An Investigation of High School Students' difficulties with Iteration-Control Constructs

Emanuele Scapin, Claudio Mirolo

Abstract

A number of studies report about students' difficulties with basic flow-control constructs, and specifically with iteration. As part of a project whose long-run goal is identifying methodological tools to improve the learning of iteration constructs, we analyzed the answers of a sample of 164 high school students to three small programming tasks and two questions on their perception of difficulty. The results of the analysis suggest that more teaching efforts should be addressed to the development of a method to approach programming tasks and, more specifically for iteration, to the treatment of loop conditions in connection with the specifications in the application domain

Keywords: Informatics education, Programming learning, High school, Iteration constructs, Novice programmers

1. Introduction

Students' difficulties to master programming concepts are well known to computer science educators [1,5,7]. The reasons may be manifold, including lack of problem solving skills, accuracy, or intensive practice; according to Gomes and Mendes, in particular, programming requires "not a single, but a set of skills" [4]. However, part of the difficulties may be related to the habits and expectations of both teachers and learners [5,9]. Recurrent problems and misconceptions have been revealed even for such basic flow-control constructs as conditionals and loops [2,6]. Although these issues have not yet been widely explored for pre-tertiary education, anecdotal evidence suggests that high-school students may fail to develop a viable model of the underlying computation or be unable to grasp the connection between code execution and functional purpose.

On this basis, we engaged on a project to investigate the teaching and learning of *iteration*, at secondary school level, as well as to identify methodological tools

to enhance code comprehension. The first steps of this project, discussed in [10], explored teachers' and students' perceptions of the main difficulties that can hinder the mastery of programming, in general, and more specifically of iteration. Here we will proceed by analyzing students' performances in three *tasklets* involving different features of the iteration constructs and the answers to two questions about their subjective perception of difficulty.

The structure of the paper is as follows. After presenting, in section 2, the two questions and the three tasklets, in section 3 we summarize the results of the analysis. Then, in section 4 we discuss our interpretation of the findings and outline possible future perspectives.

2. Tasklets and questions

The survey was administered to 164 students attending classes on introductory programming, mostly at the end their second or third year, depending on the kind of school, lyceum or technical institute. The survey included a few general questions and three small tasks addressing each of the learning dimensions introduced in [8], namely the understanding of the computation model underlying iteration, the ability to grasp the relationships between loop components and problem at hand, the ability to abstract on the program structures based on iteration constructs. Two of the questions, in particular, were meant to investigate students' perception of difficulties with iteration.

To address program comprehension, the first aspect that we considered important to explore was the ability to understand the connection between loop condition and statement of the problem (*tasklet 1*, where the program was given in flow-chart form). A second aspect addressed by our analysis was students' mastery of the "mechanics" of the functioning of a loop controlled by a non-trivial condition (*tasklet 2*). The third aspect that we wanted to investigate was the ability to grasp comprehensively combinations of conditionals and iteration constructs, for which we asked to recognize equivalence between different programs (*tasklet 3*).

2.1. Tasklet 1: identifying the correct loop condition

Task description:
The algorithm represented by the flow chart in Fig. 1 computes the number of bits of the binary representation of a positive integer n, i.e. the smallest exponent k such that 2^k is greater than n. Choose the appropriate condition among the four listed below.

The four options are: "$2^k = n$", "$2^k \leq n$", "$2^k < n$" and "$2^k > n$".

This task can be achieved by carefully reading the above statement and then by figuring out the relationship between *k* and *n* after exiting the loop, depending on the chosen condition.

2.2. Tasklet 2: determining the number of iterations

Task description:

The program whose code is shown in Fig. 2 checks if two positive integers m and n are co-prime. If the input values are m=15 and n=44, how many times the while loop will iterate?

Students had to choose among five options: "0", "1", "2", "3", "4 *or more*", and "*the loop never ends*".

As we can see in Fig. 2, the loop presents a composite condition (with two *and*s) and a nested *if-else* construct. The correct option can be identified by tracing the code execution.

