컴퓨터학부 20142468 허경영

1. 과제개요

ssu\_convert는 입력으로 주어진 java파일을 C파일로 자동 변환해주는 프로그램이다. 프로그램은 옵션 없이 실행될 때 java 파일을 c파일로 변환한다. 변환하기 위해서 java파일을 예약어, 연산자, 그 외의 문자로 파싱하는 과정이 필요하다. java와 c에서 사용하는 함수가 다른 경우가 있다. 그 경우엔 개발자가 미리 만든 참조 헤더 테이블과 사전에 정의된 자료를 이용해서 헤더를 추가하고, 함수를 변환한다. java파일은 여러 개의 클래스로 이루어져 있다. 그렇기 때문에 c 파일로 변환한 이후 여러 클래스, 즉 파일을 같이 컴파일 해야만 하는 경우가 있다. 이를 위해 ssu\_convert에서 makefile을 자동으로 생성해서 java파일을 c파일로 실행 가능하게 해야한다.

주목할 만한 옵션은 r 옵션이다. 자식 프로세스를 생성하여 java 프로그램이 c 프로그램으로 변환되는 과정을 한줄 한줄 출력하는데 터미널을 clear 하면서 과정을 보이는 옵션이다. 이를 위해 fork와 wait을 사용한다.

2. 설계

parsefile에서 파일을 파싱하고 이때 charType으로 문자의 타입을 처리한다.

javatoC에서 자바파일을 C파일로 바꾸고 writeC에서 c파일을 작성한다.

ReadHeaderTable createMakefile 에서 헤더테이블을 다룬다.

3. 구현

int charType(char);

void parsefile(void);

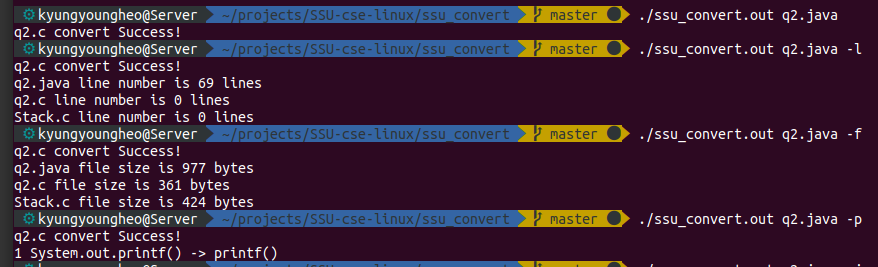
void javaToC(void);

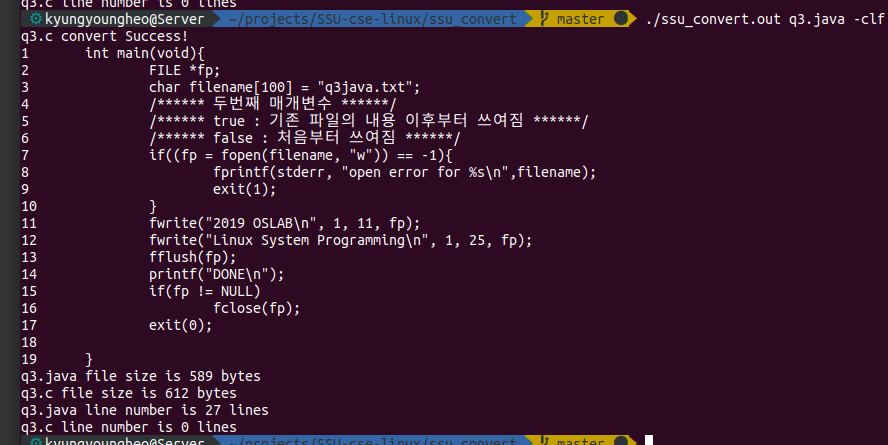
void writeC(void);

