3rd International Conference on Advanced Research in Biology, Engineering, Science and Technology
(ICARBEST'16)
Organized by
International Journal of Advanced Research in Biology, Engineering, Science and Technology (IJARBEST)
19th March 2016

Error Free Least Action Programming (LAP) Interface for Novice Programmers

S.Abishek
UG Scholor, Dept. of CSE
Francis Xavier Engineering College
Tirunelveli, India
abisheksam@hotmail.com

C.Kaushik
UG Scholor, Dept. of CSE
Francis Xavier Engineering College
Tirunelveli, India
iamkaushik@mail.com

Dr.D.C.Joy Winnie Wise
Professor and Head, Dept. of CSE
Francis Xavier Engineering College
Tirunelveli, India
joywinniewise@yahoo.com

***Abstract*— Learning to code is a difficult task. Many an instance we find students who are learning to code get discouraged and demotivated by the errors reported in their hard typed code by the compiler making them believe that they are bad programmers at the beginning itself. This is not true in most of the cases. Programming is a fun, creative and challenging task that should focus on the creativity, imagination and critical thinking of the programmer and not on the time spent for error finding, debugging and learning the syntax of code. Many expert programmers may disprove with this idea, but for novice programmers, this Least Action Programming Interface provides a motivating and easy way of learning programming.**

***Keywords*— Learning to Program, Easy Programming, Programming Interface, Novice Programmers.**

I. Introduction

Learning programming is a difficult task. It is observed in many cases that for a programmer to become an expert it takes nearly ten years [1], [2]. The focus of using a programming language is to solve a problem with the help of automation and speed obtained though computers; rather, programming languages are now requiring much dedication and learning of the syntax of the language than to solving the problem at hand. Students who are new to the programming concept are expected to spend a lot of time studying and memorizing the syntax and semantics of the programming language than understanding the features of a language and using it to solve a problem easily and efficiently. It has been shown by Hilburn [4] that this method of learning programming is ineffective. To illustrate on this matter, consider classical problem in which we have to switch the contents of two memory locations without using a third variable. It is done in C as b=a+b; a=b-a; b=b-a;. Whereas in Python, it is simply put as a,b=b,a;. If a student is more focussed on the syntax of the

code, then the time spent for learning and acquiring skill in programming is lost.

Many IDEs that are available today are complex and require more time for studying it for proper and efficient use more than the language itself. Problem solving and fast code generation are not supported [5]. There is a need for an IDE that is simple, error tolerant or helping the programmer to alternate and simple replacements of bad code with understanding the level of programmer's experience.

II. Programming Approach

Crews and Ziegler [3] state that a student who is learning to program must follow a systematic method of solving problems and translating them into a working program. These steps are referred to as the program development lifecycle, a summary of which appears in Table I.

Table I

Step	Task
1	Analyse the problem
2	Design a solution plan
3	Construct an algorithm
4	Implement the algorithm
5	Test and debug the algorithm

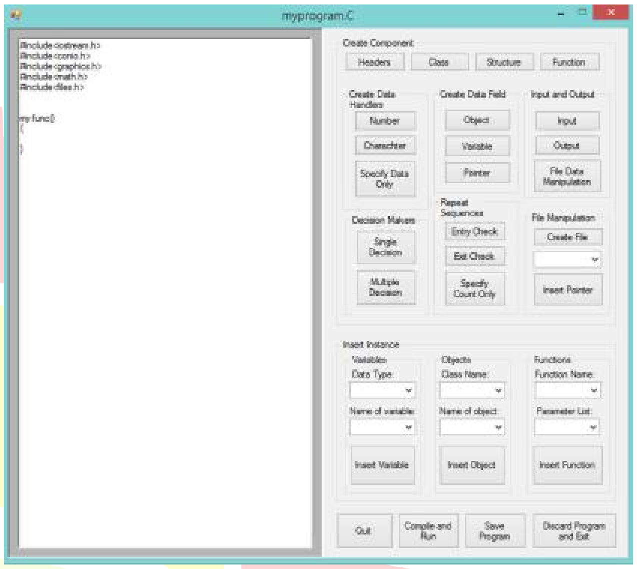
The students who are taught to focus on the analysis and design of a solution to a problem are found to have a better skill at learning new programming languages and techniques [6]. This is evident from the first two stages of solving a problem from Table I.

