Engineering Educators Bringing the World Together
2025 ASEE Annual Conference & Exposition
Palais des congrès de Montréal, Montréal, QC • June 22–25, 2025
ASEE
Paper ID #45596

Exploring Problem Solving with C++ Across All Engineering Disciplines

Dr. Edris Ebrahimzadeh P.E., University of South Alabama

Dr. Edris Ebrahimzadeh, P.E., is an Assistant Professor of Instruction in the Department of Chemical and Biomolecular Engineering at the University of South Alabama. He earned his Ph.D. in Chemical Engineering from Brigham Young University (BYU). His teaching portfolio includes thermodynamics, separations, process control, and engineering economy. Dr. Ebrahimzadeh's interests lie in engineering education, with a focus on implementing active learning strategies to enhance student engagement and comprehension in the classroom.

Exploring Problem Solving with C++ Across All Engineering Disciplines

Abstract

Freshman engineering students globally often embark on their programs with varying degrees of computer programming experience, encountering a diverse array of languages such as Fortran, Visual Basic, C, JAVA, Python, and MATLAB in their initial year. Each language brings its own unique advantages and limitations. This paper investigates the introduction of a C++ course, emphasizing how it can elevate the critical thinking skills of engineering students across diverse educational and cultural settings. By drawing on case studies and examples from various international academic environments, this paper aims to showcase how C++ enhances problem-solving and adaptability to global engineering challenges.

Unlike the higher-level nature of Python and MATLAB, C++ necessitates a more contemplative approach to code writing, cultivating a deeper understanding. The additional cognitive effort invested in learning C++ proves beneficial when students later engage with other programming languages. Furthermore, this paper illustrates how the object-oriented features of C++ can systematically address engineering problems in mechanical, civil, chemical, and electrical engineering disciplines on a global scale.

Acknowledging the evolving educational landscape, we propose a specially tailored C++ course designed for engineering disciplines. This course is uniquely structured to incorporate global engineering scenarios, fostering intercultural learning and collaboration. Additionally, the curriculum bridges engineering and computer science, providing a valuable pathway for students contemplating a shift to computer science. The paper encompasses a detailed course description, sample problems, and comparative results, showcasing the course's ability to enhance critical thinking and address international educational goals.

Keywords: computer programming, C++, object-oriented, education, curriculum, first-year engineering, international engineering education

Introduction

MATLAB and Python are among the most widely used programming languages in engineering education. However, students often engage with these languages in two distinct ways: some use them primarily for numerical analysis and linear algebra, while others leverage them as full-fledged programming languages [1]. The former group tends to write scripts for specific problem-solving, whereas students with prior coding experience develop functions and broader computational solutions.

Students proficient in programming quickly recognize the advantages of structured problem-solving, allowing them to tackle complex engineering challenges more effectively than those who view MATLAB solely as a mathematical tool. Programming is fundamentally about logical thought organization and deterministic process description, skills that extend beyond specific languages.

As students advance in their careers, the limitations of MATLAB—particularly in terms of execution speed and data I/O capabilities—become apparent. Tasks such as constructing high-resolution images from raw data or handling complex binary file operations often require a lower-level, high-performance language like C++. While C++ is not always a prerequisite, it serves as a versatile engineering toolbox, offering capabilities that go beyond MATLAB's specialized applications.

The choice of C++ in this study is based on its unique advantages in engineering applications. While Python and MATLAB are essential for data analysis and simulations, C++ provides deeper insight into memory management, object-oriented programming, and low-level system operations—skills crucial for engineering disciplines. Additionally, many industry-standard tools, such as finite element analysis (FEA) software and embedded systems, are built using C++, making it an essential language for engineering problem-solving in real-world applications. This course is designed to equip students with computational thinking skills that are directly applicable to practical engineering challenges.

Enhancing Global Context in Engineering Education

To showcase the adaptability of this C++ course across diverse educational settings, we examined its implementation in various international contexts. In Germany, C++ is a core component of mechatronics and control systems courses, where students develop software for automation and robotics [2]. In Japan, C++ plays a significant role in manufacturing and automotive applications, with universities integrating industry projects into coursework [3]. In India, where many engineering students specialize in software development, C++ is taught alongside data structures and algorithms, helping students build strong computational problem-solving skills applicable to engineering simulations [4]. These examples highlight C++'s continued relevance across global engineering disciplines and how this course can be tailored to regional industry needs.

In a globalized engineering environment, critical thinking and adaptable problem-solving strategies are essential. Integrating C++ into engineering curricula provides students with universally applicable tools that enable them to tackle real-world challenges in diverse cultural and geographical contexts. This aligns with international educational goals, equipping students for multinational engineering collaborations. By solving C++-based problems tailored to global scenarios, students develop the ability to think beyond local contexts and consider the broader implications of their solutions.

