The Case for Computer Programming Instruction for All Engineering Disciplines

Bruce E. Dunne, Andrew J. Blauch, Andrew Sterian

School of Engineering, Grand Valley State University

Introduction

There is no debate that computers are ubiquitous tools for the engineer in training as well as the practicing engineer. In addition to being able to use a computer as a tool, we propose that engineers of all disciplines, and not just those studying electrical and/or computer engineering, should be able to program computers. A student body familiar with computer programming enables a more thorough treatment of advanced courses as well as yielding graduates with valuable skills that they will need in industry.

Despite the fact that many engineering programs have altogether dropped teaching computer programming to their non-ECE majors, other programs have continued to require an introductory programming course at the freshman level for all engineering majors[1,2,3,4,5]. A variety of platforms are used in these courses, from the traditional general-purpose programming languages like C, C++, or Java to programmable engineering tools such as MATLAB, Mathcad, and even Excel. We present our case for keeping programming as a key introductory course in an engineering curriculum. Our program utilizes a common course thread for the first two years of study. Having a student population already programming-literate enables us to conduct significant course projects that go beyond simulation to involve actual design and build.

Our experiences indicate that teaching C programming is a good choice for engineering students. In order to justify our choice, we describe the evolution of programming instruction in our curriculum from low-level languages to higher-level languages to object-oriented approaches. We had found that teaching a low-level language has limited scope (i.e., it must be processor specific) and consumes too much class time while teaching an object-oriented language yields limited skills. Others have shared our concern that an introductory programming course using Java to develop GUI's, for example, does not develop necessary skills in algorithmic problem-solving and subprogram concepts[6,7,8]. Teaching C provides the foundation for the diverse programming skills required in our advanced course offerings in all areas of engineering. We also describe how fluency in C affords the practicing engineer a range of programming skills that are easily extendable.

We further discuss the benefits of possessing fundamental programming knowledge for the practicing engineer in industry. The majority of our graduates are tasked with writing programs in a variety of programming languages and environments at some point in their careers. For example, mechanical engineers often use MATLAB for modeling and analysis while manufacturing engineers are frequently involved in programmable logic controller (PLC) programming and creating scripts for automated interfaces. We describe how our close

connections with local industry influenced our decision to teach programming to all engineers, and in particular, teach C. Finally, we include results from our assessment survey that indicate the usefulness of proficiency in programming for the graduating engineer and for the industries that employ them.

Programming in the Curriculum

In this section we describe how programming is integrated into courses throughout the engineering curriculum.

Underlying Philosophy

One of the key aspects of the engineering program at Grand Valley State University is the engineer first, discipline second philosophy. Our goal is to teach students to be an engineer first, before focusing on a specific discipline. All pre-engineering majors are required to take a series of fundamentals in engineering courses before being admitted into the engineering program, thus creating the foundation all engineers should have. The upper-level discipline-specific courses build upon this foundation.

Another key aspect of the engineering program at GVSU is the design and build philosophy. Theory put into practice is an important component in all of our engineering courses. Students are often required to not only analyze and design, but to also implement and build. Several courses even involve industry related projects in addition to the industry sponsored senior projects. Having all students programming-literate enables us to conduct significant course projects that go beyond simulation to involve actual design and build.

All of our engineering students are required to take EGR261 Structured Programming in C as part of the fundamentals in engineering courses. This course goes beyond basic programming knowledge in order to teach programming as a tool that can be used and applied in courses across all disciplines. The next sections discuss how programming is used in courses in each of our engineering disciplines.

Fundamental Engineering Courses

Table 1 lists some of the fundamental engineering courses that incorporate programming, and are required for all engineering disciplines. In EGR101, the students learn about basic CNC programming using G Codes. In EGR261, they are taught how to write well-structured programs in C. In EGR226, students program a microcontroller using C, providing them with their first taste of embedded programming[9]. The senior projects (EGR485 and EGR486) are multi-disciplinary industry-based projects, often times involving the programming of a microcontroller, PLC, or DSP. Interface and integration issues are easier to handle because of the common programming foundation all students have received.

Computer Engineering Courses

Table 2 lists some of the computer engineering courses that incorporate programming. EGR474 is a manufacturing engineering course available to computer and electrical engineering students as an elective.

