Final presentation of CFG Generator

Team 5

200711453 류진렬

200711454 윤병현

200711459 이남섭

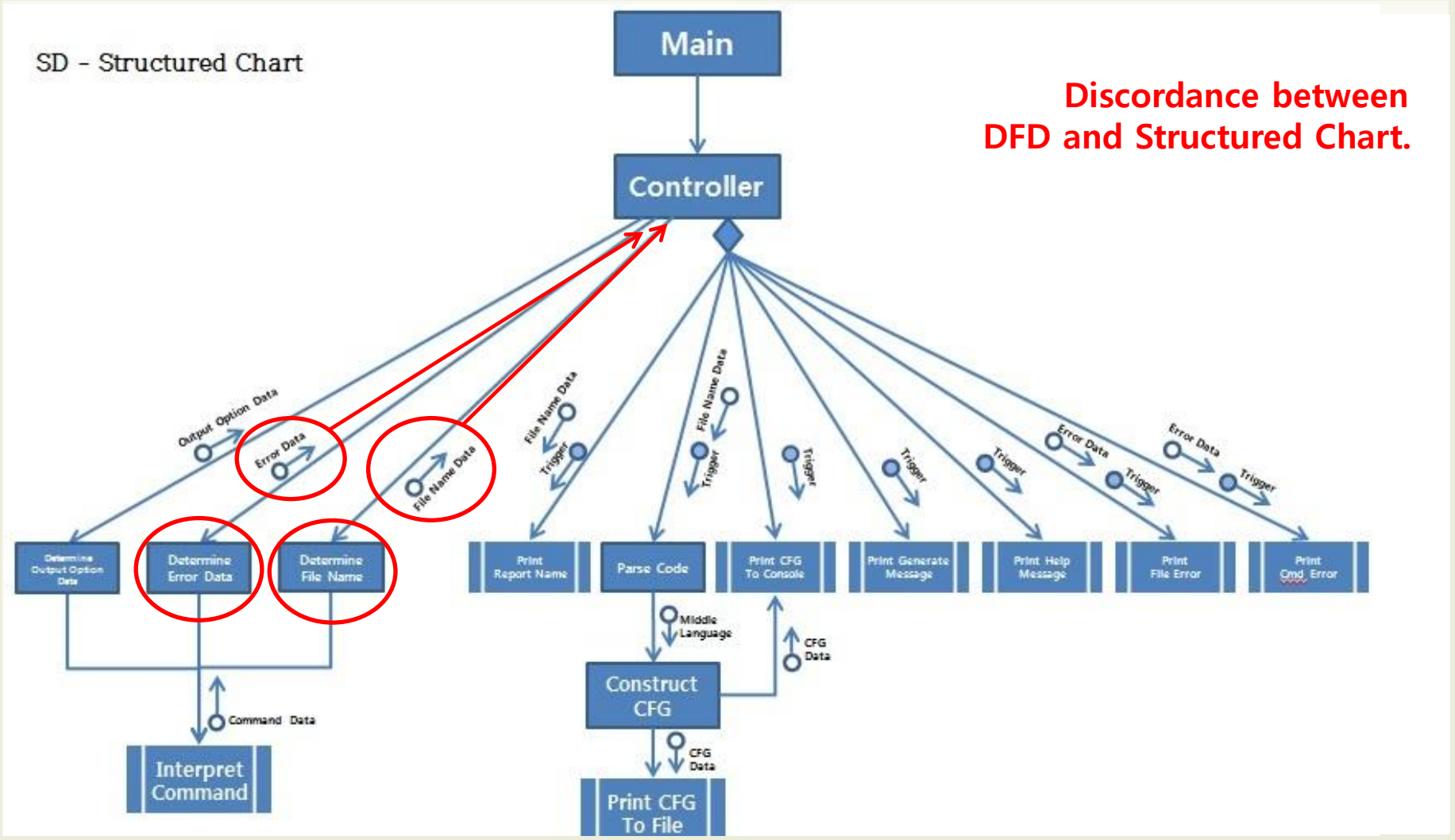
200711463 이준하

Contents

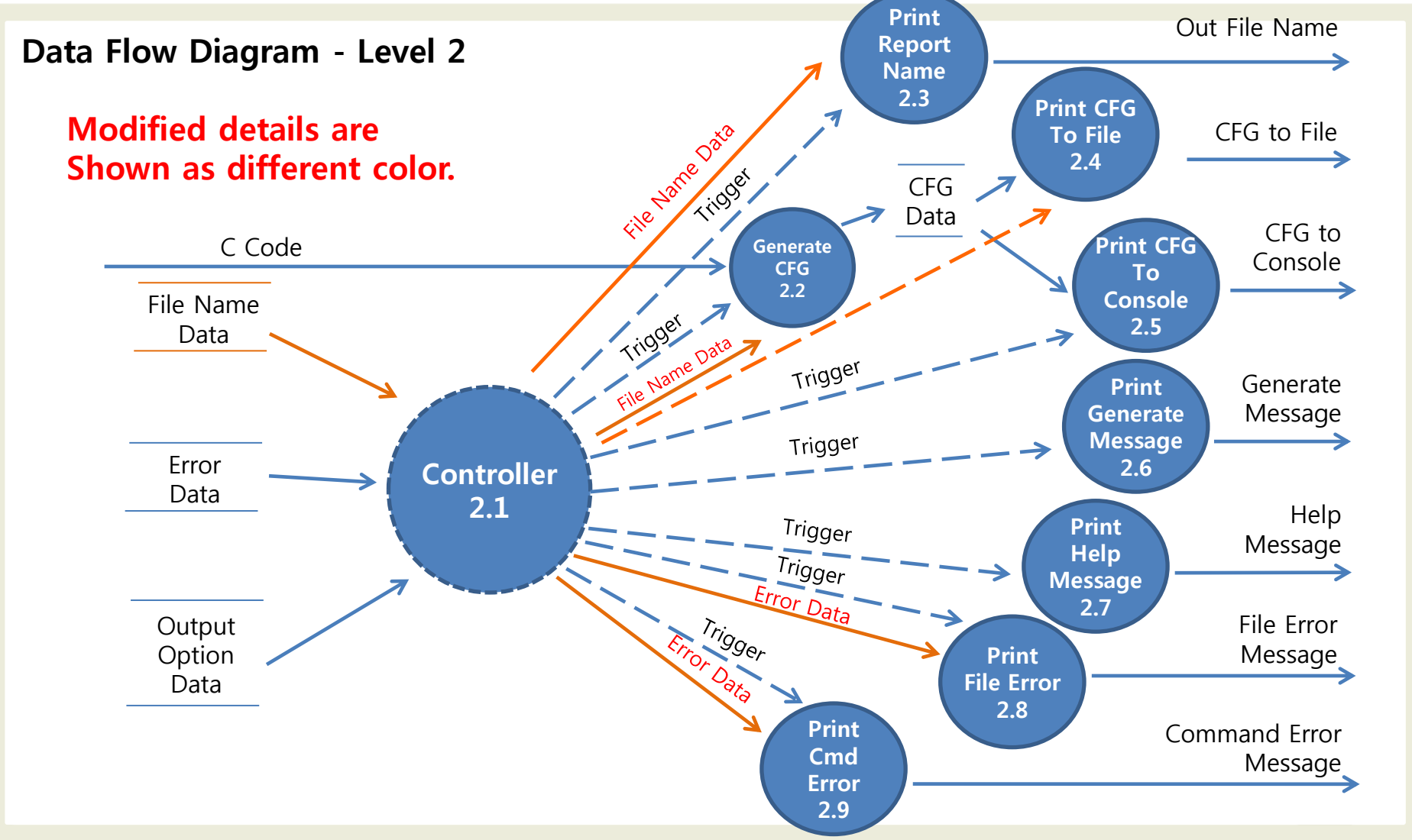
- Modified details of the original SASD(T6).
- C code explanations of finished CFG Generator.
- Examples of the program execution.

Modified details of the original SASD.