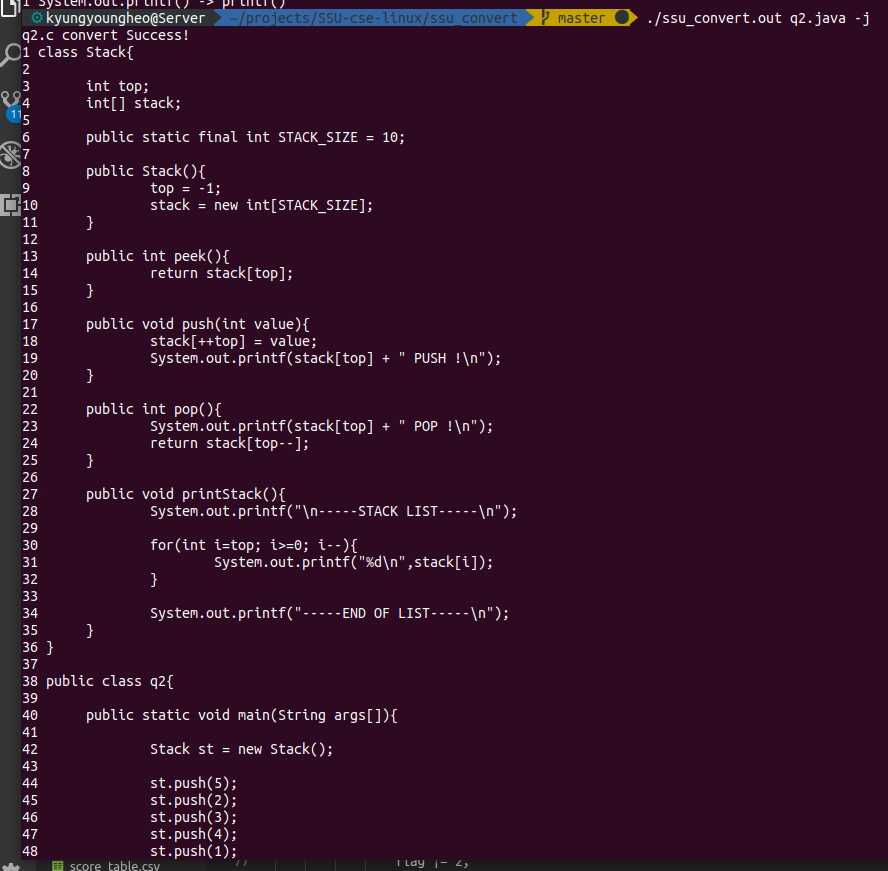
void readHeaderTable(void);

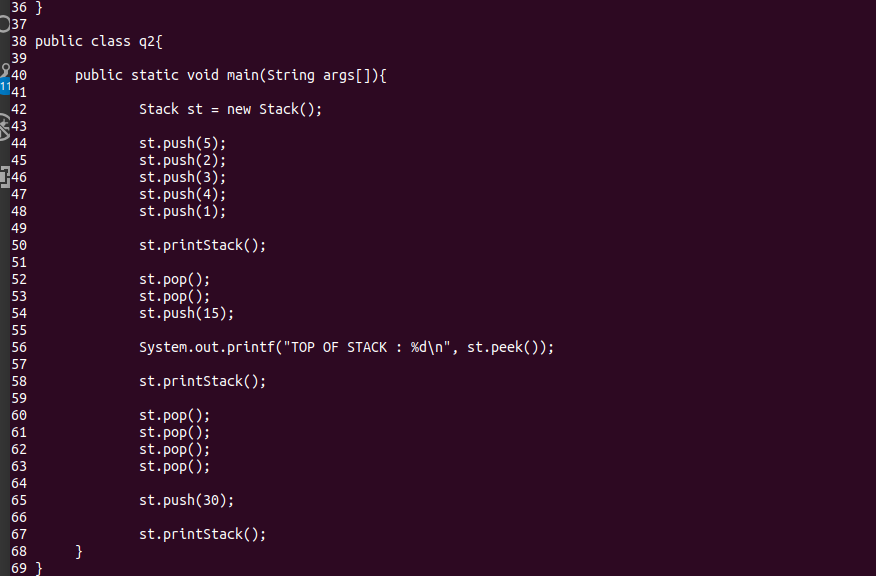
void createMakefile(void);

