

FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

Lecture56

Lecture 56: Case Study - Mathematical Computation Framework C++

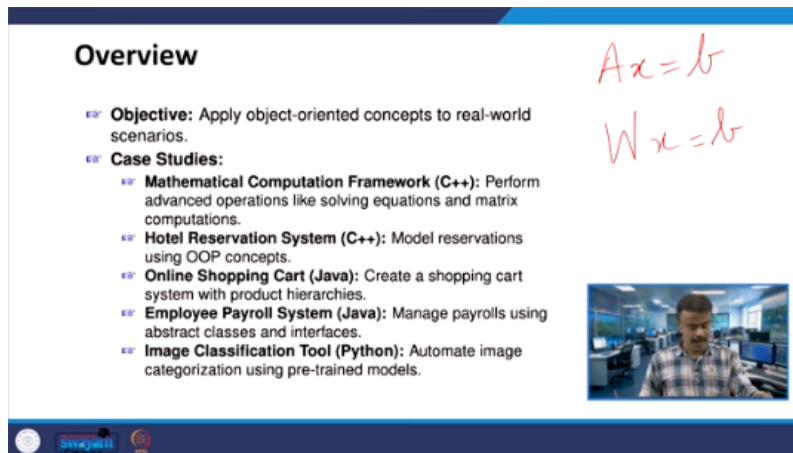
Thank you. Welcome to lecture number 56: case studies of object-oriented programming. So, whatever we have studied till week 11, right? So, we will take some of the important applications, right? So, the important syntax based on that, we will develop some projects, right?

So, that we call it as case studies, right? So, the main objective of this chapter, right? So, we are going to apply all the object-oriented concepts that we have studied in this particular course, right? So, we can apply those into real-world scenarios. For example, I am planning these case studies right in today's lecture. Let us have the mathematical computational framework using C++, right? So, I'm going to cover two examples of C++, two examples of Java, and one example with the help of Python.

So, in today's lecture, we will discuss the mathematical computational framework, right? Mathematical computation framework. So, that means we will perform certain operations like, let us say, solving equations, right? So, suppose you are given a system of equations like this: Ax is equal to b , which is very much useful, right? For problems like machine learning or neural networks. You study like wx is equal to b , solving wx is equal to b , right? System of linear equations. Similarly, the matrix computations, finding out whether the inverse exists or finding out the eigenvalue of a square matrix, right? Or you have to find out whether the matrix is singular or not, right.

So, these kind of operations, so I had given some examples, we will deal, right. And after this, we will see one more example on C++, the hotel reservation system. right with the help of object oriented programming concept the third one is online shopping cart all right so we are going to create a shopping cart system

with the product hierarchies and fourth one is we with the help of java right so the third one is with the help of java fourth one is also with the help of java employee payroll system we will manage the payrolls all right so increment all right or anything there are bonus So, these are all right we will manage the payrolls. So, here what we will do?



Overview

✦ **Objective:** Apply object-oriented concepts to real-world scenarios.

✦ **Case Studies:**

- ✦ **Mathematical Computation Framework (C++):** Perform advanced operations like solving equations and matrix computations.
- ✦ **Hotel Reservation System (C++):** Model reservations using OOP concepts.
- ✦ **Online Shopping Cart (Java):** Create a shopping cart system with product hierarchies.
- ✦ **Employee Payroll System (Java):** Manage payrolls using abstract classes and interfaces.
- ✦ **Image Classification Tool (Python):** Automate image categorization using pre-trained models.

Handwritten in red: $Ax = b$ and $Wx = b$

Video inset showing a man in a blue shirt in a computer lab.

We will use the abstract classes and interfaces. And finally, we are going to wind up with image classification tool using python right. So, most of you might have heard right. So, we are in a ML world right and image processing is one of image processing and computer vision is one of the major area. right where you can apply machine learning.

So, to apply machine learning you should know the basics of python. So, we have seen some python code. So, now you try to apply for the image classification algorithms right. So, the image categorization using pre trained models. So, this will also be interesting problem.

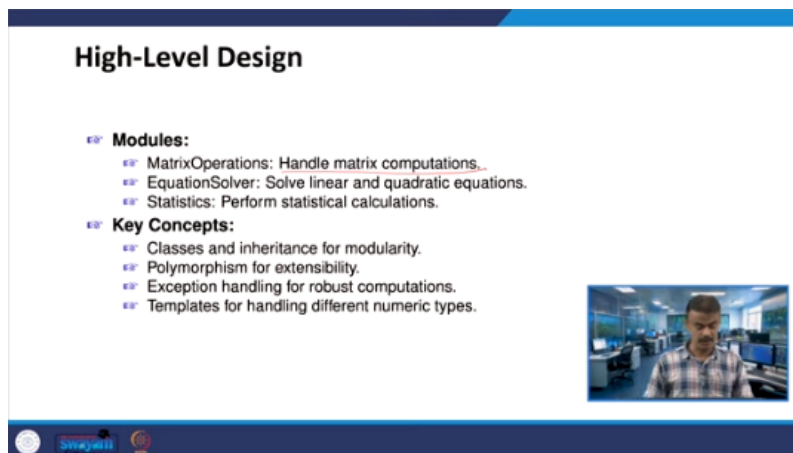
So, we are planning to have five different projects or five different interesting problems. Two with the help of C++, two with the help of Java, and one with the help of Python. So, now in this lecture, we are going to see a mathematical computation framework, right? So, the main objective is to build a modular framework for performing advanced mathematical computations.

