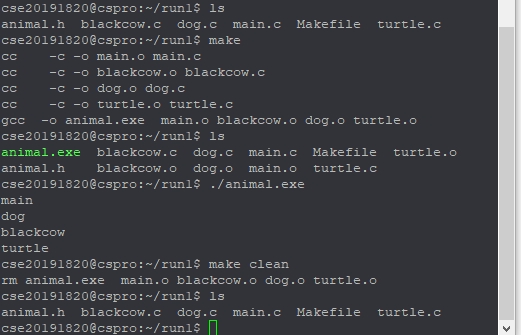
2주차 결과보고서

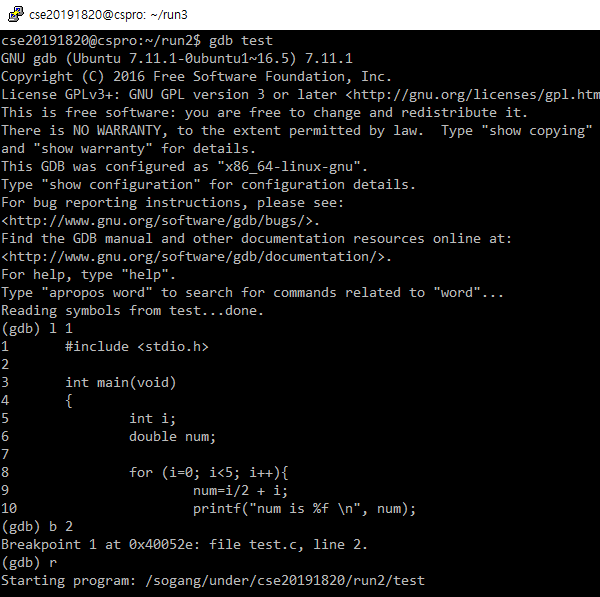
전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