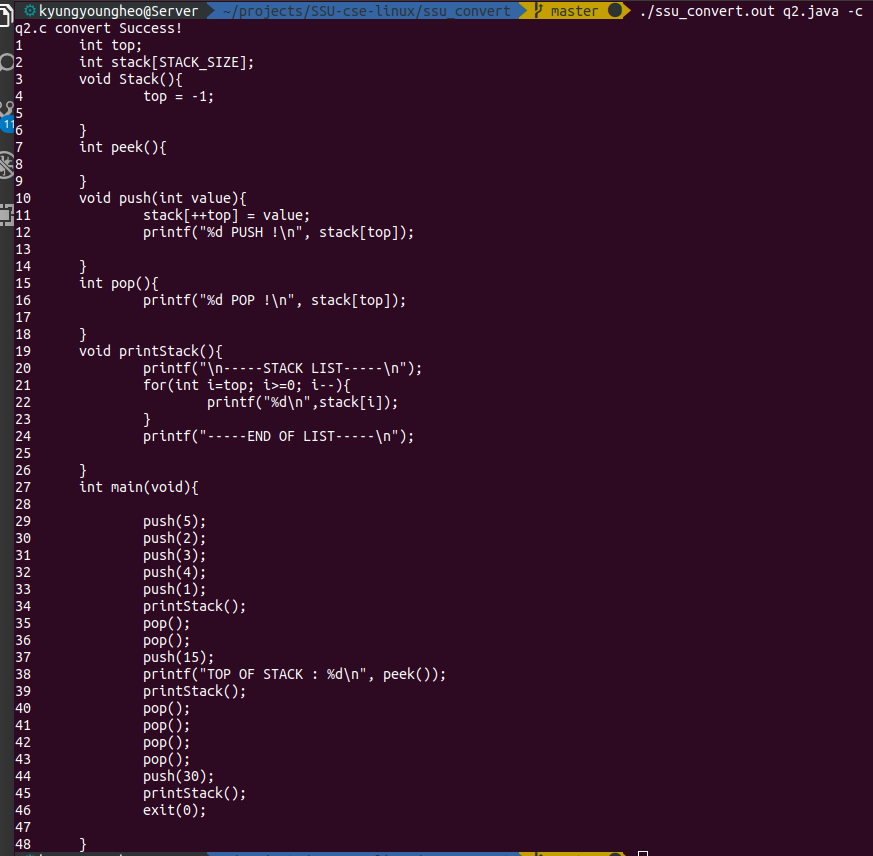
4. 테스트 및 결과



****

****

****

****

5. 소스코드

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

#include <time.h>

#include <sys/stat.h>

#include <sys/time.h>

#include <sys/types.h>

#include <sys/wait.h>

#define EOF -1

#define CHAR 0

#define DIGIT 1

#define OTHERS 2

int charType(char);

void parsefile(void);

void javaToC(void);

void writeC(void);

void readHeaderTable(void);

void createMakefile(void);

int wordsAtLine[100], lines=0, wline=0;

int mainline;

char buf[2000], word[500];

char cstr[100][100][50]; // 읽어온 string cstr[line개수][word개수][word의 크기]

char wbuf[100][500]; // .c파일에 쓸 write buf[line의 개수][line의 크기]

char filename[50], nextWord[50], searchWord[50];

char stackclassVar[50], filrWriterVar[50], fileVar[50], stackVar[50], scannerVar[50], inputVar[50];

char definebuf[100]; // define buffer

char stackfunctionbuf[100];

int flag = 0; // opt flag

char targetfilename[50]; // q1.java, q2.java, q3.java in argc[1]

char headerkey[10][20];

char headervalue[10][100];

/\*

// [0] FileWriter : fopen open #include <fcntl.h>

// [1] always. exit #include <stdlib.h>

// [2] always. printf #include <stdio.h>

\*/

FILE \*fp, \*wfp, \*cfp;

FILE \*hfp; // header table file pointer

FILE \*mfp; // makefile file pointer

FILE \*stkfp; // stack.c file pointer

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

int opt;

int javasize, csize, stacksize;

pid\_t pid;

int ret\_val, status;

// Handling options

/\*

j

c

p

f

l

r

\*/

if(argc <2){

printf("usage: ./ssu\_convert <filename>\n");

exit(1);

}

sprintf(targetfilename, "%s", argv[1]);

fp = fopen(targetfilename, "r");

while((opt = getopt(argc, argv, "jcpflr")) != -1){

switch(opt){

case 'r':

flag |= 2;

break;

case 'j':

flag |= 4;

break;

case 'c':

flag |= 8;

break;

case 'p':

flag |= 16;

break;

case 'f':

flag |= 32;

break;

case 'l':

flag |= 64;

break;

}

}

// 자바파일 읽는다.