In the later stages, the students would focus on the language specific aspect of the problem, which would take more time if the task is discouraging or tedious, [8] which is writing the code and debugging the same. For beginners in programming, writing code must be made easy with suggestions and predictions for correct syntax with least work done in the process, so as to motivate them in achieving their goal of successfully compiling and executing their code, [14] in turn solving the problem. Guidance is very important to the program learners, who has to be taught in a simple, effective and precise manner [13] for a particular language.

III. Development of a New Interface

A. Primary Focus

The primary area of concern for the new proposed IDE design is to provide a supporting, non-error denoting and learnable interface that itself easy to understand and use.



```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<files.h>


my func()
{

}
```

Fig. 1 The proposed design of a learnable interface

A simple prototype of the proposed IDE design is shown above, which has two parts, one in which the programmer can edit code like a normal text application called the "editor" and the second part, which contains all the available keywords and features of a particular programming language called a "selector" as shown in Fig. 1 These two parts give a "pick-and-click" interaction which is easy and fast in accessing the contents on screen as suggested by Wilbert [7] on GUI design. This feature is named as Least Action Programming (LAP) Interface.

B. An user friendly design

The Interface is user friendly as in it makes all the features of the language visible on the "selector" section. This allows the novice programmer to seek out and select the part of the program the he wishes to include in his code. There are in fact some constraints that govern the user access to the selector. The user has to select the header files first, followed by the variables and functions. A dynamic enablement of the particular inclusions in the selector section will provide a more comfortable interface.

C. Less Errors, More Learning and Design

As said earlier, the interface is designed to help the beginners in programming, who require a supportive and error free code compliable environment [10]. The interface promotes such demands by giving a date selective window that prompts the user for input to structure of the code as shown in Fig. 2. The figure shows prompt to function characteristics that are modifiable by the user with ease. This learning of features of the language is designed as said by Boyle [12].

D. Help for all the features in the language

It is obvious that a novice programmer will have confusions and doubts when learning a new language, [9] even in the code that he has copied from other sources without reading it prior to execution. The interface provides help text in the most elemental and simplest way possible with graphical references and examples that the user can access and learn from any time.

E. Fast accessing of features

All the features of the language are made readily available for the user to select, customize and insert into the editor from the selector part. This improves the programming speed and reduces the time spent in evaluating and correcting the syntax errors present in the code generated by novice programmers. The user who is having a basic idea about the features of the language under commencement would find this readily available and accessible contents utmost useful for least action imposed programming. For the users who have less or no experience in a language, help is provided as shown in Fig.2 following the methodology suggested by Jacobson [11] in simple and detailed explanations of the features of the language.

This reduced effort put in code generation give the name Least Action in LAP Interface.

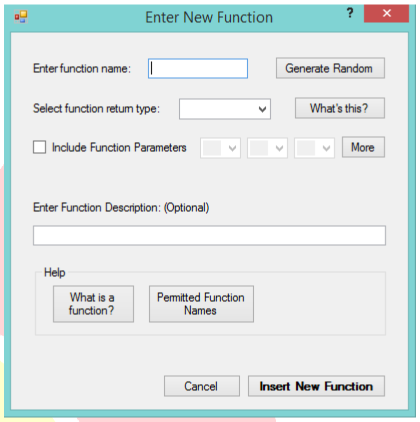


Fig. 2 Prompt for designing a function

IV. COMPARISON OF DESIGNS

The newly proposed LAP Interface was implemented in a simple prototype and given to students in the Computer Science stream to test out the usefulness of design. After the results are gathered, the same batch of students were given an IDE that is used for the same language for comparison. Most of the students reported that the old IDEs [15] lacked the aspect of helping out the programmer when coding as it only shows the errors with their line number and not the alternate solution.

