CFG Generator

Based on T5 SASD

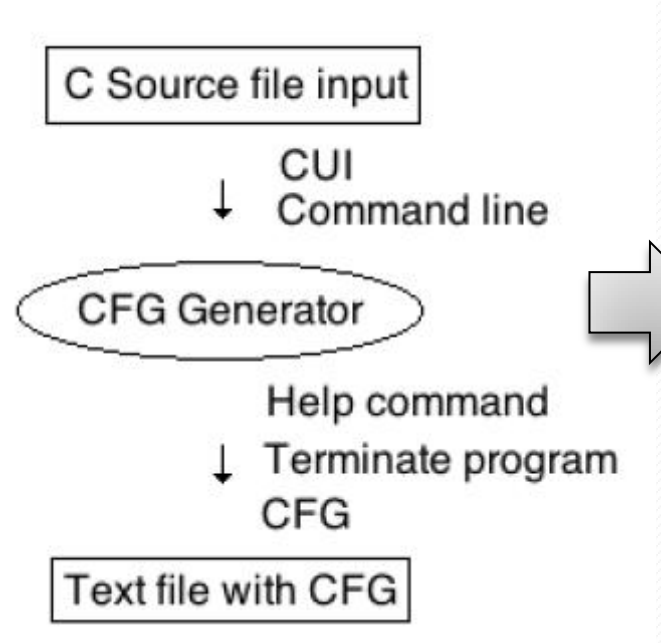
T4
200511306 김성훈
200710115 김철웅
200712692 진형남
201011316 김성엽

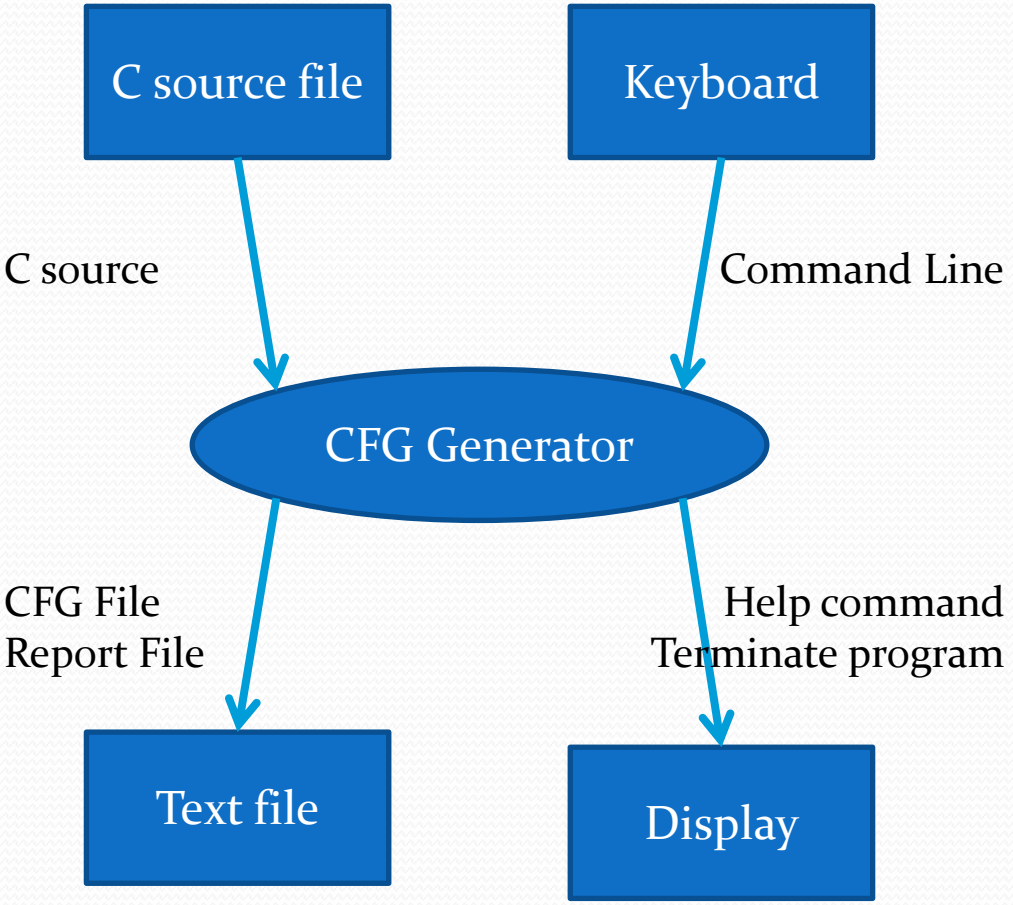
Statement of purpose

Draw a Control Flow Graph (CFG):

- A CFG is a graph of a source code for an easier understanding
- One input and one output
- The program receives a source code in input
- The program analyzes the source code and create a CFG
- The output is a CFG and a report
- The C source code doesn't have user defined header files.
- The C source code doesn't include pointers.

System Context





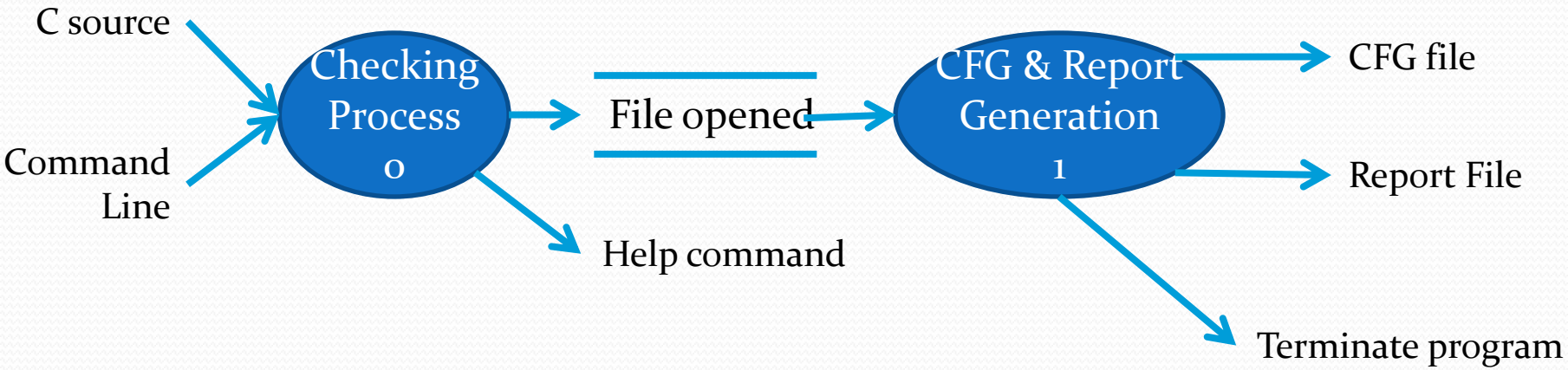
System Context Diagram

- Event list & Data Dictionary

Input / Output Event	Description	Type
Command Line	Inputted command for CFG generation [Command] [C source file name] [report file name]	String
C source	C File to be converted	C File
CFG File	TXT File with CFG information	TXT File
Report File	TXT File with Organized CFG Information	TXT File
Message	String to the monitor output	String
Terminate Program	String to output at the end of the program	String

Level 0


C source file input
Checking process
file opened
CFG & Report generation
Report
CFG file


C source
Command Line
Checking Process 0
File opened
Help command
CFG & Report Generation 1
CFG file
Report File
Terminate program

DFD Level 0

- Process Specification

Reference No.	0
Name	Checking Process
Input	C source file, Command Line
Output	File opened, Help Command
Process Description	Process command entered by the user , Print out Success or failure by reading C source that create CFG and Report File

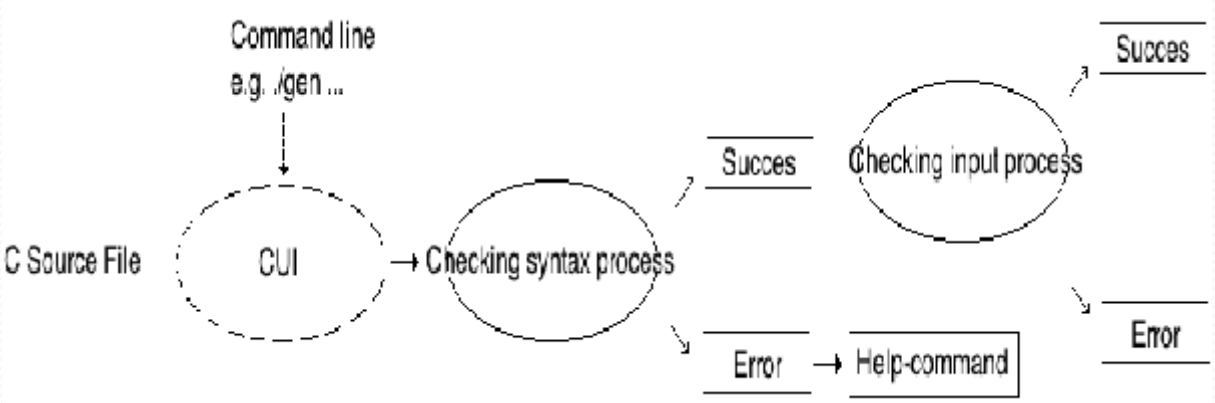
DFD Level 0

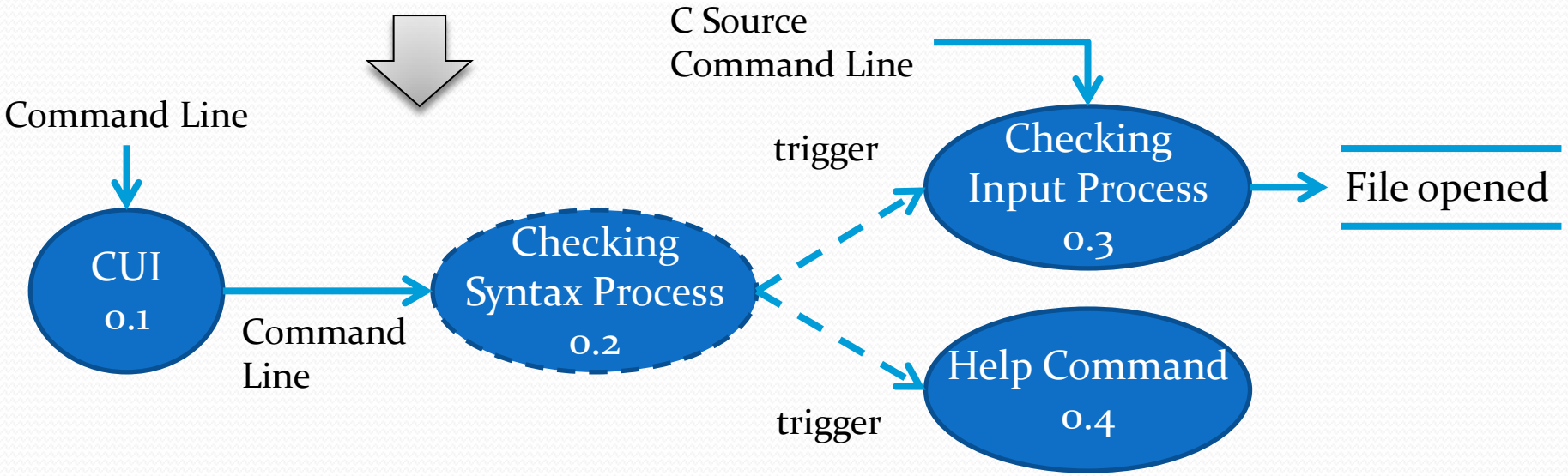
- Process Specification

Reference No.	1
Name	CFG & Report Generation
Input	File opened
Output	CFG file, Report file, Terminate Program
Process Description	Create CFG and Report Information By C source file and Print out txt File, Print out process of performing to CUI , if command entered by the user fail, notify program End or beginning of conversion. After creation of CFG and Report, print out txt File name to console