So, what do you mean by advanced mathematical computation? Let us say matrix operations: addition, multiplication, inversion, right? Finding out the determinant of the matrix, right? Or we can say how to find out the eigenvalues of

the matrix, eigenvectors of the matrix. So, whatever applications that you know, right?

So, you can even add up, right? So, we are going to see certain basics, let us say addition, multiplication, inversion, etcetera. Similarly, we are going to solve equations: solver linear and quadratic equations or important components, right? The statistical computations, right? So, under this, we can have how to find out the mean if the data is given. Right, a set of vectors is given. How to find the variance? How to find the standard deviation, right? These are all the basic problems, and then one can formulate from there, right? So, this we will see, and the use cases are numerical simulations and solving engineering and scientific problems, right? So, where we are going to use this all, right? So, this we call it as a mathematical computation framework. I will design the modules for matrix operations.

So, this can handle matrix computations, right. Equation solver such as solving linear and quadratic equations and then statistic module is for performing statistical calculations, right. So, these are all the modules, matrix operations, equation solver and statistics, all right. So, the key concepts we are going to use the object oriented point of view we are going to see inheritance where the classes and inheritance right for modularity and we are going to also see polymorphism for extensibility then our good old friend exception handling right divided by 0 right or the negative balance. So, these are all the examples we had seen.



High-Level Design

- ❏ **Modules:**
 - ❏ MatrixOperations: Handle matrix computations.
 - ❏ EquationSolver: Solve linear and quadratic equations.
 - ❏ Statistics: Perform statistical calculations.
- ❏ **Key Concepts:**
 - ❏ Classes and inheritance for modularity.
 - ❏ Polymorphism for extensibility.
 - ❏ Exception handling for robust computations.
 - ❏ Templates for handling different numeric types.

The slide features a blue header and footer. In the bottom right corner, there is a small video inset showing a man in a plaid shirt speaking. The footer contains a small logo and the text 'Sreyas'.

So, we will have we will certainly use exception handling for robust computations and then templates for handling different numeric types, right. So, these are all

some of the concepts we are going to again brush, right. So, that already we had seen several examples of all these important concepts. So, now we will take up with some applications and then we will run the code.

So, now class diagram, right. So, what we are going to have the matrix operations. So, class diagram in the sense. So, we will see we are going to use the base class for matrix computations and then we are going to use the derived class for solving equations right.

That means I am using inheritance now, right? And statistics will be an independent class. So, here you have inheritance, and statistics will be an independent class. So, now the matrix operations. So, this will provide basic matrix operations like addition, subtraction, etcetera. And equation solver.

So, this inherits additional methods for solving equations, right? So, matrix operations I put in the base class, and the equation solver will be in the derived class, which inherits certain additional methods for solving equations, right? And in the independent class, we have statistics. So, that is providing statistical methods as an independent class, right.


So, from the object-oriented programming code point of view, I will start with the matrix operations class, right. So, I will discuss, then finally, we are going to run the program. So, here the first application, let us say vector, right? I am including vector along with IO stream. Here you have the class matrix operations, right. I use vector of vector.

MatrixOperations Class

```
1 #include <vector>
2 #include <iostream>
3
4 class MatrixOperations {
5 public:
6     std::vector<std::vector<double>>> add(
7         const std::vector<std::vector<double>>>& A,
8         const std::vector<std::vector<double>>>& B);
9     std::vector<std::vector<double>>> multiply(
10        const std::vector<std::vector<double>>>& A,
11        const std::vector<std::vector<double>>>& B);
12 };

```

- ✎ Add and multiply matrices.
- ✎ Validate matrix dimensions before operations.



What do you mean by vector of vector? It is a matrix, two-dimensional array, right. So, here the usage of vector. So, double, right. So, vector, here you have the class.

Under vector, you have vector, correct. So, anyway, this is the inbuilt one, not a problem. And then you are going to have a function called add. the member function call add public member function call add correct. So, here we define A matrix A right with the reference operator vector of vector matrix A vector of vector matrix B right.

