**📖 MTH2008** Scientific Computing Logbook

School of Engineering and Physical Sciences, University of Lincoln

Semester A, 2024

*“I confirm that this logbook is entirely my own work and that all references and quotations, from both primary and secondary sources, have been fully identified and properly acknowledged.”* - William Fayers (27378661).

## 📚 Table of Contents

|  |  |
| --- | --- |
| **Section** | **Page** |
| **Disclaimer** | 1 |
| [**Week 1** – 2024-09-30 to 2024-10-07](#week-1-2024-09-30-to-2024-10-07) | 2 |
| 1.1 [Basic Use: Using Cout](#basic-use-using-cout) | 2 |
| 1.2 [Basic Use: Special Characters](#basic-use-special-characters) | 3 |
| 1.3 [Basic Use: Pascals Triangle](#basic-use-pascals-triangle) | 4 |
| *1.4* [*Pascal Triangle: Extended Project*](#pascal-triangle-extended-project) | 5 |
| [**Week 2** – 2024-10-07 to 2024-10-14](#week-2-2024-10-07-to-2024-10-14) | 6 |
| 2.1 [Basic Use: Variables And Data Types](#basic-use-variables-and-data-types) | 6 |
| 2.2 [Basic Use: Input And Math](#basic-use-input-and-math) | 7 |
| *2.3* [*Coursework: Calculator*](#coursework-calculator) | 8 |
| [**Week 3** – 2024-10-14 to 2024-10-21](#week-3-2024-10-14-to-2024-10-21) | 9 |
| 3.1 [Basic Use: Data Size](#basic-use-data-size) | 9 |
| 3.2 [Libraries: Using Cmath](#libraries-using-cmath) | 10 |
| *3.3* [*Coursework: Trig Calculator*](#coursework-trig-calculator) | 11 |
| [**Week 4** – 2024-10-21 to 2024-10-28](#week-4-2024-10-21-to-2024-10-28) | 12 |
| 4.1 [Control Structures: If Else Statements](#control-structures-if-else-statements) | 12 |
| 4.2 [Control Structures: Input Validation](#control-structures-input-validation) | 13 |
| 4.3 [Control Structures: Simple Calculator](#control-structures-simple-calculator) | 14 |
| *4.4* [*Coursework: Quadratic Calculator*](#coursework-quadratic-calculator) | 15 |
| [**References**](#references) | 16 |

## **Week 1** – 2024-09-30 to 2024-10-07

**What did you learn in the lab tasks this week?**

In the lab session this week we focused on outputting to a data stream and handling special characters. In order to output to a data stream, I learned we use cout to declare console output and insert data into this with the insertion operator, << - this differs from Python’s print function, but is essentially how it works. This is because C++ is a lower level language, just above assembly code, and is thus more lightweight and efficient, e.g. you don’t need input/output capabilities with all programs, as not all programs need a command-line interface, hence we include the line #include <iostream>to access these stream commands.

Additionally, we explored special characters (\n, \a, \b, \f, \r, and \t), escape sequences (\\, \', \", and \?), and the line terminator (endl). The special characters and escape sequences are similar to those used in Python, but the line terminator was completely new for me. Functionally, it seems to act the same as a newline character, but works by ending the data stream started by cout.

**How well did I learn it?**

I learned everything pretty quickly, making parallels with Python quite often. It’s been really fun learning a completely new, more low-level language and deepening my understanding of programming in general. I did some extra work outside of my lab session, too, which definitely helped me reinforce any new knowledge and understand C++ even quicker.

**How does my solution compare with the official solution?**

There aren’t any official solutions right now, so I’ll write this next week if they get released.

**How can I extend the concepts used in the tasks to form a new project**

In my extra work this week, I’ve extended the concepts used to make Pascal’s triangle by changing the logic to a more loop-based system so that I can use render any number of rows. I also tried an alternative to using the tab characters, setd::setw. Not only this, but I decided to take this number of rows as a user input with std::cin and validated the input, throwing an exception if invalid.

