CS 226 Mid-Term

PRESENTATION

TABLE OF CONTENTS

01 INTRODUCTION

02 Accomplishments

03 Code Explanation

04 Challenges & Solutions

05 Learning Experience

06 Optional Live demo

01 INTRODUCTION

For my C++ midterm, I chose the following projects:

Simple calculator: This C++ project implements a basic calculator, demonstrating the practical application of functions, input validation and user interaction. Users can perform the following: addition, subtraction, multiplication and division by entering two numbers.

Student Data Management System: This project is built around the `Student` class, which demonstrates the principles of object-oriented programming (OOP). It manages student records, including names and grades, through encapsulation of data and methods within the class to set and get student details.

Number Guessing Game: This interactive number guessing game in C++ showcases the use of random number generation, loops, conditional logic, and basic input/output operations.

02 Accomplishments

1) Simple calculator:

- **Core Functionality**: Successfully developed a simple calculator program that performs arithmetic operations (addition, subtraction, multiplication, division) on two numbers. The implementation uses separate functions for each operation and allows the users to keep performing calculations until they choose to exit.
- **Challenges Overcome**: Initially faced challenges with handling division operations (divide by zero primarily) which caused runtime errors.

2) Student Data Management System:

•**Core Functionality**: Achieved the creation and management of a student database using an array of `Student` objects. This included developing functionalities for setting and getting student names and grades, ensuring data encapsulation and ease of management for the user(instructor).
•**Challenges Overcome**: Faced difficulties in managing the flow of input from the user, especially in switching between numeric inputs and string inputs, which was resolved by correctly placing `cin.ignore()` to clear the input buffer before reading strings.

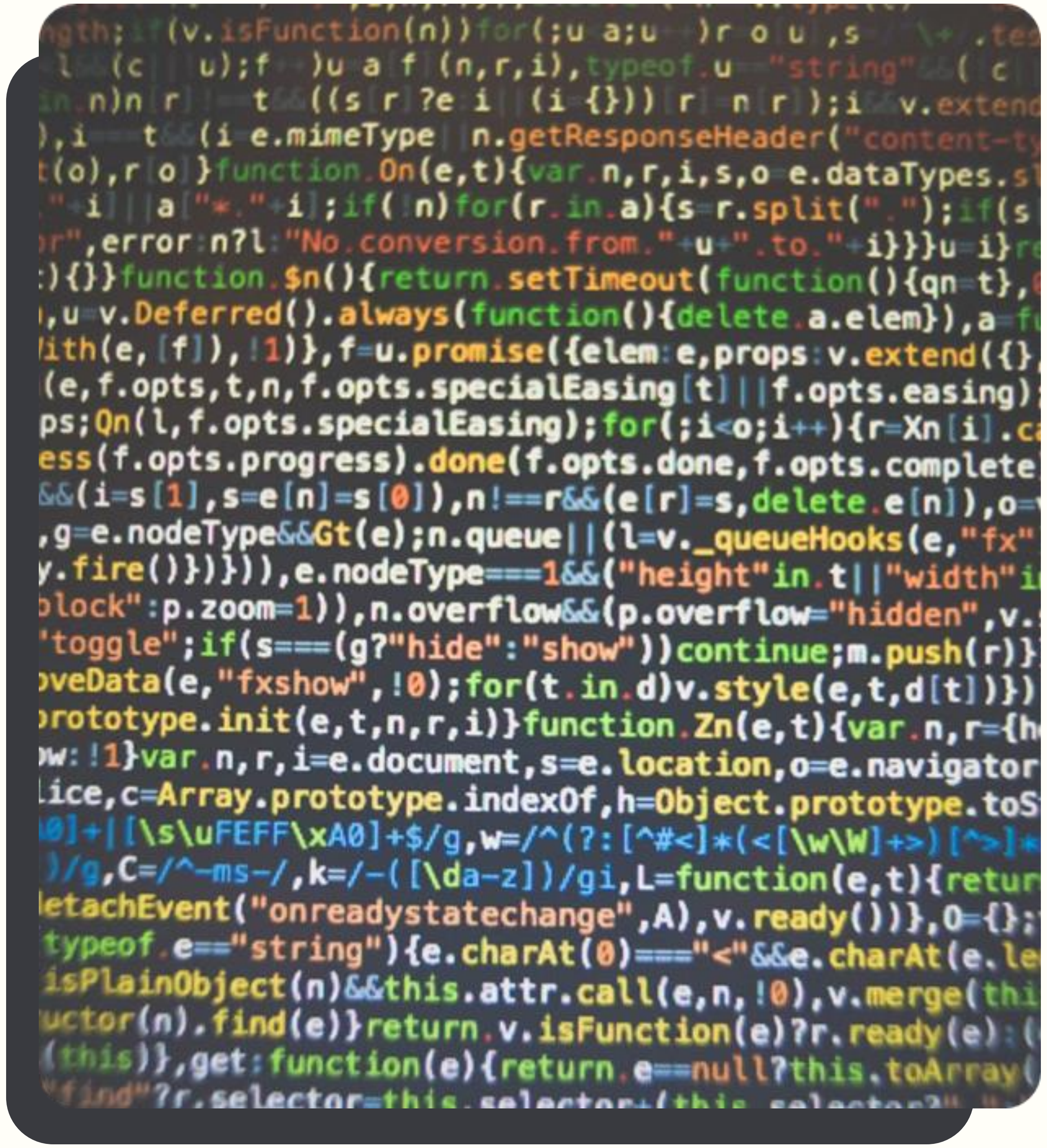
Cont..

3) Number guessing game:

Core functionality: Implemented the game with a random number generator to create a different game experience each time. The game provides feedback based on user guesses and adjusts the guidance accordingly.

