**System programming project1**

**데이터 사이언스전공**

**2223403 김찬종**

**Email** [**chan1130@naver.com**](https://mail.naver.com/)

**☎ 010-9776-8798**

**개발 환경: 윈도우 11 visual studio, gcc, make**

**먼저 gcc란 GNU C Compiler의 약자입니다.**

**GNU는 자유 소프트웨어 재단(Free Software Foundation)에서 진행 중인 OS 프로젝트를 뜻해요. gcc는 그 프로젝트 중 일부입니다. gcc는 현재 리눅스 계열 운영체제에서 표준적인 C언어 컴파일러로 기능하고 있습니다.**

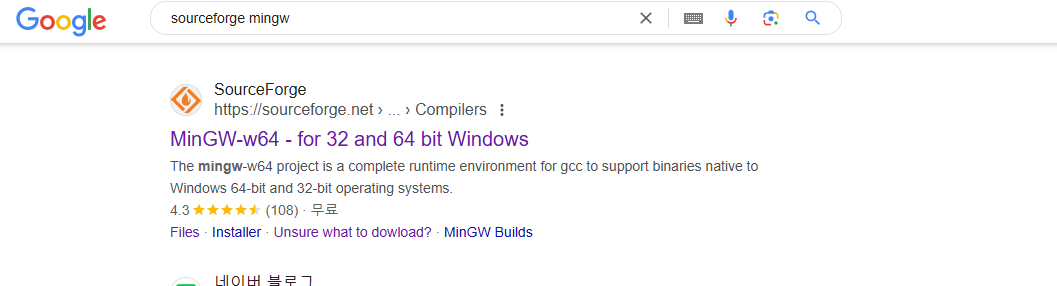
**​또 MinGW란 Minimalist GNU for Windows의 약자입니다.**

**​MinGW는 윈도우즈에 API를 이용하여, 윈도우즈 운영체제에서 gcc를 사용할 수 있게 해 주는 프로그램입니다.**

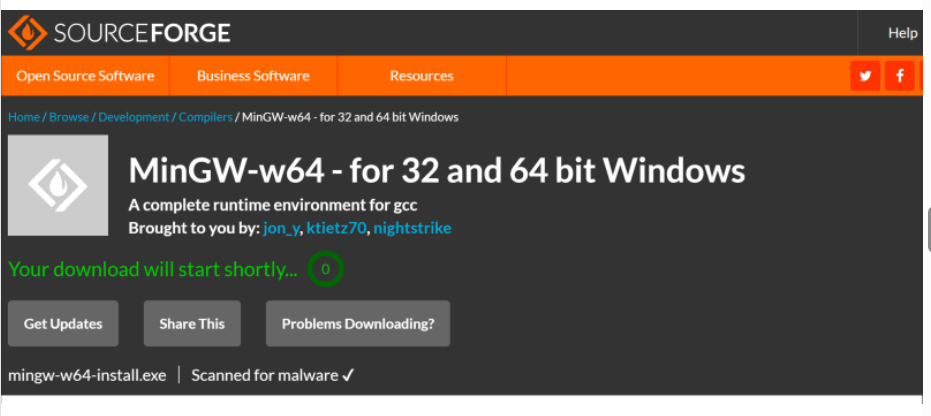
**윈도우에서 gcc 설치 방법**

[**MinGW-w64 - for 32 and 64 bit Windows - Browse Files at SourceForge.net**](https://sourceforge.net/projects/mingw-w64/files/) **에 접속해도 되고**

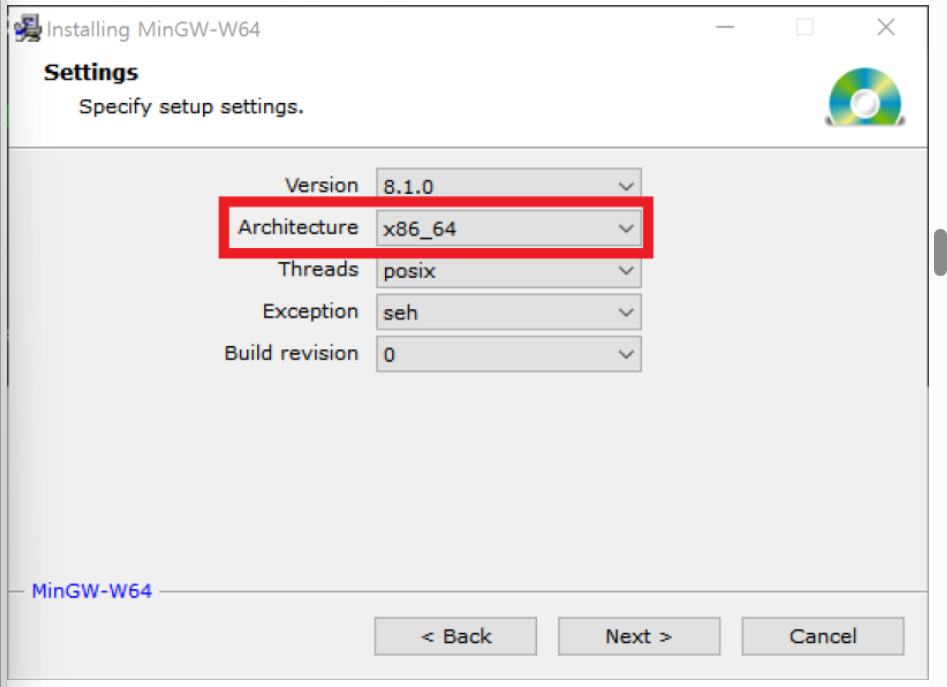
**구글에 아래 그림과 같이 검색하고 보라색 글자로 되어있는 부분을 찾아서 들어 가주시면 됩니다.**

****

**아래와 같은 그림에 접속했으면 download를 클릭해주면 설치가 됩니다.**

****

**아래 그림을 보시면 Architecture 부분을 x86\_64로 바꾸어주세요(x86\_64는 64비트 프로세서 의미)**

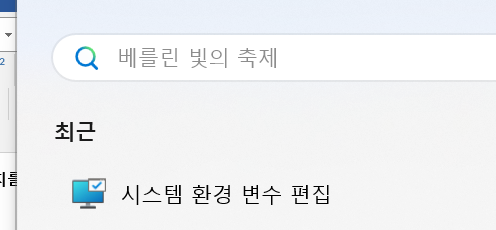
****

**이후에 Next버튼을 계속 눌러서 설치를 완료해 주시면 됩니다.**

**하지만 아직 윈도우즈 운영체제는 방금 다운로드받은 파일이 컴파일러인지 몰라요.**

**그래서 운영체제에게 이 파일의 존재와 위치를 알려줄 거예요.**

**시스템 환경 변수 편집에 들어가셔서 환경설정을 해주시면 됩니다.**

****

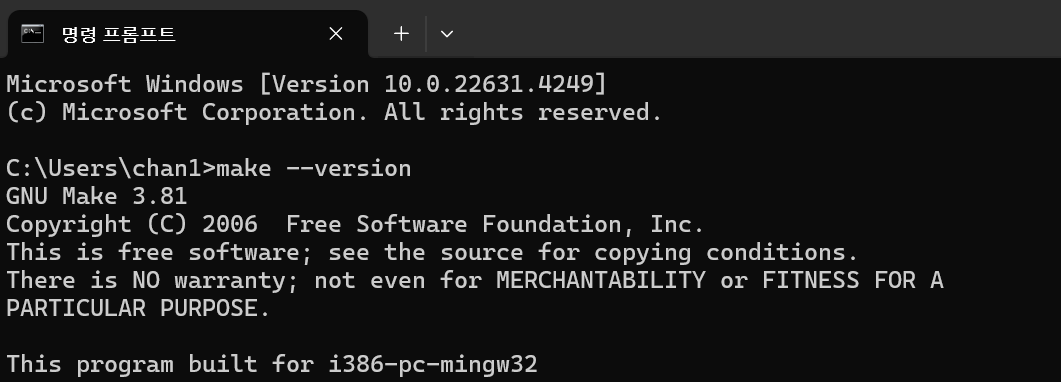
**그럼 시스템 속성에서 환경변수를 클릭합니다.**

**시스템 변수에서 path를 선택한후 편집을 눌러 주시면 됩니다.**

**새로 만들기를 클릭하시고 파란색부분으로 된**

**C:\mingw64\bin를 클릭해주시고 확인을 눌러주면 됩니다.**

**Gcc 버전 확인**

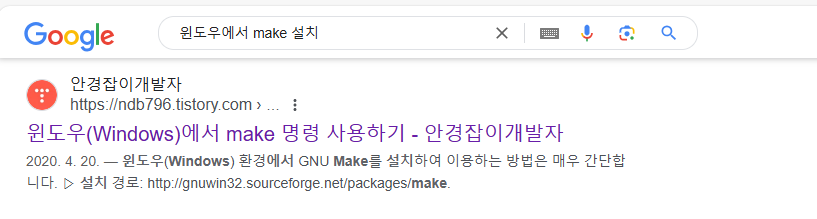
****

**이것으로 gcc가 잘 설치되었음을 알 수 있다.**

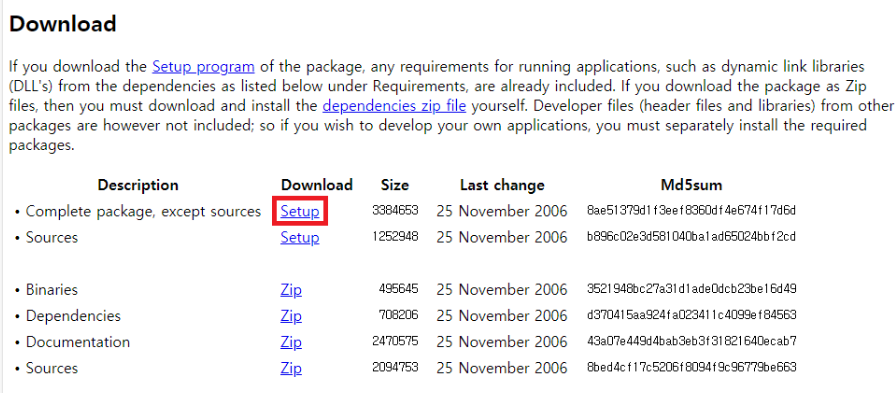
**Make 설치하기**

[**http://gnuwin32.sourceforge.net/packages/make.htm**](http://gnuwin32.sourceforge.net/packages/make.htm) **에 접속해도 되고**

