Asian Journal of Computer and Information Systems (ISSN: 2321 – 5658)
Volume 02 – Issue 06, December 2014

Programming Nation – A Learning and Teaching Programming Club

Foad Motalebi

Curtin University, Sarawak
Miri, Malaysia
Email: foad.m {at} curtin.edu.my

ABSTRACT— ***Learning and teaching programming would be beneficial to University students from any field. This is the motivation to establish a programming learning club at Curtin University, Sarawak, Malaysia. An advantage of a programming club is that participation in it would be motivated through the bonafide desire to learn rather than an extrinsic motivation . The club's constitution will be formed. The main idea behind this club will be to learn programming through learning and teaching, believing in the concept that one can learn more when one teaches. The club's instructors will facilitate the first training sessions. After finishing a programming module, a student will be encouraged to facilitate another training of the same module for a new set of students. Before a student can facilitate training, he or she has to add one page of material to the 'Lesson Notes'. After the training, the student facilitator, if so chooses can get feedback about his or her training from his or her students. Students will also be encouraged to form program development teams. A report at the end of the year will show the total classes taken and the success of the club's initiatives. The report would check if the computer programming knowledge of individual members has improved and it would check if students have started developing programs.***

Keywords— Programming Club, Learning and Teaching Programming, Programming Facilitator

1. INTRODUCTION

Modern era consists of a host of programming languages, which in their own way contribute substantially to technology and content development. However learning and mastering them can be a heavy task particularly if the practitioner is not a technology professional. Indeed this task is more difficult when a practitioner is a novice and wants to learn more than a couple of languages to develop content. Though there is an explosion of web based programs and apps, these are mainly developed by technology professionals. It would be beneficial to come out with means to increase the overall manpower of programmers and content developers.

The main goal of this paper is to delineate the idea of a programming club called 'Programming Nation' which is to be set up at an Australian offshore University and also to charter its constitution to the effect that it may help learning of programming languages and through that create practicing programmers and content developers. The paper also looks into ideas in designing modules and selecting languages for the club.

2. A LEARNING AND TEACHING CLUB

Establishing a club to teach programming is motivated mainly through the desire to share skills and knowledge to a receptive audience who are voluntarily eager to learn. There is a recognized need to broaden participation in computing. Mentoring, revised curricula, tool development, outreach programs, and programming courses for non-majors are some efforts. One area that has received surprisingly little attention is the learning of programming in community technology centers that offer free access on a daily basis (Adams, 2007). Another advantage of a programming club is that participation in it would be motivated through the bonafide desire to learn. In these venues, the learning of programming is by choice, meaning that what, when, and for how long programming takes place is at the discretion of the learner rather

than part of a required curriculum (Maloney et all. 2008). This learning system can make programming more accessible for novices by simplifying the mechanics of programming, by providing support for learners, and by providing students with motivation to learn to program (Kelleher & Pausch, 2005).
This club will be set up at Curtin University, Sarawak, Malaysia. The future members of this club will be the students of Curtin University, Sarawak, Malaysia. Students from both School of Engineering and School of Business will be accepted. Members can be undergraduate, post graduate or pre-university students. Some students may have some experience in programming and some may be novice programmers. Presently at Curtin University, Sarawak, the programming languages taught to non-technical students are C and C++, which is taught only to Engineering students. Seeing that this is not enough and there is potential for students to learn more languages a learning club can come to the rescue. Another aim of the club is to teach non computer science or non IT students programming.

The club will have instructors who are adept programmers and masters in their programming field. These instructors will facilitate the first training sessions. The classes will be held during a time when students are free of other learning engagements and commitments. A part of these lessons will be ‘Lesson Notes’, created by the respective instructors, which will be disseminated to the students for their learning. The main idea behind this club will be to learn programming through learning and teaching believing in the concept that one can learn more when one teaches. So after finishing a programming module, a student will be encouraged to facilitate another training of the same module for a new set of students. Before a student can facilitate training, he or she has to add one page of material to the ‘Lesson Notes’. Students can add content to the ‘Lesson Notes’ by looking for information and exercises from books or from the web. This will perhaps enable the student to be better prepared for the training. After the training, the student facilitator, if so chooses can get feedback about his or her training from his or her students.

