CSED101. Programming & Problem solving

Fall, 2016

Programming Assignment #3

(70 points)

류승호(seunghoryu@postech.ac.kr)

- *Due*: **2016.11.15 23:59**
- *Development Environment*: **GNU C Compiler (GCC) and Vi Editor (Editor is optional)**

- *제출물*
  - **C Code files (*.c and *.h, 프로그램에 필요한 모든 파일)**
    - 프로그램의 소스 코드를 이해하기 쉽도록 반드시 **주석**을 붙일 것.
  - **보고서 파일** (.doc(x) or .hwp) 예) assn3.doc(x) 또는 assn3.hwp
    - AssnReadMe.pdf 를 참조하여 작성할 것.
    - **리눅스 서버에 접속하는 것부터 시작해서 프로그램 컴파일 및 실행하는 과정까지를 화면 캡쳐하여 보고서에 포함시키고 간단히 설명 할 것!!**
    - **명예서약(Honor code)**: 표지에 다음의 내용을 포함한다. "나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다." **보고서 표지에 명예서약이 없는 경우는 과제를 제출하지 않은 것으로 처리한다.**
    - 소스코드와 보고서 파일을 LMS를 이용하여 제출한다.

- *주의사항*
  - **과제에 대한 질문은 수강생들과 공유하기 위해 LMS의 질의응답 게시판을 사용한다. 이외 이메일 등의 개별 연락은 하루에 한 번 일괄 처리한다.**
  - 각 문제에 해당하는 요구사항을 반드시 지킬 것.
  - 컴파일 & 실행이 안되면 무조건 0점 처리된다.
  - **하루 late시 20%가 감점되며, 3일 이상 지나면 받지 않는다. (0점 처리)**
  - 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 'POSTECH 전자컴퓨터공학부 부정행위 정의'를 따른다. (LMS의 과목 공지사항의 제목 [document about cheating]의 첨부파일인 disciplinary.pdf를 참조할 것.)
  - **과제 작성시 전역변수는 사용할 수 없으며, 사용자 정의 함수를 적절히 작성하도록 한다.**
  - **문제에서 파일이름 또는 함수이름을 제시한 경우 해당 이름을 사용 할 것. 그 외의 다른 이름을 사용하면 감점 또는 0점 처리된다.**

■ Problem: LED 전광판 프로그램

(목적)

이번 과제를 통하여 2차원 배열의 선언과 사용 및 함수에서 2차원 배열을 매개변수로 사용하는 방법을 익힌다.

(주의사항)

- 보고서는 **"assn3.doc"** or **"assn3.hwp"**로 저장 할 것 (보고서는 통합하여 작성)
- 과제의 요구사항을 지켜서 프로그램을 작성할 것
- 출력은 아래의 "실행예제"와 유사하게 작성 할 것
- 전역변수, goto문, 포인터 그리고 구조체 등은 사용하지 않는다.
- 프로그램 구현 시, main() 함수를 호출하여 사용하지 않는다.
- 적절히 사용자 정의 함수를 작성하여야 한다. (main() 함수 내에 모든 기능을 구현 시 감점)
- 과제에서 정의하는 함수와 파일은 반드시 작성한다. (미 작성시 감점)
- 사용자의 입력을 확인하고 올바르게 예외 처리를 해야 한다.
- 과제에서 제안하는 추가 기능을 구현하는 경우 추가 점수가 있다.

(설명)

이번 과제의 LED 전광판 프로그램은 아래 그림과 같은 전광판 앱 또는 상가의 LED 광고판과 유사한 기능을 한다. LED 전광판 프로그램은 사용자가 지정한 문자열과 출력방법에 맞춰 화면에 보여준다.

Figure.1 스마트폰 전광판 앱 예시

사용자가 프로그램을 종료하기 전까지 화면에 문자열이 계속적으로 출력되게 된다.

I. 기본 설정

LED 전광판 프로그램에서는 다음에서 정의하는 것과 같은 동작을 해야 한다.

(1) LED 전광판의 크기

LED 전광판의 크기는 가로 75 X 세로 20으로 사용자로 입력 받은 문자열을 출력한다. 2차원 배열을 이용하여 LED의 내용을 저장하고 화면에 출력하도록 한다.

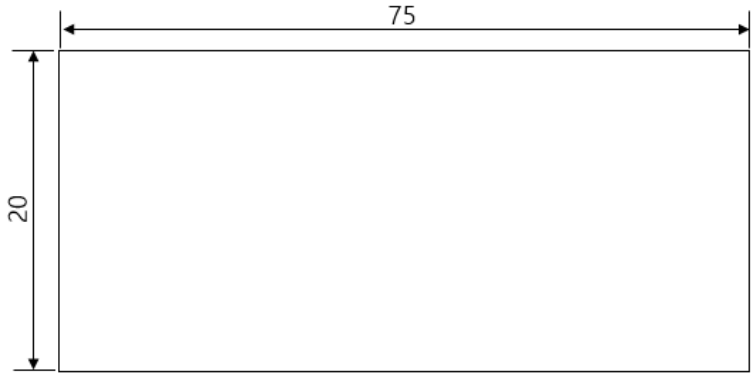


Figure.2 LED 전광판 크기

사용자가 입력한 문자열의 길이가 길어 LED 전광판에 전부 표현할 수 없으면 전광판에 출력 가능한 만큼만 출력된다. 예를 들어, 전광판에 한번에 출력 가능한 문자수가 6개이고 사용자 입력 문자열이 "123456789"이면 전광판에 출력되는 내용은 "123456"이고 전광판 이동패턴에 따라서 화면 출력내용이 변경된다.

(2) 전광판 내 문자 크기

LED 전광판에 출력되는 글자는 가로 7 X 세로 10의 크기를 가지며 연속된 글자의 구별을 위해 글자 앞뒤에 2칸의 여백을 준다. 공백 (spacebar)도 다른 글자와 동일하게 취급하여 여백을 포함하여 가로 11 X 세로 14의 크기를 가진다.

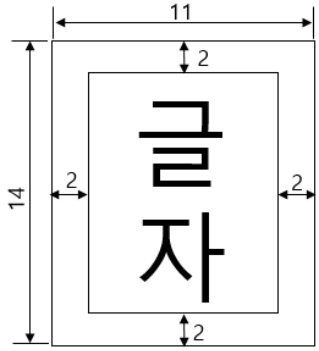


Figure.3 LED 전광판 내에서 글자 출력 크기

(3) 전광판 내 문자 위치

사용자로부터 입력 받은 문자열의 각 문자는 순서대로 전광판에 좌측부터 위치한다. 또한, 글자의 위아래에는 동일한 크기의 여백을 가진다 (세로축의 중간에 위치함을 의미).

