2주차 결과보고서

전공: 수학/컴퓨터공학 학년: 1학년 학번: 20181294 이름: 임승섭

**1 - 1.**

(fmt 구현하기 위한 알고리즘)

실습시간에 fmt 알고리즘을 구현하기 위해 Header.h, main.c, Output.c, String\_Manipulation.c 파일들을 이용했고 Makefile 을 이용해 main 이라는 실행파일을 컴파일하여 실제로 구현할 수 있었다.

main.c

main.c 파일에서는 다양한 변수와 다른 파일에서 만든 함수를 이용해 알고리즘을 구현한다. 우선 사용할 변수들을 초기화하고, 파일을 입력한다. 이 때 argc를 확인하여 파일 1개를 입력하지 않았으면 에러 처리를 해준다. 사용할 line1과 line2에는 malloc을 이용해 메모리를 동적 할당 해주고 fgets()로 line1에 데이터를 읽어온다. 읽어온 line1을 Remove\_Blanks\_At\_The\_End() 함수를 이용해 마지막 빈칸들을 모두 지워준다.

while문에 들어가서는 우선 B\_line이 0일 때, Print\_Line 함수를 호출한다. 아니라면 B\_line을 0으로 바꿔준다. 이 때 B\_Line은 위의 Remove\_Blanks\_At\_The\_End() 함수 사용 후 그 줄이 비었으면 1, 아니면 0을 나타낸다. 또 Count가 0이 아니라면 B\_Flag을 1로 선언한다. Count에는 현재 줄에서 출력한 글자 수가 저장되어있다. 마찬가지로 lin2를 읽어오고 Remove\_Blanks\_At\_The\_End 함수를 이용해 빈칸을 지워준다. line2의 첫 글자가 빈칸이고, Count가 0이 아니라면 putchar 실행 후 B\_Flag와 Count를 0으로 선언한다. 만약 line2의 첫 글자부터 ‘\n’이 등장하고 B\_Flag까지 1이라면 putchar 이후 B\_Flag를 0으로 선언한다 그렇지 않으면(B\_Flag이 1이 아닌 경우) putchar 이후 B\_Line은 1, Count는 0으로 선언한다. 마지막으로 line1과 line2를 바꿔준다. while문이 끝나고 만약 문장의 마지막 글자가 ‘\n’이 아니라면 ‘\n이나 ‘\0’가 나올 때까지 찾고, ‘\n’이 등장했다면 putchar를 해준다.

String\_Manipulation.c

Remove\_Blanks\_At\_The\_End 함수와 Get\_Blanks\_Chars 함수를 구현한다.

Remove\_Blanks\_At\_The\_End 함수는 line 문자열을 받아 문자열 끝에 있는 빈칸들을 제거한다. 우선 문장을 돌면서 ‘\n’이 등장하면 newline\_flag를 1로 바꿔주고 loop을 중단한다. 그렇지 않고 ‘\0’가 등장하면 바로 loop을 중단한다. 이후 k-1부터 i를 줄여가면서 line[i]가 빈칸이 아니라면 loop을 중단한다. i는 1을 추가해주고 newline\_flag가 1일 때는 줄바꿈&빈칸, 1이 아닐 때는 ‘\0’로 문장의 마지막을 바꿔준다.

Get\_Blanks\_Chars 함수는 line 문자열의 빈칸과 문자의 개수를 세준다. for loop을 돌면서 ‘\n’이나 ‘\0’를 만나면 바로 중단하고 빈칸을 만나면 blank\_flag의 값에 따라 N\_Blanks 값을 추가하거나 loop을 중단한다. 만약 blank\_flag가 1일 땐 N\_Chars 값을 추가한다.