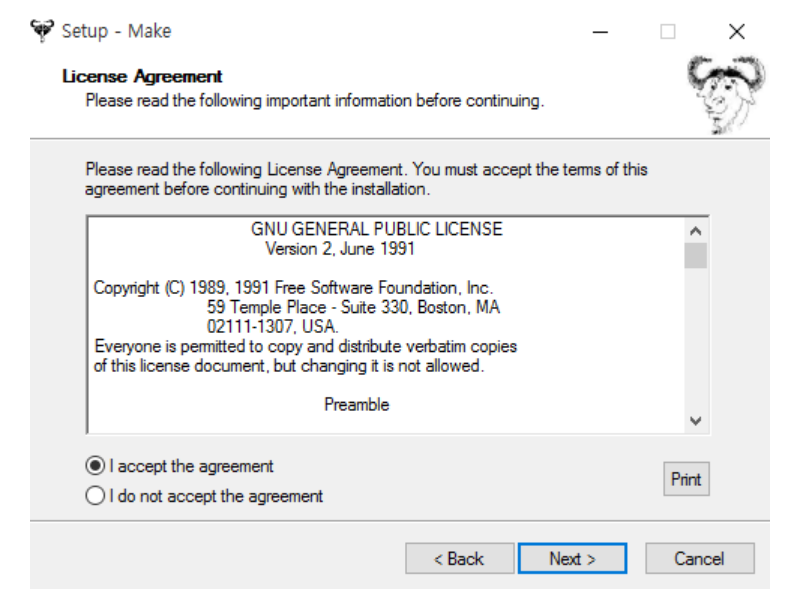
**아래 그림과 같이 구글에 검색하고 보라색 글씨로 된 부분을 찾아서 클릭해도 된다.**

****

**그런 다음 빨간색 직사각형으로 된 부분을 클릭해 설치를 하면 된다.**

****

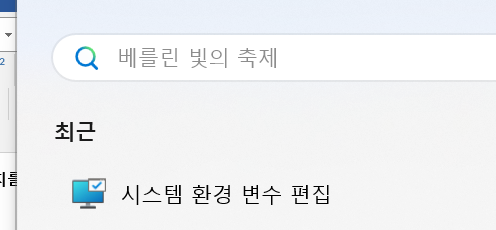
**그후 아래 2개 그림과 같이 Next를 눌러주면 됩니다.**

**** ****

**하지만 아직 윈도우즈 운영체제는 방금 다운로드받은 파일이 컴파일러인지 몰라요.**

**그래서 운영체제에게 이 파일의 존재와 위치를 알려줄 거예요.**

**시스템 환경 변수 편집에 들어가셔서 환경설정을 해주시면 됩니다.**

****

**그럼 시스템 속성에서 환경변수를 클릭합니다.**

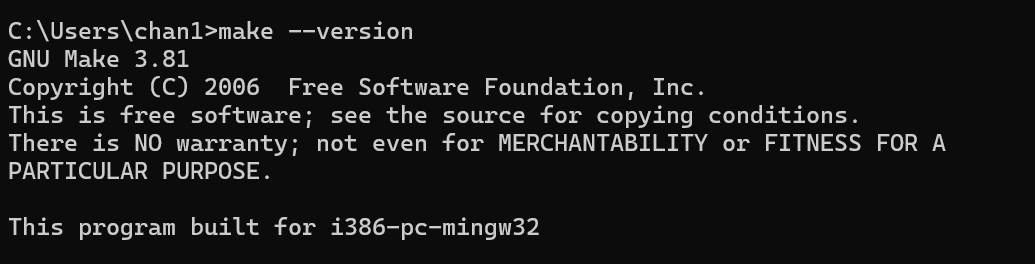
**아래 그림과 같이 사용자 변수 및 시스템 변수에서 path를 선택한후 편집을 눌러 주시면 됩니다.**