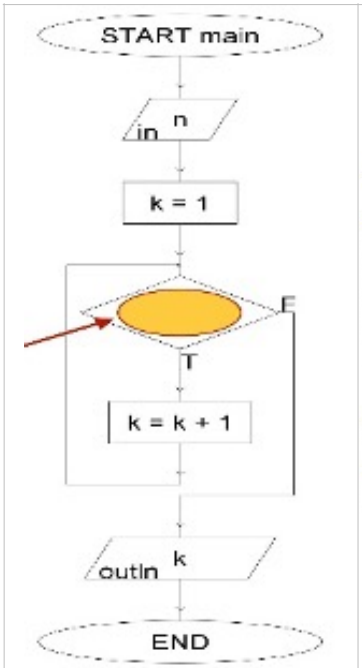

Fig. 1
Flow chart of tasklet 1

```
int x = m, y = n;

while (x > 1 && y > 1 && x != y) {
    if(x < y)
        y = y - x;
    else
        x = x - y;
}

if (x == 1 || y == 1)
    printf("m=%d and n=%d are co-prime\n", m,n);
else
    printf("m=%d and n=%d are not co-prime\n", m,n);
```

Fig. 2
Code of tasklet 2

2.3. Tasklet 3: recognizing functionally equivalent programs

Task description: *Consider the five programs in Fig. 3 and assume that the input values of m and n are always positive integers. Two such programs are equivalent if they compute and print the same output whenever they are run with the same input data. Identify the equivalent programs in Fig. 3.*

In order to approach this last task on functional equivalence, students were required to look at code execution from a higher abstraction level, so as to grasp the behavior of the nested constructs comprehensively.

2.4. Questions

The students were asked two simple questions about their difficulties and main sources of mistakes. To answer the first question, "*What do you find most difficult when you use loops*?", students could choose among the five most significant difficulties emerged from the teachers' interviews [10]: 1) To find the condition of the *while* or *do-while* loop; 2) To define a complex condition including logical operators such as AND, OR, NOT, XOR; 3) To deal with nested loops; 4) To understand, in general, when the loop should end; 5) To deal with the loop control variable.

To answer the second (open) question considered here, *"What kind of mistakes affected your performance most significantly?"*, students could indicate any potential source of error.

3. Results

In this section we will first present the data relating to the students' subjective perception; then we will analyze their performances on the three tasklets, also in connection with their perceptions of difficulties.

```
// 1
x = m;
y = n;

while ( x != y ) {

  while ( x < y )
    x = x + m;
  while ( x > y )
    y = y + n;
}

printf("result: %d\n", x);
```

```
// 2
x = m;
y = n;

while ( x != y ) {

  if ( x > y )
    y = y + m;
  else
    x = x + n;
}

printf("result: %d\n", x);
```

```
// 3
x = m;
y = n;

while ( x != y ) {

  while ( x < y || x > y ) {
    x = x + m;
    y = y + n;
  }
}

printf("result: %d\n", x);
```

```
// 4
x = m;
y = n;

while ( x != y ) {

  if ( x < y )
    x = x + m;
  else
    y = y + n;
}

printf("result: %d\n", x);
```

```
// 5
x = m;
y = n;

while ( x != y ) {

  if ( x < y )
    x = x + x;
  else
    y = y + y;
}

printf("result: %d\n", x);
```

Fig. 3
The five programs of tasklet 3

3.1. Questions

Let us first consider the multiple-choice question "*What do you find most difficult when you use loops?*". The percentages relative to each of the five options are summarized in the pie chart in Fig. 4.

The data show that, in the students' perception, the most significant difficulty is to deal with nested loops (41.5%). The definition of complex conditions comes next (23.2%), perhaps because composite conditions use Boolean operators and they do not know how to use them properly. The third most frequent option was envisaging a suitable termination condition for a *while* loop or a *do-while* loop (15.2%). Finally, the other possible answers were: understanding, in general, when an iteration should stop (12.8%) and managing the loop control variable (7.3%).

Now, the following observation is worth mentioning: while students perceived nested loops as a crucial issue, they were perhaps to a lesser extent aware of their difficulties concerning loop conditions that emerge in particular, as we will see shortly, from their responses to *tasklet 1*.

As to the open question about the mistakes having the most significant impact on students' programming performances, the recurrent answer patterns and the percentages of the related frequencies are listed in Table 1.