A fundamental difference between C++ and higher-level languages like MATLAB lies in variable handling and type definition. In MATLAB, division operations automatically account for data types, so the operation $\frac{x}{y}$ where x=6 and y=4, directly produces 1.5. In contrast, C++ requires explicit type definition or type casting to achieve the same result. If both variables are declared as integers, C++ performs integer division, returning to 1 instead of 1.5. This strict data type enforcement enhances students' understanding of programming logic and computational precision, skills that are valuable when learning other programming languages.

A January 2023 survey ranking programming languages by popularity in search engine queries provides further insight into C++'s relevance [5,6]:

Language	Percentage
Python	16.36%
C	16.26%
C++	12.91%
Java	12.21%
C#	5.73%
Visual Basic	4.64%
JavaScript	2.87%
SQL	2.50%
Assembly language	1.60%
PHP	1.39%
Others	23.53%

Notably, the C/C++/C# group collectively accounts for 35% of all programming-related searches, reinforcing the continued demand for C++ in both academic and industrial settings.

Object-Oriented Programming for Global Challenges

One of the defining features of C++ is its object-oriented programming (OOP) paradigm. Computer programs generally follow either a procedure-oriented or object-oriented approach. In procedure-oriented programming, tasks are structured into functions, with the focus on processes and operations [6]. In contrast, object-oriented programming centers around objects, which encapsulate both data and behavior.

An object can represent anything tangible-a physical entity or conceptual model-that can interact with other objects in a program. This approach enhances modularity and reusability, allowing programmers to define classes as blueprints from which multiple objects can be created with minimal modification. These advantages make OOP widely adopted in engineering applications, reducing development time and improving code efficiency.

Each object has attributes (descriptive properties) and behaviors (actions it can perform or respond to) [7]. For example, consider a Rectangle class in an engineering simulation:

- A landscaping company could use the object to estimate sod-laying costs.
- A fencing company could use the same object to calculate fencing expenses.

By defining a single class, programmers can apply the same object in multiple problem-solving scenarios, reinforcing code reusability and efficiency.

The next section explores real-world engineering applications of C++ and OOP, demonstrating how class-based design enhances problem-solving skills and computational thinking in engineering disciplines.

International Collaboration and Cultural Sensitivity in Engineering Education

Integrating intercultural learning and collaboration into engineering curricula enhances both technical proficiency and cultural adaptability, equipping students for a globalized workforce. These approaches are particularly relevant when teaching C++, a language that requires a strong grasp of fundamental programming principles and structured problem-solving.

A key distinction of C++ lies in its emphasis on data types and memory management, foundational skills essential for real-world engineering applications. Mastering these concepts enables students to tackle complex computational challenges across various engineering disciplines.

The following section explores practical engineering scenarios where C++ and object-oriented programming play a crucial role in efficient problem-solving.

1. Course description

This study was conducted over two semesters with 27 first-year engineering students. As a mandatory course in the freshman engineering curriculum, all enrolled students participated, representing a diverse mix of disciplines, including mechanical, civil, electrical, and chemical engineering. This diversity ensured that the redesigned curriculum was evaluated across various engineering problem-solving contexts.

Each class accommodated 15–20 students in a computer lab equipped with Visual Studio. The syllabus covered fundamental programming concepts, including variables, control flow, loops, string processing, arrays, object-oriented programming (OOP), and file handling.

At the start of the course, students were introduced to the engineering problem-solving process (Figure 1) [8]., which emphasizes problem identification, analysis, solution development, and presentation—a methodology reinforced throughout the engineering curriculum.

Steps in the Engineering Problem Solving Method

1. Recognize and understand the problem.
2. Accumulate data and verify accuracy.
3. Select the appropriate theory or principle.
4. Make necessary assumptions.
5. Solve the problem.
6. Verify and check results.

Figure 1. Engineering problem solving process.

In the past two semesters, the instructor used zyBooks, an interactive textbook designed for computer science majors. However, its generic problem sets-covering financial scenarios, number patterns, digit counting, date formatting, and course grades-lacked an engineering-specific focus [9].

In Semester 1, the course relied solely on the textbook's examples and exercises. In Semester 2, the curriculum was redesigned to emphasize engineering problem-solving and system analysis. The revised activities incorporated engineering-specific challenges from diverse fields, including engineering economics, thermodynamics, statics, materials and energy balance, and circuit analysis. These changes aimed at enhancing students' critical thinking and analytical skills in real-world engineering contexts

The grading criteria of the course are as follows,

Participation Activity (PA)	15%
Lab Activities	20%
Challenge Activity (CA)	15%
Projects (three)-equally weighted	50%
Total	**100%**

- **Participation Activities** (PAs): Interactive, animation-based exercises completed before class to build foundational knowledge.
- **Lab Activities**: Conducted in pairs, focusing on collaborative problem-solving in real-world engineering applications.
- **Challenge Activities** (CAs): More advanced versions of PAs, requiring deeper exploration and complexity.

