AC 2011-308: TEACHING COMPUTER PROGRAMMING SKILLS TO ENGINEERING AND TECHNOLOGY STUDENTS WITH A MODULAR PROGRAMMING STRATEGY

Wangping Sun, Oregon Institute of Technology

Dr. Wangping Sun is an associate professor at Oregon Institute of Technology.

Xian Sun, Klamath Union High School

Teaching Computer Programming Skills to Engineering and Technology Students with a Modular Programming Strategy

Abstract

Learning basic computer programming skills is important for engineering and technology students in their early years of college education. In our school, ENGR 266 Computer Programming for Engineers is a required course for freshmen and sophomores whose majors are mechanical engineering, renewable energy engineering, mechanical engineering technology and manufacturing engineering technology. In this course, students learn how to develop computer programs with VBA (Visual Basic Application) and MATLAB.

ENGR 266 has been a challenging course to teach due to three major factors: course coverage and students' previous programming experiences and technical competence to solve engineering problems. After years of trial and error, the instructor (the first author of this paper) summarized a set of core programming skills that can be mastered in one academic quarter. These programming skills are taught with a modular programming strategy. Through an in-class project, the students learned how to solve a complex problem by dividing it into small pieces, addressing each piece by a programming procedure and integrating the intermediate result from each procedure into the final solution.

It is believed that with the modular programming strategy and the core programming skills, the students will be able to develop computer code to solve most engineering problems. Meanwhile, this pedagogic model makes the computer programming course less challenging. The authors are seeking the opportunity to apply the same pedagogic model in a pilot VBA Programming course at a high school.

1. Introduction

Computer programming is an essential and integral part of any engineering program. Engineering students in their junior and senior years face the task of solving problems by using numerical approaches. Good programming skills will enable them to tackle these problems easily[1]. It is important for engineering and technology students to learn basic computer programming skills in their early years of college education [1-5]. ENGR 266 Computer Programming for Engineers is a required course in our school for freshmen and sophomores majoring in mechanical engineering, renewable energy engineering, mechanical engineering technology and manufacturing engineering technology. In this course, the students learn how to develop computer programs with VBA (Visual Basic Application) and MATLAB. The pre-requisite of this course is MATH 111 College Algebra. ENGR 266 is a 3-credit course with a 2-hour lecture and a 3-hour lab each week.

Over the years, ENGR 266 has been a challenging course to learn. Most students were afraid to take it and some students postponed taking it until their junior or senior years. The instructor was also frustrated to teach the course because the students' academic performances and the teaching evaluations of the course were both low. Even after taking this course, more than 50% of the students did not have enough confidence in their programming skills and at least 30% of the students did not meet the instructor's expectation.

2. Problems in teaching the course

According to the observation of the author, three major issues existed in the teaching of the course. The first issue was that the course coverage was too broad for one academic quarter. Our school applies the quarter system. Each quarter has 10 weeks (including the "dead week") and 1 final exam week. ENGR 266 used to cover these topics: 1) VBA (data types and variables, sub-routines, functions, decisions, all kinds of loops, data input and output, graphical user interface design, automatic macro recording, etc.), 2) MATLAB (similar to VBA coverage plus matrix manipulation, script files, function files, 2D and 3D graphics plotting, curve fittings, linear equation system, numeric symbols, etc.) and 3) applications of VBA and MATLAB in solving physics, math and engineering problems. Since there were a lot of materials to cover, every topic was taught in a hustle. The confusion from the students in class was widespread.

The second issue was that the students had little programming experience before they took ENGR 266. For example, most of the students didn't know that the data type of a variable should be defined before that variable could be used. They also thought that the expression "x = 5" was equivalent to "5 = x". To lecture these students, a step-by-step hands-on instruction would be very essential. The pace of the lecture and lab should be adjusted from time to time. However, the pace of course was unable to slow down because the scope of ENGR 266 was unreasonably defined (as mentioned above). Therefore, the students got lost very easily in the course.