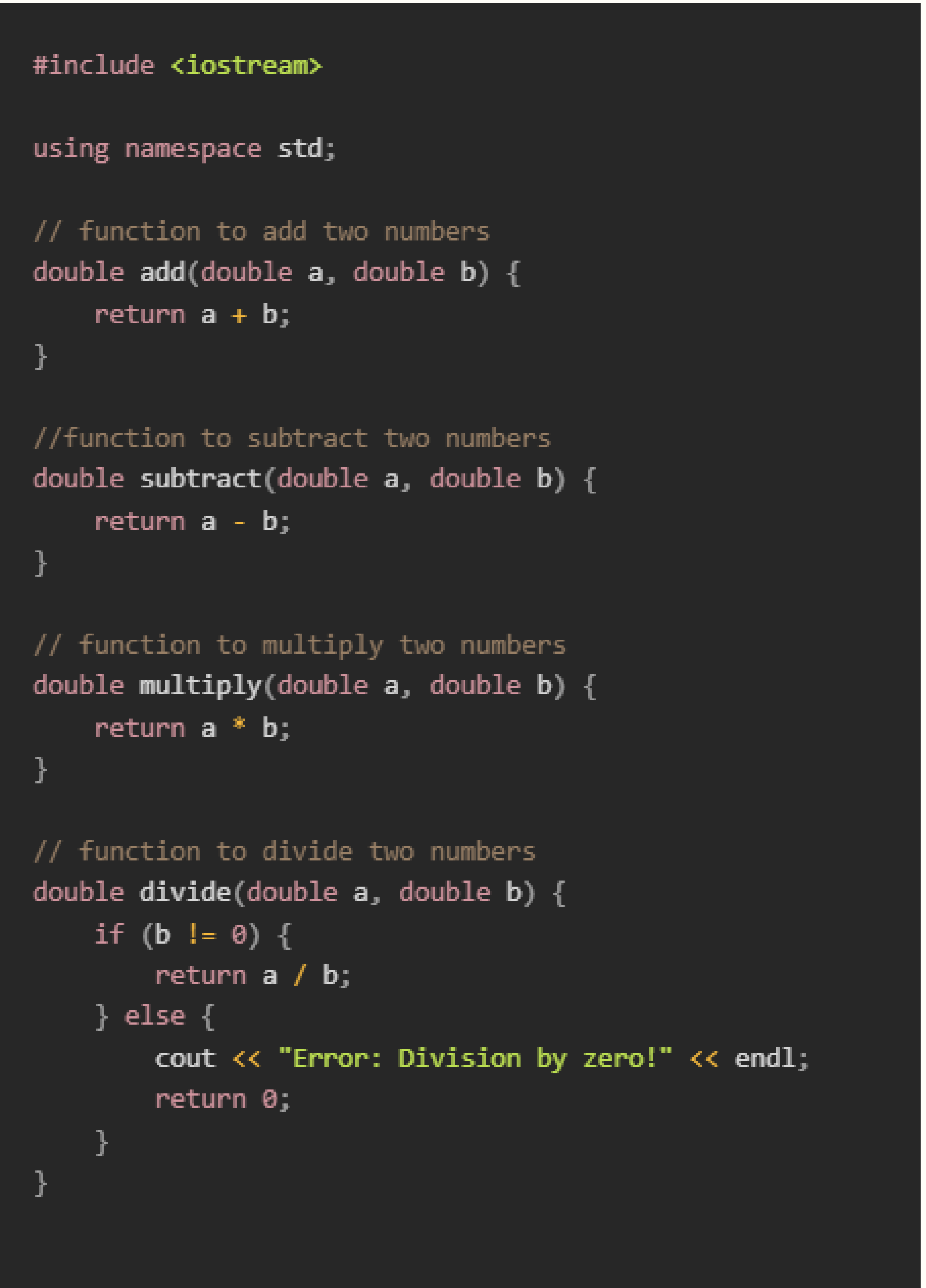
Challenges overcame:

Encountered issues with the main loop that could inadvertently continue forever if the user input was not numeric. This issue was addressed by adding simple input validation checks and clearing the input buffer also syntax for one of the features.

