An Analysis of Novice Compilation Behavior using Verificator

Danijel Radosevic and Tihomir Orehovacki
University of Zagreb, Faculty of Organization and Informatics
Pavlinska 2, 42000 Varazdin, Croatia
{danijel.radosevic, tihomir.orehovacki}@foi.hr

Abstract. *Tutor is part of a C++ learning programming interface named Verificator. It explains to students the causes of syntactic and certain logical incorrectness in their programs. In order to be helpful, tutor messages need to be more intuitive and clear in relation to the existing compiler error messages and warnings. The research presented in this paper analyses the frequencies of different kinds of error-related messages in students' programs, as well as their mutual correlations. It has been shown that tutor messages are more strongly related to error messages than to standard compiler warnings.*

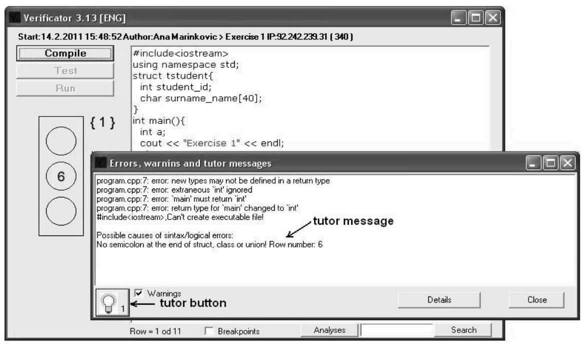
Keywords. Verificator, Tutor, Teaching Programming, Errors and Warnings Analysis

1. Introduction

It has been two years now that the C++ learning programming interface named Verificator was introduced [14]. Verificator helps students in achieving programming skills including good programmer's habits. For that purpose, Verificator disables loading and copy/pasting of program code from outside, as well as sequential program code writing (regardless of program structure). However, beside these restrictions, the purpose of Verificator is also to provide students with appropriate tools. One such tool is the tutor, which helps students in correcting syntax and certain logical errors in a program. The aim of the tutor is to explain possible causes of incorrectness in a program more clearly and in a more intuitive way than standard error messages and warnings do. As a result, instead of standard error messages and warnings, Verificator gives tutor messages, as shown in Figure 1.

This paper presents the results of a research that was performed at the Faculty of Organization and Informatics within the Programming 2 course. Students used Verificator in the development of their C++ programs during lab-based exercises. The main goal of our research was to demonstrate the effectiveness of the tutor in the interpretation of messages related to both syntax errors and warnings that are obtained from the compiler. For that purpose, all error messages, warnings and tutor messages were collected by Verificator. The analysis of the collected data revealed significant correlations between error messages and warnings, as well as error and tutor messages.

The next section of the paper deals with related work concerning different kinds of tutors and debugging aids. Section 3 describes the experiment in which we applied a module for collecting information about novice compilation behavior. The final section contains a brief discusssion and concluding remarks.



Start:14.2.2011 15:48:52 Author:Ana Marinkovic > Exercise 1 IP:92.242.239.31 (340)

```
#include<iostream>
using namespace std;
struct tstudent{
  int student_id;
  char surname_name[40];
}
int main(){
  int a;
  cout << "Exercise 1" << endl;
```

```
program.cpp:7: error: new types may not be defined in a return type
program.cpp:7: error: extraneous `int' ignored
program.cpp:7: error: `main' must return `int'
program.cpp:7: error: return type for `main' changed to `int'
#include<iostream>,Can't create executable file!

Possible causes of sintax/logical errors:
No semicolon at the end of struct, class or union! Row number: 6
```

Figure 1. Verificator with tutor

2. Related work

A programmers' essential goal is the correctness of their code. However, without familiarising oneself with the programming process by means of solving a number of different errors it is almost impossible to develop good programming skills. There are two types of error messages that a compiler provides during the writing and debugging of the programming code.

Syntax errors occur during the compilation process and indicate the violation of syntax rules in C++ statements. In order to help find and correct such errors, the compiler usually provides the line number where the error occurred and/or highlights the line with a different colour. The most common mistakes which cause syntax error messages to emerge are [15][16][20]: Undeclared Variables, Undeclared Functions, Missing Semicolons, Extra Semicolons, Incorrect Number of Braces, Unmatched Parentheses, Unterminated Strings, Left-Hand Side of Assignment does not Contain an L-Value, Value-Returning Function has no Return Statement, Converting Errors (e.g. int* to int**), and Illegal Function Overloading.