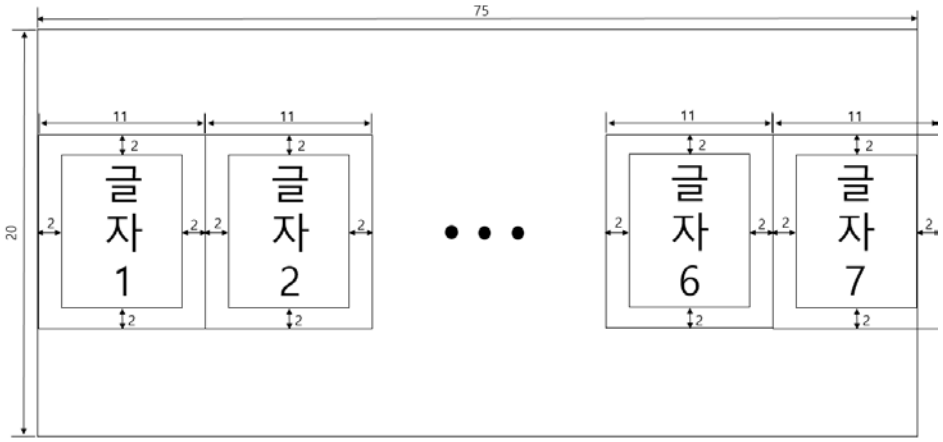


Figure.4 LED 전광판 내 글자 위치

II. 사용자 입력 화면

LED 전광판 프로그램이 실행이 되면 사용자로부터 순서대로 정보를 입력 받는다.

(1) 사용자 문자열

프로그램이 실행되면 화면을 지우고 Figure.5와 같이 사용자로부터 최대길이 20의 문자열을 입력 받을 수 있다. (화면을 지우는 방법은 문서 뒤 Tip을 참고)

Figure.5 LED 전광판에 출력할 문자열 입력 받기

사용자가 입력 가능한 글자는 **숫자 (0-9), 소문자 알파벳 (a-z), 그리고 공백 (spacebar)**로 이를 조합한 문자열을 LED 전광판에 출력하게 된다. **단, LED 전광판에 알파벳을 출력할 때는 대문자로 출력한다.**

사용자는 "abc123", "1234 5678", "i love programming", " 101" 등과 같은 문자열을 입력 할 수 있으며 "abc123"이 LED 전광판 화면에 출력될 때는 "ABC123"으로 대문자가 출력된다. (공백을 포함한 문자열을 입력 받는 방법은 문서 뒤의 Tip을 참고)

(참고) 문자열

C언어에서는 char 배열로 문자열을 표현하고 있다. 문자열과 단순 char형 배열을 구분하기 위해서 널(null)문자인 '₩0'을 문자열 마지막에 넣어준다. Figure.6은 문자열과 char 배열의 차이를 보여준다. 길이 n의 문자열을 표현하기 위해서는 n+1크기의 char 배열이 필요하다는 의미다.

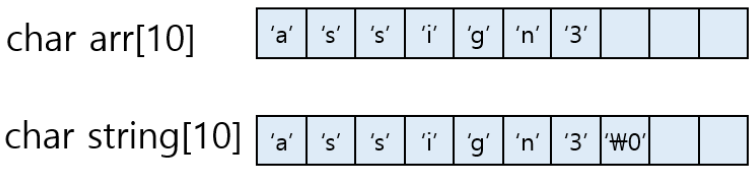


Figure 6 문자열과 단순 char 배열의 차이

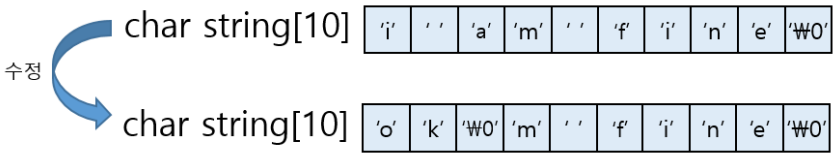


Figure.7 문자열 수정 예시

문자열을 출력할 때, 배열의 앞에서부터 출력하다가 널문자를 만나면 출력을 멈추게 된다. Figure.7는 초기에 "i am fine"이라는 문자열이 들어있는 배열에 문자열 "ok"을 쓰게 되면 Figure.7의 아래와 같이 된다. 이때 출력을 하게 되면 화면에 "ok"만 출력된다. 다시 말해, 문자열 출력을 할 때는 널문자을 만나면 출력이 종료되게 된다.

(2) 문자열 예외 처리

문자열이 20자 이하이고 숫자, 소문자 알파벳 그리고 공백으로 이루어졌는지를 확인한다. 만약에 사용자가 올바르지 않은 문자열을 입력하는 경우에는 사용자가 올바른 문자열을 입력할 때까지 다시 문자열을 입력 받는다.

사용자가 입력한 문자열에 따라서 화면에 다른 메시지를 출력한다.

- 20자를 초과하는 문자열을 입력한 경우
  Figure.8 (a)와 같이 **"please, put less than 20 letters"**라는 메시지를 출력
- 숫자, 소문자 알파벳, 공백이 아닌 글자를 입력한 경우
  Figure.8 (b)와 같이 **"please, put a string with 0-9 and a-z"**라는 메시지를 출력
- 20자를 초과하면서 지정되지 않은 문자가 포함된 문자열의 경우
  Figure.8 (c)와 같이 **"please, put a string with 0-9 and a-z"**라는 메시지를 출력