I also researched C++ best practices and styling guidelines, referencing documents like Google Style Guides [[1]](#references) and Doxygen [[2]](#references). The main points are learned from this were: consistent comment style (I opted for /\* and /\*\*), function return rationale (i.e. when to use void), explicitly returning 0 in a main function (to tell the console it ran correctly), docstring placement (before a function, contrasting with Python), and function declaration (placing the opening curly bracket on the same line).

Finally, with the project I already explained, I explored functions, for loops, conditional statements, exception handling, and some basic libraries: cmath, stdexcept, and iomanip.

**Lesson Review Question Answers**

1. Facilitates complex simulations; differs by enabling exploration beyond physical limits.
2. Climate modeling, drug discovery, astrophysics simulations.
3. First computer-assisted proof; raises questions about proof validity.
4. Cost/feasibility issues; safety concerns.
5. From serial to parallel computing; enhances computational power for complex problems.
6. Serial: sequential processing; Parallel: simultaneous processing.
7. Physical limits in miniaturization; heat dissipation, quantum effects.
8. FLOPS: Floating Point Operations Per Second; measures computational performance.
9. Binary: two digits (0, 1); used for electronic circuitry efficiency.
10. Encapsulation, inheritance, polymorphism.

Multiple choice: BCCBCCBBBA

### Basic Use: Using Cout

#### Task 1.5

The first bit of C++ code I’ve ever written!

cout << "Welcome to C++ Programming\n";

### Basic Use: Special Characters

#### Task 1.3

The bell character \a makes a sound in some environments.

cout << "With bell character: Welcome to C++ Programming\a" << "---\n";

The backspace character the last character.

cout << "With backspace character: Welcome to C++ Programming\b" << "---\n";

The return character to the start of the line.

cout << "With return character: Welcome to C++ Programming\r" << "---\n";

The tab character a tab space.

cout << "With tab character: Welcome to C++ Programming\t" << "---\n";

#### Task 1.4

Removing the backslash before the single quote causes a syntax error.

cout << "Escaped characters: Is the symbol \\n called \' Newline \' \? \n";

#### Task 2.1

The line terminator endl starts a new line.

cout << "A rose by any other name would smell as sweet";  
cout << endl;

### Basic Use: Pascals Triangle

return 0;  
}

### Pascal Triangle: Extended Project

return 0;  
}

## **Week 2** – 2024-10-07 to 2024-10-14

**What did you learn in the lab tasks this week?**

…

**How well did I learn it?**

…

**How does my solution compare with the official solution?**

…

**How can I extend the concepts used in the tasks to form a new project**

…

**Coursework Question Answers**

1. False, they’re stored as binary numbers.
2. Device.
3. .cpp extension.
4. Translate source code into machine readable code (or exectuable program) that the computer can run.
5. 8 bits.
6. Hexadecimal (base 16).
7. .
8. True.
9. cout is used without the namespace std, and line 5 is missing a semicolon at the end of the line.
10. Missing the insertion operator on line 8, correct line is cin >> num1 >> num2;

### Basic Use: Variables And Data Types

#### Task 1.2

Declare and initalise the variables

int total\_children = 10;  
int candies\_per\_child = 5;  
int extra\_candies = 10;

Output the starting values and the total number of candies

std::cout << "HALLOWEEN CANDY CALCULATOR" << std::endl;  
std::cout << "Total Children: " << total\_children << std::endl;  
std::cout << "Candies per Child: " << candies\_per\_child << std::endl;  
std::cout << "Extra Candies: " << extra\_candies << std::endl;  
std::cout << "---" << std::endl;  
std::cout << "Total Candies: " << total\_candies << std::endl;

Return 0 to indicate the program ran successfully

return 0;

### Basic Use: Input And Math

#### Task 2.3

Declare variables to store the masses and distance.

float mass1, mass2, distance;

We then input the masses and distance.

std::cout << "Enter the mass of the first object (kg): ";  
std::cin >> mass1;  
std::cout << "Enter the mass of the second object (kg): ";  
std::cin >> mass2;  
std::cout << "Enter the distance between the objects (m): ";  
std::cin >> distance;

