South Asian Studies
A Research Journal of South Asian Studies
Vol. 31, No. 1, January – June 2016, pp. 305 – 318

A Strategy for the Promotion of Computer Programming Using Urdu Language in Pakistan

Kamran Abid
University of the Punjab, Lahore.
Adnan Abid
University of Managemengt and Technolgy Punjab, Lahore.
Muhammad Shoaib Farooq
University of Managemengt and Technolgy Punjab, Lahore.
Uzma Farooq
University of Managemengt and Technolgy Punjab, Lahore.
Ansar Abbas
University of Managemengt and Technolgy Punjab, Lahore.

ABSTRACT

Software industry has proven to be a stepping stone towards changing the rank of a country in the comity of nations. In South Asian region, India has immensely excelled her economic growth by increasing its revenues with the help of software export. The development of software involves man power with technical support, where the premier focus in on computer programming. Therefore, producing a large number of skillful computer programmers in Pakistan would certainly help the cause of establishing software houses, which in turn, will attract the western world to outsource their software projects to Pakistan. Like India, this can certainly act like a game changer for Pakistan's economy by earning a huge revenue.

In this research we have presented a methodology to increase the interest of Pakistani people in computer programming by providing a bilingual computer program development environment in Urdu and English languages. This act may certainly open new dimensions of teaching computer programming, for instance, by introducing computer programming at school level. Furthermore, it may increase the interest of students and teachers to learn and teach computer programming while experiencing to program in their national language. Lastly, in technical terms, we have presented the visual design of such bilingual environment along with architectural modification in the preprocessor for C++ language to support both Urdu and English languages for writing computer programs.

Key Words: computer programming; Urdu language; programming in regional languages; computer programming in Urdu.

Introduction

Every day computer science community faces new challenges regarding finding out the solution of the problems faced by mankind. In order to solve such challenging problems, programmers take support of powerful programming languages that are designed to address the needs of a programmer (Verkroost & Eliens, 2013). Along with the programming languages comes their programming environment. These programming environments are created for professional use and consist of many built in tools such as debugger, project management etc. Ironically, majority of these programming languages are created to program the computer in order to solve the problems.

It has been a research area in the computer science community that there should be a programming language combined with a programming environment that can be used for teaching purposes, for teaching novice students who are new to the computer science and the programming world (Felleison 1998) (Farooq et al. 2014). The basic problem with the programming languages is that, they have their own complex syntax, constructs, error messages and unexpected outputs, which are difficult to understand for novice programmers (Smith et al., 2010). (Farooq et al. 2015). Moreover, In order to teach them, the instructor needs a deep knowledge of computer internal architecture and working mechanisms. The reason behind these facts is that these languages are not designed for students to learn programming, rather they have been designed to solve the problems.

In order to cope with this problem, instructors have created their own subsets of the programming language (Depasquale 2002)(Farooq et al. 2015). These subsets consist of some of the very basic constructs of the programming language. The instructor teaches his created subset and by the time the students get familiar with the programming, the instructor adds more items to his subset.

With the help of language subsetting, the instructors do manage to teach the existing programming language to a novice programmer up to some extent, but unfortunately the same technique cannot be applied to the programming environment. The instructor cannot add, remove, or modify any component of the programming environment, so their students have to code in the fully professional programming environment that seems complex and messy to a novice programmer (Farooq et al. 2015).

But the question is, are we expecting a novice programmer to build a professional and reliable software application using all the tools that comes with the environment? Also very few of these tools come with the language pack facility, whereas majority of the tools uses English as their primary interface language (Kurland et al. 1986). This makes it even more difficult for non-English speaking novice students around the globe to get familiar with the tool.

The aim of this research is to focus on defining a subset of a programming language that is helpful to the teachers and students for pedagogical and cognitive activities. Furthermore, we intend to develop a programming environment that can be totally customized according to the needs and understanding level of the novice student.

Programming environment should be very user friendly, interactive and should be according to the locale/region. To this end, we intend to augment the programming environment to not only support writing code in English but also allow the users write code in Urdu language, which is the national language of Pakistan. The syntax of the programming language should also be more readable, less complex, and should gradually improve the learning curve of the student by improvising according to the level of the student. With this kind of programming language student can focus more towards logic and solution building rather than worry about the syntax of the language.