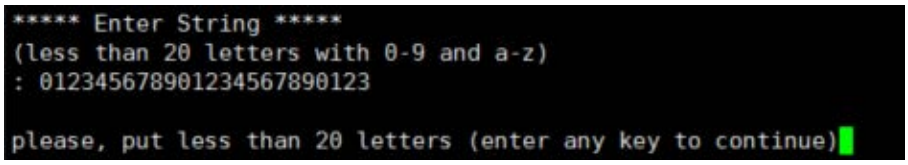


Figure.8 (a) 20자를 초과하는 문자열을 입력한 경우 출력 화면

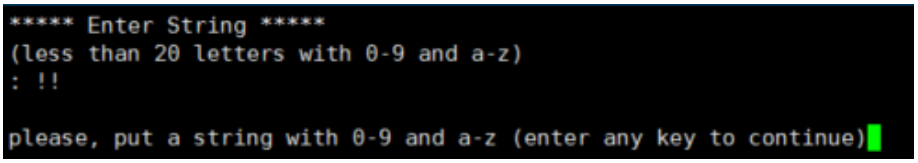


Figure.8 (b) 지정 글자 (숫자, 소문자 알파벳, 공백) 이외의 글자를 입력한 경우 출력 화면

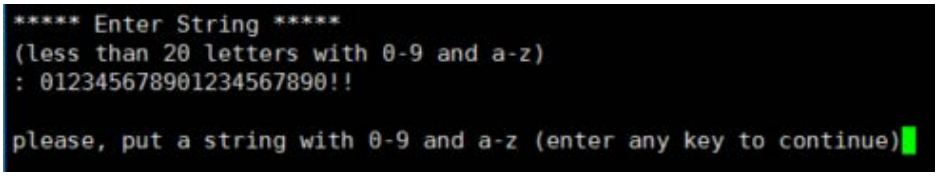


Figure.8 (c) 20자를 초과하면서 지정 글자 이외의 글자를 포함한 경우 출력화면

메시지 출력 후 사용자가 키보드 입력을 하면 **화면을 지운 뒤 Figure.5의 화면으로 돌아가서 사용자에게 다시 문자열을 입력 받는다.**

(3) 문자열 출력패턴

문자열 입력 후에는 화면을 지운 뒤 Figure.9과 같이 사용자로부터 LED 전광판의 이동패턴을 선택하게 된다. "좌로 이동", "우로 이동", "위로 이동", 그리고 "아래로 이동"의 4가지의 패턴이 존재한다. 각 이동패턴에 대한 설명은 다음 섹션에서 설명한다.

Figure.9 LED 전광판 출력 (이동) 패턴 선택 화면

사용자는 1에서 4사이의 숫자가 입력가능하고 이외의 입력이 있으면 예외처리를 한다.

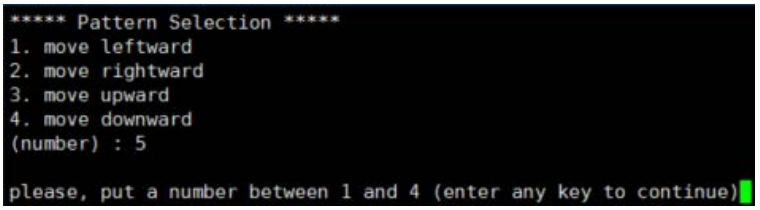


Figure.10 (a) 1-4 이외의 숫자가 입력된 경우

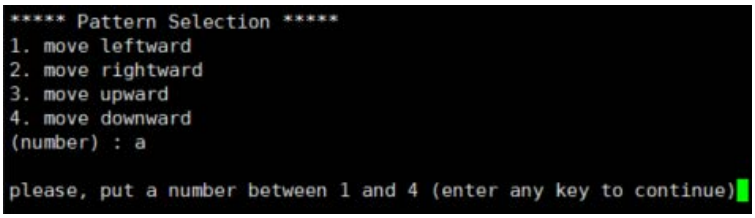


Figure.10 (b) 알파벳이 입력된 경우

위의 Figure.10은 1-4 이외의 입력을 한 경우의 실행화면을 보여주고 있다. 올바른 숫자를 입력하라는 "please, put a number between 1 and 4 (enter any key to continue)" 메시지를 출력 후 키보드 입력이 있으면 화면을 지우고 Figure.9 화면으로 이동한다.

III. 프로그램 실행 화면

사용자 입력이 끝난 뒤에는 문자열을 사용자가 지정한 방식에 따라서 LED 전광판에 출력한다.

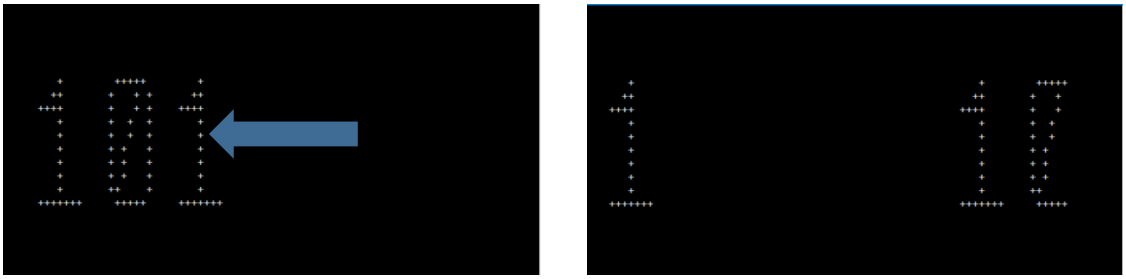
(1) 초기 LED 전광판 화면 출력

사용자 입력이 끝난 뒤에는 이전 화면을 지우고 사용자가 입력한 문자열을 화면에 출력한다. 다음 그림 (Figure.11)은 사용자 문자열이 "101"인 경우에 초기 화면을 보여준다.

Figure.11 문자열 "101"의 초기 출력화면

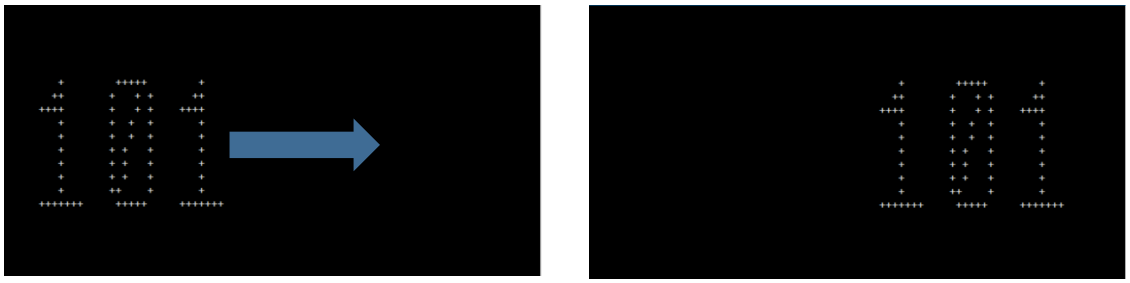
(2) 문자열 이동

사용자가 선택한 이동패턴에 따라서 LED 전광판에 출력된 내용이 변경하게 된다. 각 이동패턴에 따른 화면 변경은 다음의 Figure.12-15과 같다. "좌로 이동"은 화면에 출력된 글자가 우->좌 방향으로 움직인다 (Figure.12). 반대로 "우로 이동"은 글자가 좌->우 방향으로 이동한다 (Figure.13). "위로 이동"은 화면에 출력된 글자가 아래->위 방향으로 움직인다 (Figure.14). 이와 반대로 "아래로 이동"은 글자가 위->아래 방향으로 이동하게 된다 (Figure.15).