Output.c

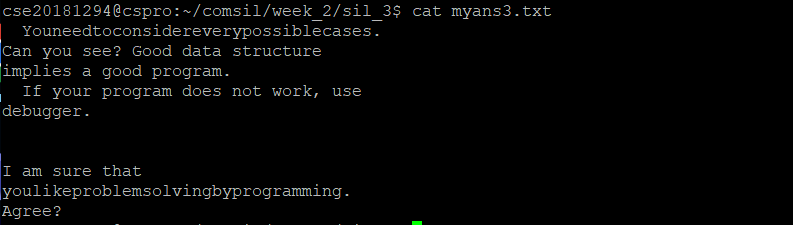
Print\_Line 함수를 구현한다. Count와 B\_flag를 이용하여 원하는 형태로 문자열을 출력해준다. 이 때, Count는 출력할 글자 수, B\_Flag은 줄바꿈을 하는 판단을 내린다.

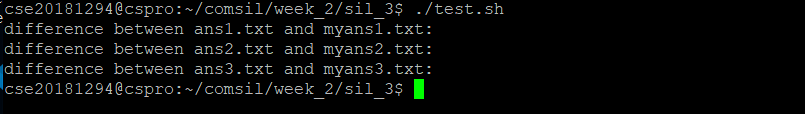
**1 - 2.**

(출력 결과)



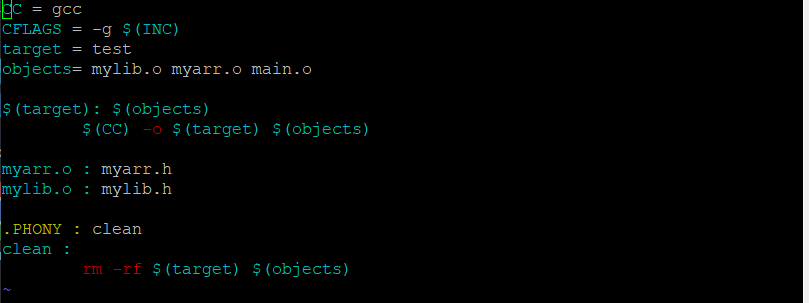
텍스트이(가) 표시된 사진

자동 생성된 설명



**2 - 1.**

(Makefile)



object파일 mylib.o, myarr.o, main.o 를 이용하여 target test를 생성하였고 각각의 필요한 헤더파일 myarr.h, mylib.h를 지정해주었다.

**2 - 2.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**gdb안에 들어간 후 l(line) 을 이용하여 파일의 10번째 줄까지의 내용을 확인하였다.

7번째 줄에 b(break)을 걸고 run을 했다.

step을 이용해 getSqurare 함수 내로 들어갔다. 이 때 next를 쓰지 않은 이유는 next는 함수 안으로 들어가지 않기 때문이다.

result 값 계산이 완료된 후 print 를 이용해 값을 출력해보았다. 여기서 값이 잘못되었음을 확인하였고, 함수 getSquare에서 실수가 있었음을 알 수 있었다

**2 - 3.**

input에 대한 myans 출력 화면은 **1 - 2.** 출력 결과에 첨부해 두었기 때문에 생략하겠습니다.

**3.**

make 옵션은 다음과 같다.

-c dir : Makefile 실행 전 dir로 이동한다.

-p : 내부적으로 세팅되는 값을 출력한다.

-k : 에러가 나더라도 계속 실행한다.

-v : make의 버전을 출력한다.

-t : 파일의 생성날짜를 현재 시간으로 생성한다.

-f file : file 인자에 해당하는 파일을 Makefile로 취급한다

-h : 옵션의 도움말을 출력한다.

-d : Makefile을 수행하면서 정보를 출력한다. 정보들을 확인할 수 있다.

**4.**

(과제 코드)

main.c

#include <stdio.h>

#include <stdlib.h>

#include "main.h" // main.h 불러옴

int main() {

    int Test; // 실행 횟수

    scanf("%d", &Test); // 실행 횟수를 입력받는다

    int \*k1 = (int\*)malloc(sizeof(int) \* Test); // 실행 횟수의 크기로 배열을 만든다. 페이지수를 저장할 배열.

    int \*\*k2 = (int\*\*)malloc(sizeof(int\*) \* Test); // 각 페이지수마다 0~9의 개수를 저장할 배열을 만든다.

    for (int i = 0; i < Test; i++) {

        k2[i] = (int\*)malloc(sizeof(int) \* 10); // 0~9까지, 총 10개의 int를 저장할 크기로 받는다.

