**/\* 소스코드 \*/**

#include *<stdio.h>*

#include *<stdlib.h>*

#include *<string.h>*

#include *<wiringPi.h>*

#define LCD\_D4 2

#define LCD\_D5 3

#define LCD\_D6 1

#define LCD\_D7 4

#define LCD\_RS 7

#define LCD\_EN 0

#define KEY\_0 23

#define KEY\_1 13

#define KEY\_2 21

#define KEY\_3 22

#define KEY\_4 27

#define KEY\_5 12

#define KEY\_6 14

#define KEY\_7 10

#define KEY\_8 11

#define KEY\_9 26

#define KEY\_EQ 24

#define KEY\_P 5

#define KEY\_M 6

void init();

void putdatacmd4(unsigned char cmd);

void write4bits(unsigned char cmd);

void putcmd(unsigned char cmd);

void putdata(char c);

char readkey();

void print\_error(int err);

void clear\_display();

int main()

{

char current\_key = 0;

char prev\_key = 0;

char buff\_exp[500] = {};

int exp\_idx;

int prev\_ans = 0;

int display\_idx;

init();

**while**(1)

{

int sign = 1;

int buff\_num[100] = {};

int num\_idx = 0;

int num = 0;

char buff\_sum[10];

int sum = 0;

int i = 0;

int err = 0;

exp\_idx = 0;

display\_idx = 0;

**while**(1)

{

current\_key = readkey();

**if**(current\_key == 0)

prev\_key = 0;

**else** **if**(current\_key != prev\_key)

{

**if**(exp\_idx == 0)

clear\_display();

**if**(display\_idx == 16)

putcmd(0xC0);

**else** **if**(display\_idx == 32)

{

display\_idx = 0;

clear\_display();

}

putdata(current\_key);

buff\_exp[exp\_idx] = current\_key;

exp\_idx++;

display\_idx++;

prev\_key = current\_key;

**if**(current\_key == '=')

**break**;

}

delay(50);

}

*/\* expression input terminate ------------------------------------ \*/*

**while**(1)

{

char token = buff\_exp[i];

**if**(token == '+' || token == '-')

{

**if**(i != 0 && num != 0)

{

buff\_num[num\_idx] = num \* sign;

num\_idx++;

num = 0;

sign = 1;

}

**if**(token == '-')

sign = sign \* -1;

}

**else** **if**(token == '=')

{

**if**(buff\_exp[i-1] == '-' || buff\_exp[i-1] == '+')

err = 100;

**else**

{

buff\_num[num\_idx] = num \* sign;

num\_idx++;

}

**break**;

}

**else**

{

num = num\*10 + token-48;

**if**((num & 0x80000000) == 0x80000000)

{

err = 200;

**break**;

}

}

i++;

}

*/\* expression convert and calcuation terminate ------------------ \*/*

**if**(buff\_exp[0] == '-' || buff\_exp[0] == '+')

sum = sum + prev\_ans;

**for**(i=0; i<num\_idx; i++)

{

char is\_sign\_1 = (sum&0x80000000)==(buff\_num[i]&0x80000000);

char is\_sign\_2;

sum = sum + buff\_num[i];

is\_sign\_2 = (sum&0x80000000)!=(buff\_num[i]&0x80000000);

**if**(is\_sign\_1 && is\_sign\_2)

{

err = 200;

**break**;

}

}

**if**(err == 0)

{

prev\_ans = sum;

**for**(i=0; i<exp\_idx; i++)

printf("%c", buff\_exp[i]);

printf("%d**\n**", sum);

sprintf(buff\_sum, "%d", sum);

**if**(strlen(buff\_sum) + display\_idx > 33)

{

clear\_display();

putdata('=');

display\_idx++;

}

**for**(i=0; i<strlen(buff\_sum); i++)

{

**if**(display\_idx == 16)

putcmd(0xC0);

putdata(buff\_sum[i]);

display\_idx++;

}

}

**else**

print\_error(err);

*/\* expression output terminate ---------------------------------- \*/*

}

**return** 0;

}

void init()

