2주차 결과보고서

전공: 수학과,컴퓨터공학과 학년: 3학년 학번: 20181256 이름: 김도현

2-1.

Ex1.

(LIMIT=72)

텍스트이(가) 표시된 사진

자동 생성된 설명

마지막 줄의 마지막 단어 뒤에 blank가 없음을 확인할 수 있다.(R1)

우리가 가지고 있는 Header.h 파일에서 LIMIT=72 이기 때문에 문장이 이어져 있는걸 확인 할 수 있다. 그리고 ‘thinking’ 다음의 한개 이상의 blank 들은 다 무시하고 한개의 blank 만 나온다.(R2)

마지막 줄에 ‘\n’ 즉 엔터키 가 있을 경우 그대로 출력함을 알 수 있다.(R8)

Ex2.

(LIMIT=72)

텍스트이(가) 표시된 사진

자동 생성된 설명

‘must’ 와 ‘be’ 사이의 blank 들은 줄이 바뀌지 않았기 때문에 그래도 출력됨을 확인 할 수 있다.(R3)

‘for’ 와 ‘that’ 을 보면 LIMIT=72 를 넘어갔기 때문에 같은 문장이라도 ‘for’ 다음 ‘that’ 을 다음 줄로 바꾸고 두 단어 사이의 blank 들은 무시한다. (R4)

마지막 줄에 ‘\n’ 즉 엔터키 가 있을 경우 그대로 출력함을 알 수 있다.(R8)

(LIMIT=20)

텍스트이(가) 표시된 사진

자동 생성된 설명

같은 문장에서 LIMIT을 36으로 바꾸면 ‘must’ ‘be’ 와 같이 다음 줄로 바뀌고 다 단어 사이의 blank는 무시한다. (R4)

Ex3.

(LIMIT=72)

텍스트이(가) 표시된 사진

자동 생성된 설명

첫번째 ‘You…’ 앞에 blank가 존재하는데 이때는 앞 줄과 합쳐지지 않고 만약 blank가 처음에 여러 개 있어도 이때는 새 줄에 blank 개수만큼 그대로 출력되어야 한다. 만족 함을 알 수 있다.(R5)

빈 줄이 있으면 그대로 나옴을 알 수 있고 빈 줄에 blank 가 있으면 이는 출력되지 않는다.(R7)

마지막 줄 ‘Aggree?’ 다음 ‘\n’ 이 아닌 ‘\0’가 있을 경우에는 ‘\n’ 없이 그대로 출력한다.(R8)

(LIMIT=36)

텍스트이(가) 표시된 사진

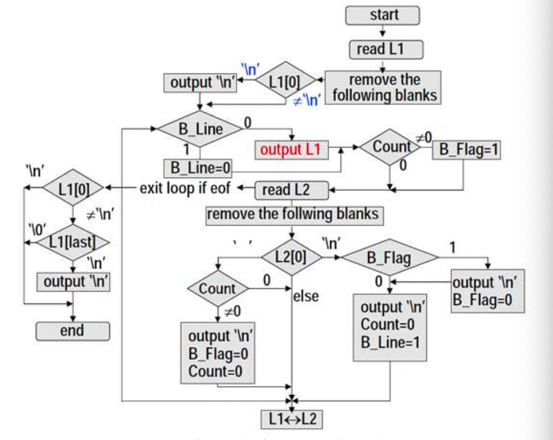
자동 생성된 설명

R6 같은 경우 LIMIT=36 일 때 ‘Youneedtoconsidereverypossiblecases’ 와 같은 단어는 한 단어가 LIMIT 를 초과하므로 그냥 한줄에 출력함을 알 수 있다.

R9 같은 경우 빈 파일(empty) 일 때는 아무것도 안 나오는 것을 알 수 있다.(실습)

2-2.

fmt 구현 프로그램 흐름도



우리는 fmt를 구현하기 위해서 main.c 파일, Header.h 파일, String\_Manipulation.c 파일, Output.c 파일을 이용하여 Makefile 을 통해 프로그램을 컴파일 했다.

Header.h 에서는 쓰일 함수들과 #include, #define 을 선언한다.

main.c 에서는 이 프로그램에서 쓰일 변수들을 정의하고 초기화한다.

이후 argc(명령인수 개수) 가 2가 아니면(파일이 1개 있어야 정상출력) 에러 메세지를 출력하고 파일이 있어도 읽었을 때 실패 시 에러메세지를 출력한다. 이후 line1과 line2에 malloc을 통해(char \*) 동적 할당 메모리를 지정하고 실패 시 에러메세지를 출력한다.

fgets() 을 통해 line1을 가져오고 이때 이 길이가 0이면 파일이 비었다는 뜻으로 return 0을 한다. 그리고 Remove\_Blanks\_At\_The\_End(line1) 으로 line1 의 마지막 blank 들을 지워준다. 만약 line1의 첫 글자가 ‘\n’ 이면 putchar(‘\n’)를 한다.

