2주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20211522 이름: 김정환

**1.**

.................

실습 1

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

실습 2

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

실습 3

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

................

**2.**

.......................

main.c

main.c에서는 우선 입력받은 인자로부터 파일 이름을 받아서 line1과 line2에 파일의 한 줄 씩 입력받는다. 이 후에 line1에 string\_manipulation.c에 있는 함수인 Remove\_Blanks\_At\_The\_End 함수를 이용하여 줄 뒷부분의 공백을 제거하고 줄바꿈만 있을 경우에만 구분하여 출력해주고 이후에는 while 반복문 안에서 문장 끝까지 line1은 상황에 맞는 출력을 해주고 line2에 Remove\_Blanks\_At\_The\_End 함수 등을 이용하여 처리해준 후에 line1과 swap 해주고 다음 반복 때 line2에 새로운 줄을 입력받는다.

string\_manipulation.c

string\_manipulation.c에는 두 개의 함수가 구현되어 있다. 첫 번째로 Remove\_Blanks\_At\_The\_End 함수이다. 해당 함수는 line 문자 배열을 인자로 받는다. 먼저 줄 끝까지 탐색하면서 개행문자가 있는지 여부를 newline\_flag를 통해 나타낸다. 개행문자 존재 시에는 newline\_flag 값을 1로 나타낸다. 그 후 개행문자 전에 공백이 있는지 검사한다. 공백이 나오지 않을 때까지 반복하고 해당위치의 바로 뒤에 개행문자가 존재했을 때는 개행문자와 ‘\0’을, 개행문자가 없었을 때는 ‘\0’만 추가한다. 다음으로는 Get\_Blanks\_Chars 함수가 있다. 해당 함수는 line 문자 배열과 Start 정수값, 그리고 N\_Blanks, N\_Chars 정수 포인터값을 인자로 받는다. 그 후 Start 인덱스 값부터 개행문자나 ‘\0’값이 나오기 전까지 line 문자열을 탐색하면서 N\_Blanks에는 공백문자의 개수를, N\_Chars에는 이외 문자의 개수를 세고 두 번째 공백이 나오면 탐색을 중지한다.

output.c

output.c에는 출력을 담당하고 있는 Print\_Line 함수가 있다. 이 함수는 string\_manipulation.c의 Get\_Blanks\_Chars 함수를 이용하여 해당 줄을 조건에 맞게 출력한다. 출력에서 if문을 통해 구현된 조건은 첫 번째로 이미 출력한 문자 수가 0이 아닐 때 이미 Count된 수와 문자 수인 N\_Chars에 1을 더한 값이 LIMIT 값을 넘지 않을 때이다. 이는 출력해도 LIMIT 값을 넘지 않는다는 것이므로 그대로 출력한다. 그 후에 Get\_Blanks\_Chars 함수로 공백과 문자 개수를 탐색한다. 그 다음은 아직 해당 줄에 아무것도 출력되지 않고 출력해야할 문자열이 LIMIT을 넘지 않을 때이다. 이 때는 Start 지점의 위치부터 Start + N\_Blanks + N\_Chars까지 출력한다. 그 후 그 다음부터 공백과 문자의 개수를 Get\_Blanks\_Chars 함수로 탐색한다. 그 다음 값들부터는 LIMIT 값을 초과하는 경우이므로 Count값이 0일 때는 마지막 줄을 분리하지 않고 출력하고, 0이 아니라면 LIMIT이 넘지 않도록 개행문자를 출력하고 문자열을 출력한다.

.........................

**3.**

.......................

cc=gcc : 컴파일러로 gcc 컴파일러 사용

cflags = -W -g : gcc에 사용할 옵션 변수로 선언

target = main : 만들어질 실행파일 이름 main으로 하는 변수 선언

objects = main.o string\_manipulation.o output.o : .o 확장자 파일로 만들어질 파일들 이름 목록 변수로 선언

$(target) : $(objects)

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

: objects 변수에 저장한 오브젝트 파일들이 있으면 cflag에 저장해둔 옵션으로 gcc 컴파일러가 target에 저장된 이름의 실행파일 생성하도록 한다.

%.o : %.c

$(cc) $(cflags) -c -o $@ $<

: gcc 컴파일러가 .c 파일로 오브젝트 파일을 생성하도록 하는 명령어이다.

main.o string\_manipulation.o output.o : Header.h : 해당 오브젝트 파일에 Header.h 헤더파일에 대한 의존성 지정한다.

.PHONY : clean

clean :

rm $(target) $(objects)

: PHONY를 통해서 파일 이름 중에 clean을 가진 것과 같이 예상치 못한 상황을 방지하며 rm 명령어를 실행하여 target과 objects에 들어있는 이름의 파일을 지우도록 한다.

.........................

**4.**

.......................

R5 조건은 입력된 줄의 첫 글자가 공백이면 앞줄과 합쳐지지 않고 줄을 바꾸도록 하는 조건이다. 이 조건을 구현한 부분은 main.c에서 확인할 수 있고, main.c에서 while문 안에 line2[0] == ‘ ‘와 Count가 0이 아닌 경우에 개행문자를 출력한 뒤에 B\_Flag = 0과 Count를 0으로 하는 것을 확인할 수 있다. 여기서 Count를 0으로 함으로써 새로운 줄에서 출력하는 것임을 확인 가능하다.

.........................

**5.**

.......................

-C : Makefile 읽는 것이 아닌 디렉터리로 이동한다.

-d : Makefile 실행하면서 디버깅 정보를 출력한다.

-h : make의 옵션에 대해 화면에 출력한다.

-f : 파일을 Makefile로 읽는다.

-i : 레시피에서 발생하는 에러를 무시한다.

-t : 파일의 생성날짜를 현재 시간으로 생성한다.

-v : make의 버전을 출력한다.

-b, -m : 호환성을 무시한다.

-e : 환경변수를 override 한다.

-k : 몇몇 target이 만들어질 수 없어도 계속 실행한다.

-p : 내부적으로 세팅되는 값을 출력한다.

-s : 레시피 출력하지 않는다.

-w : 현재 디렉터리를 출력한다.

.........................