2주차 결과보고서

전공: 아트엔테크놀로지 학년: 3학년 학번: 20191172 이름: 함승우

**1.**

텍스트이(가) 표시된 사진

자동 생성된 설명

**2.**

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

R1에 따라 출력 각 줄의 마지막에 blank를 만들지 않고, 줄 바꿈 문자가 나왔을 때 새로운 줄로 바꾸는 역할을 하고, EOS가 나왔을 때 종료한다.

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

줄 바꿈 문자와 EOS를 제외하고, blank를 찾기 위한 함수이다. blank라면 N\_Blank를, blank가 아니라면 N\_Chars를 증가시킨다.

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

Get\_Blanks\_Chars로 확인된 빈칸의 개수와 글자의 개수를 감안해 Limit에 맞게 출력 된다. 추가로 줄을 바꿔서 새로운 줄로 출력해야 하는 경우도 해당된다.

**3.**

cc=gcc

**컴파일 하기 위한 gcc를 cc로 매크로화한다.**

cflags = -W -g

**gcc 컴파일 옵션으로 W는 추가적인 경고와, g는 디버그 정보를 추가하는 의미가 있다.**

target = main

**빌드 대상의 이름이 main이다, 실행파일의 이름이 된다.**

objects = main.o String\_Manipulation.o Output.o

**‘object’라는 변수를 각 오브젝트 파일의 목록으로 설정한다.**

$(target) : $(objects)

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

**target 파일을 위한 object 파일들이 필요하고, gcc 컴파일러를 사용하여 실행파일을 만드는 과정이다.**

%.o : %.c

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

**$@ $<는 각각 현재 target 파일, 현재 의존성 중 첫번째 파일의 이름을 나타내고, 이를 통해서 c확장자를 가진 소스파일로 o확장자를 가진 목표 파일들로 만드는 과정이다.**

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

**header.h 파일이 변경된다면 다시 이 오브젝트 파일들도 컴파일해야 함을 알려준다.**

.PHONY : clean

**Phony는 실제 파일이 아니고, 더미 target이고, .phony를 사용해서 .phony에 clean을 정의한다.**

clean :

rm $(target) $(objects)

**타겟과 오브젝트 파일들을 모두 삭제하라는 명령어이다.**

**4.**

문자 한 줄 입력 받기 = line1

그 외의 경우

line1[0] = \n일 경우

line1[0] = ‘ ‘일 경우

그대로 출력

\n을 출력하고 나머지 line1을 출력한다

**5.**

|  |  |  |  |
| --- | --- | --- | --- |
| **옵션** | **의미** | **옵션** | **의미** |
| -C dir | Makefile을 읽기전 dir 디렉토리로 변경한다. | -v | Make의 버전을 출력 |
| -d | 일반적인 처리에 덧붙여 디버깅 정보를 출력한다 | -p | make에서 내부적으로 세팅되어있는 값들 출력 |
| -e | 환경으로부터 취해진 변수들이 makefile의 변수들보다 더 우선하도록 한다. | -r | 내장된 각종 규칙을 없는 것으로 간주한다. 사용자가 새롭게 정의해야 한다. |
| -f file | File을 makefile로 읽는다. | -S | -k옵션의 효력을 취소한다. |
| -h | Make가 이해하는 옵션들을 보여주고 종료한다. | -k | 에러 이후에도 가능한 한 계속하도록 한다. |
| -i | 파일들을 다시 만들기 위해서 실행되는 명령들에서 모든 에러들을 무시한다. | -n | 실행될 명령들을 인쇄하지만 실행하지 않는다. |
| -t | 파일의 생성 날짜를 현재 시간으로 갱신 |  |  |

**6**

‘fmt’를 구현하는 데 있어 필요한 함수는

2-1. main 함수

#include "Header.h"

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

FILE \*fp;

char \*line1, \*line2, \*tmpline;

int Count = 0, Blanks = 0, OBuf\_idx=0;

int i, B\_Flag = 0, B\_Line = 0;

if ( argc != 2 ) {

printf("Usuage: fmt filename | > outfile\n");

}

fp = fopen(argv[1],"r");

if ( fp == NULL ) {

printf("File open error.\n");

}

// argc는 입력받는 정보의 개수(아무것도 입력 받지 않을 때는 1)

// 그러므로 argc가 아무것도 입력 받지 않을 때와 입력 받은 텍스트

// 파일이 NULL일 때 에러 코드가 나온다.

line1 = (char \*)malloc(BNUM \* sizeof(char));

line2 = (char \*)malloc(BNUM \* sizeof(char));

if ( line1 == NULL || line2 == NULL ) {

printf("Memory allocation error\n");

exit(-1);

}

if ( fgets(line1, BNUM, fp) == NULL ) {

return 0;

// R9에 해당함

}

// Header.h에서 BNUM = 256으로 define했기 때문에 256개의 line1과 line2를 동적할당한다.