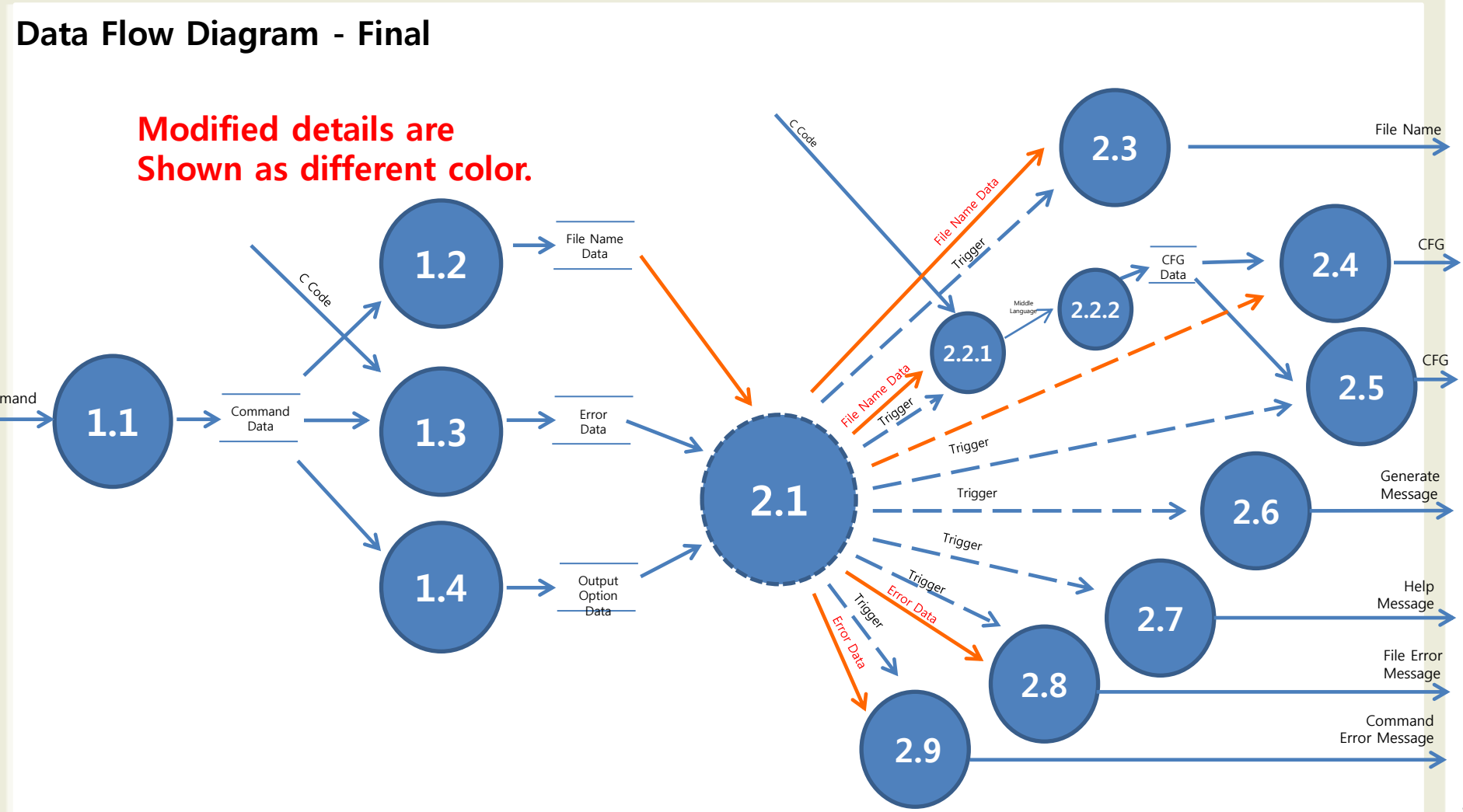
Modified detail – Original Slide #1



Modified detail – Modified Slide #1

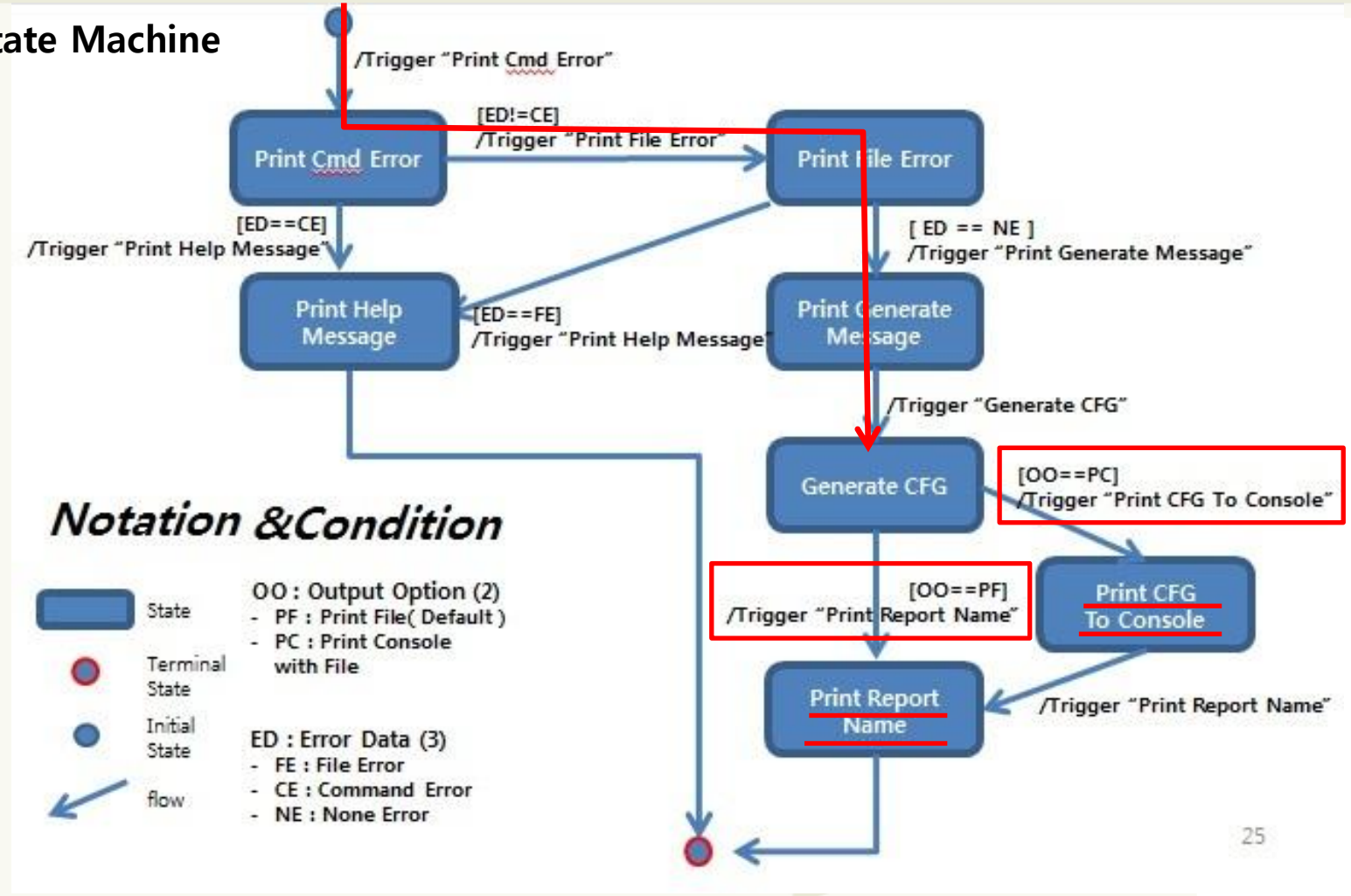


Modified detail – Modified Slide #1

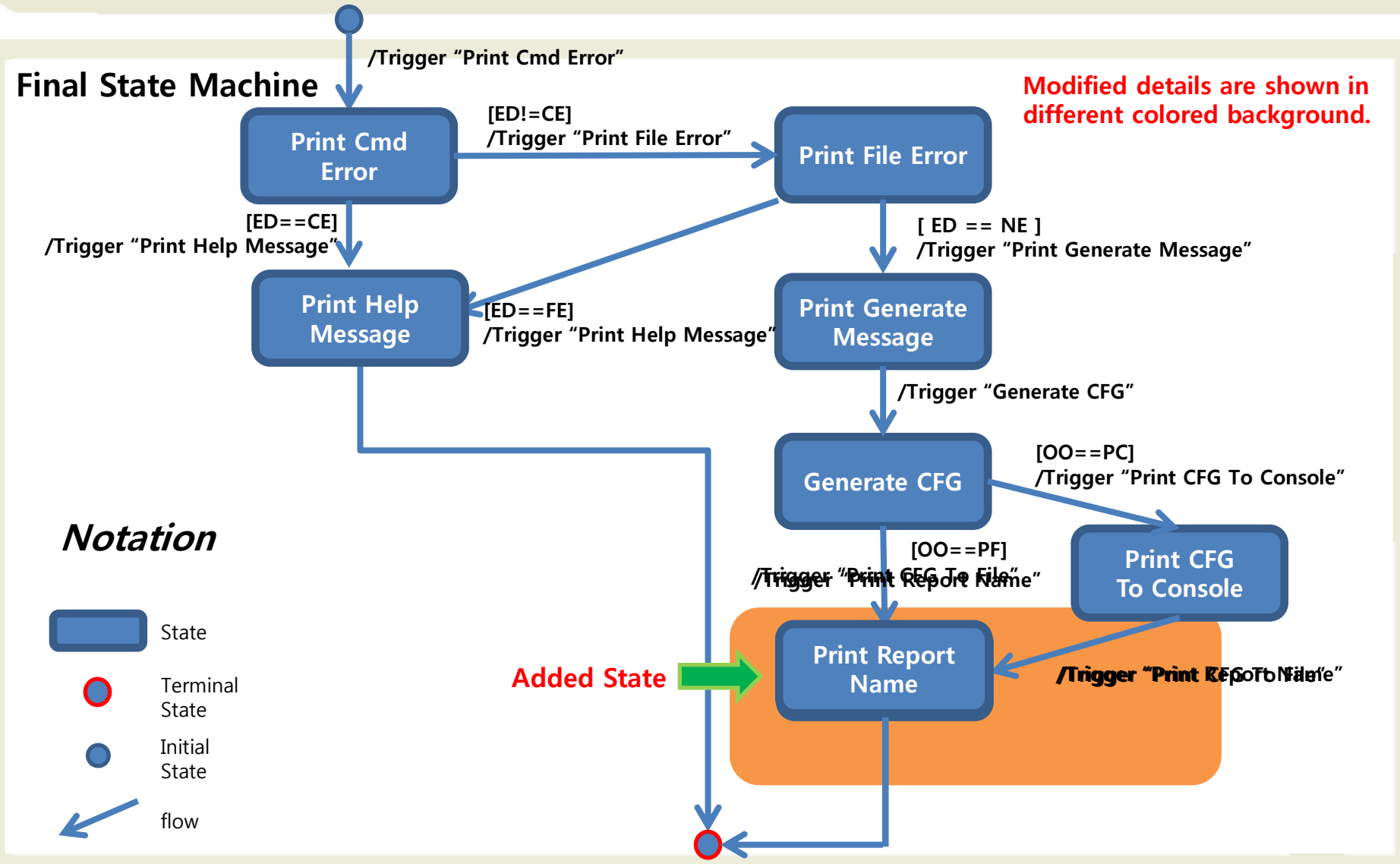


