```c
/* @file          01-prog.c
 * @author        morgan bergen
 * @date          jan 26 2023
 * @brief         program 1 for lab01
 * @description
 *
 * write a program that will work as a basic calculator that can perform:
 * addition, subtraction, multiplication, and division on a user input data for two operands
 * '5 => + => 6' user input will print result 'sum of 5 and 6 is 11'
 */


/* stdio          standard input/output library functions
 * library        standard C library (libc)
 * @brief         the standard I/O library provides a simple buffered steam I/O interface.
 *                input and output is mapped into logical data streams and the physical
 *                I/O characters are concealed from the programmer.
 *
 * #include      preprocessor directive
 * @brief         this directive tells the compiler to include the contents of the specified
 *                file, in our case this file is stdio.h which is a header file that was explained
 *                above.  the header file contains declarations for functions, variables,
 *                and other constructs used in the program.  after the compiler reads directive
 *                it will include all the contents of the header file in this program, even
 *                though we never explicitly wrote the contents of the header file.
 *
 *                this allows for code reuse and modularization, as well as separating the interface
 *                from the implementation of functions and variables.
 */
#include <stdio.h>

/* main           main function
 * @brief         the main function is the initial entry point of this c program.  it is the first function in
 *                line to be called when the program get executed by the operating system and it is the only
 *                function is has a guarantee to be called by the C standard library.
 * @pre           none
 * @param         none
 * @post          none
 * @return        int
 * @brief         as you can see the data type affixed to the main function is of type integer.
 *                the return value of this program 0 indicates that the program executed successfully
 *                and is essentially the exit status of main. if return(-1) was called at the end of
 *                this main function, then the program would have failed to execute successfully.
 */
int main() {

    /* char       character primitive data type
     * @brief     the character primitive data type can store a single ASCII standard character.
     *            the size of the character is 1 byte == 8 bits
     *            the range of values that can be stored in a character is 0 to 255
     *            ASCII standard characters are defined in the ASCII table which can be found here:
     *
     *            https://en.wikipedia.org/wiki/ASCII
     *            ASCII stands for American Standard Code for Information Interchage
     *            ascii unicode is the numerical representation of characters such as '+' or '5'
     *
     *            '+' == 43_(10) == 0x2B_(16) == 0b00101011_(2)
     *            '-' == 45_(10) == 0x2D_(16) == 0b00101101_(2)
     *            '*' == 42_(10) == 0x2A_(16) == 0b00101010_(2)
     *            '/' == 47_(10) == 0x2F_(16) == 0b00101111_(2)
     *            '5' == 53_(10) == 0x35_(16) == 0b00110101_(2)
     *
     * char operator;
     *
     *
     *            variable declaration identifier name is operator of type char
     * @brief     in c all variables must be declared before they are used/manipulated/assigned, usually
     *            stylistically declarations appear at the beginning of the function before any executable statements
     *            declaration announces the properties of the variables to the compiler such as the identifier name,
     *            and tells the compiler how much memory to allocate in the run-time stack / activation record
     *            for the variable.  the compiler will also check to make sure that the variable is not already
     *            another variable declared in the same scope, and will throw an error if it is.
     *            the run-time stack is a stack LIFO data structure that is used to store local variables and
     *            function parameters, in our case the stack will allocate 8 bits of memory for the variable operator
     */
    char operator;

    /* int        integer primitive data type
     * @brief     the integer primitive data type can store both positive or negative whole numbers
     *            the size of an int is 4 bytes == 32 bits
     *            the range of possible values that can be stored in an int is -2,147,483,648 to 2,147,483,647
     *            the compiler will allocate 92 bits of memory in total for valuel and value2
     */


    int value1, value2, total;

    /* printf     general-purpose output formatting function / standard output function
     * @brief     the printf() family of functions produce output according to a specific format
     *            other functions in the family include fprintf(), sprintf(), snprintf(), etc.