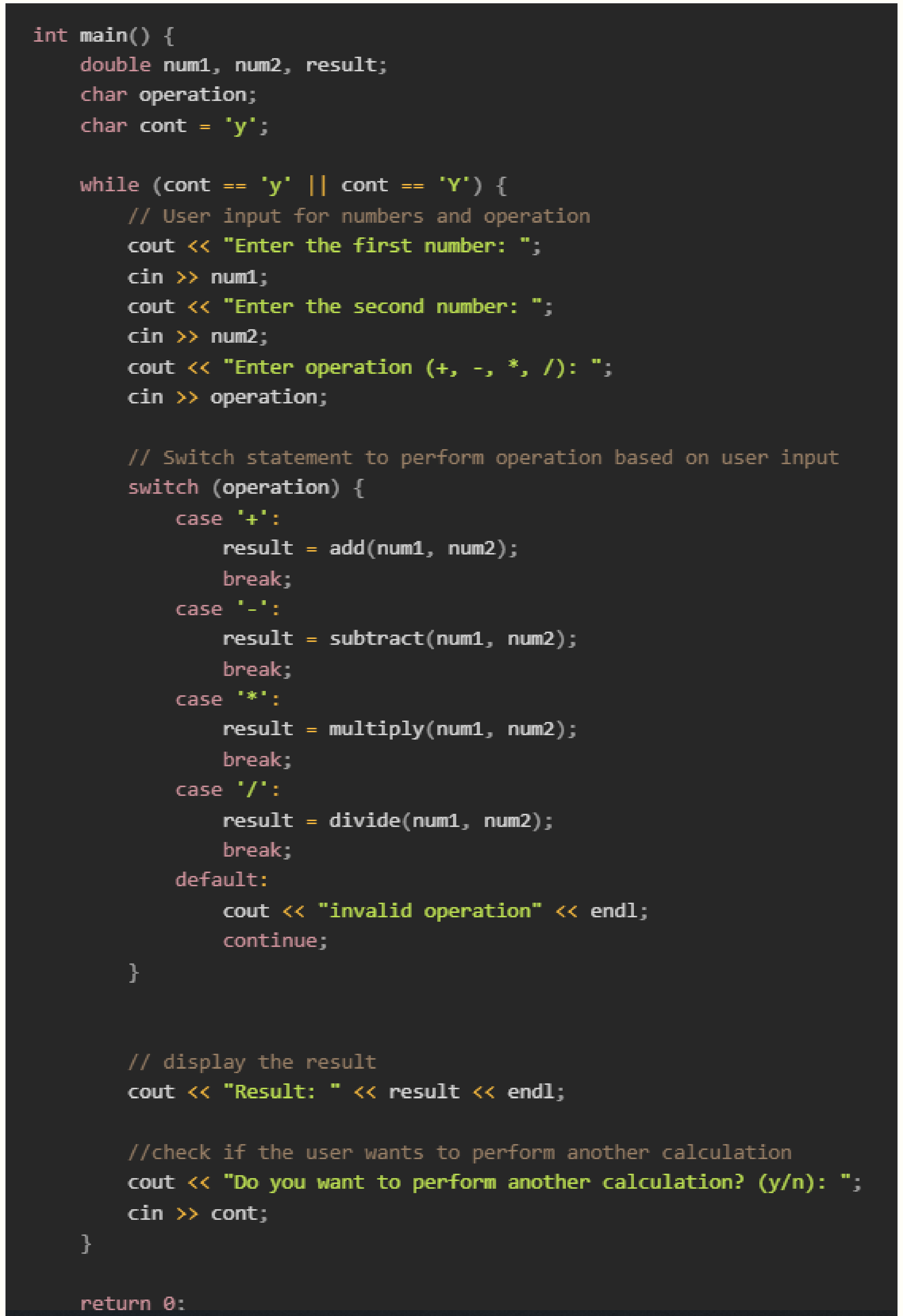


```
ngth;if(v.isFunction(n))for(;u a;u )r o u ,s \+ .tes
l (c u);f )u a f (n,r,i),typeof u "string"&&( c
in n)n r t&&((s r ?e i (i {})) r n r );i&&v.extend
),i t&&(i e.mimeType n.getResponseHeader("content-ty
t(o),r o }function On(e,t){var n,r,i,s,o e.dataTypes.s
"+i a["*."+i];if(!n)for(r in a){s r.split(" ");if(s
r",error:n?l:"No conversion from "+u+" to "+i}}}u i}r
){}}function $n(){return setTimeout(function(){qn t},
,u v.Deferred().always(function(){delete a.elem}),a fu
ith(e,[f]),!1)},f=u.promise({elem:e,props:v.extend({},
(e,f.opts,t,n,f.opts.specialEasing[t]||f.opts.easing);
ps;Qn(l,f.opts.specialEasing);for(;i<o;i++){r=Xn[i].ca
ess(f.opts.progress).done(f.opts.done,f.opts.complete
&&(i=s[1],s=e[n]=s[0]),n!==r&&(e[r]=s,delete e[n]),o=
,g=e.nodeType&&Gt(e);n.queue||(l=v._queueHooks(e,"fx"
y.fire()})}})),e.nodeType===1&&("height"in t||"width"i
lock":p.zoom=1)),n.overflow&&(p.overflow="hidden",v.
'toggle";if(s===(g?"hide":"show"))continue;m.push(r)}
veData(e,"fxshow",!0);for(t in d)v.style(e,t,d[t])})
rototype.init(e,t,n,r,i)}function Zn(e,t){var n,r={h
w:!1}var n,r,i=e.document,s=e.location,o=e.navigator
ice,c=Array.prototype.indexOf,h=Object.prototype.toS
0]+|[\s\uFEFF\xA0]+$/g,w=/^(?:[^#<]*(<[\w\W]+>)[^>]*
)/g,C=/^-ms-/,k=/-([\da-z])/gi,L=function(e,t){retur
etachEvent("onreadystatechange",A),v.ready())},O={};
typeof e=="string"){e.charAt(0)==="<"&&e.charAt(e.le
isPlainObject(n)&&this.attr.call(e,n,!0),v.merge(thi
uctor(n).find(e)}return v.isFunction(e)?r.ready(e):(
(this)},get:function(e){return e==null?this.toArray(
"find"?r.selector=this.selector+(this.selector?"
```