If programming languages and their environments are designed in the local speaking language, then novice students can learn programming and develop their familiarity with the programming environment tool in a better and more understanding way. The chances of understanding a problem increases when it is explained in your own language. If programming language and their programming environments are designed by keeping in mind the caliber and understanding level of the students, then students can grasp the concepts of programming much quicker and in more effective way.

This above mentioned technique would also help instructors to teach programming in an easier, effective and in an interesting way. And once the students grasp the basic concepts of the programming and also get themselves familiarized with the basic functionalities of the IDE, the instructors can easily shift their students, by putting minimum efforts, towards the use of professional IDE's and industry popular programming languages.

In this research, we intend to define an easy to learn subset of C++ which should be more conformant to the evaluation framework defined in (Farooq et al. 2014). Apart from this, we also aim to design and develop a prototype of a user friendly bi-lingual IDE with the support of Urdu and English languages that allows the novices to write code using the defined subset of C++ in both languages. We strongly believe that this will help the novice programmers learn computer programming with ease.

Background and Related Work

GreenFoot (Hensiksen 2004) a programming language, comes along with its highly interactive programming environment. It is designed to teach the concepts of object oriented paradigms to novice students. Like GreenFoot there are other programming languages such as Microsoft Small Basic (SmallBasic 2014) to teach programming to secondary school students by making graphical and multimedia applications using GUIs and Logo (Pea 1087) to teach programming to primary school students using visual components. The problem with all these is that they are all less relevant to the course curriculum. They distract the attention of the students from the core issue of problem solving and they have their own syntax which is not close to the professional programming language such as C++, Java etc.

Karel, The Robot (Pattis 1981) is a tool that provides visual simulation of a program, followed by its update version Karel++ that brought Karel into object oriented age. The problem with the Karel is that its code syntax is very much difficult and does not closely relate with any of the modern programming language.

Alice (Cooper 2000) is another programming environment that focuses on teaching object oriented skills by creating a 3-D virtual environment. Alice allows students to write simple scripts to create, control and to simulate the 3-D models of objects (e.g. cars) in a 3-D environment. The problem with Alice is that it focuses more on graphics and event driven programming that distracts students from the actual real life problems and the basic programming constructs used to solve that problems. Furthermore Alice does not familiarize its students with the programming language syntax and also doesn't familiarize towards the use of a professional IDE.

CS1 Sandbox project (Depasquale 2002) is a C programming development made for beginners. It also offers the instructors to make subsets of the programming language which in turn are applied on the programming environment. CS1 is very much alike to our functionality but doesn't provide any multilingual support. All the students whether novice or intermediate are to program in C++ language, which can be helpful for an intermediate student at, but for novice student with zero experience of programming, it would be wise to introduce the programming concepts in less formal way and quite easy to use syntax. Furthermore the CS1 programming environment is also no multilingual support and no event driven or automated programming facility.

Some research has been conducted to design truly educational programming environments (Jimenez 2000) and the results reveal the attributes that a programming environment must have in order to become a truly educational programming environment.

Motivation for Bilingual Programming Environment

It is a fact that most of the available programming languages use the keywords of the English language. There are several reasons for this, including the fact that most of the programming languages have been developed in English speaking countries including USA, UK, Australia, and Canada. Therefore, these languages have been developed using the English language. Similarly, some other languages which, although were not developed in English speaking countries e.g. Python in Netherland, Ruby in Japan, are also in English language to present them to the international audience. Lastly, some languages are influence by other languages which were based on the English language, so they also use English keywords.

However, at the same time there has been some effort in developing languages which use keywords from non-English languages, e.g. Dolittle, Easy etc. have been developed using keywords in Japanese and Chinese. But, these languages have not been able to get any appreciation at international level, due to the following reasons. Firstly, such languages have been created with the objective to support native language speakers. Secondly, their syntax does not conform to the requirements of professional imperative and functional languages.

Another approach has been used to create non-English programming languages is to come up with non-English versions of any widely used existing language(Hindawi, 2015). For instance, considering the widely used FPLs, it has been observed that many non-English variants of these languages exist. Table 1 shows such variants for many different FPLs.

