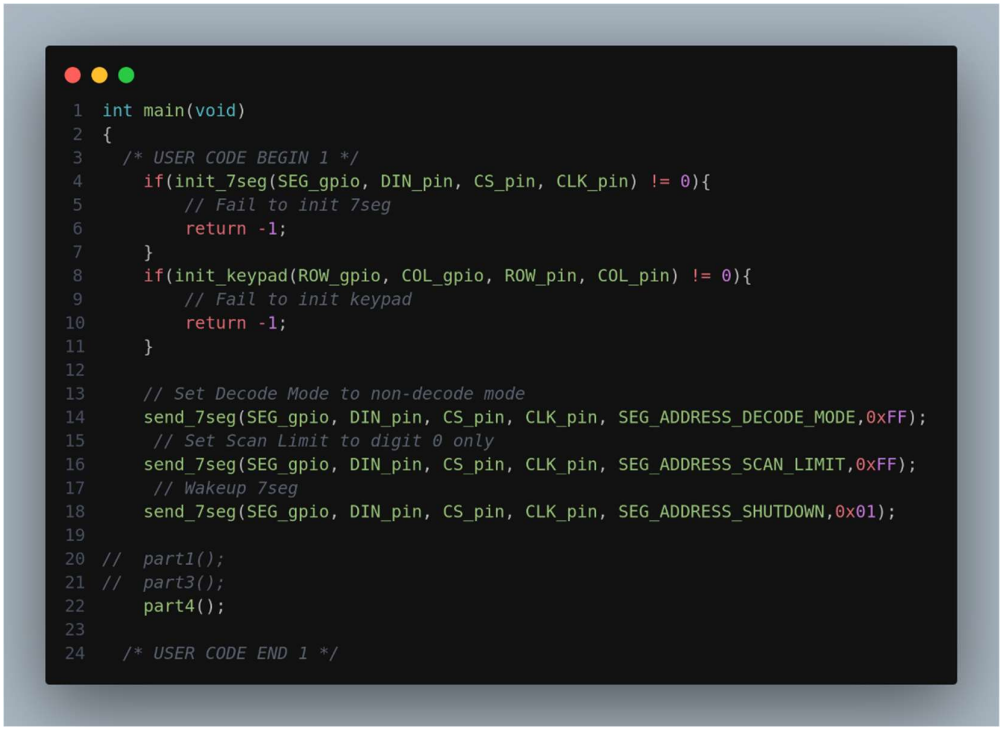
微處理機 LAB 4 KeyPad

- 在這個作業中，我們將三個 **part** 分別包成一個函式，在 **main()**中執行，並在開始前先初始化 **keypad** 和七段顯示器



```c
int main(void)
{
  /* USER CODE BEGIN 1 */
    if(init_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin) != 0){
        // Fail to init 7seg
        return -1;
    }
    if(init_keypad(ROW_gpio, COL_gpio, ROW_pin, COL_pin) != 0){
        // Fail to init keypad
        return -1;
    }

    // Set Decode Mode to non-decode mode
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DECODE_MODE,0xFF);
     // Set Scan Limit to digit 0 only
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SCAN_LIMIT,0xFF);
     // Wakeup 7seg
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SHUTDOWN,0x01);

//  part1();
//  part3();
    part4();

  /* USER CODE END 1 */
```