Table 1 highlights that several students tend to ascribe their poor achievements to generic factors, such as distraction/inattention (26.2%) or lack of time (5.5%), rather than to conceptual factors. On the other hand, they appear to underestimate other problems pointed out by their teachers, in particular carelessness while reading a text and insufficient housework practice. The most relevant conceptual issues identified by the students are connected with iteration (10.4%), confirming the importance of the subject in an introductory course, and with the management of functions and subroutines (10.4%), the latter being probably explained by novices' difficulties with parameter-passing and return values.

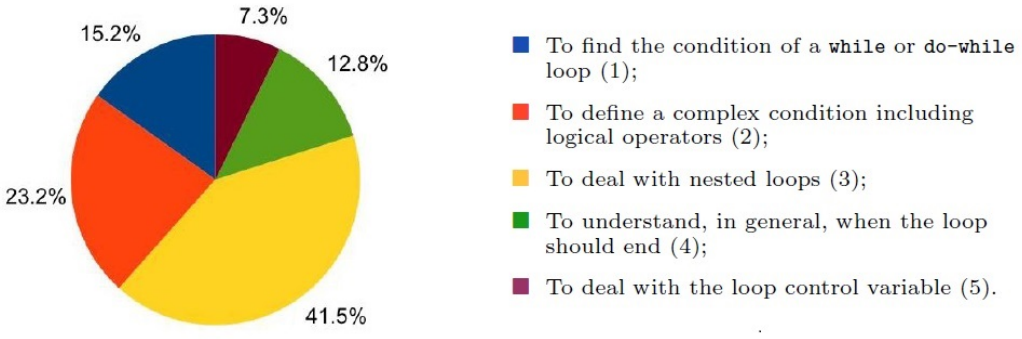

Fig. 4
Major difficulty with iteration in students' perception

Table 1
Reasons of mistakes identified by students

Kind of mistakes	Percentage
Inattention, distraction	26.2%
Iterations, loops	10.4%
Functions, subroutines	10.4%
Syntax, commands	7.3%
Understanding, interpretation	6.1%
Time, slowness, haste	5.5%
Logic, mathematics	4.9%
Objects, classes, methods	3.7%
Exercises, housework	3.7%
Without answers	10.4%
Others answers	11.6%
	100.0%

3.2. Tasklet 1: identifying the correct loop condition

The results obtained, for all the students who responded to the survey, are shown in Table 2. Less than 40% of them provided the correct answer ($2^k \leq m$), whereas more than 40% chose one of the two seriously wrong options ($2^k = n$ or $2^k > n$).

Table 2
Distribution of choices of the loop condition for tasklet 1

Option	Percentage	
$2^k = n$	3.7%	Correct
$2^k \leq n$	38.4%	
$2^k < n$	20.1%	
$2^k > n$	37.8%	
	100.0%	

The proposed problem, despite being simple, based on an elementary *while* loop and a plain condition, has highlighted some unexpected difficulties on the part of the students, probably due to the fact that they did not test their solutions by tracing the code, or perhaps because they have not a clear understanding of the mathematical meaning of the loop condition.

Below (see Table 3) the distribution of students' answers to *tasklet 1* are compared with their answers to the question regarding their difficulties with iteration[1].

Table 3
Choices of the loop condition vs. perception of difficulty with iteration

Option	1	2	3	4	5
$2^k = n$	0.0%	0.0%	1.8%	1.2%	0.6%
$2^k \leq n$	4.3%	9.2%	17.7%	5.5%	1.8%
$2^k < n$	4.9%	4.9%	7.3%	2.4%	0.6%
$2^k > n$	6.1%	9.2%	14.6%	3.7%	4.3%
	15.2%	23.2%	41.5%	12.8%	7.3%

The data presented in Table 3 show that there is not a large correlation between incorrect choices for *tasklet 1* and perception of this specific point as a major source of difficulty.

3.3. Tasklet 2: determining the number of iterations

In this case (see Table 4) about 60% of the students identified the right answer, i.e. 3 iterations. Thus, a large majority of them seem to understand the mechanics of iteration as well as the meaning of a composite loop condition. The higher rate of success in this task can probably be ascribed to the fact that the students were tacitly induced to trace the code execution, what they apparently did not do to test their conjectures about program behavior in the other two tasklets.