// 파싱한다.

parsefile();

readHeaderTable();

javaToC();

writeC();

// 버퍼에 넣는다.

// makefile 만든다

// C파일 만든다

//

// 옵션 없는 경우

if(flag & 1){

printf("%s.c convert Success!\n", filename);

}

// -r

if(flag & 2){

int max;

max = lines > wline ? lines : wline;

printf("%d\n",max);

for(int i=0; i<max; i++){

system("clear");

if((pid = fork()) == 0){ // 자식프로세스

for(int j=0; j<i; j++){

printf("%d ", j+1);

for(int k=0; k<wordsAtLine[j]; k++){

printf("%s",cstr[j][k]);

}

printf("\n");

}

sleep(1);

exit(0);

}

if(pid > 0){

for(int j=0; j<i; j++){

printf("%d %s\n", j+1, wbuf[j]);

}

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

ret\_val = wait(&status);

}

}

}

// -j

if(flag & 4){

for(int i=0; i<lines; i++){

printf("%d ",i+1);

for(int j=0; j<wordsAtLine[i]; j++){

printf("%s",cstr[i][j]);

}

printf("\n");

}

}

// -c

if(flag & 8){

for(int i=0; i<wline; i++){

printf("%d %s\n",i+1, wbuf[i]);

}

}

// -p

if(flag & 16){

if(strcmp(filename, "q1") == 0){

printf("1 System.out.printf() -> printf()\n");

printf("2 scn.nextInt() -> scanf()\n");

}

else if(strcmp(filename, "q2") == 0){

printf("1 System.out.printf() -> printf()\n");

}

else if(strcmp(filename, "q3") == 0){

printf("1 System.out.printf() -> printf()\n");

printf("2 new FileWriter() -> fopen()\n");

printf("3 writer.write() -> fwrite()\n");

printf("4 writer.flush() -> fflush()\n");

printf("5 writer.close() -> fclose()\n");

}

}

// -f

if(flag & 32){

fseek(wfp, 0, SEEK\_SET);

fseek(wfp, 0, SEEK\_END);

csize = ftell(wfp);

fseek(fp, 0, SEEK\_SET);

fseek(fp, 0, SEEK\_END);

javasize = ftell(fp);

printf("%s.java file size is %d bytes\n", filename, javasize);

printf("%s.c file size is %d bytes\n", filename, csize);

if(stkfp != 0){

fseek(stkfp, 0, SEEK\_SET);

fseek(stkfp, 0, SEEK\_END);

stacksize = ftell(stkfp);

printf("%s.c file size is %d bytes\n", stackclassVar, stacksize);

}

}

// -l

if(flag & 64){

rewind(fp);

int cline=0, javaline = 0, stackline = 0;

char ch;

fseek(wfp, 0, SEEK\_SET);

rewind(wfp);

while(1){

ch = fgetc(wfp);

if(ch == '\n'){

cline++;

}

if(ch == EOF) break;

}

while(1){

ch = fgetc(fp);

if(ch == EOF) break;

if(ch == '\n'){

javaline++;

}

}

printf("%s.java line number is %d lines\n", filename, javaline);

printf("%s.c line number is %d lines\n", filename, cline);

if(stkfp != 0){

while(1){

ch = fgetc(stkfp);

if(ch == EOF) break;

if(ch == '\n'){

stackline++;

}

}

rewind(stkfp);

printf("%s.c line number is %d lines\n", stackclassVar, stackline);

}

}

exit(0);

