INFORMATICA, 1995, Vol. 6, No. 3, 265–276

INVESTIGATION OF THE RELATIONSHIP BETWEEN PROGRAM CORRECTNESS AND PROGRAMMING STYLE

Gintautas GRIGAS

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

Abstract. The results of investigation of computer programs written by school students during the 6th International Olympiad in Informatics are presented. Pascal program texts are analyzed on the lexical level. A certain relationship is indicated between program correctness and usage of some programming constructs as well as readability of the program text. The results are discussed from the standpoint of programming teaching.

Key words: programming style, program metrics, teaching of programming, Turbo Pascal.

1. Introduction. It is widely accepted that the programming style has some impact on program correctness and the efficiency of programmer's work. However, it is difficult to define rigorously the elements of programming style and to express such impact quantitatively. We present here some results of our investigation. We choose the programs written during the International Olympiad for a number of reasons namely:

1. All programs are written, tested and evaluated in the uniform environment.
2. The authors of the programs are from different countries and gained their knowledge and experience in programming by different ways. So, they represent a world wide community of skilled school students.
3. The programs are graduated exclusively by the results of the tests. So the programming style has no direct effect on the points and the student wrote the program text only for himself/herself, freely expressing his/her own program writing habits.

The 6th International Olympiad in Informatics was held in Stockholm, July 3–10, 1994. 189 school students from 49 countries took part in this event.

The students worked two days. Each of the two days the students were asked to write programs for 3 problems within five-hour time limits. They used AST'486 33 MHz computers operating under MS-DOS 6.2.

2. Programming languages. Four programming languages have been used by the students: Turbo Pascal release 7.0, Turbo C++ release 3.0, LCN Logo release 3.0, and Quick Basic release 3.0. Each participant could freely choose any language. Programming languages, actually used by the participants, are shown in Table 1.

Table 1. Programming languages used by participants according to their success, expressed by the sort of medal

Programming Language	Gold	Silver	Bronze	None	Total
Pascal	17	25	38	56	136
C	0	9	8	10	27
Pascal/C	0	0	0	1	1
Basic	0	0	0	13	13
LCN Logo	0	0	0	2	2
? (no data available)	0	0	4	6	10
Total	17	34	50	88	189

The most popular language was Pascal.

The percentage of Pascal programmers according to the medals they were awarded is as follows:

gold	100.0%
silver	73.3%
bronze	76.0%
none	60.2%.

So Pascal was the most successful language.

The students were asked to present their compiled programs (EXE files) for testing and evaluation. This did not ensure saving all source code texts. Indeed, we have no records of source code programs of 10 participants (4 of them have gained total 0 points, and somebody have not presented programs for testing at all). However, we are sure that the missed programs have no noticeable impact on the statistical results of our investigation.

3. Programming problems and evaluation of their solutions. The formulation of programming problems as they were given to the students may be found elsewhere (e.g., in: 6th International Olympiad in Informatics, 1995; Stromberg, 1994).

Some figures describing the problems are given in Table 2. All the problems are of numerical nature. All input and output data have to be nonnegative integers or letters. It was required to find the unique solution or one of possible, in some cases, optimal.

Table 2. The characteristics of problems and their solutions (programs)

Number of problem	1	2	3	4	5	6
Maximum run time of test in sec	30	30	90	30	30	30
Number of test	6	5	4	5	6	8
Number of points per test	5	6	10	6	5	5
Maximum number of points	30	30	40	30	30	40
Average number of points assigned to a program	20.9	21.9	8.3	10.3	8.8	14.4
Number of programs assigned p points						
a) $p = \max$ (i.e. all points)	75	63	12	20	0	5
b) $0 < p < \max$	36	41	20	77	93	49
c) $p = 0$	21	19	72	24	33	26
d) total	132	123	104	121	126	80
Length of program (in lines)						
a) min	22	55	64	32	40	50
b) max	202	285	591	376	305	560
c) average	66	148	200	162	116	154

The programs were evaluated by points for successful tests. The tests reflect the correctness and partly effectiveness of programs. Execution of programs have to fit in the particular time limits. However, the limits were not so critical.

So the number of points assigned for a program may be considered as a suitable parameter which express the measure of program correctness in this context.

The complexity (difficulty in programming) of problems was estimated by assigning different number of points for a particular problem. The complexity of a problem may be also illustrated by the length of its program text, as shown in Table 2.

4. Program set or investigation. We have selected only Pascal programs for this investigation. We included all Pascal programs that were compiled and their compiled code (EXE file) was presented by the author for the jury to evaluate. However, we excluded two programs from the set the main part of the text of which was generated automatically (they included large constant arrays of prime numbers and their size exceeded by 10–20 times the size of programs written in the conventional way).

