Asian Journal of Computer and Information Systems (ISSN: 2321 – 5658)
Volume 01– Issue 04, December 2013

Towards Practical C++ Programming Projects for Pre-University Engineering Students

Foad Motalebi
Curtin University, Sarawak
Miri, Malaysia

ABSTRACT— ***One of the key assessments for C++ programming is the programming project. These programming projects encourage Project Based Learning. Creating and designing a programming project for novice programmers can be a challenging task. This can be compounded by the fact that students may find the project difficult to do and loose interest in the process. So it is befitting to create and design project topics which are practical and interesting and are perceived as such by students. This would encourage students to see the relevance of coding in the real world outside the class and generate exuberance in the programming project. The main motivation of this paper is to show the various C++ projects given to engineering pre-university students over a period of 3 years at Curtin University, Sarawak and in doing so to observe their inclination towards practical project. A project's inclination towards practicality and the interest it generated is done by analysing student performance in their programming project for every semester.***

Keywords— Practical Projects, C++ project, Programming, Group project

---

1. INTRODUCTION

Among the many assessments that a programming student has to go through, submitting a programming project would count as a substantial assessment. These projects often enhance learning and are popularly called as Project Based Learning (PBL). The PBL technique has been successful in both university and pre-university courses (Solomon 2003, Chard 1992). PBL allows increasing students' involvement in the learning process, obtaining better results in terms of knowledge and habits acquired by the students (San-Segundo et al. 2005). Some of the capabilities that students pick while going through PBL are team interaction, independent learning, taking the lead, task distribution and time management.

Creating programming projects for students is a challenging task much so if the project is designed for students who are new to programming. These students can be called novice programmers as they are in the first stages of being a programmer (Bonar & Soloway 1983, Thomas et al. 2004). In every programming course, the students first learn the syntax and semantics of the language and then use that knowledge to create problem solving programs. Programming projects ensure that students achieve both goals (Pattis, 1990). Many students who are novice programmers find programming very difficult, in part due to their inability to learn programming (Oman et al. 1989). This could be a reason why high attrition rates are often experienced in Computer Science schools (Bennedsed & Caspersen, 2007). Novice programmers struggle to grasp an early understanding of programming, which can lead to frustration and eventually surrender (Shuhidan et al. 2009). So it would benefit the whole teaching and learning experience if the students are given a programming project which is both practical and interesting.

Programming assessments show that there is a strong support for association between a student's code tracing and code writing skills (Mike Lopez, 2008). Programming projects would be difficult to do if a student has poor code tracing skill. Practical projects may encourage students to spend time and learn this skill through collaborative learning or self-study. Giving students simple yet practical project topics to which they can relate to brings programming to the real world for them. It gives them an idea about how programs can be used outside class room in the real world. This would serve as a motivation for them to solve their project using the knowledge they picked from the course as well as appending to that knowledge by researching other coding requirements for their project.

2. GROUP PROJECT

Successful engineers must be able to recognize and fill gaps in their education, work effectively in groups, and use past experience to predict future performance (Diwan et al. 2002). At Curtin University, Sarawak, Pre-University

Engineering students study programming for a year. The programming language chosen for these novice students is C++ programming. C++ as an introductory language is perceived as a better option over C as it results into improved software quality and reduced maintenance effort, and in addition to that many code bases are shifting from C to C++ (Battacharya & Neamtiu, 2011). C++ language technicalities can provide not only high level programming features such as object-orientation paradigms and generic programming, but also low-level programming features, such as bit-wise and device I/O control abilities (Lippman & Lajoie 1998, Stroustrup 2009).

The Pre-University students programming course is divided into Programming in C++ 061 for first semester and Programming in C++ 062 for second semester respectively. The first semester of C++ mainly deals with the non-OOP content, which consists of the structural and procedural aspects of the language whereas in the second semester, the OOP features of the language are introduced and its benefits elucidated.

Students are taught during the course, during each topic, with the aid of coding examples, how best to use the programming features to solve problems. However, sometimes poor models of programming concepts have a propagating effect, plunging novices into a spiral of frustration, loss of confidence and self-belief, as more complex material is covered (Caspersen & Bennedsen, 2007). The course consists of lecture sessions and lab sessions. During the lab session the students solve lab questions by writing programs that are unsubstantial programs involving little or no group work. However, at the end of the semester they have to code a project which they do so in a group. This project-based curriculum is necessary for developing student abilities to incorporate various programming skills (Lang et al. 2006). The students are given at least 3 weeks time to finish the project. It is hoped that students would work in groups and divide the task accordingly as they deem fit, mirroring the real world of code development. However, it can be observed here that many students actively seek strategies to avoid genuine group work, relying instead on dividing up the task or placing the whole responsibility on a minority of the group (Diwan et al. 2002). A practical interesting project would capture the attention of the whole group, mustering the combined efforts of the members to be involved in the process.

Students generally divide the project task among themselves. This is an important part of learning how to program where the students learn how to decompose a problem into smaller pieces (Caspersen, 2008). If the students are dividing up the project tasks in the group they are told to be aware of each other's coding, where each student in the group explains his/her contribution code to the rest of the group. The main idea of this being collaborative learning where one student shares his logic and thinking with another student thus elevating the overall problem solving, logic and programming knowledge of the group. This is an integral part of the project and it has a group assessment component that puts students at a disadvantage if they don't do this. Besides the actual project marking, the group assessment components include Self & Peer Assessment and Viva Interview, which assesses the individual contribution of coding to the group.

A strategy that seems to work to some degree and would make students enthusiastic towards collective group work would be creating a practical project that they can relate to and be exuberant about. A practical project would encourage students to be goal-directed, thus giving themselves opportunities to practice and incorporate C++ skills, open source libraries and development tools (Perez & Rosell, 2010). Practical projects can also generate the attention and interest of students who are not keen on learning programming but rather interested in the end product that the coding would produce. This would perhaps show these students the relevance and importance of learning a programming language.

3. DESIGNING PRACTICAL C++ PROJECTS

Practical projects encourage students to be involved in the coding process as they understand its significance and usefulness (Pattis, 1990). Finding practical topics on programming projects requires having an idea for a programming project and then trying to simplify it in the coding process. Keeping this in mind the points taken into consideration while creating a project were:

1. The topic and its requirements should be easy to understand.
2. The topic should be contemporary and something that the students can relate to.
3. The developed program should have a practical use, serving as an interesting and useful tool.
4. The program code should incorporate features that were taught during the semester as outlined in the Unit Outline.

4. TOWARDS PRACTICAL PROJECT TOPICS

The main motivation of this paper is to show the various C++ projects given to engineering pre-university students over a period of 3 years at Curtin University, Sarawak and in doing so to observe their inclination towards practical projects.

These practical projects were designed with the hope to garner the student's attention and interest. The detailed project requirements and instructions would not be listed as these are unique for every given project. These requirements and instructions aimed to help students code their project by mustering the knowledge they entailed throughout their programming course.

For two semesters the project topics given was the same for both Programming in C++ 061 and Programming in C++ 062. These were semester 1, 2011 and semester 1, 2012. However C++ 061 incorporated non-OOP features to solve the project whereas C++ 062 used OOP features of the language to solve the same project.

