**프로그램 구조**

int main()

{

**while**(1)

{

*/\* 키를 하나 입력받음 \*/*

char key = readkey();

**if**(IMG\_index != -1)

{

**if**(IMG\_index == 111)

*/\** *LEFT 버튼이 입력됐을 때의 처리 \*/*

**else** **if**(IMG\_index == 112)

*/\* DELETE 버튼이 입력됐을 때의 처리 \*/*

**else** **if**(IMG\_index == 113)

*/\* RIGHT 버튼이 입력됐을 때의 처리 \*/*

**else**

*/\* a ~ z, 점, 공백이 입력됐을 때의 처리 \*/*

}

*/\* 현재 위치에 커서 출력 \*/*

}

}

*/\* 원하는 RGB색을 만드는 함수 \*/*

unsigned short makepixel(ubyte r, ubyte g, ubyte b)

*/\* 프로그램 초기화 함수 \*/*

datablock init()

*/\* 13개의 버튼에 대한 입력을 받는 함수 \*/*

char readkey()

*/\* readkey() 함수에서의 입력을 적절한 정수로 변환 \*/*

int keytoindex(char key)

*/\* (x, y) 위치에 전달받은 글자를 출력하는 함수 \*/*

void printfont(int x, int y, **const** unsigned char\* img)

*/\* buffer와 함수를 동기화하는 함수 \*/*

void controlscreen(int cmd, int cursor\_pos, int buffer\_size, **const** unsigned char\*\* buffer)

*/\* (x, y) 위치에 커서를 출력하는 함수 \*/*

void printcursor(int x, int y, **const** unsigned char\* img)

*/\* cursor\_pos가 증가됨에 따라 화면 커서의 x, y 좌표도 조정하는 함수 \*/*

void increaseposition(int\* x, int\* y)

*/\* (x, y) 위치에 buffer의 start에서 end-1까지의 데이터를 출력 \*/*

void updatescreen(int start, int end, int x, int y, **const** unsigned char\*\* buffer)

**전역변수 및 구조체**

**typedef** **struct** datablock

{

int fbfd;

**struct** fb\_var\_screeninfo fbvar;

unsigned short\* pfbdata;

int prev\_key;

int last\_key;

int x, y;

}datablock;

**const** unsigned char\* IMG[30] = {

IMG\_point, IMG\_q, IMG\_z,

IMG\_a, IMG\_b, IMG\_c,

IMG\_d, IMG\_e, IMG\_f,

IMG\_g, IMG\_h, IMG\_i,

IMG\_j, IMG\_k, IMG\_l,

IMG\_m, IMG\_n, IMG\_o,

IMG\_p, IMG\_r, IMG\_s,

IMG\_t, IMG\_u, IMG\_v,

IMG\_w, IMG\_x, IMG\_y,

IMG\_empty, IMG\_empty, IMG\_empty

};

datablock db;

**/\***

**datablock : 구현의 용이성을 위한 구조체**

**IMG : index 로 데이터에 접근하기 위한 기초데이터**

**\*/**

**main() 함수**

int IMG\_index;

int is\_repeat = 0;

**const** unsigned char\* buffer[1000] = {0};

int buffer\_size = 0;

int cursor\_pos = 0;

**/\***

**IMG\_index : IMG에 접근하기 위한 변수**

**is\_repeat : 반복입력을 감지하기 위한 변수**

**buffer : 입력된 데이터를 저장하는 버퍼**

**buffer\_size : 데이터의 크기**

**cursor\_pos : buffer에서 커서의 위치**

**\*/**

**while**(1)

{

**/\* 키를 하나 입력받음 \*/**

char key = readkey();

**/\*** **key가 -1일땐 입력받지 않은 것으로 간주하고 아무것도 실행하지 않습니다. \*/**

**/\* -1이 아닐땐 무언가 입력받은 것으로 간주하고 처리를 걔속합니다. \*/**

IMG\_index = keytoindex(key);