Level 1

- Checking Process 0





Level 1

- Checking Process o Code

```c
int main( ){
    char* cml = NULL;
    int ret;
    inFunction = FALSE;

    do{
        cml = CUI( );
        ret = CheckSyntax(cml);
    }while(ret == ERROR);

    return 0;
}
```

```c
char* CUI( ) {
    printf("Command 입력 : ");
    gets(CommandLine);
    return CommandLine;
}
```

Command Line

```c
int CheckSyntax(char* cml){          // 명령어 문법 검사
    char* cm[3];
    int syntax;
    cm[0] = strtok(cml, " ");        // 문자열을 세부분으로 나눔
    cm[1] = strtok(NULL, " ");
    cm[2] = strtok(NULL, " ");

    if( strcmp(cm[0], "exit") == 0) // exit 명령어 처리
        return SUCCESS;
    if(cm[1] != NULL){               // 입력 파일 존재
        strcpy(input, cm[1]);
        strtok(cm[1], ".");
        cm[1] = strtok(NULL, ".");
    }
    if(cm[2] != NULL){               // 출력 파일 존재
        strcpy(output, cm[2]);
        strtok(cm[2], ".");
        cm[2] = strtok(NULL, ".");
    }

    if(strcmp(cm[0], "create") == 0
        && cm[1]!=NULL && cm[2]!=NULL)  // create 명령어 처리
    {
        if( strcmp(cm[1], "c") == 0
            && strcmp(cm[2], "txt") == 0 )  // c, txt 형식 검사
        {
            CheckInput(input);
            syntax = SUCCESS;
        }
    }else{
        HelpCommand( );
        syntax = ERROR;
    }
    return syntax;
}
```

```c
void CheckInput(char* inputfile)    // 파일 읽기 검사
{
    int Fileopened;
    csource = fopen(inputfile,"r"); // 읽기용 파일 포인터
    if(csource != NULL){
        printf("C 파일[%s] 열기 성공₩n", inputfile);
        Fileopened = SUCCESS;
    } else {
        printf("C 파일[%s] 열기 실패₩n", inputfile);
        Fileopened = ERROR;
    }
    GenerationControl(Fileopened);  // 생성 제어
}
```

```c
void HelpCommand( ){                    // 명령어 도움말
    printf("₩n*******************[Command Error]*******************₩n");
    printf("CFG 생성 : create [Input 파일명] [Output 파일명] 형식₩n");
    printf("  [Input 파일명] : *.c 의 형태로 확장자를 입력해 주십시오₩n");
    printf("  [Out 파일명]   : *.txt 의 형태로 확장자를 입력해 주십시오₩n");
    printf("프로그램 종료 : exit₩n");
}
```

DFD Level 1

- Process Specification

Reference No.	0.1
Name	CUI
Input	
Output	Command Line
Process Description	Receive the command input from user

Reference No.	0.2
Name	Checking Syntax Process
Input	Command Line
Output	trigger
Process Description	Check your command received from user. If the command has errors, it triggers “Help Command”.

DFD Level 1

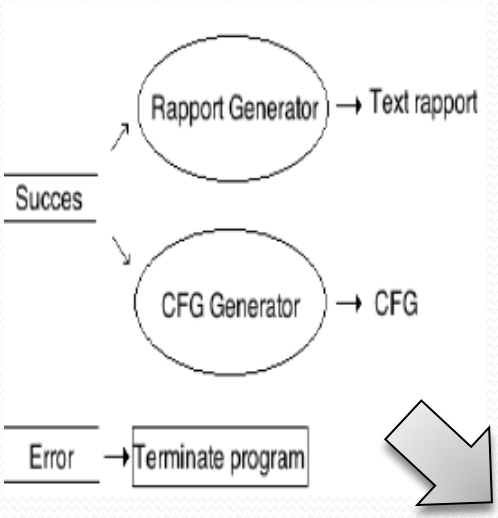
- Process Specification

Reference No.	0.3
Name	Checking Input Process
Input	Command Line, C Source
Output	File opened
Process Description	If the file input from Command Line read normally, save it " Success" but if it has error, save it "Error" in File opened

Reference No.	0.4
Name	Help Command
Input	Trigger
Output	
Process Description	Print out Help command.

Level 1

- CFG & Report Generation 1



Level 1

- CFG & Report Generation 1 Code

Generation Control

```
void GenerationControl(int Fopened)     // 생성 제어
{
    if( Fopened == SUCCESS  ){  // 파일 읽기 성공 시
        prepare();
        ExtractActualCFG();
        TerminateProgram();
    }
    else if( Fopened == ERROR ) // 파일 열기 실패 시
        TerminateProgram();
}
```

```
void prepare()              // CFG 생성 준비
{
    char line[256];
    FindFunctions();                // 함수 목록 찾기
    nodes--;
    fseek(csource, 0, SEEK_SET);    // 파일 포인트 초기화
    curLine = 1;
    while( fgets(line, 254, csource) != NULL ) // 한 줄 읽기
    {
        AnalysisLine(line);
        curLine++;
    }
    CFG_File_Write();
}
```

CFG Generator

```
void ExtractActualCFG()
{
    int i, main=-1;
    for(i=0 ; i< functions ; i++)
    {
        if( strcmp(FuncList[i].F_Name,"main") == 0 ){
            main = i*2;     // 메인 함수의 시작 노드 번호 찾기
            break;
        }
    }
    if(main!=-1){
        CheckPrint(main);   // main 함수으로 연결되는 노드를 체크
        ReportFileWrite();  // 레포트 출력
    }
}
```

Report Generator

```
void TerminateProgram()
{
    if(csource!=NULL)
        fclose(csource);
    printf("프로그램 종료₩n");
}
```

Terminate Program

DFD Level 1

- Process Specification

Reference No.	1.1
Name	Generation Control
Input	File opened
Output	trigger
Process Description	It triggers CFG Generator, Report Generator and also triggered Terminate Program if “File opened” is “Error” , mistake when creating CFG and after executing entire process normally.

Reference No.	1.2
Name	CFG Generator
Input	Trigger, C Source
Output	CFG File
Process Description	Create CFG In c source and save in txt File

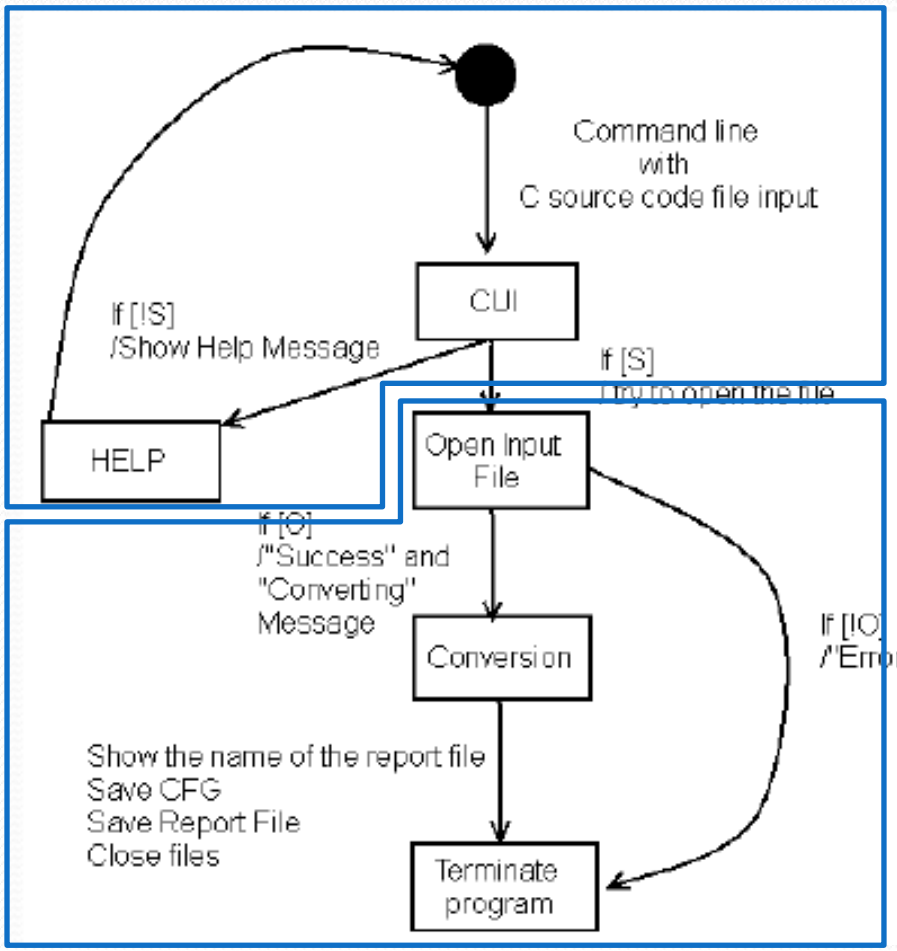
DFD Level 1

- Process Specification

Reference No.	1.3
Name	Report Generator
Input	trigger, C Source, Command Line
Output	Report File
Process Description	Save report from txt File created from CFG information According to flow in txt File