Modified detail – Original Slide #2

Final State Machine



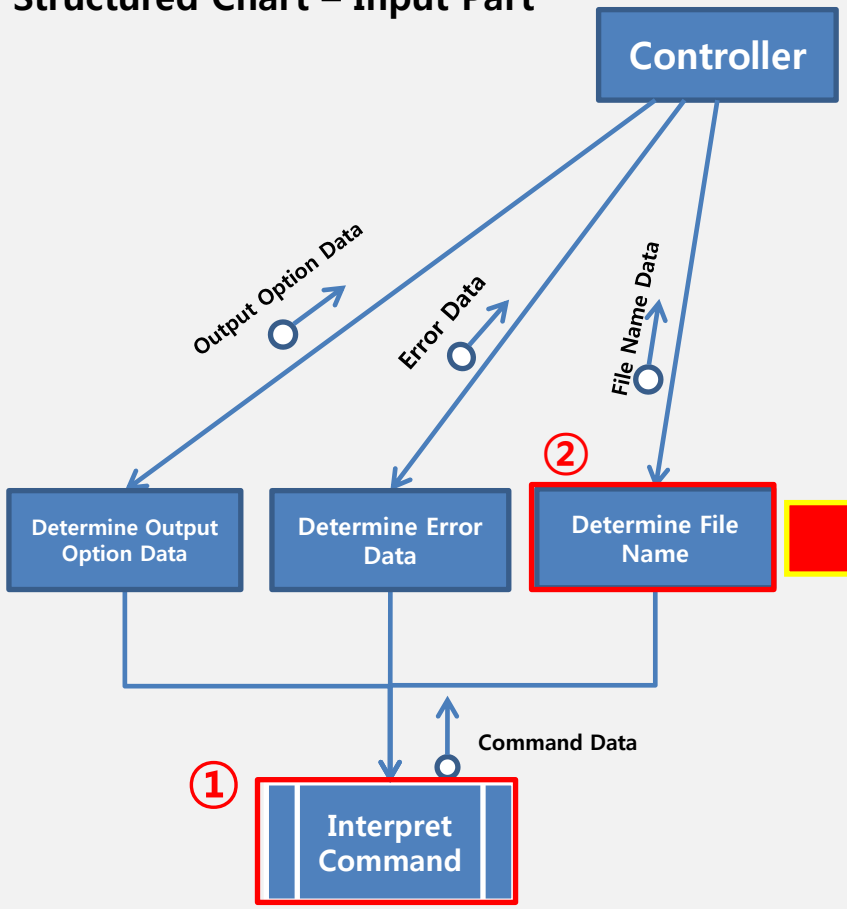
Modified detail – Modified Slide #2



C code explanations of finished CFG Generator.

C Code explanations #1,2.

