Mathematical Operations

Documentation

Header file: kaur_mathops.h

```
#pragma once

 * --Adds two integers.
 * --parameter a the first integer.
 * --param b The second integer.
 * --return The sum of a and b.

#define add_int(a, b) ((a) + (b))

/**
 * --brief Subtracts the second integer from the first.
 * --param a The first integer.
 * --param b The second integer.
 * --return The result of a - b.
 */
#define sub_int(a, b) ((a) - (b))

/**
 * -- Multiplies two integers.
 * --param a the first integer.
 * --param b The second integer.
 * --return the product of a and b.
 */
#define mult_int(a, b) ((a) * (b))

/**
 * -- Divides the first integer by the second, checking for division by zero.
 * --param a The first integer.
 * --param b The second integer.
 * --return The result of a / b if b is not zero; otherwise, returns 0 to have no error.
 * --This macro includes a check to avoid division by zero.
 */
#define div_int(a, b) ((b) != 0 ? ((a) / (b)) : 0)

/**
 * --brief Computes the remainder of the division of two integers.
 * --param a The dividend.
 * --param b The divisor.
 * --return The remainder of a / b.
 * --note This macro does not check for division by zero.
 */
#define MOD_INT(a, b) ((a) % (b))

// Defining macros for floating-point operations

/**
```

```
 * --adds two floating-point numbers.
 * --param a The first number.
 * --param b The second number.
 * --return The sum of a and b.
 */
#define add_float(a, b) ((a) + (b))

/**
 * --brief Subtracts the second floating-point number from the first.
 * --param a The first number.
 * --param b The second number.
 * --return The result of a - b.
 */
#define sub_float(a, b) ((a) - (b))

/**
 * --brief Multiplies two floating-point numbers.
 * --param a The first number.
 * --param b The second number.
 * --return The product of a and b.
 */
#define mult_float(a, b) ((a) * (b))

/**
 * --brief Divides the first floating-point number by the second, checking for division by zero.
 * --param a The first number.
 * --param b The second number.
 * --return The result of a / b if b is not zero; otherwise, returns 0.0f.
 * --This macro includes a check to avoid division by zero.
 */
#define div_float(a, b) ((b) != 0.0f ? ((a) / (b)) : 0.0f)

#include <math.h>
// Include the custom header file
#include "kaur_mathops.h"

// Function prototypes for power and factorial

/**
 * --Computes the power of a number.
* --return The result of base raised to the power of exp.
 */
double power(double base, int exp);

/**
 * --Computes the factorial of a non-negative integer.
 * --param n The integer for which the factorial is to be computed.
 * --return The factorial of n. If n is 0, returns 1. Returns -1 for negative inputs.
 */
int factorial(int n);
```

Implementation file: kaur_implementation.c

```
#include "kaur_mathops.h"

// Function definition for power that implements using the pow function from the math library

/**
 * --Computes the power of a number.
 * --param base The base number.
 * --param exp The exponent.
 * --return The result of base raised to the power of exp.
 * --This function uses the pow function from the math library.
 */
double power(double base, int exp) {
    return pow(base, exp);
}

// Function definition for factorial using int

/**
 * --Computes the factorial of a non-negative integer.
 * --param n The integer for which the factorial is to be computed.
 * --return The factorial of n. If n is 0 or 1, returns 1. If n is negative, returns 0 as factorial of a
negative number is undefined.
 * -- This function uses an iterative approach to calculate the factorial.
 */
int factorial(int n) {
    if (n < 0) {
        return 0; // Factorial of a negative number is undefined
    }
    if (n == 0 || n == 1) {
        return 1;
    }
    int result = 1;
    for (int i = 2; i <= n; ++i) {
        result *= i;
    }
    return result;
}
```

Test program: kaur_test.c

```
#include <stdio.h> // for printing and scanning functions
#include "kaur_mathops.h"

/--returns 0 upon successful execution

int main() {
    int a, b, n, choice;
    float x, y;
    double base;
    int exp;

    while (1) { //an infinite loop that will keep the program running
        //below is the menu of choices that will show up
        printf("**********************");
        printf("\nSimple Calculator\n");
        printf("1. Add two integers\n");
        printf("2. Subtract two integers\n");
        printf("3. Multiply two integers\n");
        printf("4. Divide two integers\n");
        printf("5. Add two floating-point numbers\n");
        printf("6. Subtract two floating-point numbers\n");
        printf("7. Multiply two floating-point numbers\n");
        printf("8. Divide two floating-point numbers\n");
        printf("9. Calculate power\n");
        printf("10. Calculate factorial\n");
        printf("11. Modulo two integers\n");
        printf("12. Exit\n");
        printf("Enter your choice: ");
        scanf_s("%d", &choice); //reads the choice selection of the user
        printf("*********************\n");

//executes different choices based on users inputed numbers of choice
        switch (choice) {
        case 1:
        // -- Addition of two integers
            printf("Enter two integers: ");
            scanf_s("%d %d", &a, &b);
            printf("Result: %d\n", add_int(a, b));
            break;
        case 2:
        // -- Subtraction of two integers
            printf("Enter two integers: ");
            scanf_s("%d %d", &a, &b);
            printf("Result: %d\n", sub_int(a, b));
            break;
        case 3:
         //-- Multiplication of 2 integers
            printf("Enter two integers: ");
            scanf_s("%d %d", &a, &b);
            printf("Result: %d\n", mult_int(a, b));
            break;
        case 4:
      // -- division of 2 integers
```

```
        printf("Enter two integers: ");
        scanf_s("%d %d", &a, &b);
        printf("Result: %d\n", div_int(a, b));
        break;
    case 5:
  // -- addition of 2 floating-point numbers
        printf("Enter two floating-point numbers: ");
        scanf_s("%f %f", &x, &y);
        printf("Result: %.2f\n", add_float(x, y));
        break;
    case 6:
   // -- subtraction of 2 floating-point numbers
        printf("Enter two floating-point numbers: ");
        scanf_s("%f %f", &x, &y);
        printf("Result: %.2f\n", sub_float(x, y));
        break;
    case 7:
   // -- multiplication of 2 floating-point numbers
        printf("Enter two floating-point numbers: ");
        scanf_s("%f %f", &x, &y);
        printf("Result: %.2f\n", mult_float(x, y));
        break;
    case 8:
    // -- division of 2 floating-point numbers
        printf("Enter two floating-point numbers: ");
        scanf_s("%f %f", &x, &y);
        printf("Result: %.2f\n", div_float(x, y));
        break;
    case 9:
    // -- calculation of power
        printf("Enter base and exponent: ");
        scanf_s("%lf %d", &base, &exp);
        printf("Result: %.2f\n", power(base, exp));
        break;
    case 10:
      //--calculates factorial
        printf("Enter an integer: ");
        scanf_s("%d", &n);
        printf("Result: %d\n", factorial(n));
        break;
    case 11:
    //-- modulo operation of two integers
        printf("Enter two integers: ");
        scanf_s("%d %d", &a, &b);
        if (b != 0) {
            printf("Result: %d\n", MOD_INT(a, b));
        }
        else {
            printf("Error: Division by zero is not allowed.\n");
        }
        break;
    case 12:
        return 0;
   //-- Exits the program
    default:
            printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
```