{

wiringPiSetup();

pinMode(KEY\_0, INPUT);

pinMode(KEY\_1, INPUT);

pinMode(KEY\_2, INPUT);

pinMode(KEY\_3, INPUT);

pinMode(KEY\_4, INPUT);

pinMode(KEY\_5, INPUT);

pinMode(KEY\_6, INPUT);

pinMode(KEY\_7, INPUT);

pinMode(KEY\_8, INPUT);

pinMode(KEY\_9, INPUT);

pinMode(KEY\_EQ, INPUT);

pinMode(KEY\_P, INPUT);

pinMode(KEY\_M, INPUT);

pinMode(LCD\_RS, OUTPUT);

pinMode(LCD\_EN, OUTPUT);

pinMode(LCD\_D4, OUTPUT);

pinMode(LCD\_D5, OUTPUT);

pinMode(LCD\_D6, OUTPUT);

pinMode(LCD\_D7, OUTPUT);

pullUpDnControl(KEY\_0, PUD\_UP);

pullUpDnControl(KEY\_1, PUD\_UP);

pullUpDnControl(KEY\_2, PUD\_UP);

pullUpDnControl(KEY\_3, PUD\_UP);

pullUpDnControl(KEY\_4, PUD\_UP);

pullUpDnControl(KEY\_5, PUD\_UP);

pullUpDnControl(KEY\_6, PUD\_UP);

pullUpDnControl(KEY\_7, PUD\_UP);

pullUpDnControl(KEY\_8, PUD\_UP);

pullUpDnControl(KEY\_9, PUD\_UP);

pullUpDnControl(KEY\_EQ, PUD\_UP);

pullUpDnControl(KEY\_P, PUD\_UP);

pullUpDnControl(KEY\_M, PUD\_UP);

digitalWrite(LCD\_RS, 0);

digitalWrite(LCD\_EN, 0);

digitalWrite(LCD\_D4, 0);

digitalWrite(LCD\_D5, 0);

digitalWrite(LCD\_D6, 0);

digitalWrite(LCD\_D7, 0);

delay(50);

*// Function Set*

putcmd(0x28); *// 0010 1000*

putcmd(0x28);

putcmd(0x28);

delayMicroseconds(50);

*// Display On/OFf Shift*

putcmd(0x0e); *// 0000 1110*

delayMicroseconds(50);

*// Entry Mode Set*

putcmd(0x06); *// 0000 0110*

delayMicroseconds(50);

*// Curser Home*

putcmd(0x02); *// 0000 0010*

delay(2);

*// Display Clear*

putcmd(0x01); *// 0000 0001*

delay(2);

}

void write4bits(unsigned char cmd)

{

digitalWrite(LCD\_D4, cmd&1);

cmd = cmd >> 1;

digitalWrite(LCD\_D5, cmd&1);

cmd = cmd >> 1;

digitalWrite(LCD\_D6, cmd&1);

cmd = cmd >> 1;

digitalWrite(LCD\_D7, cmd&1);

digitalWrite(LCD\_EN, 1);

delayMicroseconds(10);

digitalWrite(LCD\_EN, 0);

delayMicroseconds(10);

}

void putdatacmd4(unsigned char cmd)

{

write4bits((cmd>>4) & 0x0f);

write4bits(cmd & 0x0f);

delayMicroseconds(100);

}

void putcmd(unsigned char cmd)

{

digitalWrite(LCD\_RS, 0);

putdatacmd4(cmd);

}

void putdata(char c)

{

digitalWrite(LCD\_RS, 1);

putdatacmd4(c);

}

char readkey()

{

char key[13] = {};

char current\_key = 0;

char key\_code[13] = {

'0', '1', '2', '3', '4',

'5', '6', '7', '8' ,'9',

'=', '+', '-'};

**while**(1)

{

int i;

int count = 0;

key[0] = digitalRead(KEY\_0);

key[1] = digitalRead(KEY\_1);

key[2] = digitalRead(KEY\_2);

key[3] = digitalRead(KEY\_3);

key[4] = digitalRead(KEY\_4);

key[5] = digitalRead(KEY\_5);

key[6] = digitalRead(KEY\_6);

key[7] = digitalRead(KEY\_7);

key[8] = digitalRead(KEY\_8);

key[9] = digitalRead(KEY\_9);

key[10] = digitalRead(KEY\_EQ);

key[11] = digitalRead(KEY\_P);

key[12] = digitalRead(KEY\_M);

**for**(i=0; i<13; i++)

{

**if**(key[i] == 0x00)

{

current\_key = key\_code[i];

count++;

}

}

**if**(count == 0)

**return** 0;

**else** **if**(count == 1)

**return** current\_key;

delay(50);

}

}

void print\_error(int err)

{

int i;

char\* msg;

clear\_display();

**if**(err == 100)

msg = "syntax error";

**else** **if**(err == 200)

msg = "overflow occur";

**for**(i=0; i<strlen(msg); i++)

putdata(msg[i]);