And finally we call the function, catching any exceptions that occur.

try {  
float force = calculate\_gravitational\_force(mass1, mass2, distance);  
std::cout << "Gravitational Force: " << force << " N" << std::endl;  
} catch (std::invalid\_argument& error) {  
std::cerr << "Error: " << error.what() << std::endl;  
}

### Coursework: Calculator

#### Task 1.1

Input the radius of the circle.

float radius;  
input\_number(radius, "Enter the radius of the circle: ");

Calculate the area and circumference of the circle.

float area = M\_PI \* radius \* radius;  
float circumference = 2 \* M\_PI \* radius;  
std::cout << "Area of a circle with radius " << radius << " = " << area << std::endl;  
std::cout << "Circumference of a circle with radius " << radius << " = " << circumference  
<< std::endl;

#### Task 1.2

Input the principal amount, rate of interest, and time period.

float principal, rate, time;  
input\_number(principal, "Enter the principal amount: ");  
input\_number(rate, "Enter the rate of interest (%): ");  
input\_number(time, "Enter the time period (years): ");

Calculate the simple interest.

float simple\_interest = (principal \* rate \* time) / 100;  
std::cout << "Simple Interest on a principal amount of " << principal << " at a rate of "  
<< rate << "% for " << time << " years = " << simple\_interest << std::endl;

## **Week 3** – 2024-10-14 to 2024-10-21

**What did you learn in the lab tasks this week?**

…

**How well did I learn it?**

…

**How does my solution compare with the official solution?**

…

**How can I extend the concepts used in the tasks to form a new project**

…

**Coursework Question Answers**

Multiple choice: bba. True or false: TTTT (note for the third statement, technically the return statement can be implicit, but it’s always required even if it’s implicitly there).

### Basic Use: Data Size

#### Task 1.1

After reaching the max value of the data type, the number goes to zero.

std::cout << "Original number: " << unsigned\_small\_number << std::endl;  
unsigned\_small\_number++;  
std::cout << "Incremented number (1x): " << unsigned\_small\_number << std::endl;  
unsigned\_small\_number++;  
std::cout << "Incremented number (2x): " << unsigned\_small\_number << std::endl;

#### Task 1.3

Similar to overflow, it loops around as an underflow.

signed\_small\_number = -32768;  
std::cout << "Original number: " << signed\_small\_number << std::endl;  
signed\_small\_number--;  
std::cout << "Decremented number (1x): " << signed\_small\_number << std::endl;  
signed\_small\_number--;  
std::cout << "Decremented number (2x): " << signed\_small\_number << std::endl;

### Libraries: Using Cmath

#### Task 2.1

Calculate the square root of 9.0, which resulted in 3.

std::cout << sqrt(9.0) << std::endl;

Calculate the natural logarithm of 1.1, which resulted in 0.0953102.

std::cout << log(1.1) << std::endl;

Calculate 2.5 raised to the power of 3.0, which resulted in 15.625.

std::cout << pow(2.5, 3.0) << std::endl;

Calculate the ceiling of 4.1, which resulted in 5.

std::cout << ceil(4.1) << std::endl;

Calculate (7 + sqrt(7.0)) / 3.0, which resulted in 3.21525.

std::cout << (7 + sqrt(7.0)) / 3.0 << std::endl;

Calculate the hyperbolic tan of 0.342, which resulted in 0.329262

std::cout << tanh(0.342) << std::endl;

Calculate the remainder of 11.0 divided by 4.0, which resulted in 3.

std::cout << fmod(11.0, 4.0) << std::endl;

Calculate the absolute value of -9, which resulted in 9.

std::cout << abs(-9) << std::endl;

Calculate the (floating) absolute value of -4.5, which resulted in 4.5.

std::cout << fabs(-4.5) << std::endl;

Calculate the floor of 8.8, which resulted in 8.

std::cout << floor(8.8) << std::endl;

Calculate the base-10 logarithm of 1.0, which resulted in 0.

std::cout << log10(1.0) << std::endl;

Calculate the cube root of 10.0, which resulted in 2.15443.