Reference No.	1.4
Name	Terminate Program
Input	trigger
Output	
Process Description	Terminate the program

State Transition Diagram



Checking Syntax Process

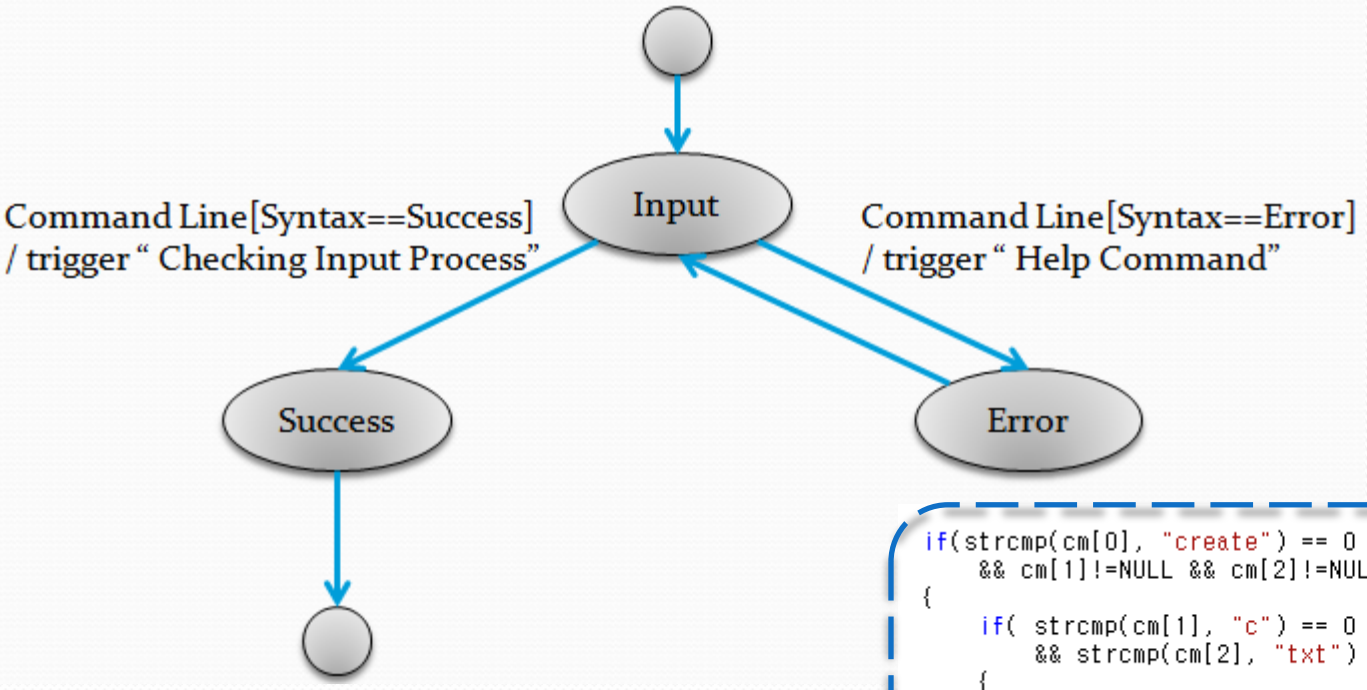
Generation Control

S : correct Syntax

O : success to open the file

State Transition Diagram

- Checking Syntax Process

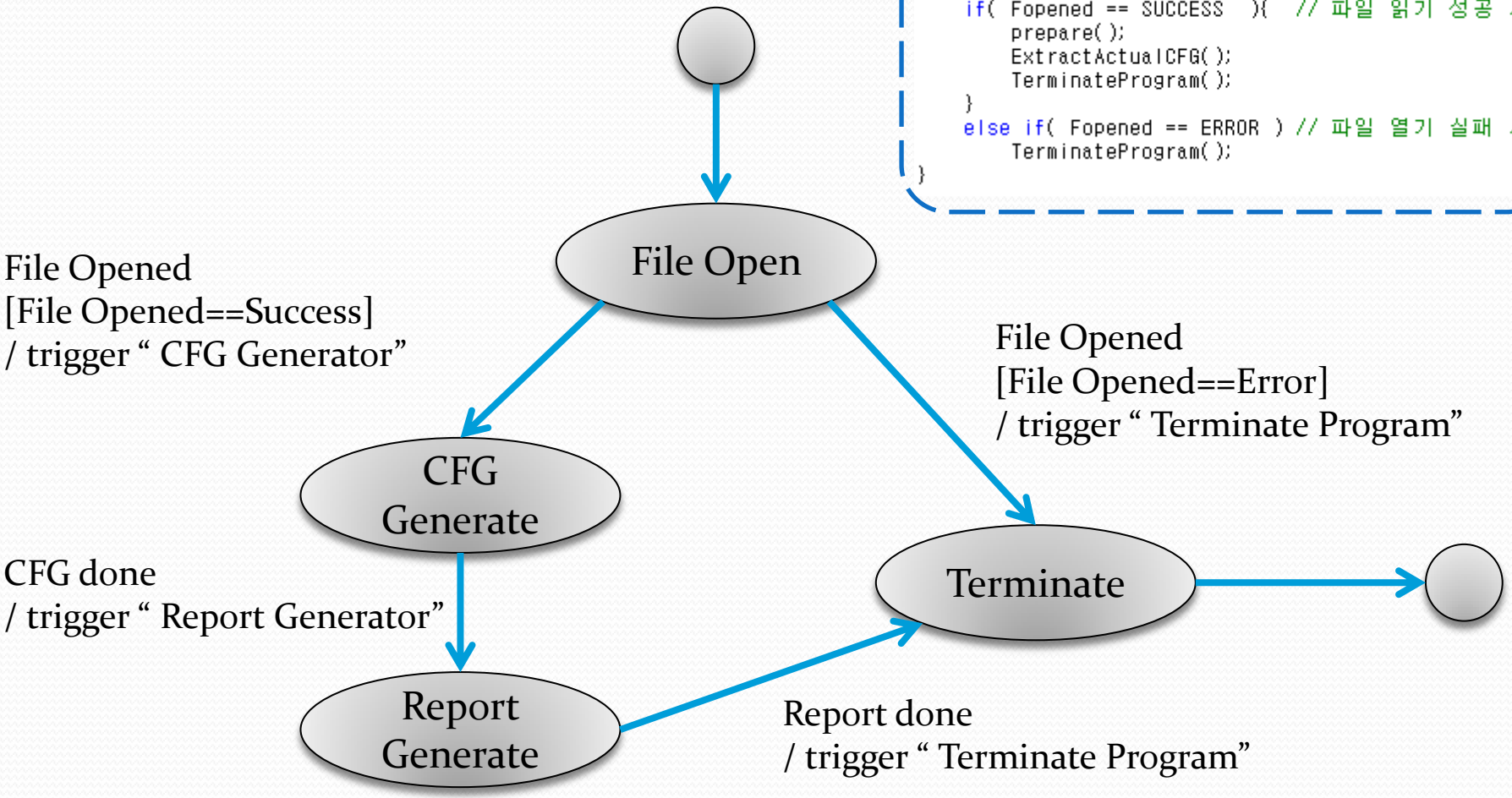



```
if(strcmp(cm[0], "create") == 0
    && cm[1]!=NULL && cm[2]!=NULL)  // create 명령어 처리
{
    if( strcmp(cm[1], "c") == 0
        && strcmp(cm[2], "txt") == 0 )  // c, txt 형식 검사
    {
        CheckInput(input);
        syntax = SUCCESS;
    }
}else{
    HelpCommand();
    syntax = ERROR;
}
return syntax;
```

State Transition Diagram

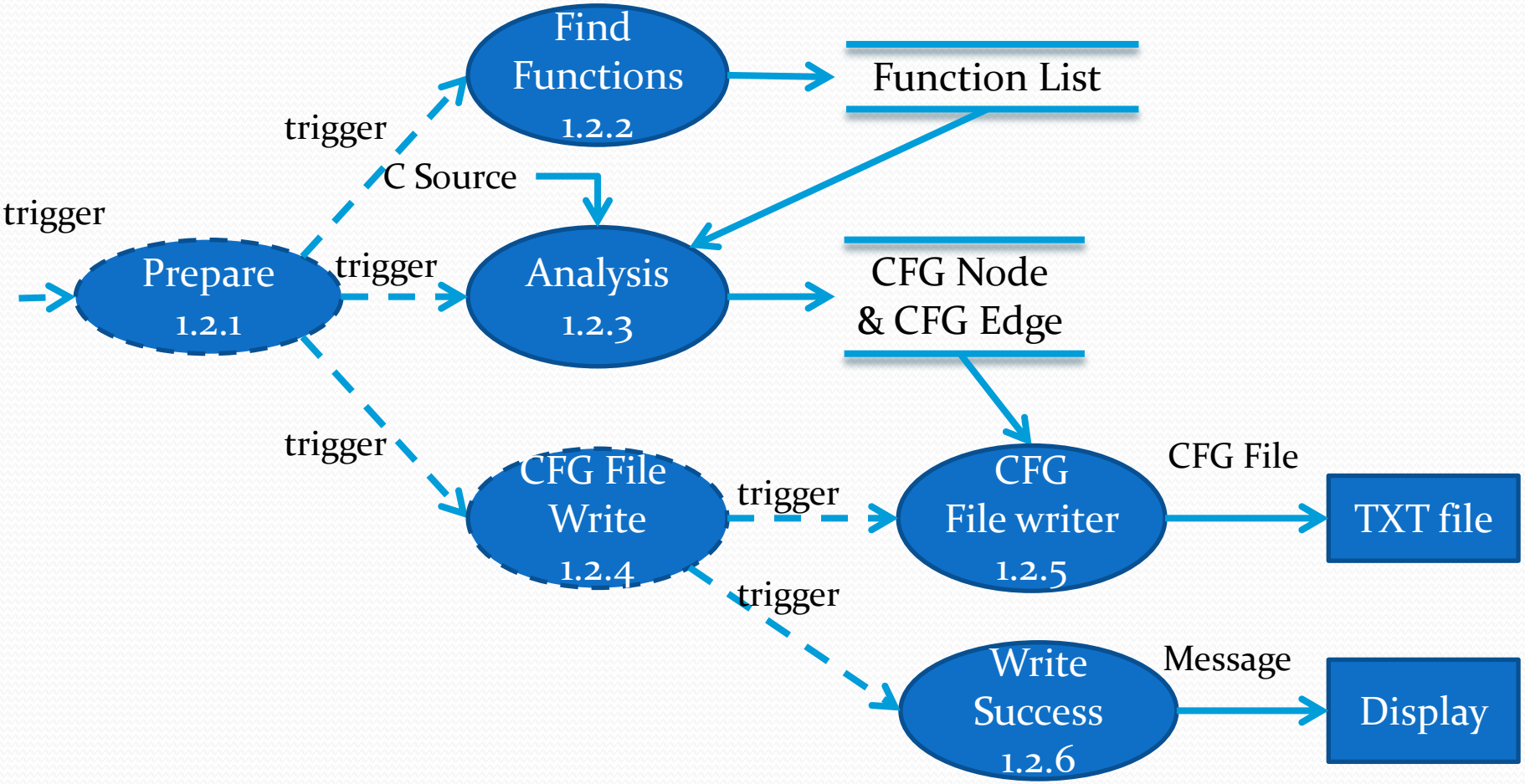
- Generation Control

```
void GenerationControl(int Fopened)     // 생성 제어
{
    if( Fopened == SUCCESS  ){  // 파일 읽기 성공 시
        prepare();
        ExtractActualCFG();
        TerminateProgram();
    }
    else if( Fopened == ERROR ) // 파일 열기 실패 시
        TerminateProgram();
}
```



