운영체제 5주차 과제

20135151 이갑성

|  |
| --- |
| 소스코드  <elevator.h>  #define TOP\_FLOOR 15  #define DELAY\_FACTOR 50000  typedef enum{OUT, IN} BUTTON\_STATUS;  typedef enum{DOWN, STOP, UP} DIRECTION\_STATUS;  typedef struct{  int current\_floor;  BUTTON\_STATUS \*buttons;  }ELEVATOR;  <main.h>  #include <stdio.h>    #include "elevator.h"  extern void init\_elevator(ELEVATOR \*);  extern void run\_elevator(ELEVATOR \*);  extern void term\_elevator(ELEVATOR \*);  <main.c>  #include "main.h"    int main(void){  static ELEVATOR elevator; //구조체 ELEVATOR의 elevator 선언  init\_elevator(&elevator); //엘베 메모리를 만들고, 층 초기화 작업해주는 함수  run\_elevator(&elevator); //해당 층으로 이동하는 함수  term\_elevator(&elevator); //엘베 메모리를 해제 시켜주는 함수  return 0;  }  <ops.h>  extern void move(ELEVATOR \*);  <ops.c>  #include <stdio.h>  #include <stdlib.h>  #include <ctype.h>  #include "elevator.h"  #include "ops.h"    //BUTTON\_STATUS 크기의 메모리를 16개 만들어주고 각 메모리에 반복문을 통해 OUT이라는 값을 넣어준다.  //엘레베이터의 현재층을 1층으로 해준다.  void init\_elevator(ELEVATOR \*elev)  {  elev->buttons = calloc(TOP\_FLOOR+1, sizeof(BUTTON\_STATUS));  for(int i = 0; i <= TOP\_FLOOR; i++){  elev->buttons[i] = OUT;  }  elev->current\_floor = 1;  return;  }  void run\_elevator(ELEVATOR \*elev)  {  char buffer[81];  int floor;  char \*p\_str\_in;  printf("\n\nThis elevator goes from basement \"0\" to floor \"%d\"", TOP\_FLOOR);  printf("\nType floors & press return to start. If no new floors,");  printf("\npress return key. To quit, key EOF(Ctrl+C)");  printf("\n\nPlease enter floors: ");  //아래 while문은 입력받은 값이 문자인지 숫자인지 구분하고 숫자일때 가고자 하는 층으로 대입하여 엘베를 이동하는 동작을 한다.  //과정을 자세히 말하면 buffer에 크기 최대 80 문자열로 입력받는 작업을 반복하는데  //p\_str\_in은 buffer의 시작주소를 가르키게 한다.  //\*p\_str\_in이 \n을 만날때까지 while문을 돌리는데 반복문의 과정은  //\*p\_str\_in이 공백을 만나면 다음 배열로 이동(공백을 만나지 않을때까지)시켜주고  //만약 p\_str\_in 배열의 값이 문자면 "Invalid floor"을 출력해주고 p\_str\_in을 다음으로 이동시켜 NULL을 가르키고, 숫자면 p\_str\_in을 정수형 floor에 넣어준다.  //넣어주고 floor가 현재 층과 같다면 "Already on floor"을 출력해주고 현재층과 다를때는  //floor가 0층보다 아래의 값인지 TO\_FLOOR보다 큰값인지 검사를 해주는데 둘중에 해당되면 "Invalid floor"을 출력해주고  //0~TOP\_FLOOR사이 값이면 elev 구조체의 멤버 button의 floor번째 메모리를 OUT에서 IN으로 해준다.  //마지막으로 \*p\_str\_in은 숫자일테니까 p\_str\_in을 증가시켜 다음 배열(NULL) 가르키게 한다.  while(fgets(buffer, 81, stdin)){  p\_str\_in = buffer;  while(\*p\_str\_in != '\n'){  while(\*p\_str\_in == ' '){  p\_str\_in++;  }  if(!isdigit(\*p\_str\_in)){  printf("\aInvalid floor %c\n", \*p\_str\_in);  p\_str\_in++;  }else {  sscanf(p\_str\_in, "%d", &floor);  if(floor == elev->current\_floor){  printf("\n\aAlready on floor %d.", elev->current\_floor);  }else {  if(floor < 0 || floor > TOP\_FLOOR){  printf("\n\a%d invalid floor", floor);  } else {  elev->buttons[floor] = IN;  }  }  while(isdigit(\*p\_str\_in)){  p\_str\_in++;  }  }  }  move(elev); //엘베를 이동하는 함수 move를 호출  printf("\n\nPlease enter floors: ");  }  return;  }  //엘베의 buttons의 메모리를 해제시켜주는 함수  void term\_elevator(ELEVATOR \*elev){  free(elev->buttons);  return;  }  <move.h>  static void move\_up(ELEVATOR \*);  static void move\_down(ELEVATOR \*);  static void time\_pass(int);  extern bool any\_up\_request(ELEVATOR \*);  extern bool any\_down\_request(ELEVATOR \*);  <move.c>  #include <stdio.h>  #include <stdbool.h>  #include "elevator.h"  #include "move.h"  void move(ELEVATOR \*elev){  static DIRECTION\_STATUS direction = STOP; //열거형 변수 direction을 STOP으로 해준다.  bool any\_up, any\_down;  any\_up = any\_up\_request(elev); //만약 현재층이 1층이고 내가 선택한 층이 5층이면 any\_up에서는 ture일 것이다.  any\_down = any\_down\_request(elev); //만약 현재층이 5층이고 내가 선택한 층이 1층이면 any\_down에서는 true일 것이다.  //any\_up과 any\_down은 서로 다른 값을 갖게 된다.예를들어 하나가 true면 나머지는 flase로 되는 케이스  //direction == STOP일때는 any\_up과 any\_down의 결과값에 따라 UP DOWN이 결정된다.  //directon == UP일때는 조건문을 통해 내려가는건지 한번 확인 해주고,그게 아니면 STOP을 해준다.  //direction == DOWN일때도 마찬가지로 조건문을 통해 올라가는건지 한번 확인해주고, 그게 아니면 STOP을 해준다.  if(direction == UP){  if(!any\_up && any\_down){  direction = DOWN;  }else {  if(!any\_up && !any\_down){  direction = STOP;  }  }  } else if (direction == DOWN){  if(!any\_down && any\_up){  direction = UP;  }else {  if(!any\_down && !any\_up){  direction = STOP;  }  }  } else if(direction == STOP){  if(any\_up){  direction = UP;  } else{  if(any\_down){  direction = DOWN;  }  }  }  //만약 direction이 UP인 경우에 move\_up함수를 호출하고 DOWN경우엔 move\_down함수를 호출해준다. 그밖에는 아래 문장을 출력해준다.  if(direction == UP){  move\_up(elev);  } else {  if(direction == DOWN){  move\_down(elev);  } else {  printf("\n\*\*\*\*\* NO BUTTON PRESSED \*\*\*\*\*");  }  }  return;  }  //위층으로 이동하는 함수이다.  //과정은 엘베의 현재층을 한층 증가시켜주고 buttons의 현재층 메모리가 IN일때까지(선택한 층까지) 1초 보내고 현재층 문장출력해주고 1초보내고 층 증가시켜주고 반복작업을 한다.  //마침내 선택한 층에 도착하면 현재 층의 buttons의 메모리값을 OUT으로 넣어주고 1초를 흐르게 한뒤 아래 문장을 출력해준다.  static void move\_up(ELEVATOR \*elev){  printf("\nThe door is being closed ... We are going up.\n");  (elev->current\_floor)++;  while(elev->buttons[elev->current\_floor] != IN){  time\_pass(10000);  printf("Passing floor %d\n", elev->current\_floor);  time\_pass(10000);  (elev->current\_floor)++;  }  elev->buttons[elev->current\_floor] = OUT;  time\_pass(10000);  printf("The door is being opened ... \n");  printf("\n \*\*\*\*\* FLOOR %d \*\*\*\*\* \n", elev->current\_floor);  time\_pass(10000);  return;  }  //아래층으로 이동하는 함수이다.  //과정은 엘베의 현재층을 한층 감소시켜주고 buttons의 현재층 메모리가 IN일까지(선택한 층까지) 1초 보내고 현재층 문장 출력해주고 1초 보내고 층 감소시켜주고 반복작업을 한다.  //마침내 선태한 층에 도착하면 현재 층의 buttons의 메모리값을 OUT으로 넣어주고 1초를 흐르게 한뒤 아래 문장을 출력해준다.  static void move\_down(ELEVATOR \*elev)  {  printf("\nThe door is being closed ... We are going down.\n");  (elev->current\_floor)--;  while(elev->buttons[elev->current\_floor] != IN){  time\_pass(10000);  printf("Passing floor %d \n", elev->current\_floor);  time\_pass(10000);  (elev->current\_floor)--;  }  elev->buttons[elev->current\_floor] = OUT;  time\_pass(10000);  printf("The door is being opened ... \n");  printf("\n \*\*\*\*\* FLOOR %d \*\*\*\*\* \n", elev->current\_floor);  time\_pass(10000);  return;  }  //코드 실행을 지연시켜주는 함수이다.  static void time\_pass(int time)  {  for(long i = 0; i < (time \*DELAY\_FACTOR); i++);  return;  }  <request.c>  #include <stdio.h>  #include <stdbool.h>  #include "elevator.h"  //이동 하기로 한 층의 buttons의 메모리에는 IN으로 값이 들어 있을 것이다.  //for문을 통해 현재층부터 TOP\_FLOOR까지 buttons메모리에 IN이라는 값이 나올때까지 for문을 돌린다.  //IN이라는 값을 만나면 is\_any는 true가 될 것이고 반복문 작업을 마치고 리턴된다.  bool any\_up\_request(ELEVATOR \*elev)  {  bool is\_any = false;  for(int check = elev->current\_floor; check <= TOP\_FLOOR && !is\_any; check++){  is\_any = (elev->buttons[check] == IN);  }  return is\_any;  }  //이동 하기로 한 층의 buttons의 메모리에는 IN으로 값이 들어 있을 것이다.  //for문을 통해 현재층부터 TOP\_FLOOR까지 buttons메모리에 IN이라는 값이 나올때까지 for문을 돌린다.  //IN이라는 값을 만나면 is\_any는 ture가 될 것이고 반복문 작업을 마치고 리턴된다.  bool any\_down\_request(ELEVATOR \*elev)  {  bool is\_any = false;  for(int check = elev->current\_floor; check >= 0; check--){  is\_any = is\_any || (elev->buttons[check] == IN);  }  } |
| 실행결과 |