// 동적할당한 line1과 line2가 NULL이라면 메모리 할당 에러를

// 한줄을 읽고 empty 파일이라면 그대로 프로그램을 끝낸다.

Remove\_Blanks\_At\_The\_End(line1);

// 한 줄 읽고 Remove\_Blanks\_At\_The\_End 함수를 적용시킨다.

if ( line1[0]=='\n' ) {

putchar('\n');

}

// 첫줄이 줄바꿈 문자라면 , 그대로 줄바꿈 문자를 출력한다. 즉, 빈줄이라면 빈줄을 출력한다 R7

Count = 0;

// Count는 현재 줄에 실제로 출력할 글자의 개수를 저장하고 있으며 글자 하나를 출력할 때

// 마다 하나씩 그 값이 증가하도록 한다.

// EOS가 나타날 때까지 while 무한루프

while ( 1 ) {

if ( B\_Line == 0 ) {

Print\_Line( line1, &Count, &B\_Flag);

}

else {

B\_Line = 0;

}

if ( Count != 0 ) {

B\_Flag = 1;

}

// B\_Line은 한 줄을 한 줄을 읽고 뒤의 blank 글자를 제거하는 전처리 과정을 완료한 후에 그 줄

// 이 빈 줄이라면 B\_Line flag을 set 한다. B\_Line이 1이라면 ‘output L1'

// 을 호출할 필요 없이 다음 단계를 수행하도록 한다.

// 현재 줄을 완전히 출력하였는데 Count≠0이라면 계속 같은 줄에 출력할 수 있으므로 B\_flag을 1로 set하여 이를 수행토록 한다.만일

// 다음 줄이 빈 줄이거나 첫 글자가 blank이면 줄을 바꾸어야하므로 B\_flag을 0으로 set하

// 여 줄바꿈을 수행하도록 한다.

// 즉 B\_Line flag가 0이라면 빈 줄이 아니기 때문에 Print\_Line함수를 사용하여 출력하고, 그 이외의 경우에는 B\_line flag을 0으로 초기화한 후

// 나머지 작업을 실행한다. 또한 Count는 글자 하나를 출력할 때마다 하나씩 그 값이 증가하니까, Count가 0이 아니라면 계속 같은 줄에 출력할 수 있

// 으므로 B\_Flag를 1로 설정해 수행토록한다.

if ( fgets(line2, BNUM, fp) == NULL ) {

break; // EOF. Exit the while loop

}

Remove\_Blanks\_At\_The\_End(line2);

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

putchar('\n');

B\_Flag = 0;

Count = 0;

}

// 두번째줄이 빈칸으로 시작하고 빈줄이 아니라면 줄바꿈 문자를 출력하여 새로운 줄로 만들고, 새로운 줄이기 때문에 B\_flag과 Count는 0으로 초기화한다.

else if ( line2[0] == '\n' ) {

if ( B\_Flag == 1 ) {

putchar('\n');

B\_Flag = 0;

}

putchar('\n');

B\_Line = 1;

Count = 0;

}

// 줄바꿈 문자로 시작할 때 만약 B\_flag가 1이라면 = 빈줄이라면, 줄바꿈 문자를 출력하고, 새로운 줄이기때문에 B\_flag를 0으로 초기화한다.

// 이후에 줄바꿈 문자를 출력하고, 새로운 빈줄이기 때문에 B\_Line과 Count를 각각 1, 0으로 할당한다.

tmpline = line1;

line1 = line2;

line2 = tmpline;

//tmpline을 이용해서 포인터 변수 line1과 line2만 바꾼다.

}

if ( line1[0] != '\n' ) {

// if the last line is not empty line,

// check if the last char is '\n' and ouput it if it is.

for ( i = 0; ; i++ ) {

if ( (line1[i] == '\n') || (line1[i] == '\0') ) {

break;

}

}

if ( line1[i] == '\n' ) {

putchar('\n');

}

}

}

// 마지막 입력 마지막 줄의 끝에 줄바꿈 문자가 없으면 출력도 마찬가지로 줄바꿈 문자를 출력하지 않는다.

// 그러나 줄바꿈 문자가 있을 경우에는 이를 출력한다

// EOS가 나오면 break를 사용해 프로그램을 종료한다.

