2주차 결과보고서

전공: 아트&테크놀로지학과 학년: 4학년 학번: 20191048 이름: 김도솔

**1.**

텍스트이(가) 표시된 사진

자동 생성된 설명

**2.**

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

: 문자열 끝의 blank들을 제거하기 위해 사용된다.

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

: Start 이후 존재하는 blank 글자의 개수(N\_Blanks에 저장)와 다음 단어의 글자 수(N\_Chars에 저장)를 세기 위해 사용된다.

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

: 문자열 출력 함수, 문자열을 단어별로 구분하여 이들을 한번에 출력한다.

int main(int argc, char \*argv[])

: 메인 함수, 프로그램 실행을 위한 시작 지점 역할을 하며 전반적인 프로그램 실행을 제어한다.

**3.**

cc=gcc

-> gcc를 이용하여 컴파일

cflags = -W -g

-> 컴파일 옵션 설정(-W: 합법적이지만 모호한 코딩에 대한 경고를 보냄, -g: 디버깅을 위한 정보를 포함하여 컴파일)

target = main

-> 실행 파일 이름 설정

objects = main.o string\_manipulation.o output.o

-> 오브젝트 파일 목록 나열

$(target) : $(objects)

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

-> 실행 파일로 컴파일

%.o : %.c

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

-> 오브젝트 파일 생성

main.o string\_manipulation.o output.o : Header.h

-> 헤더 파일이 변경되어도 오브젝트 파일을 새로 생성할 수 있게 함

.PHONY : clean

-> 파일 이름과 clean 명령이 겹치지 않게 phony 사용

clean :

rm $(target) $(objects)

-> 현재 디렉토리에 있는 target 파일과 모든 오브젝트 파일들을 지움

**4.**

다음과 같은 알고리즘을 통해 규칙 R5를 구현할 수 있다.

main함수에서 if문을 사용해 line2[0]가 공백 문자이거나 count가 0이 아닐 경우 ‘\n’을 출력하고, B\_flag와 count를 0으로 설정한다.

> line2[0]가 공백 문자일 경우 ‘\n’을 출력한다는 점에서 입력줄의 첫 글자가 blank이면 앞줄과 합쳐지지 않게 할 수 있다. 또한 첫 부분의 여러 개의 blank가 있으면 이 역시 줄을 바꾸어 새 줄에 출력할 수 있다. 문자열 앞의 공백은 따로 제거하지 않으므로 첫 부분의 blank는 첫 번째 blank를 포함해 그 개수만큼 그대로 출력한다.

**5.**

-C dir : Makefile 읽기를 중단하고 우선 dir로 이동

-d : Makefile 수행 중 각종 정보 출력

-f <filename> : <filename>에 해당되는 파일을 Makefile로 취급

-h : 옵션에 관한 도움말 출력

-k : 에러가 나도 실행을 중단하지 않고 계속 진행

-p : make에서 내부적으로 정해진 값들을 출력

-r : 내장된 규칙을 없는 것으로 간주, 따라서 사용자가 규칙을 재정의하는 것 필요

-t : 파일의 생성 날짜를 현재 시간으로 갱신

-v : make의 버전 출력

**6.**

**int main(int argc, char \*argv[]) { 두 개의 입력 버퍼를 마련해 main 함수를 작성.**

각 변수들 선언 및 초기화.

argc가 2가 아닌 경우 “Usage: fmt filename | > outfile” 출력.

파일 읽기 시도.

파일 읽기에 실패할 경우 “File open error.” 출력.

line1, line2 메모리 할당 시도.

line1, line2 메모리 할당에 실패할 경우 “Memory allocation error” 출력 후 프로그램 종료.

line1 읽어오기.

빈 파일일 경우 프로그램 종료.

line1에 대하여 필요한 전처리 과정 수행 : Remove\_Blanks\_At\_The\_End 함수 호출(문자열 마지막의 공백 문자들 제거)

line1의 첫 글자가 ‘\n’일 경우 ‘\n’ 출력.

Count 0으로 설정.

while (1) {

B\_Line이 0일 경우 Print\_Line 함수 호출(문자열 출력)

그렇지 않을 경우 B\_Line을 0으로 설정

Count가 0이 아닐 경우 B\_Flag를 1로 설정

line2 읽기 시도.

읽지 못했을 경우 EOF, while loop 탈출. -> 텍스트 파일의 마지막 줄이 빈 줄인 경우 Print\_Line 함수에서 아무것도 출력되지 않으면 다음 줄 line2를 읽을 때 EOF가 검출되어 루프의 수행을 종료하게 된다.

line2에 대하여 필요한 전처리 과정 수행 : Remove\_Blanks\_At\_The\_End 함수 호출(문자열 마지막의 공백 문자들 제거)

line2[0]가 ‘ ’이거나 Count가 0이 아닐 경우 ‘\n’ 출력 후 B\_Fla g와 Count 값 0으로 설정

else if line2[0]가 ‘\n’일 경우

B\_Flag가 1일 경우 ‘\n’ 출력, B\_Flag 0으로 설정.

‘\n’ 출력 후 B\_Line은 1으로, Count는 0으로 설정.

line1과 line2의 내용 서로 바꾸기.

