Deducing Loop Patterns in CS1: a Comparative Study
José Luis Fernández Alemán
Faculty of Computer Science, University of Murcia, Spain
aleman@um.es

Abstract
This paper presents a proposal for implementing iterative algorithms based on the definition of algorithmic schemes starting from identifying four sequential access models. A learning method is applied to get students to develop their cognitive capabilities in such a way that they are able to deduce the algorithmic schemes for a sequential access model given. In a controlled study, programming students trained using the sequential access models were found to make fewer errors than a control group which suggests that our approach helps students to construct iterative algorithms correctly.

1. Introduction
The loop is an overriding concept in an introductory programming course. However, a large number of studies have shown that CS1 students do not understand properly basic loops [3]. The approach proposed here describes a method to deduce loop patterns, taking into account four sequential access models. Our goal is to deal with and to extend the ideas proposed by Scholl and Peyrin [2], which emphasizes problem solving strategies and program design issues rather than specific language features.

2. Sequential access models and algorithmic schemes
Winslow [3] notices that models are very important to build understanding, in particular, models of control. If the instructor omits them, students will make up their own models and rules of dubious quality. Therefore, models must be taught by the instructor [1]. A deep knowledge of sequential access models is essential for beginning programmers.

In order to access a sequence, two operations are used. The first access operation is used to start the sequential access. In accordance with the underlying model, two kind of start operations are found. The former, named here Start, obtains the first element of the sequence, if the sequence is not empty. The latter, named here Initiate, obtains none of the elements, it only prepares the sequential access. The second access operation, named here Advance, is used to continue the sequential traversal. Its execution gives rise to the following element. According to the underlying model, after executing an Advance operation, the end of the sequence is detected when the last element is obtained or when the last element is exceeded, by using a query function, named here IsLast and IsEnd, respectively. As a result of combining the two ways of initiating and ending the traversal of a sequence, four sequential access models emerge.

The sequential traversing schemes can vary according to the underlying sequential access model, but the treatments of each sequence element are independent of the model. For example, “given an integer sequence S, calculate the maximum integer of S”. Here, the treatment would be checking if current element is greater than current maximum, using inductive reasoning. For this reason, the focus will be on the order of the operations sequentially executed.

Three operations are identified in a sequential traversal: advance (A), treatment (T), and checking (C). The operations A and C are dependent of the sequential access model and are part of the concrete pieces of the algorithmic scheme, whereas the operation T is dependent of the problem. The operations used to check the end of the loop determine the order of execution of A, T, and C. An IsEnd operation (first and fourth model) forces to introduce an order T-A-C, whereas an IsLast (second and third model) operation establishes an order A-T-C. Figure 1 shows the sequentiation in deploying the loop operations according to each model. The IsEmpty operation is used to check if a sequence is empty. In some data types of a programming language, pairs of these operations such as (IsEnd, IsEmpty), (IsLast, IsEmpty) or (Start, Advance) might be confounded. Observe that, from a point on any sequence, operations repeat themselves. Depending on the position of the checking operation in the repeated subsequence, the right iterative statement is chosen: while-endwhile (checking is performed at the beginning), repeat-until (checking is performed at the end), loop-exit-endloop.
(checking is performed in a middle point). Figure 2 shows three algorithmic schemes of the first sequential access model obtained from the deployment of operations shown in Figure 1a.

![Figure 1. Deployment of the operations.](image)

![Figure 2. Traversing algorithmic schemes.](image)

![Figure 3. Four searching algorithmic schemes.](image)

In the sequential searching scheme there is no treatment, but the continuation (or termination) condition has more importance. For example, “given a character sequence \( S \), indicate if \( S \) contains some occurrence of the character ‘x’”. Two operations are identified in a sequential search: advance (A), and checking (C). The operation A is part of the concrete pieces of the algorithmic scheme, whereas the operation C allows us to check the end of the sequence (CheckEnd or CheckIsLast) and to check certain property in the current element (CheckProperty). Therefore, the checking operation is dependent of the problem. To determine if some element satisfies certain property, ending treatments are required. If it is used the IsEnd operation (first and fourth models), the ending treatments consist on checking the end of the sequence by querying the IsEnd operation, whereas if it is used the IsLast operation (second and third models), the ending treatments consist on checking if the current element satisfies the property. Figure 3 shows four searching algorithmic schemes, one for each model.

### 3. Evaluation of the approach

The approach proposed here has been used effectively in an introductory computer programming course called Methodology and Technology of Programming at the University of Murcia.

To measure the educational effectiveness of the scheme-based learning, an experiment was designed and conducted during the second semester of 2008. Data were collected from 114 programming students, divided into two groups. An experimental group of fifty seven participants had gathered experience in constructing iterative algorithms using sequential access models. A control group of fifty seven subjects lacked experience with regard to work on control models. The data obtained suggest that students were able to comprehend algorithmic schemes with the greatest of ease (over 82% of solved problems for the experimental group and 70% of solved problems for the control group). However, they demonstrated more difficulty debugging and developing programs, with a rate of 57% and 44%, respectively, for the experimental group, and 41% and 27%, respectively, for the control group.

### 4. Conclusions

A simple and disciplined method to design loops has been proposed. Four sequential access models have been identified. The results of a comparative study between an experimental group and a control group show that a significant improvement in the learning process can be obtained using our approach.

### 5. References

