```c
/*
 * CE2812 - 021
 * Winter 2016
 * Lab 4 - CALCULATOR
 * Name: Yahui Liang
 * Created: 01/05/2017
 */

/* Header files */
#include "keypad_api.h"
#include "lcd_api.h"
#include <inttypes.h>
#include <ctype.h>
#include <math.h>

/* The structure which represents a calculation equation */
typedef struct {
	int num1;
	int num2;
	uint8_t operator;
	uint8_t equal_sign;
} calculation;

/* private methods prototypes */
static void print_inputs(char);
static void get_inputs(uint8_t[]);
static calculation get_equation(uint8_t[]);
static int get_value_of_the_digit_array(uint8_t[], int);

/**
 * The program is a calculator which can do
 * basic calculations.
 */
int main() {
	/* Initiate peripherals */
	key_init();
	lcd_init();
	/* Repeat asking for inputs and computing results */
	while (1) {
		uint8_t inputs[32];
		get_inputs(inputs);
		calculation equation = get_equation(inputs);
		int result;
		switch ((char)equation.operator) {
			case 'A':
				result = equation.num1 + equation.num2;
				break;
			case 'B':
				result = equation.num1 - equation.num2;
				break;
			case 'C':
				result = equation.num1 * equation.num2;
				break;
			case 'D':
				result = equation.num1 / equation.num2;
				break;
		}
```

```
            /* clear lcd for displaying result */
            lcd_clear();
            lcd_home();
            lcd_print_num(result); // display the final result
      }
      return 0;
}

/**
 * Gets inputs from the user.
 */
static void get_inputs(uint8_t inputs[]) {
      char key_input = key_getchar();
      lcd_clear();
      lcd_home();
      int num_elements = 0;
      /* finish getting inputs if the user types '#' because '#'
       * is also '=' in our calculator. */
      while (key_input != '#' && num_elements < 32) {
            print_inputs(key_input);
            inputs[num_elements] = (uint8_t) key_input;
            num_elements++;
            key_input = key_getchar();
      }
      print_inputs(key_input);
      inputs[num_elements] = (uint8_t) key_input;
}

/**
 * The method fills the calculation structure by iterating through
 * the inputs array.
 * Args:
 * inputs: the inputs array.
 * Returns the calculation structure which is filled with numbers and the
operator.
 */
static calculation get_equation(uint8_t inputs[]) {
      calculation nums_and_ops;
      int adding_digit_index = 0;
      uint8_t one_input = inputs[0];
      int num_of_digits = 0;
      uint8_t num1[32];
      uint8_t num2[32];
      int inputs_size = 32;
      int operator_index = 0;
      /* get the first number */
      /* Repeat scanning inputs until the operator is reached. */
      for (int i = 1; i < inputs_size && isdigit(one_input); i++) {
            num1[adding_digit_index] = one_input - 0x30;
            one_input = inputs[i];
            num_of_digits++;
            adding_digit_index++;
            operator_index = i;
      }
      int num1_value = get_value_of_the_digit_array(num1, num_of_digits);
      nums_and_ops.num1 = num1_value;
      /* get the operator */
```

```
	if (adding_digit_index < inputs_size) {
		nums_and_ops.operator = inputs[operator_index];
	}
	/* get the second number */
	num_of_digits = 0;
	one_input = inputs[operator_index + 1];
	adding_digit_index = 0;
	int equal_sign_index = 0;
	/* Repeats scanning inputs until the equal sign is reached. */
	for (int i = operator_index + 2; i < inputs_size && isdigit(one_input);
i++) {
		num2[adding_digit_index] = one_input - 0x30;
		one_input = inputs[i];
		num_of_digits++;
		adding_digit_index++;
		equal_sign_index = i;
	}
	int num2_value = get_value_of_the_digit_array(num2, num_of_digits);
	nums_and_ops.num2 = num2_value;
	/* get the equal sign */
	if (adding_digit_index < inputs_size) {
		nums_and_ops.equal_sign = inputs[equal_sign_index];
	}
	return nums_and_ops;
}

/**
 * The method can determine what the number is by iterating through an array
which stores
 * all digits of the number.
 * Args:
 * num: the array which stores all digits of the number.
 * num_of_digits: how many digits the number has.
 * Returns the value of the number.
 */
static int get_value_of_the_digit_array(uint8_t num[], int num_of_digits) {
	int num_value = 0;
	int max_num_loops = num_of_digits;
	for (int i = 0; i < max_num_loops; i++) {
		num_value += num[i] * pow(10, (num_of_digits - 1));
		num_of_digits--;
	}
	return num_value;
}

/**
 * Prints out different inputs to the lcd screen.
 * Args:
 * key_pressed: the key which is pressed.
 */
static void print_inputs(char key_pressed) {
	char printable[2];
	printable[0] = key_pressed;
	printable[1] = 0;
	switch (key_pressed) {
		case 'A':
			lcd_print_string("+");
```

```
                        break;
                case 'B':
                        lcd_print_string("-");
                        break;
                case 'C':
                        lcd_print_string("*");
                        break;
                case 'D':
                        lcd_print_string("/");
                        break;
                case '#':
                        lcd_print_string("=");
                        break;
                default:
                        lcd_print_string(printable);
        }
}
```