}

void clear\_display()

{

putcmd(0x02);

delay(2);

putcmd(0x01);

delay(2);

}

**/\* 소스코드 구조 \*/**

int main()

{

**while**(1)

{

**while**(1)

{

**/\* 수식을 입력받고 CLCD에 띄움 \*/**

}

*/\* expression input terminate ---------------------- \*/*

**while**(1)

{

**/\***

**수식을 분석함**

**연산자와 피연산자를 추출하거나 오류를 검사함**

**\*/**

}

*/\* expression convert and calcuation terminate ----- \*/*

**if**(buff\_exp[0] == '-' || buff\_exp[0] == '+')

**/\* 첫 입력이 부호이면 이전 수식의 결과를 더함 \*/**

**for**(i=0; i<num\_idx; i++)

{

**/\***

**추출해낸 정보를 바탕으로 결과값을 계산한다.**

**계산 중에 오버플로우가 발생하면 계산을 중단하고 에러코드를 저장**

**\*/**

}

**if**(err == 0)

{

**/\***

**에러가 발생하지 않았을 경우(err == 0)**

**추출해낸 정보를 바탕으로 결과값을 계산하고**

**CLCD에 출력함**

**\*/**

}

**else**

**// 오류가 발생하였을 경우(err != 0) 오류 안내문 출력**

*/\* expression output terminate --------------------- \*/*

}

**return** 0;

}

void init() **// wiringPi, CLCD의 핀과 CLCD 화면의 초기화**

char readkey() **// 버튼을 입력받아서 반환해줌**

void print\_error(int err) **// 전달받은 오류를 출력**

void clear\_display() **// 화면을 지움**

**/\***

**예제로부터 참조한 함수들**

**\*/**

void write4bits(unsigned char cmd) **// 4개 Data레지스터에 비트연산**

void putdatacmd4(unsigned char cmd) **// 8비트의 데이터를 4비트로 나누어 2번 전송**

void putcmd(unsigned char cmd) **// CLCD에 제어 명령어 전송 (RS == 0)**

void putdata(char c) **// CLCD에 데이터 전송 (RS == 1)**

**/\* 프로그램 초기화 \*/**

void init()

{

wiringPiSetup();

pinMode( **KEY\_0 ~ KEY\_9, KEY\_EQ, KEY\_P, KEY\_M** , INPUT);

pinMode( **LCD\_RS, LCD\_EN, LCD\_D4 ~ LCD\_D7**, OUTPUT);

pullUpDnControl( **KEY\_0 ~ KEY\_9, KEY\_EQ, KEY\_P, KEY\_M** , PUD\_UP);

digitalWrite( **LCD\_RS, LCD\_EN, LCD\_D4 ~ LCD\_D7** , 0);

delay(50);

*// Function Set*

putcmd(0x28); *// 0010 1000*

putcmd(0x28);

putcmd(0x28);

delayMicroseconds(50);

*// Display On/OFF Shift*

putcmd(0x0e); *// 0000 1110*

delayMicroseconds(50);

*// Entry Mode Set*

putcmd(0x06); *// 0000 0110*

delayMicroseconds(50);

*// Return Home*

putcmd(0x02); *// 0000 0010*

delay(2);

*// Display Clear*

putcmd(0x01); *// 0000 0001*

delay(2);

}

**/\***

**버튼으로부터 입력받는 핀을 INPUT, CLCD로 출력하는 핀을 OUTPUT, PULL-UP으로 설정합니다.**

**Function Set**

**데이터의 길이를 4비트, 화면 표시 행수를 2행, 문자 폰트를 5x7로 설정합니다.**

**Display On/Off Shift**

**화면 표시를 On, 커서 표시를 On, 커서를 깜빡이지 않게 설정합니다.**

**Entry Mode Set**

**데이터를 읽기/쓰기 할 때 커서를 증가 시키고 화면을 shift하지 않게 설정합니다.**

**Return Home**

**커서를 home위치로 옮깁니다.**

**Display Clear**

**전체 화면을 지웁니다.**

**\*/**

**/\* 버튼에서 데이터를 읽어오는 함수 \*/**

char readkey()

{

char key[13] = {};

char current\_key = 0;

char key\_code[13] = {

'0', '1', '2', '3', '4',

'5', '6', '7', '8' ,'9',

'=', '+', '-'};

**while**(1)

{

int i;

int count = 0;

key[ **0 ~ 12** ] = digitalRead( **KEY\_0 ~ KEY\_9, KEY\_EQ, KEY\_P, KEY\_M** );