Level 2

- CFG Generator 1.2



Level 2

- CFG Generator 1.2

prepare

```
void prepare()              // CFG 생성 준비
{
    char line[256];
    FindFunctions();                // 함수 목
    nodes--;
    fseek(csource, 0, SEEK_SET);    // 파일 포
    curLine = 1;
    while( fgets(line, 254, csource) != NULL )
    {
        AnalysisLine(line);
        curLine++;
    }
    CFG_File_Write();
}
```

```
void FindFunctions()
{
    char line[255];
    char* result;
    int i, found = FALSE;

    if(csource == NULL)         // 파일 포인터 검사
        return;

    while( fgets(line, 254, csource) != NULL) { // 한줄 읽기
        result = getFunctionName(line);         // 함수 이름 얻기
        if(result!=NULL) {  // 함수 선언인 경우 - 원형, 선언 모두 감지
            if( strstr(result, " ") != NULL )   // 함수 이름에 공백 존재: 원형이 아님
                continue;
            for( i = 0 ; i < functions ; i++ ) {    // 이미 등록된지 검사
                if( strcmp(result, FuncList[i].F_Name) == 0 ) {
                    found = TRUE;
                    break;
                }
            }

            if(found == FALSE){ // 이미 등록되지 않은 경우 등록
                strcpy(FuncList[functions].F_Name, result);
                strcpy(FuncList[functions].retType, strtok(line," "));
                FuncList[functions].Node_Num = nodes++;
                FuncList[functions].Node_End = nodes++;
                functions++;
            }else
                found = FALSE;
        }
    }
```

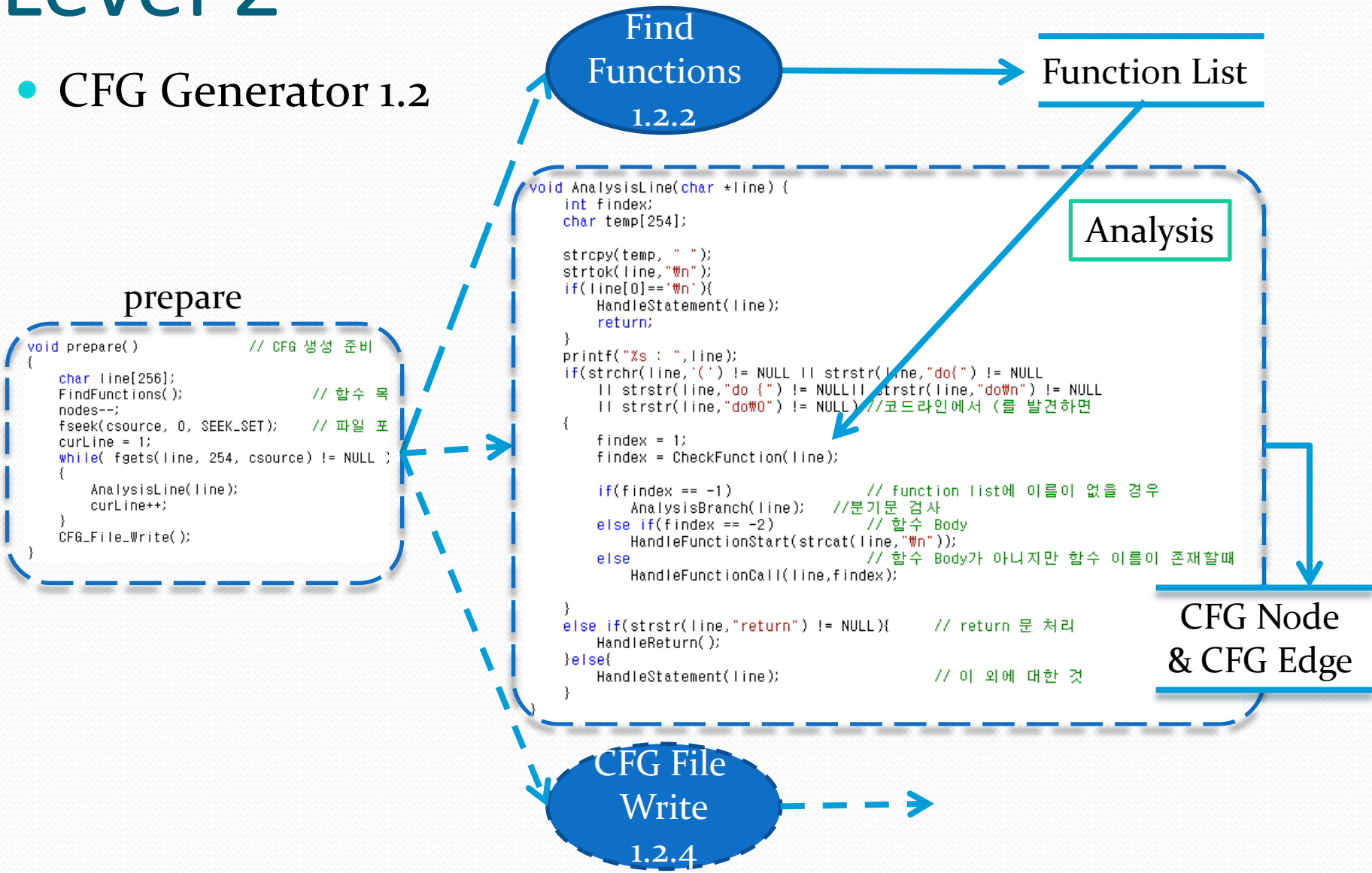
Analysis 1.2.3

Function List

CFG File Write 1.2.4

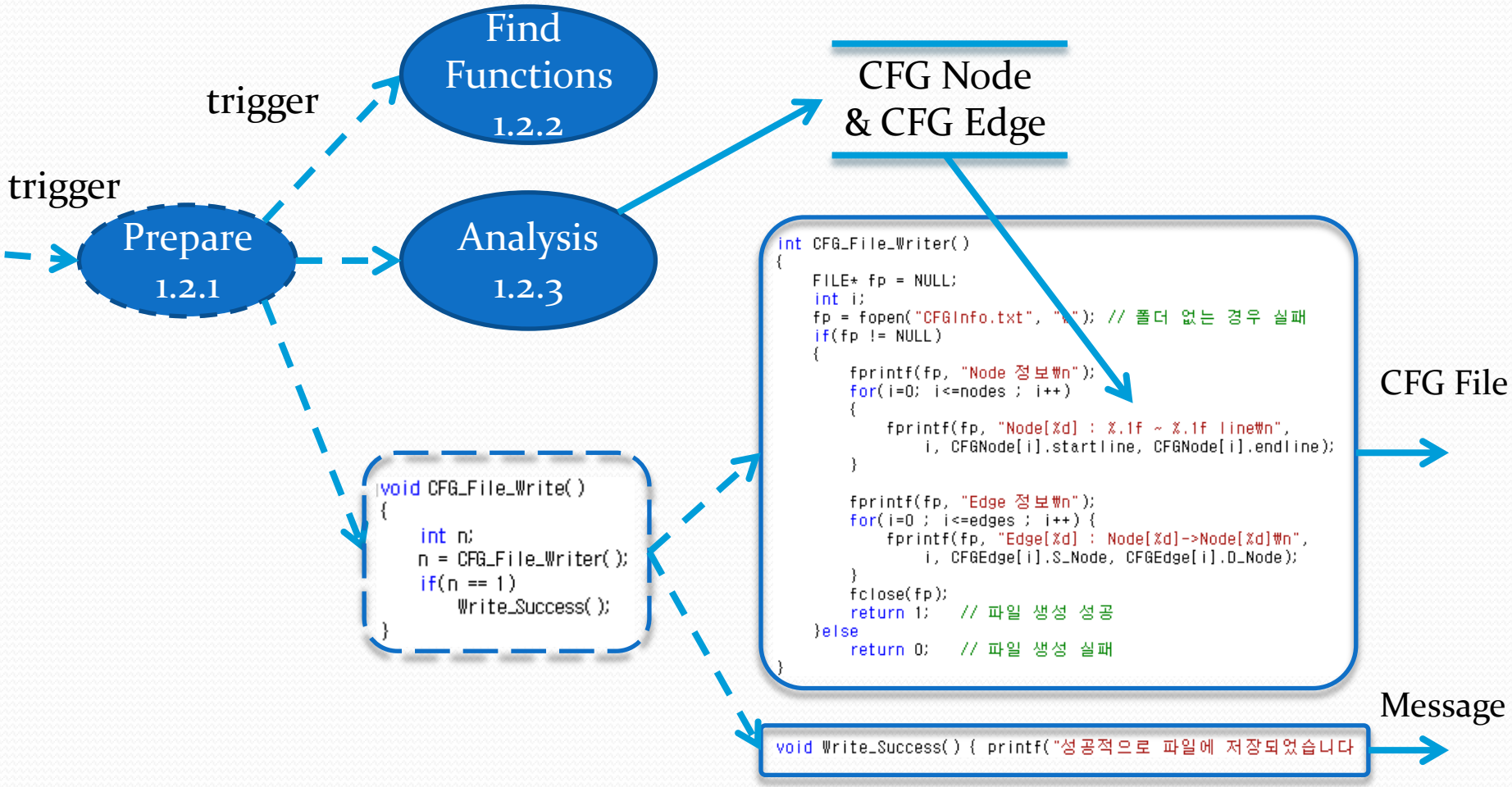
Level 2

- CFG Generator 1.2



Level 2

- CFG Generator 1.2



DFD Level 2

- Process Specification

Reference No.	1.2.1
Name	prepare
Input	trigger
Output	trigger
Process Description	Trigger Find Functions, Analysis, CFG File Write.