**새로 만들기를 클릭하시고 파란색부분으로 된**

**C:\Program Files (x86)\GnuWin32\bin 를 클릭해주시고 확인을 눌러주면 됩니다.**

**C:\Program Files (x86)\GnuWin32\bin**

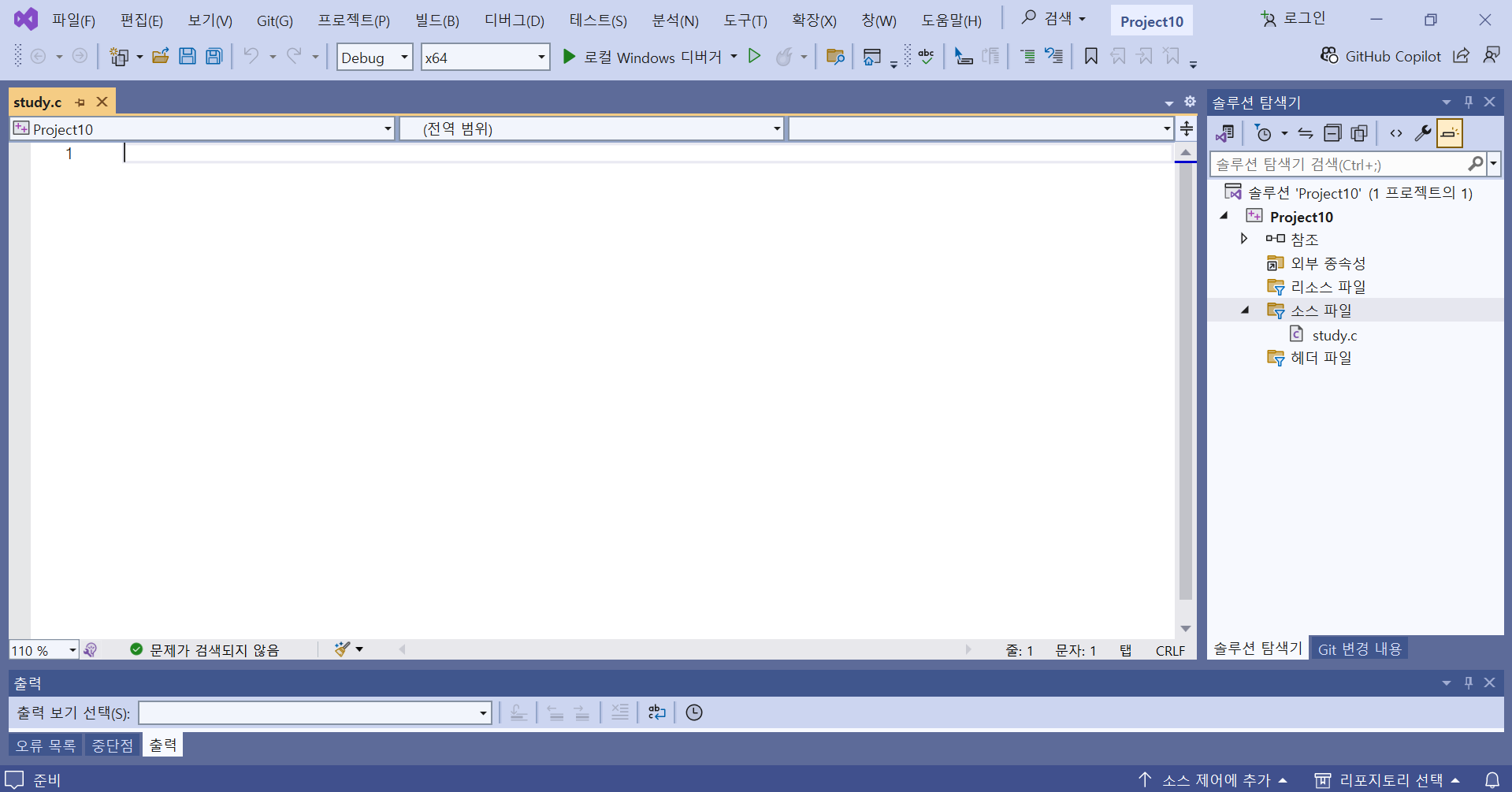
**Make 버전 확인**

****

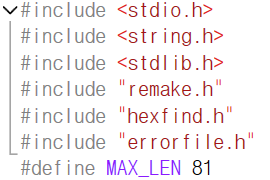
**이것으로 make가 잘 설치되었음을 알 수 있다.**

**이 둘은 윈도우 환경상에서 Makefile을 실행시키기 위한 도구입니다.**

**프로그래밍을 하기 위한 visual studio 2022(C언어로 구축)**

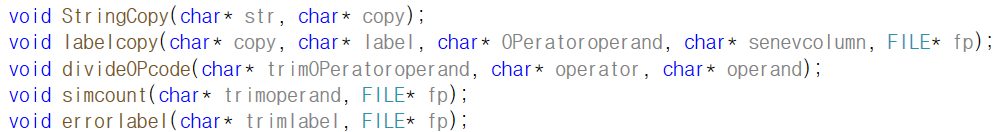
****

**(main.c 파일)**

**->그림1**

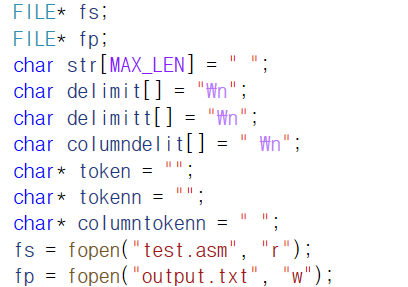
**->그림1**

**그림1 과 같은 코드를 작성한다. 그림의 맨 아래에 부분은 파일 한 줄을 읽을 때 저장할수 있는 문자의 수는 80이라 했고 \0을 포함 시키기위해 81로 설정해 두었다.**

**->그림 2**

**그림2에있는 함수 원형을 작성한다.**

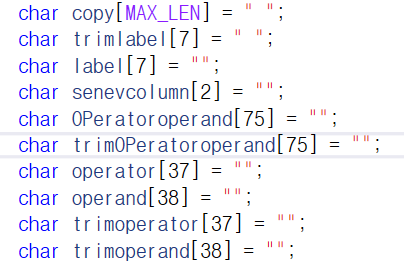
**그 다음 int main(void){}를 작성하고 {}안에 들어 가 있는 코드 내용을 설명할 것이다.**

**->그림 3**

**그림3에”test.asm” 파일을 여는 fs와 출력 값 결과를 적는 fp(“output.txt”생성)를 작성하였다. 파일 한줄을 저장하는 str문 크기 81지정 label 문의 줄바꿈 기준을 자르는 delimit, 그걸 생성하는 token, 7번째 colum을 빈칸 및 줄바꿈을 기준을 자르는 columndelit과 그걸 생성하는 columntoken, operator를 줄바꿈을 기준으로 자르는 delimitt과 그 걸 생성하는 tokenn을 작성하였다.**

**다음은 아래와 같은 코드를 작성하였다.**

**while (fgets(str, sizeof(str), fs) != NULL) {} -> 읽는 파일의 끝에 도달할 때 까지 str에 저장하는 것이며 {}안에 담겨 있는 코드에 대해 설명 할 것이다.**

**->그림4**

**파일 행마다 주석을 파악하기위해 복사를 한 copy문 크기81 설정 label은 6열까지 저장할 수 있어서 크기는7, 앞,뒤 공백을 제거한 문장을 저장하는 trimlabel 크기는 7로 설정**

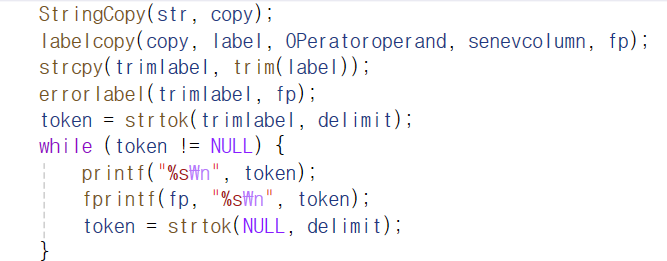
**7열에는 주석 ‘.’ 및 빈칸 올 수 있고 그 외의 문자가 오면은 오류처리를 하는 코드를 작성했는데 7열에 오는 문자를 저장하는 sevencolumn 크기는 2로 설정**

**Label 뒤에는 operator, operand를 저장하는 OPeratoroperand문 크기는 75설정 이의 앞뒤 공백을 제거한 문장을 저장하는 trimOPeratoroperand 크기는 75설정**

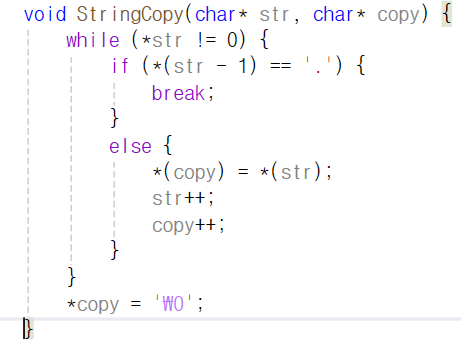
**trimOPeratoroperand에서 이제 operator과 operand를 구분해야하는데 각각 저장하는 operator,** **operand**

**크기는 37, 38로 설정해 두었고 앞뒤 공백을 제거한 문장을 저장하는 trimoperator,** **trimoperand 크기는 각각 37,38로 설정했습니다.**

**다음 그림5와 같은 코드를 작성한다.**

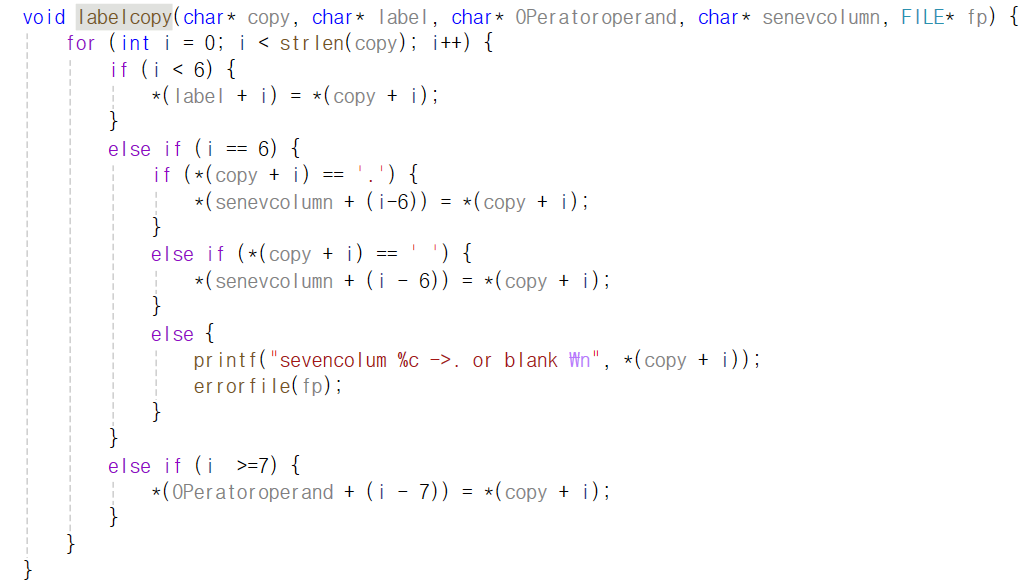
**->그림 5**

**StringCopy함수는 아래와 같다.**

**->그림6**

**각 행을 읽는 문자열 str에 copy문자열에 저장하는 것인데 str이 문장이 끝날 때까지 실행하고 str의 포인터를 증가해 str포인터 값의 -1에 주석 ‘.’이면 실행을 멈추고 문장의 끝을 알리는 \0을 저장한다. 그 외의 상황이면 정상적으로 복사가 이루어지고 반복조건이 끝나면 문장의 끝을 알리는 \0을 저장한다.**

**다음은 labelcopy에 관한 함수 설명이다. 함수는 아래 그림과 같다.**

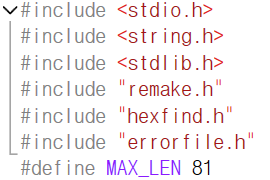
**->그림7**

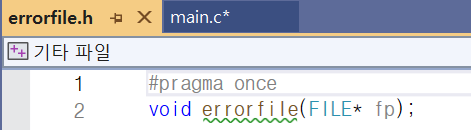
**->그림 7**

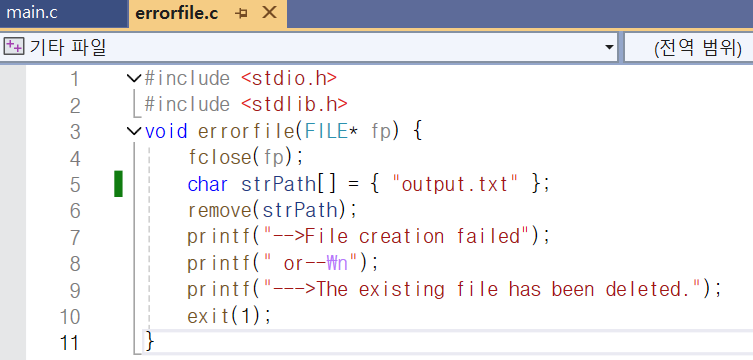
**이 함수는 copy문장에 label과 7열 OPeratoroperand를 구분해주는 문장이다.**

