INFORMATICA, 1993, Vol.4, No.3–4, 295–302

A PSEUDO-DIRECT EXECUTION OF ALGORITHMS USING TURBO PASCAL APPLIED IN PROGRAMMING TEACHING PROCESS

Valentina DAGIENĖ

Institute of Mathematics and Informatics
2600 Vilnius, Akademijos St. 4, Lithuania

Abstract. An algorithm testing methodology used in the programming teaching process is presented. Turbo Pascal system is used. Due to systematic use of some features of this system a pseudo-direct execution of algorithms expressed by Pascal procedures and functions may be ensured. The need to use data read and write statements is excluded. This enables a novice learner to concentrate himself on the main actions of an algorithm.

Key words: algorithm, program, programming, teaching of programming, distance teaching, Turbo Pascal.

1. Introduction. An algorithm describes a sequence of actions describing how to get output data from input data. A computer program adds to the algorithm a set of extra actions describing how to get the input data and how to display the results. Using Pascal these actions are expressed by read, write and other statements operating on files. On the other hand, the main task of the development of an algorithm is to find a solution of the given problem while a program includes all phases of problem solving by computer.

There are several teaching cases when it is desirable to concentrate on main actions of an algorithm and thus learning of actions concerned with data input and output are irrelevant and redundant. Let us mention two samples of such cases.

1. Student's access to a computer and/or communication be-

tween a student and a teacher is limited. This situation is common in a distance teaching of informatics.

2. There is a tendency to achieve more general goals in the algorithm teaching process. Such goals may be as follows: to develop thinking, especially logic thinking, to develop problem solving abilities, especially if the problems are unusual and require deep thinking or consideration analytic.

From the methodological viewpoint of teaching (Dagienė and Grigas, 1991). Pascal procedures and functions are more suitable entities for algorithms representation than programs. However, Pascal translators can't accept these constructions (procedures and functions) for direct execution. Turbo Pascal is no exception. Therefore, procedures and functions have be incorporated into a computer program.

Therefore, a demand for a direct execution of Pascal procedures and functions is evident. This problem can be solved in two ways.

1. Using software designed a specially for direct execution of the procedures and functions. An interpreter of algorithms (Dagienė and Žandaris, 1992) may be given as an example of such software.

2. Systematically using some peculiarities of a standard software in order to imitate direct execution of algorithms. Let us call such approaches as pseudo-direct.

Later on we will describe the approach which corresponds the second way.

2. Turbo Pascal features useful for algorithm testing. The purpose of the design of Pascal was to teach the discipline of programming. Turbo Pascal was designed for professional programming. It contains a great number of the Standard Pascal extensions. (To be precise – of a subset-very close to the Standard Pascal.) Let us denote the Standard Pascal by letter S and Turbo Pascal by letter T and use the Pascal notations for presenting operations. Let us divide all extension of Turbo Pascal into two groups: TP – those, which are useful (positive) for teaching (e.g., operations

with strings, structured constants) and TN – those, which are useless (negative) or even harmful in this respect (e.g., operations with codes of the values). So, we have

$$T - S = TP + TN,$$
$$TP * TN = [\ ].$$

Turbo Pascal was designed for professional programming, so it is reasonable to expect that

$$TP > TN.$$

It is reasonable to use a set of

$$S + TP$$

for teaching purpose. The systematic use of TP (or its subset) together with S (or its subset) may increase the chances for Turbo Pascal to regain some pedagogical power which was lost because of TN. Here we concentrate ourselves on some distinguishable features of TP which make Turbo Pascal suitable for executing and testing of algorithms presented by procedures and functions.

1. We can use typed constants of structured data types in the constant definition part.

EXAMPLE:

```
type point  = record
                x, y : real
              end;
     vector = array [0..1] of point;
const M : point = (x : 0.0; y : 0.0);
      line : vector = ((x : 3.1; y : 1.5),
                       (x : 5.9; y : 3.0));
```

2. Structured values of variables (and constants) can be displayed on the screen by Turbo Pascal compiler when we use the debugging option.

3. Program heading can be omitted.

4. The order of definitions and declarations is unimportant. Definition or declaration parts of the same sort may appear more than once.

EXAMPLE:

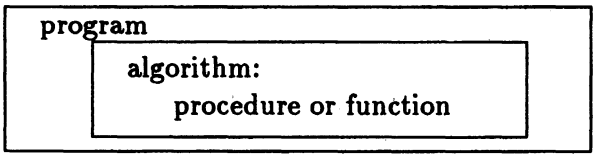
const *pi* = ...
var *x* : ...
const *u* = ...
function *p* ...
type *t* = ...
var *z* : ...

Features named in the points 1, 3 and 4 concern the language of Turbo Pascal and that in the second point the compiler.

The most significant points in our context are the last two. They allow us to reconstruct relationship between the main part of a program and of the procedure or function. According to the Standard Pascal syntax the components of programs may be laid out in the strictly prescribed order as follows:

program heading,
label declaration part,
constant definition part,
type definition part,
variable declaration part,
procedure and function declaration part,
statement part.