1) Calculator

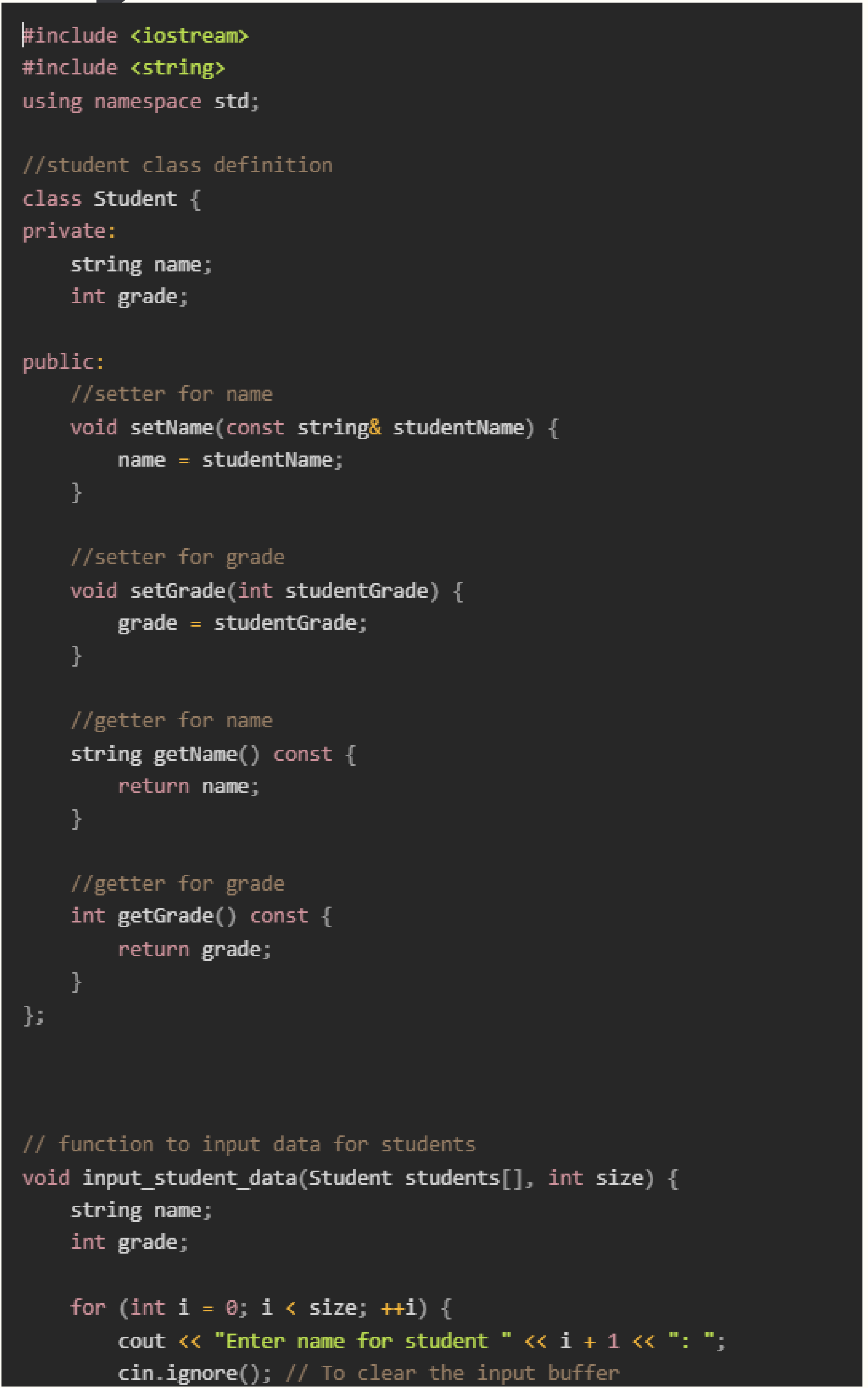


```cpp
#include <iostream>

using namespace std;

// function to add two numbers
double add(double a, double b) {
    return a + b;
}

//function to subtract two numbers
double subtract(double a, double b) {
    return a - b;
}

// function to multiply two numbers
double multiply(double a, double b) {
    return a * b;
}

// function to divide two numbers
double divide(double a, double b) {
    if (b != 0) {
        return a / b;
    } else {
        cout << "Error: Division by zero!" << endl;
        return 0;
    }
}
```

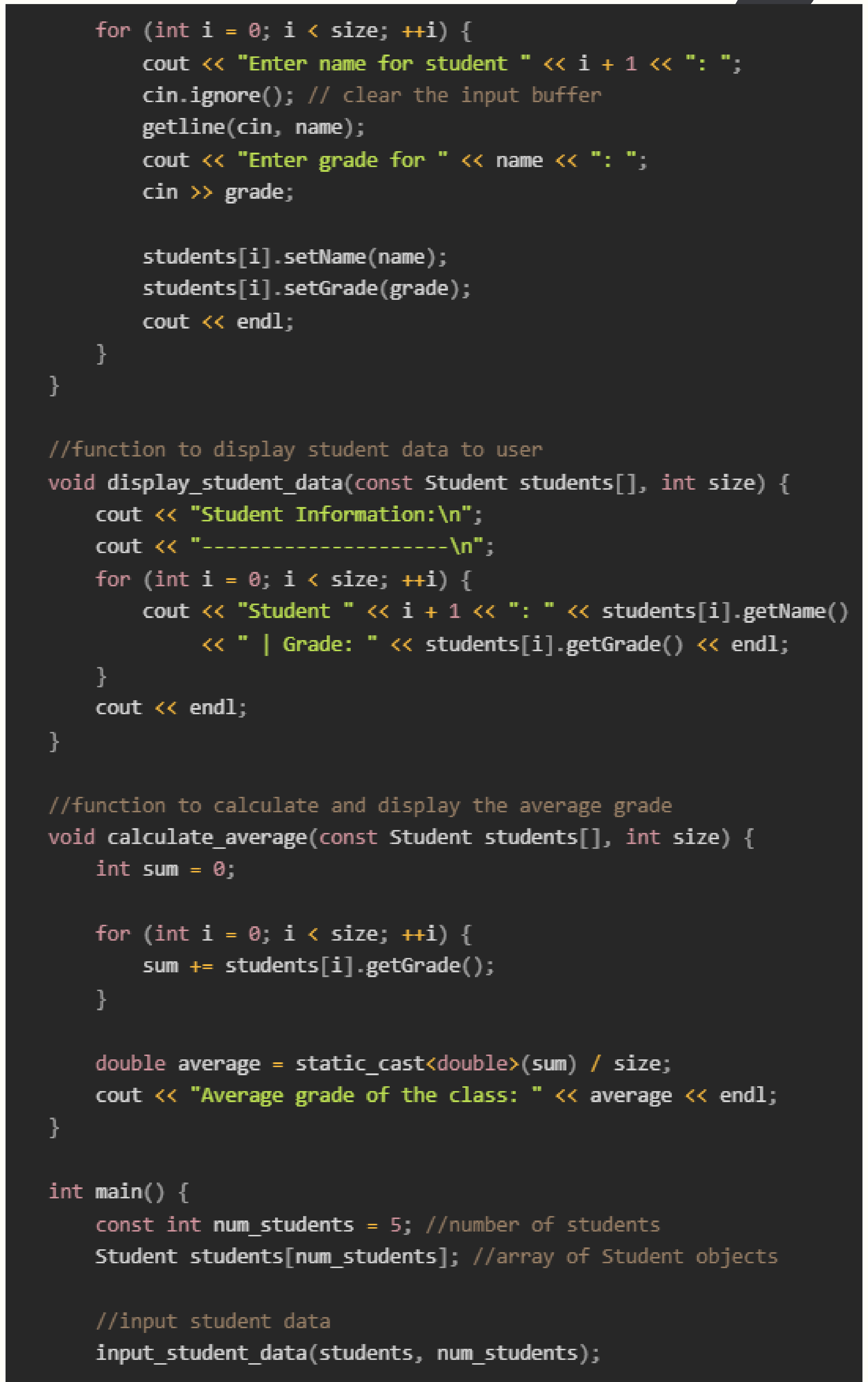


```cpp
int main() {
    double num1, num2, result;
    char operation;
    char cont = 'y';

    while (cont == 'y' || cont == 'Y') {
        // User input for numbers and operation
        cout << "Enter the first number: ";
        cin >> num1;
        cout << "Enter the second number: ";
        cin >> num2;
        cout << "Enter operation (+, -, *, /): ";
        cin >> operation;

        // Switch statement to perform operation based on user input
        switch (operation) {
            case '+':
                result = add(num1, num2);
                break;
            case '-':
                result = subtract(num1, num2);
                break;
            case '*':
                result = multiply(num1, num2);
                break;
            case '/':
                result = divide(num1, num2);
                break;
            default:
                cout << "invalid operation" << endl;
                continue;
        }


        // display the result
        cout << "Result: " << result << endl;

        //check if the user wants to perform another calculation
        cout << "Do you want to perform another calculation? (y/n): ";
        cin >> cont;
    }

    return 0;
```