The third issue was that the programming practices demanded tremendous knowledge in pre-calculus, calculus, physics, statics, kinematics and electric circuits, while most students only had a college algebra background. The textbooks used by ENGR 266 are: 1) MATLAB – An Introduction with Applications (by A. Gilat) and 2) Introduction to VBA for Excel (by S. Chapra). The students usually spent too much time understanding the question and formulating the answer instead of focusing on practicing their programming skills.

The instructor tried to address the above issues by using Software Development Cycle under the team environment [2]. However, the success was very limited and the students' learning outcome did not improve very much. The instructor also did research on how other universities taught the similar course. It was found that some universities used animation, GUI design or computer games to stimulate students' study interests [1,3], some integrated engineering projects (such as robotics) into the teaching [4,5] and some used other programming languages (such as C and C++) or software packages (such as LabView) to teach the course [6-9]. Even though all these previous research achievements are helpful, they are unable to answer the following questions:

- What are the core programming skills that the engineering and technology students have to master?

- How can the core programming skills be effectively mastered by the students in a short time, say 10 weeks?

2. Core programming skills for engineering and technology students

The inclusion of computer programming in the engineering and technology curricula has been seen as a way to improve students' skills in logical reasoning, quantitative problem solving and the application of technical knowledge [5]. Therefore, it is important to define a core knowledge that the students can use to solve engineering problems. Based on the author's previous experiences in industry and his discussions with other engineering faculty members, the author defined the following topics as the core concepts:

- In VBA, the students should learn and practice fundamental programming concepts. They should learn data types, variable definitions, sub-routines, functions, modules, for-loops, if-elseif-else statement, input/output of data files, VBA built-in functions, interaction between Excel spreadsheet cells and VBA programs and engineering problem solving by using VBA.
- In MATLAB, the students should practice solving more engineering problems after they have learned the fundamental rules on how to use this tool. They should learn matrix manipulation, linear equation systems, element-by-element calculation of matrices, script files, functions, if-statement, for-loops, 2-dimensional plots, MATLAB built-in functions, input/output of data files and engineering problem solving by using MATLAB.

After defining the scope, the workload of ENGR 266 was significantly reduced. Some topics, such as pass-by-reference, recursive functions in VBA, 3D plotting, curve fitting in MATLAB were eliminated from the course. Some other topics were delegated to other courses. For example, topics such as using Excel spreadsheet and recording macros are now included in MET 111 Orientation. The programming practice problems were also simplified. All the problems required for skills no more than college algebra, high school trigonometry and high school physics.

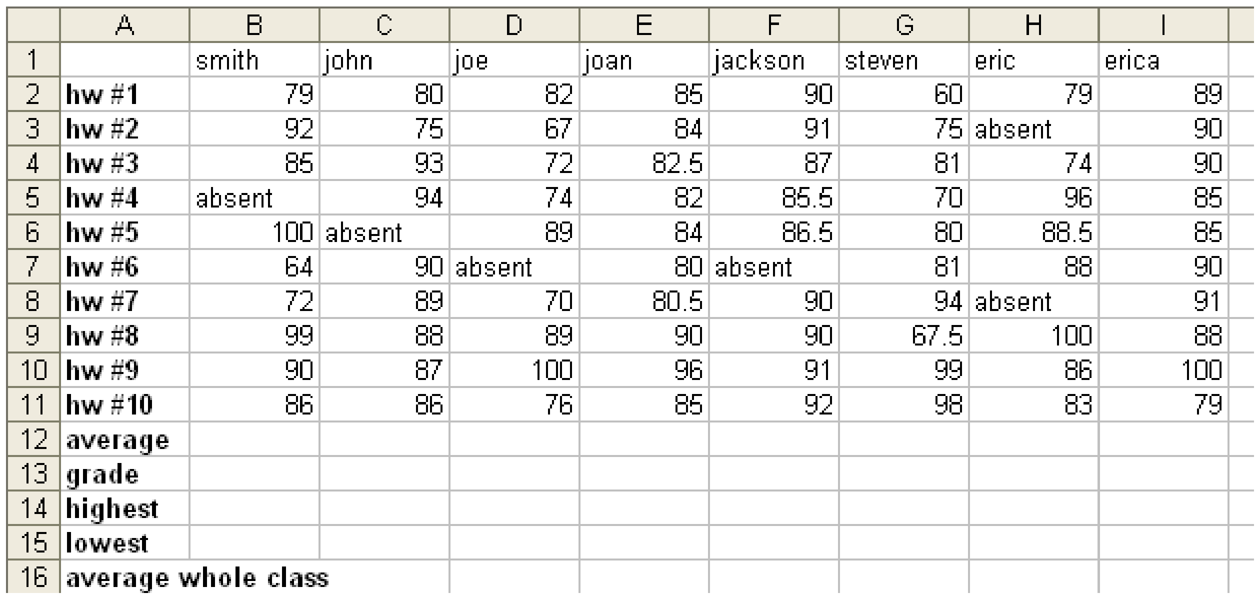
The above teaching scope has been applied since fall 2009. Both lecture and labs were both taught in a computer lab where the students could practice on computer what they had just learned. The instructor took 5.5 weeks to cover VBA and 4.5 weeks to teach MATLAB. The instructor had sufficient time to go through all the course materials while the students also had enough time to practice their programming skills through in-class exercises, homework assignments and labs.