Students who finish a particular programming language will be asked to form a team and develop programs and content in the hope that practicing their new learned skill would improve their knowledge. These teams will be supervised by an expert instructor. The club would also host events like programming quizzes, programming games and programming competitions. At the end of the year, a report would be prepared about the total classes taken and the success of the club's initiatives would be weighed. The report would check if the computer programming knowledge of individual members has improved and it would check if students have started developing programs.

3. PROGRAMMING NATION CONSTITUTION

Vision: To be recognized as a club that provides quality teaching and learning of Programming languages.

Mission: To make adept programmers by engaging them to disseminate programming knowledge through teaching and learning and content development.

Organizing committee/Office Bearers: The club will consist of a committee of office bearers. The club’s office bearers will call for a free and fair election every year. The election would be based on a secret ballot. Counting of votes will be transparent.

Designation
Advisor 1
Advisor 2
Advisor 3
Advisor 4
President
Vice-President
Secretary
Vice-Secretary
Treasurer
Vice-Treasurer

Training Classes: Training will be provided first by the club's instructors. The instructor gives training pertaining to a lesson. The instructor distributes the soft copy of 'Lesson Notes' to students for their study.

Training Class Database: The club's organizing committee will maintain a database that lists every training provided under the auspices of the club. This database will be updated regularly with projected addition/deletion of new variables by the organizing committee.

Becoming a Facilitator: The goal of the club is to engage all members and make them facilitators. Following training, successful trainees will give training to other members of the club as a facilitator. In order to qualify as a facilitator, a student should complete the respective training and add one page of written notes to the 'Lesson Notes' document. This document should be properly vetted by the club's organizing body before the training is given.

Training Class Feedback: At the end of each training the facilitator can if they want to, distribute "Student Evaluation Of Teaching" forms to get feedback from the students. This feedback will be given to the facilitator only for his/her safekeeping to be used for developing his/her teaching and learning.

Training Hours: Every facilitator will have a chart in the club which documents the number of hours rendered in teaching programming and or any other activity which is contributing to the facilitator's programming experience. This chart will be maintained by the club's organizing committee in line with the 'Training Class Database'.

Developing Projects: Students will be encouraged to develop projects and content in a team after they finish their respective training.

Experience Letter: At the end of a student's term with Curtin University, Sarawak, he or she can request an experience letter from the club. This letter will be generated by the organizing committee. It would detail the following with respect to the student:

- Total programming languages studied.
- Total hours of classes taken.
- Total programming languages taught as a facilitator.
- Total number of hours as a facilitator.
- Name of developed projects involvement.

Constitution Amendment: Amendment to the constitution can be achieved by the club's organizing committee through a quorum and voting at committee level.

