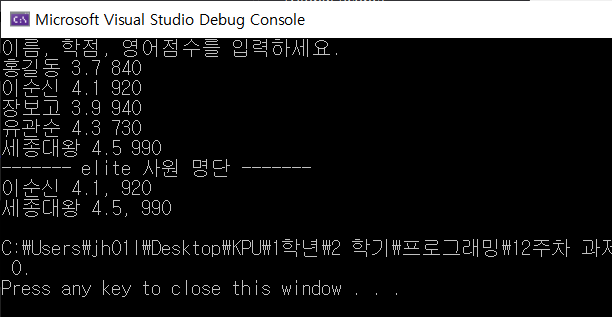
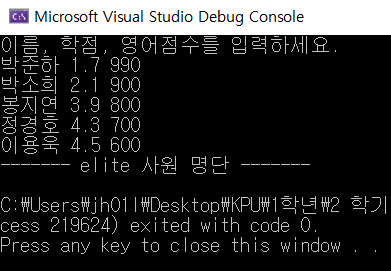
p1

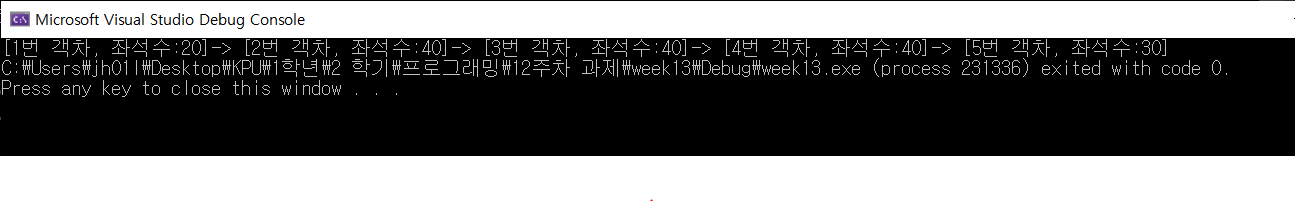
#define \_CRT\_SECURE\_NO\_WARNINGS   
#include <stdio.h>   
   
struct profile { //구조체 생성   
 char name[20];   
 double grade;   
 int english;   
};   
   
void input\_data(struct profile\* employee) {// 구조체 변수에 데이터를 입력하는 함수 원형   
 printf("이름, 학점, 영어점수를 입력하세요.\n");   
 for (int i = 0; i < sizeof(\*employee) / sizeof(double); i++) {   
 scanf("%s %lf %d", &(employee[i].name), &(employee[i].grade), &(employee[i].english)); // 구조체 배열 element에 접근 후, 대입   
 }   
   
}   
   
void elite(struct profile\* employee) { // 엘리트 사원을 출력하는 함수 원형   
 printf("------- elite 사원 명단 -------\n");   
   
 for (int i = 0; i < sizeof(\*employee) / sizeof(double); i++) { // 구조체 배열의 크기만큼   
   
 if (employee[i].grade >= 4.0 && employee[i].english >= 900) //조건식   
 printf("%s %.1lf, %d\n", employee[i].name, employee[i].grade, employee[i].english);   
 else   
 continue;   
 }   
}   
   
int main(void) {   
 struct profile new\_staff[10];   
 input\_data(new\_staff);   
 elite(new\_staff);   
   
 return 0;   
}





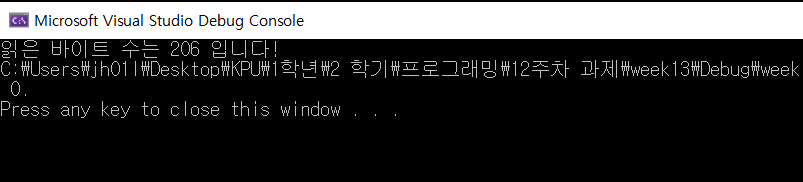
p2

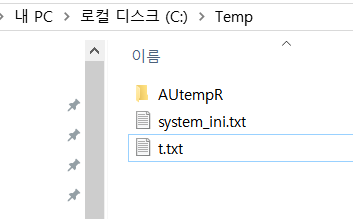
#include <stdio.h>   
#include <stdlib.h>   
   
typedef struct train Train;   
   
struct train   
{   
 int number; // 객차 번호   
 int seats; // 좌석 수   
 Train\* next;   
};   
   
void print\_train(Train\*); //객차 번호와 좌석수를 출력하는 함수원형   
   
int main(void)   
{   
 Train\* head = NULL, \* tail = NULL;   
 int i;   
 int train\_seats[5] = { 20, 40, 40, 40, 30 }; // 객차별 좌석수 정보   
   