**copy문장 길이만큼 반복을 실행하고 i의 값이 6보다 적으면 label에 저장하고 i가 6일 때 ‘.’, 빈칸이면 저장하고 그 외의 문자가 왔을 경우 실행오류를 발생시키는 것이다.**

**여기서 errorfile(fp); 에 대해 설명할 것이다.**

**->그림1에서 #include “errorfile.h”을 추가하고 errorfile.c, errorfile.h를 만든다.**

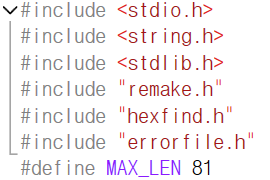
**->errorfile.h**

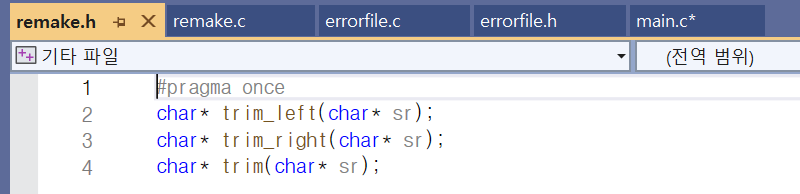
**->errorfile.c**

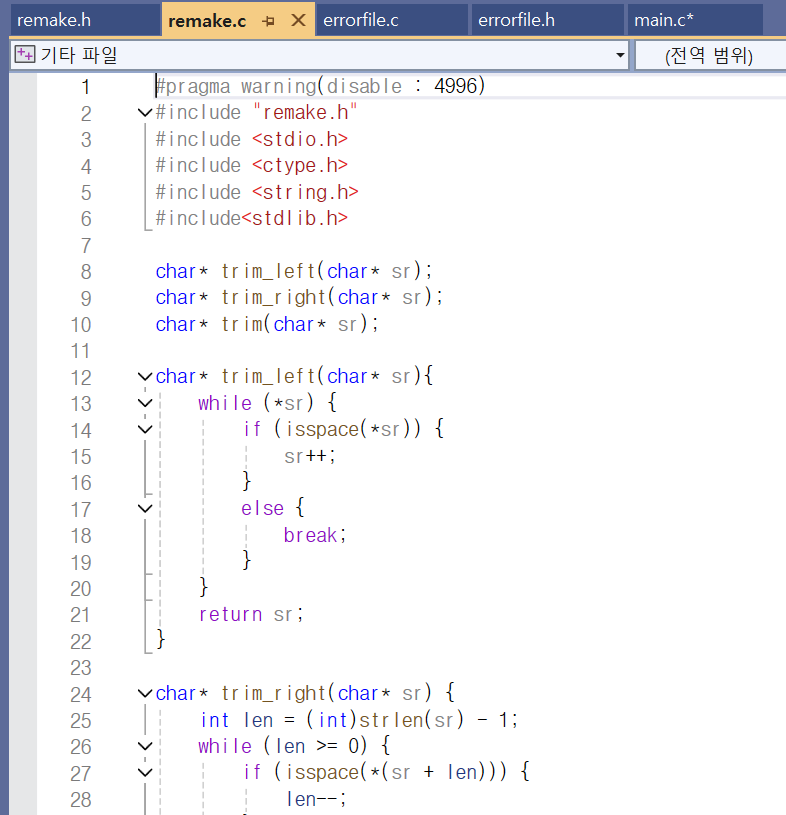
**결과값을 작성하는 fp를 받아들여와 fp를 닫고 파일이름인 output.txt이 존재할 경우 수정하지 않고 삭제가 되는 파일생성실패문구를 출력한 뒤 실행을 종료하는 코드이다. 만약에 output.txt가 존재하지 않을 경우에는 파일이 생성되지 않습니다.**

**계속해서 그림 6에 가자면 i가 7이상이면 OPeratoroperand에 저장하는 코드이다.**

**그림5에 이어서 설명하자면 다음은 strcpy(trimlabel, trim(label)); 인데 strcpy는 문자열을 복사하는 것이고 여기서 trim에 대해 설명할 것이다.**

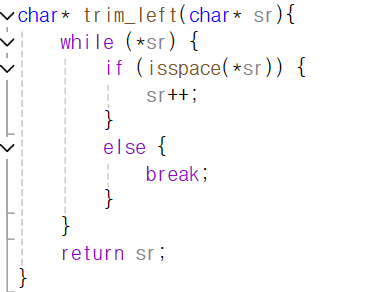
**->그림1에서 #include “remake.h”을 추가하고 remake.c, remake.h를 만든다.**

**-> remake.h**

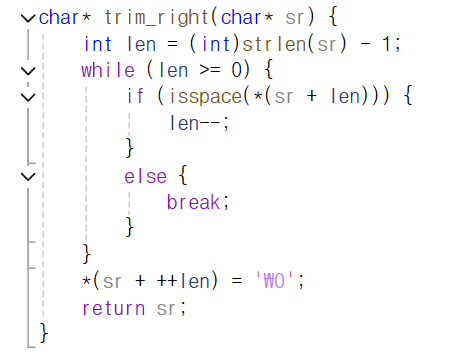
**-> 그림8 remake.c**

**그림8에 대해 설명하 자면**

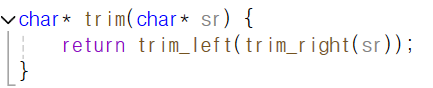
**먼저 그림8에 1행부터~6행을 적고 8행부터 10행까지의 함수의 원형을 적는다.**

****

**->문자열의 맨 왼쪽에서 부터 있는 공백을 제거하고 문자열을 반환하는 함수이다.**

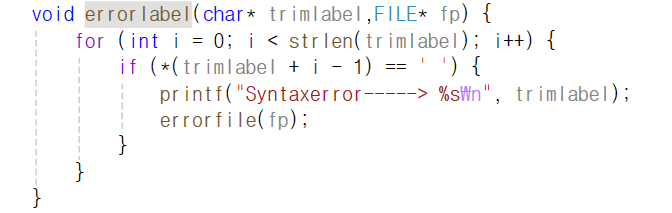
****

**->문자열의 맨 오른쪽에서 부터 있는 공백을 제거하고 문자열을 반환하는 함수이다.**

****

**->최종적으로 trim함수는 trim\_right을 실행하고 trim\_leftf를 실행해 맨 왼쪽, 오른쪽으로 부터의 문자열에 공백이 없는 문자열을 반환하는 함수이다.**

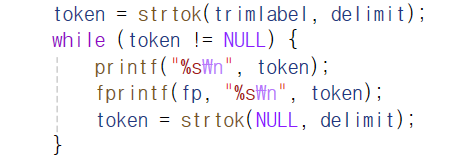
**그림 5에 이어서 errorlabel(trimlabel, fp);에 대해 설명 하고자 한다.**

****

**->이는 label의 맨 왼쪽 오른쪽 공백을 제거한 것을 trimlabel에 저장하고 도중에 빈칸이 있으면 실행오류를 시키는 함수이다.**

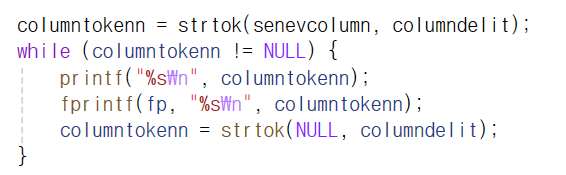
**Ex) trimlabel에 “ST ART”->(trim은 맨 왼쪽, 오른쪽으로부터의 공백을 제거하지 중간은 제거 안 하도록 코드를 만들 었다.) ST ART는 도중에 빈칸이 있으므로 실행오류가 된다.**

**그림 5에 이어서**

****

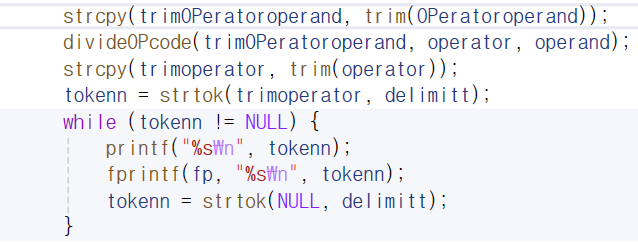
**Strtok에서 trimlabel을 delimit를 기준으로 잘라 토큰을 생성하고 token이 null이아닐 때 까지 token값을 출력 하고 출력결과를 작성 하고 여기서 중요한 것은 token=stork(NULL,delimit)**

**strtok 함수는 문자열 부분에 NULL을 넣어줍니다. 자세히 설명하자면 NULL을 넣었을 때는 직전 strtok 함수에서 처리했던 문자열에서 잘린 문자열만큼 다음 문자로 이동한 뒤 다음 문자열을 자릅니다. 만약에 밑줄 친 부분을 작성 안 하게 되면 잘린 문자열의 포인터를 다시 넣었을 때는 다음 문자로 이동하지 못하고 처음에 나오는 문자열만 계속 자르게 됩니다.**