The Projects given to Programming in C++ 061 students for the past three years and their basic specifications are listed in Table 1:

Table 1: Projects given to Programming in C++ 061 Students

Semester	**C++ 061 Programming Project Topic**
Sem 1 2011	Create **13 Lab Exercises**, for every topic of Programming in C++ 061 taught during the semester. Every lab exercises must consist of a question pertaining to the topic taught. The exercise must ask students to do a relevant task and it must also provide the correct answer in code.
Sem 2 2011	Create a **Windows Shortcut Program**. The program asks the user to enter a 2 character string shortcut following which it starts the respective application associated with the shortcut.
Sem 1 2012	Create a **Computer Scrabble Game Program**. The game consists of Ten words that start from a 3 letter word to a 12 letter word subsequently. (Coding done by using non OOP features)
Sem 2 2012	Miri City Council would like to implement LCCF (Low Carbon Cities Framework). As such develop a **Carbon Calculator Program** that helps Miri City Council to test and simulate where they can reduce GHG (Green House Gases) in the city by 15%. Miri City Council has identified the following variables to be major contributors to GHG in the city: Industry – 19%, De-forestation – 17%, Power & Energy – 15%, Agriculture – 14%, Transportation – 10%, Waste – 10%, Water Management – 10% and Buildings – 5%. The program must ask the user to input the percentage of cuts in the relevant 8 variables in the given order amounting to a total of 15% cut of GHG.
Sem 1 2013	Create a program that emulates a game titled **Guess The Right Word For The Picture**, a game that shows the user a picture and awaits the user's correct word input that best describes the picture.
Sem 2 2013	Create a **Jukebox** program that plays a selection of songs based on user's choice. In addition it should also have a song delay feature that plays a song afters the delay time set by the user.

These projects for Programming in C++ 061 assessed the following features of C++ programming language:

1. Algorithm depiction in the form of flow chart or pseudocode.
2. Decision constructs usage.
3. Usage of loops.
4. Usage of arrays.
5. Incorporation of modular programming (Functions with and without arguments).
6. Incorporation of system functions.
7. Incorporation of function overloading.
8. Adhering to project specification and guidelines.
9. Reducing redundancy in code.

The Projects given to Programming in C++ 062 students for the past three years and their basic specifications are listed in Table 2:

Table 2: Projects given to Programming in C++ 062 Students

Semester	C++ 062 Programming Project Topic
Sem 1 2011	Create **16 Lab Exercises**, for every topic of Programming in C++ 061 taught during the semester. Every lab exercise must consist of a question pertaining to the topic taught. The exercise must ask students to do a relevant task and it must also provide the correct answer in code.
Sem 2 2011	Create a **Periodic Table Program**. The program asks the user to enter an element's symbol following which it shows all relevant information regarding the typed element.
Sem 1 2012	Create a **Computer Scrabble Game Program**. The game consists of Ten words that start from a 3 letter word to a 12 letter word subsequently. (Coding done by using OOP features)
Sem 2 2012	Create a **Traffic Light Simulation** program for a busy intersection in a city. The intersection has 4 roads namely ROAD-A, ROAD-B, ROAD-C and ROAD-D. Each road displays the green light as per its traffic size - ROAD-A (30 seconds), ROAD-B (20 seconds), ROAD-C (15 seconds) and ROAD-D (10 seconds). The program should follow the output displayed as seen in the video file "C++062-Project-2012-Sem-2".
Sem 1 2013	Create a program for a **Cineplex** where the user can do the following tasks: User is shown the choice of movies being screened, user can get tickets to the show by seeing the seating arrangement & seats available on screen, user can choose his/her refreshments and user can give movie reviews.

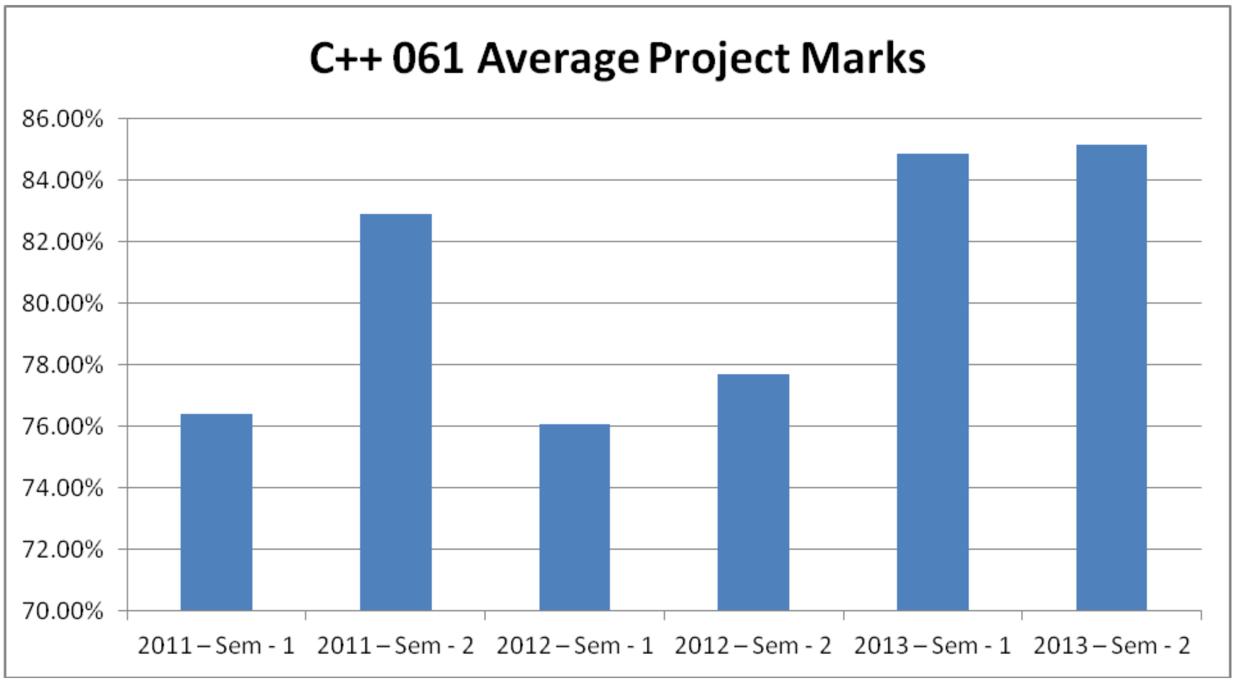
These projects for Programming in C++ 062 assessed the following features of C++ programming language:

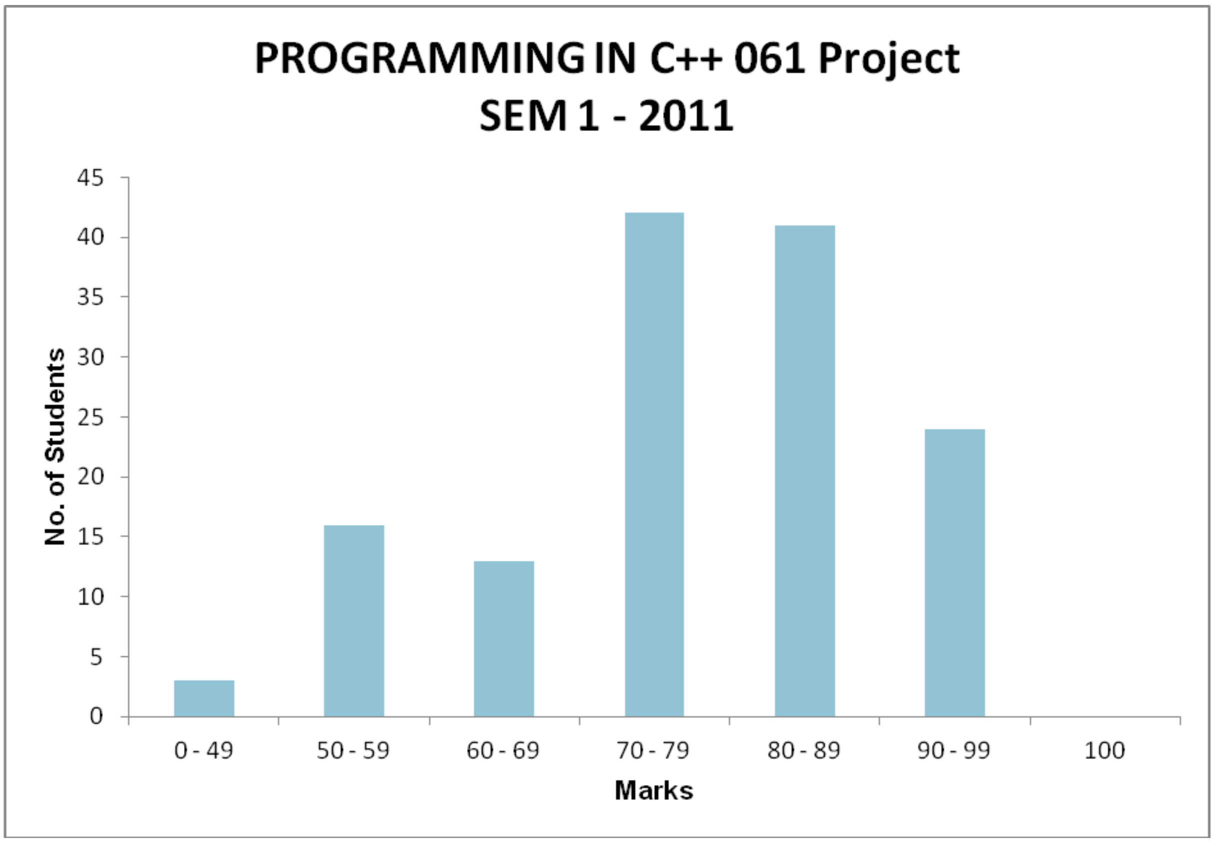
1. Incorporation of OOP.
2. Usage of Arrays.
3. Usage of decision constructs and loop constructs.
4. Usage of static and extern data storage types.
5. Incorporation of user-defined functions and system functions.
6. Incorporation of function overloading or constructor overloading.
7. Incorporation of inheritance in code.
8. Incorporation of Input and Output Streams for reading and writing files.
9. Incorporation of friend functions and friend classes.
10. Incorporation of exception handling in code.
11. Incorporations of at least 6 new features in the code that has not been taught in both Programming in C++ 061 and Programming in C++ 062.
12. Adhering to project specifications and guidelines.
13. Reducing redundancy in code.

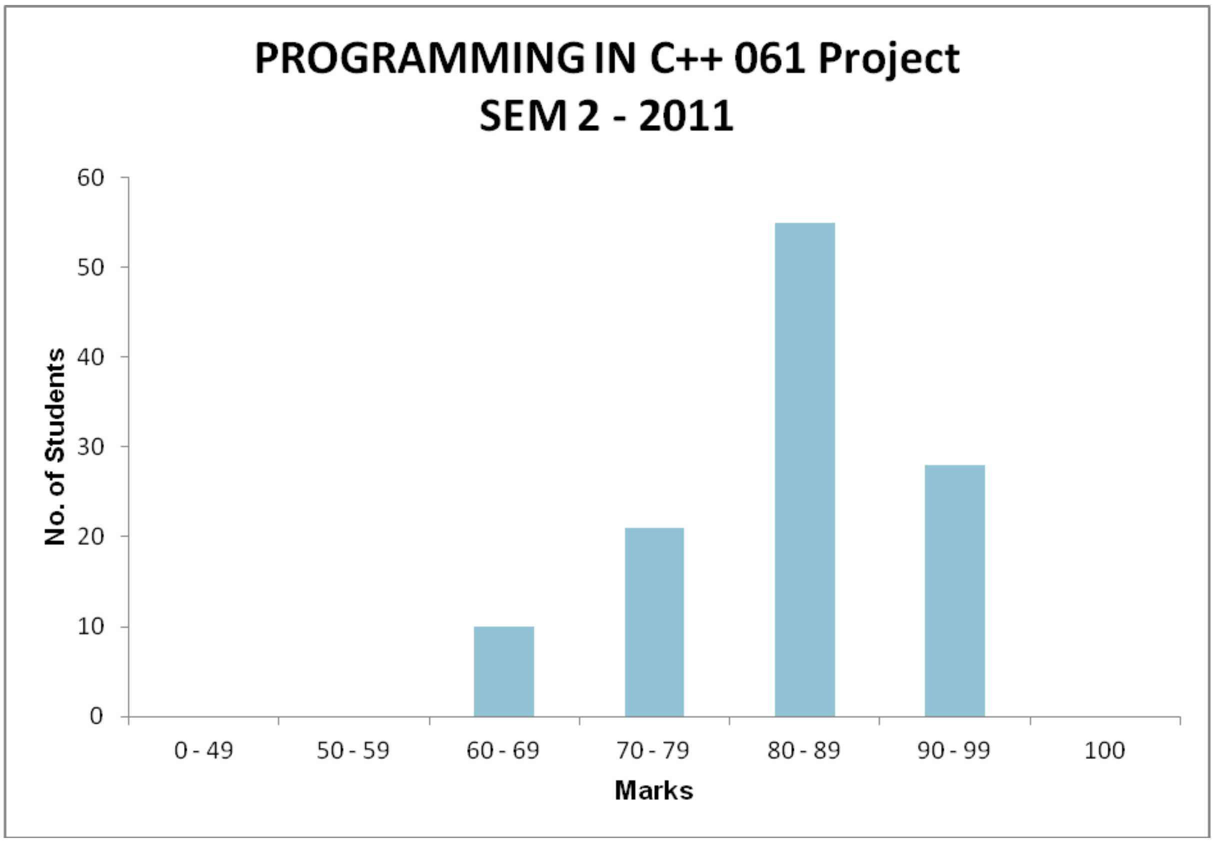
5. STUDENT PERFORMANCE IN C++ 061

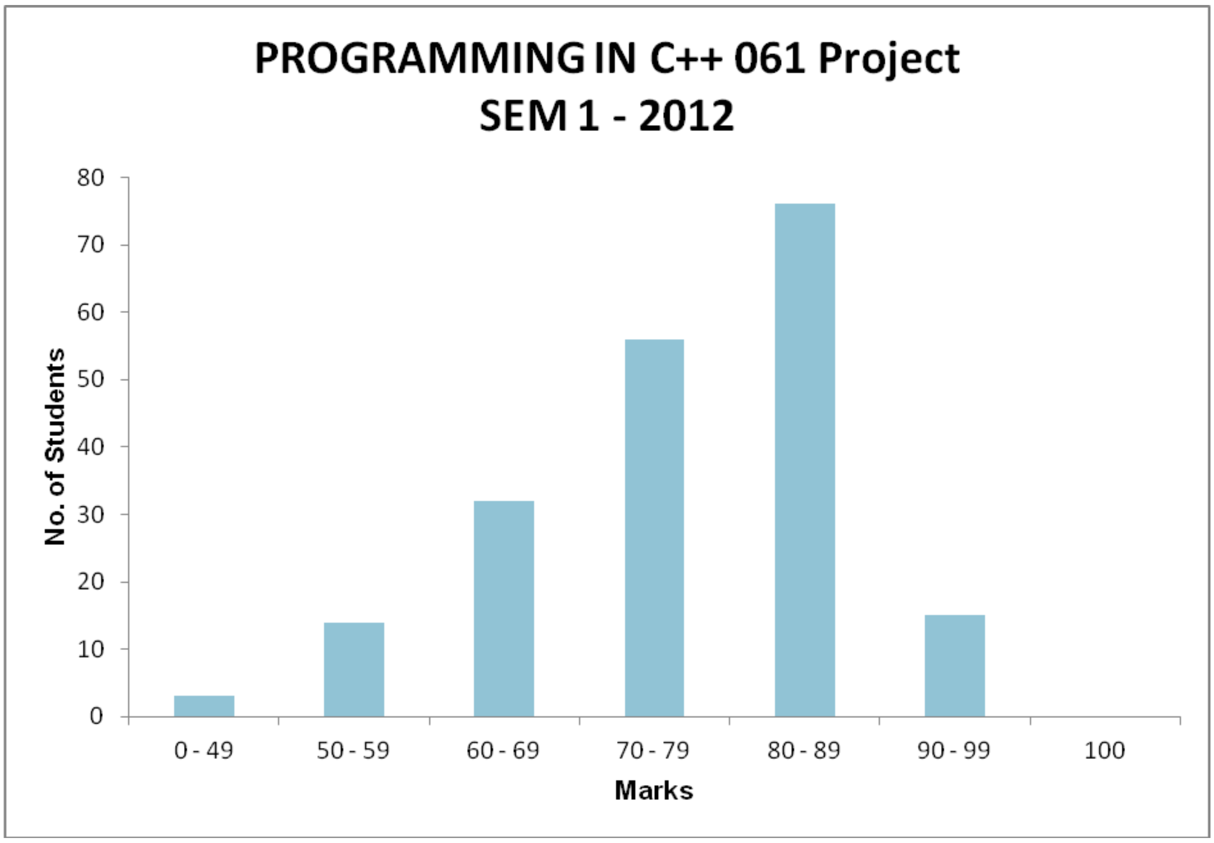
Table 3: Average marks attained by students in Programming in C++ 061 Project