- **Projects**: Replacing traditional tests, students complete three major projects, applying C++ to interdisciplinary engineering problems. This approach reinforces practical applications and interdisciplinary learning

2. Methodology

The study evaluates the effectiveness of integrating C++ into an engineering curriculum, comparing student performance in a traditional programming course versus a redesigned course with an engineering-focused C++ approach. The study was conducted over two semesters with all students enrolled in the course participating.

Sampling and Group Assignment

Students were not randomly assigned to the traditional versus redesigned course, as each semester had only one section offered. However, both cohorts had similar demographics, including academic background and prior programming exposure. The course instructor remained the same to minimize teaching variability. Future studies will seek to implement a randomized or quasi-experimental design for improved data robustness.

Data Collection and Assessment

Student performance was assessed using a combination of:

- **Participation Activities (PAs)** – interactive exercises completed before class.
- **Lab Activities** – real-world engineering problems solved collaboratively.
- **Challenge Activities (CAs)** – advanced versions of PAs that required deeper problem-solving skills.
- **Project-Based Assessment** – three interdisciplinary programming projects.

Rather than relying solely on final grades, we examined project-based assessments as a proxy for critical thinking development. While direct critical thinking evaluations were not conducted, student reflections and instructor observations indicated improved problem-solving engagement. Future iterations will incorporate validated assessment tools such as pre/post-tests or structured rubrics for critical thinking evaluation.

3. Sample Course Exercises

Several exercises are taught in the class to demonstrate and stress the importance of engineering problem solving integrated with *object-oriented* features of C++. Some of them are:

1. Finding the moment of inertia for single and composite areas
2. Locating the centroid of single and composite bodies
3. Cartesian vector formulation of moment of a force about a point and about an axis
4. Performing operations on complex numbers
5. Finding the specific heat of ideal gases
6. Calculating the relative humidity

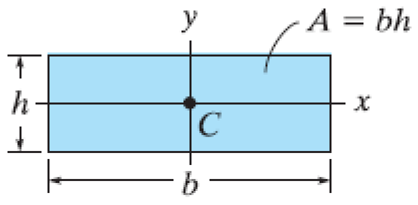
3.1. Moment of inertia of single and composite areas-Statics

The moment of inertia is a crucial geometric property used to assess a structural member's strength or determine the location of a resultant pressure force in submerged plates. It is defined as the second moment of an area about an axis, calculated by squaring the distance from the axis to each area element. Standard moment of inertia values for basic shapes (triangles, rectangles, circles, etc.) are commonly introduced in Statics courses.

To automate these calculations, students can develop a *MomentOfInertia* class in C++, defining key attributes and methods. The class performs three primary functions:

1. Initialize private data members (e.g., length and width).
2. Assign values from user input or a program.
3. Calculate and return moments of inertia along the x- and y- axes using standard formulas:

$$I_x = \frac{1}{12} b \times h^3, \quad I_y = \frac{1}{12} h \times b^3$$



Rectangular area

By executing these tasks, the *MomentOfInertia* object becomes a versatile tool for computing moments of inertia for various shapes, contributing to the efficiency and modularity of the overall program.

The partial implementation of the *MomentOfInertia* class is outlined below. This class encapsulates the necessary attributes and methods to facilitate the calculation of area moments of inertia for different shapes. This partial class includes methods for setting dimensions and calculating moments of inertia for both x- and y- axes. You can integrate this class into a program by initializing an instance of *MomentOfInertia*, setting dimensions, and then utilizing the calculated values as needed.

```
#include<cmath>

//declaration section
class MomentOfInertia
{
public:
   MomentOfInertia();
   void setDimensions(double, double);
   double calcIx();
   double calcIy();

private:
   double length;
   double width;
};

//Implementation

MomentOfInertia::MomentOfInertia() :   length (0.0), width(0.0) {}
double MomentOfInertia::calcIx() { return (1.0 / 12) * length * pow(width,3); }

double MomentOfInertia::calcIy() {  return (1.0 / 12) * width * pow(length, 3); }
```

This object-oriented approach enhances modularity and efficiency, allowing students to extend the class to compute moments of inertia for composite areas. Additionally, the class can be expanded to determine moments of inertia about inclined axes (Figure 2) using trigonometric transformations:

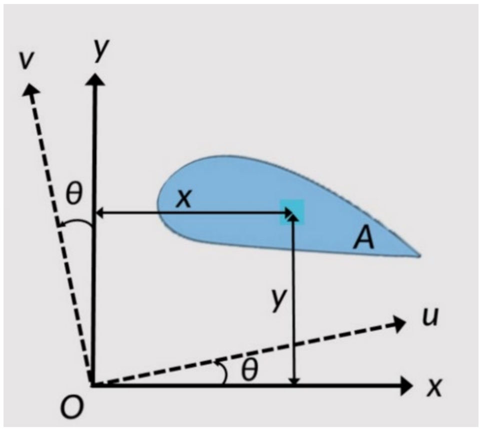


Figure 2. Moment of Inertia for an area about inclined axes

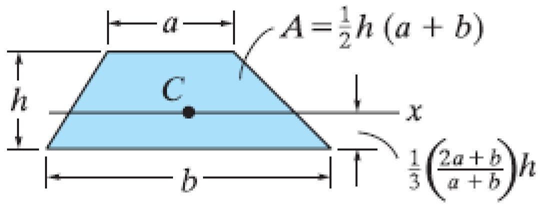
$$I_u = \frac{I_x + I_y}{2} + \frac{I_x - I_y}{2} cos\, 2\theta - I_{xy} \, sin\, 2\theta \qquad \text{Eq. 1}$$