So, given two matrices A and B when I am calling add this will give addition of two matrices that is the meaning right. So, this will give addition of two matrices and similarly we have multiply right it is multiplication of two matrices right. So, we are passing A and B the reference operator A and B vector of vector that means the matrix A and matrix B similar to the previous explanation right and here your class this is a class the class is over here matrix operations right. So, basically here

we are going to have right this is a function member function so the member function which will be adding two matrices another member function which will be multiplying two matrices right so this is the basic matrix operation class matrix operations class and what we have to do before going into the program we have to validate the dimension right if both the dimensions are same then you can do the addition right so you have to check Either complete or let us say the row, first row, right, all the columns, right, or matching width, right. Suppose A is $n \times n$, B is also $n \times n$, yes, I can do both addition and multiplication, right. If suppose B's dimension is changing, $m \times n$, I cannot do addition, right. So, this test we have to make.

right in in the programming point of view always right the edge case we called as the edge case or the base case we have to be right I mean we have to worry right. So, suppose you have let us say b is $n \times m$ a is $n \times n$ right and b is $n \times n$ addition is not possible whereas, multiplication of a cross b is possible a star b is possible right because one is $n \times n$ another one is $n \times m$. So, the resultant you will get $n \times m$ right. So, these are all the conditions right. So, those who do not have the mathematics background

because it is possible that you can do. So, you have to brush your knowledge on the matrix addition multiplication and the base case or the edge case you have to be very careful. And the next one is equation solver class right. So, we solve the quadratic equation or the linear equation right or linear system right. So, here we are including C math along with IO stream

name of the class is equation solver. So, here you have pair right the pair contains double and double right even you can use a template right assume that you are going to use some other right template we can use we will start using it. So, pair double double right. So, here you have two variables right. So, double and double and the name of the function is

solve quadratic right solve quadratic under the class pair right the inbuilt class pair the pair contains double and double okay. So solve quadratic we are passing a b c right so that means you have the arguments a b c so you pass as a parameter a b c so that means I am solving $ax^2 + bx + c = 0$. So, give any a b c value right. So, give any a b c value I should get the output right. So, that is one which is called as a solve quadratic member function.

Another one is solving linear system right. So, here I have capital A x is equal to small b. A is n cross n matrix x is the vector n cross 1 obviously, b will also be n cross 1 right. So, this is a linear system Ax is equal to b right. So, I have to worry whether A inverse exists where x is nothing, but A inverse b right.

So, here the problem is yeah we can solve by any numerical method let us say Gauss elimination solving the linear system by Gauss elimination right. So, basically we are solving this. So, how I can write in as a function member function. So, you have vectors of double right vector of double that is a class inbuilt class and solve linear system is the name of the member function.

Here you have vector of vector we have seen that the last program a b right the matrix multiplication matrix addition vector of vector double all right. So, a this is nothing, but the coefficients which are nothing, but a matrix. We name it as a coefficient, which is nothing but a matrix, correct? Vector of vector. So, one vector size is m, another one is n. You call it as m cross n. Or if you go for, let us say, square, right?

EquationSolver Class


```

1 #include <iostream>
2 #include <cmath>
3
4 class EquationSolver {
5 public:
6     std::pair<double, double> solveQuadratic(
7         double a, double b, double c);
8     std::vector<double> solveLinearSystem(
9         const std::vector<std::vector<double>>& coefficients,
10         const std::vector<double>& constant);
11 };

```

$ax^2 + bx + c = 0$
 $Ax = b$
 $n \times n \quad n \times 1$
 $x = A^{-1}b$

Solve quadratic equations using the quadratic formula.
 Solve linear systems using Gaussian elimination.



So, you have to be careful. If we are not non-square, let us say a is n cross m , right? So, x should be m cross 1 , right? So, the resultant you will get n cross 1 , correct? Because you have to be careful with the multiplications.

That is what we have done previously, right. So, the resultant B will be n cross 1 . So, that means, yeah, your n and m cannot be same, right. When you are solving system of linear equations, n and m need not be same, right. So, one is coefficients.

That means, A is called as the coefficients, capital A , vector of vector, size in general n cross m . and in particular always right we try to choose the square matrix a is a n cross n if both n and m are equal and then you have the vector right which is constant that is nothing but your b that is a vector and then I am trying to find the solution for x . a and b are the input I am trying to find out the solution for x all right. So, you can solve using Gauss elimination or several other methods all right. So, Gauss elimination Gauss ZDL all right.

So, these are all the approaches that you can use to solve LU decomposition etcetera. The first one is solving quadratic equations using quadratic formula $ax^2 + bx + c$. So, this is the first one that is your first function and this is your second and here you have the second function solve linear system ok. So, this is the equation solver class and the third one is the statistics class right. So, here you have vector numeric C math class is statistics.

So, I am going to perform let us say mean. So, I want to have the mean means I need a array right vector of data right vector of data the reference operator vector

of data the double right it is like this 40.4, 80.7 right. Assume that you have under data like this right. So, that is called as a vector data.

So, for this I am going to compute mean. The member function is mean and for the same data I am going to compute variance or for different data also right. So, here in this case data we are using assume that this is a data right vector this is a vector data is a vector right and for this I am going to compute variance the member function is variance and for this I am going to find out the standard deviation right. So, these are all some simple statistical operations right.

So, this will perform the simple statistical computations and also this will efficiently handle very large data sets right. Let us say we are going to use the file The file contains millions of data. So, for the millions of data is it possible to find out the mean? Is it possible to find out the variance?