**for**(i=0; i<13; i++)

{

**if**(key[i] == 0x00)

{

current\_key = key\_code[i];

count++;

}

}

**if**(count == 0)

**return** 0;

**else** **if**(count == 1)

**return** current\_key;

delay(50);

}

}

**/\***

**입력 핀에서 버튼 입력 여부를 50ms마다 한번씩 읽어옵니다.**

**동시에 버튼 두개가 입력됐을 경우 입력을 무효화 하기 위해서 입력된 버튼 수를 셉니다.**

**버튼이 하나도 입력되어 있지 않은 상태이면 0을 반환합니다.**

**버튼이 하나만 입력되면 버튼에 해당하는 숫자나 기호의 ASCII를 반환합니다.**

**버튼이 두 개 이상 입력되면 입력을 무시하고 루프를 계속합니다.**

**\*/**

**/\* 에러가 발생했을시 에러를 출력하는 함수 \*/**

void print\_error(int err)

{

int i;

char\* msg;

clear\_display();

**if**(err == 100)

msg = "syntax error";

**else** **if**(err == 200)

msg = "overflow occurred";

**for**(i=0; i<strlen(msg); i++)

putdata(msg[i]);

}

**/\***

**에러코드를 받으면 그에 해당하는 문구를 CLCD화면으로 출력하는 함수입니다.**

**err == 100 이면 수식의 마지막이 숫자로 끝나지않은 경우로 “syntax error”를 출력합니다.**

**example) 100+200-=**

**err == 200 이면 피연산자의 크기나 계산결과의 크기가 이나 을 넘은 경우로**

**“overflow occurred”를 출력합니다.**

**example) 2000000000+2000000000=**

**\*/**

**/\* CLCD화면을 지우는 함수 \*/**

void clear\_display()

{

putcmd(0x02);

delay(2);

putcmd(0x01);

delay(2);

}

**/\***

**CLCD의 화면 전체를 지우고 커서를 Home위치로 옮깁니다.**

**\*/**

**/\* 예제로부터 참조한 함수들 \*/**

void write4bits(unsigned char cmd)

{

digitalWrite(LCD\_D4, cmd&1);

cmd = cmd >> 1;

digitalWrite(LCD\_D5, cmd&1);

cmd = cmd >> 1;

digitalWrite(LCD\_D6, cmd&1);

cmd = cmd >> 1;

digitalWrite(LCD\_D7, cmd&1);

digitalWrite(LCD\_EN, 1);

delayMicroseconds(10);

digitalWrite(LCD\_EN, 0);

delayMicroseconds(10);

}

**/\***

**넘겨받은 인자의 첫번째 비트를 추출하여 각 data pin에 씁니다.**

**\*/**

void putdatacmd4(unsigned char cmd)

{

write4bits((cmd>>4) & 0x0f);

write4bits(cmd & 0x0f);

delayMicroseconds(100);

}

**/\***

**CLCD의 데이터의 길이가 4비트모드이므로 8비트로 받은 데이터를 4비트 2개로 쪼개서 CLCD에 씁니다.**

**\*/**

void putcmd(unsigned char cmd)

{

digitalWrite(LCD\_RS, 0);

putdatacmd4(cmd);

}

**/\***

**제어 명령을 쓰기위한 함수이므로 LCD\_RS를 0으로 설정하고 명령어를 전달합니다.**

**\*/**

void putdata(char c)

{

digitalWrite(LCD\_RS, 1);

putdatacmd4(c);

}

**/\***

**데이터를 쓰기위한 함수이므로 LCD\_RS를 1로 설정하고 데이터를 전달합니다.**

**\*/**

**/\* 프로그램 본문 \*/**

int main()

{

char current\_key = 0; **// 현재 입력**

char prev\_key = 0; **// 이전 입력**

char buff\_exp[500] = {}; **// 수식을 문자열 형태로 저장하는 버퍼**

int exp\_idx; **// buff\_exp의 index**

int prev\_ans = 0; **// 이전 수식의 결과 값**

int display\_idx; **// CLCD 화면의 index**

init(); **// 프로그램 초기화**

**while**(1)

{

int sign = 1; **// 현재 입력받고 있는 피연산자의 부호**

int buff\_num[100] = {}; **// buff\_exp의 피연산자를 정수로 변환한 결과**

int num\_idx = 0; **// buff\_num의 index**

int num = 0; **// 문자열 피연산자를 정수로 변환하기 위한 임시변수**

char buff\_sum[10]; **// 정수형태의 계산결과를 문자열로 변환하기 위한 버퍼**

int sum = 0; **// 수식의 계산결과**

int i = 0;

int err = 0; **// 에러를 전달하기 위한 변수**

exp\_idx = 0; **// 수식을 다시 입력할 때마다 초기화**

display\_idx = 0;