}

void javaToC(void){

int len, keyIdx;

int wlen;

char cmp1[50], cmp2[50], cmp3[50], cmp4[50], cmp5[50], cmp6[50], cmp7[50], cmp8[50], cmp9[50];

strcpy(headerkey[1], "exit");

strcpy(headerkey[2], "printf");

for(int i=0; i<lines; i++){

len = wordsAtLine[i];

memset(cmp1, 0, 50);

memset(cmp2, 0, 50);

memset(cmp3, 0, 50);

memset(cmp4, 0, 50);

memset(cmp5, 0, 50);

memset(cmp6, 0, 50);

memset(cmp7, 0, 50);

memset(cmp8, 0, 50);

memset(cmp9, 0, 50);

strcpy(cmp1, cstr[i][0]);

if(len>=8)

strncpy(cmp9, cstr[i][8], 50);

if(len>=7)

strncpy(cmp8, cstr[i][7], 50);

if(len>=6)

strncpy(cmp7, cstr[i][6], 50);

if(len>=5)

strncpy(cmp6, cstr[i][5], 50);

if(len>=4)

strncpy(cmp5, cstr[i][4], 50);

if(len>=3)

strncpy(cmp4, cstr[i][3], 50);

if(len>=2)

strncpy(cmp3, cstr[i][2], 50);

if(len>=1)

strncpy(cmp2, cstr[i][1], sizeof(cmp2));

// strstr을 사용하면 현재 키워드의 위치를 알 수 있지만

// 원하는 단어를 찾기 위해서 parsing이 이후에 필요하다.

// 지금 구현방법과 크게 달라지지 않을듯.

// printf("%d %d \n1:%s 2:%s 3:%s 4:%s 5:%s 6:%s 7:%s 8:%s 9:%s\n", i, len, cmp1, cmp2, cmp3, cmp4, cmp5, cmp6, cmp7, cmp8, cmp9);

// printf("%d %d\n", i, len);

// cmp1 == import

if(strcmp(cmp1, "import") == 0){

if(strcmp(cmp7, "Scanner") == 0){

strcpy(headerkey[2], "printf");

}

else if(strcmp(cmp7, "FileWriter") == 0){

strcpy(headerkey[0], "open");

}

}

// cmp1 == public

else if(strcmp(cmp1, "public") == 0){

// cmp2 == class

if(strcmp(cmp3, "class") == 0){

// filename is q1, q2, q3

strcpy(filename, cmp5);

}

}

// cmp1 == class

else if(strcmp(cmp1, "class") == 0){

// cmp3 == Stack

// Stack.c 파일 포인터 열고 새 파일 만들어야함.

// stkfp와 wfp 를 분리해서 wbuf에서 써야함. line 카운트 별도로!

strcpy(stackclassVar, cmp3);

char stackfilename[50];

sprintf(stackfilename, "%s.c",stackclassVar);

stkfp = fopen(stackfilename, "w");

}

else if(strcmp(cmp1, "\t") == 0){

if(strcmp(cmp2, "public") == 0){

if(strcmp(cmp4, "static") == 0){

// cmp3 == void

if(strcmp(cmp6, "void") == 0){

// cmp4 == main

if(strcmp(cmp8, "main") == 0){

// copy int main(void){

strcpy(wbuf[wline], "\tint main(void){");

// printf("%d %d %d\n", i, len, wline);

mainline = wline;

wline++;

}

}

else if(strcmp(cmp6, "final") == 0){

sprintf(definebuf, "#define %s%s%s %s", cstr[i][9], cstr[i][10], cstr[i][11], cstr[i][15]);

// printf("%s 9:%s 10:%s 11:%s 15:%s\n",definebuf, cstr[i][9], cstr[i][10], cstr[i][11], cstr[i][15]);

}

}

// cmp2 == int || cmp2 == void. public functions

else if(strcmp(cmp4, "int") == 0 || strcmp(cmp4, "void") == 0){

strcat(wbuf[wline], "\t");

for(int j=3; j<len; j++){

strcat(wbuf[wline], cstr[i][j]);

if(j != len-1)

strcat(stackfunctionbuf, cstr[i][j]);