The screenshot shows a C++ code editor with the following content:

```
Statistics Class

1 #include <vector>
2 #include <numeric>
3 #include <cmath>
4
5 class Statistics {
6 public:
7     double mean(const std::vector<double>& data);
8     double variance(const std::vector<double>& data);
9     double stddev(const std::vector<double>& data);
10 };

// Perform statistical computations.
// Efficiently handle large datasets.
```

Handwritten annotations in red include:

- Underlines under `<vector>`, `<numeric>`, and `<cmath>`.
- A circle around the `mean` function.
- A circle around the `stddev` function.
- Handwritten text: `data: 20.3 40.4 80.7` with a dashed line below it.
- Red arrows pointing from the handwritten data to the `data` parameter in the function signatures.

A small video inset in the bottom right corner shows a man speaking.

Is it possible to find out the standard deviation? The answer is yes, all right. So, these are all we will start with a basic program right, and then you can work further. In fact, when you are going to the last program, we are going to have the image classification correct. So, like this, we can handle by starting with a simple example. So, let us have

the implementation of matrix addition right. So, here we have taken a simple example. So, what we can do is you can extend this to, let us say, multiplication of two matrices once you know this right, and also you can try A inverse and also you can find the determinant of A right. So, these are all the famous problems right, and in fact, you can develop the knowledge on algorithms right, what is the efficient algorithm to do matrix multiplication right. What are the ways to do A

inverse? Once you know the ways, then you can also ask the question, what is the efficient algorithm to find out A inverse right? Also, try to learn how to find out the determinant of A. So, if you already know, well and good, but one more question you can ask is, what is the efficient way of finding out the determinant of A? That means you are also going to get the knowledge on the algorithms right and computational complexity.

Right, efficiency means the computational complexity. So, these things you can try right. So, now we will take up the simple example of addition of two matrices. So, we are going to do this right. So, here what we have to do is we have to be very careful with the size right.

So, make sure if this is m cross n b is also m cross n then you can do the addition right. If it is different size So, that is what we are going to discuss right assume that a is n cross m and b is m cross n you cannot do addition right. So, this is your a and this is your b means you cannot do the addition. So, we have to have the check.

So, now under matrix operations class right I have that add function that I already defined right in the class if you recall maybe I will just go back and show it right. So, this you have matrix operation operations So, under this I have addition so that means, I have already explained A is a matrix and B is a matrix. So, now I am checking the condition right. So, M right M cross N let us say A is M cross N B is also M cross N right A is also M cross N and B is also M cross N . So, A dot size is nothing, but the rows correct.

So, here it is M this is also M then no problem right I can check the next condition. Here it is n and this is n . If suppose they are not. Let us say A is assume that m cross n but B is m cross m . I will take this condition. One is true. A dot size not equal to B dot size.

False. Because both are m . One is false. Or A dot size not equal to B dot size. n not equal to m . True. So, false or true is true. I cannot do.

Statistics Class


```

1 #include <vector>
2 #include <numeric>
3 #include <cmath>
4
5 class Statistics {
6 public:
7     double mean(const std::vector<double>& data);
8     double variance(const std::vector<double>& data);
9     double stddev(const std::vector<double>& data);
10 };

```

data: 20.3 40.4 80.7

- ☛ Perform statistical computations.
- ☛ Efficiently handle large datasets.



So, if you are given this kind of matrices, I cannot do the addition, right. You can check the various cases, right. So, here it is true, there it is false, or both are true. Both are true means I can take this example. Let us say A is m cross n, B is some m dash cross n dash, m dash n dash are not equal with m and n respectively, right.

So, here also I cannot do the addition. That is what I mean, both are false, right. So, m is not equal to m dash, true. n not equal to n dash, true. So, true or true is true. So, I still cannot do the matrix addition, right.

So, that is what we are going to have, right. So, these are all the cases. So, find out whether another one is true or false, right. Finally, you will have true or true, right. If all these conditions are occurring,

then that means the dimensions are not matching, right. If all these conditions are occurring, then the dimensions are not matching, right. So, if the dimensions are the same, all right. So, let us take this case. So, then you can have the matrix addition.

Implementation - Matrix Addition

```

1 std::vector<std::vector<double>>> MatrixOperations::add(
2     const std::vector<std::vector<double>>>& A,
3     const std::vector<std::vector<double>>>& B) {
4     if (A.size() != B.size() || A[0].size() != B[0].size()) {
5         throw std::invalid_argument("Matrix dimensions must match");
6     }
7     std::vector<std::vector<double>>> result(A.size(),
8         std::vector<double>(A[0].size(), 0));
9     for (size_t i = 0; i < A.size(); ++i) {
10         for (size_t j = 0; j < A[0].size(); ++j) {
11             result[i][j] = A[i][j] + B[i][j];
12         }
13     }
14     return result;
15 }

```

Handwritten notes and diagrams:

- Diagram showing $A + B$ with dimensions $m \times n$ and $m \times n$ leading to $n \times n$.
- Diagram showing A ($m \times n$) and B ($m \times n$) being added to produce $A + B$ ($m \times n$).
- Diagram showing A ($m \times n$) and B ($n \times m$) being added to produce $A + B$ ($m \times m$).
- Diagram showing A ($m \times n$) and B ($n \times m$) being added to produce $A + B$ ($m \times m$).