$$I_v = \frac{I_x + I_y}{2} + \frac{I_x - I_y}{2} cos\, 2\theta + I_{xy} \, sin\, 2\theta \qquad \text{Eq. 2}$$

3.2. Centroid of single and composite bodies-Statics

The centroid serves as the geometric center of a body, coinciding with the center of mass or gravity only when the material composing the body is uniform or homogeneous [10]. Formulas for calculating the centroid of regular areas (such as circular arc segments, quarters, semicircle arcs, trapezoids, semi-parabolas, and parabolic areas) can be found at the end of Statics textbooks.

To simplify these calculations, one can create a *Centroid* class that prompts for (a) the shape of the area and (b) the required dimension(s) to calculate the centroid. For example, for a trapezoidal area, the centroid can be determined by providing the magnitudes of the two bases and the altitude.


a
$A = \frac{1}{2}h\,(a+b)$
h
C
x
$\frac{1}{3}\left(\frac{2a+b}{a+b}\right)h$
b

In engineering mechanics, there are instances where finding the centroid of a composite body is necessary, consisting of a series of connected "simpler" shaped bodies. After dividing the body into simpler composite parts, the centroid of the composite body can be calculated using the following formulas:

$$\bar{x} = \frac{\sum \tilde{x} W}{\sum W}, \quad \bar{y} = \frac{\sum \tilde{y} W}{\sum W}, \quad \bar{z} = \frac{\sum \tilde{z} W}{\sum W} \qquad \text{Eq. 3}$$

Here $\tilde{x}, \tilde{y},$ and $\tilde{z}$ represent the coordinates of the centroid of each composite part of the body, which can be calculated by utilizing the *Centroid* class.

3.3. Moment of a Force about a Point and an Axes: Vector Formulation-Statics

The moment of a force **F** about a point O (Figure 3), is determined by using the vector cross product:

$$\boldsymbol{M}_o = \boldsymbol{r} \times \boldsymbol{F} \qquad \text{Eq. 4}$$

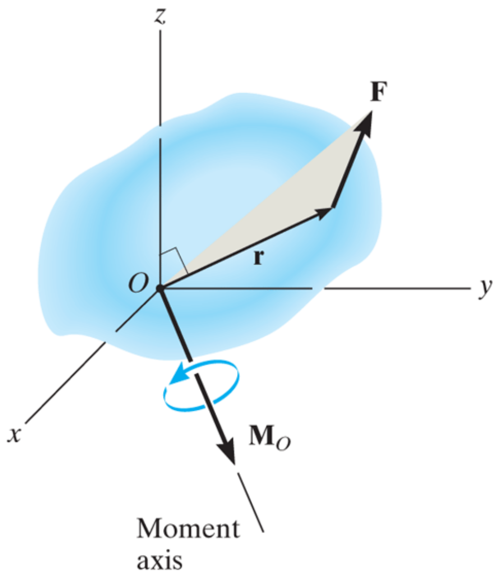


Figure 3. Moment of a force about a point-Vector formulation

Here $\boldsymbol{r}$ is the position vector from O to any point on the line of action of $\boldsymbol{F}$ [10]. In Cartesian form:

$$\boldsymbol{M}_o = \boldsymbol{r} \times \boldsymbol{F} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ r_x & r_y & r_z \\ F_x & F_y & F_z \end{vmatrix} \qquad \text{Eq.5}$$

The resulting components $(M_{o_x}, M_{o_y}, M_{o_z})$ define the moment in 3D space, and its magnitude is given by:

$$|\boldsymbol{M_o}| = \sqrt{M_{o,x}^2 + M_{o,y}^2 + M_{o,z}^2} \qquad \text{Eq. 6}$$

The direction angles relative to the coordinate axes are:

$$\alpha = cos^{-1}\left(\frac{M_{o,x}}{|\boldsymbol{M_o}|}\right), \qquad \beta = cos^{-1}\left(\frac{M_{o,y}}{|\boldsymbol{M_o}|}\right), \quad \gamma = cos^{-1}\left(\frac{M_{o,z}}{|\boldsymbol{M_o}|}\right) \qquad \text{Eq. 7}$$

For moments about a specific axis a (Figure 4), the determinant form is:

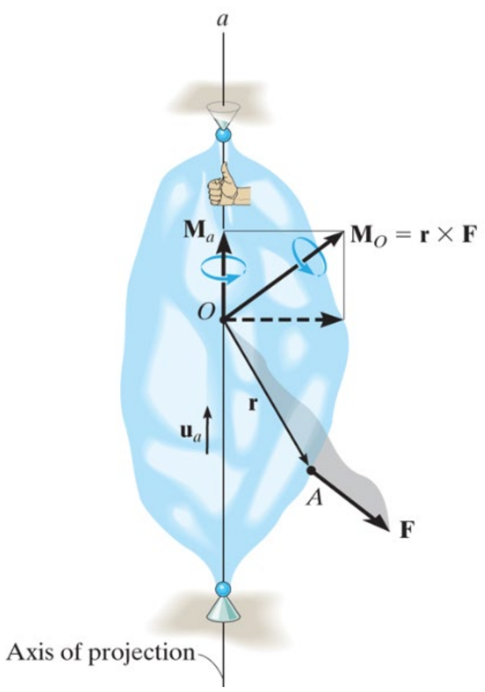


Figure 4. Moment of a force about an axes

$$\boldsymbol{M}_a = \boldsymbol{u}_a \cdot (\boldsymbol{r} \times \boldsymbol{F}) = \begin{vmatrix} u_{a_x} & u_{a_y} & u_{a_z} \\ r_x & r_y & r_z \\ F_x & F_y & F_z \end{vmatrix} \qquad \text{Eq. 8}$$

where u_a is the unit vector along axis a. These calculations can be implemented in a *MomentVector* class to determine moments about a point or an axis.

3.4. Complex Number-AC Circuit Analysis

In the field of Electrical Engineering, complex numbers play a crucial role in the AC Circuit Analysis course. Complex numbers are extensively used by students to represent phasors, analyze impedance, and perform calculations related to alternating current circuits. The phasor representation simplifies the analysis of sinusoidal signals, making it a powerful tool for understanding voltage and current relationships, as well as power calculations in circuits subjected to varying frequencies.

A complex number in mathematics is defined as $a + bi$, where a defines the real part of the number and b is the imaginary part [11]. The letter i represents the square root of -1 (which means i^2 is -1). To perform basic operations on complex numbers, one can create a *Complex* class.

1. Addition:

$$(a_1 + b_1 i) + (a_2 + b_2 i) = (a_1 + a_2) + i(b_1 + b_2) \qquad \text{Eq. 9}$$

2. Subtraction**:**

$$(a_1 + b_1 i) - (a_2 + b_2 i) = (a_1 - a_2) + i(b_1 - b_2) \qquad \text{Eq. 10}$$

3. Multiplication**:**

$$(a_1 + b_1 i) * (a_2 + b_2 i) = (a_1 a_2 - b_1 b_2) + i(a_1 b_2 + a_2 b_1) \qquad \text{Eq. 11}$$

4. Division**:**

$$(a_1 + b_1 i)/(a_2 + b_2 i) = \frac{(a_1 a_2 + b_1 b_2) + i(-a_1 b_2 + a_2 b_1)}{a_2^2 + b_2^2} \qquad \text{Eq. 12}$$

The *Complex* class will prompt for:

- The operation to be performed.
- The real and imaginary parts of the host object.
- The real and imaginary parts of the new complex number.

3.5. Specific Heats of Ideal Gases-Thermodynamics

The change in internal energy or enthalpy for an ideal gas during a process from state 1 to state 2 is determined by the following integrations:

$$\Delta u = u_2 - u_1 = \int_1^2 c_v(T)dT \;\; (\frac{kJ}{kg}) \qquad \text{Eq. 13}$$

$$\Delta h = h_2 - h_1 = \int_1^2 c_p(T)dT \;\; (\frac{kJ}{kg}) \qquad \text{Eq. 14}$$

To carry out these integrations, relations for c_v and c_p as functions of temperature are required [12]. At low pressures, all real gases approach ideal-gas behavior, and their specific heats depend on temperature only. The specific heats of real gases at low pressures are known as ideal-gas specific heats or zero-pressure heats, often denoted as c_{p_0} and c_{v_0}. Accurate analytical expressions for ideal-gas specific heats, based on direct measurements or calculations from statistical behavior of molecules, are available as third-degree polynomials:

$$\bar{c_p} = a + bT + cT^2 + dT^3 \qquad \text{Eq. 15}$$

where T is in Kelvin, c_p in kJ/kmol.K, and values for a, b, and c for each substance are tablualted.

3.5.1. Specific Heat Class:

One can create a *SpecificHeat* class that will:

1. Prompt for the substance and the temperature range for specific heat calculation.

2. Check if the temperature range is within the valid range from the specific heat tables; otherwise, display an error.
3. Set up the polynomial with obtained constants and perform integration to obtain a fourth-order polynomial as the antiderivative.
4. Apply the Fundamental Theorem of Calculus to calculate the change in enthalpy of the substance for the given temperature range.
5. Calculate $\bar{c_v}$ from $\bar{c}_p$ using $\bar{c_v} = \bar{c_p} - R \ \ (\frac{kJ}{kg.K})$ (where R is the universal gas constant).
6. Allow the user to specify whether to find the change in internal energy or enthalpy, and whether these changes are to be found on a mass or molar basis.

