

Fostering Strategic Knowledge and Program Comprehension Skills in Students Struggling with CS1

Violetta Lonati, Anna Morpurgo - {lonati, morpurgo}@di.unimi.it
Università degli Studi di Milano, Milan, Italy



UNIVERSITÀ
DEGLI STUDI
DI MILANO

Summary

In order to support students struggling with learning to program, we designed new learning material, around two basic principles from recent research on introductory programming education.
The material was prepared and used in the context of a peer tutoring initiative targeted at under-graduate CS major students.

Basic principles

Program comprehension vs Program writing

Program comprehension is the process of building a “mental model of a program”.

- There is strong correlation between novices’ tracing, reading and writing skills
- Novices tend not to abstract beyond the concrete code, they have difficulties at summarizing the purpose of a piece of code.
- A mix of code-writing and code-reading is deemed to be the most effective way for students to ultimately develop good programming skills

C. Izu et Al., 2019, *Fostering Program Comprehension in Novice Programmers - Learning Activities and Learning Trajectories*.
T. Clear et Al., 2011. *Report on the final BRACElet workshop*.
M. Corney et Al., 2014. “*Explain in Plain English*” *Questions Revisited: Data Structures Problems*.

Teaching strategic knowledge explicitly

Strategic knowledge is the knowledge of stereotypical solutions to typical problems and the ability to apply, tailor, and combine them to solve new problems.

- Soloway’s “goals & plans” provide the terminology to address strategic knowledge.
- Many authors suggest explicitly teaching strategic knowledge, since for many students this represents a real challenge.

T. McGill, S. Volet, 1997, *A conceptual framework for analyzing students’ knowledge of programming*.
E. Soloway, 1986, *Learning to Program = Learning to Construct Mechanisms and Explanations*

Preliminary validation

The general reception of the material was positive; students claimed that they did improve.

Self-evaluate your skills (from 1 = none, to 4 = total)	Before	After	Δ
Trace a piece of code on a given input	1.94	2.88	0.94
Detect errors in a piece of code	2.06	3.00	0.94
Apply to a new problem solution already seen	1.94	3.00	1.06
Recognize problems that are similar	2.00	2.97	0.97
Write programs starting from requirements spec.	1.91	2.81	0.91

Some example tasks from a sequence on iteration plans

A. Code analysis

Cluster the 9 code snippets based on similarities. Which criterium did you use to cluster them? What do the first and third snippet have in common?

Description and purpose

Code snippets with loops (here just a subset of the original task) are presented; similarities and differences are to be found in their structure and semantics, besides the syntactical surface. This program comprehension task prepares for the introduction of the typical iteration goals and plans (*repetition, total, count, min/max, first occurrence, prev/curr*) with their terminology.

<pre>product := 1 for i := 0; i < DIM; i++ { product *= numbers[i] }</pre>	<pre>alternating := true _, err := fmt.Scan(&curr) for err != EOF { prec := curr fmt.Scan(&curr) if (prev > 0) == (curr > 0) { alternating = false break } }</pre>	<pre>sum, i := 0, 0 for sum < TARGET { sum += numbers[i] i++ if sum >= TARGET { break } } fmt.Println(TARGET, "reached at", i)</pre>	<pre>var min byte = text[0] for i := 1; i < DIM; i++ { if text[i] < min { min = text[i] } }</pre>
Total plan	Prev/Curr plan	Total plan	Min/Max plan

B. Analysis of requirements

For each of the following requirements specifications, answer these questions.

1. In the implementation of the program, which is the main iteration goal to achieve?
2. Which variables are needed in the implementation of the corresponding iteration plan? How do they have to be initialized?

Solved example

Requirements: “Write a program that reads a sequence of words from standard input and prints how many of them contain the character ‘a’.”
Answers: The main goal is a counting goal. To implement the corresponding iteration plan, a *counter* variable is needed which keeps track of how many ‘a’s have been encountered so far, which must be initialized to 0.

Description and purpose

The task aims at practicing in analyzing requirements specifications, in order to recognize the main iteration goal before designing and implementing a solution.

C. Parsons problem

The function `strangeProduct` is given a series of integers and returns the product of all numbers in the series that are greater than 7 and multiple of 4. For example, if the numbers in the series are 12, 3, 4, 8, 9, 2, the function will return 96 (the product of 12 and 8). The lines of the function have been jumbled. Place the lines into the correct order.

```
product *= n
}
return product
}
for _, n := range numbers {
  if n > 7 && n%4 == 0 {
    product := 1
    func strangeProduct(numbers []int) int {
  }
```

Description and purpose

In this task the student must recognize the underlining iteration plan and reconstruct it by reordering the instructions into a proper arrangement.
The purpose of this kind of tasks is to consolidate the knowledge on the specific aspects and structure of each iteration plan.

D. Code writing guided through observations

Requirements

Write a program that reads a series of numbers from standard input and, starting from the second one, prints the difference between the current number and the previous one, that is it prints the difference between the second and the first, between the third and the second, and so on. The program halts when it reads 0.

Consider that:

- You need to handle two *adjacent elements* at the same time, so you need the *prev/curr* plan;
- you will need to compute the difference between the last read number and the number that was read just before it;
- you will need two variables, `prev` and `curr`, that must be properly updated.

Description and purpose

The program writing task is guided by some observations that help the students in focusing on specific aspects of the iteration plan useful for this problem.
The purpose of this kind of tasks is consolidating the knowledge about iteration plans and practicing in the plan implementation once the plan has been identified.

The material is available at <https://aladdin.unimi.it/procosk>