(a) 이동 전 LED 화면 (b) 이동 후 LED 화면

Figure.12 좌로 이동 예시 (move leftward 선택 시)

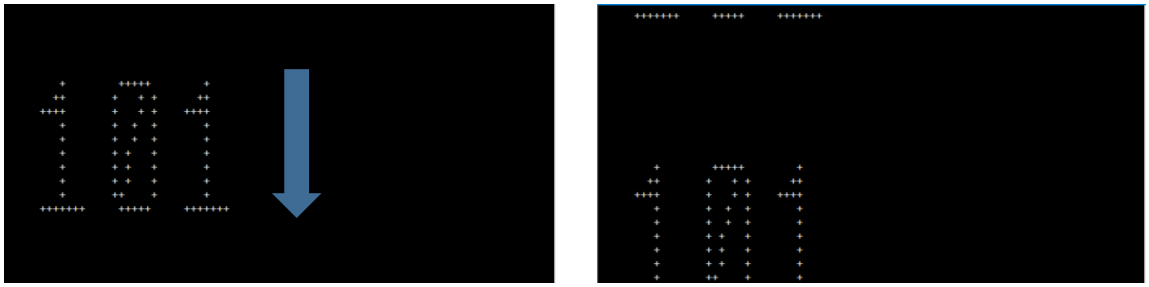


(a) 이동 전 LED 화면 (b) 이동 후 LED 화면

Figure.13 우로 이동 예시 (move rightward 선택 시)

(a) 이동 전 LED 화면 (b) 이동 후 LED 화면

Figure.14 위로 이동 예시 (move upward 선택 시)



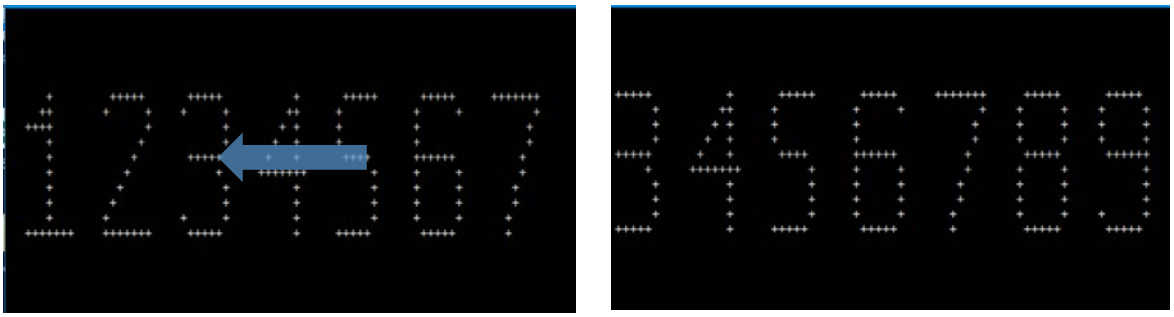
(a) 이동 전 LED 화면 (b) 이동 후 LED 화면

Figure.15 아래로 이동 예시 (move downward 선택 시)

LED 전광판이 출력패턴에 따라서 내용이 이동하게 되며 한쪽 경계면 (상, 하, 좌, 우)에 도달하면 반대쪽으로 출력된다. 예를 들어, 문자열 "1234"를 좌측으로 이동하는 경우에 맨 앞의 글자인 1이 좌측 경계로 들어가면 반대쪽에서 나타나면서 화면에는 "2341"로 보여지게 된다.

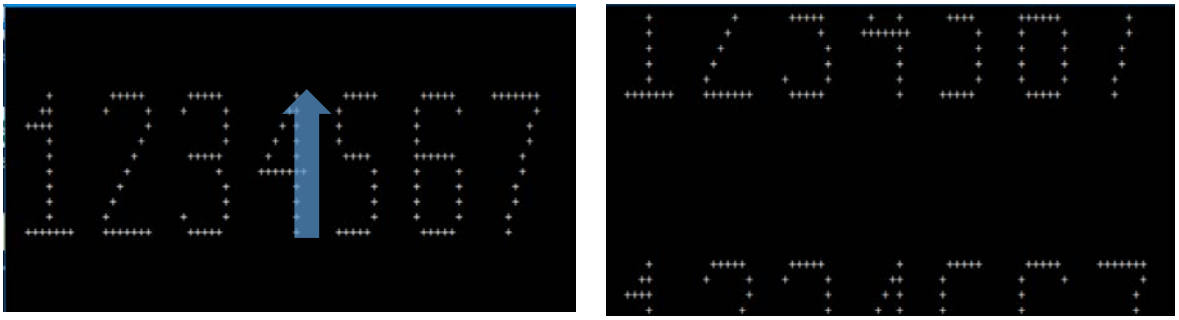
(3) 길이가 긴 문자열 출력

본 과제에서는 LED 전광판의 크기를 75 X 20로 하고 각 글자를 11 X 14 크기로 표현함에 따라서 한 번에 출력 가능한 글자의 수는 약 6.8개이다. 표현 가능한 문자열의 최대 길이가 20임에 따라서 길이 8이상의 문자열은 한번에 화면에 출력할 수 없게 된다. 다음 그림들은 문자열 "123456789"을 "좌로 이동" (Figure.16)과 "위로 이동" (Figure.17)을 보여준다.



(a) 이동 전 LED 화면 (b) 이동 후 LED 화면

Figure.16 "123456789"를 "좌로 이동"하는 경우 예시



(a) 이동 전 LED 화면 (b) 이동 후 LED 화면

Figure.17 "123456789"를 "위로 이동"하는 경우 예시

Figure.16을 보면 문자열이 좌측으로 이동하면서 우측에서 화면에 표시되지 않았던 8과 9가 나타난다. 이후 좌측으로 사라진 숫자들이 9 다음에 이어서 나타나게 된다. 위로 이동하는 Figure.17을 보면 초기에 화면에 표시된 1-7까지의 숫자만 위쪽 방향으로 이동하게 된다.

(4) 문자열 이동속도

화면에 출력되는 LED 전광판은 1초에 한 번씩 정해진 방향으로 이동하게 된다. 75 X 20 크기의 LED 전광판을 가로 75칸 세로 20칸의 사각 셀의 집합으로 보자. **1초 마다** 각 셀에 내용이 지정된 방향으로 옮겨가게 된다. 그림 Figure.18은 "좌로 이동"으로 출력되는 LED 전광판의 이동을 나타낸다. 30번째 열에 들어있는 값들이 매 1초마다 좌측 열로 옮겨간다. 따라서 처음에 30번째 열에 들어있던 내용은 29초후에 1번째 열에 담겨 있게 되고 30초에는 화면에서 사라지게 된다. "우로 이동"의 경우에는 반대로 동작하게 된다. "위로 이동"과 "아래로 이동"의 경우에는 각 행의 내용이 위쪽 행과 아래쪽 행으로 매초마다 이동하게 된다.