The procedures and functions are enclosed inside the program, and the structure of the whole program text may be expressed by this scheme:



This scheme suits for the Turbo Pascal too.

Turbo Pascal however, allows program headings to be omitted, and the order of definition and declaration parts to be arbitrary, e. g.,

procedure and function declaration part,

label declaration part,

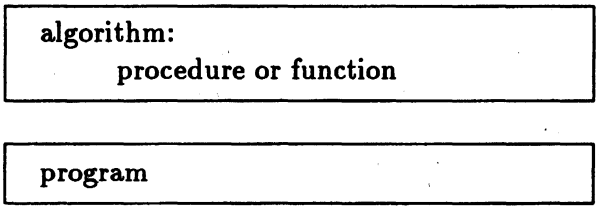
constant definition part,

type definition part,

variable declaration part,

statement part.

So we considered, that procedures and functions can be presented in the very beginning of the program text. It makes an impression that an algorithm is not enclosed in a program. The situation can be expressed by a scheme:



EXAMPLE:

```
{ algorithm }
 procedure minmax (a, b : integer;
                   var min, max : integer);
 begin
     if a < b
        then begin
                min := a;
                max := b
             end
        else begin
                min := b;
                max := a
             end
 end;
 { ———————————————————— }
```

```
{ program }
  var mn, mx : integer;
  begin
      minmax (6,7,mn,mx)
  end.
```

3. Methodologies of algorithm testing. The Turbo Pascal features described earlier ensure convenient direct-like communication with an algorithm (procedure or function). This communication is achieved by presenting data values directly on a display screen and thus excluding the use of data read and write statements. The procedure and function calls are written in the program. All values of scalar input data are put down directly into procedure or function call statement. Such values, which are of structured data types or which change within a loop (in the case of a multiple execution of the algorithm), are denoted by identifiers and can be changed appropriately (as a loop variable or by a structured constant definition). All output data values are denoted by variable identifiers in the procedure call. The values of these variables (including structured ones) can be observed in the watch window of debugging option. This is valid for functions too, but function call must be written in the left side of an assignment statement.

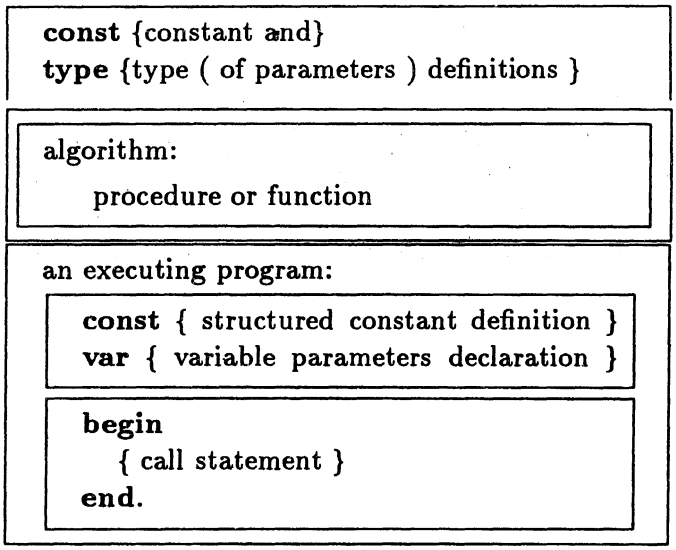
Thus the testing scheme consists of to parts:

1) an algorithm (procedure or function or their collection);

2) an executing program.

Procedure or function can have parameters, whose types are defined by a user. In this case their type definitions must (immediately) proceed the algorithm. Those definitions may be considered as a component part of the algorithm.

Let us give the pattern of testing algorithm.



4. Conclusions. Turbo Pascal allows more freedom using the particular language constructions than the Standard Pascal. In particular the program heading may be omitted, the order of definition and declaration parts may be arbitrary. Systematic and objective use of these features together with certain debugging options of Turbo Pascal system may ensure direct supply of procedures and functions with input test data and direct observation of their results. On the basis of these features an approach of the pseudo-direct execution of Pascal procedures and functions is elaborated. The approach is useable for programming teaching especially in distance teaching.

REFERENCES

Dagienė, V., and G. Grigas (1991). An environment for teaching of algorithms. *Informatica,* **2**(4), 473–477.

Dagienė, V., and G. Grigas (1991). *Informatika. 10–12.* Šviesa, Kaunas (In Lithuanian).

Dagienė, V., and A. Žandaris (1992). Algorithms interpreter for schools. *Informatika,* **20**, 39–51 (In Lithuanian).

Received 1993 November

V. Dagienė graduated Vilnius University in 1978 and received the Degree of Doctor of Mathematics (Ph.D) from Vytautas Magnus University (Kaunas, Lithuania) in 1993. She is a senior research worker of the Department of Programming Methodology at the Institute of Mathematics and Informatics. Published ten books and many articles of programming. Her research interests include teaching of informatics, programming and algorithmization methodology.