So, what can you do? So, here you have the result, right. So, the result is a matrix, correct. So, a dot size, right. So, the size will be a dot size, and then here you have a vector.

right so comma vector of double vector of double which has a 0 of psi so all the elements you are making it 0 all you are initializing to 0 right the result matrix will be like this that is what that line is telling assume that is a 2 cross 3 matrix both are 2 cross 3 a is also 2 cross 3 initially you are making like this correct and then You have one for loop that for loop is running for the rows a dot size as I already said rows right and a 0 dot size column. So, for i is equal to 0 i less than rows j equal to 0 j less than columns you are adding the component wise addition right and then you are putting in the result of i comma j right the result matrix i comma j you are adding two matrices right a and b is given right a and b are given. and then you are adding put in result i j and return result right. So, this is what exactly will happen because the resultant is a vector of vector matrix right.

So, this is about your matrix addition and similarly we have quadratic equation solver right. So, solve quadratic. So, this equation solver is the class already we have on pair double and double. So, you are passing a b c a b c are doubles. Now, discriminant

You are calculating b squared minus 4ac. So, you are given ax squared plus bx plus c is equal to 0. This I am going to solve, right? a, b, c you are passing. If the determinant less than 0, right?


If the determinant is less than 0, throw a runtime error. So, just throw the exception, and no real roots will be printed. Yes, if b squared minus 4ac is less

than 0, then you cannot find any real root, right? Otherwise, root 1 is minus b plus the square root of the discriminant, plus or minus the root of b squared minus 4ac, divided by 2a.

Implementation - Quadratic Equation Solver

```
1 std::pair<double, double> EquationSolver::solveQuadratic(  
2     double a, double b, double c) {  
3     double discriminant = b * b - 4 * a * c;  
4     if (discriminant < 0) throw std::runtime_error("No real roots");  
5     double root1 = (-b + sqrt(discriminant)) / (2 * a);  
6     double root2 = (-b - sqrt(discriminant)) / (2 * a);  
7     return {root1, root2};  
8 }
```

$ax^2 + bx + c = 0$ $b^2 - 4ac$



So, this is what you are writing as code, right? Minus b plus or minus b squared. So, I will write b plus or minus the root of b squared minus 4ac, divided by 2a, as root 2. So, return root 1 and root 2 as a pair, right? That is what you are returning, right? So, this is the quadratic equation solver. So, here we have the mean

member function, right? So, we are passing vectors of data, all right. So, it will be like 20.3, 80.4, as I said already, all right, like this: 100.2, all right, whatever the size is, right. So, begin, and I hope this is the last element, let us say 104.7, right. So, this is data.begin, and this is data.end, right? Your sum is initialized to 0.

that is a meaning over here alright and then whatever be the data initially 0.0 will be added with 20.0 right. So, we will have the sum and then sum will be divided by size of the data. So, we know it is a mean right. So, that will be returned ok. So, that is that will be returned correct.

So, this is the mean calculation with the help of the member function right. So, here we are using the standard accumulate inbuilt all right. So, you also you can write your own code which is not very difficult ok. So, this is how we do the mean calculation and also we are going to have the error handling exception right.

Suppose you are having the invalid matrix dimension or in a system of equations right you have the singular matrix assume that you are having a singular matrix right and the negative discriminant in the quadratic equation. So, for example, I

have the equation $x^2 + x + 4 = 0$, right. What is $-b \pm \sqrt{b^2 - 4ac}$ negative, right, $4ac$, $2a$, 2 into 1 , right. So, $1 \pm \sqrt{16 - 15}$, right. So, we cannot get the real root.


So, in that case, what will happen? So, it is trying to throw, right, when you are, it is trying to So if it is a negative root, it will throw some exception, right? So that exception will be caught over here, error, right? And then we are going to say something, right?

In fact, we have written. To recall, we have written, right? So throw runtime error, no real roots, line number 4, right? So that will happen, right? So that we are going to have, right?

So this is one of the examples where we can have the error handling exception as well, right? So with this in mind, So, let us have this particular program the complete case study right. So, here you have void save results a function right. So, which has results of vector right and matrix result vector of vector that means this is a matrix.