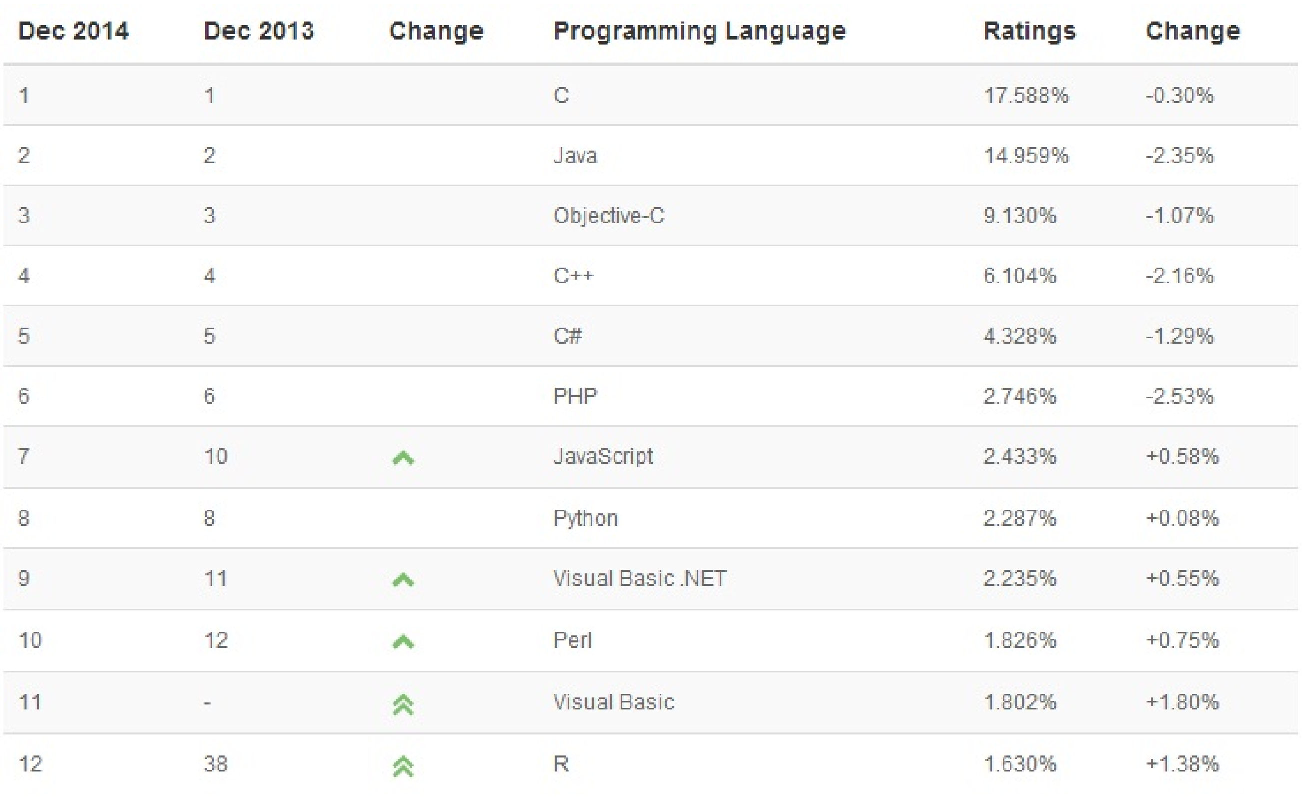
4. DESIGNING MODULES

How to design modules for a programming club? What module should be taught first? There is a worldwide discussion about academic engagement in teaching practice and to an increased emphasis on a scholarly approach to teaching (Boyer, 1997). Changing an approach to teaching requires first the knowledge that other approaches are possible and secondly it requires reflective practitioners (Fincher, 1999). Keeping this in mind, it may appear wise to make the first module of this club to be programming logic and techniques. If teaching programming via the vehicle of any given language constrains the learning process unacceptably, then teaching programming without language would seem to have the attraction of avoiding all these pitfalls (Fincher, 1999). Indeed this has been the practice in many learning institutions where students are first thought how to solve problems through algorithms. Well-educated college students should be exposed to the basic tenets of algorithmic thinking. The algorithm-oriented agenda is not for Computer Science majors only. It is offered as 'foundational twenty-first century knowledge' for a broad population of students from across the family of academic citizens (Shackelford, 1998). Keeping in mind that potential students will also belong to non-technical or non-engineering schools, giving them a basic lesson of algorithmic thinking would help them in their future programming learning. The biggest learning problems are that the students have to handle concepts to which they do not have a concretic model in their everyday life (Robins et all. 2003). Both Gries (Gries, 1974) and Schneider (Schneider,

1978) agree that the main components of an introductory programming course should be algorithmic development and problem solving.

Following algorithms, programming languages can be chosen and taught. Programming language research has unearthed numerous lessons about the nature of Web, reactive, interactive, and asynchronous programming, but these have not made it into virtually any textbook (Krishnamurthi, 2008). One approach to teaching these languages and developing lesson plans for them can be through problem solving (Barnes et al. 1997). Previous research has shown connections between learning programming languages and problem solving skills (Palumbo, 1990). Another approach to teaching the students will be by making the plans and schemes of the lesson visible to them during instruction (Soloway, 1986). This will help students to get the overall picture of their lesson.

Some of the most widely used languages in education and industry are C, Java and C++ as listed in Tiobe programming community index, which is updated once a month. Tiobe uses search engines to give an indication of the popularity of programming languages. The club can concentrate on the main languages depicted in Tiobe. Another important factor to choosing which language to teach will be which language would be best suited to app development as many students are eager to learn that. Studies have found that market appeal, industry demand and student demand is one of the most important factors affecting language choice (de Raadt et all. 2004).