Table 1 Non English based Tools

Tool Title	Supported Language	Based on
Chinese C++	Chinese	C++
Hindawi Programming language	Indian	C++
Jeem	Arabic	C++
Phoenix	Arabic	C++
Farsinet	Persian	C#
Hindi Programming language	Hindi	C#
Kumir	Russian	Pascal
LSE	French	Pascal
ZhPy	Chinese	Python
Perunis	Russian	Python

Expect outcomes of the Proposed Bilingual Programming Environment

This research aims to address the problems faced by the students of different degree programs in learning computer programming. This work aims to help the students with the provision of a customized subset of C++ language that is more conformant to the evaluation framework for a first programming language. Furthermore, we also aim to provide a user friendly and bi-lingual IDE to help the programmers learn computer programming in Urdu and English languages. To this end, we would like to incorporate one of our existing published work to define a pedagogically effective subset of C++. The user friendly IDE will enforce the student write simple programs using simpler version of C++, with the support of Urdu and English, and will certainly help the students learn computer programming with more interest.

Proposed Integrated Development Environment

This section presents a tool which is a prototype implementation of the subset of C++ language. This tool not only eliminates the use of the unwanted constructs from C++, but also enforces the desired quality coding standards. The tool helps the programmers write code in an editor, or alternatively allows the programmer to write codes using template dialog boxes for different constructs. The tool automatically generates syntax error free code through the values entered in the fields of the dialog boxes.

Subsetting and preprocessing are two main operations, which help in eliminating some constructs of C++, and help enforcing the prescribed usage of certain other constructs of the language, respectively. However, it is pertinent to note that the implementation of both these operations is non-trivial and requires changes in the pre-processor of the language.

Supported Constructs

We focus on the programming constructs used in the CS1 course outline. It is pertinent to mention here that these short listed constructs facilitate the programmers to write simple programs in the main block. We have extracted these constructs using the guidelines provided in (Farooq et al. 2014).We have included variable declarations, arithmetic and Boolean expressions, and assignment statements. Furthermore, we have included the *if* statement and have dropped the switch statement, as the if statement is more comprehensive as compared to the switch statement. The while loop has been selected from the available loops, as it is closer in syntax with the *if* statement, and hence, is easy to learn. Lastly, from the comments we have chosen end of line comments, as they are clear and less prone to error, particularly for the novice programmers. Certainly, C++ is an object oriented language, but the scope of CS1 courses in imperative first approaches is restricted to basic programming constructs which do not include any object oriented features.

In order to limit the language constructs for the IDE we have included those constructs of C++ language which cover the scope of an introductory course in computer programming. This includes the constructs presented in Table 2.

Table 2 Selected Constructs for Programming

Concepts	**Language Constructs**
main block	main function
Variables	Declaration, initialization
Operators	Arithmetic and Relational operators
Conditional Statement	*if* statement
Iterations	*while* loop
Input/Output Statements	Console input and output
Comments	Inline comments

Additional Constraints on Supported Constructs

The tool implements all the conformance requirements set forth in (Farooq et al, 2014). for the improvement in C++ language to make it a better FPL, on the above mentioned language constructs. It eliminates the usage of for and do/while loops, and does not allow block and mega comments. Thus, this tool fully supports feature uniformity with feature exclusiveness and does not offer any feature multiplicity. Furthermore, it enforces the following recommended practices through strict preprocessing:

Every variable should be initialized explicitly by the programmer or implicitly by environment.

Variable name has been unique with parent as well as within child scope, so here is no chance to scope overriding

Support less effort for writing simple programs by just selecting program basic Skeleton without including core libraries.

It also provides support for writing basic input and output statement without writing single line of code.

Basic C++ naming conventions have been supported and some constraints such as variable names are case insensitive and declaration of the variable using template is also supported by ensuring these rules.

Pretty printer has also been supported by environment for neat and clean properly indented code.

Novice programming Environment support program editing in two ways i) writes a program explicitly by key punching in program editor ii) writing program implicitly through templates without writing any code implicitly with key punch. This process ensures writing program without any syntax error.

Environment also supports bilingual programming by non-English programmers. She can write code in her own selected language. She has also option to convert her code written in native language into English language C++ code.

The program can be compiled with simple compile option through toolbar, menu or shortcut key.

Similarly, program can be executed with a simple run option through toolbar, menu or shortcut key.

It also generates simple and easily understandable error messages so that even a novice user can easily identify the problem in the code. The environment provides the facility to the user to modify the error messages according to their own understanding.