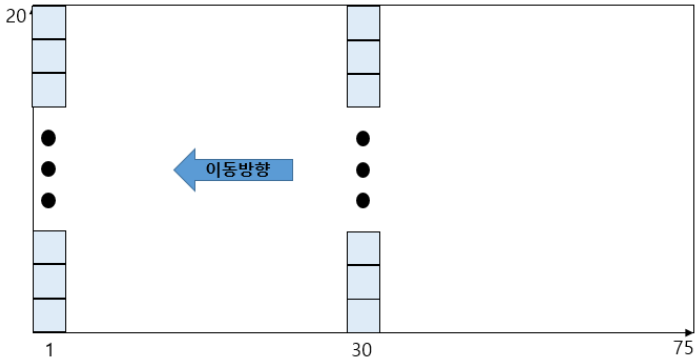


Figure.18 LED 전광판의 문자열 이동 예시

(5) LED 전광판 출력 종료

화면에 LED 전광판이 사용자가 지정한 이동패턴에 따라서 출력되고 있을 때, 사용자의 입력에 따라서 프로그램 종료 또는 초기 화면으로 돌아가게 된다. 사용자가 'q' 키를 누른 경우에는 프로그램이 종료된다. 사용자가 'q' 이외의 키를 입력한 경우에는 프로그램은 아무 반응을 하지 않는다.

(참고) 키보드 입력 여부 확인하기

윈도우의 비쥬얼스튜디오에서는 conio.h에 포함된 kbhit 함수를 이용하여 키보드 입력이 있었는지 확인을 할 수 있다. 하지만, conio.h는 윈도우를 위해 제작된 라이브러리로 리눅스 환경에서는 사용할 수 없다. 따라서 본 과제에서는 conio.h의 kbhit 함수와 동일한 역할을 하는 kbhit()을 제공하도록 한다. (제공하는 함수의 사용법은 문서 뒤의 Tip을 참고)

IV. 요구사항

이 섹션에서는 본 과제에서 구현해야 할 요구사항을 설명한다.

(1) LED전광판

- LED전광판을 표현하기 위해서 2차원배열을 이용한다.
- 화면에 출력 시 문자 75개를 한 줄로 이용한다.
- 화면에 출력 시 20줄을 이용한다.

(2) 사용자 입력 화면

- 프로그램 실행 시 사용자로부터 LED 전광판에 출력할 문자열과 이동패턴을 입력 받는다.
- 각 정보입력화면이 출력되기 전에는 이전 화면을 지운다.
  - 화면을 지운 후 사용자 문자열 입력 화면 출력한다.
  - 사용자 문자열 입력이 완료되면 화면을 지운 후 이동 패턴 출력화면을 출력한다.

(3) 사용자 입력 문자열

- 입력 가능한 글자는 숫자 (0-9)와 소문자 알파벳 (a-z) 그리고 공백 (spacebar)이다.

- 입력 문자열의 최대 길이는 20이다. (예, "12345678901234567890", "1234 1234 1234 12345")
- 올바르지 않은 문자열이 입력된 경우 경고메시지를 출력하고 다시 입력 받는다.
  - 길이가 20을 초과하는 문자열의 경우, "please, put less than 20 letters "라고 출력한다.
  - 비허가 글자가 포함된 경우, "please, put as string with 0-9 and a-z "라고 출력한다.
  - 길이가 20을 초과하면서 비허가 글자가 포함된 경우, "please, put as string with 0-9 and a-z "라고 출력한다
  - 이후 "(enter any key to continue)"라고 출력 후 사용자가 키보드 입력을 하면 화면을 지우고 다시 문자열 입력화면으로 돌아가서 문자열을 입력 받는다.
- (채점 시 프로그램의 동작확인을 위해서는 25자 이하의 문자열을 이용한다.)

(4) 출력 패턴 선택

- 입력 가능한 글자는 숫자 (1-4)이다.
- 사용자가 올바르지 않은 입력을 하면 예외처리를 한다. (예, "12", "a", "123a", "ab123")
  - 화면에 "please, put a number between 1 and 4 "를 출력한다.
  - 이후 "(enter any key to continue)"라고 출력 후 사용자가 키보드 입력을 하면 화면을 지우고 다시 출력 패턴 선택화면으로 돌아간다.

(5) LED 전광판 출력

- 사용자로부터 입력을 받은 후에는 화면을 지운 후 75 X 20의 LED 전광판을 출력한다.
- LED 전광판에 출력되는 각 글자는 7 X 10의 크기를 가진다. (공백 문자 포함)
- 각 글자는 상하좌우에 2의 여백을 가진다.
- 각 글자는 LED 전광판에서 여백 포함 11 X 14의 크기를 차지한다.
- 문자열은 LED 전광판에 좌측에 위치한다.
- 사용자의 문자열의 길이가 일정길이 이상이면 표현 가능한 만큼만 화면에 출력된다.

(6) LED 전광판 이동

- LED 전광판이 화면에 출력된 후 지정된 이동패턴에 따라서 화면 내용이 변경된다.
- LED 전광판의 내용은 1초에 한 번씩 정해진 방향으로 이동하게 된다.
- 글자의 이동방향에 따라서 LED 전광판의 경계로 넘어가게 되면 반대편에서 나타난다.
- 길이가 긴 문자열을 좌/우로 이동할 때 초기 LED 전광판 화면에서 보이지 않았던 글자가 나타나게 된다.
- 길이가 긴 문자열을 위/아래 이동할 때는 초기 LED 전광판 화면에 보였던 글자만 이동하게 되며 화면에 출력되지 않은 글자들은 화면에 보이지 않는다.

(7) LED 전광판 출력 종료

- 화면에 출력되고 있는 LED 전광판은 사용자의 입력에 따라서 다르게 동작한다.
  - 'q' 키를 입력한 경우, 프로그램이 종료된다.
  - 이외의 키를 입력한 경우, 프로그램은 계속 LED 전광판을 출력한다.

(8) 프로그램 화면 구성

- 프로그램 실행 시 화면에 출력되는 내용들은 본 과제의 예시와 최대한 유사하게 작성한다.

(9) 소스코드

- 유사한 목적의 함수들을 모아서 한 파일을 작성한다. 다음의 파일들은 반드시 작성한다.
  - 사용자 입력을 위한 메뉴 출력을 위한 **menu.h**와 **menu.c**
  - 사용자 입력을 체크하기 위한 **checker.h**와 **checker.c**
  - LED 전광판 처리를 위한 **led.h**와 **led.c**
  - main함수가 적성되어 있는 **led_main.c**
- 이외에 작성한 프로그램에 필요한 모든 파일을 제출한다.
- 제출한 파일들이 컴파일 되지 않으면 0점 처리된다.
- (다수의 소스코드를 컴파일 하는 방법은 문서 뒤의 Tip을 참고한다.)