Table 1: Programming Incorporated into Engineering Courses for All Disciplines

Course	Programming Language(s)
EGR101: Computer Aided Design and Manufacturing	CNC (G Codes)
EGR226: Introduction to Digital Systems	C (68HC11)
EGR261: Structured Programming in C	C
EGR485: Senior Project I EGR486: Senior Project II	C, Assembly, PLC, DSP

Electrical Engineering Courses

Table 3 lists some of the electrical engineering courses that incorporate programming. Many of the electrical engineering courses use MATLAB as an analysis and design tool. Students who have a foundation in structured programming are able to quickly pick up the MATLAB programming environment. A significant number of the courses also involve embedded programming (using assembly or C). In EGR326, students design and build a complete microcontroller-based system. In EGR423, students work with interfacing DSP software and hardware. In EGR424, students develop drivers for an embedded application. The experience of writing structured programs (EGR261) and applying it to an embedded controller (EGR226) allows the development of more mature projects in these courses than would otherwise be possible.

Table 2: Programming Incorporated into Computer Engineering Courses

Course	Programming Language(s)
EGR323: Signals and Systems Analysis*	MATLAB
EGR326: Advanced Digital Systems*	C, Assembly (AVR)
EGR423: Digital Signal Processing Systems*	MATLAB
EGR424: Design of Microcontroller Applications*	C, Assembly (68HC11)
EGR474: Systems Integration†	C
CS162: Computer Science I	Java
CS163: Computer Science II	Java
CS263: Data Structures and Algorithms	Java
CS451: Computer Architecture	Assembly
CS452: Operating Systems Concepts	C
CS459: Embedded Computer Systems	C

* Discussed as part of the electrical engineering courses.
† Discussed as part of the manufacturing engineering courses.

Mechanical Engineering Courses

Table 4 lists some of the mechanical engineering courses that incorporate programming. In EGR345, students develop C programs as part of the laboratory experience to control an embedded system[10]. In addition, they write C programs and use MATLAB to implement numerical algorithms for solving non-linear systems of equations and simulating dynamic systems. A foundation in structured C programming is essential to the successful and efficient completion of these assignments. Some of the other mechanical engineering courses, such as EGR350, use MATLAB as an analysis tool.

Table 3: Programming Incorporated into Electrical Engineering Courses

Course	Programming Language(s)
EGR323: Signals and Systems Analysis	MATLAB
EGR326: Advanced Digital Systems	C, Assembly (AVR)
EGR415: Communications Systems	C, MATLAB, DSP
EGR423: Digital Signal Processing Systems	MATLAB
EGR424: Design of Microcontroller Applications	C, Assembly (68HC11)
EGR455: Automatic Control Systems	MATLAB
EGR474: Systems Integration*	C

* Discussed as part of the manufacturing engineering courses.

Table 4: Programming Incorporated into Mechanical Engineering Courses

Course	Programming Language(s)
EGR345: Dynamic System Modeling and Control	C (AVR), MATLAB
EGR350: Vibration	MATLAB
EGR450: Manufacturing Control Systems*	PLC

* Discussed as part of the manufacturing engineering courses.

Manufacturing Engineering Courses

Table 5 lists some of the manufacturing engineering courses that incorporate programming. EGR450 and EGR474 deal with the programming of various industrial controllers and networks. EGR450 focuses on the design and implementation of manufacturing control systems typically involving PLCs, while EGR474 focuses on the integration of systems across local area networks. A strong programming foundation allows these courses to focus on application and implementation issues.

Table 5: Programming Incorporated into Manufacturing Engineering Courses

Manufacturing Engineering Courses	Programming Language(s)
EGR345: Dynamic System Modeling and Control*	C (AVR), MATLAB
EGR450: Manufacturing Control Systems	PLC
EGR474: Systems Integration	C

* Discussed as part of the mechanical engineering courses.

Evolution of Programming Instruction

In this section we describe the evolution of programming instruction for our undergraduate engineering students over the last 6 years.