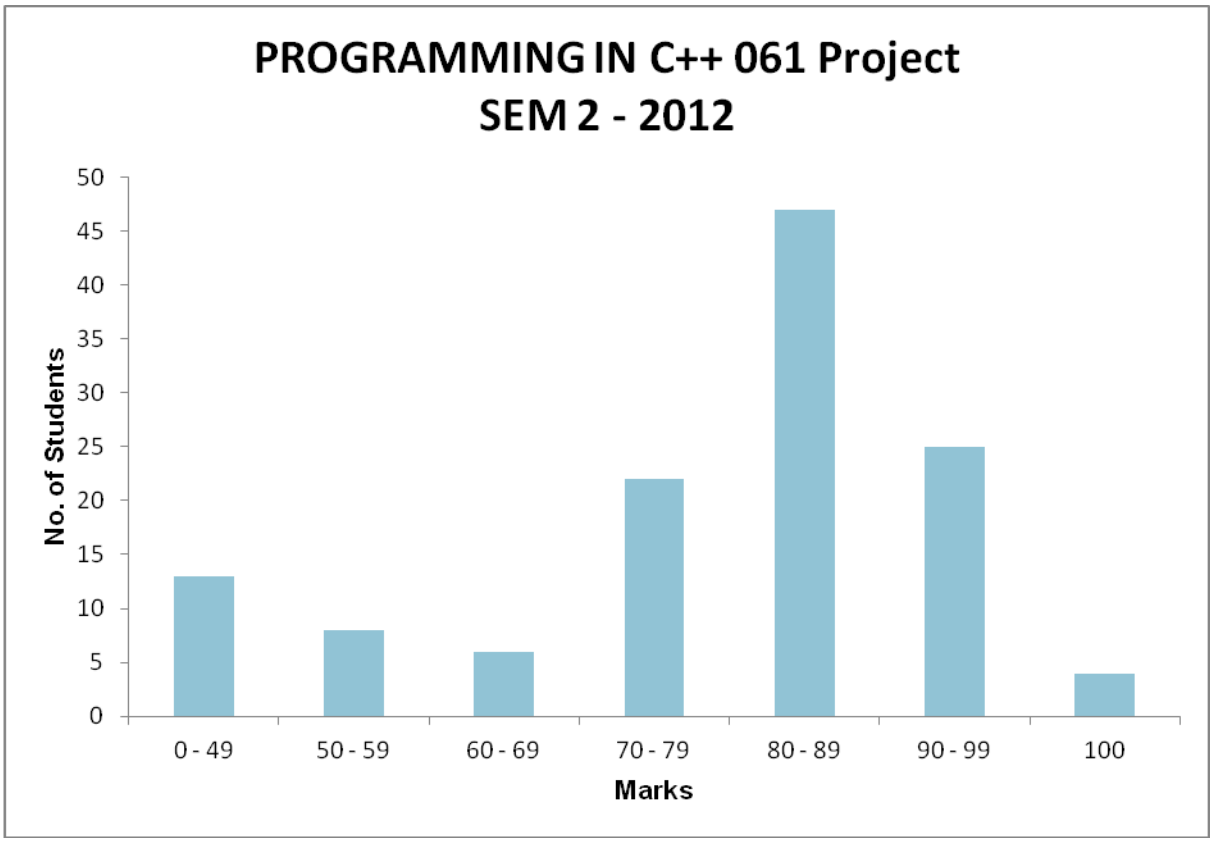
Semester	Average Marks
2011 – Sem - 1	76.41 %
2011 – Sem - 2	82.91 %
2012 – Sem - 1	76.08 %
2012 – Sem - 2	77.70 %
2013 – Sem - 1	84.87 %
2013 – Sem - 2	85.16 %

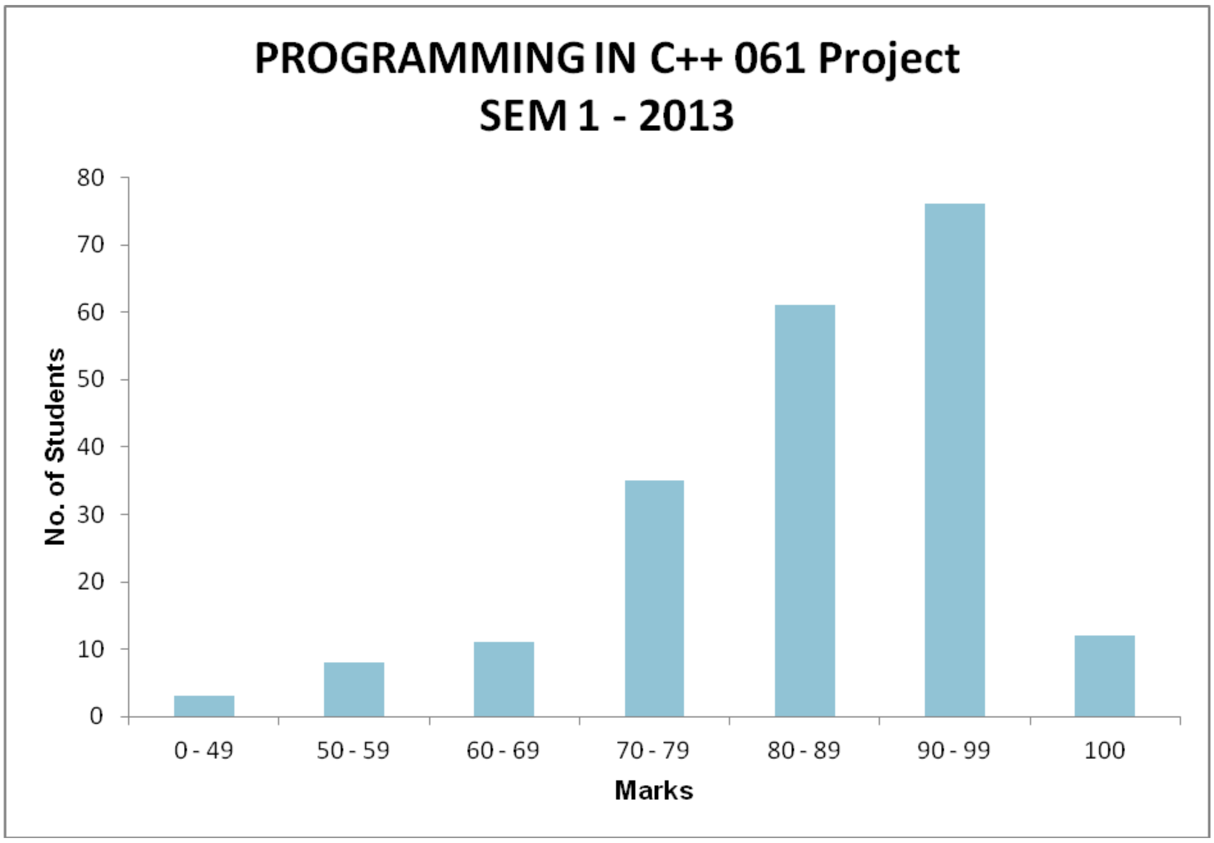

C++ 061 Average Project Marks
86.00%
84.00%
82.00%
80.00%
78.00%
76.00%
74.00%
72.00%
70.00%
2011 – Sem - 1
2011 – Sem - 2
2012 – Sem - 1
2012 – Sem - 2
2013 – Sem - 1
2013 – Sem - 2