Input/Output Handling

```
1 void saveResults(const std::vector<double>& results,
2               const std::vector<std::vector<double>>& matrixResult,
3               const std::string& filename) {
4     std::ofstream file(filename);
5     if (!file) {
6         throw std::runtime_error("Unable to open file for writing.");
7     }
8     file << "Results\n";
9     file << "Quadratic Equation Roots:\n";
10    file << "Root 1: " << results[0] << "\n";
11    file << "Root 2: " << results[1] << "\n\n";
12
13    file << "Mean Calculation:\n";
14    file << "Mean: " << results[2] << "\n\n";
15
16    file << "Matrix Addition Result:\n";
17    for (const auto& row : matrixResult) {
18        for (const auto& val : row) {
19            file << val << " ";
20        }
21        file << "\n";
22    }
23    file.close();
24 }
```



So, the first one is a one dimensional array second one is a matrix result is a matrix and here you have a file name right the string address file name right. So, you use OF stream. Recall, whatever we have studied that we are going to use. OF stream, we have already done. File of file name, right.

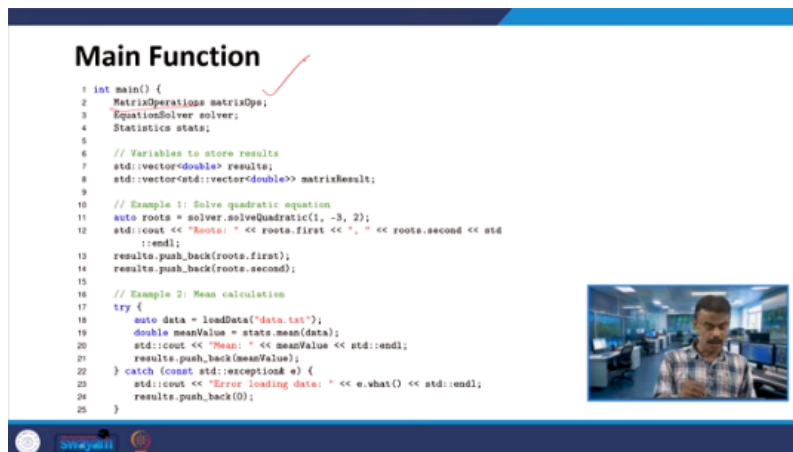
If not file, that means file, if it is if file, file not equal to 0, file not equal to null. So, if it is null, right, if the file does not exist, unable to open, right. If the file does not exist, unable to open will be displayed, right. Otherwise, here you have file, right. So, this will be returned.

This will be writing in the file, whatever you are writing in the code, number results, quadratic equations, roots and then root 1, whatever you are going to calculate, results 0. It is a vector, results of 0 and then results of 1. So these things will be returned in the file and file mean calculation will be returned, mean, results of 2. So here you have a root 1 and root 2, results 0, result 1. And mean calculation I will use another result results 2 right.

In fact, it is a one dimensional array vector right. So, result 0 result 1 I will use for root 1 and root 2 and result 2 I will use for mean calculation. Then here you have the matrix addition file matrix addition result right. So, I here I have the range for loop right matrix result copied into row right which is the auto variable and the row will be copied into val. So, that means you have two for loops, one is running for the row, another one is for column, right.

And then you are, right, whatever you are getting the output as addition, right. So, you are writing in the file value, right. So, then file new line and file close. So, here we have handled the file, file handling system we have studied, right. So, now we will go to the computations, the mathematical computations.

So now I am going into the main right. So here you have the function save results function right. So that means you are going to handle the file. And then in the case of a main function I have under matrix operations class I have matrix OPS object. Under equation solver I have solver object right.



```
Main Function ✓  
  
1 int main() {  
2     MatrixOperations matrixOps;  
3     EquationSolver solver;  
4     Statistics stats;  
5  
6     // Variables to store results  
7     std::vector<double> results;  
8     std::vector<std::vector<double>> matrixResult;  
9  
10    // Example 1: Solve quadratic equation  
11    auto roots = solver.solveQuadratic(1, -3, 2);  
12    std::cout << "Roots: " << roots.first << ", " << roots.second << std  
13    ::endl;  
14    results.push_back(roots.first);  
15    results.push_back(roots.second);  
16  
17    // Example 2: Mean calculation  
18    try {  
19        auto data = loadData("data.txt");  
20        double meanValue = stats.mean(data);  
21        std::cout << "Mean: " << meanValue << std::endl;  
22        results.push_back(meanValue);  
23    } catch (const std::exception& e) {  
24        std::cout << "Error loading data: " << e.what() << std::endl;  
25        results.push_back(0);  
26    }  
27 }
```

Under statistics class I have stats object right. I have vector results right one dimensional array. here i have a matrix result two-dimensional array so now i am passing 1 minus 3 2 that means i am trying to solve 1 x squared minus 3 x plus 2