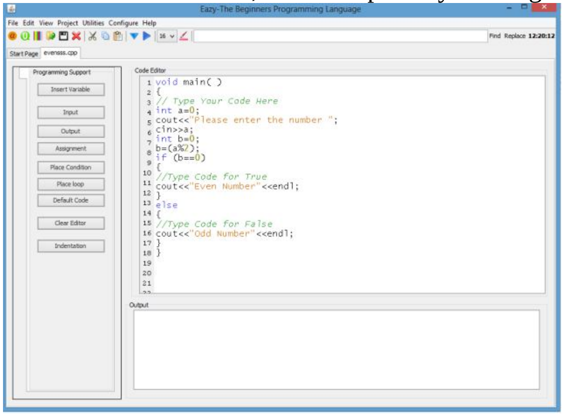
The students can benefit from already provided sample example codes, and can also add sample codes as an example.

Eazy Programming Environment

As a prototype implementation of this research work a programming environment named Eazy has been developed. Eazy only allows to write programs using a defined subset of C++ language, and enforces strict coding rules. The subsetting helps simplify the learning curve for the novices, while the enforcement of strict rules helps them writing unambiguous and good quality codes, thus avoids accidental mistakes.

Main Code Editing Options

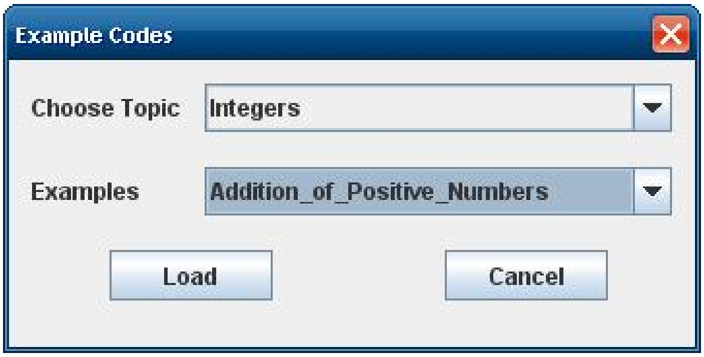
The figure 1 shows the screenshot of the main user interface for novice programmer where a text editor is provided with the support of many different templates on the left side for generating code for various language constructs. The programmer can either write a code in the code editor, or may use any of the templates. This prototype implementation provides the templates for variable declaration, console input/output, assignment operator, if condition, and while loop. The figure also shows how the IDE helps the novice programmer in writing a Boolean condition in the *if* statement using a template. The novice programmer can save her code on hard disk in a file, and can open any existing file as well.



```
void main( )
{
// Type Your Code Here
int a=0;
cout<<"Please enter the number ";
cin>>a;
int b=0;
b=(a%2);
if (b==0)
{
//Type Code for True
cout<<"Even Number"<<endl;
}
else
{
//Type Code for False
cout<<"Odd Number"<<endl;
}
}
```

Additional Features

The Eazy IDE also gives the provisions of adding a default code which adds the default code for main function in the editor. Similarly, it also helps in clearing the editor so as to write a new code, it offers the support of indentation to beautify the presentation of the code in the text editor. Likewise, the text editor includes pretty printer which gives different colors for comments, keywords, and message strings. The sample code written in Figure 1 highlights the functionality of pretty printer



and code indenter. Furthermore, it also provides example codes to make the learning easier for the novice programmers. The student can load different example codes on different topics, e.g. Figure 2 shows the screenshot where a student is choosing an example from the topic Integers.

Another important additional feature in Eazy is the provision of simpler error messages to the novice programmers, which would certainly help them understand their mistakes easily. Table 3 shows the conventional error messages generated by traditional compilers, and the corresponding simplified error messaged which are presented by Eazy to the novice programmers. These simpler error messages are less technical and help the novice programmer to find errors and correct their code.

Table 3: Simple Error Messages for Novices

Code	C++ Error Messages/Warnings	Eazy Error Messages

void main( ){
  cout<< “Hello World”;
	Unexpected End of File found	} is Missing

main( ){
  cout<< “Hello World”;
}
	‘main' : function should return a value; 'void' return type assumed	void is missing in main

void min( ){
  cout<< “123”;
}
	unresolved external symbol _main	'main' is missing or not properly used

void main( ){
  int x=0
}
	Missing ‘;’	';' is missing or not properly used

void main( ){
  int x=7;
  if(x>5){
    cout<< “hi”;
  else {
    cout<< “bye”;
  }
}
	illegal else without matching if
	fatal error C1004: unexpected end of file found
		} is Missing in If Statement