 for (i = 0; i < 5; i++)   
 {   
 if (head == NULL)   
 {   
 head = tail = (Train\*)malloc(sizeof(Train));   
 if (head == NULL) {   
 printf("메모리가 부족합니다!!\n");   
 exit(1);   
 }   
 head->number = i + 1; // 객차 번호는 1부터 시작   
 head->seats = train\_seats[i];   
 }   
 else   
 {   
 tail->next = (Train\*)malloc(sizeof(Train));   
 if (tail->next == NULL) {   
 printf("메모리가 부족합니다!!\n");   
 exit(1);   
 }   
   
 tail = tail -> next; // 빈 칸을 채울 것   
 tail->number = i + 1;   
 tail->seats = train\_seats[i];   
 tail->next = NULL; // 빈 칸을 채울 것   
 }   
 }   
   
 print\_train(head); // 객차 번호와 좌석수 출력   
   
 return 0;   
}   
   
//객차 번호와 좌석수를 출력하는 함수   
void print\_train(Train\* tp)   
{   
 while (tp->next != NULL) {   
 printf("[%d번 객차, 좌석수:%d]-> ",tp->number, tp->seats);   
 tp = tp->next;   
 }   
 printf("[%d번 객차, 좌석수:%d]", tp->number, tp->seats);   
}

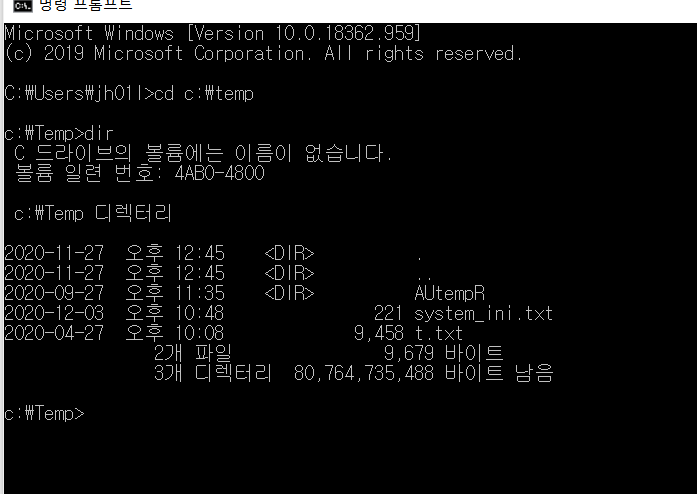


p3

#define \_CRT\_SECURE\_NO\_WARNINGS   
#include <stdio.h>   
int main(void) {   
 FILE\* origin;   
 FILE\* copy; //파일 포인터 선언   
 char ch;   
 int bytes = 0;   
   
 origin = fopen("C:\\Windows\\system.ini", "r"); //대상 파일   
 copy = fopen("C:\\temp\\system\_ini.txt", "w"); //복사 파일   
   
 while (1) {   
 ch = fgetc(origin);   
 if (ch == EOF) //끝을 만나면 빠져나감   
 break;   
   
 fputc(ch, copy);   
 bytes++;   
 }   
   
 fputc('\n', copy);   
   
 fclose(origin);   
 fclose(copy);   
   
 printf("읽은 바이트 수는 %d 입니다!", bytes); //출력 바이트   
 return 0;   
}







206바이트로 읽는 이유:

텍스트 I/O 모드로 읽을 때, get() 은 라인의 끝에 있는 ‘\r\n’의 두 바이트를 ‘\n’의 한 바이트로 리턴한다. c:\windows\system.ini는 총 13 라인의 219 바이트이지만, 실제 읽은 바이트 수는 각 라인의 ‘\r’ 개수 만큼 13개 모자란 206으로 카운트 된다.

p4

#define \_CRT\_SECURE\_NO\_WARNINGS   
#include <stdio.h>   
int main(void) {   
 FILE\* ifp = NULL, \* ofp = NULL;   
 char name[20];   
 int kor, eng, math, programming;   
 int total;   
 int avg;   
 int res;   
 int repeat = 0;   
 ifp = fopen("prob2\_data.txt", "r");   
 if (ifp == NULL) {   
 printf("입력 파일을 받지 못했습니다");   
 return 1;   
 }   
   
 ofp = fopen("prob2\_output.txt", "w");   
 if (ofp == NULL) {   
 printf("출력 파일을 받지 못했습니다");   
 return 1;   
 }   
 fprintf(ofp, " 학번 평균\n");   
 printf(" 학번 평균\n");   
   
 while (1) {   
 res = fscanf(ifp, "%s%d%d%d%d", name, &kor, &eng, &math, &programming);   
   
 if (res == EOF) {   
 break;   
 }   
 total = kor + eng + math + programming;   
 avg = total / 4.0;   
   
 fprintf(ofp, "%s : %5d\n", name, avg);   
   
 printf("%s : %5d\n", name, avg);   
 }   
 fclose(ifp);   
 fclose(ofp);   
   
   
 return 0;   
}