**->그림9**

**그림 5에 다음 코드의 그림 9에 대해 설명하자면 7열에 있는 토큰을 생성하고 출력하는 코드이며**

**7열에는 주석인’.’ 빈칸만 올 수 있으며 분리기준은 빈칸 및 \n으로 설정했고 이를 출력하고 출력결과를 작성하는 코드를 이다.**

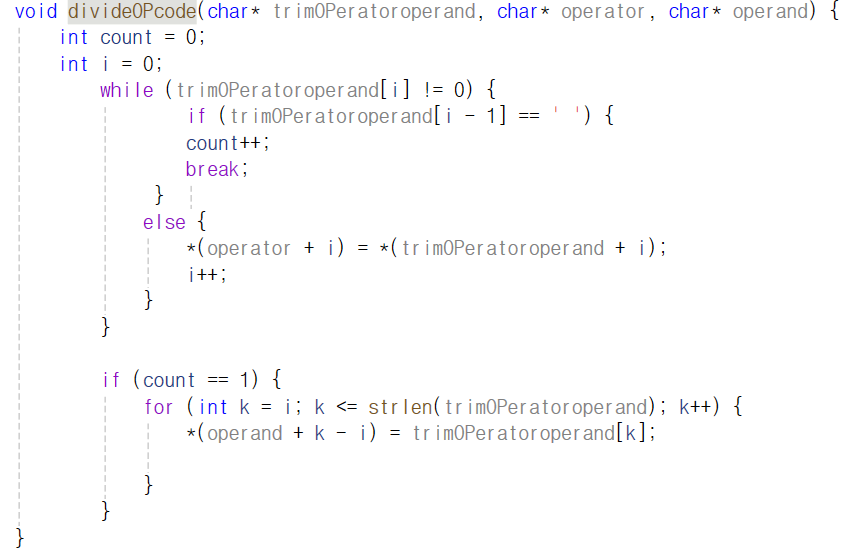
**->그림10**

**그림 9에 다음인 그림 10코드를 작성하고**

**strcpy(trimOPeratoroperand, trim(OPeratoroperand));은 trim된 OPeratoroperand를**

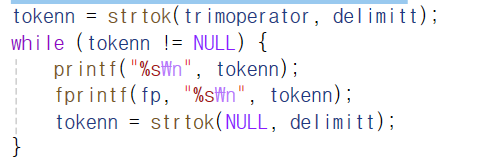
**trimOPeratoroperand에 복사하는 것이다. divideOPcode(trimOPeratoroperand, operator, operand);에**

**divideOPcode함수는 아래 그림과 같다.**

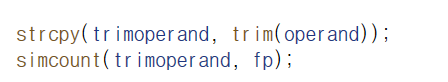
****

**trimOPeratoroperand 에 operator과 operand를 구분해주는 것이며 먼저 이 둘은 빈칸으로 구분 되는데 첫 번째로 빈칸이 나오면 빈칸 나온 전까지 operatror에 저장하고 그 이후에 나온 것들은 operand에 저장한다.**

**그림10에 이어서 다음 문장인 strcpy(trimoperator, trim(operator)); 은 operator에는 중간 빈칸은 없지만 맨 오른쪽부터의 빈칸이 남아 있을 수 있으므로 trim 함수를 사용하고 trimoperator에 복사하는 것이다.**

****

**이어서 위 그림은 trimoperator를 \n을 기준으로 토큰을 분리해 출력하고 출력 결과를 저장하는 코드이다.**

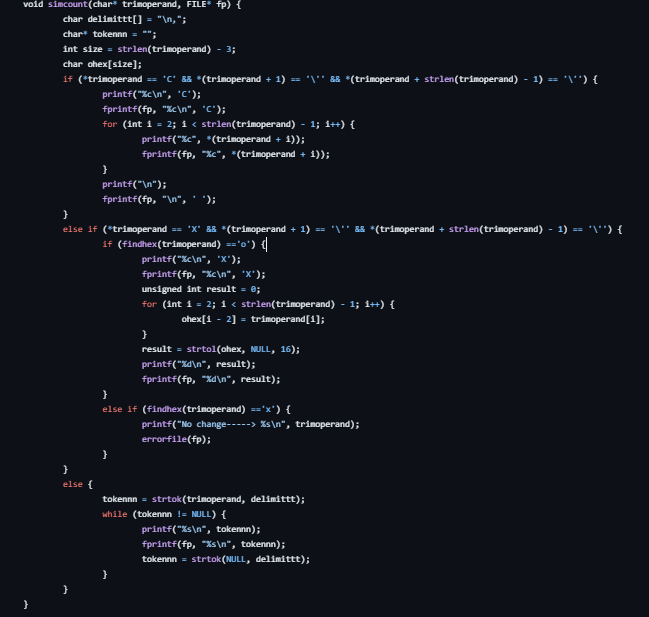
**->그림11**

**그림10에 이어서 그림 11을 작성하고**

**strcpy(trimoperand, trim(operand)); 는 operand를 trim해주고 trimoperand에 복사하는 코드이고**

**simcount(trimoperand, fp); 의 simcount 함수에 대해서 설명하고 자 한다.**

**simcount함수.**

****

**,를 분리**

**C’EOF’ 출력결과 EX1)**

**EX1)**

**C**

**EOF**

**X’F1’ 출력결과 EX2)**

**EX2)**

**X**

**241 을 출력하고 이에 대한 오류가 있을 시 실행오류를 시키는 코드이다.**

**char delimittt[] = "\n,";**

**char\* tokennn = "";**

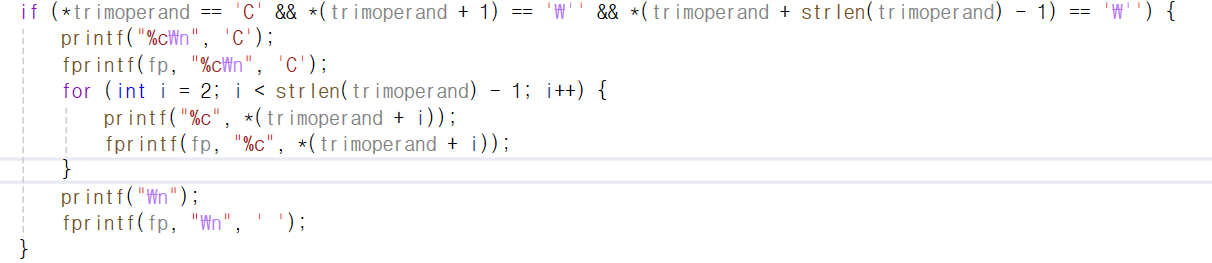
**int size = strlen(trimoperand) - 3;**

**char ohex[size];**

**먼저 이 코드 위에서부터 2번쨰 까지 delimittt는 줄바꿈과 , 기준으로 tokennn을 생성하고**

**양쪽’’안에 있는 16진수를 10진수로 바꾸기 위해 크기를 strlen(trimoprand)-3이라 했는데**

**이는 예를 들어 trimoperand가 X’F1’일 때 strlen은 길이를 나타내는데 strlen은 5이고 X, 양쪽 ‘은 개수가 3개 이므로 strlen(trimoprand)-3이라고 설정했고 16진수를 저장하는 ohex를 저장하는 문자열을 설정해주었고 strlen(trimoprand)-3는 상수가 아니므로 char ohex[size]; 이라는 ohex 문자열을 설정해 주었다.**

****

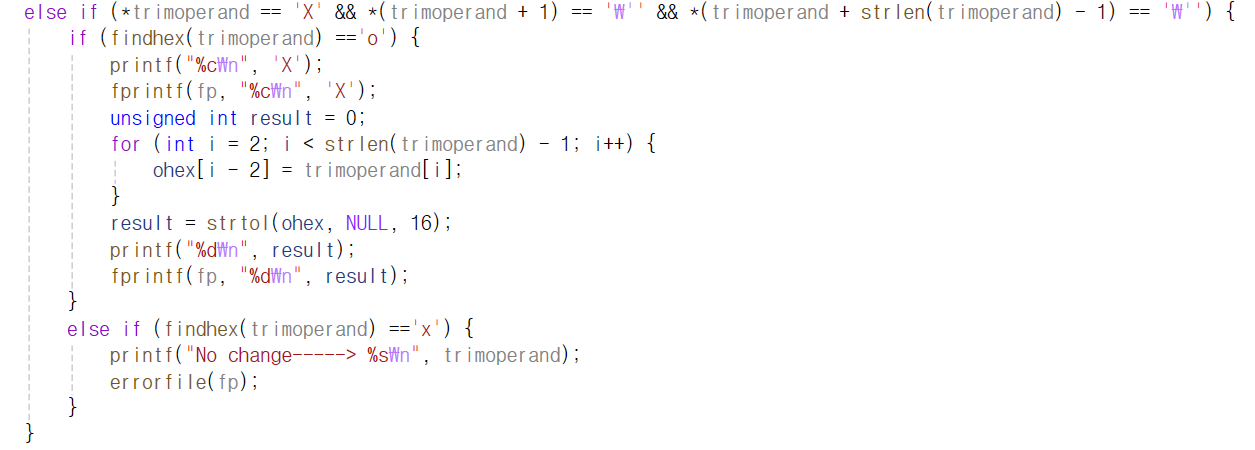
**는 \*(trimoperand)가 C이고 다음\*( trimoperand+1) 이 ‘ 맨 끝 부분이 ‘ 일 때 C를 출력하고 양쪽 ‘안에 있는 문자열을 출력해 각각 출력 값을 저장하는 코드이다.**