The total number of programs investigated was 686. The programs were written by 135 students from 46 countries.

5. Parameters of programs. Over fifty parameters of the program text were investigated. For illustration the majority of them is presented in Table 3. The numbers of the parameters are used for references later in this paper.

Table 3. The list of program parameters and their average values calculated for all students and in accordance with the number of their points expressed by the sort of medal they are awarded

No	Program parameter	Average	Gold	Silver	Bronze	None
	Lenght of:					
1	program	846.3	913.5	812.9	852.5	832.7
2	block	165.5	175.0	147.2	172.4	166.5
3	Turbo lexic %	14.0	13.6	14.7	13.2	14.3
	Number of:					
4	functions	1.3	1.0	1.1	1.4	1.3
5	procedures	4.0	4.2	4.0	4.1	3.8

to be continued

Table 3. Continuation

No	Program parameter	Average	Gold	Silver	Bronze	None
	% of programs with:					
6	heading (**program**)	68.5	74.3	59.9	81.0	60.9
7	**uses**	40.8	19.8	28.9	34.0	62.1
8	**const**	50.4	61.4	61.3	49.0	40.7
9	**type**	52.6	53.5	57.7	53.0	49.0
10	pointers	8.3	10.9	7.0	12.0	4.9
11	**goto**	5.5	9.9	3.5	3.0	7.0
12	boolean	65.3	65.3	54.9	69.5	67.9
13	*longint*	24.3	37.6	23.2	21.0	22.2
14	*char*	16.0	10.9	16.2	17.0	17.3
15	*string*	16.3	10.9	6.3	15.0	25.5
16	**record**	19.5	21.8	18.3	21.0	18.1
17	**set**	5.5	5.9	8.5	4.0	4.9
18	**object**	1.6	0.0	0.7	1.5	2.9
19	comments for compiler	40.1	50.5	39.4	47.0	30.5
	Number of statements					
20	**for**	8.9	9.9	8.9	9.2	8.3
21	**while**	1.0	1.2	1.2	1.0	0.7
22	**repeat**	0.5	0.2	0.4	0.6	0.8
23	**if**	10.7	11.0	9.9	10.4	11.2
24	**case**	0.2	0.2	0.2	0.2	0.3
25	**exit**	1.4	2.0	1.7	1.4	1.0
	Ratio of statements in %					
26	**for** : all cycles	85.1	87.7	85.4	85.6	83.3
27	**repeat** : all cycles	5.2	1.3	3.5	4.9	8.2
28	Comments %	3.5	2.0	6.7	2.6	3.0
29	Line density %	90.9	93.2	89.1	91.7	90.5
30	Character density %	90.4	90.7	90.5	89.7	90.9
31	Lines per identation	2.5	2.1	2.3	2.6	2.7
32	Lenght of identation	2.8	2.5	2.8	2.7	3.1
33	Text layout	3.1	3.2	3.4	3.1	2.7

Let us comment some of the parameters.

(1). The length of program is given in lexems. The lexem is an indivisible entity of a program text. The lexem may not be split by blank spaces or end-of-line characters. Examples of lexems: an identifier, a number, an operator, a reserved word. Comments, empty lines or blank spaces are not lexems. The number of lexems does not depend on splitting the text into lines, the length of comments or strings.

(2). The length of block is expressed in lexems. The function, procedure as well as constructor or destructor for those who use object-oriented features of Turbo Pascal, the main program body, or the unit (each) is considered as a block.

(3). The ratio between Turbo Pascal and Standard Pascal lexic expresses the level of deviation from Standard Pascal towards Turbo Pascal. It is measured by the amount of percents of Turbo Pascal words (reserved words and Turbo Pascal identifiers) in the total number of reserved words and standard identifiers.

(28). In order to consider only the comments for the reader, the draft pieces of program texts enclosed within comment parentheses, were excluded from the program texts. The comments for compiler (marked by $ in the program text) were also not included here for the same reason (such comments are considered as separate program parameter (see 19)).

(29). Line density is a ratio (expressed in %) between the number of nonempty lines and the total number of lines. Empty lines (resulting lower line density) may increase program readability.

(30). Character density is a ratio (expressed in %) between the number of nonblank characters and the total number of characters including blank spaces. Extra spaces (resulting lower character density) may increase program readability. Leading identation blanks in the left of lines are not included here.

(31). Lines per identation is an average number of lines within the single identation step.