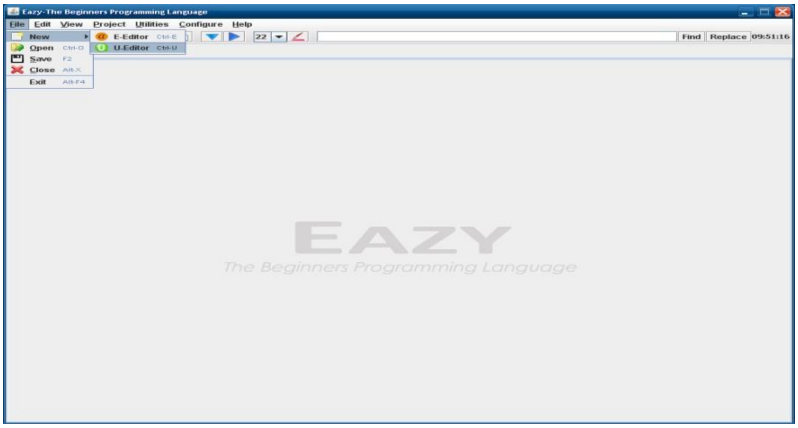
void main( ){
  int x=;
}
	syntax error : ';'	Variable value is missing

if(2+3){
cout<< “true”
}else{
cout<< “false”;
}
	No Error	Arithmetic Operator (+,-,*,/) is not allowed here

Edit, Compile, and Execute Options

Eazy is also equipped with the support of conventional menu options, toolbars, and shortcuts, just like any traditional code editor. The programmers can use any of the three ways to invoke a certain functionality. The editor exposes rich editing options available in traditional text editors, including the settings for font color, size; cut, copy, paste; and find and replace options. Furthermore, the user interface also supports multi-tab editors in which the user can open multiple files at a time, and can perform editing on multiple open programs. The user interface also provides separate buttons and shortcuts to compile and execute the code. Figure 3 shows the main menu bar and ribbon that display the above mentioned features.

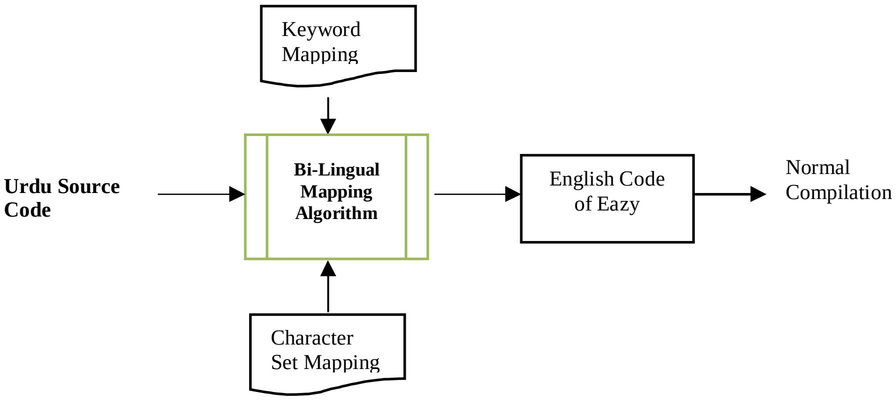
Similarly, the output of a program will be in the form of some console application, which is terminated after execution, in order to hold the output on screen, the pre-processor auto-detects the end of main function and adds the necessary code to hold the output on the screen.