Further evaluation of this case revealed that the LAP Interface was more easily navigable and creating code from scratch was not that much of a hard task for the beginners themselves. Fig. 3 shows the comparison between LAP Interface and conventional IDEs over various parameters.

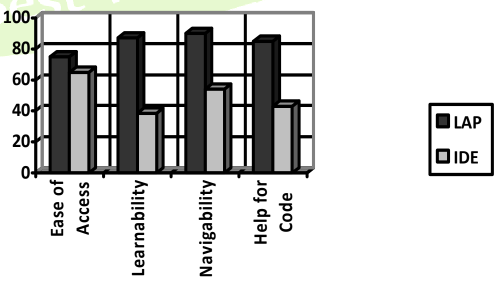


Fig. 3 Graph showing comparisons between LAP Interface and IDE

V. Conclusion

The Least Action Programming Interface is designed in a way that it will be useful to novice programmers and slow learners in computers. The results obtained from prototype trials have been promising, giving rise to a new era of programmable interfaces, the ones with high usability and error tolerance, guidance to the learning programmers maintaining the integrity of the code at all times.

We believe that this new type of approach in providing the learners of programming will create more enthusiastic, creative and knowledgeable programmers in the future.

References

[1] L. E.Winslow, "Programming Pedagogy—A Psychological Overview," ACM SIGCSE Bulletin, vol. 28, no. 3, pp. 17–22, Sep. 1996.

[2] A.Robins, J.Rountree, and N.Rountree, "Learning and Teaching Programming: A Review and Discussion," Computer Science Education, vol. 13, no. 2, pp. 137–172, Mar. 2003.

[3] Crews, T and Ziegler, U. "The flowchart interpreter for introductory programming courses", Proceeding of the Frontiers in Education 1998 Conference. Tempe, Arizona, USA, 1996.

[4] Hilburn, T.B. (1993). A top-down approach to teaching an introductory computer science course. 24th SIGSCE Technical Symposium of Computer Science Education. Indianapolis, USA, 1993.

[5] Calloni, B.A. and Bagert, D.J. "*Iconic Programming proves effective for teaching the first year programming sequence*". Proceedings of the 28th SIGCSA Technical Symposium on Computer Science Education, 1997.

[6] Prolux, V.K. (1996). *Foundations of Computer Science: What are they and how do we teach them?* Annual Joint Conference Integrating Technology into Science Education, 1996.

[7] Wilbert O. Galitz, *The Essential Guide to User Interface Design*, 2nd Edition, Wiley Publications 2012.

[8] Byrne, P. and Lyons, G. 2001. "*The effect of student attributes on success in programming*." SIGCSE Bulletin, Vol. 33 (3), pp. 49-52.

[9] Santos, A., Gomes, A. and Mendes, A. J. 2010. "*Integrating New Technologies and Existing Tools to Promote Programming Learning*." Algorithms, Vol. 3 (2), pp. 183-196.

[10] Pattis, R. *Karel the Robot: A Gentle Introduction to the Art of Programming*. John Wiley & Sons, 1981.

[11] Jacobsen, David, Eggen, Paul, and Kauchak, Don. (1993). *Methods for Teaching: A Skills Approach*. 4th ed.

[12] Boyle, Tom. (1997). *Design for Multimedia Learning*. Prentice Hall Europe.

[13] P. A. Kirschner, J. Sweller, and R. E. Clark, "Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experimental, and InquiryBased Teaching," Educational, Psychologist, vol. 41, no. 2, pp. 75–86, 2006.

[14] A. Robins, J. Rountree, and N. Rountree, "Learning and Teaching Programming: A Review and Discussion," *Computer Science Education*, vol. 13, no. 2, pp. 137–172, Mar. 2003.

[15] R. C. Holt, D. B. Wortman, D. T. Barnard, and J. R. Cordy, "SP/k: A System for Teaching Computer Programming," *Communications of the ACM*, vol. 20, no. 5, pp. 301–309, Apr. 1977.