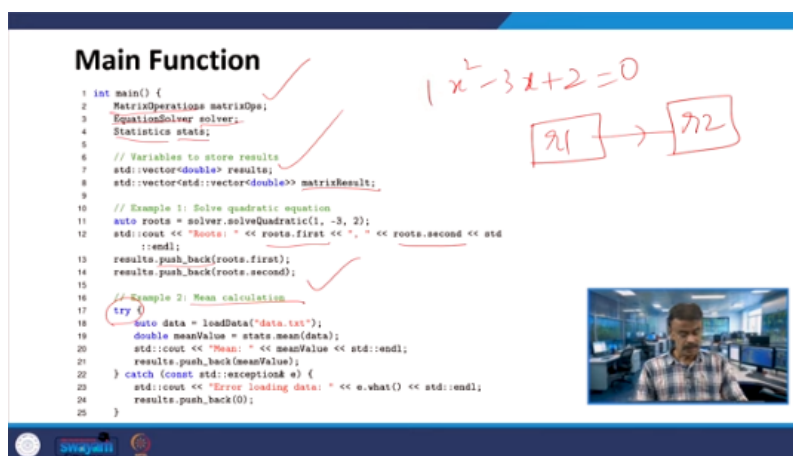
equal to 0 right so we have already seen right the solving equation right all the programs we have already seen right so this will be called so in fact we have defined previously right all these functions we have defined previously solve quadratic right. So, here you go solve quadratic all right.

So, these are all I explain maybe when I when we are seeing it as a program we are going to see this all right solve quadratic you are passing 1 minus 3 2. So, this function will be called I this function we have member function I have already defined. So, auto root. So, whatever you are getting as output all right. So, double all right that you are going to get

So now see out roots. You are writing roots first and roots second, right? So then you are having the pushback, push underscore back. This we had seen several times. So the first root will be returned, right?

And then let us say first root, whatever the root, let us say root R1. R1 is some number, right? And then pushback second, R2 in the list, right? So this you are writing, the root 1 and root 2, right? And then mean calculation, you are trying.

You use a try-throw catch, right? So, load data data dot txt will be loaded right the sum file will be loaded and from that file you are reading storing as a data which is the auto variable and you are calculating the mean we already written the function mean right. So, start data end data and then right divided by data size you are calculating the mean. So, which you are doing over here see out mean right. So, push back mean value.



The image shows a C++ code editor window titled "Main Function". The code is as follows:

```
1 int main() {
2     MatrixOperations matrixOps;
3     EquationsSolver solver;
4     Statistics stats;
5
6     // Variables to store results
7     std::vector<double> results;
8     std::vector<std::vector<double>> matrixResult;
9
10    // Example 1: Solve quadratic equation
11    auto roots = solver.solveQuadratic(1, -3, 2);
12    std::cout << "Roots: " << roots.first << ", " << roots.second << std::endl;
13    results.push_back(roots.first);
14    results.push_back(roots.second);
15
16    // Example 2: Mean calculation
17    try {
18        auto data = loadData("data.txt");
19        double meanValue = stats.mean(data);
20        std::cout << "Mean: " << meanValue << std::endl;
21        results.push_back(meanValue);
22    } catch (const std::exception& e) {
23        std::cout << "Error loading data: " << e.what() << std::endl;
24        results.push_back(0);
25    }
26 }
```

Handwritten annotations in red include:

- Checkmarks next to lines 4, 6, 11, 14, and 16.
- A handwritten equation $x^2 - 3x + 2 = 0$ with arrows pointing to boxes containing the roots 1 and 2.

A small video inset in the bottom right corner shows a man speaking.

So, here you can call another right mean all right so if there is any problem of handling the data right so it will throw the exception so error loading the data and

the dot what right so whatever you have written so you cannot handle the file right unable to open the file for writing right so something like that will be printed all right so if this is happening so pushback zero third one is matrix addition so you are taking one matrix from the text right a another matrix from another text b txt file right and then you are calling the add function we have already seen matrix a matrix b you are passing we are calling this right so whatever you are reading storing in matrix a whatever you are reading storing in matrix b so use again try throw catch right so we are passing matrix a and matrix b we have already done this function So matrix result will be computed right. So matrix addition result.


So to range for loop matrix row and column and then you are printing that value right. So this you are doing printing the matrix right. So in case any problem reading the file it will throw the exception right. So exception will be caught here right. So now you are doing the save results.

You are calling save results. So save results, you are pausing results, matrix result, right, and results.txt, right. So it will store it in results.txt file. We will see all the files, whatever file we are using, right. So one file we are using, data.txt.

Another file you are using, matrix A.txt. Third, matrix B.txt. And the result you are going to write, results.txt, right. So here you have results, right. You are pausing results, matrix result.