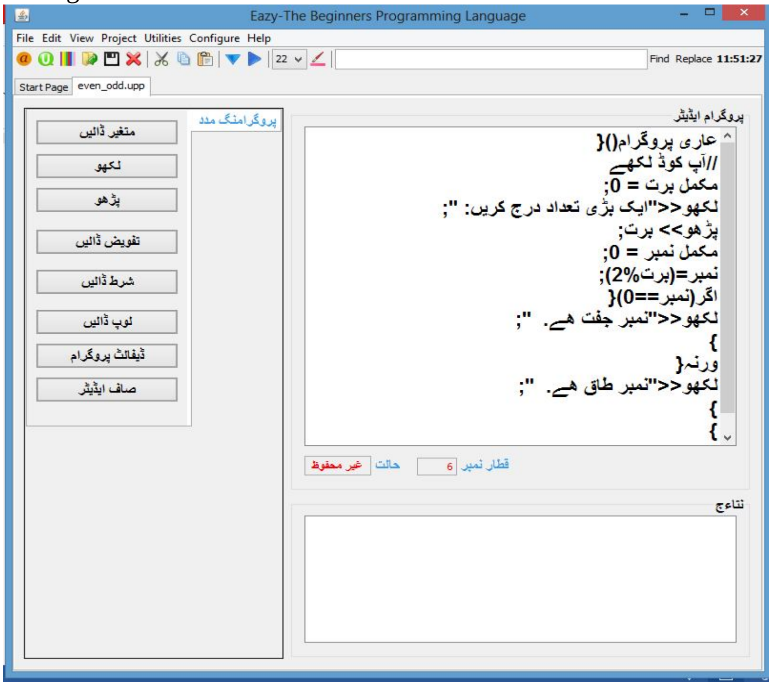
Incorporating Urdu Language in the IDE

In this work, we have developed a programming language which is based on C++, but certainly on the improved C++ that we obtained in (Farooq et al.,2014). We argue that by defining a mapping between the English keywords of C++ to the Urdu keywords can help us writing code in Urdu. In such a scenario, the only requirement is to map the code written in Urdu to its corresponding English version, and then compile and execute the English version of the code. Whereas, the error messages and output can also be shown in Urdu language.

The inclusion of Urdu in the IDE requires modification in the preprocessor of the language. Figure 4 shows the flow of the execution of the Urdu source code to its compiled output. The Urdu source code is converted into corresponding English source code using the bilingual mapping algorithm which uses keyword mapping, as well as, character set mapping files (Humayoun, 2007). This, in turn, produces the corresponding English language code of the input program. Now, this code can be compiled with any existing standard compiler for C++ language. However, the error/success messages are obtained from the compiler and they are again translated into corresponding Urdu messages, before they are reported to the programmer.



Environment also supports Urdu program editing in the same way as in English, i.e. i) write program explicitly by key punching in program editor ii) writing program implicitly through templates without writing any code implicitly by key punch. Second method ensures writing program without any syntax error as shown in Figure 5.



```
عاری پروگرام(){
//آپ کوڈ لکھے
مکمل برت = 0;
لکھو<<"ایک بڑی تعداد درج کریں: ";
پڑھو>> برت;
مکمل نمبر = 0;
نمبر=(برت%2);
اگر(نمبر==0){
لکھو<<"نمبر جفت ہے. ";
}
ورنہ{
لکھو<<"نمبر طاق ہے. ";
}
}
```

Figure 5 Urdu Programming Environment

To this end, the translation requires the following two major ingredients:

i) Keyword Mapping

ii) Character set file

Table 4 shows English to Urdu keyword mapping given in (khan 2011), whereas Table 5 shows the mapping of English character set onto the Urdu character set.

Table 4 keyword Mapping Table

Sr.No.	English	Urdu
1	void	عاری
2	main	پروگرام
3	int	مکمل
4	if	اگر
5	else	ورنہ
6	cout	لکھو
7	cin	پڑھو
8	while	جبتک

Table 5 Character set mapping File

Sr.No.	English (Small)	Urdu	English (Capital)	Urdu
1	a	ا	A	آ
2	b	ب	B	ب
3	c	چ	C	ث
4	d	د	D	ڈ
5	e	ع	E	ع
6	f	ف	F	ف
7	g	گ	G	غ
8	h	ھ	H	ح
9	i	ی	I	ا
10	j	ج	J	ض
11	k	ک	K	خ
12	l	ل	L	ل
13	m	م	M	ا
14	n	ن	N	ں
15	o	ہ	O	ۃ
16	p	پ	P	ٍ
17	q	ق	Q	ق
18	r	ر	R	ڑ
19	s	س	S	ص
20	t	ت	T	ٹ
21	u	ء	U	ء
22	v	ط	V	ظ
23	w	و	W	و
24	x	ش	X	ژ
25	y	ے	Y	ے
26	z	ز	Z	ذ

Conclusion and Future Directions

In this research we have argued to teach computer programming to Pakistani students in their national language. We have presented some existing tools and variants of programming languages which have been developed in languages other than English. In our neighboring countries Indians and Chinese people have developed non-English programming environments in their national languages Hindi and Chinese. But no such environment exists for Pakistani students to study in Urdu. To this end, we have presented the need of a bilingual programming environment. Our proposed environment supports the users to write computer programs in English as well as Urdu language.