Structured Chart – Input Part



Programmed Code

① Return the struct CD

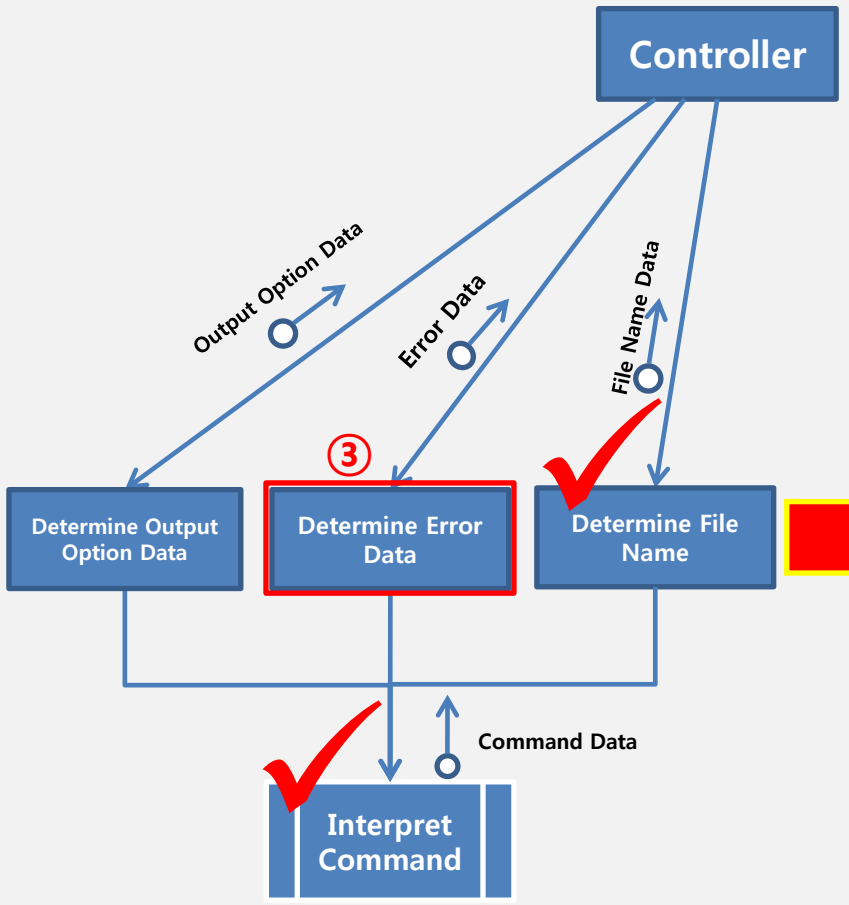
```
CD interpretCommand(char* infile, char* outfile, char* option)
{
        CD command;
        strcpy(command.option, option);
        strcpy(command.infile, infile);
        strcpy(command.outfile, outfile);

        return command;
}
```

② Return the struct FD

```
FD determineFileName(char* argv[])
{
        FD filename;
        CD command = interpretCommand(argv[1], argv[2], argv[3]);

        filename.ipName = (char*) malloc(strlen(command.infile)+1);
        strcpy(filename.ipName, command.infile);
        filename.opName = (char*) malloc(strlen(command.outfile)+1);
        strcpy(filename.opName, command.outfile);

        return filename;
}
```

C Code explanations #3.

Structured Chart – Input Part



Programmed Code ③

```
char* determineErrorData(char *argv[])
{
        char *ED = (char*) malloc(sizeof(char)*2);
        char temp[5] = ".txt";
        char temp1[3] = ".c";
        int i, len, error = 0, error1 = 0, dotNum=0;
        CD command = interpretCommand(argv[1], argv[2], argv[3]);
        FILE *fp;

        len = strlen(command.outfile);
        for(i=3; i>=0; i--){
                if(command.outfile[len - i - 1] != temp[3-i]){
                        error1 = 1;
                        break;
                }
        }

        if(error1==0)
        {
                for(i=0; i<=len; i++){
                        if(command.outfile[i] == '.')
                                dotNum++;
                }
                if(dotNum>=2)
                        strcpy(ED, "CE");
                else
                        strcpy(ED, "NE");
        }
        else
                strcpy(ED, "CE");

        dotNum = 0;
        if(strcmp(ED, "CE"))
        {
                len = strlen(command.infile);
                for(i=1; i>=0; i--){
                        if(command.infile[len - i - 1] != temp1[1-i]){
                                error = 1;
                                break;
                        }
                }

                if(error==0)
                {
                        for(i=0; i<=len; i++){
                                if(command.infile[i] == '.')
                                        dotNum++;
                        }
                        if(dotNum>=2){
                                strcpy(ED, "FE");
                        }
                        else
                                strcpy(ED, "NE");
                }
                else
                        strcpy(ED, "CE");

                if(!strcmp(ED, "NE")){
                        if(!strcmp(command.option, "pf") || !strcmp(command.option, "pc") ||
                                !strcmp(command.option, "PF") || !strcmp(command.option, "PC")){
                                fp = fopen(command.infile, "rt");
                                if(!fp){
                                        strcpy(ED, "FE");
                                }//
                                else
                                        fclose(fp);

                        }
                        else{
                                strcpy(ED, "CE");
                        }
                }
        }
```

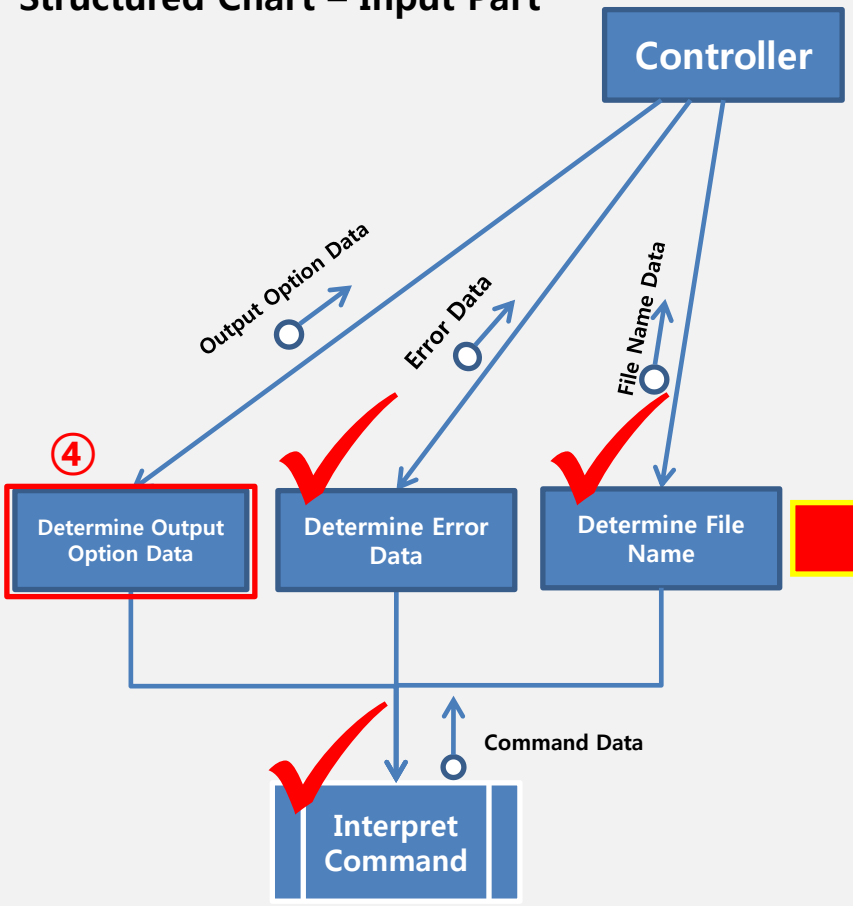
Check the syntax of outfile name from the Command Data.