**C’EOF’ 출력결과 EX1)**

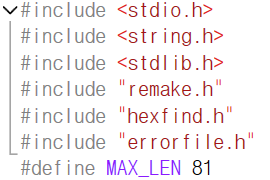
**EX1)**

**C**

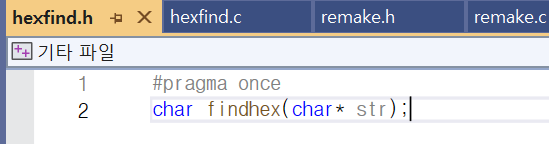
**EOF**

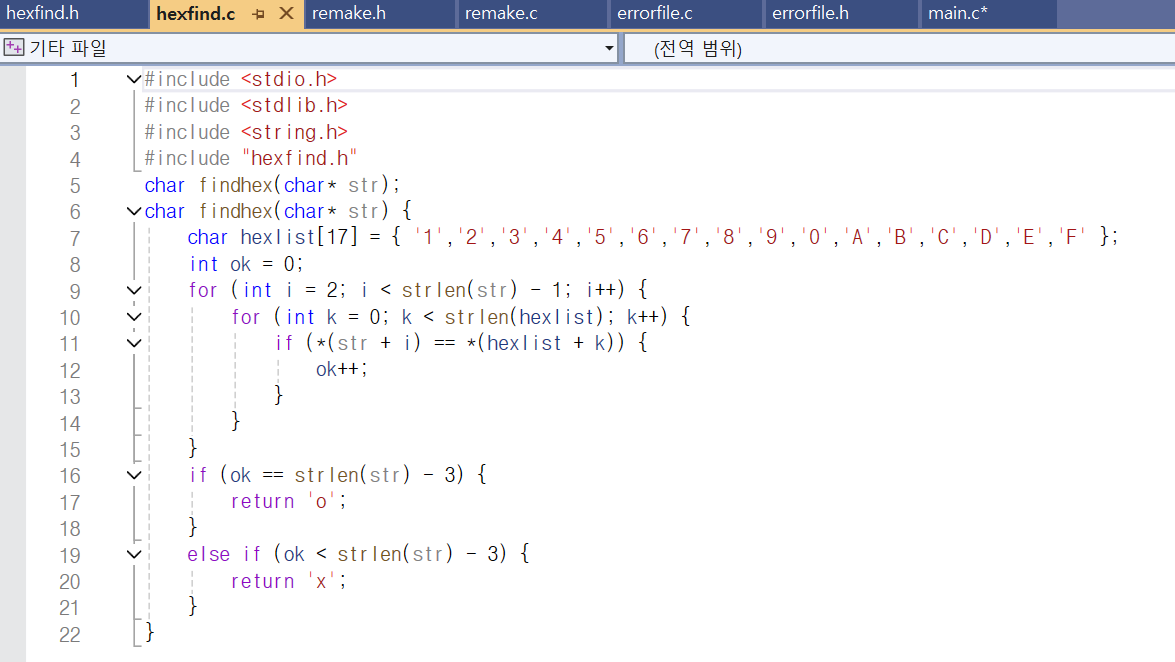
****

**이 코드는 \*(trimoperand)가 X이고 다음\*( trimoperand+1) 이 ‘ 맨 끝 부분이 ‘ 일 때 X를 출력하고 양쪽 ‘안에 있는 16진수를 출력해 각각 출력 값을 저장하는 코드이다. 단 양쪽 끝’ 안에 16진수가 아니면 실행 오류를 시키는 거다 그것은 findhex이다.**

**->그림1 main.c에 “hexfind,h”를 작성한다.**

**그런 다음 hexfind.c , hexfind.h를 만든다.**

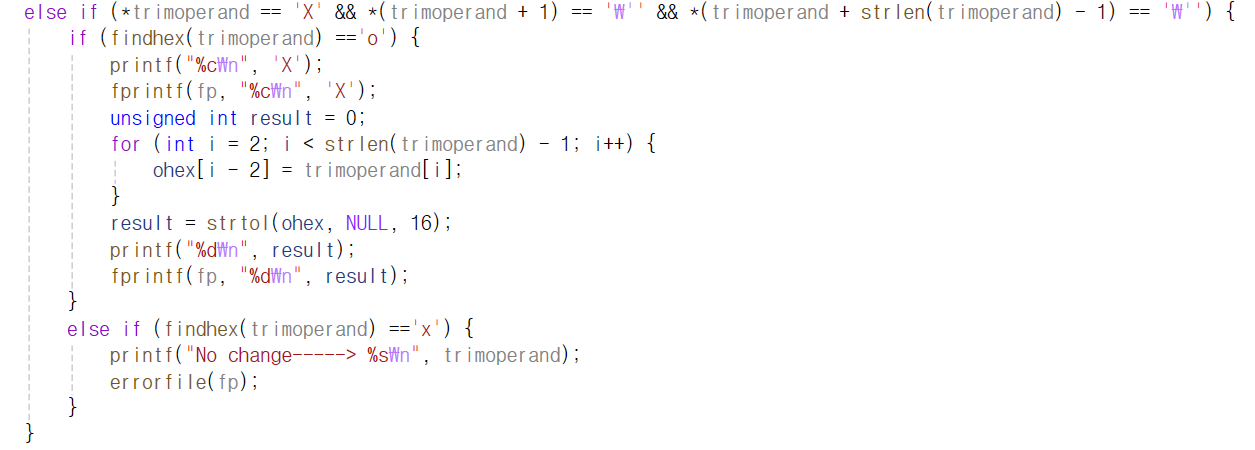
**-> hexfind.h**

**->hexfind.c**

**위 코드에 대해 설명하자면 trimoperand의 두번째 문장부터 끝에서부터 한 칸까지**

**Ex)**

**X’F1’에 F,1 만 판별하는 건데 hexlist에 들어있는 문자가 없으면 ok라는 정수 값은 strlen(str)-3을 가지지 못하고 (여기서 strlen(str)-3이나면 X’F1’에 X와 양쪽 ‘의 개수를 제외한 것이다.) x의 문자값을 반환하는데**

**->(ex-)**

**(ex-)코드를 보자면 x이면 errofile을 실행시켜 실행 오류를 시키는 것이다.**

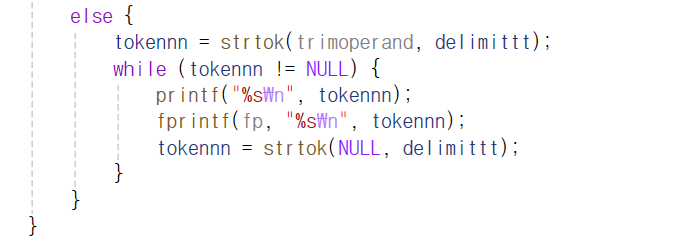
**만약에 o이면 양쪽’에 있는 16진수를 10진수로 변환해 아래와 같은 결과를 출력한다.**

**(X’F1’일 때 아래 예시)**

**X**

**241**

**Simcount 함수에 이어서 마지막 부분인**

****

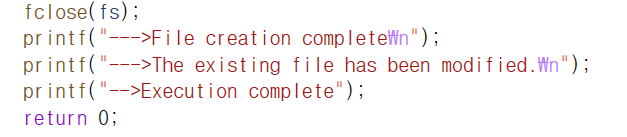
**는 줄바꿈과 ,를 기준으로 토큰을 생성해 출력하고 출력결과를 저장하는 코드이다.**

**Ex] BUFFER,X**

**BUFFER**

**X**

**마지막으로 그림11에 이어서 그림 12코드는 아래와 같다.**

**->그림 12**

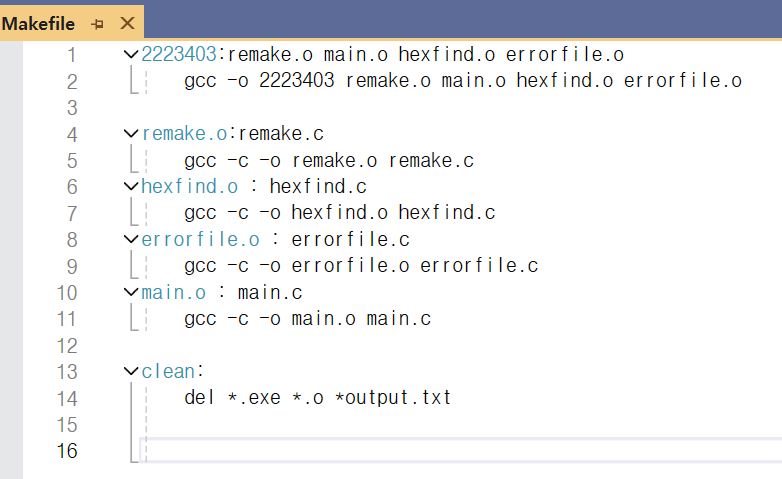
**그림 12는 파일을 다 읽었으면 while (fgets(str, sizeof(str), fs) != NULL) {}의 {}부분을 나가**