Logic (semantic) errors cannot be detected with a compiler and thus the program is executed without error messages. However, some parts of the program will not provide correct results. Uninitialized variables, Setting a variable to an uninitialized value, Using a single equal sign to check equality, Divide by Zero, Forgetting a Break in a Switch Statement, Overstepping Array Boundaries, Misusing the && and || operators, Infinite Loop, Misunderstanding of Operator Precedence, Dangling Else, Off-By-One Error, Code inside a Loop that does not Belong There, and Not Using a Compound Statement When One is Required are the most common causes of this type of error messages [15][16][20]. In cases when A programming solution suffers from logic errors, novices have to find and correct them all by themselves.

In order to facilitate the writing and debugging of the programming code and therewith help develop the skills of tracking and removing the aforementioned errors, a number of desktop and web environments have been proposed. Based on their implementational features, they are decomposed into two different categories: tutors and debugging aids.

The main purpose of the tutor is to guide novices through the entire process of developing programming solutions. J-LATTE [7] is a tutor designed for learning programming in Java which contains several types of constraints: syntax (testing the written code against syntax rules), semantic (checking the accuracy of student solutions) and style (encouraging students to follow good practice when writing a code). Interaction between the tutor and the student is conducted through three types of messages: simple feedback (information about the validity of the submitted solution), hint (advice related to the violated constraint) and all errors (the list of all errors in the code). A similar environment in which students can develop skills of writing code in the Java programming language through eight steps (introduction, item familiarity, item identification, item learning, row familiarity, row identification, row learning and program learning) was introduced by Emurian [4]. Object Oriented Programming System (OOPS) [5] is an online problem-solving environment designed for problem-solving practising (free mode) and assessment of the acquired knowledge (guided mode). During the compilation process, the tutor provides feedback whose proloxity depends on the student progress (low, medium, high).

PAGP [9] is a set of online tutors where novices can learn to analyse a complex problem tasks through three steps: mental visualization of how the program interacts with a user, variable analysis, and flow analysis. PAGP provides feedback related to an error as soon as it appears, which facilitates its detection and removal. From the group of tutors aimed at learning logic programming languages we must extract Prolog Tutor [8], whose implemented features allow the detection of 46 correct and 74 incorrect code implementations along with the identification of the programming style and provide guidelines for correcting errors in the code. A problem-solving tutor (problet) that explains code line by line during a compiling session and thus helps students to learn how to analyze the structure of a program was presented by Kumar [10].

M-PLAT [13] is an intelligent tutoring system for teaching various programming languages. It consists of three basic parts: a theoretical module which contains the theoretical foundations of programming languages, a pedagogical module, which adapts the exercise level to the knowledge and skills that a student possesses based on implemented strategies, and an expert module which evaluates the correctness of student solutions using a number of tests. A notional machine based intelligent system which describes instructions of every action step by step and thus helps students to learn the syntax and semantics of procedural programming languages was proposed by Dadić et al. [2]. C-Tutor [19] is an intelligent tutor designed for both learning the theoretical foundations of the C language and the analysis of program code. Its fundamental segment is the knowledge based program analyzer, which consists of two parts. The first one is GOES, a reverse engineering system that automatically generates a description of the problem from the teacher's sample program. Subsequently, based on the generated description, the didactic system ExBug, as the other part of the tutor, identifies errors in students' solutions.

A problem-solving environment designed for writing programming code and testing its validity is called a debugging aid. The first representative of this group of tools is Program Judgement System [6], an online compiler that using the specifications related to correct and incorrect program codes stored in the database automatically checks the accuracy of student solutions written in programming languages Pascal and C . Apart from offering the facilities for testing and debugging of the code, AnimPascal [17] captures all the activities that student undertakes during task solving. As a result, the identification of students' behavior patterns in all the phases of program code development has been facilitated. E-learning Java Trainer (ELJT) [12] is a web-based system which evaluates the syntactic correctness of the written code with a Java compiler. On the other hand, the semantic code validity is tested through the execution of test data and comparison of the obtained and expected results. An online environment designed for teaching the basics of imperative programming was introduced by Efopoulos et al. [3]. Its compiler provides precise error messages and provides hints for removing run-time errors.