(10) 코드작성

- main함수를 재귀로 호출 하지 않는다. (예. main함수에서 main함수 호출하면 안됨)
- 전역변수를 사용하지 않는다.
- 포인터 전까지 다루었던 문법 (조건문, 반복문, 함수, 배열 등)을 이용하여 작성하며 포인터, goto문, 구조체 등을 사용하지 않는다. 또한, 외부 라이브러리(string.h 등)의 사용은 불가하다.
- *menu 파일*에 반드시 작성해야 하는 함수는 다음과 같다.
  - **void textMenu(char string[], int stringSize)**
    Figure.5와 같이 LED 전광판에 출력할 문자열을 입력 받음.
    사용자가 올바르지 않은 문자열을 입력하면 다시 요청 (inputStringChecker 함수 사용)
  - **int patternMenu()** // LED전광판 이동패턴 선택 화면출력 및 메뉴 선택 받음

- *checker 파일*에는 다음의 함수를 작성한다.
  - **int inputStringLength(char string[])** // 문자열의 길이를 구하는 함수
    배열의 처음에서 널문자 전까지의 길이를 출력. "1234₩0" 문자열의 경우 4를 리턴함.
    사용자가 입력한 문자열의 길이가 필요할 때 사용함.
  - **int inputStringChecker(char string[], int stringSize)**
    LED 전광판에 출력할 문자열이 올바른지 확인하는 함수로 검사 결과를 리턴.
    반드시 의미 있는 값을 리턴해야하며 리턴값은 각자 정의한다.

- *led 파일*에는 다음의 함수를 작성한다.
  - **void moveLEDLeft(char led[][], ...)** //LED전광판 내용을 좌로 이동
  - **void moveLEDRight(char led[][], ...)** //LED전광판 내용을 우로 이동
  - **void moveLEDUp(char led[][], ...)** //LED전광판 내용을 위로 이동
  - **void moveLEDDown(char led[][], ...)** //LED전광판 내용을 아래로 이동
  - 배열 이외에 필요한 매개변수가 있으면 추가하여 구현한다. (문서 뒤의 Tip 참고)

V. (보너스) 추가 구현 내용

위에서 설명한 LED 전광판 프로그램의 기능 이외에 추가적인 기능을 구현하면 추가 점수를 부여한다. **단, 기본기능을 구현하지 않았거나 동작하지 않았을 경우에는 추가 점수는 없다**. 요구하는 추가 기능은 LED 전광판 이동속도와 LED 전광판 색 설정이다. 각 기능은 선택적으로 구현 가능하며 한 개만 구현한 경우 1점의 추가점수가 있고 두 기능을 다 구현하면 총 2점의 추가점수가 있다. 추가 기능은 위에서 설명한 이동패턴과 같이 동작해야 한다. 예를 들어, LED 전광판 색 기능을 구현하면 사용자가 선택한 색과 이동패턴으로 화면에 LED 전광판이 나타나게 된다.

(1) LED 전광판 이동속도 (추가 1점)

이동패턴 선택 이후 LED 전광판의 이동속도를 사용자에게 선택하게 한다. Figure.19은 선택 가능한 이동속도를 보여준다.

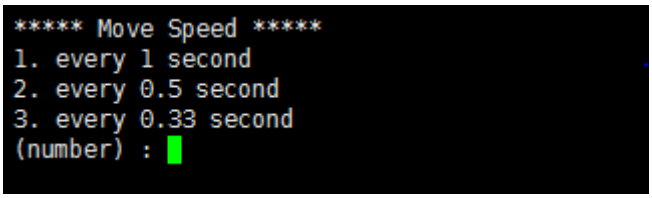


Figure.19 LED 전광판 출력 시 이동 속도 선택 화면

선택 가능한 이동속도는 1부터 3까지 3 단계가 존재하며 각 단계는 다음과 같은 의미를 가진다.

- 1 단계: 선택한 이동패턴에 따라서 LED 전광판이 1초 마다 이동
- 2 단계: 선택한 이동패턴에 따라서 LED 전광판이 0.5초 마다 이동 (1초에 2번 이동)
- 3 단계: 선택한 이동패턴에 따라서 LED 전광판이 0.33초 마다 이동 (약 1초에 3번 이동)

따라서, 3단계가 1단계에 비해서 빠르게 이동하게 된다. 앞선 섹션에서 설명한 것과 같이 사용자의 입력에 따라서 예외처리를 해야 한다 (Figure.20).

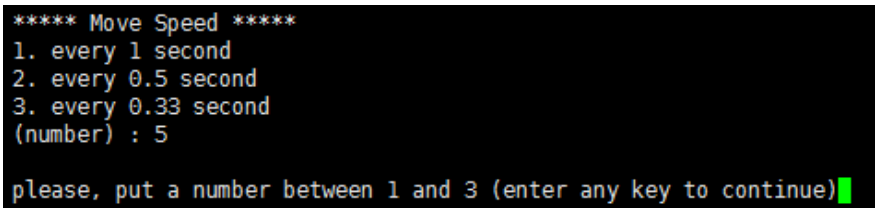


Figure.20 1-3 이외의 입력을 한 경우

(힌트)

sleep함수는 초 단위로 프로그램을 정지시킬 수 있다. 하지만 본 기능에서는 밀리초 수준의 프로그램 정지가 필요하다. 이때, usleep 함수를 대신 사용한다 (sleep 함수와 마찬가지로 unistd.h에 포함되어 있음).

(2) LED 전광판 색상 (추가 1점)

Figure.21은 LED 전광판의 색을 선택하는 화면으로 사용자는 4가지 색상 (빨강, 노랑, 녹색, 파랑) 중 하나를 선택할 수 있다.

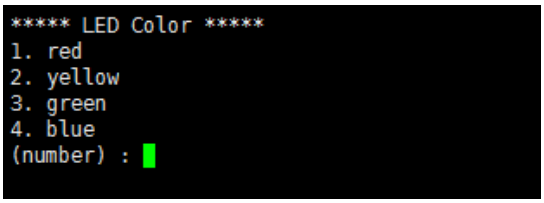


Figure.21 LED 전광판 색상 선택 화면

사용자가 색상을 선택을 하고 나면 LED 전광판 화면이 지정한 색으로 출력이 된다. 마찬가지로 사용자의 입력에 따라서 예외처리를 해야 한다 (Figure.22).