The Curriculum in 1999

With our common course thread in place in 1999, all engineering students were exposed to programming. All freshman engineering students were required to take CS162 (Computer Science I), an introductory programming course. This course assumed no prior programming knowledge. CS162, which continues to exist to this day, has kept up with trends in the software development industry by using first C, then C++, and finally Java as the main language of

instruction (C++ was used in 1999). In the sophomore year, the next engineering course that involved programming instruction was EGR226 (Introduction to Digital Systems). This course introduced students to basic digital systems concepts (Boolean algebra, combinational logic, sequential logic, microprocessor architecture) and Motorola 68HC11 microcontroller programming in assembly language.

Following the sophomore year, the use of programming in the different disciplines diverged. For electrical and computer engineering (ECE) students, the next course to make use of programming was EGR326 (Advanced Digital Systems). In this course, students were required to design a microcontroller-based system and demonstrate its operation by writing an assembly-language program. The 68HC11 and Microchip PIC microcontrollers were used. EGR326 was followed by EGR424 (Design of Microcontroller Applications), which made further use of the 68HC11 to provide instruction in advanced microcontroller interfacing, also in assembly language. For mechanical and manufacturing engineering students, no programming was required until EGR450 (Manufacturing Control Systems), a senior-level elective course that introduced PLC's and their programming using ladder logic.

There were several perceived problems with the above curriculum structure:

- Learning and programming in assembly language is difficult and time-intensive, thus the laboratory and project assignments in EGR226, EGR326, and EGR424 were necessarily limited in scope and practical application.
- A large part of the instructional time in EGR226 was spent simply on presenting the many instructions and addressing modes available on the 68HC11, as opposed to presenting fundamental concepts in digital systems.
- Because of the lack of practical laboratories in EGR226, students in the mechanical and manufacturing emphases did not see the applicability of the concepts in EGR226.
- The programming foundation of CS162 was not used.

In addition, we frequently interact with local employers through advisory board meetings and on-site visits as part of our integrated co-operative education program. While our local employers were generally satisfied with our students, we heard several concerns that our students did not have adequate programming skills. Our largest employers of ECE students specifically required expertise in the C and Ada programming languages, while employers of mechanical and manufacturing engineering students required expertise in PLC programming and data acquisition/industrial automation systems such as LabWindows.

It was clear that we needed to revise our approach to programming instruction. We chose to use the C programming language as the first language of instruction. We felt the language would provide a good foundation in learning algorithm design and implementation, one that was general enough to be easily augmented to the specific programming languages or environments in demand by the various disciplines.

The Curriculum in 2000

In the fall of 2000, EGR226 was modified to use the C programming language (the freeware GCC compiler targeted for the 68HC11) instead of assembly language. The intention was to

build upon the programming foundation provided by CS162 to spend less class and laboratory time on programming instruction and more time on practical applications of digital systems.

This change identified a new problem: the CS162 course was not preparing students for writing structured programs in C. It was expected that by using C++, students would have mastered basic concepts in structured programming such as loops, function calls, parameter passing, etc. In practice, since CS162 was made available as a general education course, the course presented an object-oriented view of programming and focused on connecting together pre-written classes in a visual development environment without much actual code development. As a result, a large amount of class time in EGR226 was spent actually teaching the basics of structured programming (in C). In 2000, the problem was further exacerbated when Java became the main language of instruction for CS162. This change moved the course farther away from supporting the engineering curriculum since Java was not a viable language at the time for low-level microcontroller programming (and, in our opinion, continues to remain so).

By switching to C from assembly language in EGR226, we were able to increase the relevance of the course, but not yet to the point where it could properly support subsequent courses.

The Curriculum in 2002

In response to the perceived deficiency in the preparation afforded by CS162, the Engineering and CSIS departments collaborated to offer a new course, EGR261 (cross-listed as CS261), Structured Programming in C. The purpose of this course was to introduce structured programming to students with no programming experience, ensuring that students had ample opportunity to write code and become proficient in the basics of variables, expressions, control structures, function calls, parameter passing, and most importantly, algorithm design and implementation.

This new course represented a fundamental separation in programming education between the Engineering school and CSIS. For engineering students, we decided to go “back to C” after having left it for C++ and Java, while CSIS students continue to learn Java as their primary language. We believe this separation is consistent with that of industry. Graduates of our CSIS program are frequently expected to work with large, object-oriented software systems while our Engineering graduates will work with smaller, non-object-oriented programs for such tasks as automating industrial processes, creating simple GUI’s for data acquisition and control, and developing embedded system software (where C continues to be the dominant language).