}

strcat(stackfunctionbuf, ";\n");

wline++;

}

// cmp2 == 'Stack' initializer

else if(strcmp(cmp4, stackclassVar) == 0){

// printf("생성자\n");

strcat(wbuf[wline], "\tvoid ");

for(int j=3; j<len; j++){

strcat(wbuf[wline], cstr[i][j]);

}

wline++;

}

// printf("스택클래스변수%s\n", stackclassVar);

}

// cmp1 == System

else if(strcmp(cmp2, "\t") == 0){

if(strcmp(cmp3, "Scanner") == 0){

continue;

}

else if(strcmp(cmp3, "\t") == 0){

if(strcmp(cmp6, "close") == 0){

strcat(wbuf[wline], "\t\t\t");

strcat(wbuf[wline], "fclose(fp);");

wline++;

}

else if(strcmp(cmp4, "System") == 0){

strcat(wbuf[wline], "\t\t\t");

for(int j=7; j<len; j++){

strcat(wbuf[wline], cstr[i][j]);

}

wline++;

}

else{

for(int j=0; j<len; j++){

strcat(wbuf[wline], cstr[i][j]);

}

wline++;

}

}

else if(strcmp(cmp3, "System") == 0){

// cmp3 == out

if(strcmp(cmp5, "out") == 0){

// cmp5 == printf

if(strcmp(cmp7, "printf") == 0){

// printf("%s\n", cstr[i][6]);

// printf("프린트\n");

// printf("") 인 경우

if(strcmp(cstr[i][8], "\"") == 0){

strcat(wbuf[wline], "\t\t");

for(int j=6; j<len; j++){

// st.peek()포함하는 경우

if(strcmp(cstr[i][j+1], "peek") == 0 ||strcmp(cstr[i][j+2], "peek") == 0 )

continue;

strcat(wbuf[wline], cstr[i][j]);

}

// printf("프린트\n%s\n",wbuf[wline]);

// printf("wbuf = %s\n",wbuf[wline]);

wline++;

}

// printf(stack[top] + "") 인 경우

else {

char printbuf[50];

memset(printbuf, 0, 50);

for(int j=8; j<12; j++){

strcat(printbuf, cstr[i][j]);

}

// printf("프린트에프 %s\n", printbuf);

sprintf(wbuf[wline], "\t\t%s%s\"%%d", cstr[i][6], cstr[i][7]);

for(int j=16; j<22; j++){

strcat(wbuf[wline], cstr[i][j]);

}

sprintf(wbuf[wline], "%s\", %s);", wbuf[wline],printbuf);

wline++;

}

}

}

}

else if(strcmp(cmp9, "nextInt") == 0){

strcat(wbuf[wline], "\t\tscanf(\"%d\", &");

strcat(wbuf[wline], cmp3);

strcat(wbuf[wline], ");");

wline++;

}

// cmp1 == Stack q2 main에서 Stack st = new Stack()하는 부분.

else if(strcmp(cmp3, stackclassVar) == 0){

}

// File

else if(strcmp(cmp3, "File") == 0){

strcat(wbuf[wline], "\t\tFILE \*fp;");

wline++;

strcat(wbuf[wline], "\t\tchar filename[100] = \"");

for(int j=13; j<len-2; j++){

strcat(wbuf[wline], cstr[i][j]);

}

strcat(wbuf[wline], ";");

wline++;

}

// FileWriter

else if(strcmp(cmp3, "FileWriter") == 0){

// 두번째 인자가 false

strcat(wbuf[wline], "\t\tif((fp = fopen(filename, \"");

if(strcmp(cstr[i][len-3], "false") == 0){

strcat(wbuf[wline], "w\"");

}

// 두번째 인자가 true

else if(strcmp(cstr[i][len-2], "true") == 0){

strcat(wbuf[wline], "a\"");

}

strcat(wbuf[wline], ")) == -1){");

wline++;

strcat(wbuf[wline], "\t\t\tfprintf(stderr, \"open error for %s\\n\",filename);");

wline++;

strcat(wbuf[wline], "\t\t\texit(1);");

wline++;

strcat(wbuf[wline], "\t\t}");

wline++;