Check the syntax of infile name from the Command Data.

Check the syntax of option name from the Command Data.

C Code explanations #4.

Structured Chart – Input Part



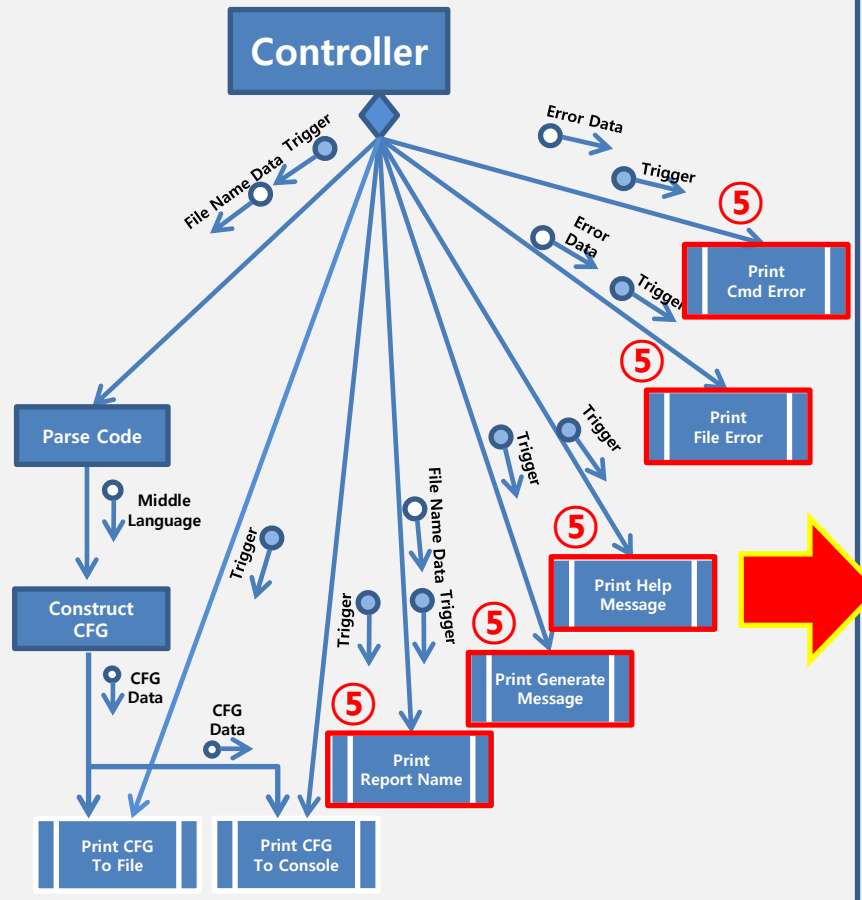
Programmed Code

④

```
char* determineOutputOptionData(char* argv[])
{
        char *option = (char*)malloc(sizeof(char)*2);
        CD command = interpretCommand(argv[1], argv[2], argv[3]);
        if(!strcmp(command.option, "PF"))
                strcpy(option, command.option);
        else if(!strcmp(command.option, "pf"))
                strcpy(option, "PF");
        else if(!strcmp(command.option, "PC"))
                strcpy(option, command.option);
        else if(!strcmp(command.option, "pc"))
                strcpy(option, "PC");
        else
                option = "PF";

        return option;
}
```

C Code explanations #5.