At the same time, EGR226 continued with instruction in C. After two years of having done so, however, we found that upper-level courses and the capstone senior project course had greatly increased their usage of both C programming and programming in more specialized languages such as MATLAB and LabWindows. Whereas in the past any student work requiring programming in anything but assembly language would have required significant just-in-time instruction, the solid foundation in C provided by EGR226 enabled upper level courses to expect programming proficiency.

The Curriculum in 2004

By the sophomore year, we now expect all students to have taken EGR261 thus we no longer provide basic C instruction in EGR226. We continue, however, to teach more advanced aspects of C such as bit-wise manipulation, accessing memory at predefined locations, and interrupt handling. By spending less time on C instruction, we are able to give students a deeper education in digital systems and allow them to work on a mini-project requiring the design of a C program for controlling a mobile robot.

Assessment Survey

Survey Participants

In order to support our position regarding the importance of teaching programming to all engineers, we surveyed both managers and engineers of local industry and our current student body. West Michigan, in which Grand Valley State University is located, has an industry base that includes automotive OEM, aerospace, furniture manufacturing and other light industry. Managers from all of these areas were included in our survey, with a breakdown of industry participation given in Table 6. Students surveyed include juniors, seniors and graduate students. Students of all four majors offered (CE, EE, ME, and MFG) are represented; a breakdown of those included is given in Table 7. Note: because of our extensive co-op program, all undergraduate students surveyed have at least one semester of industry co-op experience with the seniors having completed a full year of co-op employment.

Table 6: Engineering Professionals Participating in Assessment Survey

	Industry							
Field	All	Aerospace	Automotive	Boiler	Construction	Consumer	Furniture	HVAC
Management – Electrical	5	3	1		1			
Technical – Electrical	4	3	1					
Management – Mechanical	6		2	1	1		1	1
Technical – Mechanical	3		2			1		
Unspecified	1		1					
Total	19	6	7	1	2	1	1	1

Survey Questions

Different surveys were used for our industry and student populations, although they are very similar in theme. The survey questions were designed to query the respondents as to their opinion on the importance of programming skills for all engineering disciplines as well as their chosen field. Additionally, we asked the respondents to indicate the appropriateness of teaching the C language as a first programming course. In the student survey, we also asked how important it was to have previous programming knowledge when taking a course that required

programming. Furthermore, we wanted to know the students opinion regarding programming instruction in courses that used programming in class activities: is the course better if students come prepared with knowledge? The industry and student survey questions are found along with the results in Table 8 and Table 9, respectively.

Table 7: Student Participation in Assessment Survey

Engineering Major	Class			
	All	Juniors	Seniors	Graduate
Computer	6		6	
Electrical	11		8	3
Mechanical	23	2	21	
Manufacturing	5	1	4	
Total	45	3	39	3

All survey questions were designed to be answered with an increasing numeric rating of 1–5, with a rating of 1 to indicate "Not At All", a rating of 3 to indicate "Somewhat" up to a rating of 5 to indicate "Very". The respondents were allowed to enter "No Comment" if they did not want to reply to a particular question. Additionally, both student and industry surveys included a statement to clarify our general definition of "programming skills" as "the ability to understand, write, debug or test any of the following: source code (C, Java, etc.), macros (script and batch files) or interpretive languages (MATLAB, CNC, etc.)". Finally, the respondents were given the option to add comments to their rating responses.

Tabulation of Survey Results

The survey results are given for the entire set of respondents as well as for certain sub-groups. For the industry survey, the results were grouped according to respondents who were in management areas versus those who are technical contributors. This was done to highlight the possible perceived differences in the importance of programming between these two groups. To similarly highlight differences, the results were also separately calculated for those in the electrical engineering field versus those in the mechanical engineering field. For the student survey, the results were grouped according to engineering major. We wanted to determine if the results for the majors that are traditionally seen as having a strong programming component (CE, EE) differed significantly from those that do not (ME, MFG). The survey results are given in Table 8 for the industry participants and in Table 9 for the students.