**if**(IMG\_index != -1)

{

**if**(IMG\_index == 111) */\* LEFT \*/*

{

**/\***

**1.** **LEFT 버튼이 입력됐을 때의 처리입니다.**

**2. LEFT 버튼이 입력되면 커서를 왼쪽으로 이동합니다.**

**3. 따라서 기존 커서 위치에 있던 글자를 원래 색으로 다시 표현해야 하므로 기존 위치에 printfont()함수를 이용하여 폰트를 재출력합니다.**

**4. 커서를 왼쪽으로 이동할 때 커서가 있던 기존 위치의 문자가 공백이고 buffer의 맨 끝이면 공백을 buffer에서 삭제합니다.**

**쓸데없는 공간낭비를 줄이기 위함입니다.**

**\*/**

}

**else** **if**(IMG\_index == 112) */\* DELETE \*/*

{

**/\***

**1. DELETE 버튼이 입력됐을 때 처리입니다.**

**2. DELETE 버튼은 현재 커서위치의 문자 하나를 제거합니다.**

**3. 문자를 제거하고 뒤에 이어지는 문자들을 앞으로 한 칸 당깁니다.**

**\*/**

}

**else** **if**(IMG\_index == 113) */\* RIGHT \*/*

{

**/\***

**1. RIGHT 버튼이 입력됐을 때의 처리입니다.**

**2. RIGHT 버튼이 입력되면 커서를 오른쪽으로 이동합니다.**

**3. 기존 커서 위치가 오른쪽 끝에 위치했으면 공백을 입력한 것으로 간주합니다.**

**\*/**

}

**else** */\* INPUT \*/*

{

**/\***

**1. a ~ z, 점, 공백이 입력됐을 때의 처리입니다.**

**2. 현재 커서위치에 입력된 문자를 넣습니다.**

**3. 한 버튼에 여러 글자가 할당되어 있으므로 반복해서 누르거나 다른 입력버튼을 누르면 글자가 바뀝니다.**

**4. is\_repeat 변수를 이용하여 반복입력인지 아닌지 상태를 결정합니다.**

**5. 문자 사이에서 버튼이 입력됐을 때는 글자를 삽입하도록 처리합니다. 따라서 커서 뒤쪽에 있는 문자들이 한 칸씩 밀립니다.**

**\*/**

}

}

**/\* 어떤 처리를 마치고 나서는 항상 cursor\_pos 위치에 커서를 출력합니다. \*/**

}

**init() 함수**

datablock init()

{

**/\***

**1. 프로그램을 초기화하는 함수입니다.**

**2. 파일열기, mmap, wiringPi등등에 관한 초기화를 처리합니다.**

**\*/**

}

**readkey() 함수**

char readkey()

{

**/\***

**1. 13개의 버튼의 입력을 받는 함수입니다.**

**- key[0 ~ 13] = digitalRead(KEY1 ~ KEY\_RIGHT);**

**2. 버튼 입력갯수에 따라 처리가 달라집니다.**

**- 아무것도 입력되지 않았을 땐 0을 반환합니다.**

**- 1개의 버튼만 입력됐을 땐 버튼의 번호를 반환합니다.**

**- 2개 이상의 버튼이 입력됐으면 아무것도 반환하지 않고 루프를 계속합니다.**

**\*/**

}

**keytoindex() 함수**

int keytoindex(char key)

{

**/\***

**1. readkey() 함수로 입력된 버튼의 번호를 정수로 변환합니다.**

**2. 입력 종류에 따라 처리가 달라집니다.**

**- a ~ z, 점, 공백이 입력됐을 IMG배열을 참조하기 위한 index를 반환합니다.**

**- i를 이용하여 버튼하나에 글자 3개를 입력할 수 있도록 합니다.**

**=> index = (key – 1)\*3 + i**

**- LEFT, RIGHT, DELETE 입력은 전달인자에 100을 더하여 반환합니다.**

**=> index와 구분하기 위함**

**3.**

**\*/**

}

**printfont() 함수**

void printfont(int x, int y, **const** unsigned char\* img)

{

**/\* (x, y) 위치에 전달받은 글자를 출력하는 함수입니다. \*/**

**/\* 비트 연산을 통해 비트단위로 픽셀을 입력합니다. \*/**

}

**printcursor() 함수**

void printcursor(int x, int y, **const** unsigned char\* img)

{

**/\* (x, y) 위치에 커서를 출력하는 함수입니다. \*/**

**/\* printfont() 함수와 거의 동일합니다. \*/**

}

**increaseposition() 함수**

void increaseposition(int\* x, int\* y)

{

**/\* 전달받은 (x, y) 좌표를 화면에 맞춰서 증가시킵니다 \*/**

}

**updatescreen() 함수**

void updatescreen(int start, int end, int x, int y, **const** unsigned char\*\* buffer)

{

**/\* (x, y) 좌표에 buffer의 start에서 end-1까지 출력해줍니다.**

}

**controlscreen() 함수**

**/\* controlscreen() 함수는 buffer를 바탕으로 화면과 데이터를 동기화합니다. \*/**

void controlscreen(int cmd, int cursor\_pos, int buffer\_size,

**const** unsigned char\*\* buffer)

{

**static** int scrolled = 0; **/\* 얼마나 스크롤 됐는지 나타내는 변수 \*/**

**if**(cmd == 111) */\* LEFT \*/*

{

**/\***

**1. 커서를 왼쪽으로 옮깁니다.**

**2. 현재 (x, y) 좌표에 따라 처리가 다릅니다.**

**- x가 0일 때**

**- y가 1이상일 때**

**=> 윗줄로 커서를 옮김**

**- y가 0일 때**

**=> 스크롤 되어있으면 위로 한 줄 스크롤 함**

**- x가 1이상일 때**

**=> 커서를 왼쪽으로 옮김**

**\*/**

}

**else** **if**(cmd == 112) */\* DELETE \*/*

{

**/\***

**1. 커서 위치의 문자를 삭제합니다.**

**2. 커서 뒷 문자들을 앞으로 한 칸 당깁니다.**

**=> updatescreen(start, end, db.x, db.y, buffer);**

**\*/**

}

**else** **if**(cmd == 113) */\* RIGHT \*/*

{

**/\***

**1. 커서를 오른쪽으로 옮깁니다.**

**2. 현재 (x, y) 좌표에 따라 처리가 다릅니다.**

**- x가 9일때**

**- y가 6이하일 때**

**=> 아랫줄로 커서를 옮김**

**- y가 7일 때**

**=> 아래로 한 줄 스크롤함**

**- x가 8이하일 때**

**=> 커서를 으론쪽으로 옮김**

**\*/**

}

**else** */\* INPUT \*/*

{

**/\***

**1. 커서 위치에 문자를 삽입합니다.**

**2. 커서 뒷 문자들을 뒤로 한 칸 밀어냅니다.**

**=> updatescreen(start, end, db.x, db.y, buffer);**

**\*/**

}

}