Table 4
Answers reporting different numbers of iterations

Number of iterations	Percentage	
0	3.7%	
1	9.1%	
2	15.2%	
3	60.4%	Correct
4 or more	6.1%	
never ends	5.5%	
	100.0%	

[1] For simplicity, in Table 3, 5 and 7 we refer to the answers to the first question by the numbers reported in Fig. 4, namely: 1) To find the condition of the while or do-while loop; 2) To define a complex condition including logical operators such as AND, OR, NOT, XOR; 3) To deal with nested loops; 4) To understand, in general, when the loop should end; 5) To deal with the loop control variable.

Again, we can compare the distribution relative to the different options of *tasklet 2* with students' subjective perception of difficulties with iteration, see Table 5. In this respect, it is interesting to note that the incidence of (awareness of) difficulties pertaining to the management of complex conditions is higher among those students who responded to *tasklet 2* correctly (15.2%) than among those who provided a wrong number of iterations (overall 7.9%), and it is almost negligible in connection with severely wrong options (no or unending repetitions: 1.2%).

Table 5
Choices of the number of iterations vs. perception of difficulty with loops

Number of iterations	1	2	3	4	5
0	1.2%	0.0%	1.2%	1.2%	0.0%
1	2.4%	0.6%	2.4%	1.8%	1.8%
2	3.1%	4.3%	6.7%	0.6%	0.6%
3	7.9%	15.2%	25.6%	7.3%	4.3%
4 or more	0.6%	2.4%	1.8%	1.2%	0.0%
never ends	0.0%	0.6%	3.7%	0.6%	0.6%
	15.2%	23.2%	41.5%	12.8%	7.3%

3.4. Tasklet 3: recognizing functionally equivalent programs

As shown in Table 6, less than 20% of all the students who answered the survey were able to recognize the two equivalent programs among those proposed in *tasklet 3*, namely program 1 and 4.

Table 6
Programs reported as equivalent in tasklet 3

Program	Percentage	
Programs 1 and 4	18.9%	Correct
Programs 4 and 5	13.4%	
Programs 2 and 4	11.0%	
Programs 1 and 3	7.3%	
Programs 1,4,5	3.7%	
Programs 1,2,3,4,5	1.8%	
Others	43.9%	
	100.0%	

In particular, a large part of the students (43.9%) provided meaningless or decidedly incorrect answers, often indicating only one program (30.5%), which suggests that they either did not understand the problem statement or made a random choice just to proceed with the next items in the survey. A number of students (11.0%) selected the pair of programs 2 and 4 as equivalent, perhaps

equivocating the condition of the nested *if*. In any case it seems that most students did not try to trace the code execution to test their conjectures. However, at least it appears that students' perception of difficulty with nested constructs is consistent with the actual state of affairs.

Finally, Table 7 compares the answers to *tasklet 3* with the subjective perception of difficulties when dealing with iteration. Again, there does not seem to emerge a clear correlation between actual and perceived difficulties with nested loops.

Table 7
Choices about program equivalence vs. perception of difficulty with iteration.

Program	1	2	3	4	5
Programs 1 and 4	1.8%	5.5%	7.9%	3.1%	0.6%
Programs 4 and 5	2.4%	4.3%	6.1%	0.6%	0.0%
Programs 2 and 4	1.2%	2.4%	5.5%	0.6%	1.2%
Programs 1 and 3	1.2%	2.4%	2.4%	0.6%	0.6%
Programs 1,4,5	1.2%	0.0%	1.8%	0.0%	0.6%
Programs 1,2,3,4,5	0.0%	0.0%	1.8%	0.0%	0.0%
Others	12.2%	8.5%	12.8%	6.7%	3.7%
	20.1%	23.2%	38.4%	11.6%	6.7%

4. Discussion

The findings summarized in the previous section, relative to the high school context, appear in accordance with the conclusions of previous work addressing novices' difficulties with conditionals and loops, as pointed out for instance in [2,6]. More specifically, dealing with nested flow-control structures and, to a lesser extent, with loop conditions seem to be major challenges, the former also in the students' perception.