Survey Discussion

In both the industry and student surveys, the first question measures the perceived importance of programming skill for all types of engineers. The overall score for both the industry group and the student group rated this skill as important, which affirms our premise. Our industry technical respondents scored this question lower than the management respondents; arguably this may be

due to the notion that managers tend to view technical resources with a wider, longer-term viewpoint while practicing engineers tend to consider the "job at-hand". Respondents in the traditional programming fields (CE, EE) rated the importance of this ability higher than those in other fields (ME, MFG). This result is expected since it is probable that these engineers use their programming skills with greater frequency than those in other fields. Those who rated this question lower indicated that rather than programming, "general PC skills such as PowerPoint and Word are of much more importance".

Table 8: Industry Professionals Survey Results

Question	All	Mgmt.	Tech.	Elect.	Mech.
How important is it for all engineers to have programming skills?	3.5	3.8	3.1	3.7	3.4
How important is it for engineers in your industry to have programming skills?	3.5	3.6	3.4	4.0	3.1
How important is it for engineers in your department to have programming skills?	3.4	3.1	4.0	4.0	3.1
How familiar are you with the C programming language?	2.6	2.5	2.7	2.8	2.7
Is C an appropriate language for teaching basic programming skills?	3.2	3.1	3.6	4.1	2.7

Table 9: Student Survey Results

Question	All	CE	EE	ME	MFG
How important is it for engineering students of all emphasis areas to have programming skills?	3.6	4.0	3.9	3.4	3.0
How important is it for engineering students in your emphasis area to have programming skills?	3.6	4.8	4.6	2.9	3.0
How important is it to have good programming skills in order to succeed as an engineering student?	3.7	3.7	4.1	3.7	3.0
How important is it to have programming skills prior to taking a class versus learning programming as a part of the class?	3.4	3.3	2.9	3.9	2.6
How important are programming skills in industry?	3.1	3.8	3.7	2.8	2.6
How important is the ability to program in terms of succeeding in your engineering career?	3.0	4.2	3.7	2.5	2.2
Is C an appropriate language for teaching basic programming skills?	3.8	4.2	4.2	3.6	3.2

The second question considers the importance of programming skills for students, engineers and managers in their respective fields. Across all groups, we would expect a high degree of agreement with the first question, which is exactly the case. Considering the sub-groups, those

with a traditional programming background responded higher than others, which follows the results noted above regarding the first question.

The third question on the industry survey narrows the focus downward from a particular industry to the specific engineering group employing our respondents. Interestingly, the importance of programming skills is seen to be more important by the technical group than the manager group, which is the opposite of question 1. Again, the possibility that engineers tend to focus on the "job at-hand" may explain this result since most likely programming skills are applied routinely while the managers view the project from a more peripheral, customer-focused viewpoint.

The third and fourth questions on the student survey seek to qualify the importance of programming while studying engineering. Our survey indicates that engineering students think it is important to have good programming skills to succeed, with the EE students rating this skill the highest. The students indicated that it was somewhat less important to have these skills on the first day of class. Interestingly, the EE students (who rated programming skills highest in terms of importance to their success) indicated that having programming skills on the first day of class is less than somewhat important. This result possibly implies that some level of programming instruction is desired for all courses that require programming.

Students think that programming is somewhat important for industry in general and for success in their own careers, as measured by the overall responses to questions five and six. As expected, students in CE and EE think programming is more important than ME or MFG students. Of interest, as gauged by comparing the student survey with the industry survey, these ME and MFG students will come to regard programming skills as more important as their careers evolve.

The last question in both surveys asks about the appropriateness of teaching introductory programming using C. Both groups indicate the appropriateness of C, however, the students favored teaching C more so than the industry respondents, most likely due to the fact that many of their courses exploit a C background. Both students and industry technical contributors in the electrical and computer areas view C as more than appropriate as an introductory language. Industry participants who indicated that C was not an ideal first programming language offered that "some concepts of programming are too cerebral to just jump into C" and instead think that Visual Basic is more appropriate.

Anecdotal Comments

While not part of our formal survey discussed above, we think the following statements from two mechanical engineering students are relevant. These students were both on their first co-op assignment (having just completed their sophomore year) and were required to keep a journal of their activities and their impressions of the engineering profession. Both of the quotations below were found in their journals, and are unsolicited comments.