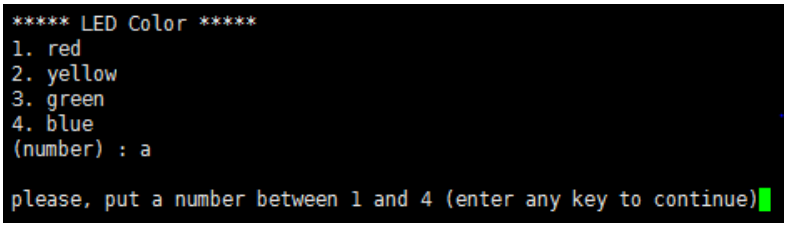


Figure.22 1-4 이외의 입력을 한 경우

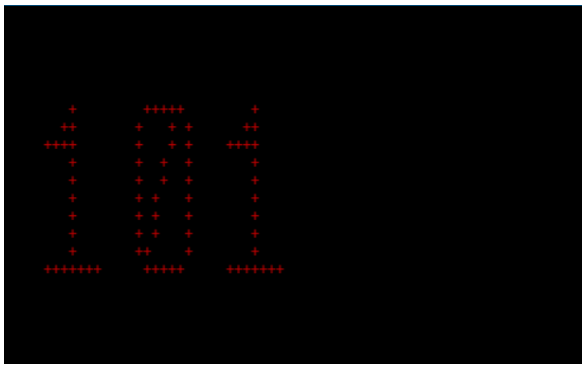


Figure.23 선택된 색으로 LED 전광판 출력 예시 (빨강 선택)

사용자가 1-4 사이의 숫자를 제대로 입력하게 되면 지정된 색으로 LED 전광판이 출력된다. 위의 Figure.23는 사용자가 1 (빨강)을 입력한 경우의 출력화면을 나타낸다. (이후 이동패턴에 따라서 내용이 변경된다.)

(힌트)

printf 함수를 이용하여 화면 출력표현을 변경 할 수 있다. printf("₩033[%dm", **출력표현**)으로 출력 지정할 수 있으며 "출력표현"으로 되어 있는 부분에 숫자를 넣으면 숫자가 해당하는 표현방법으로 화면에 출력 된다. 또는 printf("₩033[숫자m")으로도 사용할 수 있다. Figure.24는 출력표현으로 4를 이용한 것으로 밑줄표시가 되는 것을 확인할 수 있다.

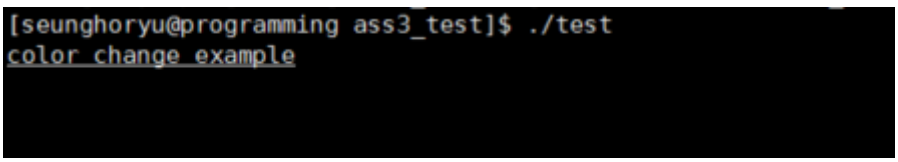


Figure.24 print("₩033[4mcolor change example") 출력 화면

위의 방법은 콘솔 화면 출력방법 자체에 적용되는 것이므로 LED 전광판 출력이 끝날 때 (프로그램 종료 또는 초기화면으로 돌아갈 때) printf("₩033[0m")을 이용하여 설정한 표현방법을 제거한다.

Tips

■ 화면 지우기

화면을 지우기 위해서는 stdlib.h를 포함된 system 함수를 사용한다. 다만, 윈도우의 비주얼 스튜디오와 리눅스의 프로그래밍 환경이 다름에 따라서 system 함수로 넘겨줄 매개변수로 "cls" 대신 "clear"를 사용한다.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
	printf("HELLO WORLD\n");
	system("clear");

	return 0;
}
```

■ 공백을 포함한 문자열 입력 받기

사용자로부터 키보드 입력을 받기 위해서 일반적으로 scanf 함수를 많이 사용한다. scanf("%s")를 이용하면 공백 앞까지를 문자열로 입력 받게 된다. 예를 들어, 사용자가 "123 456"이라 입력하고 엔터키를 입력하면 scanf 함수로는 공백문자 전까지인 "123"만 문자열로 입력 받는다. 공백을 포함한 문자열을 입력 받기 위해서는 fgets 함수를 사용한다. fgets 함수를 이용하여 입력버퍼에서 정해진 크기만큼을 문자열로 입력 받을 수 있다.

fgets 함수 사용 후에도 입력버퍼에 남아있는 값이 있을 수 있다. 이로 인해 입력버퍼로부터 원치 않은 값을 받아오게 되어 프로그램이 이상하게 동작할 수 있다. 이러한 문제를 방지하기 위해서 입력버퍼를 비워줄 필요가 있다. __fpurge(stdin)를 이용하여 입력버퍼의 내용을 지우도록 한다.

아래에 있는 예제 코드는 fgets와 __fpurge 함수의 사용 예제이다. **이를 참고하여 LED 전광판에 출력할 사용자 문자열을 입력 받도록 한다.**

```
#include <stdio.h>

int main()
{
	char str[25];

	// str: 문자열을 입력 받을 char 배열
	// 25 : 입력 받을 문자열 길이 (널문자 포함)
	// stdin: 문자열을 읽어 들일 곳, stdin은 표준 입력으로 키보드 입력 의미
	fgets(str, 25, stdin);	// 키보드로부터 최대 25자 입력받음

	__fpurge(stdin);		// 입력받고 버퍼에 남은 내용 지움

	return 0;
}
```

■ 배열 사용

- 배열의 크기를 선언할 때 변수를 사용하면 안 된다. (char led[i][j];).
- 충분한 크기로 배열을 선언 한 뒤 필요한 부분만 화면에 출력한다.
- 2차원 배열을 함수의 매개변수로 넘겨주기 위해서는 led[][size]와 같은 표현을 사용한다.

```
int arrayTestFunc(int arr[][colSize], int …);
int main()
{
	int arr[3][3] = {0, 1, 2, 3, 4, 5, 6, 7, 8};
	int arrSize = 9;

	arrayTestFunc(arr, arrSize);

	return 0;
}
```

■ 프로그램 일시 정지

LED 전광판의 내용은 1초에 한 번씩 이동한다. 이는 1초간 프로그램을 일시 정지 시켰다가 다시 동작하면 된다. unistd.h에 이러한 기능을 하는 함수인 sleep가 있다. sleep 함수는 second 단위로 프로그램을 일시 정지시킨다.

```
#include <stdio.h>
#include <unistd.h>
int main()
{
	while(1)
	{
		printf("1 second\n");
		sleep(1);		// 매개변수는 초단위. 1초 일시정지
	}
	return 0;
}
```

■ 키보드 입력여부 확인 및 입력 값 확인

본 과제에서 제공하는 keyboard.h와 keyboard.c를 사용한다. 해당 파일에는 키보드 입력여부를 알 수 있는 kbhit 함수가 구현되어 있다. (리눅스에서는 윈도우와 달리 kbhit 함수가 없다.) 컴파일할 때 keyboard.c를 포함하여 프로그램을 빌드한다. 각 함수의 사용법은 다음과 같다.