3. Using modular programming strategy

Engineers are modelers [10]. A programming course should train the students to gain the ability to develop algorithmic solutions to engineering problems [8]. In the engineering world, a complex project is always divided into small pieces. To finish the whole project, individual engineers must work on each piece and the team lead must be responsible for integrating every piece together. The same strategy has been successfully used in software industry and is called “modular programming”, by which a complicated programming assignment is divided into small, manageable procedures [11]. To use the modular programming strategy, a main procedure always needs to be created to coordinate and integrate the work of all individual procedures.

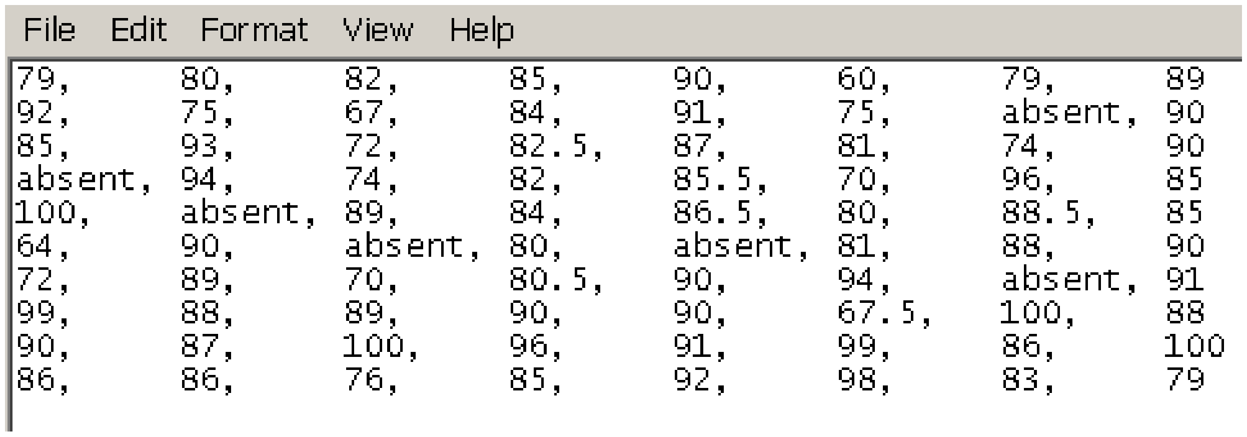
An efficient code development process is helpful to reduce the frustration of the students during programming. It can also improve their confidence and learning interests. By using the modular programming strategy, a clear and concise programming structure can be easily built. This structure breaks a complex problem into small pieces and provides a roadmap to solve that problem efficiently. Therefore, the author started using the modular program strategy in ENGR 266 from fall 2009.

To help students better understand the modular programming concept, an in-class project was designed as shown in Figure 1. In this figure, there are 8 mock students in a mock class. Each mock student has 10 grades. The project was to use VBA to obtain the average and letter grades of each student, the highest and lowest average grades among the students, and the average grade of the class. If the student is “absent”, the grade for that specific homework would be 0. All the grades should be imported from a plain text file (as shown in Figure 2) to the Excel spreadsheet. The letter grades should then be output to another plain text file after all VBA programs are executed.