     *            the printf() function write output to stdout, the standard output stream
     *            the argument(s) surrounding the parenthese in our case is a sequence of characters known
     *            as a character string literal or string constant.  this will be presented to the user
     *            upon execution of the program as a prompt for this calculator program
     * @note      printf never supplied a newline character automatically
     */
    printf("enter a calculation: ");

    // scanf reads input from the user
    // %d reads a double value
    // svalue stores the value in the variable value

    /* scanf      standard library input function
     * @brief     the scanf() family of functions reads input according to a specific format, the format may contain
     *            something called conversion specifiers which are used to specify the type of input to be read
     *            and are stored through the pointer arguments that follow the format string
     *            The scanf function reads usually the keyboard and converts them into values of the specified types
     *
     * %          conversion specifier
     * @brief     this is used as a placeholder in a format string that is passed to various input functions
     *            each % construction in the first argument of printf is paired with a corresponding argument
     *            characters that follow the % specify the data type of input to be read
     *
     * %d         decimal integer
     *            this forat specifier is for an integer value in base 10 for the variable value1
     *
     * %c         character
     *            this format specifier is for a character value for the variable operator
     *
     * &          address-of operator
     * @brief     this operator is used to reference the address of the variable in memory
     *            in our case the address of the variable value1 is passed into the function scanf and the value
     *            after the user presses enter will be stored at the address of value1
     *
     * &value1    pass by reference variable value1
     * @brief     value1's memory address 0x16d00f204 is passed to scanf so that the function can store the value
     *            read in from the users input directly into value1's memory location
     */
    scanf("%d", &value1);
    scanf(" %c", &operator);
    scanf("%d", &value2);

    /*
     * if-else    conditional statement
     * @brief     the if-else statement is a conditional statement that executes a block of code if the condition
     *            is true, otherwise it will execute another block of code
     */


    /* if         conditional statement
     * @brief     the if statement is a conditional statement results in a boolean value of true or false
     *            proceeding the parenthese is the condition that will be evaluated to determine if the block
     *            of code following the if statement will be executed depending on the result of the condition
     *
     * operator == '+'
     * @brief     == operator is a comparison operator that compares the value of the left and right operands
     *            this checks the value of operator and compares it to the value of the character '+'
     *            if the value of operator is equal to the value of the character '+' then the condition is true
     */

    // conditional for sum operation
    if (operator == '+') {
        /*
         * total    variable declaration identifier name is total of type int
         * @brief   the variable total is declared and initialized to the value of value1 + value2
         */
        total = value1 + value2;

        // print statments for user output
        printf("sum of ");
        printf("%d", value1);
        printf(" and ");
        printf("%d", value2);
        printf(" is ");
        printf("%d", total);

    // conditional for subtraction operation
    } else if (operator == '-') {

        /*
         * total    variable declaration identifier name is total of type int
         * @brief   the variable total is declared and initialized to the value of value1 - value2
         */
        total = value1 - value2;

        // print statements for user output
        printf("subtraction of ");
        printf("%d", value1);
        printf(" and ");
        printf("%d", value2);
        printf(" is ");
        printf("%d", total);

    // conditional for multiplication operation
    } else if (operator == '*') {

        /*
         * total    variable declaration identifier name is total of type int
         * @brief   the variable total is declared and initialized to the value of value1 * value2
         */
        total = value1 * value2;

        // print statements for user output
        printf("multiplication of ");
        printf("%d", value1);
        printf(" and ");
        printf("%d", value2);
        printf(" is ");
        printf("%d", total);

    // conditional for division operation
    } else if (operator == '/') {

        /*
         * total    variable declaration identifier name is total of type int
         * @brief   the variable total is declared and initialized to the value of value1 / value2
         */
        total = value1 / value2;

        // print statements for user output
        printf("division of ");
        printf("%d", value1);
        printf(" and ");
        printf("%d", value2);
        printf(" is ");
        printf("%d", total);

    } else {

        // edge case if user input is invalid
        printf("invalid operator");
    }
    // adding an additional line for readability
    printf("\n");

    /* return(0);
     * @brief   the return statement is used to exit a function and return a value to the calling function
     */
    return(0);
}
```