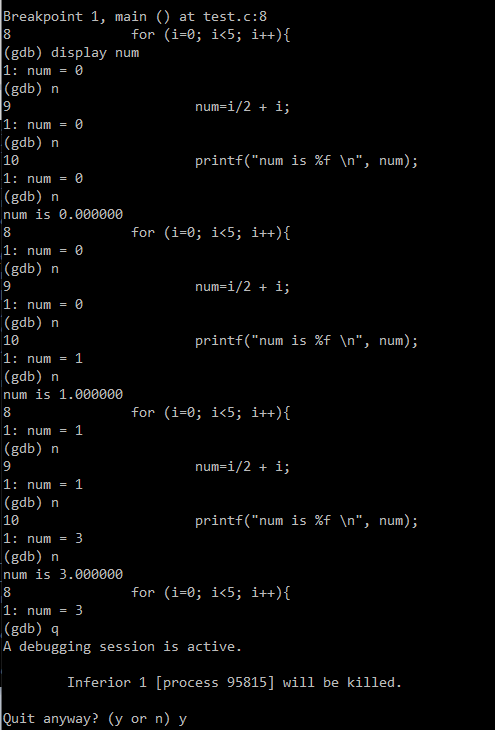
**1. 실습 결과 화면**

**1-1. 실습1 화면**

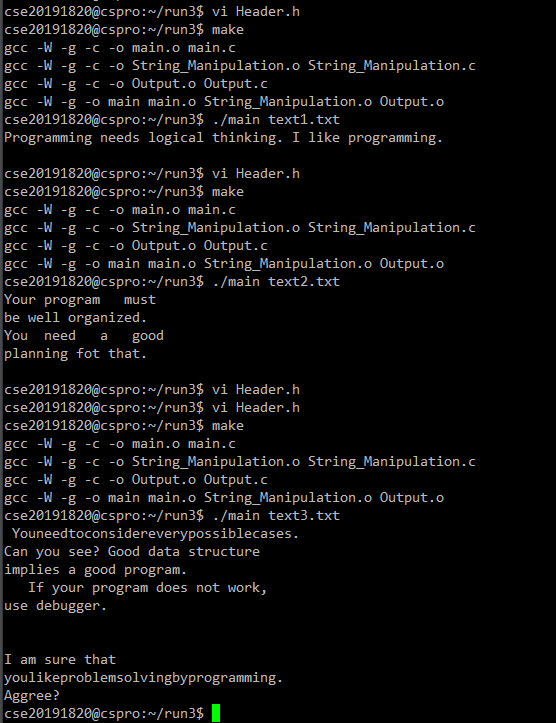
****

**1-2. 실습2 화면**

****

****

**1-3. 실습3 화면**

****

**2. fmt의 함수들과 목적**

int main(int argc, char \*argv[]) : 프로그램의 main함수, 텍스트 파일을 입력받고

전체 프로그램의 흐름을 제어한다.

void Remove\_Blanks\_At\_The\_End(char \*line)

: 전처리용 함수, 문자열 마지막에 빈 칸(‘ ‘)들을 삭제한다.

void Get\_Blanks\_Chars(char \*line, int Start, int \*N\_Blanks, int \*N\_Chars)

: 문자열에서 Start부터 빈 칸(‘ ‘)과 문자의 개수를 세는 함수

void Print\_Line(char \*line, int \*Count, int \*B\_Flag)

: 입력된 문자열을 조건에 맞게 출력하는 함수

**3. 실습3 Makefile**

# 변수에 값을 대입함 (매크로 정의)

cc=gcc # 컴파일러 종류: gcc로 컴파일

cflags = -W –g

# 컴파일러 플래그, (-W) : 모호한 코드에 대해 경고메세지 출력, (-g) : 디버깅 가능

target = main # 생성될 실행파일명: main으로 생성함

objects = main.o String\_Manipulation.o Output.o # 생성될 오브젝트 파일명

$(target) : $(objects) # 룰1 : 오브젝트 파일을 사용해서 실행파일을 생성함

$(cc) $(cflags) -o $(target) $(objects) # 명령 : gcc를 사용해 오브젝트 파일로 실행파일을 생성한다.

# 룰2 : 현재 디렉토리에 있는 모든 .c(소스코드)파일을 컴파일해 각각의 .o(오브젝트)파일을 생성

%.o : %.c

$(cc) $(cflags) -c -o $@ $< # 실행파일을 만들지 않고, 오브젝트 파일만 생성함

$(objects): Header.h # 룰3 : 헤더파일이 변경될 경우 오브젝트파일을 재생성함

.PHONY : clean # 파일명 충돌방지

clean : # 룰4 : make clean으로 오브젝트파일과 실행파일을 삭제함

rm $(target) $(objects) # rm 명령어로 생성된 오브젝트파일과 실행파일을 삭제

**4. 규칙 R5의 알고리즘**

Main 함수에 규칙 R5를 구현하기 위한 코드가 위치해있다.

if ( line2[0] == ' ' && Count != 0) {

putchar('\n');

B\_Flag = 0;

Count = 0;

}

L2의 첫 글자가 blank(‘ ‘)이고, Count가 0이 아니면 \n을 출력하고, B\_Flag와 Count에 모두 0을 저장함 (\n을 출력했으므로 앞줄과 합쳐지지 않고 다음 줄에 출력됨)

tmpline = line1;

line1 = line2;

line2 = tmpline;

tmpline을 사용해 L1과 L2의 값(포인터의 값)을 서로 교환함

이후 B\_Line이 0이 아니면

Print\_Line(line1, &Count, &B\_Flag);으로 L1의 값을 출력함 (첫부분의 blank 개수 그대로 새 줄에 출력됨)

**5. make의 옵션**

-C dir : Makefile을 읽기 전에 dir 디렉토리로 변경함

-d : Makefile 수행중 디버깅 정보를 추가로 출력

-f file : file에 해당하는 파일을 Makefile로 지정

-h : 옵션에 관한 도움말 출력

-i : 파일생성시 발생되는 모든 에러를 무시함

-k : 에러 발생시 무시하고 계속 진행시킴

-n : 실행될 명령을 출력 (해당 명령을 실행하지는 않음)

-s : 명령 실행시 해당 명령을 출력하지 않음

-t : 파일의 생성날짜를 현재시간으로 갱신

-v : make의 버전을 출력

**6. 알고리즘**

- BNUM과 LIMIT는 Header.h에 매크로로 정의되어있음 (각각 256, 72)

int main(int argc, char \*argv[]) : 프로그램의 main함수, 텍스트 파일을 입력받는다.