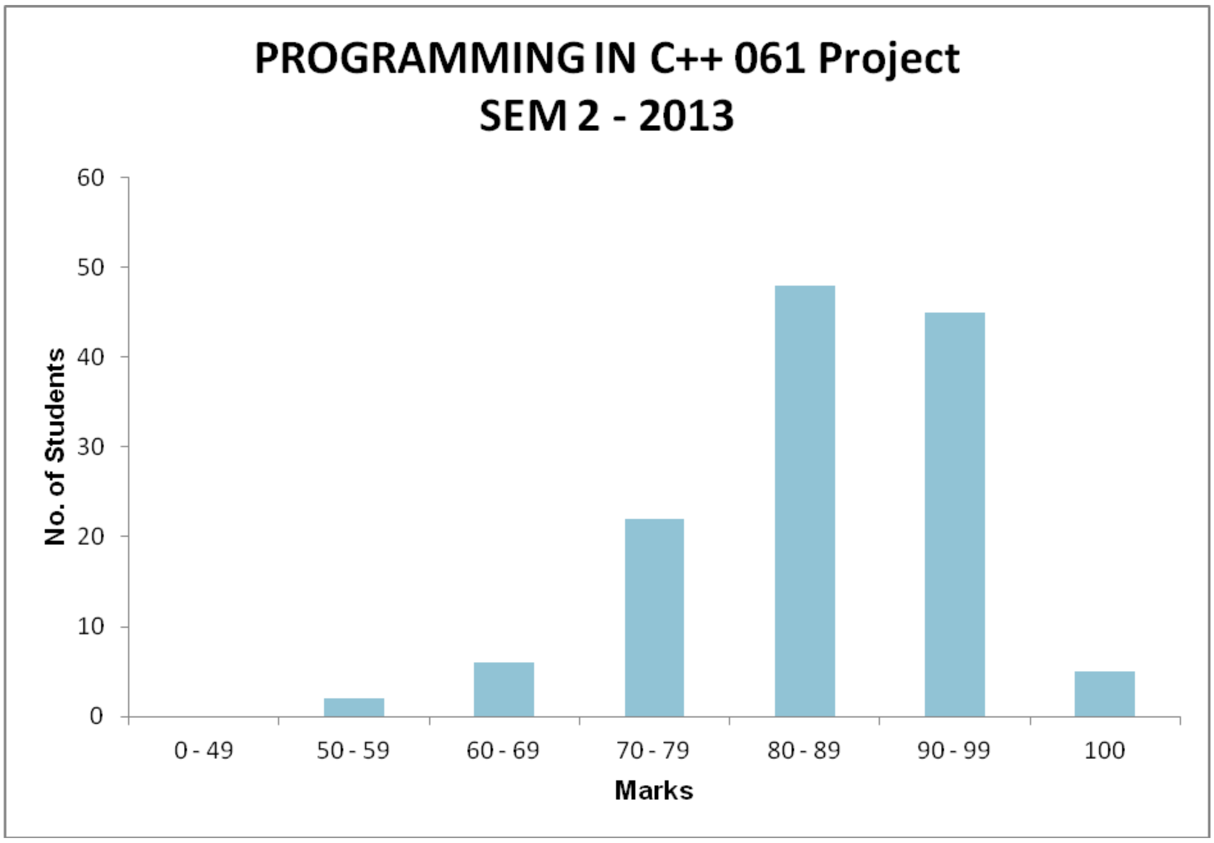

PROGRAMMING IN C++ 061 Project
SEM 1 - 2011
No. of Students
45
40
35
30
25
20
15
10
5
0
0 - 49
50 - 59
60 - 69
70 - 79
80 - 89
90 - 99
100
Marks


PROGRAMMING IN C++ 061 Project
SEM 2 - 2011
No. of Students
0
10
20
30
40
50
60
0 - 49
50 - 59
60 - 69
70 - 79
80 - 89
90 - 99
100
Marks


PROGRAMMING IN C++ 061 Project
SEM 1 - 2012
No. of Students
0
10
20
30
40
50
60
70
80
0 - 49
50 - 59
60 - 69
70 - 79
80 - 89
90 - 99
100
Marks


PROGRAMMING IN C++ 061 Project
SEM 2 - 2012
No. of Students
0
5
10
15
20
25
30
35
40
45
50
0 - 49
50 - 59
60 - 69
70 - 79
80 - 89
90 - 99
100
Marks

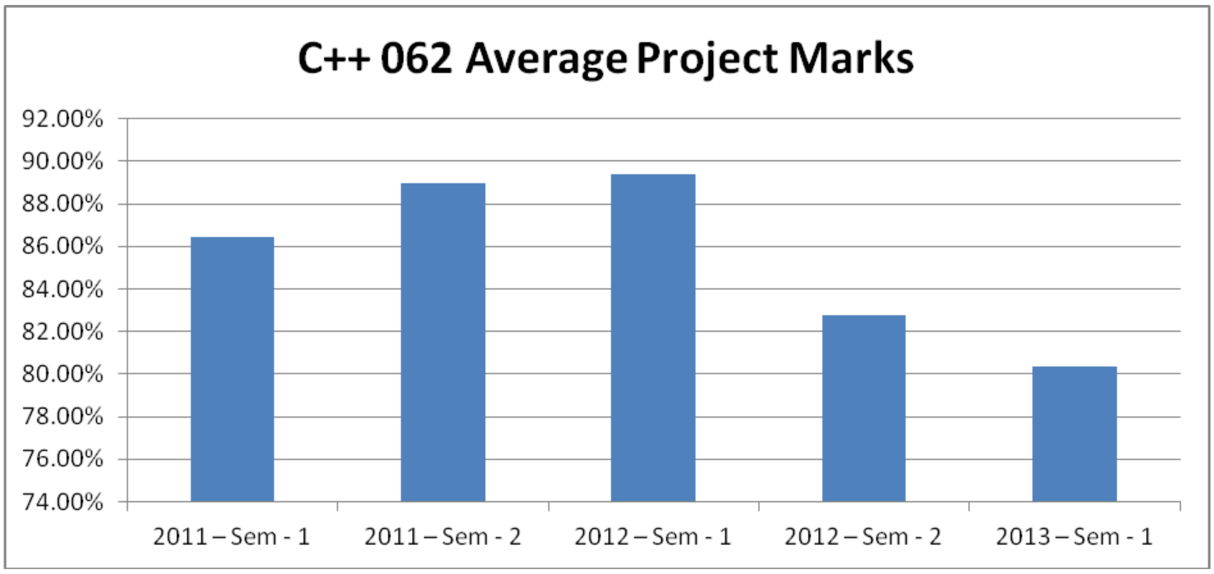

PROGRAMMING IN C++ 061 Project
SEM 1 - 2013
No. of Students
0
10
20
30
40
50
60
70
80
0 - 49
50 - 59
60 - 69
70 - 79
80 - 89
90 - 99
100
Marks