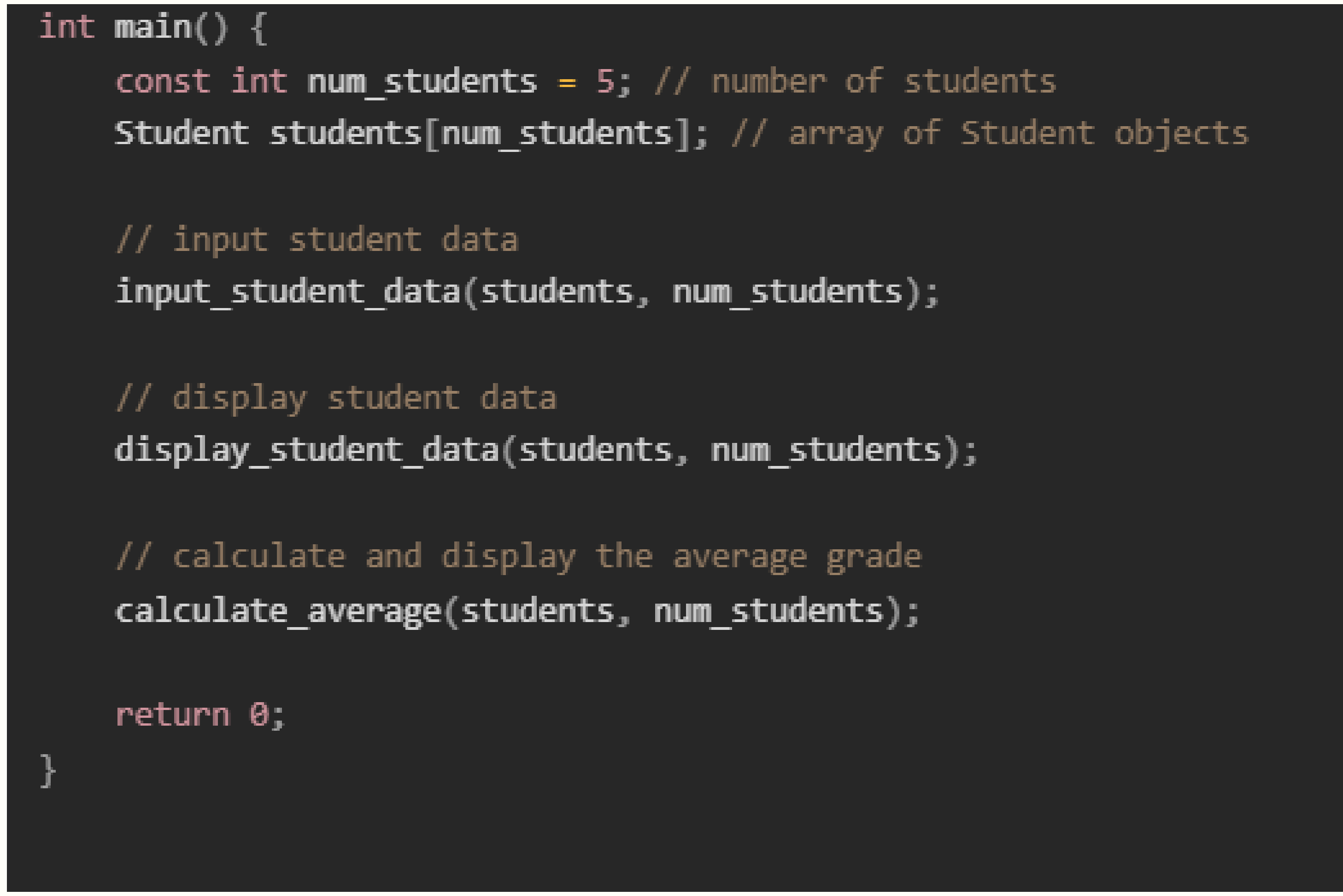
2) student data management system



```cpp
#include <iostream>
#include <string>
using namespace std;

//student class definition
class Student {
private:
    string name;
    int grade;

public:
    //setter for name
    void setName(const string& studentName) {
        name = studentName;
    }

    //setter for grade
    void setGrade(int studentGrade) {
        grade = studentGrade;
    }

    //getter for name
    string getName() const {
        return name;
    }

    //getter for grade
    int getGrade() const {
        return grade;
    }
};



// function to input data for students
void input_student_data(Student students[], int size) {
    string name;
    int grade;

    for (int i = 0; i < size; ++i) {
        cout << "Enter name for student " << i + 1 << ": ";
        cin.ignore(); // To clear the input buffer
```