JOSH-online [1] is an integration of tutor and server-based interpreter where students can learn the basics of Java and test their own programming solutions. A similar combination for developing programming skills in C++ was proposed by Shaffer [18]. After comparing the students' submission with the solution that was made by the teacher, the system informs the students about the differences in the code, and gives them two options: to either modifiy the written code or submit their solution for grading.

Although the aforementioned environments have shown a significant level of usefulness, they all suffer from a common limitation. Namely, the majority of them focus on providing the tips related to tracking and correcting syntax errors while the emphasis on logic (semantic) errors is fairly rare. Furthermore, a recent study [21] has shown that in most cases current C++ compilers provide ambiguous and misleading error messages, which makes teaching programming to novices even more demanding. Finally, McLaren et al. [11] concluded that novice students learn much better from a tutor whose feedback is presented politely rather than in a straightforward manner. In order to solve the aftorementioned problems, two extensions of the educational tool Verificator [14] are briefly introduced in this paper. The first extension is the tutor that helps students in identifying and eliminating errors during code writing and testing. The second extension refers to a module that monitors student work and, at the end of a programming session, saves the information related to the frequency of occurrence of syntax errors, warnings and tutor messages.

3. Analysis of C++ error related messages

The research described in this paper was conducted by using the Verificator [14] educational programming interface on a sample of 154 students in Programming 2 course in the academic year 2010/2011. The data about students' compilation behavior (i.e., total number of each particular error message, warning and Verificator's tutor messages during lab-based exercises) were uploaded onto a web server while the students were performing their programming tasks.

The set of messages considered in this paper that were collected during the lab-based exercises is a subset of all possible error related messages. All tutor messages were taken into consideration, but the number of considered errors and warnings was approximated according to the most common categories, as described in [15][16][20]. The set of received messages is a subset of considered messages that appeared during lab-based exercises at least once. Four programming tasks performed by students in their programming lab sessions over 4 weeks were included in the research. The results are as follows:

1) A comparison of considered and received (at least once) messages is shown in Table 1.

Table 1. Considered messages

Message type	Number of messages that were considered	Number of messages that occurred
errors	82	38
warnings	29	7
tutor	19	18

The total number of considered error massages and warnings is much higher than the number of messages that actually occurred. However, tutor messages are intended to help students in their examinations, so almost all of them occurred.

2) The average number of messages per student (for 4 exercises) that occurred is presented in Table 2.

Table 2. Number of messages per student

Messages	Average message types per student (st. dev.)	Average messages per student (st. dev.)
errors	5.71 (3.77)	27.36 (30.57)
warnings	1.27 (0.69)	112.57 (114.02)
tutor	3.97 (2.16)	69.78 (63.22)
Total	10.95 (5.68)	284.03 (171.06)

The average message type per student is the lowest (1.27) for warnings, while their average number of occurrences is the highest. The reason for this is an extreme repeated occurrence of the same warning (*unused variable*), as shown in Table 3b.

3) The most frequent error, warnings and tutor messages per student (average for the 4 exercises) are presented in Table 3a, 3b and 3c, respectively. Moreover, all the three tables show that the most frequent errors and warnings are related to variable declaration and usage.

Table 3a. Most frequent errors per student

ERRORS	Avg. per student	% of total
undeclared	9.18	33.66
expected `;' before	2.79	10.23
: note:	2.50	9.16
expected primary-expression before	2.17	7.96
expected `;' before '}' token	1.36	4.98
has no member named	1.25	4.60
expected `,' or `;' before	1.19	4.38
expected unqualified-id before	0.96	3.52
expected `)' before	0.82	3.02
has not been declared	0.68	2.50
expected declaration before	0.60	2.19
undefined reference to	0.55	2.01
ld returned 1 exit status	0.50	1.83
missing terminating " character	0.42	1.54
no match for	0.31	1.14
: note: candidates are:	0.31	1.13
expected constructor, destructor, or type conversion before	0.29	1.08
request for member	0.28	1.02
error: within this context	0.20	0.73
no matching function for call to	0.15	0.54
stray '\' in program	0.14	0.52
cannot convert	0.10	0.35
expected init-declarator before	0.08	0.30
No such file or directory	0.07	0.27