	A	B	C	D	E	F	G	H	I
1		smith	john	joe	joan	jackson	steven	eric	erica
2	**hw #1**	79	80	82	85	90	60	79	89
3	**hw #2**	92	75	67	84	91	75	absent	90
4	**hw #3**	85	93	72	82.5	87	81	74	90
5	**hw #4**	absent	94	74	82	85.5	70	96	85
6	**hw #5**	100	absent	89	84	86.5	80	88.5	85
7	**hw #6**	64	90	absent	80	absent	81	88	90
8	**hw #7**	72	89	70	80.5	90	94	absent	91
9	**hw #8**	99	88	89	90	90	67.5	100	88
10	**hw #9**	90	87	100	96	91	99	86	100
11	**hw #10**	86	86	76	85	92	98	83	79
12	**average**								
13	**grade**								
14	**highest**								
15	**lowest**								
16	**average whole class**								

Figure 1. An in-class project to apply modular programming concept.

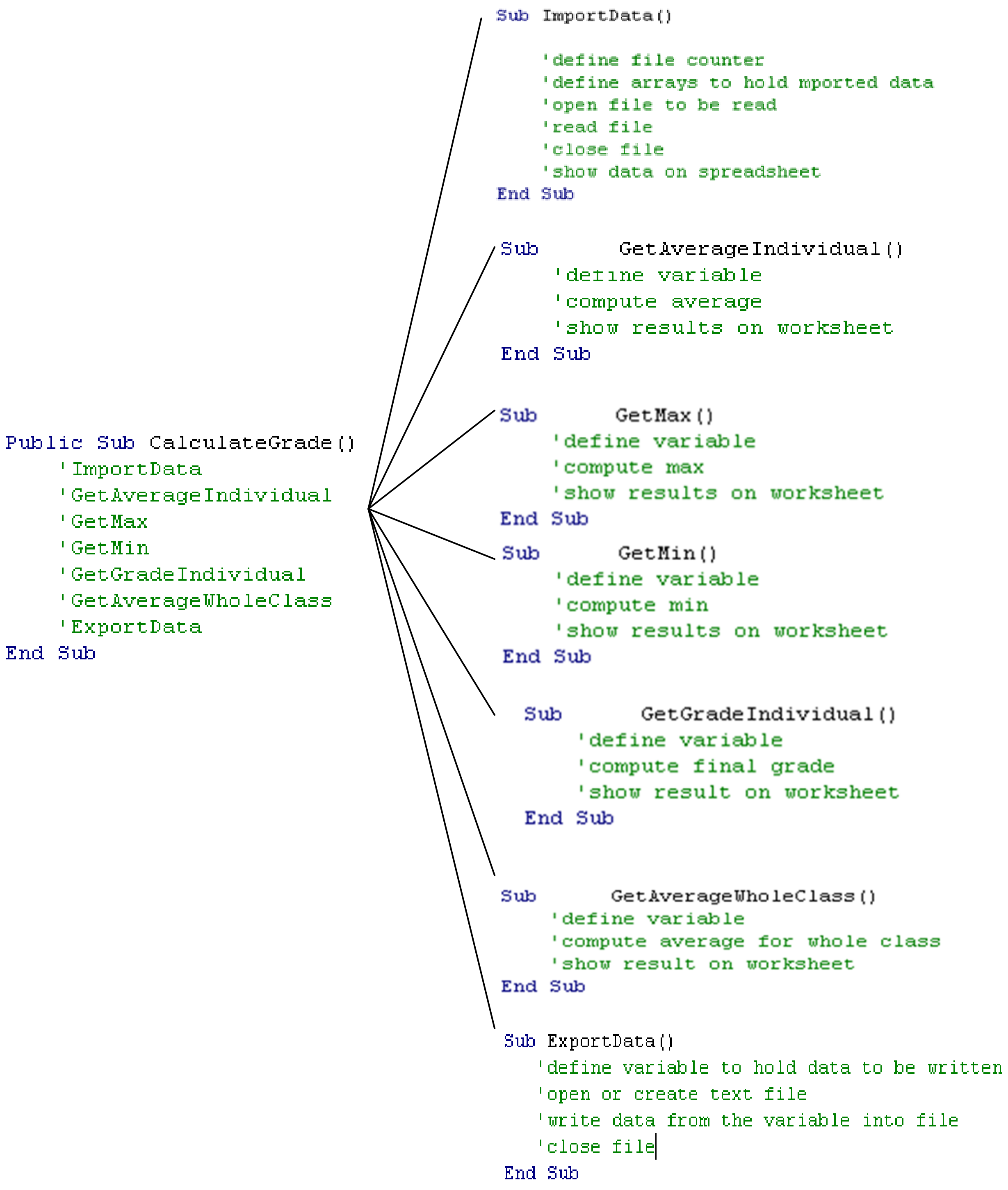


```
79,      80,      82,      85,      90,      60,      79,      89
92,      75,      67,      84,      91,      75,      absent,  90
85,      93,      72,      82.5,    87,      81,      74,      90
absent,  94,      74,      82,      85.5,    70,      96,      85
100,     absent,  89,      84,      86.5,    80,      88.5,    85
64,      90,      absent,  80,      absent,  81,      88,      90
72,      89,      70,      80.5,    90,      94,      absent,  91
99,      88,      89,      90,      90,      67.5,    100,     88
90,      87,      100,     96,      91,      99,      86,      100
86,      86,      76,      85,      92,      98,      83,      79
```