**fs파일을 닫고 파일생성을 완료했다는 문구와 프로그램이 성공적으로 수행되었음을 알리는**

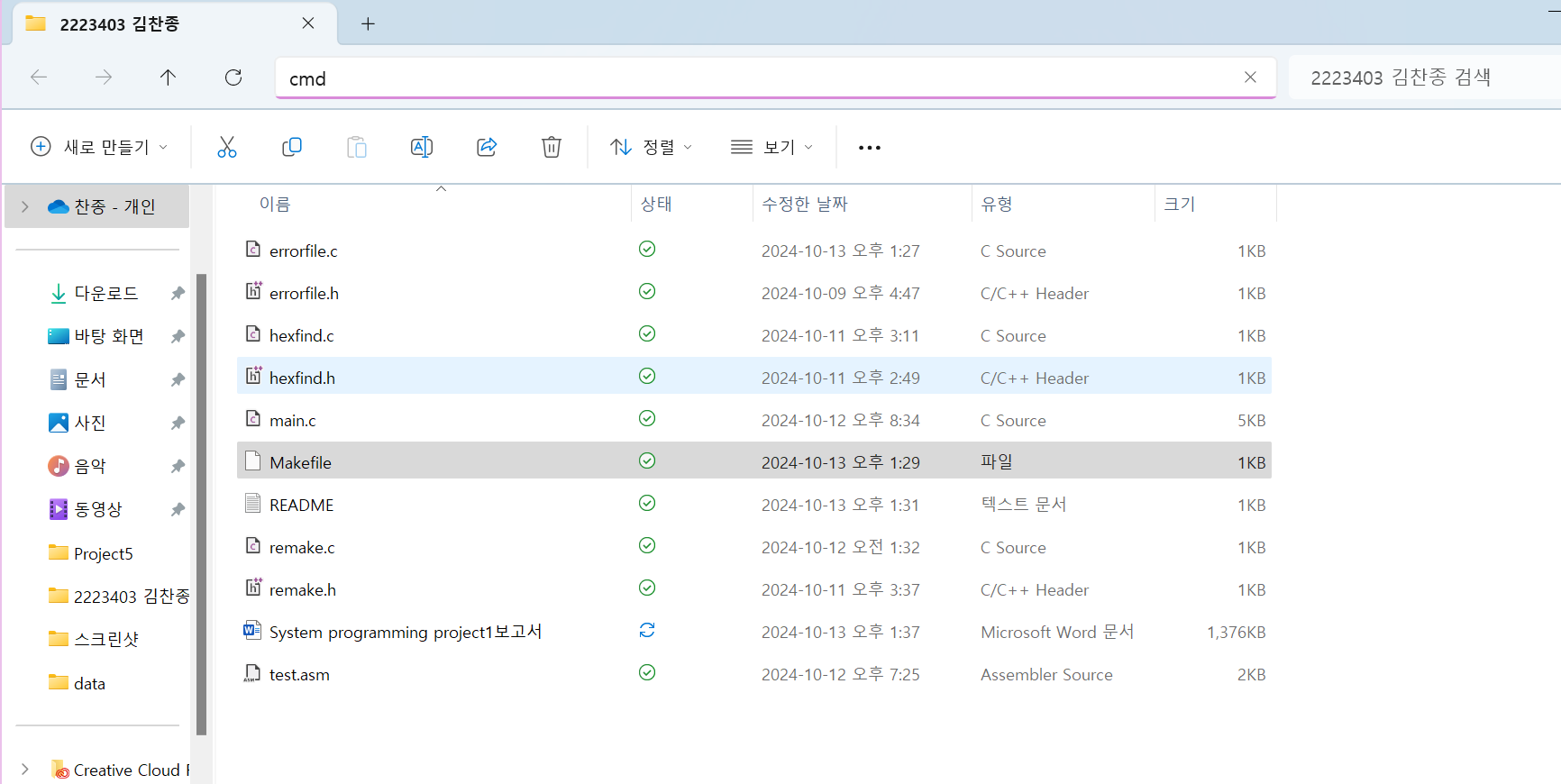
**return 0; 코드를 작성했다.**

**실행과정**

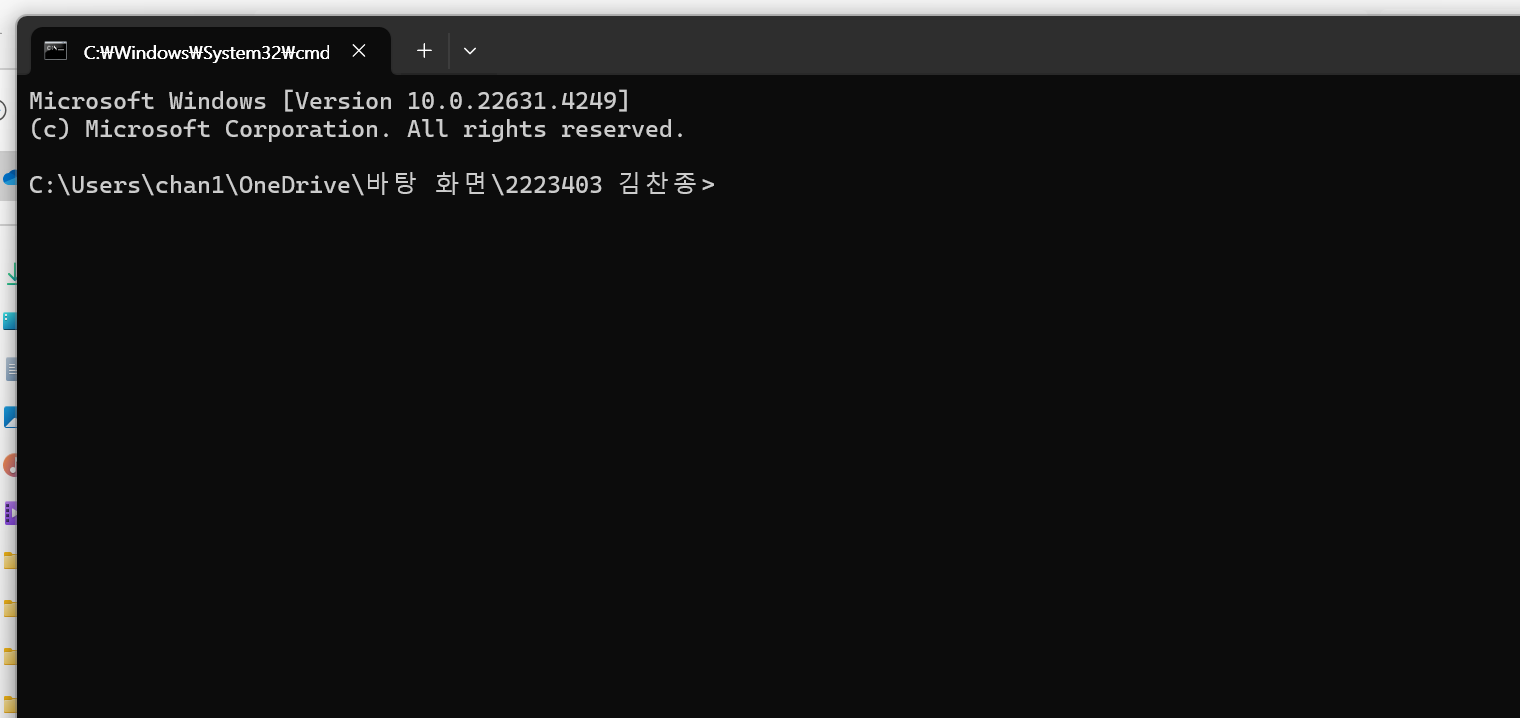
**Makefile 만들기**

****

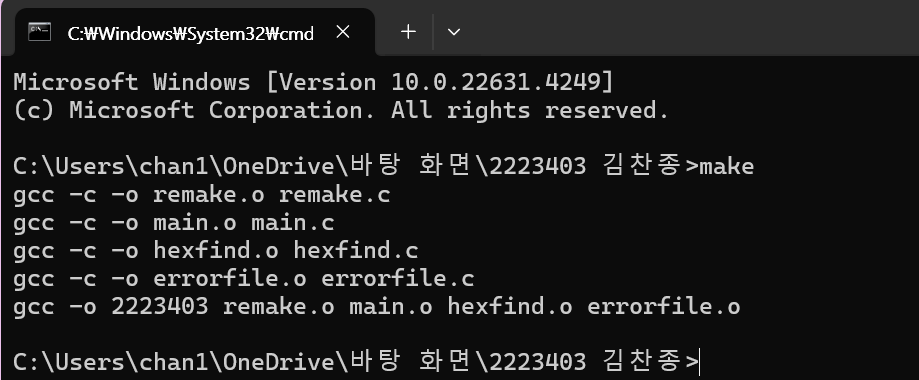
**실행하기**

****

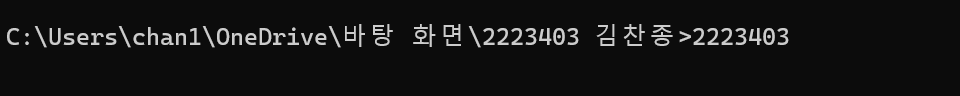
**cmd입력 아래와 같이 cmd창이 뜹니다.**

****

**여기서 make를 입력하면 2223403.exe가 생성됨**

****

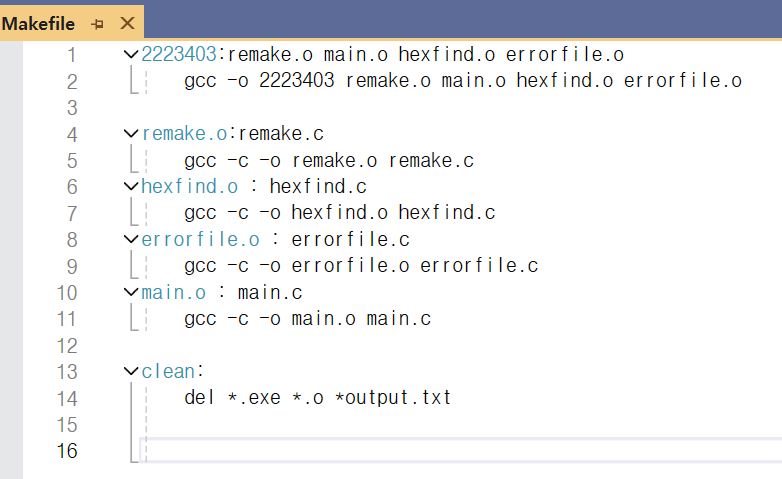
**그 다음 2223403를 입력하면 출력 이 됩니다.**

****

****

**그 다음 make clean을 입력하면**

****

**참조1**

**참조1에 보시면 .exe파일 .o output.txt가 삭제 됩니다.**

**전체 코드**

**(main.c)**

**#include <stdio.h>**

**#include <string.h>**

**#include <stdlib.h>**

**#include "remake.h"**

**#include "hexfind.h"**

**#include "errorfile.h"**

**#define MAX\_LEN 81**

**void StringCopy(char\* str, char\* copy);**

**void labelcopy(char\* copy, char\* label, char\* OPeratoroperand, char\* senevcolumn);**

**void divideOPcode(char\* trimOPeratoroperand, char\* operator, char\* operand);**

**void simcount(char\* trimoperand, FILE\* fp);**

**void errorlabel(char\* trimlabel, FILE\* fp);**

**int main(void) {**

**FILE\* fs;**

**FILE\* fp;**

**char str[MAX\_LEN] = " ";**

**char delimit[] = "\n";**

**char delimitt[] = " \n";**

**char columndelit[] = "\n";**

**char\* token = "";**

**char\* tokenn = "";**

**char\* columntokenn = " ";**

**fs = fopen("test.asm", "r");**

**fp = fopen("output.txt", "w");**

**while (fgets(str, sizeof(str), fs) != NULL) {**

**char copy[MAX\_LEN] = " ";**

**char copylabel[7] = " ";**

**char trimlabel[7] = " ";**

**char label[7] = "";**

**char senevcolumn[2] = "";**

**char OPeratoroperand[74] = "";**

**char trimOPeratoroperand[75] = "";**

**char operator[37] = "";**

**char operand[38] = "";**

**char trimoperator[37] = "";**

**char trimoperand[38] = "";**

**int error = 0;**

**StringCopy(str, copy);**

**labelcopy(copy, label, OPeratoroperand, senevcolumn);**

**strcpy(trimlabel, trim(label));**

**errorlabel(trimlabel, fp);**

**token = strtok(trimlabel, delimit);**

**while (token != NULL) {**

**printf("%s\n", token);**

**fprintf(fp, "%s\n", token);**

**token = strtok(NULL, delimit);**

**}**

**columntokenn = strtok(senevcolumn, columndelit);**

**while (columntokenn != NULL) {**

**printf("%s\n", columntokenn);**

**fprintf(fp, "%s\n", columntokenn);**

**columntokenn = strtok(NULL, columndelit);**

**}**

**strcpy(trimOPeratoroperand, trim(OPeratoroperand));**

**divideOPcode(trimOPeratoroperand, operator, operand);**

**strcpy(trimoperator, trim(operator));**

**tokenn = strtok(trimoperator, delimitt);**

**while (tokenn != NULL) {**

**printf("%s\n", tokenn);**

**fprintf(fp, "%s\n", tokenn);**

**tokenn = strtok(NULL, delimitt);**

**}**

**strcpy(trimoperand, trim(operand));**

**simcount(trimoperand, fp);**

**}**

**fclose(fs);**

**printf("--->File creation complete\n");**

**printf("--->The existing file has been modified.\n");**

**printf("-->Execution complete");**

**return 0;**

**}**

**void errorlabel(char\* trimlabel,FILE\* fp) {**

**for (int i = 0; i < strlen(trimlabel); i++) {**

**if (\*(trimlabel + i - 1) == ' ') {**

**printf("Syntaxerror-----> %s\n", trimlabel);**

**errorfile(fp);**

**}**

**}**

**}**

**void StringCopy(char\* str, char\* copy) {**

**while (\*str != 0) {**

**if (\*(str - 1) == '.') {**

**break;**

**}**

**else {**

**\*(copy) = \*(str);**

**str++;**

**copy++;**

**}**

**}**

**\*copy = '\0';**

**}**

**void labelcopy(char\* copy, char\* label, char\* OPeratoroperand, char\* senevcolumn) {**

**for (int i = 0; i < strlen(copy); i++) {**

**if (i < 6) {**

**\*(label + i) = \*(copy + i);**

**}**

**else if (i == 6) {**

**if (\*(copy + i) == '.') {**

**\*(senevcolumn + (i-6)) = \*(copy + i);**

**}**

**}**

**else if (i >=7) {**

**\*(OPeratoroperand + (i - 7)) = \*(copy + i);**

**}**

**}**

**}**

**void divideOPcode(char\* trimOPeratoroperand, char\* operator, char\* operand) {**

**int count = 0;**

**int i = 0;**

**while (trimOPeratoroperand[i] != 0) {**