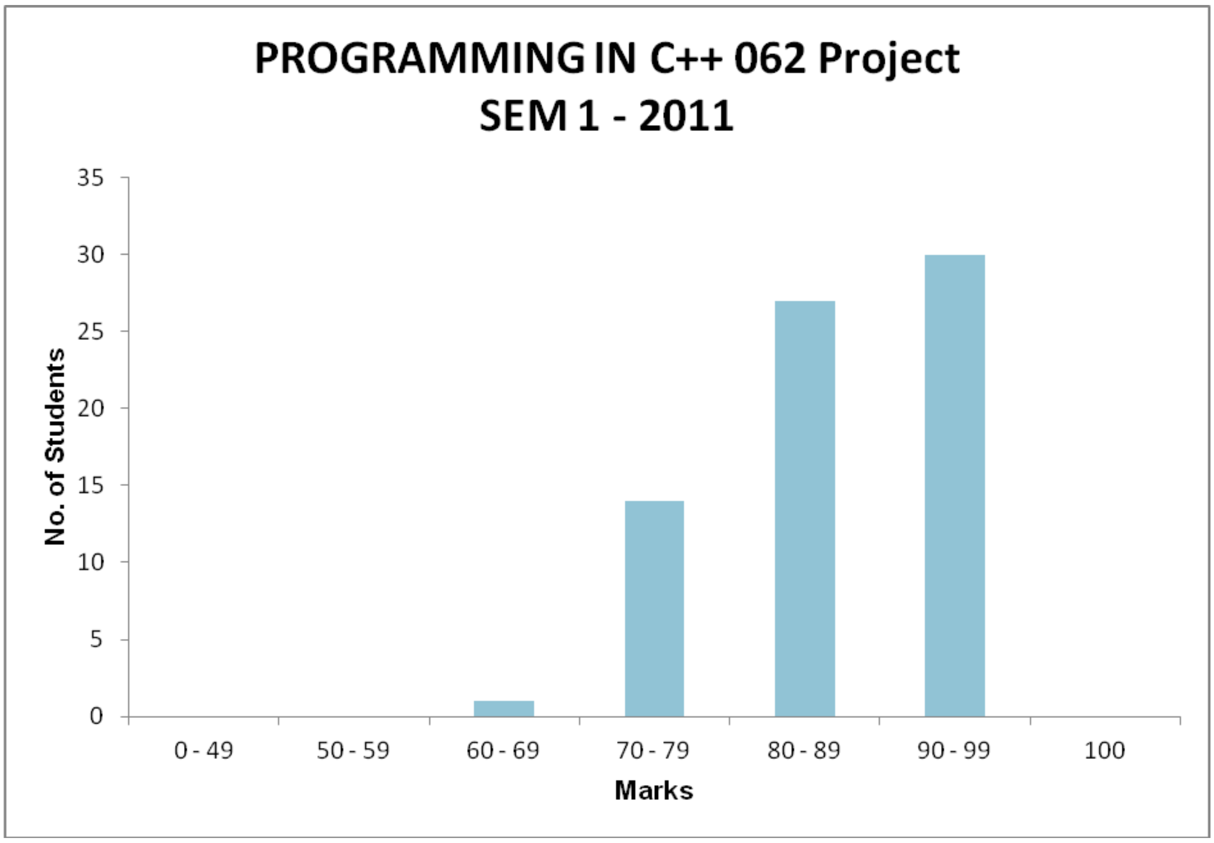


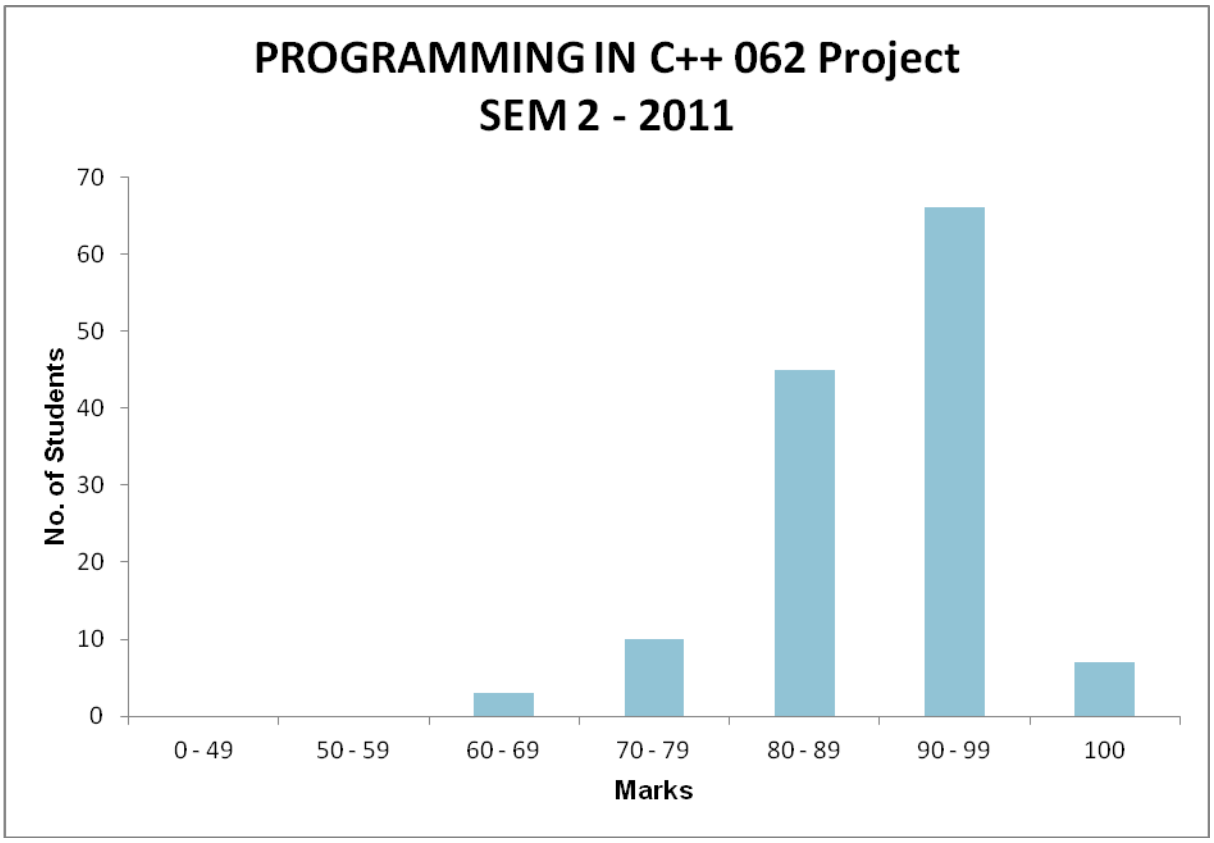
4. Student Performance in C++ 062

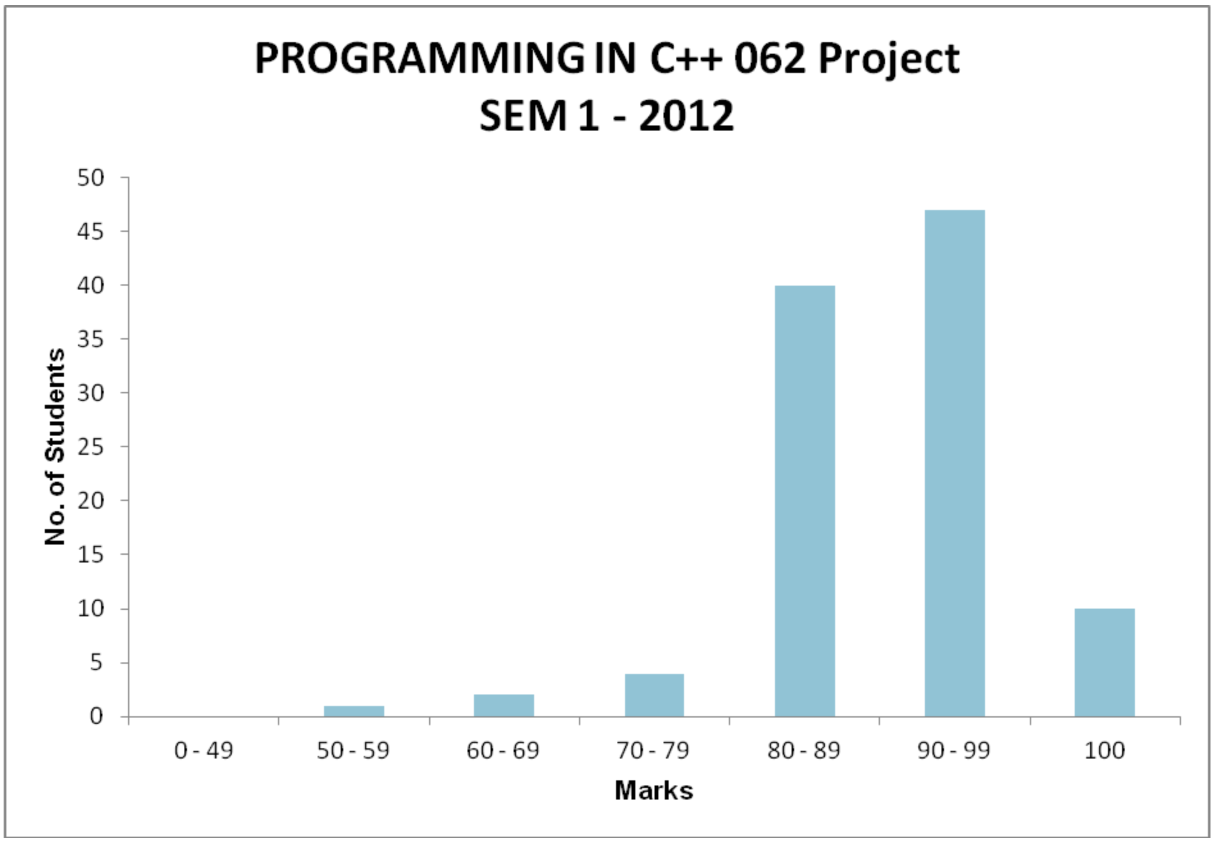
Table 4: Average marks attained by students in Programming in C++ 062 Project