- 변수 초기화 및 예외처리를 실행함 (0으로 초기화, 파일명 미입력시 사용예 출력,

파일을 열 수 없는 경우 에러 출력, 문자열 저장용 메모리 할당 실패시 에러 출력)

- (규칙 R9의 처리) 크기가 0인 파일인 경우 아무것도 출력하지 않고 프로그램을 종료함

- L1을 전처리 하기 위해 Remove\_Blanks\_At\_The\_End()을 호출함

- L1의 첫 글자가 \n일경우 \n을 출력함

- Count에 0을 저장함

- 무한 반복 루프문 (break로 탈출할 때까지 반복) {

- B\_Line이 0인 경우 (‘L1이 빈 줄이라 이미 \n이 출력된 경우’가 아닌 경우)

Print\_Line()을 호출해 L1을 출력함, B\_Line이 1인 경우 B\_Line을 0으로 변경함

- Count가 0이 아니면 B\_Flag에 1을 저장함

- fgets로 파일에서 한 줄을 L2에 저장함, 이때 L2가 EOF(파일의 끝)면 루프를 종료함

- L2를 전처리 하기 위해 Remove\_Blanks\_At\_The\_End()을 호출함

- (규칙 R5의 처리) L2의 첫 글자가 blank(‘ ‘)이고, Count가 0이 아니면 \n을 출력하고, B\_Flag와 Count에 모두 0을 저장함

위 조건이 아니고, L2의 첫 글자가 \n일 경우 \n을 출력하고 B\_Line에 1을, Count에 0을 저장함 (B\_Flag가 1일 경우 \n 출력 이전에 \n을 출력하고, B\_Flag에 0을 저장함)

- tmpline을 사용해 L1과 L2의 값(포인터의 값)을 서로 교환함

} - 무한 루프문 종료

- L1의 첫 글자가 \n이 아닐 경우 L1을 출력함 (\n이 있는 경우 \n까지 포함해서 출력)

- 프로그램 종료

void Remove\_Blanks\_At\_The\_End(char \*line) : 전처리용 함수, 문자열 마지막에 빈 칸(‘ ‘)들을 삭제한다.

- (변수 초기화) newline\_flag에 0을 저장한다.

- 문자열의 맨 끝 위치(Index)를 k에 저장한다. 이때, 문자열이 \n으로 끝나면

newline\_flag에 1을 저장한다.

- 문자열의 마지막에 존재하는 빈 칸(‘ ‘)의 시작 위치를 찾아서 i에 저장한다.

i에는 최종적으로 전처리 후 \n(\n으로 끝나는 경우) 또는 \0(\n이 없는 경우)이 위치할 위치(Index)가 저장된다.

- newline\_flag가 1인 경우 (문자열이 \n으로 끝나는 경우) line[i]에 ‘\n’를, line[i+1]에 ‘\0’을 저장한다. newline\_flag가 0인 경우 (문자열에 \n이 없는 경우) line[i]에 ‘\0’를 저장한다.

최종적으로 문자열은 ‘문자열+빈칸+(\n)+\0’에서 ‘문자열+(\n)+\0’으로 변경된다.

void Get\_Blanks\_Chars(char \*line, int Start, int \*N\_Blanks, int \*N\_Chars)

: 문자열에서 Start부터 빈 칸(‘ ‘)과 문자의 개수를 세는 함수

- (변수 초기화) blank\_flag, \*N\_Blanks, \*N\_Chars에 0을 저장한다.