**if (trimOPeratoroperand[i - 1] == ' ') {**

**count++;**

**break;**

**}**

**else {**

**\*(operator + i) = \*(trimOPeratoroperand + i);**

**i++;**

**}**

**}**

**if (count == 1) {**

**for (int k = i; k <= strlen(trimOPeratoroperand); k++) {**

**\*(operand + k - i) = trimOPeratoroperand[k];**

**}**

**}**

**}**

**void simcount(char\* trimoperand, FILE\* fp) {**

**char delimittt[] = "\n,";**

**char\* tokennn = "";**

**int size = strlen(trimoperand) - 3;**

**char ohex[size];**

**if (\*trimoperand == 'C' && \*(trimoperand + 1) == '\'' && \*(trimoperand + strlen(trimoperand) - 1) == '\'') {**

**printf("%c\n", 'C');**

**fprintf(fp, "%c\n", 'C');**

**for (int i = 2; i < strlen(trimoperand) - 1; i++) {**

**printf("%c", \*(trimoperand + i));**

**fprintf(fp, "%c", \*(trimoperand + i));**

**}**

**printf("\n");**

**fprintf(fp, "\n", ' ');**

**}**

**else if (\*trimoperand == 'X' && \*(trimoperand + 1) == '\'' && \*(trimoperand + strlen(trimoperand) - 1) == '\'') {**

**if (findhex(trimoperand) =='o') {**

**printf("%c\n", 'X');**

**fprintf(fp, "%c\n", 'X');**

**unsigned int result = 0;**

**for (int i = 2; i < strlen(trimoperand) - 1; i++) {**

**ohex[i - 2] = trimoperand[i];**

**}**

**result = strtol(ohex, NULL, 16);**

**printf("%d\n", result);**

**fprintf(fp, "%d\n", result);**

**}**

**else if (findhex(trimoperand) =='x') {**

**printf("No change-----> %s\n", trimoperand);**

**errorfile(fp);**

**}**

**}**

**else {**

**tokennn = strtok(trimoperand, delimittt);**

**while (tokennn != NULL) {**

**printf("%s\n", tokennn);**

**fprintf(fp, "%s\n", tokennn);**

**tokennn = strtok(NULL, delimittt);**

**}**

**}**

**}**

**(remake.h)**

**#pragma once**

**char\* trim\_left(char\* sr);**

**char\* trim\_right(char\* sr);**

**char\* trim(char\* sr);**

**(remake.c)**

**#pragma warning(disable : 4996)**

**#include "remake.h"**

**#include <stdio.h>**

**#include <ctype.h>**

**#include <string.h>**

**#include<stdlib.h>**

**// 함수 원형 선언**

**char\* trim\_left(char\* sr);**

**char\* trim\_right(char\* sr);**

**char\* trim(char\* sr);**

**char\* trim\_left(char\* sr){**

**while (\*sr) {**

**if (isspace(\*sr)) {**

**sr++;**

**}**

**else {**

**break;**

**}**

**}**

**return sr;**

**}**

**char\* trim\_right(char\* sr) {**

**int len = (int)strlen(sr) - 1;**

**while (len >= 0) {**

**if (isspace(\*(sr + len))) {**

**len--;**

**}**

**else {**

**break;**

**}**

**}**

**\*(sr + ++len) = '\0';**

**return sr;**

**}**

**char\* trim(char\* sr) {**

**return trim\_left(trim\_right(sr));**

**}**

**(hexfind.h)**

**#pragma once**

**char findhex(char\* str);**

**(hexfind.c)**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include "hexfind.h"**

**char findhex(char\* str);**

**char findhex(char\* str) {**

**char hexlist[17] = { '1','2','3','4','5','6','7','8','9','0','A','B','C','D','E','F' };**

**int ok = 0;**

**for (int i = 2; i < strlen(str) - 1; i++) {**

**for (int k = 0; k < strlen(hexlist); k++) {**

**if (\*(str + i) == \*(hexlist + k)) {**

**ok++;**

**}**

**}**

**}**

**if (ok == strlen(str) - 3) {**

**return 'o';**

**}**

**else if (ok < strlen(str) - 3) {**

**return 'x';**

**}**

**}**

**(errofile.h)**

**#pragma once**

**void errorfile(FILE\* fp);**

**(errorfile.c)**

**#include <stdio.h>**

**#include <stdlib.h>**

**void errorfile(FILE\* fp) {**

**fclose(fp);**

**char strPath[] = { "output.txt" };**

**remove(strPath);**

**printf("-->File creation failed");**

**printf(" or--\n");**

**printf("--->The existing file has been deleted.");**

**exit(1);**

**}**

**Makefile**

**2223403:remake.o main.o hexfind.o errorfile.o**

**gcc -o 2223403 remake.o main.o hexfind.o errorfile.o**

**remake.o:remake.c**

**gcc -c -o remake.o remake.c**

**hexfind.o : hexfind.c**

**gcc -c -o hexfind.o hexfind.c**

**errorfile.o : errorfile.c**

**gcc -c -o errorfile.o errorfile.c**

**main.o : main.c**

**gcc -c -o main.o main.c**

**clean:**

**del \*.exe \*.o \*output.txt**