    }

    for (int i = 0; i< Test; i++) {

        scanf("%d", &k1[i]); // k1 배열에 페이지수를 저장한다

    }

    for (int i = 0; i< Test; i++) {

        cal\_func(k1[i], k2[i]); // cal\_func을 실행시켜서 k1의 페이지수마다 숫자의 개수를 구하여 k2에 저장한다

    }

    pnt\_func(k2, Test); // pnt\_func을 실행시켜서 k2의 데이터를 출력한다.

    // malloc을 이용해 동적할당을 받았던 메모리를 해제한다.

    for (int i = 0; i< Test; i++) {

        free(k2[i]);

    }

    free(k1);

    free(k2);

    return 0;

}

cal\_func.c

#include "main.h" // main.h 불러옴

void cal\_func(int num, int \*arr) { // num은 페이지 수, arr은 0~9 숫자의 개수를 저장할 배열

    // a는 1쪽, b는 마지막 쪽을 나타낸다.

    int a = 1;

    int b = num;

    // 처음은 1의 자릿수부터 시작하고 10씩 곱해가면서 끝자리의 위치를 변경한다.

    int place = 1;

    while( a <= b) {

        // a를 10이 되게 한다. (a는 항상 1로 초기화되기 때문에 1씩 추가하면서 10으로 만든다.)

        while (a < 10 && a <= b) {

            arr[a] += place; // a가 커지는 구간의 숫자들은 skip하기 때문에 이 부분에서 arr에 추가한다.

            a++;

        }

        if (a > b) // a가 b보다 커졌다면 함수를 종료한다.

            break;

        // b의 끝자리가 9가 되도록 줄인다. (10으로 나눈 나머지가 9가 될 때까지 줄인다.)

        while (b %10 != 9 && a <= b) {

            // b가 줄어드는 구간의 수의 숫자 개수를 세는 과정

            int b\_copy = b; //b와 같은 값을 갖는 b\_copy를 만든다.

            while( b\_copy > 0) {

                arr[b\_copy%10] += place; // 일의 자리 수부터 arr에 추가하고

                b\_copy = b\_copy/10;      // 10으로 나눠가면서 각 자리 수를 모두 확인한다.

            }

            b--;

        }

        a = a/10; // a를 10으로 나눈 몫, 1로 만든다.

        b = b/10; // b를 10으로 나눈 몫으로 만든다.

        /\* 10으로 나누기 전, a부터 b까지 0~9 숫자가 각각 (b/10 - a/10 + 1) \* place 개가 존재한다.

        각각을 이미 10으로 나누었기 실제로 계산할 때는 (b - a + 1) \* place로 할 수 있어서 간편하다

        이 때, a는 무조건 1의 값을 가지기 때문에 b \* place 로 더 간단하게 계산한다.\*/

        // 0부터 9까지 b \* place만큼의 개수를 추가해준다.

        for (int i = 0; i<10; i++) {

            arr[i] += b \* place;

        }

        place = place \* 10;  // 마지막으로 place에 10을 곱해줌으로써 한 자리씩 올린 다음 loop을 진행한다.

    }

}

pnt\_func.c

#include "main.h" // main.h 불러옴.

void pnt\_func(int\*\* k, int T) {

    for (int i = 0; i < T; i++) { // 실행 횟수 T 만큼,

        for (int j = 0; j< 10; j++) { // 0 ~ 9, 총 10 만큼,

            printf("%d ", k[i][j]); // 배열에 저장된 데이터를 출력한다.

        }

        printf("\n"); // 다른 페이지수를 구분하기 위한 줄바꿈.

    }

}

main.h

#include <stdio.h>

#include <stdlib.h>

void cal\_func(int, int\*); // cal\_func 함수 선언

void pnt\_func(int\*\*, int); // pnt\_func 함수 선언

Makefile

CC = gcc

target = main

objects = main.o cal\_func.o pnt\_func.o

$(target) : $(objects)

    $(CC) -o $(target) $(objects)

$(objects) : main.h

./PHONY : clean

clean :

    rm $(target) $(objects)