std::cout << cbrt(10.0) << std::endl;

#### Task 2.3

Prompt the user for a value of x.

double x;  
std::cout << "Calculation of the hyperbolic cosecant function..." << std::endl;  
std::cout << "Enter a value for x: ";  
std::cin >> x;

Calculate the hyperbolic cosecant of x using the custom csch function.

double y = csch(x);  
if (!std::isnan(y)) {  
std::cout << "Hyperbolic cosecant of " << x << " is: " << y << std::endl;  
} else {  
std::cout << "Hyperbolic cosecant is undefined for x = 0." << std::endl;  
}

### Coursework: Trig Calculator

#### Task 3.0

Start with a verbose number input function to re-use.

void input\_number(double& number, const std::string& input\_message) {  
while (true) {  
std::cout << input\_message;  
std::cin >> number;  
  
if (std::cin.fail()) {  
std::cerr << "Invalid input. Please enter a valid number." << std::endl;  
std::cin.clear();  
std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
} else {  
std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
return;  
}  
}  
}

Implement the main program loop with the atanh and acosh functions.

int main() {  
while (true) {  
double choice;  
std::cout << "Select an operation:\n1. Inverse Hyperbolic Tangent\n"  
<< "2. Inverse Hyperbolic Cosine\n0. Exit\n";  
input\_number(choice, "Enter your choice: ");  
  
if (choice == 0) {  
break;  
}  
  
double x;  
switch (static\_cast<int>(choice)) {  
case 1:  
input\_number(x, "Enter a value for x (between -1 and 1): ");  
std::cout << "The inverse hyperbolic tangent of " << x << " is " << atanh(x)  
<< std::endl;  
break;  
case 2:  
input\_number(x, "Enter a value for x (greater than or equal to 1): ");  
std::cout << "The inverse hyperbolic cosine of " << x << " is " << acosh(x)  
<< std::endl;  
break;  
default:  
std::cerr << "Error: Invalid operation number." << std::endl;  
break;  
}  
}  
  
return 0;  
}

#### Task 3.1

Implement the atanh function, with error checking.

double atanh(double x) {  
if (x < -1 || x > 1) {  
std::cerr << "\nError: The input value must be between -1 and 1." << std::endl;  
return NAN;  
}  
return 0.5 \* log((1 + x) / (1 - x));  
}

#### Task 3.2

Implement the acosh function, with error checking.

double acosh(double x) {  
if (x < 1) {  
std::cerr << "\nError: The input value must be greater than or equal to 1." << std::endl;  
return NAN;  
}  
return log(x + sqrt(x \* x - 1));  
}

## **Week 4** – 2024-10-21 to 2024-10-28

**What did you learn in the lab tasks this week?**

…

**How well did I learn it?**

…

**How does my solution compare with the official solution?**

…

**How can I extend the concepts used in the tasks to form a new project**

…

**Coursework Question Answers**

### Control Structures: If Else Statements

#### Task 1.1

Copy the function to calculate the first piecewise function.

std::tuple<double, double> evaluate\_function1(double x, double &y) {  
if (x >= 2) {  
std::cout << "x is greater than or equal to 2, so \n";  
std::cout << "y = x^3 + sqrt(x) = ";  
y = pow(x, 3) + sqrt(x);  
} else if (x > 0.5 && x < 2) {  
std::cout << "x is between 0.5 and 2, so \n";  
std::cout << "y = 1 - exp(x) = ";  
y = 1 - exp(x);  
} else {  
std::cout << "x is less than or equal to 0.5, so \n";  
std::cout << "y = |x + 5| = ";  
y = fabs(x + 5);  
}  
  
return std::make\_tuple(x, y);  
}

#### Task 1.3

Evaluate the second piecewise function.