Figure 2. Data to be imported to Excel spreadsheet by using VBA.

In the second week of the term, the instructor worked together with the students to design the programming structure for the project. As shown in Figure 3, there are 8 procedures in the project module. A main procedure (CalculateGrade) coordinates and integrates all the work to be done by 7 other procedures. Among these 7 procedures, ImportData procedure imports data from a plain text file to the Excel spreadsheet, GetAverageIndividual procedure calculates the average for each mock student, GetMax procedure gets the highest average grade, GetMin procedure obtains the lowest average grade, GetGradeIndividual procedure gives the letter grade to each mock student, GetAverageWholeClass procedure finds the average of the class and ExportData procedure exports the students' grades from the Excel spreadsheet into a plain text file. By the time the related programming topics were covered, the students put the code into each procedure under the guidance of the instructor.

When MATLAB was taught, the students were required to do the same project again by following the same modular structure (as shown in Figure 3) and by using script files, functions files and built-in functions.



```
Public Sub CalculateGrade()
    'ImportData
    'GetAverageIndividual
    'GetMax
    'GetMin
    'GetGradeIndividual
    'GetAverageWholeClass
    'ExportData
End Sub
```

```
Sub ImportData()

    'define file counter
    'define arrays to hold mported data
    'open file to be read
    'read file
    'close file
    'show data on spreadsheet
End Sub
```

```
Sub       GetAverageIndividual()
    'define variable
    'compute average
    'show results on worksheet
End Sub
```

```
Sub       GetMax()
    'define variable
    'compute max
    'show results on worksheet
End Sub
```

```
Sub       GetMin()
    'define variable
    'compute min
    'show results on worksheet
End Sub
```

```
Sub       GetGradeIndividual()
    'define variable
    'compute final grade
    'show result on worksheet
End Sub
```

```
Sub       GetAverageWholeClass()
    'define variable
    'compute average for whole class
    'show result on worksheet
End Sub
```

```
Sub ExportData()
   'define variable to hold data to be written
   'open or create text file
   'write data from the variable into file
   'close file
End Sub
```

Figure 3. A modular structure in VB format for the in-class project.

4. Using a flow chart to model each procedure

A flow chart is an ideal vehicle to visualize fundamental control structures in a program, such as if-elseif-else statement, for-loops, etc. Before programming each procedure (as shown in Figure 3), the students were required to create a flow chart. Then, they were required to follow the chart and put the code into the procedure. Using the flow part made the programming process more efficient because it provided a clear, step-by-step logic on how to solve a problem and reduced the students' time in debugging the code. Figure 4 is an example of the flow chart for GetGradeIndividual procedure, which assigns the letter grade to each mock student. The corresponding VB program is shown in Figure 5.