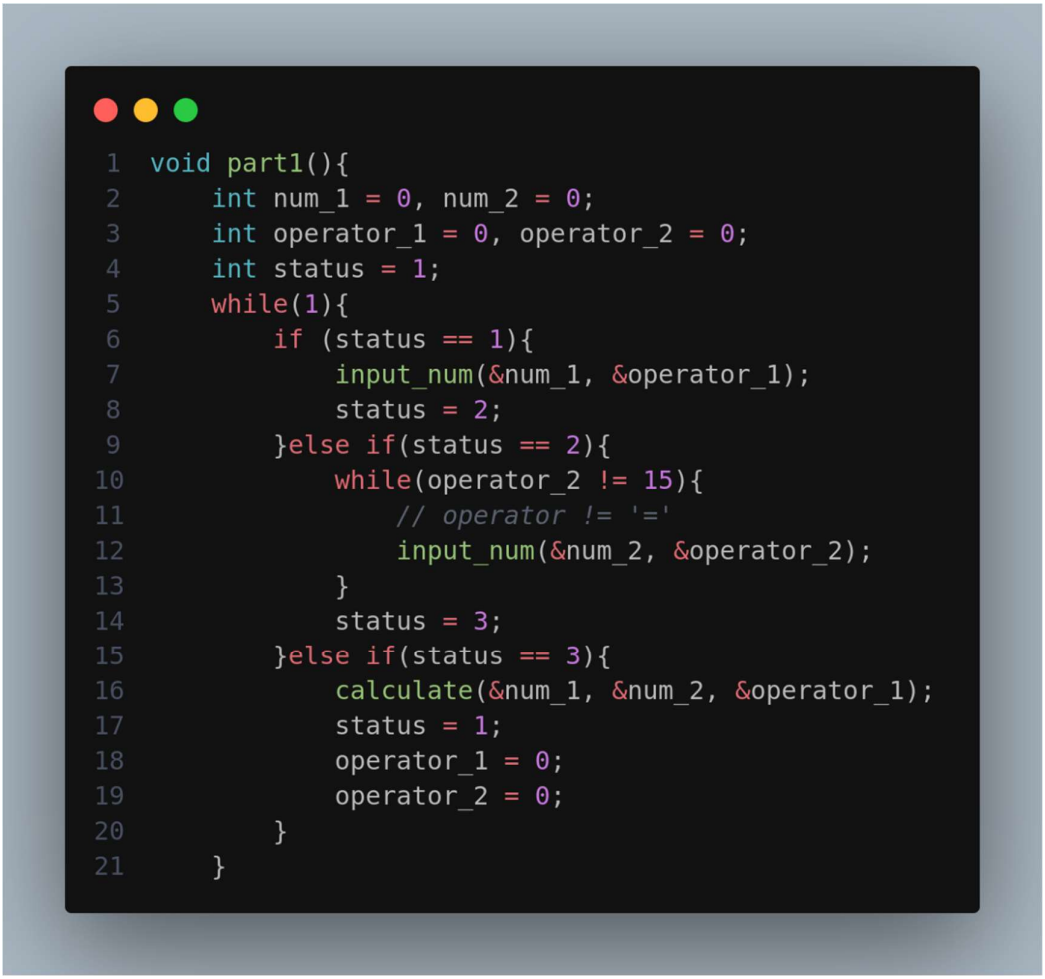
- 一些用來減少迴圈數量及程式碼長度的宣告

```c
int check_press(int i, int j){
    return check_keypad_input_one(ROW_gpio, COL_gpio, ROW_pin, COL_pin, i, j);
}
void display(int a, int b){
    display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin, a, b);
}
int keypad_space[16][2]= { {0,0}, {0,1}, {0,2}, {0,3},
                           {1,0}, {1,1}, {1,2}, {1,3},
                           {2,0}, {2,1}, {2,2}, {2,3},
                           {3,0}, {3,1}, {3,2}, {3,3}};

int get_keyboard_input(){
    for(int i=0;i<16;i++){
        int a = keypad_space[i][0];
        int b = keypad_space[i][1];
        if(check_press(a,b)){
            while(check_press(a,b)!=0) continue;
            return keypad[a][b];
        }
    }
    return -1;
}
```

- **Part 1**

```c
void input_num(int *num, int *operator){
    *num = 0;
    int input = 0;
    int times = 0;
    int negative = 0;
    while(1){
        if(negative == 1 && times == 0)
            send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, 1,10);
        else
            display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin, *num, num_digits(*num));
        input = get_keyboard_input();

        if (input == -1 || (times > 2 && input >= 0 && input < 10 )){
            // no input or number bigger than 999
            continue;
        }else if (times <= 2 && input >= 0 && input < 10 ){
            // acceptable number
            if(negative == 1){
                *num = *num * 10 - input;
            }else {
                *num = *num * 10 + input;
            }
            times++;
        }else if(input == 14){
            // clear
            *num = 0;
            times = 0;
        }else if(input == 11 && times == 0){
            negative = 1;
        }else{
            // operator
            *operator = input;
            return;
        }
    }
}
```

```c
void calculate(int *num1, int *num2, int *operator){
    int input = 0;
    int result ;
    switch(*operator){
        case 10:
            result = *num1 + *num2; break;
        case 11:
            result = *num1 - *num2; break;
        case 12:
            result = *num1 * *num2; break;
        case 13:
            result = *num1 / *num2; break;
    }
    while(input != 14 && result != 0){
        input = get_keyboard_input();
        display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin, result, num_digits(result));
    }
    return;
}
```



```
void part1(){
    int num_1 = 0, num_2 = 0;
    int operator_1 = 0, operator_2 = 0;
    int status = 1;
    while(1){
        if (status == 1){
            input_num(&num_1, &operator_1);
            status = 2;
        }else if(status == 2){
            while(operator_2 != 15){
                // operator != '='
                input_num(&num_2, &operator_2);
            }
            status = 3;
        }else if(status == 3){
            calculate(&num_1, &num_2, &operator_1);
            status = 1;
            operator_1 = 0;
            operator_2 = 0;
        }
    }
```