- 무한 반복 루프문 (i = Start로 시작, i를 1씩 증가시킴, break로 탈출할 때 까지 반복) {

- line[i]가 \n이나 \0이면 루프를 종료함

- 위 조건이 아니고, line[i]가 빈 칸(‘ ‘)이면 blank\_flag가 0인 경우 \*N\_Blanks의 값을 1 증가시키고, blank\_flag가 0이 아닌 경우 루프를 종료함

- line[i]가 \n, \0, 빈칸(‘ ‘)이 아닐 경우 blank\_flag에 1을 저장하고, \*N\_Chars의 값을 1 증가시킴

} - 무한 루프문 종료

최종적으로 \*N\_Blanks에는 문자열의 Start(line[Start]) 이후의 빈 칸(‘ ‘)의 길이가 저장되고, \*N\_Chars에는 문자열의 Start(line[Start]) 바로 뒤에 있는 단어의 길이(빈칸 길이 미포함)가 저장된다.

void Print\_Line(char \*line, int \*Count, int \*B\_Flag) : 입력된 문자열을 조건에 맞게 출력하는 함수

- (변수초기화) Start에 0을 저장한다.

- Get\_Blanks\_Chars()를 호출해 N\_Blanks, N\_Chars에 각각 첫 빈 칸의 길이와 첫 단어의 길이를 저장한다.

- 루프문 (N\_Chars가 0이면 탈출함) {

- \*B\_Flag가 1이고, (\*Count + N\_Chars + 1) <= LIMIT인 경우, (규칙 R2의 처리)

- N\_Blanks가 0이 아닌 경우 오류 출력후 함수를 종료함

- 빈 칸(‘ ‘)을 1개 출력하고, \*B\_Flag에 0을 저장함

- 문자열의 현재 단어를 출력함 (빈 칸이 없으므로 i=Start~Start+NChars까지

출력)

- Start에 Start+N\_Chars를 저장한다.

- \*Count에 \*Count+N\_Chars+1을 저장한다. (빈 칸 1개를 추가로 출력했으므로 +1이 추가됨)

- Get\_Blanks\_Chars()를 호출해 N\_Blanks, N\_Chars의 값을 문자열의 다음 단어로 갱신한다.

- (\*Count + N\_Blanks + N\_Chars) <= LIMIT 인 경우,

- 문자열의 빈 칸을 포함한 현재 단어를 출력함

(i=Start~Start+N\_Blanks+NChars까지 출력)

- Start에 Start+N\_Blanks+N\_Chars를 저장한다.

- \*Count에 \*Count+ N\_Blanks+N\_Chars을 저장한다.

- Get\_Blanks\_Chars()를 호출해 N\_Blanks, N\_Chars의 값을 문자열의 다음 단어로 갱신한다.

- 위 두 조건이 아닌 경우 (else)

- \*Count가 0인 경우 (규칙 R6의 처리)

- 문자열의 빈 칸을 포함한 현재 단어를 출력함 (i=Start~Start+N\_Blanks

+NChars까지 출력)

- Start에 Start+N\_Blanks+N\_Chars를 저장하고, \n을 출력함

- Get\_Blanks\_Chars()를 호출해 N\_Blanks, N\_Chars의 값을 문자열의

다음 단어로 갱신한다.

- Start에 Start+N\_Blanks를 저장함

- \*Count가 0이 아닌 경우 (규칙 R4의 처리)

- \n을 출력하고, \*B\_Flag에 0을 저장함

- 문자열의 빈 칸을 포함한 현재 단어를 출력함

(i=Start~Start+N\_Blanks+NChars까지 출력)

- Start에 Start+N\_Blanks+N\_Chars를, \*Count에 N\_Chars를 저장함

- Get\_Blanks\_Chars()를 호출해 N\_Blanks, N\_Chars의 값을 문자열의

다음 단어로 갱신한다.

} - 루프문 종료

최종적으로 line에 있는 모든 단어가 규칙(R1~R9)에 맞게 출력된다.

**7. 테스트**