> "The biggest thing I learned this week was how to write the code for posting results to our website. The class I thought would never have any real value, computer programming, has paid off."
>
> "I'm learning a big part of being a design engineer is not only knowing how to use the software, but actually understanding how the software is written. There are so many

options and different files you can manipulate in order to make the program run how you want, and I'm now glad we had to take CS261."

Conclusion

In this paper, we described the benefits of programming instruction for engineers of all disciplines. We explained how courses further along in the curriculum are able to take advantage of the programming background and offer more in-depth projects. We described how our programming instruction evolved, and then took a back-step and returned to C as the main language for introductory programming instruction. Finally, we presented the results of our industry and student surveys that echoed our position regarding the importance of programming for all engineers as well as the appropriateness of the C language.

Bibliography

1. J.C. Musto, J.H. Lumkes Jr. and W. Carnell, "A Freshmen Programming Course for Mechanical Engineers Using Mechatronics Applications," *Proceedings of the 2004 ASEE Annual Conference and Exposition*, Salt Lake City, Utah.

2. D.E. Clough, "ChE's Teaching Introductory Computing to ChE Students -- A Modern Computing Course with Emphasis on Problem Solving and Programming," *Proceedings of the 2002 ASEE Annual Conference and Exposition*, Montreal, Quebec, Canada.

3. J.D. Bowen, "Motivating Civil Engineering Students to Learn Computer Programming With a Structural Design Project," *Proceedings of the 2004 ASEE Annual Conference and Exposition*, Salt Lake City, Utah.

4. K-Y. D. Fan and D.I. Schwartz, "First Programming Course in Engineering: Balancing Tradition and Application," *Proceedings of the 2003 ASEE Annual Conference and Exposition*, Nashville, Tennessee.

5. D. Budny, L. Lund, J. Vipperman and J.L. Patzer II, "Four Steps to Teaching C Programming," *Proceedings of the 32nd Frontiers in Education Conference*, Boston, Massachusetts. (2002)

6. O.L. Astrachan, "Non-Competitive Programming Contest Problems as the Basis for Just-in-time Teaching," *Proceedings of the 34th Frontiers in Education Conference*, Savannah, Georgia. (2004)

7. W. Campbell and E. Bolker, "Teaching Programming by Immersion, Reading and Writing," *Proceedings of the 32nd Frontiers in Education Conference*, Boston, Massachusetts. (2002)

8. J.J. McConnell and D.T. Burhans, "The Evolution of CS1 Textbooks," *Proceedings of the 32nd Frontiers in Education Conference*, Boston, Massachusetts. (2002)

9. A. Blauch and A. Sterian, "A Practical Application Digital Systems Course For All Engineering Majors," *Proceedings of the 2002 ASEE Annual Conference and Exposition*, Montreal, Quebec, Canada.

10. H. Jack and A. Blauch, "A Modeling and Controls Course using Microcontrollers," *Proceedings of the 2004 ASEE Annual Conference and Exposition,* Nashville, Tennessee.

BRUCE E. DUNNE
Bruce E. Dunne is currently an Assistant Professor in the Padnos College of Engineering and Computing at Grand Valley State University. He received his B.S.E.E. and M.S.E.E. from the University of Illinois at Urbana-Champaign and Ph.D. in Electrical Engineering from the Illinois Institute of Technology. His interests include digital signal processing and communications systems.

ANDREW J. BLAUCH
Andrew J. Blauch is currently an Assistant Professor in the Padnos College of Engineering and Computing at Grand Valley State University. He received his B.S. in Electrical Engineering from Messiah College, M.S. in Electrical and Computer Engineering from Carnegie Mellon University, and Ph.D. in Electrical Engineering from the Pennsylvania State University. He has taught courses on digital systems, microprocessors, and controls.
<http://claymore.engineer.gvsu.edu/~blaucha>

ANDREW STERIAN
Andrew Sterian is an Assistant Professor in the Padnos College of Engineering and Computing at Grand Valley State University. He received his B.A.Sc. in Electrical Engineering from the University of Waterloo, Canada, and the M.S.E. and Ph.D., both in Electrical Engineering, from the University of Michigan. His interests include embedded system hardware and software design, mechatronics in education, and signal processing.
<http://claymore.engineer.gvsu.edu/~steriana>