while문에서 다음 과정을 반복 실행한다. B\_Line(공백 지운 이후 그 줄이 비었으면 1, 안 비었으면 0) 이 0 이면 Print\_Line(line1, &Count, &B\_flag)을 하고 아니면 B\_Line 을 0 으로 선언한다. 그리고 Count(지금 줄에서 실제 출력한 글자 수 저장) 가 0이 아니면 B\_flag(Count 가 0이 아니면 1 즉 같은 줄에 계속 출력 가능, 다음 줄이 비었거나 첫 단어가 blank 는 0, 줄바꿈 수행을 해야한다는 의미) 를 1로 선언한다. fgets() 를 통해 line2를 가져오는데 EOF가 나오면 break 를 통해 while문을 빠져 나온다. 그리고 Remove\_Blanks\_At\_The\_End(line2) 로 line2 의 마지막 blank 들이 지워진다. line2의 첫 글자가 blank 이고 Count 가 0 이 아니면 putchar(‘\n’)를 하고 B\_Flag 과 Count를 0 으로 선언한다. 그렇지 않고 line2의 첫 글자가 ‘\n’이고 B\_flag가 1 이면 putchar(‘\n’)을 하고 B\_flag 를 0으로 선언한다. 아니면 putchar(‘\n’), B\_line 를 1, Count 를 0 으로 선언한다. 그리고 tmpline을 통해 line1 과 line2를 서로 바꾼다.

while문 후에 마지막 줄이 빈 줄이면 출력되지 않는 결과가 나오는데 만약 line1 의 첫 글자가 ‘\n’ 이 아니면 line1[i] 의 글자가 ‘\n’ 이나 ‘\0’ 일때 까지 찾고 line1[i] 가 ‘\n’ 이면 putchar(‘\n’)을 하는 과정을 추가한다.

String\_Manipulation.c

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

이 함수는 line 이라는 문자열을 받아 for 문을 돌리면서 문자열 끝에 blank들이 있을 때 이 blank 들을 제거하는 역할 을 한다.

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

이 함수는 line 의 blank 와 문자열들을 세는 함수인데 문장이 아닌 단어 위주로 확인하고 단어 앞의 blank 수와 단어의 개수를 통해 나중에 Print\_Line에서 같은 줄에 써야할지 새로운 줄에 써야할지 도움을 준다.

Output.c

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

이 함수는 line 을 Count(실제 출력할 글자 수) 와 B\_Flag(Count=0 이 아니면 계속 출력할수 있도록 도와줌, 줄 바꿈 수행에 도움을 줌) 를 가지고 우리가 원하는 형태(R1,~R9) 로 문자열들을 출력할 수 있다.

2-3.

실습시간에 작성한 Makefile은 다음과 같다.

Variable 을 선언해서 Makefile을 간단하게 하는 실습을 진행하였다.

make 입력 시 Makefile이 실행된다.

cc = gcc //gcc를 이용하여 컴파일(코드에서 ‘gcc’ 대신 ‘cc’ 를 사용, 매크로) 하겠다는 의미이다.

target = animal\_exe //target 파일 즉 실행 파일 이름을 animal\_exe 로 선언했다.

objects = main.o dog.o blackcow.o turtle.o //컴파일을 하기 위한 .o 파일들의 모음이다.

$(target) : $(objects) // target 을 위해 필요한 prerequisites 를 objects 로 두었다.

$(cc) -o $(target) $(objects)

//gcc -o 실행파일 소스파일 형식을 따랐다. 명령어 부분은 tab 시작

//이때 .c 파일이 없어도 implicit rule 을 적용하여 자동으로 .c 파일을 찾아 gcc -c를 수행한다.

$(objects) : animal.h //헤더 파일이 변경 시 object 들도 다시 만들 수 있게 해준다.

.PHONY : clean // recipe 의 대표 이름이여서 .PHONY 를 통해 돌발 상황을 막을 수 있다.

clean : //make clean 을 하면 target 파일과 objects 파일 모두 지울 수 있다.

rm $(target) $(objects)

2-4.

R5 는 입력 줄의 첫 글자가 blank 면 앞줄과 따로 써야 하고 여러 개의 blank 가 있다면 새로운 줄에서 그 개수의 blank 만큼 출력하는 것이다.

main 함수 중에서…

If (line2[0] == ‘ ‘ && Count != 0)//line2의 첫 글자가 blank 이고 Count 가 0이 아니면

{

Putchar(‘\n’); //줄바꿈을 한다.

B\_Flag=0;

Count=0;

}

비록 Count 가 0이 아니기 때문에 계속 줄을 더 이을 수 있지만 첫 글자가 blank 이기 때문에 다음 줄로 넘기고 B\_Flag 와 Count 를 0으로 바꿔 다음 줄에서 출력할 수 있게 한다. 그리고 만약 Count 가 0이면 그대로 수행시키면 된다.

이후 Print\_Line( char \*line, int \*Count, int \*B\_Flag) 에서 출력 시 처음 시작하는 공백들도 포함 시키면 된다.

2-5.

make에서 필요한 것은 대부분 Makefile 에서 해결 할 수 있지만 make 입력 할 때 옵션을 줄 수도 있다. 옵션들은 다음과 같다.

-f file : file을 Makefile로써 간주한다.

-d : Makefile의 수행에서 나오는 모든 정보들을 확인 할 수 있다.

-h : 옵션의 help(도움말)를 출력할 수 있다.

-t : 파일을 만든 날짜를 지금 시간으로 업데이트한다.

-v : make 버전을 확인할 수 있다.

-p : make 내부 세팅값들을 확인 할 수 있다.

-C dir : Makefile을 실행하기 전에 우선 dir로 먼저 이동한다.

-k : Makefile 에서 수행 도중 에러가 나와도 멈추지 않고 계속 실행시킨다.

-S : -k 의 반대 과정으로 에러가 나오면 멈추게 한다.

-r : 내장된 규칙을 생각하지 않고 프로그래머의 새로운 규칙을 따른다.

-I dir : Makefile 들이 어디에 있는지 알기위해 dir 을 통해 구체화한다.

-n : 수행될 명령어들을 출력만 한다.