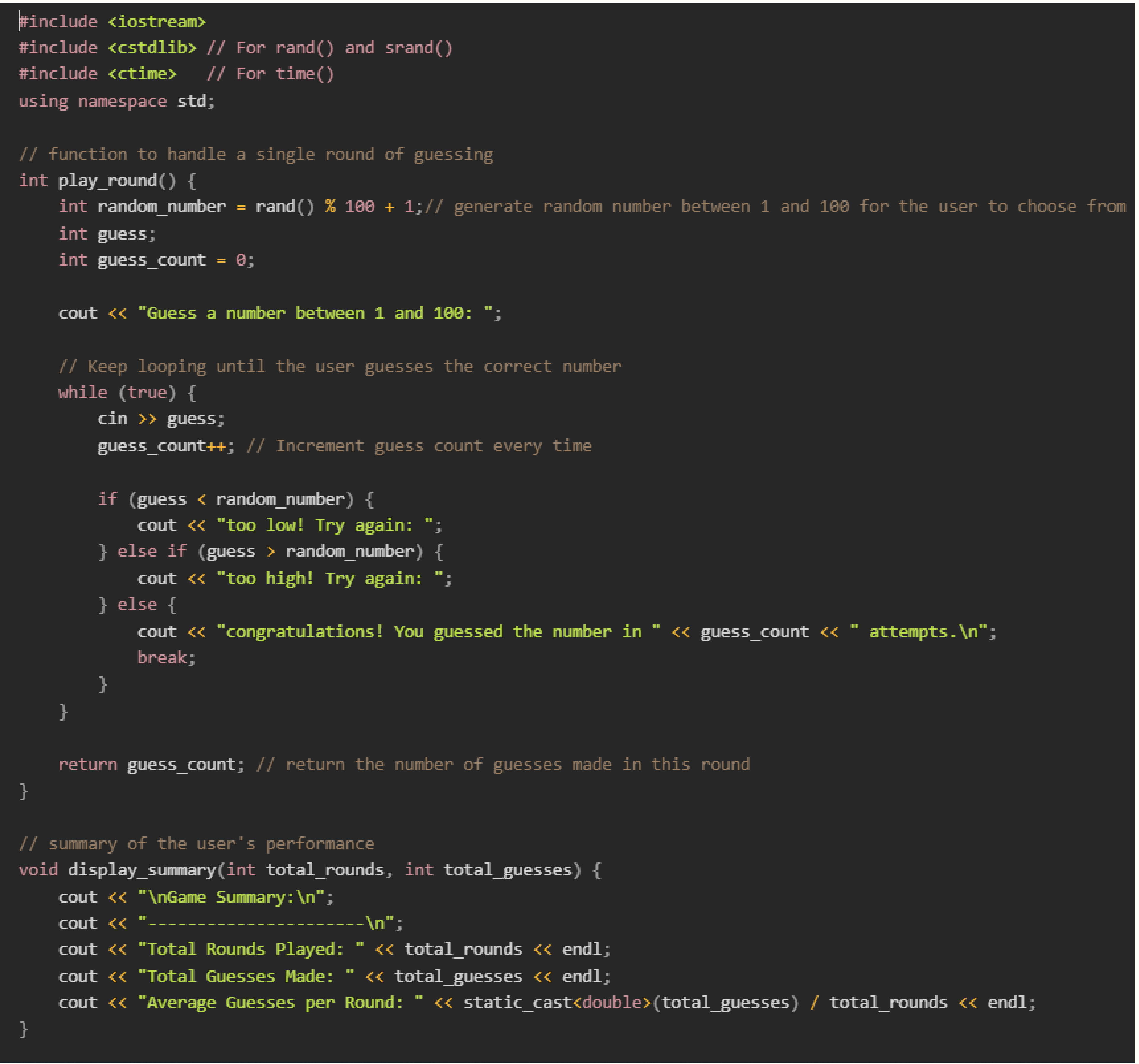


```cpp
    for (int i = 0; i < size; ++i) {
        cout << "Enter name for student " << i + 1 << ": ";
        cin.ignore(); // clear the input buffer
        getline(cin, name);
        cout << "Enter grade for " << name << ": ";
        cin >> grade;

        students[i].setName(name);
        students[i].setGrade(grade);
        cout << endl;
    }
}

//function to display student data to user
void display_student_data(const Student students[], int size) {
    cout << "Student Information:\n";
    cout << "--------------------\n";
    for (int i = 0; i < size; ++i) {
        cout << "Student " << i + 1 << ": " << students[i].getName()
             << " | Grade: " << students[i].getGrade() << endl;
    }
    cout << endl;
}

//function to calculate and display the average grade
void calculate_average(const Student students[], int size) {
    int sum = 0;

    for (int i = 0; i < size; ++i) {
        sum += students[i].getGrade();
    }

    double average = static_cast<double>(sum) / size;
    cout << "Average grade of the class: " << average << endl;
}

int main() {
    const int num_students = 5; //number of students
    Student students[num_students]; //array of Student objects

    //input student data
    input_student_data(students, num_students);
```