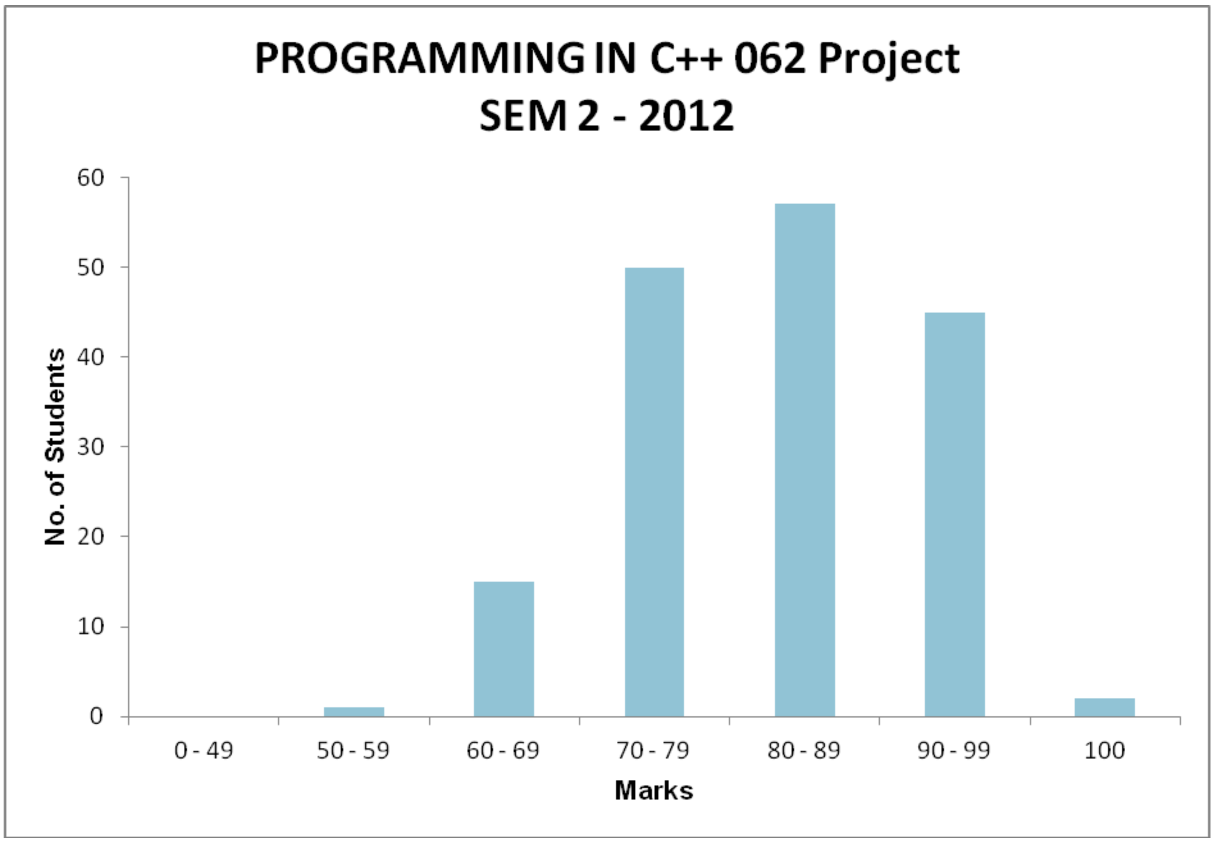
Semester	Average Marks
2011 – Sem - 1	86.41 %
2011 – Sem - 2	88.96 %
2012 – Sem - 1	89.39 %
2012 – Sem - 2	82.78 %
2013 – Sem - 1	80.36 %