2-2. Remove\_Blanks\_At\_The\_End( char \*line )

2-3. Get\_Blanks\_Chars( char \*line, int Start, int \*N\_Blanks, int \*N\_Chars ) (string)

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

int i, k, newline\_flag = 0;

for ( k = 0; ; k++ ) {

if( line[k] == '\n') {

newline\_flag = 1;

break;

}

else if ( line[k] == '\0' ) {

break;

}

}

// 전체 문장에서 줄바꿈 문자가 있다면 newline\_flag가 1이된다. EOS인 경우에 break한다.

for ( i = k-1; i >= 0; i-- ) {

if ( line[i] != ' ') {

break;

}

}

//전체 문장에 대하여 line[i]가 space가 아니라면 break

i++;

//flag가 1일때 문장의 마지막을 줄바꿈&space, newline\_flag가 1이 아니라면 '\0'

if ( newline\_flag == 1 ) {

line[i] = '\n';

line[i+1] = '\0';

}

else {

line[i] = '\0';

}

}

// 이 과정을 거쳐 문장 끝의 Blank를 삭제하고 줄바꿈 문자가 있다면 줄을 바꿔주고 새로운 줄을 시작한다. 줄바꿈문자가 없다면

// 정해진 크기의 줄을 채워놓고 줄을 끝낸다.

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

int i, blank\_flag = 0;

\*N\_Blanks = \*N\_Chars = 0;

for ( i = Start; ; i++ ) {

if ( (line[i] == '\n') || (line[i] == '\0') ) {

break;

}

// 입력받은 Start에서 전체 문장까지 줄바꿈 문자가 있거나, EOS가 있는 경우 break를 한다.

else if ( line[i] == ' ' ) {

if ( blank\_flag == 0 ) {

++(\*N\_Blanks);

}

else {

break;

}

}

// 입력받은 Start에서 전체문장까지 Space가 있으면 break를 하나 blan\_flag가 0이었다면 \*N\_Blanks += 1을 한다.

else {

blank\_flag = 1;

++(\*N\_Chars);

}

// 만약 줄바꿈문자도 EOS도 space도 없다면, blank\_flag를 1로 set하고 \*N\_Chars += 1을 한다.

}

}

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

#include "Header.h"

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

int i, N\_Blanks, N\_Chars, Start=0;

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

// N\_Chars != 0이라는 것은 EOS, \n, space가 아닌 경우에 해당한다.

while ( N\_Chars != 0 ) {

if ( (\*B\_Flag == 1) && ((\*Count + N\_Chars + 1) <= LIMIT) ) {

// 줄을 바꾸지 않아도 되는 경우(줄바꿈X, blankX) 그리고 (단어의 개수가 Limit을 넘지 않을 때)

// N\_Blanks != 0은 다른 줄에 쓰여져 ERROR CODE를 발생시킨다.

if ( N\_Blanks != 0 ) {

printf("Something Wrong!\n");

exit(-1);

}

putchar(' ');

\*B\_Flag = 0;

for ( i = Start; i < Start + N\_Chars; i++ ) {

putchar(line[i]);

}

// line 출력한다.

Start = Start + N\_Chars;

\*Count = \*Count + N\_Chars + 1;

// Count를 N\_Chars를 이용해 단어의 개수를 세게끔 만들어준다.

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

}

else if ( (\*Count + N\_Blanks + N\_Chars) <= LIMIT ) {

// 줄을 바꿔야 하는 경우나 , space의 개수까지 포함한다.

for ( i = Start; i < Start + N\_Blanks + N\_Chars; i++ ) {

putchar(line[i]);

}

// 마찬가지로 출력한다.

Start = Start + N\_Blanks + N\_Chars;

\*Count = \*Count + N\_Blanks + N\_Chars;

// 줄이 다시 시작해야하는 부분 Start와 , 단어의 개수 Count를 다시 할당한다.

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

}

else { //나머지의 경우에도 마찬가지로 출력한다.

if ( \*Count == 0 ) {

for ( i = Start; i < Start + N\_Blanks + N\_Chars; i++ ) {

putchar(line[i]);

}

Start = Start + N\_Blanks + N\_Chars;

putchar('\n');

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

Start = Start + N\_Blanks;

}

else {

putchar('\n');

\*B\_Flag = 0;

for ( i = Start + N\_Blanks; i < Start + N\_Blanks + N\_Chars; i++ ) {

putchar(line[i]);

}

Start = Start + N\_Blanks + N\_Chars;

\*Count = N\_Chars;

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

}

}

}

}