Dec 2014	Dec 2013	Change	Programming Language	Ratings	Change
1	1		C	17.588%	-0.30%
2	2		Java	14.959%	-2.35%
3	3		Objective-C	9.130%	-1.07%
4	4		C++	6.104%	-2.16%
5	5		C#	4.328%	-1.29%
6	6		PHP	2.746%	-2.53%
7	10	⌃	JavaScript	2.433%	+0.58%
8	8		Python	2.287%	+0.08%
9	11	⌃	Visual Basic .NET	2.235%	+0.55%
10	12	⌃	Perl	1.826%	+0.75%
11	-	⏫	Visual Basic	1.802%	+1.80%
12	38	⏫	R	1.630%	+1.38%

Table 1 – Tiobe Programming Community Index

5. CONCLUSION

Establishing a programming club may seem an easy endeavour, but in reality it would require the involved parties to muster their combined effort to make it successful. There may be difficulties in choosing which programming language to teach first and which one to teach next but these issues have to be properly discussed by the club's organizing committee and a consensus should be reached. The next step for this paper is to continue researching after the club creation and measure the outcomes of the club against the expected result of this paper. In addition to that, feedback can be taken from the members of the club with the hope of adding some empirical data to the study. The plan and constitution of the club has been laid out. Next step is carrying out the plan and enforcing the constitution. Hopefully this endeavour will create programmers out of future novice Programming Nation members.

6. REFERENCES

[1] Adams, J. C. (2007). "Alice, middle schoolers & the imaginary worlds camps.", 38th SIGCSE Technical Symposium on Computer Science Education, New York, USA.

[2] Barnes, David J., Fincher, Sally and Thompson, Simon. (1997). "Introductory Problem Solving in Computer Science", 5th Annual Conference on the Teaching of Computing, Dublin, Ireland, August 1997.

[3] Boyer, E. (1997). Scholarship Reconsidered: Priorities of the Professoriate. Jossey-Bass, Hillsdale, NJ, 1997.

[4] de Raadt, M., Watson, R. & Toleman, M. (2004). "Introductory programming: what's happening today and will there be any students to teach tomorrow?", 6th Conference on Australasian Computing Education, pages 277–282. Australian Computer Society, Inc., 2004.

[5] Gries, D. (1974), "What should we teach in an introductory programming course?", 4th SIGCSE Technical Symposium on Computer Science Education, pages 81–89. ACM Press, 1974.

[6] Fincer, S. (1999). "What are We Doing When We Teach Programming?", 29th ASEE/IEEE Frontiers in Education Conference. San Juan, Puerto Rico.

[7] Kelleher, C. & Pausch, R. (2005). Lowering the barriers to programming: a taxonomy of programming environments and languages for novice programmers. ACM Computing Surveys, 37(2), 88-137.

[8] Krishnamurthi, S. (2008). SIGPLAN Workshop on Programming Language Curriculum, Cambridge, MA, USA.

[9] Maloney, J., Peppler, K., Kafai, Y.B., Resnick, M. & Rusk, N. (2008) "Programming by Choice: Urban Youth Learning Programming with Scratch", SIGCSE'08, Portland, Oregon, USA.

[10] Palumbo, D. (1990). Programming language/problem-solving research: a review of relevant issues. Review of Educational Research, 60(1):65–89, 1990.

[11] Robins, A., Rountree, J. & Rountree, N. (2003). "Learning and teaching programming: A review and discussion", ComputerScience Education, vol. 13, no. 2, pp. 137–172, 2003.

[12] Schneider, G.M. (1978). "The introductory programming course in computer science: ten principles", 9th SIGCSE/CSA Technical Symposium on Computer Science Education, pages 107–114. ACM Press, 1978.

[13] Shackelford, R. (1998). Introduction to Computing and Algorithms, Addison-Wesley.

[14] Soloway, E. (1986). Learning to program = learning to construct mechanisms and explanations. Communications of the ACM, 29(9):850–858, 1986.

[15] TIOBE Prramming Community index. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html