**/\***

**수식 입력, 분석, 계산, 결과출력 . . .**

**\*/**

}

**return** 0;

}

**/\***

**가장 바깥쪽 while문은 한 바퀴 돌때마다 수식 하나를 입력받고 결과를 계산합니다.**

**\*/**

**/\***

**첫 번째 while**

**버튼으로부터 값을 받아서 내부 버퍼에 저장**

**\*/**

**while**(1)

{

**while**(1)

{

current\_key = readkey(); **// 값을 받아옴**

**if**(current\_key == 0) **// 값이 0일경우(입력이 없을경우)**

prev\_key = 0; **// 이전 값도 입력 없음으로 초기화**

**else** **if**(current\_key != prev\_key)**// 현재 값과 이전 값이 다를경우**

{ **// ( = 중복입력이 아닐경우)**

**if**(exp\_idx == 0) **// 새로운 수식일경우**

clear\_display(); **// 화면을 전부 지움**

**if**(display\_idx == 16) **// 한 행을 전부 채웠을 경우**

putcmd(0xC0); **// 다음 행으로 커서를 옮김**

**else** **if**(display\_idx == 32) **// 화면을 전부 채웠을 경우**

{

display\_idx = 0; **// 화면을 지우고 커서를**

clear\_display(); **// Home으로 옮김**

}

putdata(current\_key); **// 값을 CLCD에 씀**

buff\_exp[exp\_idx] = current\_key;**// buff\_exp에 순차적으로 저장**

exp\_idx++;

display\_idx++;

prev\_key = current\_key; **// 현재 값을 이전 값 변수에 저장**

**if**(current\_key == '=') **// 값이 ‘=’면 수식입력 종료**

**break**;

}

delay(50);

}

**if**(buff\_exp[0] == '-' || buff\_exp[0] == '+'){ **/\* . . . \*/** }

**for**(i=0; i<num\_idx; i++){ **/\* . . . \*/** }

**while**(1){ **/\* . . . \*/** }

**if**(err == 0) { **/\* . . . \*/** }

**else** { **/\* . . . \*/** }

}

**/\***

1. readkey함수로 입력을 받아옵니다.

2. 이전 값과 비교해서 같으면 입력을 무시하도록 합니다. (중복입력을 방지하기 위함)

3. 유효한 값을 받았을 경우 CLCD화면의 상황을 체크하여 화면지우기, 줄 바꾸기 등을 수행하여

입력 값을 CLCD에 출력합니다.

4. 이후 입력 값을 수식 버퍼에 저장하고 현재 값을 이전 값 변수에 저장합니다.

5. 이어서 다음 값을 계속 받아옵니다.

6. 반복문은 ‘=’ 기호를 입력받을 때까지 반복합니다.

\*/

**/\***

**두 번째 while**

**수식을 분석하여 계산할 수 있는 형태로 변환**

**\*/**

**while**(1)

{

**while**(1){ **/\* . . . \*/** }

**while**(1)

{

char token = buff\_exp[i]; **// 수식의 문자를 하나씩 꺼냄**

**if**(token == '+' || token == '-') **// token이 부호이면**

{

**if**(i != 0 && num != 0) **// token이 수식의 첫번째 값이 아니면**

{

buff\_num[num\_idx] = num \* sign;

num\_idx++; **// 이전에 분석하던 피연산자를**

num = 0; **// buff\_num에 저장하고**

sign = 1; **// 새로운 피연산자를 받을 준비**

}

**if**(token == '-') **// 부호가 ‘-’이면**

sign = sign \* -1; **// 부호를 반전시킴**

}

**else** **if**(token == '=') **// token이 ‘=’이면**

{ **// ‘=’을 제외한 수식의 마지막 값이 부호이면 에러**

**if**(buff\_exp[i-1] == '-' || buff\_exp[i-1] == '+')

err = 100;

**else // 숫자이면 유효한 수식**

{

buff\_num[num\_idx] = num \* sign;

num\_idx++; **// 마지막 피연산자를 buff\_num에 저장**

}

**break**; **// 수식분석 종료**

}

**else // token이 숫자이면**

{

num = num\*10 + token-48; **// token을 정수로 변환하고**

**if**((num & 0x80000000) == 0x80000000) **// 오버플로우 검사**

{

err = 200;