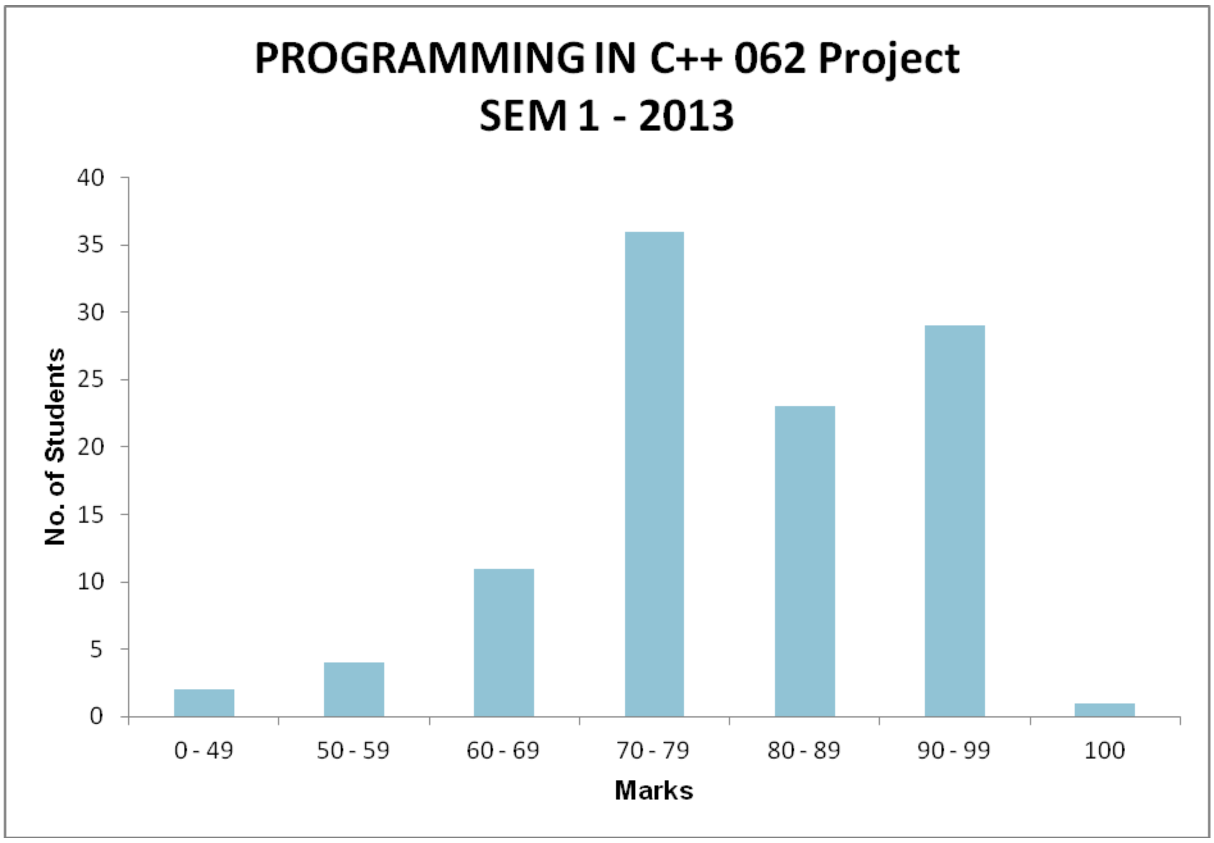

C++ 062 Average Project Marks
92.00%
90.00%
88.00%
86.00%
84.00%
82.00%
80.00%
78.00%
76.00%
74.00%
2011 – Sem - 1
2011 – Sem - 2
2012 – Sem - 1
2012 – Sem - 2
2013 – Sem - 1


PROGRAMMING IN C++ 062 Project SEM 1 - 2011
No. of Students
35
30
25
20
15
10
5
0
0 - 49
50 - 59
60 - 69
70 - 79
80 - 89
90 - 99
100
Marks


PROGRAMMING IN C++ 062 Project
SEM 2 - 2011
No. of Students
0
10
20
30
40
50
60
70
0 - 49
50 - 59
60 - 69
70 - 79
80 - 89
90 - 99
100
Marks


PROGRAMMING IN C++ 062 Project
SEM 1 - 2012
No. of Students
0
5
10
15
20
25
30
35
40
45
50
0 - 49
50 - 59
60 - 69
70 - 79
80 - 89
90 - 99
100
Marks


PROGRAMMING IN C++ 062 Project
SEM 2 - 2012
No. of Students
60
50
40
30
20
10
0
0 - 49
50 - 59
60 - 69
70 - 79
80 - 89
90 - 99
100
Marks


PROGRAMMING IN C++ 062 Project
SEM 1 - 2013
No. of Students
40
35
30
25
20
15
10
5
0
0 - 49
50 - 59
60 - 69
70 - 79
80 - 89
90 - 99
100
Marks

6. CONCLUSION

Though attempt was made to create and design practical projects throughout all the semesters for the past three years, students found some projects more practical and interesting than others. The next step for this paper would be to try and find out what variables made those project topics more interesting than the rest and somehow try and replicate those variables for future projects. In addition to that feedback can be taken from academics and students associated with the unit to try and improve the project topics.

Creating and designing practical C++ projects was a satisfying experience. Especially when students took the project into their stride and found it interesting. This caused them to be more engrossed and active in their project work. Thus there would be a continuation to aim to create a Project that is both practical and interesting.

7. REFERENCES

[1] Battacharya, P., and Neamtiu, L. (2011). "Assessing Programming Language Impact on Development and Maintenance: A Study on C and C++", ICSE, Honolulu, Hawaii, USA, May 2011.
[2] Bennedsen, J. & Caspersen, M. E. (2007). "Failure rates in introductory programming", SIGCSE Bull. 39(2), 32–36.
[3] Bonar, J. & Soloway, E. (1983), "Uncovering principles of novice programming", in 'POPL '83: Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages', ACM, New York, NY, USA, pp. 10–13.
[4] Casperson, D. (2008). "Experience With Team Projects In A Second-Semester C++ Programming Course", http://www.cs.ubc.ca/wccce/Program03/papers/Casperson.html
[5] Caspersen, M. E. & Bennedsen, J. (2007), "Instructional design of a programming course: a learning theoretic approach", in 'ICER '07: Proceedings of the third international workshop on Computing education research', ACM, New York, NY, USA, pp. 111–122.
[6] Chard, S.C. (1992). "The Project Approach: A Practical Guide for Teachers", Edmonton, Alberta: University of Alberta Printing Services.
[7] Diwan, A., Waite, W.M., and Jackson, M.H. (2002). "An Infrastructure for Teaching Skills for Group Decision Making and Problem Solving in Programming Projects", Special interest group on computer science education (SIGCSE) (Covington, Kentucky, USA, March 2002), pp. 276-277.
[8] Lang, J., Nugent, G.C., Samal, A. & Soh, L.K. (2006). "Building Communication Software: A Project Based Approach For Teaching C++ Object-Oriented Programming", 'International Journal of Innovative Computing, Information and Control', Volume 9, Number 8: pp 3415 – 3436.
[9] Lippman, S. & Lajoie, J. (1998). C++ Primer, 3rd Edition, Addison-Wesley, MA, USA.
[10] Mike Lopez, Jacqueline Whalley Phil Robbins, R. L.(2008), "Relationships between reading, tracing and writing skills in introductory programming", in 'ICER'08'.
[11] Oman, P. W., Cook, C. R. & Nanja, M. (1989). "Effects of programming experience in debugging semantic errors", J. Syst. Softw. 9(3), 197–207.
[12] Pattis, R.E. (1990). "A Philosophy and Example of CS-1 Programming Projects", Association for Computing Machinery 089791: pp. 34 – 39.
[13] Perez, A. & Rosell, J. (2010). "A roadmap to robot motion planning software development", Comput. Appl. Eng. Educ., Vol 18: pp. 651-660.
[14] San-Segundo, R., Montero, J.M., Macias-Guarasa, J., Cordoba, R. & Ferreiros, J. (2005). "Automatic Tools for Diagnosis and Feedback in a Project Based Learning Course", 35th ASEE/IEEE Frontiers in Education Conference. Indianapolis, IN, USA.
[15] Shuhidan, S., Hamilton, M. & D'Souza, D. (2009). "A Taxonomic Study of Novice Programming Summative Assessment", 11th Australasian Computing Education Conference (ACE2009), Wellington, New Zealand, January, 2009.
[16] Solomon, G. (2003). "Project-Based Learning: a Primer", Technology and Learning. Volume 23(6), pp. 20-27.
[17] Stroustrup, B. (2009). Programming: Principles and Practice Using C++, Addison-Wesley.
[18] Thomas, L., Ratcliffe, M. & Thomasson, B. (2004). "Scaffolding with object diagrams in first year programming classes: some unexpected results", SIGCSE Bull. 36(1), 250–254.