We have also identified a subset of C++ programming language that should be used in the first course of computer programming. We also apply constraints on the selected constructs so as to minimize the learning curve for the programmers. We have presented initial design and mapping from English to Urdu language along with the visual user interface for our proposed IDE. Furthermore, in technical terms, we have highlighted the modification in the existing preprocessor of the language to incorporate Urdu language for coding.

In future, we intend to develop this bilingual programming environment to materialize the idea and theoretical concepts presented in this research work. We also plan to evaluate the effectiveness of this new IDE by performing empirical analysis on the developed IDE, so as to figure out the impact that inclusion of Urdu language in the IDE has created in order to produce more apt computer programmers.

References

Abid, A., Farooq, M. S., Farooq, U., Abid, K., & Shafiq, M.(2015) A Strategy for the Design of Introductory Computer Programming Course in High School. Journal of Elementary Education, 25(1), 145-165.

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. In Journal of Computing Sciences in Colleges (Vol. 15, No. 5, pp. 107-116). Consortium for Computing Sciences in Colleges.

DePasquale, P. (2002). Subsetting language elements in novice programming environments. In Proceedings of the RESOLVE Workshop (pp. 108-111).

Farooq M.S, Khan S.A, Ahmad F, Islam S, Abid A (2014) An Evaluation Framework and Comparative Analysis of the Widely Used First Programming Languages. PLoS ONE 9(2): e88941. doi:10.1371/journal.pone.0088941.

Farooq, M. S., Abid, A., Khan, S. A., Naeem, M. A., Farooq, A., Abid, K. (2012). A Qualitative Framework for Introducing Programming Language at High School, Journal of Quality and Technology Management, Punjab University, Pakistan. 8(2).

Farooq, M. S., Khan, S. A., Abid, K., Ahmad, F., Naeem, M. A., Shafiq, M., & Abid A. (2015). Taxonomy and design considerations for comments in programming languages: a quality perspective. Journal of Quality and Technology Management, Punjab University, Pakistan. 10(2).

Farooq. M.S, Khan S.F, Ahmed. F, Islam. S, Abid A, (2014) Choice of Pedagogical Approaches towards First Programming Languages, J. Appl. Environ. Biol. Sci, vol 4(7), pp:311-317..

Felleisen, M., Findler, R. B., Flatt, M., & Krishnamurthi, S. (1998). The DrScheme project: an overview. ACM Sigplan Notices, 33(6), 17-23.

Henriksen, P., & Kölling, M. (2004). Greenfoot: combining object visualisation with interaction. In Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications (pp. 73-82). ACM.

Hindawi Programming System, http://hindawi.in/, Accessed 10 October,2015.
Humayoun, M., Hammarström, H., & Ranta, A. (2007). Implementing Urdu Grammar as Open Source Software. Corpus, 1, 23-696.
Jiménez-Peris, R., et al.(2000). Towards truly educational programming environments. In Computer science education in the 21st century (pp. 81-111). Springer New York.
Khan, A. U., Ayyub, K., & Khan, H. F. U: (2011) A Computer Programming Language in Urdu.
Kurland, D. M., Pea, R. D., Clement, C., & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school students. Journal of Educational Computing Research, 2(4), 429-458.
Pattis, R. E. (1981). Karel the robot: a gentle introduction to the art of programming. John Wiley & Sons, Inc..
Pea, R. D. (1987). Logo programming and problem solving.
Small Basic 1.0 is here! - Small Basic - Site Home - MSDN Blogs. Blogs.msdn.com. 2011-07-12. Retrieved 2014-03-08.
Smith D., Lameras, P., Moumoutzis, N. (2010). Using educational programming language to enhance teaching in computer science" Edge conference.
Verkroost, Y., & Eliens, a. (2013). Seriousify and prettify our educational system!.

Biographical Note

Dr. Kamran Abid Assistant Professor, Punjab University College of Information Technology, University of the Punjab, Lahore, Pakistan.

Dr. Adnan Abid Associate Professor and Director Academics, Department of Computer Science, University of Management and Technology, Lahore, Pakistan.

Dr. Muhammad Shoaib Farooq Associate Professor and Director Graduate Studies, Department of Computer Science, University of Management and Technology, Lahore, Pakistan.

Uzma Farooq Assistant Professor, Department of Computer Science, University of Management and Technology, Lahore, Pakistan.

Ansar Abbas is a post-graduate student at University of Management and Technology, Lahore, Pakistan.