Here are a few additional thoughts:

- The poor performance in *tasklet 1* suggests that several students cannot master the loop condition in connection with the problem at hand, sometimes possibly because they are undecided between the roles of "exit" vs. "continue" condition.
- When students have to trace the code execution, as in *tasklet 2*, most of them are able to determine the correct outcome or make minor mistakes. This suggests that most high school students develop a viable mental model of the *notional machine* [3] underlying code execution.
- It is interesting to observe that as many as 77% of the students who achieved *tasklet 2* successfully, provided *seriously* incorrect answers to *tasklet 1* or *tasklet 3*. So, it seems that students are not inclined to exploit their tracing abilities in order to test their first conjectures about code execution. This may be due either to laziness or to lack of method to

approach programming tasks, the latter being relevant from a pedagogical perspective.

- Finally, the students seem to underestimate their difficulties to deal with loop conditions. Indeed, only a small percentage of those who were wrong in *tasklet 1* or *tasklet 2* seem also to be aware of that problem.

Insights for instructors

Two major insights can be drawn from the points listed above. First, more effort should be focused on the development of a method to approach programming tasks, in particular to identify suitable test cases. Second, the role and treatment of loop conditions is worth more attention, especially in connection with the problem domain.

5. Conclusions

In this paper we have described one of the preliminary steps of a project whose main goal is to identify methodological instruments to improve high school students' mastery of iteration. We have analyzed how a sample of students responded to a survey including three small programming tasks and two questions about their subjective perception of difficulty. The results seem to suggest that most students have developed a sufficiently accurate model of the notional machine, but probably lack an operating method to approach programming tasks.

We are currently planning to extend the scope and depth of this exploratory study by designing a survey to cover a range of (small) programming tasks, to be used as a basis to devise effective instructional strategies.

References

1. Bennedsen, J., Caspersen, M. (2007). "Failure rates in introductory programming", *SIGCSE Bulletin* 39, 3-36
2. Cherenkova, Y., Zingaro, D., Petersen, A. (2014). "Identifying challenging cs1 concepts in a large problem dataset", Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14). 695-700.
3. du Boulay, B. (1986). "Some Difficulties of Learning to Program". *J. of Educational Comput. Research* 2(1). 57-73.
4. Gomes, A., Mendes, A. (2007). "Learning to program - difficulties and solutions", *Proceeding of the International Conference on Engineering Education (ICEE)*. 283-287.
5. Jenkins, T. (2002). "On the difficulty of learning to program", *Proceedings of the 3rd annual LTSN ICS Conference*.
6. Kaczmarczyk, L.C., Petrick, E.R., East, J.P., Herman, G.L. (2010). "Identifying student misconceptions of programming", *Proceedings of the*

41st ACM Technical Symposium on Computer Science Education (SIGCSE '10). 107-111.

7. Lewandowski, G., Gutschow, A., Mccartney, R., Sanders, K., Shinners-Kennedy, D. (2005). "What novice programmers don't know", *Proceedings of the first international workshop on computing education research (ICER '05)*. 1-12.
8. Mirolo, C. (2012). "Is iteration really easier to learn than recursion for cs1 students?", *Proceedings of the Ninth Annual International Conference on International Computing Education Research* (ICER '12). 99-104.
9. Robins, A., Haden, P., Garner, S. (2006). "Problem distributions in a cs1 course", *Proceedings of the 8th Australasian Conference on Computing Education (ACE '06)*. 165-173.
10. Scapin, E., Mirolo, C. (2019). "An Exploration of Teachers' Perspective About the Learning of Iteration-Control Constructs" in Pozdniakov S., Dagienė V. (eds) *Informatics in Schools. New Ideas in School Informatics.* ISSEP 2019. Lecture Notes in Computer Science, vol 11913. Springer, Cham.

Biography

Claudio Mirolo is researcher in Computer Science at the University of Udine, department of Mathematics, Computer Science and Physics, where he currently teaches introductory programming, computational geometry and computer science education. He is also responsible for the education and training programmes offered to prospective high school teachers of computer science. His research interests include students' learning of programming, both at the university and upper secondary levels of instruction, as well as the role of computational thinking in general primary and secondary education.

Email: claudio.mirolo@uniud.it

Emanuele Scapin is an experienced Computer Science teacher at the I.T.T. "G. Chilesotti" in Thiene (VI). Currently, he is pursuing a PhD programme at the University of Udine, with a research project in Computer Science Education. His main topic of interest concerns task-related models to improve the learning of programming, and more specifically iteration, at the upper secondary instruction level.

Email: scapin.emanuele@spes.uniud.it