This class provides a comprehensive tool for thermodynamic calculations, automating the process and offering flexibility in choosing the basis for energy changes.

3.6. Relative Humidity-Material and Energy Balance

Relative humidity (RH) is a vital consideration in the Materials and Energy Balance course, particularly in processes involving drying, evaporation, and reactions influenced by water content. A profound understanding and accurate accounting for relative humidity are crucial in material balance calculations, where moisture content significantly affects mass transfer and energy considerations in various engineering applications. Furthermore, in contexts such as heat exchangers and environmental control systems, engineers must factor in the impact of relative humidity on heat transfer properties and overall energy balances within a system.

3.6.1. Calculation of Relative Humidity

The relative humidity (RH) at sea level can be calculated using the dry-bulb temperature (T_{db}) and the wet-bulb temperature (T_{wb}), with temperatures in degrees Celsius [13].

The formula is given by:

$$RH = \frac{VP}{SVP} \times 100 \qquad \text{Eq. 16}$$

where VP is the vapor pressure and SVP is the saturated vapor pressure. Vapor pressure (VP) is determined by the formula:

$$VP = e^{\frac{16.78T_{wb} - 116.9}{T_{wb} + 237.3}} - 0.066858(1 + 0.00115T_{wb})(T_{db} - T_{wb}) \qquad \text{Eq. 17}$$

And saturated vapor pressure (SVP) is calculated using:

$$SVP = e^{\frac{16.78T_{db} - 116.9}{T_{db} + 237.3}} \qquad \text{Eq. 18}$$

To facilitate user convenience, a *RelativeHumidity* class can be designed to find RH when provided with dry-bulb and wet-bulb temperatures. The class includes a unit conversion section for temperatures from Celsius to Fahrenheit. This not only saves time for the user but also enables quick sensitivity analysis to understand how changes in T_{db}or T_{wb} impact relative humidity.

4. Results and Discussion

4.1. Impact Evaluation

The data for this study were collected from student projects assigned throughout the semester. Each project was designed to assess students' ability to apply C++ programming concepts to solve engineering-related problems. The grading rubric emphasized logical problem decomposition, code efficiency, debugging capability, and the correct implementation of object-oriented principles. To ensure consistency, all projects were evaluated using standardized rubric, and students received detailed feedback on their approach and problem-solving strategies.

Students were tasked with translating real-world engineering challenges into computational models using C++. For example, they implemented algorithms to compute the centroid of composite bodies, analyzed AC circuits using complex numbers, and modeled thermodynamic processes. These tasks challenged them to break down complex problems, apply object-oriented principles, and refine their solutions through iterative debugging. The depth of their solutions and their ability to troubleshoot errors served as key indicators of their critical thinking development.

The study was conducted over two semesters with a total of 27 students enrolled in the C++ programming course, representing multiple engineering disciplines-including mechanical, civil, and electrical engineering. Although the sample size is limited, the structured, project-based approach provides a meaningful assessment of programming skill development. Future studies will expand the dataset to include additional course iterations and institutions to enhance the generalizability of the findings.

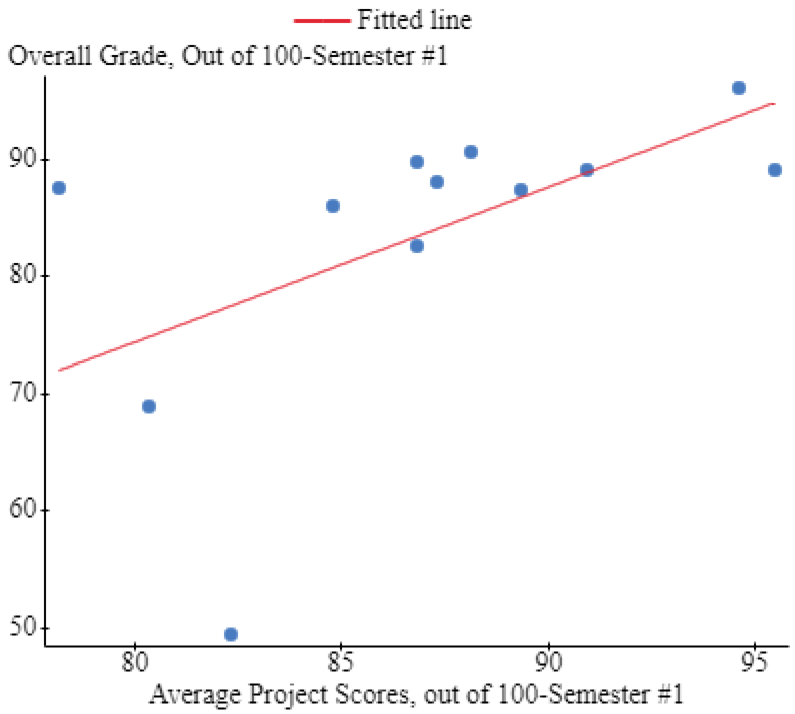
The impact of the new teaching style was evaluated using two primary measures. First, a comparative analysis of the average overall grades between Semester 1 and Semester 2 was performed using a t-Test for two samples with unequal variances (see Table 1).