}

line1의 첫 글자가 ‘\n’이 아닐 경우

for { (i값 0으로 초기화 loop 한번 돌아갈 때마다 i값 1씩 증가)

line1[i]가 ‘\n’ 혹은 ‘\0’일 경우 break.

line1[i]가 ‘\n’일 경우 ‘\n’ 출력.

}

-> 파일을 읽는 while loop가 종료된 후에 마지막 줄이 빈 줄인지를 검사하여 이를 출력한다.

} 프로그램 종료.

**void Remove\_Blanks\_At\_The\_End(char \*line) { 문자열 끝의 blank들을 제거하기 위해 사용된다.**

int i, k 선언. newline\_flag 0으로 초기화.

for { (k값 0으로 초기화 loop 한번 돌아갈 때마다 k값 1씩 증가)

line[k]가 개행 문자일 경우 newline\_flag를 1로 set, break.

line[k]가 ‘\0’일 경우 break.

}

for { (i값 k-1로 설정 loop 한번 돌아갈 때마다 i값 1씩 감소, i가 0보다 작을 경우 loop 탈출)

line[i]가 공백 문자가 아닐 경우 break.

}

i값 1 증가.

newline\_flag가 1이면 line[i]를 ‘\n’으로 설정, line[i+1]을 ‘ ’로 설정.

그렇지 않으면 line[i]를 ‘\0’으로 설정.

}

**void Get\_Blanks\_Chars(char \*line, int Start, int \*N\_Blanks, int \*N\_Chars) { Start 이후 존재하는 blank 글자의 개수(N\_Blanks에 저장)와 다음 단어의 글자 수(N\_Chars에 저장)를 세기 위해 사용된다.**

int i 선언 및 blank\_flag 0으로 초기화

\*N\_Blanks, \*N\_Chars 0으로 설정.

for { (i = Start, 루프 한 번 돌 때마다 i값 1씩 증가)

line[i]이 개행 문자 혹은 NULL 문자이면 break.

line[i]이 공백 문자이면

blank\_flag가 0일 경우 (\*N\_Blanks)값 1 증가

그렇지 않으면 break.

line[i]이 개행 문자, NULL 문자, 공백 문자 중 어느 것도 아닐 경우

blank\_flag를 1로 설정

(\*N\_Chars) 값 1 증가

}

}

**void Print\_Line(char \*line, int \*Count, int \*B\_Flag) { 문자열 출력 함수, 문자열을 단어별로 구분하여 이들을 한번에 출력한다.**

int i, N\_Blanks, N\_Chars 선언 Start값 0으로 초기화

Get\_Blanks\_Chars(line, Start, &N\_Blanks, &N\_Chars) 문자열 끝의 blank 제거.

while { (N\_chars가 0이 되면 루프 종료)

if { \*B\_Flag가 1이고, (\*Count + N\_Chars + 1)가 LIMIT보다 작거나 같으면

N\_Blanks가 0이 아니면 “Something Wrong!” 출력

프로그램 에러 알리고 종료.

‘ ’출력.

\*B\_Flag를 0으로 set.

for { (i값 Start로 설정, 루프 한 번 돌 때마다 i값 1씩 증가, i값이 Start + N\_Chars보다 크거나 같으면 루프 종료)

line[i] 출력.

}

Start값에 N\_Chars 더해서 Start에 저장.

\*Count값에 N\_Chars + 1 더해서 \*Count에 저장.

Get\_Blanks\_Chars(line, Start, &N\_Blanks, &N\_Chars)

}

else if { (\*Count + N\_Blanks + N\_Chars)가 LIMIT보다 작거나 같은 경우

for { (i값 Start로 설정, 루프 한 번 돌 때마다 i값 1씩 증가, i값이 Start + N\_Blanks + N\_Chars보다 크거나 같으면 루프 종료)

line[i] 출력

}

Start값을 Start + N\_Blanks + N\_Chars로 설정.

\*Count값을 \*Count + N\_Blanks + N\_Chars+1로 설정.

Get\_Blanks\_Chars(line, Start, &N\_Blanks, &N\_Chars).

}

else { 위의 두 조건 모두 만족하지 않으면

\*Count가 0인 경우

for { (i값 Start로 설정, 루프 한 번 돌 때마다 i값 1씩 증가, i값이 Start + N\_Blanks + N\_Chars보다 크거나 같으면 루프 종료)

line[i] 출력.

}

Start값을 Start + N\_Blanks + N\_Chars로 설정.

개행 문자 하나 출력.

Get\_Blanks\_Chars(line, Start, &N\_Blanks, &N\_Chars). Start값을 Start + N\_Blanks로 설정.

\*Count가 0이 아닌 경우

개행 문자 하나 출력.

\*B\_Flag값 0으로 set.

for { (i값 Start + N\_Blanks로 설정, 루프 한 번 돌 때마다 i값 1씩 증가, i값이 Start + N\_Blanks + N\_Chars보다 크거나 같으면 루프 종료)

line[i] 출력.

}

Start값 Start + N\_Blanks + N\_Chars로 설정.

\*Count값 N\_Chars로 설정.

Get\_Blanks\_Chars(line, Start, &N\_Blanks, &N\_Chars).

}

}

}