**break**;

}

}

i++;

}

**if**(buff\_exp[0] == '-' || buff\_exp[0] == '+'){ **/\* . . . \*/** }

**for**(i=0; i<num\_idx; i++){ **/\* . . . \*/** }

**if**(err == 0) { **/\* . . . \*/** }

**else** { **/\* . . . \*/** }

}

**/\***

**1. 수식 버퍼로부터 한 token씩 꺼냅니다.**

**2. token이 숫자이면 적절하게 변환하고 오버플로우 여부를 검사한 뒤 변수 num에 저장합니다.**

**이 때 overflow가 발생하면 에러코드를 저장하고 계산을 종료합니다.**

**3. token이 ‘+’, ‘-’이면 이에 맞춰 변수 sign의 부호를 맞춥니다.**

**부호가 나왔다는 것은 피연산자 하나가 완전히 꺼내어졌다는 것이므로 변수 num을**

**buff\_num에 저장합니다. (buff\_num은 이후 결과를 계산하는 데에 사용됩니다.)**

**4. token이 ‘=’이면 수식이 유효한가를 검사하고 문제가 없으면 마지막 피연산자를 buff\_num에**

**저장하고 반복문을 빠져나옵니다.**

**example) 100+200--= (유효하지 못한 수식)**

**example) 100+200--0= (유효한 수식)**

**\*/**

**/\* 계산 및 결과 출력 \*/**

**while**(1)

{

**while**(1){ **/\* . . . \*/** }

**while**(1){ **/\* . . . \*/** }

**if**(buff\_exp[0] == '-' || buff\_exp[0] == '+') **// 수식이 부호로 시작하면**

sum = sum + prev\_ans; **// 이전 결과에 이어서 계산함**

**for**(i=0; i<num\_idx; i++) **// 이전 단계에서 분석한 수식으로**

{ **// 최종 결과를 계산함**

char is\_sign\_1 = (sum&0x80000000)==(buff\_num[i]&0x80000000);

char is\_sign\_2; **// 계산과 동시에 오버플로우 검사**

sum = sum + buff\_num[i];

is\_sign\_2 = (sum&0x80000000)!=(buff\_num[i]&0x80000000);

**if**(is\_sign\_1 && is\_sign\_2)

{

err = 200;

**break**;

}

}

**if**(err == 0) **// 계산 결과에 오류가 없으면**

{

prev\_ans = sum; **// 다음 수식에서 사용하기 위해 결과 저장**

**for**(i=0; i<exp\_idx; i++) **// 수식과 계산결과를 모니터에 출력**

printf("%c", buff\_exp[i]);

printf("%d**\n**", sum);

sprintf(buff\_sum, "%d", sum); **// 정수형태의 계산결과를 문자열로 변환**

**if**(strlen(buff\_sum) + display\_idx > 33)

{ **// 수식의 길이 + 계산결과의 길이가**

clear\_display(); **// CLCD화면을 꽉 채우면**

**// 화면을 지움**

putdata('=');

display\_idx++;

}

**for**(i=0; i<strlen(buff\_sum); i++) **// CLCD에 계산결과 출력**

{

**if**(display\_idx == 16) **// 첫번째 행이 꽉 차면**

putcmd(0xC0); **// 다음 행으로 커서 이동**

putdata(buff\_sum[i]);

display\_idx++;

}

}

**else**

print\_error(err); **// 에러가 발생했을 시**

} **// 출려과정을 거치지 않고 에러 출력**

**/\***

**1. 수식의 첫번째 값이 부호이면 이전 수식의 값을 sum에 더하고 아니면 더하지 않습니다.**

**2. 이전 단계에서 buff\_num에 정리해 놓은 피연산자들을 모두 sum에 더합니다.**

**이 때 sum이 오버플로우가 발생할 수 있으므로 더할 때마다 검사합니다.**

**3. 문제없이 결과가 계산되었으면 err == 0일 것입니다.**

**err != 0이면 CLCD 출력과정을 거치지 않고 에러를 출력합니다.**

**4. 새로운 계산결과를 다음 수식에서 사용하기 위해 prev\_ans에 저장합니다.**

**5. 이후 정수형태의 계산결과를 문자열로 변환하고, CLCD화면의 상황을 체크하여 화면지우기,**

**줄바꾸기 등 적절한 조치를 취한 후 결과를 CLCD화면으로 출력합니다.**

**\*/**