Reference No.	1.2.2
Name	Find Functions
Input	trigger
Output	Function List
Process Description	Find function from C source and Save it in Function List.

DFD Level 2

- Process Specification

Reference No.	1.2.3
Name	Analysis
Input	trigger, Function List, C Source
Output	CFG Node & CFG Edge
Process Description	Read C source every other line and create CFG Node, CFG Edge.

Reference No.	1.2.4
Name	CFG File Write
Input	trigger, CFG Node & CFG Edge
Output	trigger
Process Description	Execute TXT File writer and Write Success outputting CFG regarding Node's connection from CFG Node and CFG Edge.

DFD Level 2

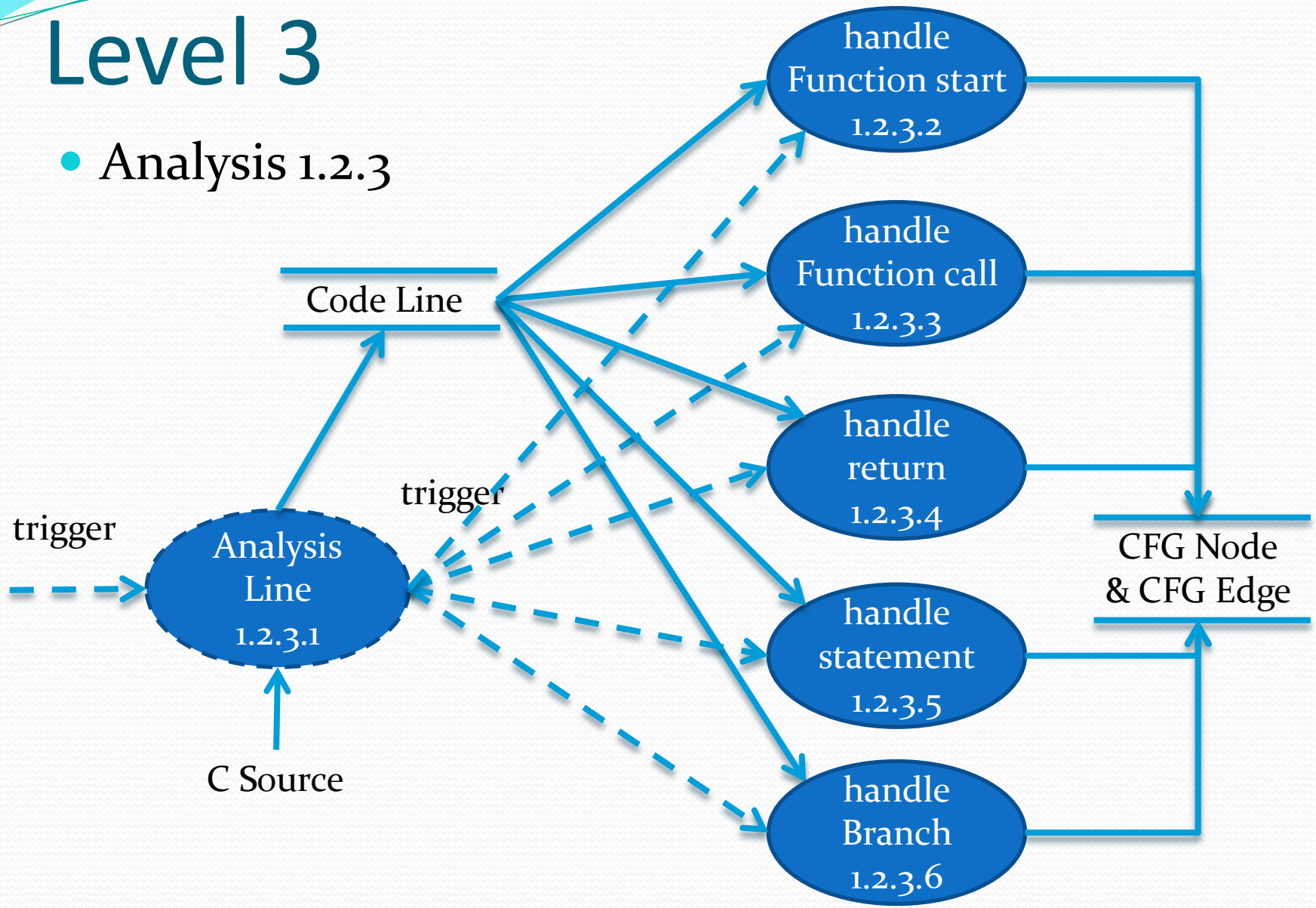
- Process Specification

Reference No.	1.2.5
Name	CFG File writer
Input	trigger
Output	CFG File
Process Description	If CFG information is created normally, CFG information print txt file.

Reference No.	1.2.6
Name	Write Success
Input	trigger
Output	Message
Process Description	If file creation is success, printf message

Level 3

- Analysis 1.2.3



Level 3

- Analysis 1.2.3 Code

Handle Function Start 1.2.3.2

```
void HandleFunctionStart(char* line)
{
    int i;
    char func[254];
    char* fname;

    strcpy(func, line);
    fname = getFunctionName(func);

    inFunction = TRUE;                          // 함수 내부 진입
    CFGNode[++nodes].startline = (float)curLine;    // 노드 추가 : 함수 시작
    curNode= nodes;

    for(i=0 ; i<functions ; i++)            //function list에서 같은 함수명이 있을 때
    {
        if( strcmp( fname, FuncList[i].F_Name) == 0 )
        {
            curFunc = i;
            CFGEdge[++edges].S_Node = i*2;  // 엣지 생성 : Function Node -> 현재 노드
            CFGEdge[edges].D_Node = curNode;
            break;
        }
    }
    printf("HandleFucntionStart Node[%d->%d]₩n",curFunc*2, curNode);
}
```

Handle Function Call 1.2.3.3

```
void HandleFunctionCall(char *line, int index){
    int end;

    CFGNode[curNode].endline = (float)curLine;  // 현재 노드의 마지막 : 현재 라인
    CFGNode[++nodes].startline = (float)curLine+1;// 노드 생성 : 다음 줄부터 시작
    end = nodes;

    CFGEdge[++edges].S_Node = curNode;      // 엣지 생성 : 현재 노드 -> 함수 시작
    CFGEdge[edges].D_Node = index*2;
    CFGEdge[++edges].S_Node = curNode;      // 엣지 생성 : 현재 노드 -> 다음 라인
    CFGEdge[edges].D_Node = end;
    curNode = end;
    printf("HandleFucntionCall Node[%d->%d,%d]₩n",curNode, index*2,curNode+1);
}
```

Handle Return 1.2.3.4

```
void HandleReturn( )
{
    CFGNode[curNode].endline = (float)curLine;
    printf("HandleReturn Node[%d->%d] %.f~%.f₩n",
        curNode,(curFunc*2)+1, CFGNode[curNode].startline, CFGNode[curNode].endli
    CFGEdge[++edges].S_Node = curNode;              // 엣지 생성 : 현재 노드 -> 함
    CFGEdge[edges].D_Node = FuncList[curFunc].Node_End;
    CFGNode[++nodes].startline = (float)curLine+1;  // 노드 생성 : 다음 라인
    curNode++;
}
```

Handle Statement 1.2.3.5

```
void HandleStatement(char *line)        // 일반 statement 처리
{
    if(inFunction == TRUE){             // 함수 내부인 경우
        if( strchr(line, '{') !=NULL)
            brackets++;
        if( strchr(line, '}') !=NULL)
            brackets--;
        CFGNode[curNode].endline = (float)curLine;
        if( brackets != 0 )     // 함수 내부 Statement
            printf("HandleStatement Node[%d] %.f~%.f₩n", curNode,
                CFGNode[curNode].startline, CFGNode[curNode].endline);
        else{                   // 함수 끝 Statement
            printf("HandleStatement(funcend) Node[%d->%d] %.f~%.f₩n", curNode,
                (curFunc*2)+1,CFGNode[curNode].startline, CFGNode[curNode].endline);
            CFGEdge[++edges].S_Node = curNode;      // 엣지 생성 : 현재 노드 -> 함수 끝
            CFGEdge[edges].D_Node = (curFunc*2)+1;
            inFunction = FALSE;
        }
    }else{                  // 함수 외부인 경우 : 전처리문, 전역변수, 함수 원형 등등
        printf("₩n");
    }
}
```

DFD Level 3

- Process Specification

Reference No.	1.2.3.1
Name	Analysis line
Input	Trigger, C Source
Output	Code Line , trigger
Process Description	Trigger CFG creation process pursuant to type of code that analyzed each line from C source

Reference No.	1.2.3.2
Name	Handle Function start
Input	Code Line , trigger
Output	CFG Node & CFG Edge
Process Description	Create CFG information of Code Line starting function.

DFD Level 3

- Process Specification

Reference No.	1.2.3.3
Name	Handle Function call
Input	Code Line, trigger
Output	CFG Node & CFG Edge
Process Description	Create CFG information of Code Line calling function.