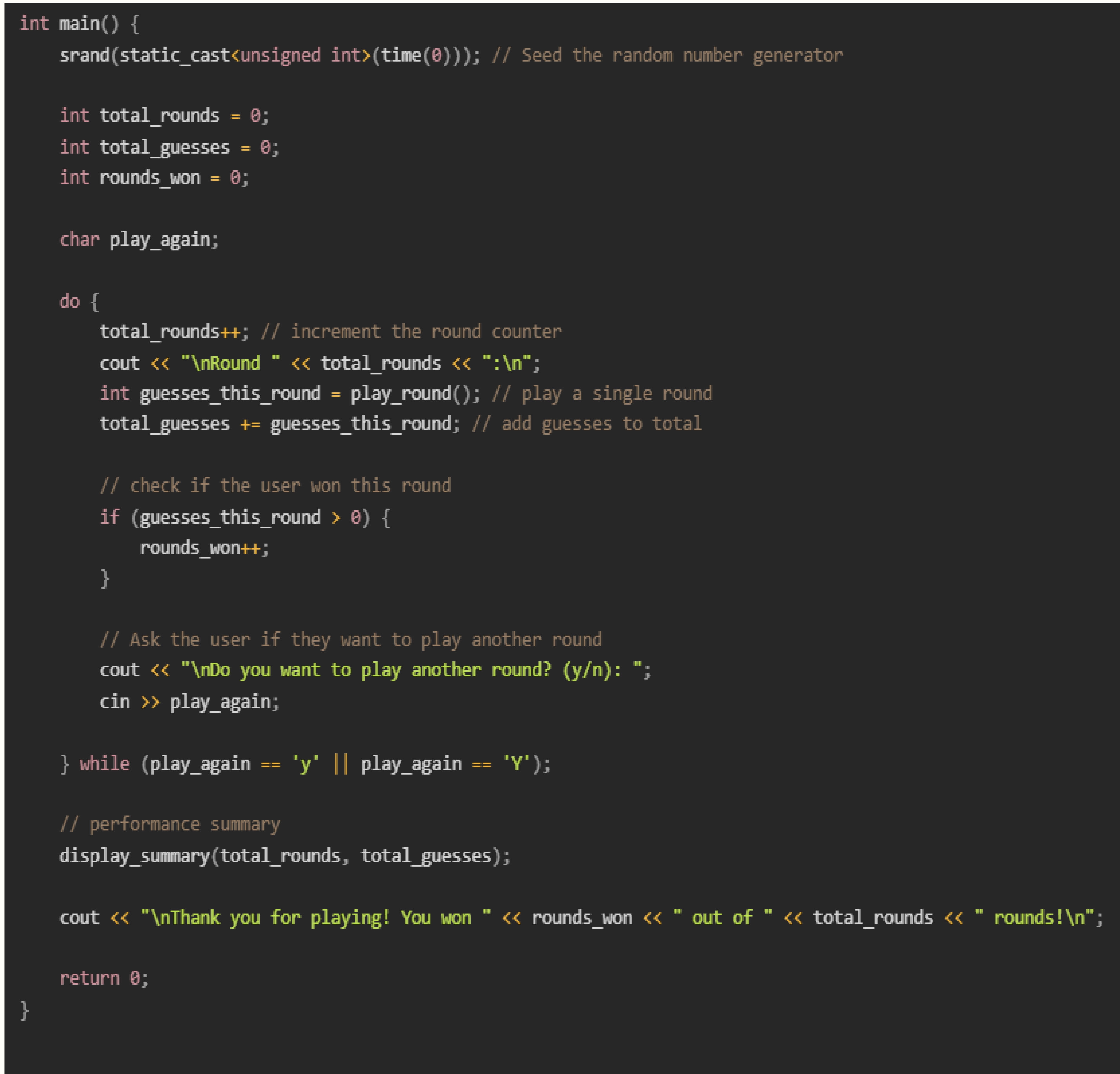


```cpp
int main() {
    const int num_students = 5; // number of students
    Student students[num_students]; // array of Student objects

    // input student data
    input_student_data(students, num_students);

    // display student data
    display_student_data(students, num_students);

    // calculate and display the average grade
    calculate_average(students, num_students);

    return 0;
}
```

3) Number guessing game



```cpp
#include <iostream>
#include <cstdlib> // For rand() and srand()
#include <ctime>   // For time()
using namespace std;

// function to handle a single round of guessing
int play_round() {
    int random_number = rand() % 100 + 1;// generate random number between 1 and 100 for the user to choose from
    int guess;
    int guess_count = 0;

    cout << "Guess a number between 1 and 100: ";

    // Keep looping until the user guesses the correct number
    while (true) {
        cin >> guess;
        guess_count++; // Increment guess count every time

        if (guess < random_number) {
            cout << "too low! Try again: ";
        } else if (guess > random_number) {
            cout << "too high! Try again: ";
        } else {
            cout << "congratulations! You guessed the number in " << guess_count << " attempts.\n";
            break;
        }
    }

    return guess_count; // return the number of guesses made in this round
}

// summary of the user's performance
void display_summary(int total_rounds, int total_guesses) {
    cout << "\nGame Summary:\n";
    cout << "----------------------\n";
    cout << "Total Rounds Played: " << total_rounds << endl;
    cout << "Total Guesses Made: " << total_guesses << endl;
    cout << "Average Guesses per Round: " << static_cast<double>(total_guesses) / total_rounds << endl;
}
```



```cpp
int main() {
    srand(static_cast<unsigned int>(time(0))); // Seed the random number generator

    int total_rounds = 0;
    int total_guesses = 0;
    int rounds_won = 0;

    char play_again;

    do {
        total_rounds++; // increment the round counter
        cout << "\nRound " << total_rounds << ":\n";
        int guesses_this_round = play_round(); // play a single round
        total_guesses += guesses_this_round; // add guesses to total

        // check if the user won this round
        if (guesses_this_round > 0) {
            rounds_won++;
        }

        // Ask the user if they want to play another round
        cout << "\nDo you want to play another round? (y/n): ";
        cin >> play_again;

    } while (play_again == 'y' || play_again == 'Y');

    // performance summary
    display_summary(total_rounds, total_guesses);

    cout << "\nThank you for playing! You won " << rounds_won << " out of " << total_rounds << " rounds!\n";

    return 0;
}
```