Structured Chart – Output Part



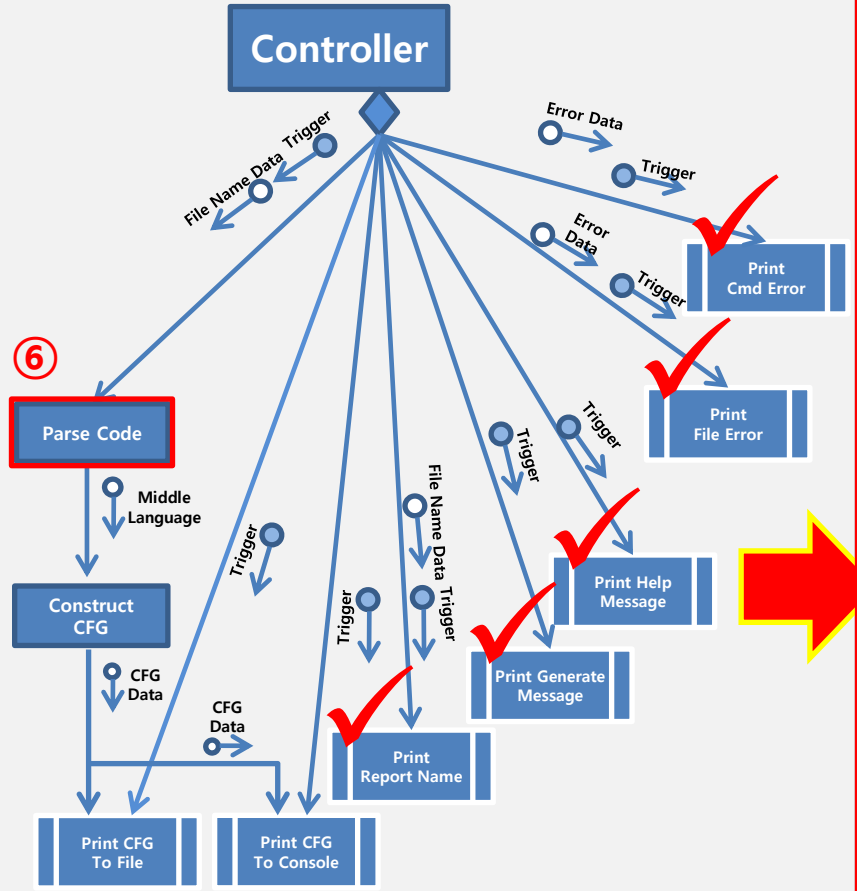
Programmed Code

⑤

```
void printCmdError(char *ED)
{
        if(!strcmp(ED, "CE"))
                printf("Command Error!\n");
}
void printFileError(char *ED)
{
        if(!strcmp(ED, "FE"))
                printf("File Error!\n");
}
void printHelpMessage(void)
{
        printf("==========Help Message========\n");
        printf("Input the correct command.\n");
        printf("    ex) ./CFG input.c result.txt PC\n");
        printf("./Input the execution command, Input the input filename, Input the ou
tput filename, Input the output option.\n");
        printf("Inputfile must be single source file and has no syntax error.(extensi
on : .c)\n");
        printf("Output file is single text file. (extension : .txt)\n Output option c
an accept two type of input(PC,PF).\nIf you input the output option PF, CFG generator
 ouputs the CFG data to file. If you input 'PC', it outputs the CFG data to console a
nd file.\n");
        printf("File Error! : When non-exist-filename inputted. Please check the inpu
tfilename again.\n");
        printf("Command Error! : When incorrect extension inputted by user. Please ma
ke sure the inputfile extension and check the previous correct input command example
again.\n");
        printf("If you input the incorrect Output Option(another input excepts 'PC','
PF'), it doesn't print any other error message. It just outputs the CFG data only to
file. (PrintCFGtoFile is Default)\n");
        printf("=============================\n");
}
void printGenerateMessage(void)
{
        printf("No Error!\n");
        printf("Start generating CFG Data\n");
}
void printReportName(FD filename)
{
        printf("================================\n");
        printf("OutputFile name is %s\n", filename.opName);
        printf("================================\n");
}
```

C Code explanations #6.