Reference No.	1.2.3.4
Name	Handle return
Input	Code Line , trigger
Output	CFG Node & CFG Edge
Process Description	Create CFG information of Code Line that is return statement

DFD Level 3

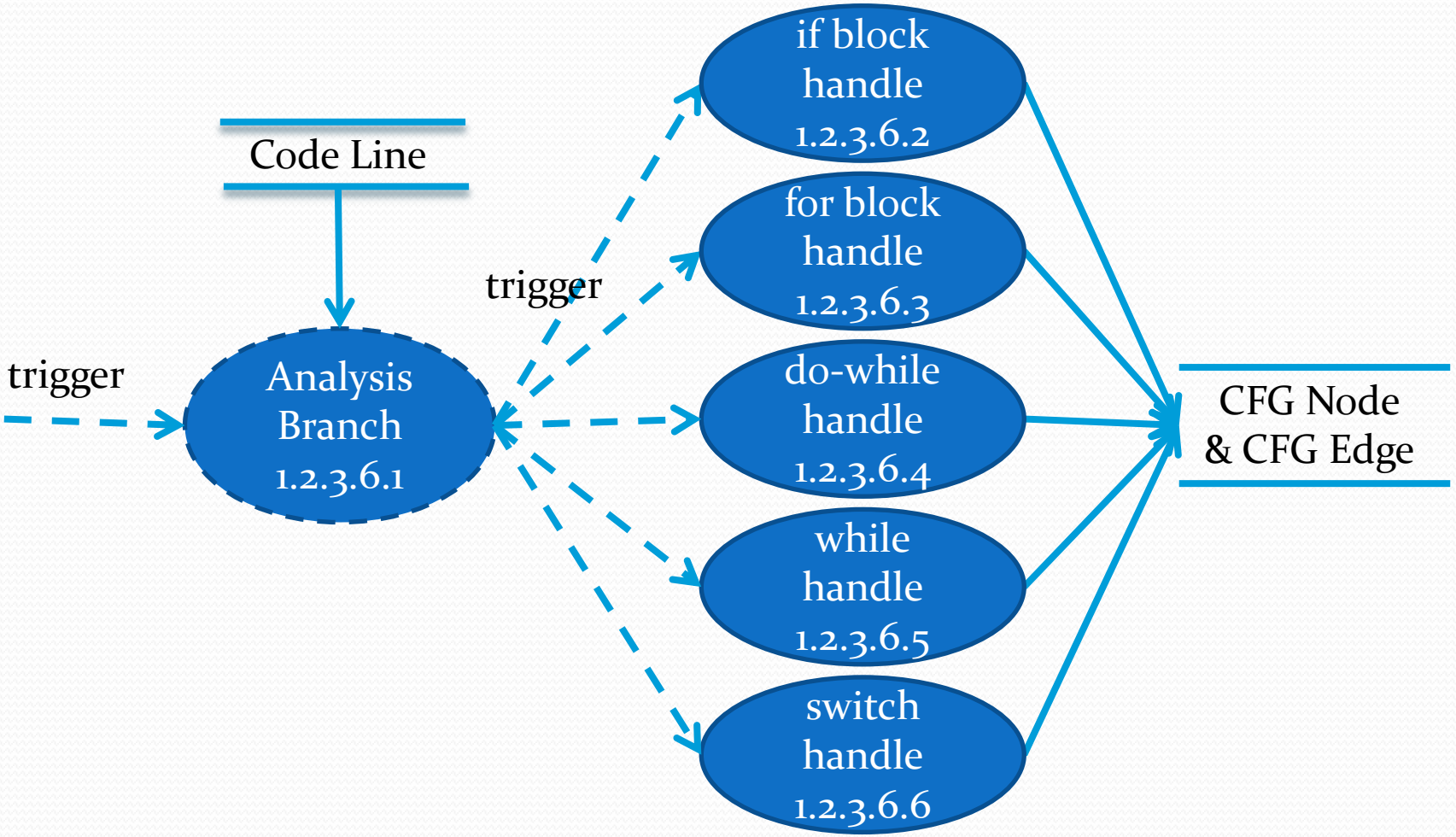
- Process Specification

Reference No.	1.2.3.5
Name	Handle statement
Input	Code Line, trigger
Output	CFG Node & CFG Edge
Process Description	Create CFG information of general Code

Reference No.	1.2.3.6
Name	Handle Branch
Input	Code Line, trigger
Output	CFG Node & CFG Edge
Process Description	Create CFG information of Branch Code

DFD Level 4

- Handle Branch 1.2.3.6



DFD Level 4

- Handle Branch 1.2.3.6

Code Line

trigger

```
void AnalysisBranch(char *line)
{
    if(strstr(line,"if(") != NULL)
        IfBlockHandle(line);     //if 문
    else if(strstr(line,"for(") != NULL)
        ForBlockHandle(line);    //for문
    else if(strstr(line,"while(") != NULL)
        WhileHandle(line);       //while 문
    else if(strstr(line,"switch(") != NULL)
        SwitchHandle();          //switch 문
    else if(strstr(line,"do{") != NULL
        || strstr(line,"do₩n") != NULL
        || strstr(line,"do₩0") != NULL)
        Do_whileBlockHandle();   //do-while문
    else            //분기문 아니라면 일반문
        HandleStatement(line);
}
```

```
void IfBlockHandle()

    while(fgets(if_line,254,csource) != NULL)
    {
        if(strstr(if_line,"else if") != NULL)       // else if 발견
            while(fgets(if_line,254,csource) != NULL)       // else if state 처리

        if(strstr(if_line,"else if") == NULL && strstr(if_line,"else") != NULL) // else문 처리

        if(con == 2){                           // if state 노드 안
            CFGNode[ifState].endline = curLine;         // If state 노드 증가
```

```
void ForBlockHandle(){
    while(fgets(for_line, 254, csource) != NULL)    // 반복 state 처리
    {   curLine++;
        CFGNode[state].endline = curLine;
        strtok(for_line,"₩n");
        printf("%s for state Node[%d] ₩n", for_line, state);
        if(strchr(for_line,'}') != NULL)            // 반복 state 종료 발견
        break;
    }
    CFGNode[end].startline = curLine+1;             // 노드 생성 : for문 end
```

```
void WhileHandle(){
        curLine++;
        CFGNode[state].endline = curLine;
        strtok(while_line,"₩n");
        printf("%s Node[%d]₩n",while_line,state);
        if(strchr(while_line,'}') != NULL)
            break;
    CFGNode[end].startline = curLine+1;         // 노드 생성 : end
```

```
void SwitchHandle(){
    while(fgets(switch_line,254,csource) != NULL)
    {   if(strstr(switch_line,"default") != NULL)       // default문 처리
        else if(strstr(switch_line,"case") != NULL) // case문 처리
        if( (strchr(switch_line,'{') != NULL) && (strstr(switch_line,"case") == NULL)
        if(strchr(switch_line,'}') != NULL){ // } 처리
        if(back==TRUE){                     // 한 줄 전 이동
    if(defaultState == -1)                  // default가 없으면 switch -> end
        CFGEdge[tempEdge[0]].D_Node = end;
    for(i=1;i<con;i++)                      // 기존 엣지 : case -> end
```

```
void Do_whileBlockHandle()
    while(fgets(doWhile_line, 254, csource) != NULL)    // state 처리
    {
        curLine++;
        strtok(doWhile_line,"₩n");
        if(strstr(doWhile_line,"while(") != NULL)       // 조건문 처리
        {
            CFGNode[state].endline = curLine - 1;
            printf("%s do-while Node[%d]₩n",doWhile_line,con);
            break;
        }
        printf("%s In do-while Node[%d]₩n",doWhile_line,state);
```

DFD Level 4

- Process Specification

Reference No.	1.2.3.6.1
Name	Analysis Branch
Input	trigger, Code Line
Output	trigger
Process Description	Extract branch block and Trigger suitable process

Reference No.	1.2.3.6.2
Name	if block handle
Input	trigger
Output	CFG Node & CFG Edge
Process Description	Create CFG information of Code Line that is return statement

DFD Level 4

- Process Specification

Reference No.	1.2.3.6.3
Name	for block handle
Input	trigger
Output	CFG Node & CFG Edge
Process Description	Create CFG Information in for block

Reference No.	1.2.3.6.4
Name	Do-while handle
Input	trigger
Output	CFG Node & CFG Edge
Process Description	Create CFG Information in do-while block

DFD Level 4

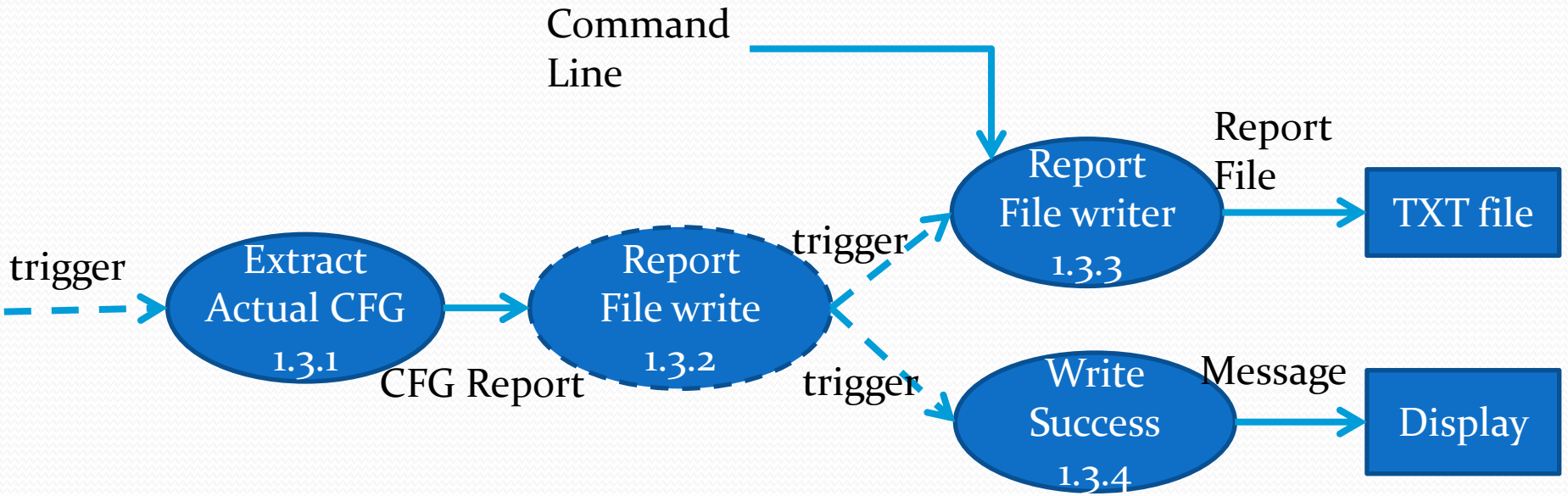
- Process Specification

Reference No.	1.2.3.6.5
Name	while handle
Input	trigger
Output	CFG Node & CFG Edge
Process Description	Create CFG Information in for block

Reference No.	1.2.3.6.6
Name	switch handle
Input	trigger
Output	CFG Node & CFG Edge
Process Description	Create CFG Information in do-while block