std::tuple<double, double> evaluate\_function2(double x, double &y) {  
if (x < -M\_PI/2) {  
std::cout << "x is less than -pi/2, so \n";  
std::cout << "y = -1 - pi/2 = ";  
y = -1 - M\_PI/2;  
} else if (x >= -M\_PI/2 && x <= 0) {  
std::cout << "x is between -pi/2 and 0, so \n";  
std::cout << "y = sin(x) + x = ";  
y = sin(x) + x;  
} else {  
std::cout << "x is greater than 0, so \n";  
std::cout << "y = x = ";  
y = x;  
}  
  
return std::make\_tuple(x, y);  
}

### Control Structures: Input Validation

#### Task 2.0

Initalises the constant variables.

const double banned\_number = 2.0;  
const double small\_number = 10e-12;

Fetch input from the user.

double x;  
std::cout << "Enter a value for x (other than 2.0): ";  
std::cin >> x;

Calculate the function and output the result.

cout << "When x = " << x << ", the function y = 1/(x^2 - 2) is equal to ";  
double y = 1 / (std::pow(x, 2.0) - 2.0);  
cout << y << endl;

### Control Structures: Simple Calculator

#### Task 3.0

Create a function to input a number, avoiding a banned number.

void input\_number(const std::string &input\_message, double &number, const double &banned\_number){  
const double float\_error\_resolution = 10e-12;  
  
while (true){  
std::cout << input\_message;  
std::cin >> number;  
  
if (std::cin.fail()){  
std::cerr << "Invalid input. Please enter a valid number." << std::endl;  
std::cin.clear();  
std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
} else if (std::fabs(number - banned\_number) < float\_error\_resolution){  
std::cerr << "Error: x cannot be " << banned\_number << std::endl;  
std::cin.clear();  
std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
} else {  
std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
return;  
}  
}  
}

### Coursework: Quadratic Calculator

#### Task 5.0

Include the necessary libraries.

#include <cmath>  
#include <iostream>  
#include <limits>  
#include <tuple>

Start with a verbose number input function to re-use.

void input\_number(double& number, const std::string& input\_message) {  
while (true) {  
std::cout << input\_message;  
std::cin >> number;  
  
if (std::cin.fail()) {  
std::cerr << "Invalid input. Please enter a valid number." << std::endl;  
std::cin.clear();  
std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
} else {  
std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
return;  
}  
}  
}

Calculate the quadratic’s solution.

std::tuple<double, double> solve\_quadratic(const double a, const double b, const double c) {  
double discriminant = pow(b, 2) - 4 \* a \* c;  
if (discriminant < 0) {  
return std::make\_tuple(NAN, NAN);  
}  
  
double x1 = (-b + sqrt(discriminant)) / (2 \* a);  
double x2 = (-b - sqrt(discriminant)) / (2 \* a);  
  
return std::make\_tuple(x1, x2);  
}

Implement the main program loop.

int main() {  
while (true) {  
double a, b, c;  
input\_number(a, "Enter the coefficient of x^2: ");  
input\_number(b, "Enter the coefficient of x: ");  
input\_number(c, "Enter the constant term: ");  
  
std::tuple<double, double> x = solve\_quadratic(a, b, c);  
  
std::cout << "The quadratic equation " << a << "x^2 + " << b << "x + " << c << " = 0 has ";  
  
if (std::isnan(std::get<0>(x))) {  
std::cout << "no real solutions." << std::endl;  
} else if (std::get<0>(x) == std::get<1>(x)) {  
std::cout << "one solution: x = " << std::get<0>(x)  
<< std::endl;  
} else {  
std::cout << "two solutions: x = " << std::get<0>(x)  
<< " and x = " << std::get<1>(x) << std::endl;  
}  
  
char response;  
std::cout << "Would you like to solve another quadratic equation? (y/n): ";  
std::cin >> response;  
  
if (response != 'y') {  
break;  
}  
}  
  
return 0;  
}

## 📓 References

[1] “A comprehensive guide to C++ coding standards and best practices,” *C++ Style Guide*, 2024. [Online] Available: https://google.github.io/styleguide/cppguide.html (accessed 2024-10-05).

[2] “Official documentation for Doxygen’s documentation blocks and usage,” *Doxygen Manual - Documentation Blocks*, 2024. [Online] Available: https://www.doxygen.nl/manual/docblocks.html (accessed 2024-10-05).