Main Function (contd.)

```
26 // Example 3: Matrix addition
27 try {
28     auto matrixA = loadMatrix("matrixA.txt");
29     auto matrixB = loadMatrix("matrixB.txt");
30     matrixResult = matrixOps.add(matrixA, matrixB);
31     std::cout << "Matrix Addition Result:\n";
32     for (const auto& row : matrixResult) {
33         for (const auto& val : row) {
34             std::cout << val << " ";
35         }
36         std::cout << std::endl;
37     }
38 } catch (const std::exception& e) {
39     std::cout << "Error: " << e.what() << std::endl;
40 }
41
42 // Save all results to a file
43 saveResults(results, matrixResult, "results.txt");
44 std::cout << "Results saved to results.txt." << std::endl;
45
46 return 0;
47 }
```



and the file that it is going to write and then it will save it in result save to result dot txt right so when i run the code i have to get the output so before that we will see what are all the files we'll go and see the files so data dot txt so this is the

data right one two three four five six all right so you have we can write whatever we want and matrix a all right and matrix b and then results dot txt should be empty all right so what we will do we will cut it and save right now initially empty maybe one more time we close it and i will open again right so you can find it is empty so now we will run the code so when we run the code the code that i have explained right yes so you can have some mean and then it is showing certain errors the reason is you are we will see matrix a 2 cross 4 and here 2 cross 3 yes i should get the error right so now yeah so i should get the error So, now what I will do I will delete that one also. So, now both are 2 cross 3 correct.

So, now when I again run the code when again run the code I should get the output for matrix let us see yeah. So, when I look the results yeah. So, now matrix addition result is we are getting right and if you look at data dot txt I will explain one by one. So, when you open data dot txt. So, here you have

1, 2, 3, 4, 5 and purposefully I put one blank space then 4, 5, 5 right. So, the result should be addition of all these divided by 8 all right. So, you can make the computation you can just check also and if you look at the output we are getting here you go 3.625 all right. And the second one is if you look at the code all right. So, we will go to the code

And here you have we are solving the quadratic equation for $x^2 + x + 4 = 0$ right $x^2 + x + 4 = 0$. So the discriminant will be obviously less than 0 right. So in that case if it is a negative so it has to throw the exception. So the exception is here no real roots error colon no real roots. And then we will open matrix A dot txt already we had I have told you but still matrix A is 2 by 3 matrix B is also 2 by 3 right you add both right you will get the result.

So result is here right the addition of two matrices. So now also we are getting one more information results saved to results dot txt right. So if you look at results dot txt initially it was empty if you recall yes now you can see. no real rules mean matrix addition result ok. So, these are all the results we are getting ok.

So, this is what the output is right this is the some other ok equivalent to this we are getting right. So, here we have kept on changing few things So, we have to get the output like this. In fact, one more time we can see the output file can you

go back to that yes if you the result results dot t x right you can see that. If there are any roots right if a discriminant is greater than 0 we will get the output and then mean is printing matrix addition result is printing.

So, here I am printing the equivalent one, assuming that you have the roots, right. So, the equivalent one will be printed. So, these kinds of problems you can test. So, here we have handled so many. So, we have used error handling and exceptions.

Here you have a class. So, various classes we have used. So, mainly we have used the files, the file handling system. So, like this. We are going to see some more programs also in the next few classes, right.

So, that means whatever the object-oriented programming concepts that we have seen, some of them, right, and then we have got the results, right. So, one has to be very careful with the test cases, right. So, suppose you are using singular matrices. So, let us say solving for Ax is equal to b kind of problem or in the addition or multiplication, right. Suppose you are taking the


right the dimensions right the different dimensions right so one has to be very careful for these edge cases you call it as a edge cases right so suppose the negative discriminant happens in fact one of the examples we had seen how to manage that or even the large data sets how to handle large data sets correct and suppose you have the input right the wrong or invalid input file right, or malformed input file. So, in that cases, how do you use, right? So, all the cases, most of the cases we use the try-through catch, right, the error handling exception. So, you one has to be very careful and try to use the exceptions, right.

So, we have used the class file, right, and error handling exceptions in the previous case study, all right. So, the main challenges that you may face, right, handling the large matrices, Suppose 3000 by 3000 matrix, you have to do the multiplication, right. So, now you have to think about the optimized code. Either you can do or take out the library, right.

For example, eigenvalue computation, eigenvector computation, right. So, one has to be very careful how to handle the large matrices, right. So, the error propagation. So, when there are errors, I mean you carefully handle with error handling exceptions. and extensibility right.

Challenges and Solutions

- ❖ **Handling Large Matrices:** Use optimized libraries like Eigen for efficiency.
- ❖ **Error Propagation:** Isolate errors using exception handling.
- ❖ **Extensibility:** Design modular classes to support future enhancements.



At the bottom of the slide, there is a blue bar containing a circular logo on the left, the text 'Smagati' in the center, and a small red circular icon on the right.

So, I mean we will talk about this right. So, suppose I am going to use that inheritance concept right. So, we have to design the modular classes to support future enhancements right. So, these are all the challenges and in fact, we have the solutions as well. So, this is the summary of the framework.

So, we built a modular framework for advanced computations particularly the mathematical computations. So, we have implemented addition. So, what you can do? You can try multiplication, finding out the inverse, finding out the eigenvalue, determinant right you can extend. We have seen quadratic equation, you try to solve the linear equation and in fact, you take the matrix which is singular right $a \cdot x = b$, a singular and see how you can use error handling exception for those, we have solved for quadratic.

we have seen mean you try to compute variance all right. So, and moreover with the help of file handling or exception handling. So, you can solve these kind of problems. So, with this I am concluding this particular case study. Thank you very much.