- **PART 2. (20%)** 問答題
  - 說明 **open-drain** 和 **push-pull** 在本實驗應用上的差異。

    A: 如果使用 **Open-drain** 模式控制 **Keypad** 和七段顯示器，需要外接上拉電阻，這樣可以使 **GPIO** 針腳上的電位在沒有輸出時被拉升至高電位，防止電位漂移和干擾。如果使用 **Push-pull** 模式控制 **Keypad** 和七段顯示器，則不需要外接上拉電阻。

  - 本實驗 **KeyPad** 的運作原理。

    A: 4x4 的 **KeyPad** 透過 **4** 個 **input** 端與 **4** 個 **output** 端連接到 **STM32** 的 **GPIO** 針腳上來進行行列掃描，當一個按鈕按下時，會有一組 **input** 與 **output** 形成通路，再根據按鈕位置映射出對應的按鈕值。

- **Part 3**

```c
int input_checkError(int *num, int *operator){
    int input = 0;
    int times = 0;
    while(1){
        display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin, *num, num_digits(*num));
        input = get_keyboard_input();
        if (input == -1 || (times > 2 && input >= 0 && input < 10 )){
            // no input or number bigger than 999
            continue;
        }else if (input >= 0 && input < 10 ){
            // acceptable number
            *num = *num * 10 + input;
            times++;
        }else if(input == 14){
            // clear
            *num = 0;
            times = 0;
        }else if(times ==0 && (input >= 10 && input <= 13)){
            // if enter a operator which is not neg at first time
            return -2;
        }else if(input == 15){
            *operator = input;
            return -1;
        }
        else{
            // operator
            *operator = input;
            return 0;
        }
    }
}
```

```c
int BODMAS(int list[]){
    int length = 0;
    while(list[length] !=15)
        length++;
    int tmp[20] = {0};

    int tmp_i = 0;
    for(int i = 0; i<=length; i++){
        tmp[tmp_i] = list[i];
        if(list[i+1] == 12){
            tmp[tmp_i] *= list[i+2];
            i+=2;
        }else if(list[i+1] == 13){
            tmp[tmp_i] /= list[i+2];
            i+=2;
        }
        tmp_i++;
    }

    int num = tmp[0];
    length = 0;
    while(tmp[length] !=15)
        length++;

    for(int i = 2; i<=length; i+=2){
        if(tmp[i-1] == 10)
            num += tmp[i];
        else if (tmp[i-1] == 11)
            num-= tmp[i];
    }
    return num;
}

```

```c
void part3(){
    while(1){
        int result = 0;
        int state = 0, error = 0;
        // state == -1 : end
        // state == -2 : error

        int list[20] = {0};
        int times = 0;

        // get a num and an operator at a time
        while(state != -1){
            state = input_checkError( &list[times], &list[times+1]);
            times+=2;
            if(state == -2) error = 1;
        }

        if(error == 1)
            result = -1;
        else if(state == -1 && error == 0)
            result = BODMAS(list);

        int input = 0;
        while(input != 14 && result != 0){
            input = get_keyboard_input();
            display_number(SEG_gpio, DIN_pin, CS_pin, CLK_pin, result, num_digits(result));
        }
    }
}
```

- **Part 4**

```c
void input_multi(){
    int input = 0, num = 0;
    int first = 1;
    while(1){
        if(first == 1)
            // show nothing if the first time
            display(0, 0);
        else
            display(num, num_digits(num));

        for(int i=0;i<16;i++){
            int a = keypad_space[i][0];
            int b = keypad_space[i][1];
            if(check_press(a,b)){
                // if one been pressed
                input = keypad[a][b];

                if(input == 14)// clear
                    return;
                if(num_digits(num)>=8) break;
                if(first == 1) first--;

                int keyboard_state[16] = {0};
                int press = 1;
                while(press == 1){
                    press = 0;
                    for(int j=0;j<16;j++){
                        int c = keypad_space[j][0];
                        int d = keypad_space[j][1];
                        if(check_press(c,d)){
                            if(keypad[c][d]==14) return;
                            keyboard_state[keypad[c][d]] = 1;
                            press = 1;
                        }
                    }
                }
                int tmp = 0;
                for(int j = 0; j<16; j++){
                    if(keyboard_state[j] == 1)
                        tmp+=j;
                }

                for(int j = 0; j<num_digits(tmp); j++){
                    num *=10;
                }
                num+=tmp;
                break;
            }
        }
    }
}
void part4(){
    while(1)
        input_multi();
}
```