2주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231515 이름: 김다은

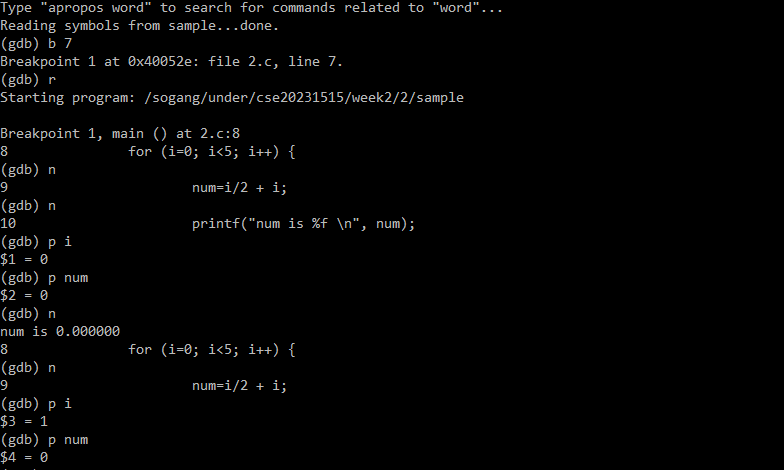
1.

1-1. 실습1 결과화면

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

1-2. 실습2 결과화면



수정한 코드

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

1-3. 실습3 결과화면

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

2.

**2-1. Remove\_Blanks\_At\_The\_End(char \*line) 함수**

텍스트에서 줄바꿈 ’\n’이 나타나면 해당 텍스트의 마지막에 나타나는 공백 문자가 아닌 문자 뒤에 줄바꿈과 space를 삽입한다. 줄바꿈이 나타나지 않는다면 NULL 문자를 삽입한다.

**2-2. Print\_Line(char \*line, int \*Count, int \*B\_Flag) 함수**

Header.h에 define된 LIMIT 값에 따라서 텍스트를 변환한 결과를 출력한다.

Get\_Blanks\_Chars() 함수로 공백과 문자의 수를 알아낸다. B\_flag는 이전 문장에 이어서 출력할지 여부를 결정한다. 첫번째로, B\_Flag가 1이고 전 문장의 문자 수와 현재 문장의 문자 수가 LIMIT보다 작으면 공백을 출력하고 현재 문장을 출력한다. 두 번째로, 이전 문장의 문자 수와 현재 문장의 공백의 수, 문자 수를 도한 것이 LIMIT보다 작은 경우 현재 문장을 출력한다. 나머지 경우, 줄바꿈을 삽입한 후 현재 문장을 출력한다.

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

텍스트에서 처음으로 줄바꿈이나 NULL 문자가 나타나기 전까지, 문장에서 공백의 수와 문자의 수를 세는 함수이다.

**2-4. int main(int argc, char \*argv[]) 함수**

argv[1]의 이름을 가진 텍스트 파일을 열고 내용을 읽어온다.

또한 이 알고리즘에서는 현재 처리하고 있는 문장과 다음에 처리해야 하는 문장을 모두 고려해야 하므로 각각을 저장할 line1과 line2를 선언한다. 이후 while문에서 Print\_Line 함수, Remove\_Blanks\_At\_The\_End 함수를 호출하며 텍스트의 형식을 변화시킨다.

3.

cc=gcc # cc: 사용할 컴파일러를 저장(gcc)

cflags = -W -g # cflags: 컴파일 옵션을 저장

# -W는 Warning Message를 활성화하는 옵션

# -g는 디버그 정보를 포함하도록 컴파일하는 옵션

target = main # target: 만들어질 실행 파일의 이름을 저장

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

# objects: 소스 파일(.c)을 처리하여 만들어지는 오브젝트파일(.o)의 목록을 저장한다.

$(target) : $(objects) # target이 objects에 의존한다는 규칙을 정의함

# 이 규칙은 컴파일된 오브젝트 파일들을 링크하여 최종 실행 파일을 생성함

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

%.o : %.c # 모든 .o파일은 각각의 .c파일에 대한 컴파일 규칙을 정의

# 이 규칙은 해당 .c 파일을 컴파일하여 .o 파일을 생성

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

main.o String\_Manipulation.o Output.o : Header.h

# main.o String\_Manipulation.o Output.o가 Header.h에 의존한다는 규칙을 정의

# 위 세 오브젝트 파일은 Header.h에 변경사항이 있을 때마다 재컴파일 된다.

.PHONY : clean # clean 규칙을 정의한다.

# .PHONY는 실제 파일 이름이 clean인 경우에도 이 규칙이 정상적으로 실행되도록 한다.

clean :

rm $(target) $(objects) # target, objects를 삭제한다.

# 빌드된 파일을 정리하여 프로젝트 폴더를 깔끔하게 유지할 수 있다.

4.

입력줄의 첫 글자가 blank이면 앞줄과 합쳐지지 않게 한다. 만일, 줄의 첫 부분에 여러 개의 blank가 있으면 이 역시 줄을 바꾸어 새 줄에 출력하고 첫 부분의 blank는 첫 번째 blank를 포함해 그 개수만큼 그대로 출력한다.

main.c에서 아래 부분이 R5의 내용을 수행한다.

|  |
| --- |
| while(1) {  if ( B\_Line == 0 ) {  Print\_Line( line1, &Count, &B\_Flag);  }  ~생략~  if ( line2[0] == ' ' && Count != 0) {  putchar('\n');  B\_Flag = 0;  Count = 0;  }  ~  tmpline=line1;  line1 = line2;  line2 = tmpline;  } |

Count!=0(이전 문장이 존재)이고 문장의 첫 자리가 공백인 경우 (line2[0] == ‘ ‘), ‘\n’을 출력 버퍼에 cat 삽입 함으로서 앞 문장과 합쳐지지 않도록 한다. 이때 line1과 line2는 현재 처리하고 있는 문장과 다음에 처리해야 하는 문장을 모두 고려하는 데에 사용된다.

반복문이 다시 실행되고 Print\_Line 함수가 call 된다. Print\_Line 함수에서 Get\_Blank\_Chars 함수를 실행한다. 여기에서 앞부분에 존재하는 공백의 개수를 구한다. 이후 다시 Print\_Line 함수에서 구한 개수만큼 공백을 출력한다.

5.

|  |  |
| --- | --- |
| 옵션 | 설명 |
| -p | print database  ‘make’ 내부에 세팅되어 있는 값 출력 |
| -k | keep going  에러가 발생하더라도 계속 실행 |
| -r | no built-in rules  내장된 규칙 무시 |
| -d | debugging  디버깅 관련 모든 정보 출력 |
| -f <FILE> | 인자로 받은 <FILE>를 Makefile로 취급 |
| -e | 환경 변수가 makefile 내의 매크로 지정을 재정의하도록 지정 |
| -n | 명령을 표시하지만 실행하지는 않음 |
| -B | 강제로 모든 대상을 다시 빌드 |
| -C DIR | 지정된 디렉토리에서 make 명령을 실행 |
| -q | 대상이 최신 상태인지 확인하고, 빌드할 필요가 있는지 여부를 결정 |
| -t | 대상 파일의 수정시간을 현재로 설정 |