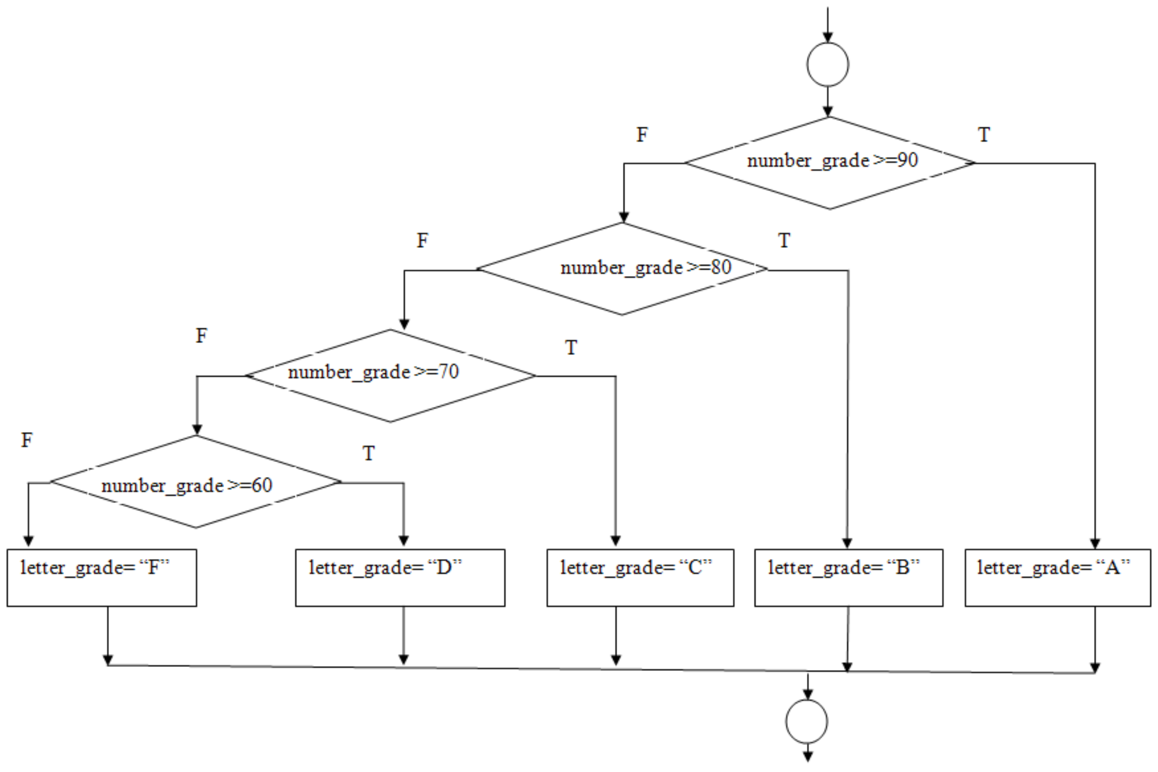
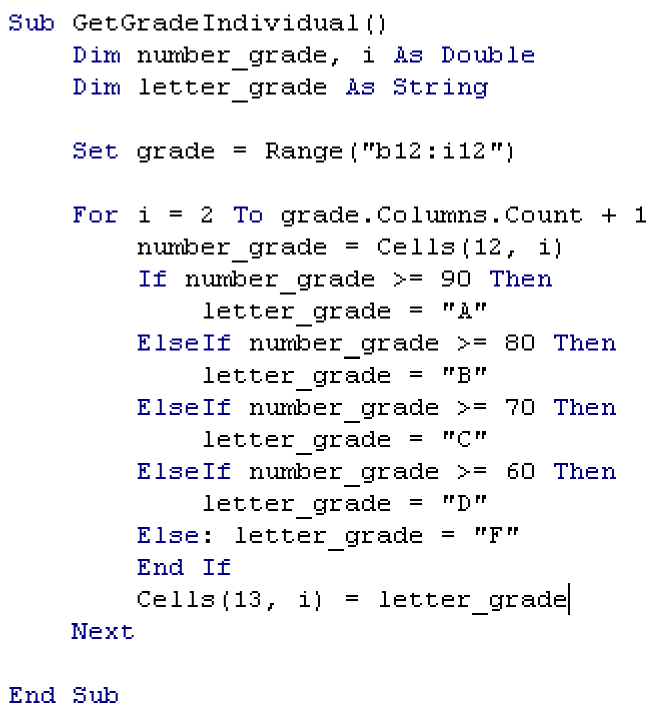


Figure 4. An example of the flow chart used in programming.