Level 2

- Report Generator 1.3



Level 2

- Report Generator 1.3 Code

Extract Actual CFG 1.3.1

```
void ExtractActualCFG( )
{
    int i, main=-1;
    for(i=0 ; i< functions ; i++)
    {
        if( strcmp(FuncList[i].F_Name,"main") == 0 ){
            main = i+2;      // 메인 함수의 시작 노드 번호 찾기
            break;
        }
    }
    if(main!=-1){
        CheckPrint(main);    // main 함수으로 연결되는 노드를
        ReportFileWrite( );  // 레포트 출력
    }
}

void CheckPrint(int nodenum)
{
    int i;
    if(pNode[nodenum] != -1) {  // 이미 체크한 것이 아니라면
        pNode[nodenum] = -1;            // 노드 체크
        for( i =0; i<= edges ; i++) {
            if(CFGEdge[i].S_Node == nodenum) {
                if(pEdge[i] != -1) {     // 체크된 것이 아니라면
                    pEdge[i] = -1;       // 엣지 체크
                    CheckPrint(CFGEdge[i].D_Node);  // 연결된 노드 체크
                }
            }
        }
    }
}
```

Report File Write 1.3.2

```
void ReportFileWrite( )
{
    int ret;
    ret = ReportFileWriter( );
    if(ret == SUCCESS)
        Write_Success( );
}
```

Report File Writer 1.3.3

```
int ReportFileWriter( )
{
    FILE* fp = NULL;
    int i,j, po2;
    char line[254];
    char* forline;
    float po;
    fseek(csource, 0, SEEK_SET);     // 파일 포인트 초기화
    curLine = 1;
    fp = fopen(output, "w");         // 폴더 없는 경우 실패
    if(fp==NULL)
        return ERROR;

    for(i=0 ; i<=nodes ; i++)   {
        if(pNode[i] == -1)  {
            if( CFGNode[i].startline != 0 ){
                fprintf(fp,"Node [%d]₩n",i);
                while(1)    {
                    po = (CFGNode[i].startline*10);
                    po2 = ((int)po)%10;
                    if( curLine < CFGNode[i].startline )    {
                        if(po2 == 0){
                            fgets(line,254,csource);
                            curLine++;
                        }else {
                            if (po2 == 1){
                                fgets(line,254,csource);
                                forline = strtok(line, ";");
                                fprintf(fp,forline);
                                fprintf(fp,";₩n");
                            }else if(po2 == 2){
                                forline = strtok(NULL, ";");
                                fprintf(fp,forline);
                            }else if(po2 == 3){
                                forline = strtok(NULL, ";");
                                fprintf(fp,forline);
                                curLine++;
                            }
                            break;
                        }
                    }else if( ( curLine >= CFGNode[i].startline )
                        && ( curLine <= CFGNode[i].endline ) ) {
                        fgets(line,254,csource);
                        fprintf(fp, "%s", line);
                        curLine++;
                    }else if( curLine > CFGNode[i].endline )
                        break;
                }
                for(j=0; j<=edges ; j++) {
                    if(pEdge[j] == -1){
                        if(CFGEdge[j].S_Node== i ) {
                            fprintf(fp, "Edge[%d] : Node[%d] -> Node[%d]₩n",
                                j, CFGEdge[j].S_Node,CFGEdge[j].D_Node);
                            pEdge[j]=1;
                        }
                    }
                }
            }
        }
    }
    fclose(fp);
    return SUCCESS;
}
```

Report File

Write Success 1.3.4

```
void Write_Success( ) { printf("성공적으로 파일에 저장되었습니다
```

Message

DFD Level 2

- Process Specification

Reference No.	1.3.1
Name	Extract Actual CFG
Input	CFG File
Output	CFG Report
Process Description	Write Report according to flow from CFG information gotten from CFG file.

Reference No.	1.3.2
Name	Report File write
Input	trigger
Output	trigger
Process Description	Execute Write Success printing success or failure and Report File Writer producing information file regarding individual Nodes.

DFD Level 2

- Process Specification

Reference No.	1.3.3
Name	Report File writer
Input	trigger, Command Line, CFG Report
Output	Report File
Process Description	Print out txt file to CFG Report information.

Reference No.	1.3.4
Name	Write Success
Input	trigger
Output	Message
Process Description	If file creation is success, print out message

Data Dictionary

Type	Description	Struct

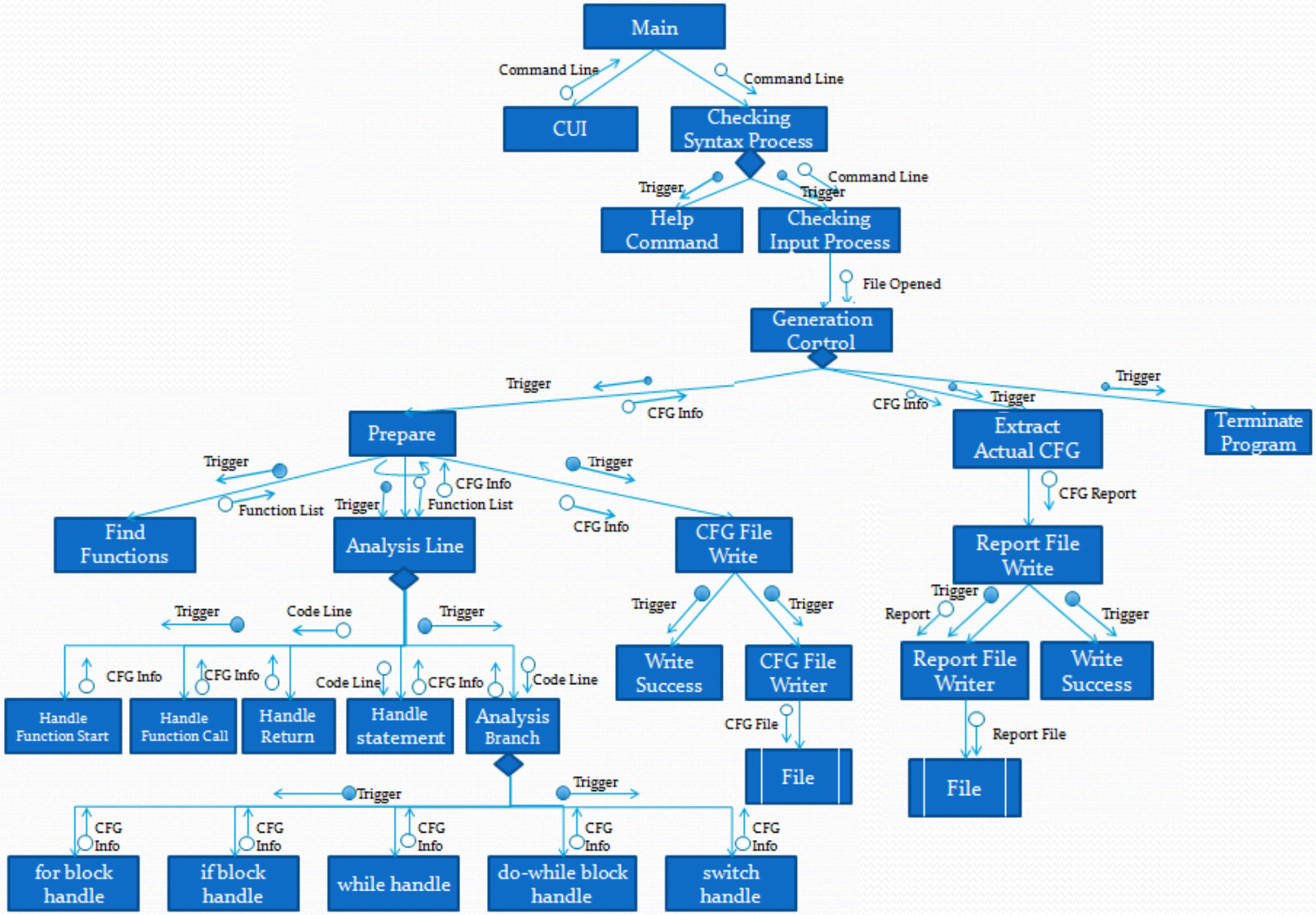
Node	Save Basic Block Information of CFG
	Node number , Code String
		Struct {
		int Node_Num ;
		float startline ;
		float endline ;
		}

Edge	Save Edge information of CFG
	Start / Destination Node number
		Struct {
		int S_Node ;
		int D_Node ;
		}

Function List	Save Declared Function Name and Return Type, Start and End Node number	Struct {
		String F_Name ;
		String retType ;
		int Node_Num ;
		int Node_End ;
		}