}

// writer.write

else if(strcmp(cmp5, "write") == 0){

int size=0;

char strsize[10];

strcat(wbuf[wline], "\t\tfwrite(\"");

for(int j=7; j<len - 3; j++){

size += strlen(cstr[i][j]);

strcat(wbuf[wline], cstr[i][j]);

}

// printf("size = %d\n",size);

sprintf(strsize, "%d",size-1);

strcat(wbuf[wline], "\", 1, ");

strcat(wbuf[wline], strsize);

strcat(wbuf[wline], ", fp);");

wline++;

}

//writer.flush

else if(strcmp(cmp5, "flush") == 0){

strcat(wbuf[wline], "\t\tfflush(fp);");

wline++;

}

else if(strcmp(cmp5, "writer") == 0){

for(int j=0; j<len; j++){

if(j == 4){

strcat(wbuf[wline], "fp");

}

else if(j == 9){

strcat(wbuf[wline], "NULL");

}

else{

strcat(wbuf[wline], cstr[i][j]);

}

}

wline++;

}

// q2에서 이 부분 수정해야함.

else if(strcmp(cmp3, "return") == 0){

continue;

}

else if(strcmp(cmp5, "push") == 0 || strcmp(cmp5, "pop") == 0 || strcmp(cmp5, "printStack") == 0){

strcat(wbuf[wline], "\t\t");

for(int j=4; j<len; j++){

strcat(wbuf[wline], cstr[i][j]);

}

wline++;

}

else if(strcmp(cmp7, "new") == 0){

continue;

}

else {

for(int j=0; j<len; j++){

strcat(wbuf[wline], cstr[i][j]);

}

wline++;

}

}

else if(strcmp(cmp2, "int") == 0){

if(strcmp(cmp3, "[") == 0 && strcmp(cmp4, "]") == 0){

// 변수이름은 stack으로 고정

strcat(wbuf[wline], "\tint stack[STACK\_SIZE];");

wline++;

}

else{

for(int j=0; j<len; j++){

strcat(wbuf[wline], cstr[i][j]);

}

wline++;

}

}

else{

// strcat(wbuf[wline], "\t\treturn 0;");

wline++;

for(int j=0; j<len; j++){

strcat(wbuf[wline], cstr[i][j]);

}

// printf("wbuf = %s\n",wbuf[wline]);

wline++;

}

}

// printf("\n");

if(i == lines-3 ){

strcat(wbuf[wline], "\t\texit(0);");

wline++;

}

}

// printf("----------------\n");

// printf("filename : %s\n", filename);

// for(int i=0; i<wline; i++){

// printf("%s\n",wbuf[i]);

// }

}

void parsefile(void){

char ch;

int idx = 0;

while((ch = fgetc(fp)) != EOF){

// printf("%c",ch);

if(charType(ch) == CHAR){

word[idx++] = ch;

word[idx] = 0;

while(1){

ch = fgetc(fp);

if(ch == '\n'){

lines++;

continue;

}

if(charType(ch) == CHAR || charType(ch) == DIGIT){

word[idx] = ch;

idx++;

}

else{

word[idx] = 0;

idx = 0;

ungetc(ch, fp);

strcpy(cstr[lines][wordsAtLine[lines]], word);

wordsAtLine[lines]++;

strcat(buf, word);

break;

}

}

}

else if(charType(ch) == DIGIT){

word[idx++] = ch;

word[idx] = 0;

while(1){

ch = fgetc(fp);

if(ch == '\n'){

lines++;

continue;

}

if(charType(ch) == DIGIT){

word[idx++] = ch;

word[idx] = 0;

}

else{

word[idx] = 0;

idx = 0;

ungetc(ch, fp);

strcpy(cstr[lines][wordsAtLine[lines]], word);

wordsAtLine[lines]++;

strcat(buf, word);

break;

}

}

}

else if(charType(ch) == OTHERS){

if(ch == '\n'){

lines++;

continue;

}

word[idx++] = ch;

word[idx] = 0;

idx = 0;

strcat(buf, word);

strcpy(cstr[lines][wordsAtLine[lines]], word);

wordsAtLine[lines]++;