(32). Identation length is an average number of leading spaces for a single identation.

(33). Text layout of program text is evaluated by points from 0 to 5 and expresses the readability of the program text from the aesthetical and typographical points of view, as described in Baecker and Marcus (1992). The parameter was evaluated by the experts.

All parameters but the last one (33) were examined automatically by the program of lexical and limited syntax analysis.

We did not include in the Table 3 the parameters used by all (or almost all) authors in every programs (e.g., integer and array types), or those used in a very small fraction (less than 1 %) of programs (e.g., units, the assembler code pieces, reals, statement with).

It would be of interest to pick out from Table 3 the parameters the values of which monotonously decrease or increase in line with the skill of the authors of programs. They are collected and presented in Table 4.

Table 4. The values of some selected program parameters expressed by % relative to average value (considered as 100%)

No	Program parameter	Gold	Silver	Bronze	None
7	**uses**	48.5	70.7	83.3	152.2
8	**const**	121.7	121.5	97.2	80.8
14	*char*	67.9	101.0	106.0	107.8
18	**object**	0.0	43.9	93.6	179.6
21	**while**	122.2	120.8	102.0	77.0
22	**repeat**	30.9	69.8	101.8	144.9
25	**exit**	140.5	118.6	102.9	70.0
27	**repeat** : all cycles	25.5	66.7	92.7	156.4
31	lines per identation	83.8	93.3	103.7	107.6

7. Correlation coefficients. The level of linear dependence between two parameters may be expressed by the correlation coefficient. Three coefficients were calculated for each parameter:

1) Parameter p of a program relative to the number of points gained for the program of a particular problem (k_p). It characterizes the dependance between the parameter and program correctness for the given problem. There is a separate value of the coefficient for each problem. The values of the all 6 coefficients are given in Table 5.

Table 5. Correlation coefficients k_p between program parameter p and the number of points gained for the program of particular problem

No	Program parameter	Problem No					
		1	2	3	4	5	6
7	**uses**	−0.44	−0.39	−0.37	−0.36	−0.27	0.02
8	**const**	0.20	0.22	0.21	0.30	0.09	0.17
13	*longint*	−0.05	−0.05	0.17	0.18	0.22	0.46
15	*string*	−0.38	−0.28	−0.15	−0.09	−0.24	−0.01
19	comm. comp.	0.19	0.18	0.18	0.12	0.15	0.11
22	**repeat**	−0.39	−0.14	−0.13	−0.26	−0.19	−0.22
25	**exit**	−0.01	0.06	0.28	0.08	0.22	0.29
26	**for** %	0.42	0.17	0.09	0.02	−0.08	0.23
27	**repeat** %	−0.38	−0.17	−0.23	−0.22	−0.23	−0.25
33	text layout	0.23	0.28	0.15	0.08	0.20	0.15

2) Parameter p of a program relative to the total number of points gained by the author of the program under investigation (k_t). It characterizes the program together with its author. There is a separate value of the coefficient for each problem. The values of the all 6 coefficients and their average are given in Table 6.

3) Average value of the parameter of all programs written by the same author relative to the total number of points gained by their author (k_u). It characterizes mostly the skills and the programming style of the author. The values of the coefficients are given in Table 7.

The correlation coefficients of most parameters are close to zero. E.g., there is no linear dependence of program correctness and that parameter.

Only the coefficients of the parameters, whose absolute values exceed 0.1 for at least in one of the coefficients k_p or k_t are given in the Tables.

8. Interpretation of statistical results. Let us discuss the results given in Tables 4, 5, 6, and 7 from the standpoint of the programming style and programming teaching.

Table 6. Correlation coefficients k_t between program parameter p and the total number of points gained by the author of the program

No	Program parameter	Problem No					
		1	2	3	4	5	6
7	**uses**	−0.35	−0.26	−0.33	−0.43	−0.04	0.13
8	**const**	0.09	0.35	0.10	0.13	−0.01	0.15
13	*longint*	−0.05	−0.11	0.11	0.06	0.28	0.33
15	*string*	−0.31	−0.18	−0.04	−0.03	−0.17	0.01
19	comm. for comp.	0.07	0.19	0.10	0.04	0.14	0.04
22	**repeat**	−0.46	−0.12	−0.05	−0.24	−0.11	−0.28
25	**exit**	0.05	0.01	0.12	−0.10	0.37	0.34
26	**for** %	0.44	0.19	0.05	0.16	−0.15	0.30
27	**repeat** %	−0.44	−0.16	−0.10	−0.25	−0.06	−0.30
33	text layout	0.27	0.29	0.13	0.10	0.09	0.19