CFG Node & CFG Edge	Save Node and Edge information of created CFG	Node, Edge

Structured Charts

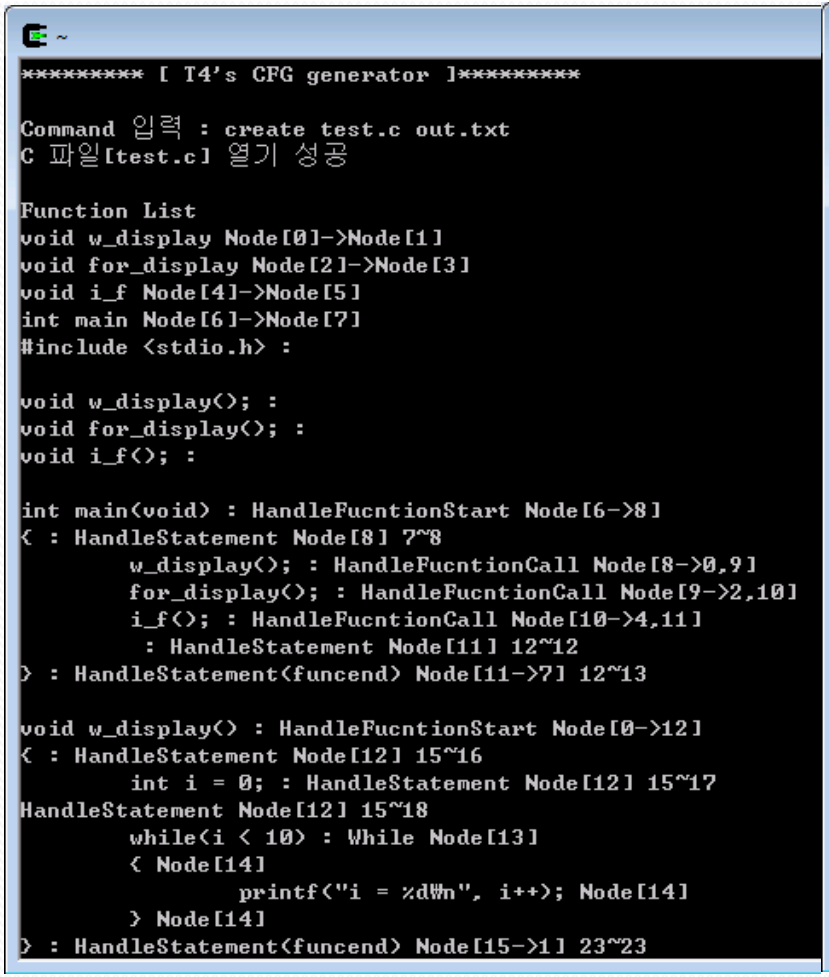


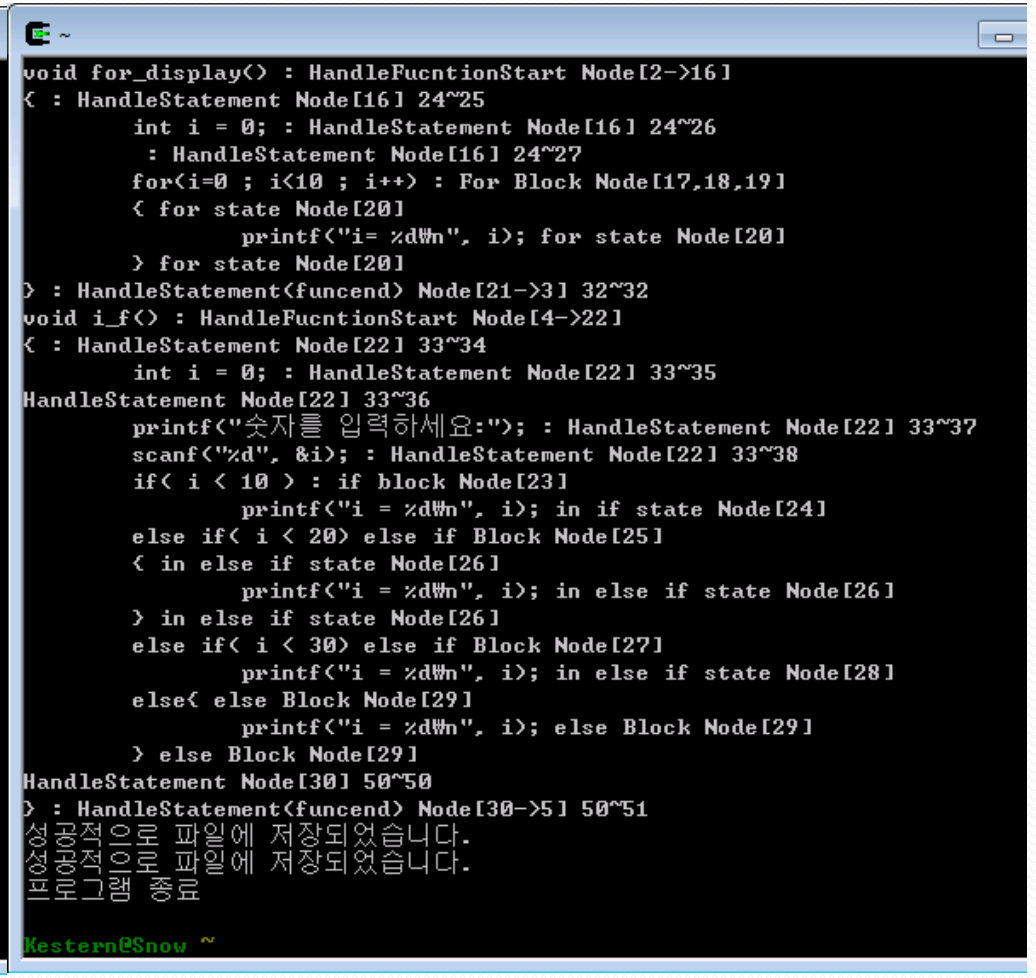
Example source

```c
#include <stdio.h>

void w_display();
void for_display();
void i_f();

int main(void)
{
    w_display();
    for_display();
    i_f();

}

void w_display()
{
    int i = 0;

    while(i < 10)
    {
        printf("i = %d\n", i++);
    }
}
void for_display()
{
    int i = 0;

    for(i=0 ; i<10 ; i++)
    {
        printf("i= %d\n", i);
    }
}
```

```c
void i_f()
{
    int i = 0;

    printf("숫자를 입력하세요:");
    scanf("%d", &i);
    if( i < 10 )
        printf("i = %d\n", i);
    else if( i < 20)
    {
        printf("i = %d\n", i);
    }
    else if( i < 30)
        printf("i = %d\n", i);
    else{
        printf("i = %d\n", i);
    }

}
```

Execution Screen





CFG Information

```
Node 정보
Node[0] : 0.0 ~ 0.0 line
Node[1] : 0.0 ~ 0.0 line
Node[2] : 0.0 ~ 0.0 line
Node[3] : 0.0 ~ 0.0 line
Node[4] : 0.0 ~ 0.0 line
Node[5] : 0.0 ~ 0.0 line
Node[6] : 0.0 ~ 0.0 line
Node[7] : 0.0 ~ 0.0 line
Node[8] : 7.0 ~ 9.0 line
Node[9] : 10.0 ~ 10.0 line
Node[10] : 11.0 ~ 11.0 line
Node[11] : 12.0 ~ 13.0 line
Node[12] : 15.0 ~ 18.0 line
Node[13] : 19.0 ~ 19.0 line
Node[14] : 20.0 ~ 22.0 line
Node[15] : 23.0 ~ 23.0 line
Node[16] : 24.0 ~ 27.0 line
Node[17] : 28.1 ~ 28.1 line
Node[18] : 28.2 ~ 28.2 line
Node[19] : 28.3 ~ 28.3 line
Node[20] : 29.0 ~ 31.0 line
Node[21] : 32.0 ~ 32.0 line
Node[22] : 33.0 ~ 38.0 line
Node[23] : 39.0 ~ 39.0 line
Node[24] : 40.0 ~ 40.0 line
Node[25] : 41.0 ~ 41.0 line
Node[26] : 42.0 ~ 44.0 line
Node[27] : 45.0 ~ 45.0 line
Node[28] : 46.0 ~ 46.0 line
Node[29] : 47.0 ~ 49.0 line
Node[30] : 50.0 ~ 51.0 line
```

```
Edge 정보
Edge[0] : Node[0]->Node[0]
Edge[1] : Node[6]->Node[8]
Edge[2] : Node[8]->Node[0]
Edge[3] : Node[8]->Node[9]
Edge[4] : Node[9]->Node[2]
Edge[5] : Node[9]->Node[10]
Edge[6] : Node[10]->Node[4]
Edge[7] : Node[10]->Node[11]
Edge[8] : Node[11]->Node[7]
Edge[9] : Node[0]->Node[12]
Edge[10] : Node[12]->Node[13]
Edge[11] : Node[13]->Node[14]
Edge[12] : Node[14]->Node[13]
Edge[13] : Node[13]->Node[15]
Edge[14] : Node[15]->Node[1]
Edge[15] : Node[2]->Node[16]
Edge[16] : Node[16]->Node[17]
Edge[17] : Node[17]->Node[18]
Edge[18] : Node[18]->Node[20]
Edge[19] : Node[20]->Node[19]
Edge[20] : Node[19]->Node[18]
Edge[21] : Node[18]->Node[21]
Edge[22] : Node[21]->Node[3]
Edge[23] : Node[4]->Node[22]
Edge[24] : Node[22]->Node[23]
Edge[25] : Node[23]->Node[24]
Edge[26] : Node[24]->Node[30]
Edge[27] : Node[23]->Node[25]
Edge[28] : Node[25]->Node[26]
Edge[29] : Node[26]->Node[30]
Edge[30] : Node[23]->Node[27]
Edge[31] : Node[27]->Node[28]
Edge[32] : Node[28]->Node[30]
Edge[33] : Node[23]->Node[29]
Edge[34] : Node[29]->Node[30]
Edge[35] : Node[30]->Node[5]
```

Report File

```
Node [8]
int main(void)
{
    w_display();
Edge[2] : Node[8] -> Node[0]
Edge[3] : Node[8] -> Node[9]
Node [9]
    for_display();
Edge[4] : Node[9] -> Node[2]
Edge[5] : Node[9] -> Node[10]
Node [10]
    i_f();
Edge[6] : Node[10] -> Node[4]
Edge[7] : Node[10] -> Node[11]
Node [11]

}
Edge[8] : Node[11] -> Node[7]
Node [12]
void w_display()
{
    int i = 0;

Edge[10] : Node[12] -> Node[13]
Node [13]
    while(i < 10)
Edge[11] : Node[13] -> Node[14]
Edge[13] : Node[13] -> Node[15]
Node [14]
    {
        printf("i = %d\n", i++);
    }
Edge[12] : Node[14] -> Node[13]
Node [15]
}
Edge[14] : Node[15] -> Node[1]
Node [16]
```

```
void for_display()
{
    int i = 0;

Edge[16] : Node[16] -> Node[17]
Node [17]
    for(i=0 ;
Edge[17] : Node[17] -> Node[18]
Node [18]
 i<10 ;
Edge[18] : Node[18] -> Node[20]
Edge[21] : Node[18] -> Node[21]
Node [19]
 i++)
Edge[20] : Node[19] -> Node[18]
Node [20]
    {
        printf("i= %d\n", i);
    }
Edge[19] : Node[20] -> Node[19]
Node [21]
}
Edge[22] : Node[21] -> Node[3]
Node [22]
```

```
void i_f()
{
    int i = 0;

    printf("숫자를 입력하세요:");
    scanf("%d", &i);
Edge[24] : Node[22] -> Node[23]
Node [23]
    if( i < 10 )
Edge[25] : Node[23] -> Node[24]
Edge[27] : Node[23] -> Node[25]
Edge[30] : Node[23] -> Node[27]
Edge[33] : Node[23] -> Node[29]
Node [24]
        printf("i = %d\n", i);
Edge[26] : Node[24] -> Node[30]
Node [25]
    else if( i < 20)
Edge[28] : Node[25] -> Node[26]
Node [26]
    {
        printf("i = %d\n", i);
    }
Edge[29] : Node[26] -> Node[30]
Node [27]
    else if( i < 30)
Edge[31] : Node[27] -> Node[28]
Node [28]
        printf("i = %d\n", i);
Edge[32] : Node[28] -> Node[30]
Node [29]
    else{
        printf("i = %d\n", i);
    }
Edge[34] : Node[29] -> Node[30]
Node [30]

}
Edge[35] : Node[30] -> Node[5]
Edge[1] : Node[6] -> Node[8]
Edge[9] : Node[0] -> Node[12]
Edge[15] : Node[2] -> Node[16]
Edge[23] : Node[4] -> Node[22]
```