Table 3b. Most frequent warnings per student

WARNINGS	Avg. per student	% of total
unused variable	105.46	93.99
control reaches end of non-void function	4.21	3.75
unknown escape sequence	2.11	1.88
converting to	0.29	0.26

Table 3c. Most frequent tutor messages per student

TUTOR	Avg. per student	% of total
[<name>] is not referenced.	25.73	36.97
It's not a good practice to define variables within for or while!	19.03	27.35
Semicolon after a control instruction	14.02	20.14
Improper case (no space after case)!	3.26	4.68
Wrong operator in the logical expression: = instead of ==	1.63	2.35
Odd parentheses	1.21	1.73
No matching right curly bracket	1.09	1.57
Namespace is not specified.	0.85	1.22
No semicolon at the end of struct, class or union!	0.69	0.99
<iostream> is not included!	0.66	0.95
main is not specified!	0.47	0.67
Odd quotes!	0.44	0.63
No matching left curly bracket	0.31	0.44

The most frequent tutor warning was related to unused functions, methods, structs and classes (which was not reported by warnings).

4) The most significant correlations between errors and other error-related messages (warnings and tutor) are presented in Table 4. It was found that the error messages are more often correlated with the tutor messages than with the warnings.

Table 4. The most significant correlations between errors and other messages (p<0.01)

Corr.	Error message	Correlated message	Mess. type
0.988	missing terminating " character	Odd parentheses	tutor
0.679	expected declaration before	No matching left curly bracket	tutor
0.491	ld returned 1 exit status	main is not specified!	tutor
0.474	expected primary-expression before	No matching right curly bracket	tutor
0.466	expected unqualified-id before	No matching left curly bracket	tutor
0.422	undeclared	[<name>] is not referenced.	tutor
0.414	expected primary-expression before	It's not a good practice to define variables within for or while!	tutor
0.393	expected `;' before '}' token	It's not a good practice to define variables within for or while!	tutor
0.390	stray '\' in program	Odd parentheses	tutor
0.382	expected primary-expression before	[<name>] is not referenced.	tutor
0.381	undeclared	No matching right curly bracket	tutor
0.359	expected declaration before	It's not a good practice to define variables within for or while!	tutor
0.353	undefined reference to	main is not specified!	tutor
0.346	expected primary-expression before	unused variable	warning
0.340	expected primary-expression before	No matching left curly bracket	tutor
0.336	expected `;' before '}' token	No matching right curly bracket	tutor
0.333	undeclared	No matching left curly bracket	tutor
0.325	undeclared	unused variable	warning
0.318	expected declaration before	No matching right curly bracket	tutor
0.309	undeclared	It's not a good practice to define variables within for or while!	tutor
0.299	expected `;' before '}' token	unused variable	warning
0.299	expected `;' before '}' token	No matching left curly bracket	tutor
0.291	request for member	Odd parentheses	tutor
0.290	two or more data types in declaration of	No semicolon at the end of struct, class or union!	tutor
0.283	expected `;' before '}' token	Odd quotes!	tutor
0.280	expected unqualified-id before	It's not a good practice to define variables within for or while!	tutor
0.271	missing terminating " character	No matching left curly bracket	tutor
0.267	expected unqualified-id before	Odd parentheses	tutor
0.265	expected `;' before '}' token	[<name>] is not referenced.	tutor
0.265	expected declaration before	[<name>] is not referenced.	tutor
0.261	no match for	semicolon after control instruction	tutor
0.260	undeclared	No semicolon at the end of struct, class or union!	tutor
0.257	expected unqualified-id before	No matching right curly bracket	tutor
0.255	cannot convert	Improper case (no space after case)!	tutor

4. Conclusion

One of the most popular features of different educational systems aimed at learning programming is the tutor. A tutor can help in achieving programming skills in a number of different ways. In case of Verificator, the tutor is intended to explain possible incorrectness in a program more clearly and in a more intuitive way than standard error messages and warnings do. The main difference between the presented tutor and other approaches is its focus on logic errors and student's experiences. The research conducted on a sample of 154 students in their lab-based exercises resulted in identifying the most common error messages, warnings and tutor messages, as well as their mutual correlations. Moreover, it was shown that error messages are more often correlated with tutor messages than with compiler warnings. Therefore, we can conclude that the tutor is effective in helping students when performing programming activities.

5. References

[1] Bieg C, Diehl S. Educational and technical design of a Web-based interactive tutorial on programming in Java. Science of Computer Programming 2004, 53(1): 25–36.