```
#include <stdio.h>
#include <unistd.h>
#include "keyboard.h"	// kbhit()을 사용하기 위해서 추가
int main() {
	while(1)
	{
		while(!kbhit())  // 키보드 입력이 없을 때
		{
			printf("not pressed any key\n");
			sleep(1);		// 매개변수는 초단위. 1초 일시정지
		}

		if(getchar() == 'q') // 'q'를 입력했을 때
		{
			printf("program will be terminated\n");
			break;  // 프로그램 종료를 위해 반복문 탈출
		}
	}
	return 0;
}
```

■ 다수의 소스코드로 작성된 프로그램 빌드

본 과제에서는 유사한 기능을 하는 함수를 모아서 별도의 파일에 작성하도록 하였다 (예, menu, led 등). 다수의 소소코드를 컴파일하여 프로그램을 생성하기 위한 두 가지 방법이 있다. 여기서는 이 두 가지 방법에 대해서 설명하도록 한다. 설명을 간단히 하기 위해서 main.c, led.c, led.h 3개의 파일을 작성하였다고 하자.

1. 리눅스 명령을 사용하는 방법

```
[seunghoryu@programming test]$ ls
led.c  led.h  main.c
[seunghoryu@programming test]$ gcc -o assign3 main.c led.c
[seunghoryu@programming test]$ ls
assign3  led.c  led.h  main.c
[seunghoryu@programming test]$
```

위의 그림은 리눅스 접속화면에서 gcc를 이용하여 소스파일들을 컴파일하여 프로그램을 생성하는 과정을 나타낸다. 위에서 이야기한 3개의 파일이 동일한 디렉토리에 존재할 때, gcc 명령을 이용하여 소스코드를 컴파일하여 프로그램을 생성할 수 있다. gcc 명령의 의미는 다음과 같다.

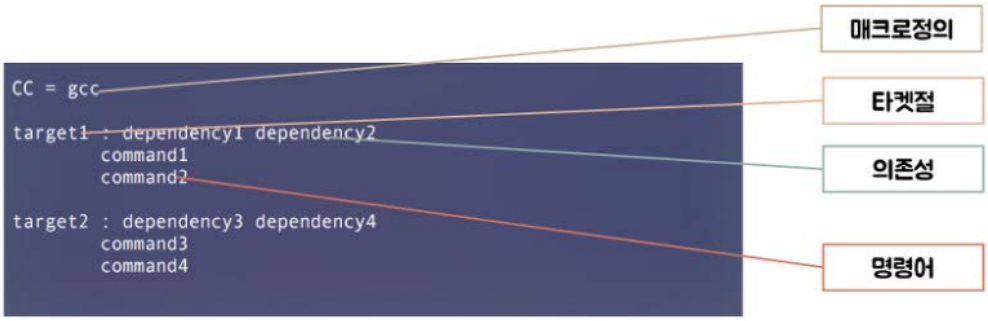
- **gcc -o "프로그램 이름" 소스파일**
  - -o 옵션: 생성할 프로그램을 지정하는 것을 의미
  - "프로그램 이름": 생성될 프로그램의 이름
    (-o "프로그램 이름"이 없는 경우, a.out라는 이름으로 프로그램이 생성된다.)
  - 소스파일: 프로그램에 필요한 소스파일 (.c 파일)

gcc 명령 이후에 지정한 이름인 assign3으로 프로그램이 생성된 것을 확인 할 수 있다.
(프로그램을 실행하기 위해서는 ./assign3라고 입력한다.)

2. make를 이용하는 방법

리눅스에서 명령어를 이용하여 컴파일하는 과정을 보다 편리하고 간편하게 하기 위해서 make를 이용한다. make는 makefile에 작성되어 있는 내용에 따라서 소스코드를 컴파일하여 프로그램을 생성한다.

- makefile의 구성



- 목적파일 (타켓): 명령어가 수행되어 나온 결과를 저장할 파일
- 의존성: 목적파일을 만들기 위해 필요한 재료
- 명령어: 실행 되어야 할 명령어들
- 매크로: 코드를 단순화 시키기 위한 방법

- Makefile을 생성

```
[seunghoryu@programming test]$ vim makefile
```

vim을 이용하여 makefile을 작성한다.

- makefile 내용

```
1 assign3 : main.o led.o
2     gcc -o assign3 main.o led.o
3
4 main.o : led.h main.c
5     gcc -c main.c
6
7 led.o : led.c
8     gcc -c led.c
9
10 clean :
11     rm *.o assign3
```

위와 유사하게 makefile 내용을 작성하고 vim을 종료한다.

- 프로그램 생성

```
[seunghoryu@programming test]$ ls
led.c  led.h  main.c  makefile
[seunghoryu@programming test]$ make
gcc -c -o main.o main.c
gcc -c -o led.o led.c
gcc -o assign3 main.o led.o
[seunghoryu@programming test]$ ls
assign3  led.c  led.h  led.o  main.c  main.o  makefile
[seunghoryu@programming test]$
```

소스파일과 makefile이 있는 디렉토리에서 make 명령을 사용하면 프로그램이 생성된다.

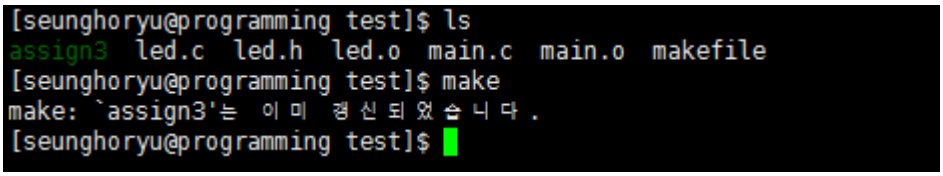
- 생성파일 제거

```
[seunghoryu@programming test]$ ls
assign3  led.c  led.h  led.o  main.c  main.o  makefile
[seunghoryu@programming test]$ make clean
rm *.o assign3
[seunghoryu@programming test]$ ls
led.c  led.h  main.c  makefile
[seunghoryu@programming test]$
```

makefile에서 정의한 clean을 보면 리눅스의 삭제 명령어인 rm이 사용되는 것을 것을 볼 수 있다. "make clean" 명령을 실행하면 프로그램 빌드시 생성된 오브젝트 파일(*.o)과 실행파일이 제거된다.

(참고)
makefile에 대한 추가적인 정보는 다음을 참고 (http://bowbowbow.tistory.com/12)

makefile을 이용하여 프로그램 빌드를 하려고 할 때 아래의 그림과 같이 "make: '실행파일 이름'는 이미 갱신되었습니다."라고 출력되고 빌드가 수행되지 않는 경우가 있다.



make에서는 소스파일과 오브젝트파일(*.o)의 갱신시간을 비교하여 컴파일 여부를 결정한다. 따라서 마지막 컴파일 시점으로부터 수정된 소스코드가 없어서 다시 컴파일을 할 필요가 없으므로 나오는 메시지이다.

다시 컴파일 하고 싶으면 리눅스 프롬프트에서 make clean 이라고 입력하여 생성된 object(*.o)파일을 삭제한 후 하면 된다.