Table 7. Correlation coefficients k_u between average parameter p of all programs written by the sane student and the total number of points gained by the author of the program

No	Program parameter	k_u
7	**uses**	−0.41
8	**const**	0.29
13	*longint*	0.25
15	*string*	−0.37
19	comm. for comp.	0.20
22	**repeat**	−0.34
25	**exit**	0.26
26	**for** %	0.13
27	**repeat** %	−0.37
33	text layout	0.22

Correlation coefficients show the existence of some dependence between program correctness and the programming style expressed by some formalized parameters of the program text. However, the dependencies are not very strong or some of them not very stable (the values are changing from problem to problem). So the conclusions may be not very categorical.

We observed that constant definitions (8) and cycles **for** (26) have the most positive correlation coefficients. While module definitions **uses** (7) and cycles **repeat** (22, 27) – the most negative. The constant definitions is an indication of good programming style. Correlation coefficients of other parameters (7, 22, 26, 27) advocate that it is better to rely on simple and well known programming language features (as the cycle **for**) rather than to use advanced but more complicated or less common features, especially if they are not necessary for the problem solution (e.g., **uses**). Maybe for the same reason the usage of objects (18) have some negative effect (see Table 4).

Long integers have positive coefficients. The 16 bit integers are an obvious shortcoming of PC. It may be simply eliminated by declaring long integers.

The statement **goto** has a bad reputation in programming. However, its coefficients trend to zero. The statement was used only by 5.5% of students (Table 3) – very advanced ones (they know what to do with **goto**) and those with moderate scores.

Program texts were not examined by the jury. So the layout of the program text (33) and its readability did not have effect on the score. Students wrote program texts only for themselves. Extra efforts are required to make a program text more readable. This is not feasible in the limited time in the Olympiad. Nevertheless, the program readability has a positive coefficient. This confirms the idea that program readability has a positive impact on its correctness.

We see (Table 3) that the layout of program text (33) and other parameters having positive effect on readability (e.g., comments (28)) are higher of the winners of the silver medals. The same parameters for gold medal winners are lower. This may lead to the hypothesis that those students who have achieved the highest results in the Olympiad have organized the work so that any extra efforts not affecting the score were minimized.

Comparing values of all three coefficients of the sane program parameter we see that their values have a tendency to be ordered $k_p < k_t < k_u$. From this observation we may conclude that usefulness of particular programming language constructs depends on the programmer in greater extent that this on programming problem.

The interpretation of the results is rather of subjective nature. So we end it at this point and leave it for the reader.

9. Acknowledgements. Many thanks to the President of the 6th International Olympiad in Informatics Mr. Yngve Lindberg and especially to the chairman of the Scientific Committee of the Olympiad Mr. Hakan Stromberg who ensured the availability of student programs for investigation. Thanks for Miss Jurate Bulotaite for expert evaluation of program texts. Thanks for all students – the authors of programs.

REFERENCES

6th International Olympiad in Informatics. (1995) *INFO-STAR,* **1**, 14–24.

Stromberg, H. (1994). *International Olympiad in Informatics.* Hanige, Sweden. 3–10 July. Final Report.

Baecker, R.M., and A. Marcus (1992). *Human Factors and Typography for More Readable Programs.* Addison-Wesley.

Received July 1995

G. Grigas received the Ph.D. degree from the Kaunas Polytechnic Institute (Kaunas, Lithuania) in 1970. He heads the Department of Systems Programming at the Institute of Mathematics and Informatics. His research interests include abstract data types, programming methodology and teaching.

PRIKLAUSOMYBĖS TARP PROGRAMŲ TEISINGUMO IR PROGRAMAVIMO STILIAUS TYRIMAS

Gintautas GRIGAS

Pateikiami programų statistinės analizės rezultatai. Tyrimui imamos programos, parašytos Paskalio kalba Šeštojoje tarptautinėje olimpiadoje, vykusioje Švedijoje 1994 m. Analizuojamos 6 uždavinių 686 programos, kurias parašė 135 moksleiviai iš 46 valstybių. Pastebėta nedidelė, bet pastovi priklausomybė tarp programų teisingumo ir tam tikrų programavimo kalbos konstrukcijų vartojimo dažnio bei programos teksto išdėstymo kultūros. Teigiamą įtaką programų teisingumui darė konstantų apibrėžtys, ciklai "for" bei ilgi sveikieji skaičiai (longint), neigiamą – ciklai "repeat", moduliai bei kitos sudėtingesnės kalbos konstrukcijos.