[2] Dadić T, Stankov S, Rosić M. Meaningful learning in the Tutoring System for Programming. Proceedings of the 30th International Conference on Information Technology Interfaces (ITI), 2008 June 23-26; Cavtat, Croatia. Zagreb: SRCE University Computing Centre, University of Zagreb; 2008. pp. 483–88.

[3] Efopoulos V, Dagdilelis V, Evangelidis G, Satratzemi M. WIPE: A Programming Environment for Novices. Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, 2005 June 27-29; Monte de Caparica, Portugal. pp. 113-17.

[4] Emurian HH. A programmed instruction tutoring system for Java^{TM}: consideration of learning performance and software self-efficacy. Computers in Human Behavior 2004, 20(3): 423-459.

[5] Gálvez J, Guzmán E, Conejo R. A blended E-learning experience in a course of object oriented programming fundamentals. Knowledge-Based Systems 2009, 22(4): 279–286.

[6] Hattori N, Ishii N. An extended educational system for programming and its evaluation. Information and Software Technology 1999, 41(8): 525–532.

[7] Holland J, Mitrovic A, Martin B. J-LATTE: a Constraint-based Tutor for Java. In: Kong SC et al., editors. Proceedings of the 17th International Conference on Computers in Education; 2009 November 30-December 4; Hong Kong. pp. 142-46.

[8] Hong J. Guided programming and automated error analysis in an intelligent Prolog tutor. International Journal of Human-Computer Studies 2004, 61(4): 505–534.

[9] Jin W. Pre-programming Analysis Tutors Help Students Learn Basic Programming Concepts. Proceedings of the 39th ACM Technical Symposium on Computer Science Education; 2008 March 12–15; Portland, Oregon, USA. New York, pp. 276-280.

[10] Kumar AN. Explanation of step-by-step execution as feedback for problems on program analysis, and its generation in model-based problem-solving tutors. Technology, Instruction, Cognition and Learning 2006, 4(1): 65-107.

[11] McLaren BM, DeLeeuw KE, Mayer RE. A politeness effect in learning with web-based intelligent tutors. International Journal of Human-Computer Studies 2011, 69(1-2): 70–79.

[12] Mendes A, Ivanov V, Marcelino M. A Web-Based System to Support Java Programming Learning. Proceedings of the International Conference on Computer Systems and Technologies; 2005 June 16-17; Varna, Bulgaria. http://ecet.ecs.ru.acad.bg/cst05/ Docs/cp/sIV/IV.4.pdf [2/10/2011]

[13] Núñez A, Fernández J, Garcia JD, Prada L, Carretero J. M-PLAT: Multi-Programming Language Adaptive Tutor. Proceedings of the 8th IEEE International Conference on Advanced Learning Technologies. 2008 July 1-5; Santander, Cantabria, Spain pp. 649-651.

[14] Radošević D, Orehovački T, Lovrenčić A. Verificator: Educational Tool for Learning Programming. Informatics in Education 2009, 8(2): 261-280.

[15] Rhodes G. Common Beginner C++ Programming Mistakes. Valencia Community College. http://fd.valenciacc.edu/file/grhodes4/CommonBeginnerMistakes.pdf [2/11/2011]

[16] Rinker B. Error Messages and Debugging in C++. University of Idaho, Computer Science Department, 2002. http://www2.cs.uidaho.edu/~rinker/cs113/errors.pdf [2/11/2011]

[17] Satratzemi M, Dagdilelis V, Evagelidis G. A system for program visualization and problem-solving path assessment of novice programmers. Proceedings of the 6th annual conference on Innovation and technology in computer science education; 2001 June 25-27; Canterbury, UK, pp. 137-140.

[18] Shaffer SC. Ludwig: An Online Programming Tutoring and Assessment System. ACM SIGCSE Bulletin 2005, 37(2): 56-60.

[19] Song JS, Hahn SH, Tak KY, Kim JH. An Intelligent Tutoring System for Introductory C Language Course. Computers & Education 1997, 28(2): 93-102.

[20] Teorey TJ, Ford AR. Practical Debugging in C++. Prentice Hall; 2001.

[21] Traver VJ. On Compiler Error Messages: What They Say and What They Mean. Advances in Human-Computer Interaction 2010; 2010: 1-26.