regard to question 8. Most students think that the emphasis on the principles of "proper programming" must prevail over the criterion of commercial success when choosing the teaching language for an introductory course. However, the majority are less explicit as we would like to be.

We are somewhat surprised by the responses to question 9. Despite the impression from previous years that the majority of students are not satisfied with Pascal as a teaching language, Pascal received most votes. Oberon was in second place, followed by C and Delphi. A relatively large part of students remained undecided, thus supporting the opinion that the choice of the programming language is a problem that must be resolved by experts and not by students. If we add up the number of votes for Pascal and Oberon on one side, and the number of votes for C and Delphi on the other, we get (compared to question 8) a much better ratio between languages that primarily support good programming style and languages that are admittedly widespread in industry.

Responses to question 10 show that most students share the opinion that Oberon is suitable as a teaching language for the introductory programming course. The number of responses in favour of Oberon would be

much higher if the Oberon system were more reliable and elaborated. That is, a detailed comparison of responses to questions 9 and 10 shows that 15 students who characterized Oberon as unsuitable chose Pascal as the most suitable teaching language, arguing that it is more reliable and elaborated than Oberon. This is surely one of the reasons why Pascal got so many votes in question 9.

Responses to question 11 show that the students know the essential advantages of Oberon: the modular design of programs, the interesting approach to OOP, the possibility of writing clear and understandable programs, and simple and precise syntax. Responses to question 12 clearly revealed Oberon's main deficiencies: the unreliable and under-elaborated environment, and insufficient commercial use.

Surprisingly, quite a lot of students think that strong typing is a negative feature, although strong typing is one of the foundations of proper programming. This opinion can be explained by a general objection to all restrictions, but it also reflects the fact that some students have not realized the importance of strong typing for writing clear and reliable programs.

5 Students' opinions about OOP

New topics tend to appear first in advanced courses and then to progress downwards in the curriculum until they find place in introductory courses. Object-oriented programming is no exception: many institutions are planning, or have already introduced, object techniques into first-year courses. However, there is heated debate on how to achieve the most efficient integration of OOP concepts into course (Parsons [6]). Our approach (Mahnič & Vilfan [3]) assumes that procedural programming is the foundation upon which we develop concepts of object-oriented programming. Using Oberon all object-oriented techniques and concepts can be explained in terms of concepts already known from procedural programming.

The last group of questions was intended to obtain students' opinions about the decision to include OOP in the introductory programming course, about the portion of content devoted to OOP, and about the approach to OOP used by Oberon.

Question 13: *Is it appropriate to teach the basics of OOP as part of the introductory programming course or would it be better to treat the OOP later, as part of a higher level course?*

- | | |
|--|----|
| a) OOP must be a part of the introductory course | 48 |
| b) OOP must be a part of subsequent courses | 29 |

Question 14: *The portion of content devoted to OOP this year was:*

- | | |
|---------------|----|
| a) too small | 25 |
| b) just right | 35 |
| c) too large | 18 |

Question 15: *Would it be possible to increase the number of hours devoted to OOP without affecting other topics that are essential for a complete knowledge of programming in Oberon?*

- | | |
|--------|----|
| a) yes | 42 |
| b) no | 36 |

Question 16: *Using Oberon, the object-oriented features are introduced as simple extensions of concepts already known from procedural programming, while some other languages (e.g. Smalltalk) require a more radical approach that often neglects existing knowledge of procedural programming. Do you think that Oberon's approach is better?*

- | | |
|------------------|----|
| a) yes | 55 |
| b) no | 17 |
| c) I do not know | 6 |

Comment

Responses to questions 13, 14, and 15 show that the majority of students support our decision to include the basics of OOP in the first programming course. Most students also agree that the portion of content devoted to OOP was adequate, although (according to responses to questions 14 and 15) it could also be slightly increased. Thus, in the forthcoming year more attention will be devoted to the implementation of objects using messages and handlers.

Question 16 was intended to evaluate Oberon's approach to OOP. Although the responses to this question must be treated cautiously (that is, the students do not know the other approaches enough to make an objective judgment) it is evident that the approach that builds on existing knowledge is much more popular.

6 Conclusions

The results of the survey confirm the correctness of the basic considerations used in the design of the new introductory programming course. The previous knowledge of new students was estimated correctly, and the level of difficulty of the course was chosen appropriately. Most students found the course useful or even useful and interesting. However, there is still room for improvement in the quality of lectures and laboratory practice which was judged approximately the same as in other courses.

It became evident that the choice of teaching language is a very delicate problem. There are different opinions among students over which factor should prevail: the support of principles of proper programming, or commercial use in practice. However, we strongly advocate the thesis that the programming language for the introductory course must primarily support the elements of good programming style, and most students agree. Despite its inadequately elaborated implementation Oberon was judged suitable by the majority of students.

The majority of students also agree with the incorporation of OOP in the introductory programming course. They feel that the portion of content devoted to OOP was adequate, but it should be slightly increased. They also support Oberon's approach to OOP, which treats OOP as an extension of traditional procedural programming.

Given the results of the survey, we decided not to make major changes in the course content; in particular we intend to continue using Oberon as the teaching language. We are planning only some minor improvements: a slight increase of the portion devoted to OOP, and more practical examples. Special attention will be devoted to the preparation of appropriate visual aids and Slovene literature with the aim of increasing the quality of lectures.

The major problem seems to be the present Oberon environment based on the Oberon operating system which has not been favorably received by the students. At the present we are evaluating a number of compilers that operate in standard environments (DOS, Windows, etc.), and that could serve our purpose better.

We are planning to especially motivate the best students, giving them additional, more difficult exercises which will be taken into account as a part of examinations. With all these improvements we are expecting still better results in the forthcoming year.

Key words: introductory programming course, object-oriented programming, Oberon

References

1. Wirth, N. The programming language Oberon, *Software Practice and Experience*, 1988, **18**, pp. 671-690.
2. Reiser, M. & Wirth, N. *Programming in Oberon – Steps beyond Pascal and Modula*, Addison-Wesley, Wokingham, 1992.
3. Mahnič, V. & Vilfan, B. A first course in object-oriented programming using Oberon, in *Software Engineering in Higher Education II* (ed J-L. Uso, P. Mitic & L.J. Sucharov), pp. 329-336, *Proceedings of the 2nd Int. Conf. on Software Engineering in Higher Education*, Alicante, Spain, 1995, Computational Mechanics Publications, Southampton, 1995.
4. Reiser, M. *The Oberon System*, Addison-Wesley, Wokingham, 1991.
5. Burgess, C.J. & Jones, B.F. Should software quality be a major issue when teaching first year programming to software engineers?, in *Software Engineering in Higher Education* (ed G. King, C.A. Brebbia, M. Ross & G. Staples), pp. 75-81, *Proceedings of the 1st Int. Conf. on Software Engineering in Higher Education*, Southampton, UK, 1994, Computational Mechanics Publications, Southampton, 1994.
6. Parsons, D. Pragmatism over purism? An incremental approach to teaching of object-oriented programming, in *Software Engineering in Higher Education* (ed G. King, C.A. Brebbia, M. Ross & G. Staples), pp. 375-382, *Proceedings of the 1st Int. Conf. on Software Engineering in Higher Education*, Southampton, UK, 1994, Computational Mechanics Publications, Southampton, 1994.