challenges

Challenge #1	Challenge #2	Challenge #3	Challenge #4
Calculator: **Challenge:** The primary challenge I faced was the divide by zero edge case. **Solution:** implemented a simple condition check to see if the user was attempting to divide by zero.	**Student Data Management System:** **Challenge:** initially struggled with managing the transition between different types of inputs, which caused incorrect data capture and errors in the display data. **Solution:** I used cin.ignore() before each getline() to clear the newline character left in the input buffer by previous inputs which solved the issue.	**Number Guessing Game:** **Challenge:** Random number generator initially produced predictable results, reducing the games challenge. **Solution:** Improved the randomness by seeding the random number generator with srand(time(NULL)) to make sure the sequence of numbers was different each execution of the program.	**Number Guessing Game:** **Challenge:** The program would not respond to non-numeric inputs, causing it to crash or fill the output box with errors. **Solution:** Similarly, I added an input validation system using cin.fail() & cin.clear() to check and clear for possible inputs that could cause an error.

05 Learning Exp

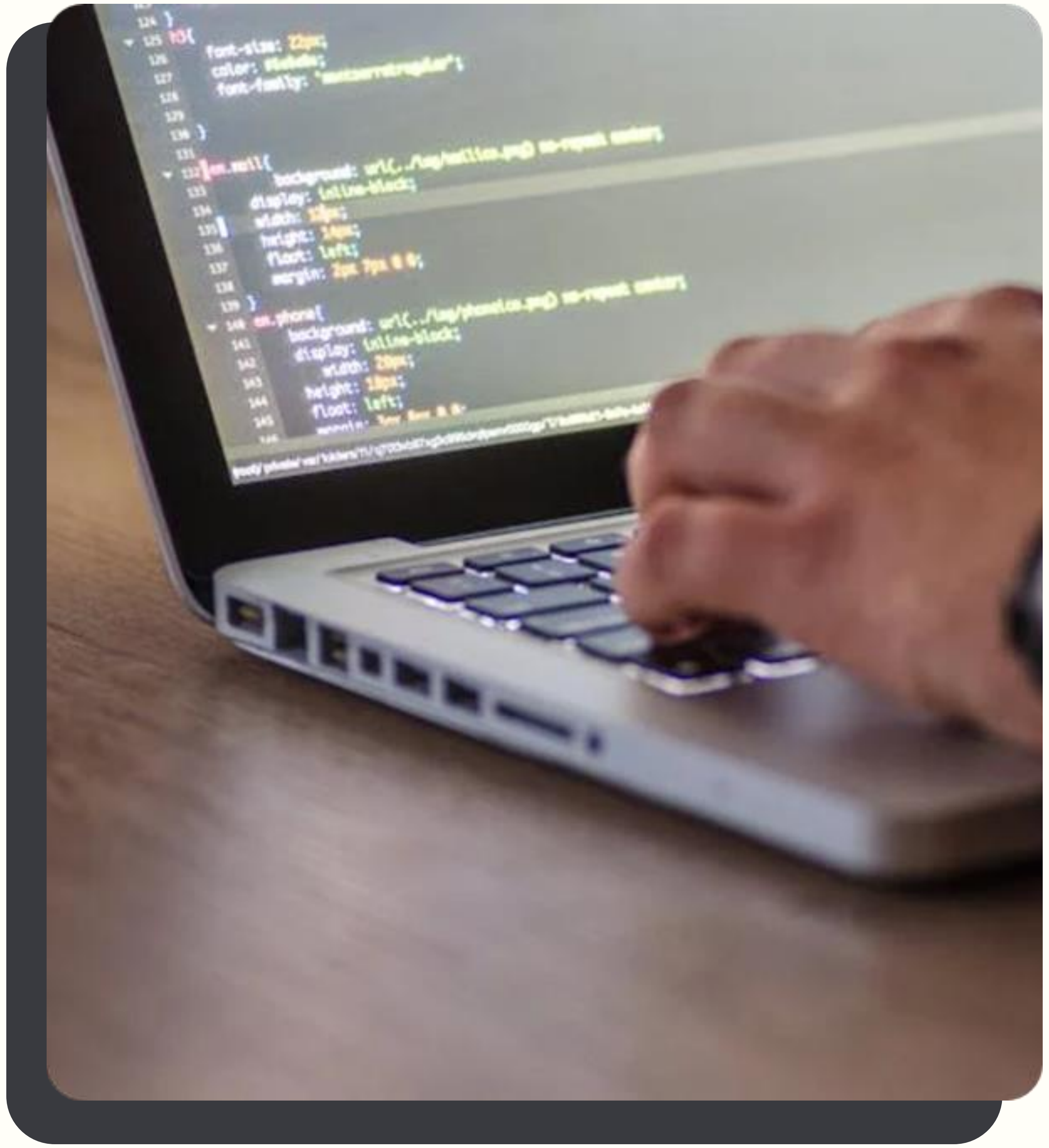
The primary take aways from coding these examples include gaining experience in functions & classes and further understanding object orientated programming through class design and data encapsulation.

Personal skill enhancements:

- I noticed I improved handling control flow and error handling after completing these examples additionally I developed some new and effective debugging strategies that helped me streamline code and solve issues faster.

Future improvements:

- User interface enhancements
- Database integration
- File handling enhancements



End, THANK YOU!