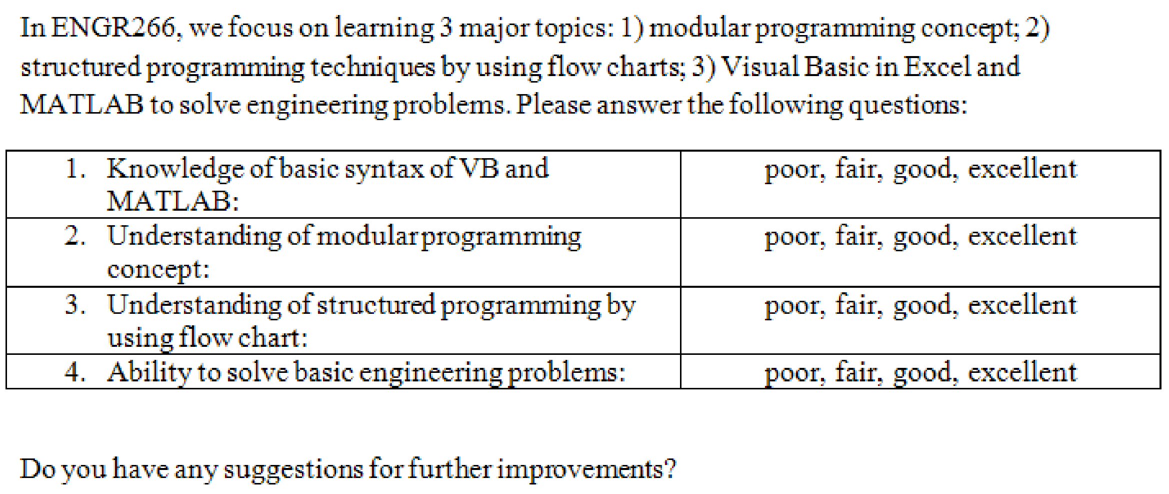


```
Sub GetGradeIndividual()
    Dim number_grade, i As Double
    Dim letter_grade As String

    Set grade = Range("b12:i12")

    For i = 2 To grade.Columns.Count + 1
        number_grade = Cells(12, i)
        If number_grade >= 90 Then
            letter_grade = "A"
        ElseIf number_grade >= 80 Then
            letter_grade = "B"
        ElseIf number_grade >= 70 Then
            letter_grade = "C"
        ElseIf number_grade >= 60 Then
            letter_grade = "D"
        Else: letter_grade = "F"
        End If
        Cells(13, i) = letter_grade
    Next

End Sub
```

Figure 5. The corresponding VB program generated from the flow chart.

5. Assessment from the students

The Individual Development and Educational Assessment (IDEA) (http://www.theideacenter.org/) has been used in ENGR 266 to assess the student learning outcomes. The evaluation score has improved by 30% over the past 2 years. In addition to using IDEA assessment tool, the instructor made a brief survey at the end of each term to evaluate students' confidence in programming skills. The questionnaire is shown in Figure 6:



In ENGR266, we focus on learning 3 major topics: 1) modular programming concept; 2) structured programming techniques by using flow charts; 3) Visual Basic in Excel and MATLAB to solve engineering problems. Please answer the following questions:

1. Knowledge of basic syntax of VB and MATLAB:	poor, fair, good, excellent
2. Understanding of modular programming concept:	poor, fair, good, excellent
3. Understanding of structured programming by using flow chart:	poor, fair, good, excellent
4. Ability to solve basic engineering problems:	poor, fair, good, excellent

Do you have any suggestions for further improvements?

Figure 6. A student survey to assess students' confidence in programming skills.