Table 1.Semester Comparison

	Semester #1	**Semester #2**
Number of Students	12	15
Average Overall Grade	83.8	87.4
Standard Deviation	12.6	8.9
P-value (two-tail)	0.414	

The statistical analysis yielded a p-value of 0.414, indicating no statistically significant difference between the traditional and redesigned courses. Although the redesigned course showed a 3.6-point increase in average grades, this difference may be attributed to variability rather than the strong effect of the new teaching methodology. Moreover, the small sample size (n=27) limits the statistical power of this comparison, making it challenging to detect small but meaningful differences. Importantly, the redesigned curriculum also aimed to enhance students' problem-solving abilities, as evidenced by qualitative improvements in project complexity, debugging skills, and algorithmic efficiency. Future research will incorporate larger sample sizes and additional performance metrics to further assess the curriculum's impact.

Second, the evaluation examined whether the new approach provides a meaningful measure of students' problem-solving abilities—a facet not previously studied—by comparing average project scores with final exam scores. As illustrated in Figure 5 and Figure 6, a positive correlation exists between project scores and final grades; students with higher project scores tend to achieve higher final grades. Although the correlation coefficients of 0.55 and 0.60 indicate a relatively weak relationship, they affirm the value of the new teaching approach as a potential leading indicator of overall student performance.



Sample size: 12
R (correlation coefficient) = 0.55003957
R-sq = 0.30254353
Estimate of error standard deviation: 11.009782

Figure 5. The average project score correlation with the overall grade for semester 1

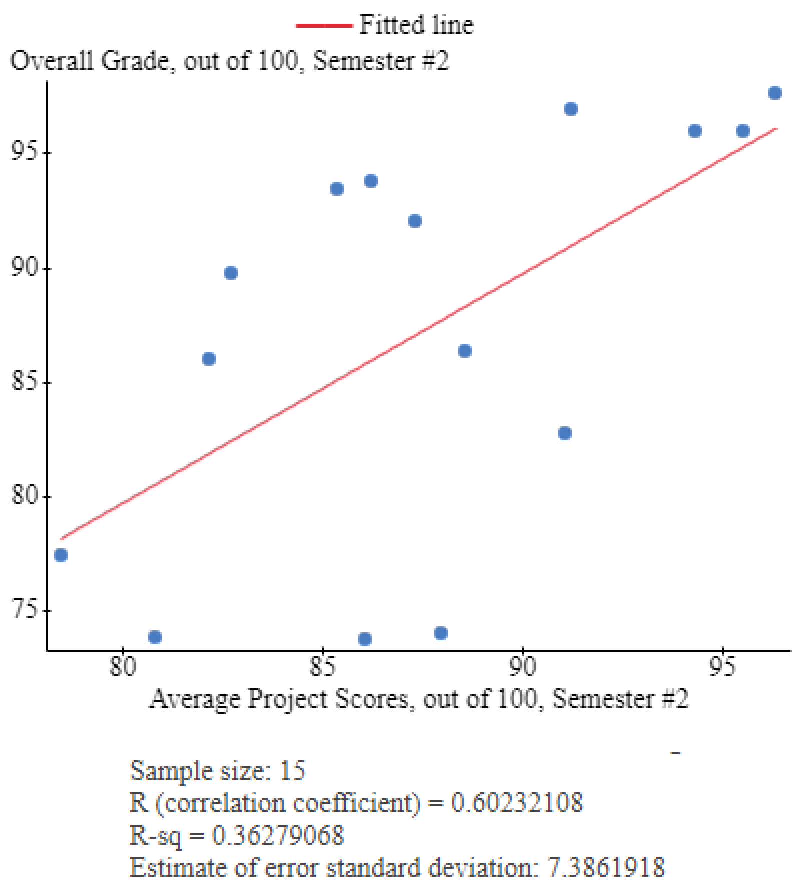


Figure 6. The average project score correlations with the overall grade for semester 2

This correlation analysis provides valuable insights into the effectiveness of the new teaching approach in assessing and enhancing students' problem-solving abilities, highlighting its potential predictive value for overall academic performance.

4.2. Future works

As part of our ongoing commitment to enhancing the learning experience, we plan to broaden and deepen our analysis. To achieve this, future studies will employ larger sample sizes and more robust assessment tools. We will also explore longitudinal studies that track students' programming proficiency over multiple semesters to evaluate long-term benefits. In addition, pre- and post-course coding assessments and structured student interviews will be conducted to directly measure improvements in critical thinking.