Structured Chart – Output Part

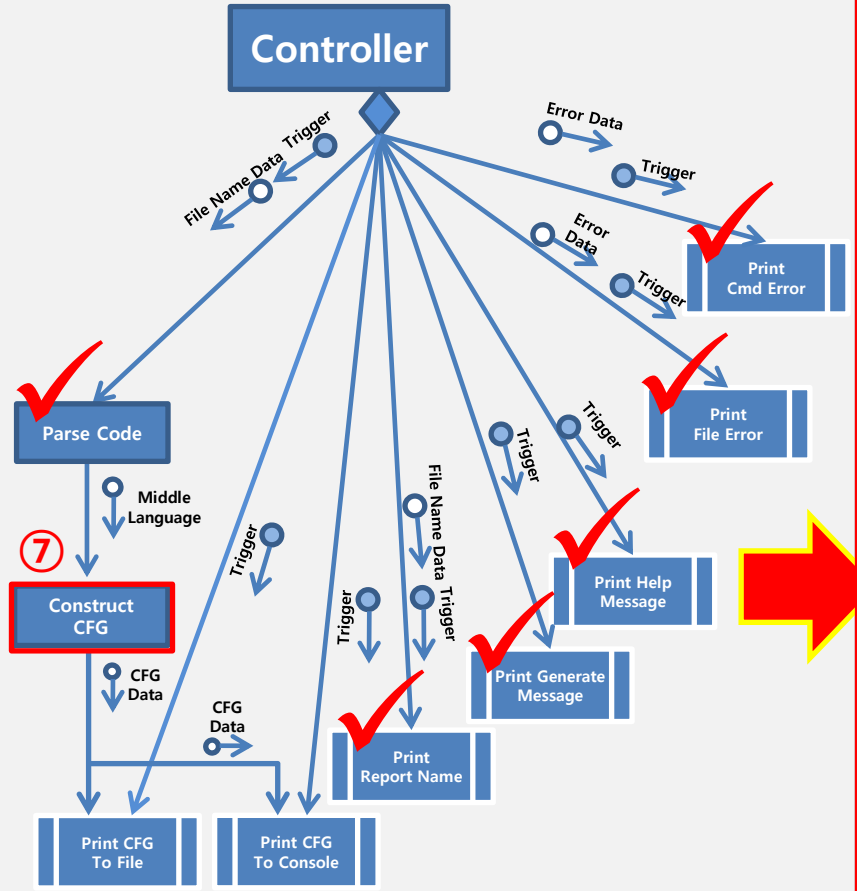


Programmed Code ⑥

```
void parseCode(FD filename)
{
    FILE *fp;
    ML *mo[100] = {NULL, };
    ML *temp;
    char str[256], temp_fname[20], fname[20];
    int value[50] = {0, };
    int value_c[50] = {0, };
    int value_i[50] = {0, };
    int i, j=0;
    int if_block = 0, index_mo = 0, same_func =
    char *index, *index1, *temp_c, *temp_sv;
    fp = fopen(filename.ipName, "rt");
    if(fp!=NULL){
        while(fgets(str, sizeof(str), fp)){
            if(strstr(str, "main")){
                ismain = 1;
                top = 0;
        .....Omitted.....
 if(strstr(str, "}")){
     if(ismain == 1)
     {

         if(value[top-1] != 0){ //int main()의 }이 나올때는 아래코드 실행 안함
             if(strstr(str, "while")){

        .....Omitted.....
 }
 else{//for, while, if, else if, 의 닫는 거에 대한 처리
     temp = ml[value[top-1]];
     if(temp->isback_e == 0){//if, else if
         temp->ptr = (ML*) malloc(sizeof(ML));
         temp->ptr->format = (char*) malloc(sizeof(char)*5);
         strcpy(temp->ptr->format, "JUMP");
         if(value[top]==-1){//이전의 닫힌괄호('}')가 switch 케이스 문이였을 경우
             for(i=before_front_case; i<top_case; i++){
```

C Code explanations #7.

Structured Chart – Output Part



Programmed Code ⑦

```
void ConstructCFG(void)
{
        int i=0;
        int j=0;
        int k=0;
        CFGdata *temp;
        while(j<sum)
        {
                cfg[j] = (CFGdata*)malloc(sizeof(CFGdata));
                for(i=0;i<20;i++)
                {
                        cfg[j]->cblock[i] = NULL;
                }
                cfg[j]->cnum = ml[j]->num;
                if(!(strcmp(ml[j]->format, "BODY")))
                {
                        cfg[j]->block = (char*)malloc(sizeof(char)*8);
                        cfg[j]->cname = (char*)malloc(strlen(ml[j]->detail)+1);
                        strcpy(cfg[j]->block ,"Code : ");
