🖊️ **MTH2008** Scientific Computing Coursework

School of Engineering and Physical Sciences, University of Lincoln

Tuesday 8th October, 2024

*This document was generated in the same way as my logbook, hence its appearance. The tasks are broken into sections; in task 1, the main parts of code are shown and the entire script is found at the of the document.* - William Fayers (27378661)

## 📚 Table of Contents

| **Section** | **Page** |
| --- | --- |
| [**Task 1** - Creating a calculator](#task-1-creating-a-calculator) | 2 |
| 1.1 [Circle Calculations](#circle-calculations) | 2 |
| 1.2 [Simple Interest](#simple-interest) | 2 |
| [**Task 2** - Questions](#task-2-questions) | 3 |
| [*CPP Scripts*](#cpp-script) | 4 |

### Task 1 (Creating a calculator)

#### Circle Calculations

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;

#### Simple Interest

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;

### Task 2 (Questions)

1. True.
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;

### CPP Script

#include <iostream>  
#include <cmath>  
#include <limits>  
#include <string>  
#include <functional>  
#include <map>  
  
/\*\*  
 \* @file e01-coursework-calculator.cpp  
 \* @brief A simple calculator program for basic mathematical operations.  
 \* @details This program allows the user to perform basic mathematical operations  
 \* defined in the operation map. The user can select an operation from the menu  
 \* and input the required values to perform the operation.  
 \*/  
  
  
/\*\*  
 \* @brief Fetch a number from the user with error checking.  
 \*   
 \* @param number   
 \* @param input\_message   
 \*/  
void input\_number(float& number, const std::string& input\_message) {  
 while (true) {  
 /\* Ask the user for a number \*/  
 std::cout << input\_message;  
 std::cin >> number;  
  
 /\* Check if the input is valid \*/  
 if (std::cin.fail() || number == 0) {  
 /\* Display an error message \*/  
 std::cerr << "Invalid input. Please enter a valid number." << std::endl;  
  
 if (number == 0) {  
 std::cerr << "Zero is not a valid input, because of division by zero." << std::endl;  
 }  
  
 /\* Clear the error flag and ignore the rest of the input \*/  
 std::cin.clear();  
 std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
 } else {  
 /\* Discard any extra characters in the input buffer \*/  
 std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');  
  
 /\* Exit the loop if the input is valid \*/  
 return;  
 }  
 }  
}  
  
  
/\*\*  
 \* @brief Add two numbers and output the result.  
 \*/  
void perform\_addition() {  
 /\* Input the two numbers \*/  
 float number1, number2;  
 input\_number(number1, "Enter the first number: ");  
 input\_number(number2, "Enter the second number: ");  
  
 /\* Calculate the sum \*/  
 std::cout << number1 << " + " << number2 << " = " << (number1 + number2) << std::endl;  
}  
  
  
/\*\*  
 \* @brief Subtract two numbers and output the result.  
 \*/  
void perform\_subtraction() {  
 /\* Input the two numbers \*/  
 float number1, number2;  
 input\_number(number1, "Enter the first number: ");  
 input\_number(number2, "Enter the second number: ");  
  
 /\* Calculate the difference \*/  
 std::cout << number1 << " - " << number2 << " = " << (number1 - number2) << std::endl;  
}  
  
  
/\*\*  
 \* @brief Multiply two numbers and output the result.  
 \*/  
void perform\_multiplication() {  
 /\* Input the two numbers \*/  
 float number1, number2;  
 input\_number(number1, "Enter the first number: ");  
 input\_number(number2, "Enter the second number: ");  
  
 /\* Calculate the product \*/  
 std::cout << number1 << " \* " << number2 << " = " << (number1 \* number2) << std::endl;  
}  
  
  
/\*\*  
 \* @brief Calculate the modulo of two numbers and output the result.  
 \*/  
void perform\_modulo() {  
 /\* Input the two numbers \*/  
 float number1, number2;  
 input\_number(number1, "Enter the first number: ");  
 input\_number(number2, "Enter the second number: ");  
  
 /\* Calculate the modulo \*/  
 std::cout << number1 << " mod " << number2 << " = "  
 << (number1 - (static\_cast<int>(number1 / number2) \* number2)) << std::endl;  
}  
  
  
/\*\*  
 \* @brief Calculate the area and circumference of a circle.  
 \*/  
void perform\_circle\_calculations() {  
 /\* Input the radius of the circle \*/  
 float radius;  
 input\_number(radius, "Enter the radius of the circle: ");  
 /\* ANSWER (Task 1.1): Input 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;  
 /\* ANSWER (Task 1.1): Calculate the area and circumference of the circle. \*/  
}  
  
  
/\*\*  
 \* @brief Calculate the simple interest.  
 \*/  
void perform\_simple\_interest() {  
 /\* 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): ");  
 /\* ANSWER (Task 1.2): Input the principal amount, rate of interest, and time period. \*/  
  
 /\* 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;  
 /\* ANSWER (Task 1.2): Calculate the simple interest. \*/  
}  
  
  
/\*\*  
 \* @brief Main function to run the program.  
 \*  
 \* @details This function displays a menu of operations to the user and performs the selected  
 \* operation. The program continues to run until the user chooses to exit.  
 \*   
 \* @return int   
 \*/  
int main() {  
 /\* Map the operation index to the corresponding function \*/  
 std::map<int, std::pair<std::function<void()>, std::string>> operations = {  
 {1, {perform\_addition, "Addition"}},  
 {2, {perform\_subtraction, "Subtraction"}},  
 {3, {perform\_multiplication, "Multiplication"}},  
 {4, {perform\_modulo, "Modulo"}},  
 {5, {perform\_circle\_calculations, "Area and Circumference of a Circle"}},  
 {6, {perform\_simple\_interest, "Simple Interest"}}  
 };  
  
 /\* Run the program until the user chooses to exit \*/  
 while (true) {  
 /\* Display the menu of operations \*/  
 std::cout << "Select an operation to perform:" << std::endl;  
 for (const auto& operation : operations) {  
 std::cout << operation.first << ". " << operation.second.second << std::endl;  
 }  
 std::cout << operations.size() + 1 << ". Exit" << std::endl;  
  
 /\* Input the operation \*/  
 float operation\_input;  
 input\_number(operation\_input, "Enter the operation number: ");  
 int operation = static\_cast<int>(operation\_input);  
  
 /\* Exit the program if the user chooses to exit \*/  
 if (operation == static\_cast<int>(operations.size() + 1)) {  
 break;  
 }  
  
 /\* Check if the operation is valid \*/  
 if (operations.find(operation) != operations.end()) {  
 /\* Perform the selected operation \*/  
 operations[operation].first();  
 } else {  
 /\* Display an error message for an invalid operation \*/  
 std::cerr << "Invalid operation. Please select a valid operation." << std::endl;  
 }  
 }  
  
 /\* Return 0 to indicate the program ran successfully \*/  
 return 0;  
}