The following initiatives are planned:

1. **Extended Cohort Analysis:** We intend to replicate our analysis with successive cohorts, expanding our dataset. This approach will provide a more solid foundation for our conclusions, enabling us to discern trends and patterns over time.

2. **Longitudinal Surveys:** To understand the lasting impact of the new teaching approach, we will conduct follow-up surveys at yearly intervals. These surveys will examine retention rates, sustained academic performance, engagement in engineering-related activities, and eventual career placement. This longitudinal perspective will offer a comprehensive view of the enduring effects of our educational methodologies.

These initiatives are integral to our commitment to continuous improvement and to ensuring the sustained effectiveness of our teaching strategies.

5. Discussion & Learning Outcomes

The redesigned C++ course aimed to enhance engineering students' problem-solving capabilities by integrating computational thinking with real-world applications. While the results showed a modest improvement in average student grades (+3.6 points), statistical analysis did not indicate a significant difference. However, qualitative indicators suggested meaningful improvements in student engagement and analytical thinking.

Student Performance & Project-Based Learning

The correlation analysis between project scores and final grades (r = 0.55, 0.60 for two semesters) suggests that strong performance in engineering-based programming projects is a positive predictor of overall course success. Students who engaged more deeply in problem-solving activities tended to perform better, reinforcing the value of project-based assessments.

Qualitative Feedback & Observations

- Students reported higher confidence in debugging and structuring their code logically.
- Increased engagement was noted in interdisciplinary applications, particularly in mechanical and electrical engineering contexts.
- Instructor observations highlighted better retention of programming concepts compared to previous iterations of the course.

While the lack of a formal critical thinking assessment is a limitation, future studies will integrate qualitative surveys and rubric-based evaluations to measure cognitive skill development

6. Conclusion

In conclusion, this paper has presented a comprehensive exploration of a reimagined teaching approach in engineering education, seamlessly integrating practical problem-solving exercises with object-oriented programming concepts in C++. The incorporation of real-world applications, such as calculating the area moment of inertia, locating centroid, vector form of moment, operations on complex numbers, and computations of specific heat of ideal gases and relative humidity, has effectively demonstrated the versatility and practicality of the proposed pedagogy in connecting theoretical knowledge with real-world engineering scenarios.

The evaluation of the impact of this innovative teaching style has unveiled intriguing insights. While statistical analyses showed a slightly higher average overall grade in the redesigned course compared to the traditional section, the difference, although not statistically significant, suggests the promise of the new approach. Moreover, examining correlations between project scores and final exam grades indicated a logical association, highlighting the new teaching approach as a valuable predictor of students' problem-solving abilities on a global scale.

Looking ahead, future endeavors will extend this analysis to encompass broader international cohorts, ensuring a more comprehensive understanding of the sustained impact of these innovative teaching methodologies. Longitudinal surveys will provide valuable insights into students' academic journeys, participation in global engineering activities, and career placements, contributing to ongoing refinement and optimization of our educational strategies. This commitment to a global perspective underscores our dedication to nurturing well-rounded engineers capable of addressing challenges on a worldwide scale.

References

[1] “Python for Engineers”: The First Course of Computing for a General Engineering Curriculum, Edris Ebrahimzadeh, Nick Safai, ASEE, 2019

[2]https://www.mec.ed.tum.de/en/ais/study-and-teaching/fundamental-and-specialized-courses/industrial-software-development-of-mechatronic-systems-and-implementation-in-c/?utm_source=chatgpt.com

[3]https://japan-dev.com/jobs/mujin/mujin-software-engineering-for-robotics-c-python-lktsvj?utm_source=chatgpt.com

[4] https://www.cse.iitm.ac.in/course_details.php?arg=ODk%3D&utm_source=chatgpt.com

[5] https://www.tiobe.com/tiobe-index/2023

[6] Introduction to C++, zyBooks, 2023

[7] An Introduction to Programming with C++, Diane Zak, Cengage Learning, 8th Edition, 2016

[8] Engineering Fundamentals and Problem Solving, Arvid Eide, Steven Mickelson, Cheryl Eide, Roland Jenison, Larry Northup, 8th Edition, McGraw Hill, 2023.

[9] Animations for Learning: Design Philosophy and Student Usage in Interactive Textbooks, Nikitha Sambamurthy, Alex Daniel Edgcomb, Frank Vahid, ASEE, 2019.

[10] Engineering Mechanics, R.C. Hibbeler, 5th Edition, Pearson, 2022

[11] C Programming: An Object-Oriented Approach, Behrouz A. Forouzan, Richard F. Gilberg, McGraw Hill, 2020

[12] Thermodynamics: An Engineering Approach, Yunus Cengel, Michael Boles, Mehmet Kanoglu,10th Edition, McGraw Hill, 2024.

[13] MATLAB: An Introduction with Applications, Amos Gilat, 6th Edition, John Wiley & Sons, Inc., 2017.