}

else{

// printf("EOF\n");

break;

//EOF

}

}

}

void readHeaderTable(void){

char ch;

char word[10];

char headerread[100];

int len=0;

int firstword = 1, firstheader = 1;

hfp = fopen("headerTable.txt", "r");

while((ch = fgetc(hfp)) != EOF){

if(firstword == 1){

// 공백 전까지 글자 입력받아서 word에 저장

while(ch != ' '){

word[len] = ch;

len++;

ch = fgetc(hfp);

}

word[len] = 0;

firstword = 0;

len = 0;

firstheader = 1;

// printf("\n");

}

else{

// 개행 전까지 글자 입력받아서 headerread에 저장

while(ch != '\n'){

if(ch == '#'){

if(firstheader == 1){

firstheader = 0;

}

else{

headerread[len] = '\n';

len++;

}

}

headerread[len] = ch;

len++;

ch = fgetc(hfp);

if(ch == -1)

break;

}

headerread[len] = '\n';

len++;

headerread[len] = 0;

firstword = 1;

len = 0;

// printf("\n");

// 개행이면 firstword = 1

}

// printf("%s %s\n",word,headerread);

if(strcmp(word, "open") == 0){

strcpy(headervalue[0], headerread);

}

else if(strcmp(word, "exit") == 0){

strcpy(headervalue[1], headerread);

}

if(strcmp(word, "printf") == 0){

strcpy(headervalue[2], headerread);

}

}

// 공백 이전까지 글자 읽는다.

// 읽은 단어가

// open

// exit

// printf

}

void writeC(){

char cfilename[50];

sprintf(cfilename, "%s.c",filename);

wfp = fopen(cfilename, "w");

if(stkfp != 0){

fwrite(headervalue[2], 1, strlen(headervalue[2]), stkfp);

fwrite("\n", 1, 1, stkfp);

if(strlen(definebuf) != 0){

fwrite(definebuf, 1, strlen(definebuf), stkfp);

}

fwrite("\n", 1, 1, stkfp);

for(int i=0; i<mainline; i++){

fwrite(wbuf[i], 1, strlen(wbuf[i]), stkfp);

fwrite("\n", 1, 1, stkfp);

}

}

// 헤더 검사해서 #include 추가.

if(strcmp(headerkey[0], "open") == 0){

//write headerread[0]

fwrite(headervalue[0], 1, strlen(headervalue[0]), wfp);

}

if(strcmp(headerkey[1], "exit") == 0){

//write headervalue[1]

fwrite(headervalue[1], 1, strlen(headervalue[1]), wfp);

}

if(strcmp(headerkey[2], "printf") == 0){

//write headervalue[2]

fwrite(headervalue[2], 1, strlen(headervalue[2]), wfp);

}

fwrite("\n", 1, 1, wfp);

// #define 추가

if(stkfp != 0){

fwrite(stackfunctionbuf, 1, strlen(stackfunctionbuf), wfp);

}

// main전까지 읽는다.

// 헤더는 메인에만 추가.

for(int i=mainline; i<wline; i++){

fwrite(wbuf[i], 1, strlen(wbuf[i]), wfp);

fwrite("\n", 1, 1, wfp);

}

createMakefile();

}

void createMakefile(){

char makefilename[100];

char writebuf[2000];

sprintf(makefilename, "%s\_Makefile", filename);

mfp = fopen(makefilename, "w+");

if(stkfp != 0){

sprintf(writebuf, "%s: %s.c %s.c\n\tgcc -o %s.out %s.c %s.c", filename, filename, stackclassVar, filename, filename, stackclassVar);

}

else{

sprintf(writebuf, "%s: %s.c\n\tgcc -o %s.out %s.c", filename, filename, filename, filename);

}

fwrite(writebuf, 1, strlen(writebuf), mfp);

}

int charType(char c){

if(isalpha(c)){

return CHAR;

}

else if(isdigit(c)){

// digit이 아닐 때까지 읽는다.

return DIGIT;

}

else if(c == -1){

return EOF;

}

else{

return OTHERS;

}

}