Among the feedback from the students, only 10% student reported “poor in understanding of structured programming by using flow chart” and 10% student reported “fair in knowledge of basic syntax of VB and MATLAB”. All other students reported either “excellent” or “good” in every question.

The suggestions from the students included:

- “Give more examples of problems we might see in another class and how we might go about solving it using VB or MATLAB”;
- “Give more explanations of how to use flow charts”;
- “Give more walk-through on basic operations”;
- “Demonstrate more flow charts on advance math problems”.

Based on performance in homework assignments, lab assignments, mid-term exam and final exam, 80% of the class met the instructor’s expectation (and they received A’s or B’s as their final grades).

6. Conclusions

Computing has become a fundamental part of engineering education. However, teaching programming skills to freshmen and sophomores has been a pedagogical challenge for many colleges and universities. After years of trial and error, the authors summarized a set of core programming skills that can be grasped in one academic quarter. These skills are taught with the modular programming strategy. Through an in-class project, the students learned how to solve a complicated problem by dividing it into small pieces, addressing each piece by a programming procedure and integrating the intermediate result from each procedure to get the final solution. This pedagogic model makes the computer programming course less challenging for the students. The preliminary result of implementing such a model has been promising. It is believed that with the modular programming strategy and the core programming skills defined in this paper, the students will be able to develop computer programs to solve most engineering problems, such as those in statics, dynamics, thermal dynamics, in their future learning.

Programming is an intensive, hands-on design process. The amount of programming skills of a student is largely dependent on how much programming experience she or he has had. The engineering students should get some basic programming experiences as early as possible. Therefore, the authors are communicating with the Klamath Union High School to seek an opportunity to offer a VBA computer programming course as a high school senior elective. The authors are planning to offer this course to a small class of 3 to 5 senior students (or high school teachers) in fall 2011. Meanwhile, the authors will design a set of assessment methods to evaluate the student learning outcomes for the course. After that, the course materials and teaching methods will be revised and improved. It is hoped that a regular VBA programming class will be offered to the high school students in fall 2012.

References

1. Mohammad H.N. Naraghi and Bahman Litkouhi, 2001, “An Effective approach for teaching computer programming to freshman engineering students”, Proceedings of the 2001 American Society for Engineering Education Annual Conference & Exposition.
2. Wangping Sun, 2007, “A new paradigm to improve computer education for engineering students: applying industry-based software development cycle into programming practices”, Proceedings of the 2007 American Society for Engineering Education Annual Conference & Exposition.
3. Ramzi Bualuan, 2006, “Teaching computer programming skills to first-year engineering students using fun animation in MATLAB,” Proceedings of the 2006 American Society for Engineering Education Annual Conference & Exposition.
4. David P. Miller, 2004, “Using robotics to teach computer programming & AI concepts to engineering students”, Proceedings of the 2004 Association for the Advancement of Artificial Intelligence Symposia.
5. James D. Bowen, 2004, “Motivating civil engineering students to learn computer programming with a structural design project,” Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition.
6. Kathleen Harper, 2009, “Comparing the use of a graphical programming language to a traditional text-based language to learn programming concepts in a first-year course”, Proceedings of the 2009 American Society for Engineering Education Annual Conference & Exposition.
7. Gregory Bucks and William Oakes, 2010, “Integration of graphical programming into a first-year engineering course,” Proceedings of the 2010 American Society for Engineering Education Annual Conference & Exposition.
8. Scott J. Schneider, 2005, “Developing an introductory software programming course for engineering students”, Proceedings of the 2005 American Society for Engineering Education Annual Conference & Exposition.
9. Bruce E. Dunne, Andrew J. Blauch and Andrew Sterian, 2005, “The case for computer programming instruction for all engineering disciplines”, Proceedings of the 2005 American Society for Engineering Education Annual Conference & Exposition.
10. Jeff Joines, 2007, “Computer-based modeling for engineers using Excel and VBA”, Proceedings of the 2007 American Society for Engineering Education Annual Conference & Exposition.
11. Steven Chapra, 2010, Introduction to VBA for Excel, p. 38, Prentice Hall, ISBN 970-0-13-239667-7.