                        strcpy(cfg[j]->cname, ml[j]->detail);
                }
                else if(!strcmp(ml[j]->format, "COND"))
                {
                                ....
                        }       ...Omitted..
                        k=0;
                }
                else;

                if(!strcmp(ml[j]->format,"FUNC"))
                {
                        i=0;
                        while(ml[j]->block[i] != NULL)
                        {
                                if(ml[j]->block[i]->ptr == NULL){

                                ...Omitted...



j=0;
k=0;
while(j<sum)
{
        if(j==sum-1)
        {
                cfg[j]->edge = -1;
                cfg[j]->jumpedge = -1;
                cfg[j]->backedge = -1;
                j++;
        }
        else
        {
                if(strcmp(ml[j]->format,"FUNC"))
                {
                        if(cfg[j]->cisback_e == 0)
                        {
                                if(ml[j]->ptr == NULL)
                                {
                                        cfg[j]->edge = ml[j]->num+1;
                        {
                                cfg[j]->cisback_e = 0;
                        }
                }
                else                    omitted...
                {
                                {
                                        cfg[j]->cisback_e = 0;
                                }
                        }
                        else
                                cfg[j]->cisback_e = 0;
                }
                j++;
        }
```

According to the keyword of Middle Language, Specify the block and code.

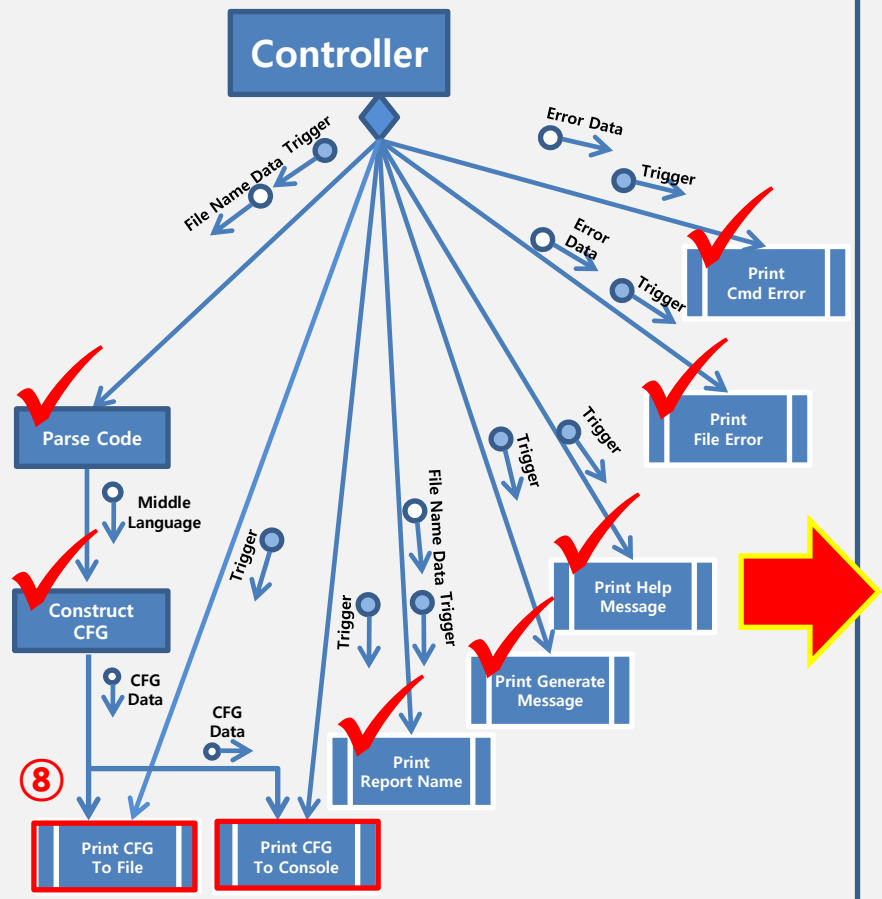
Specifying about the user-define-function.

Specifying Edges.

Specifying Edges.

C Code explanations #8.

Structured Chart – Output Part



Programmed Code ⑧

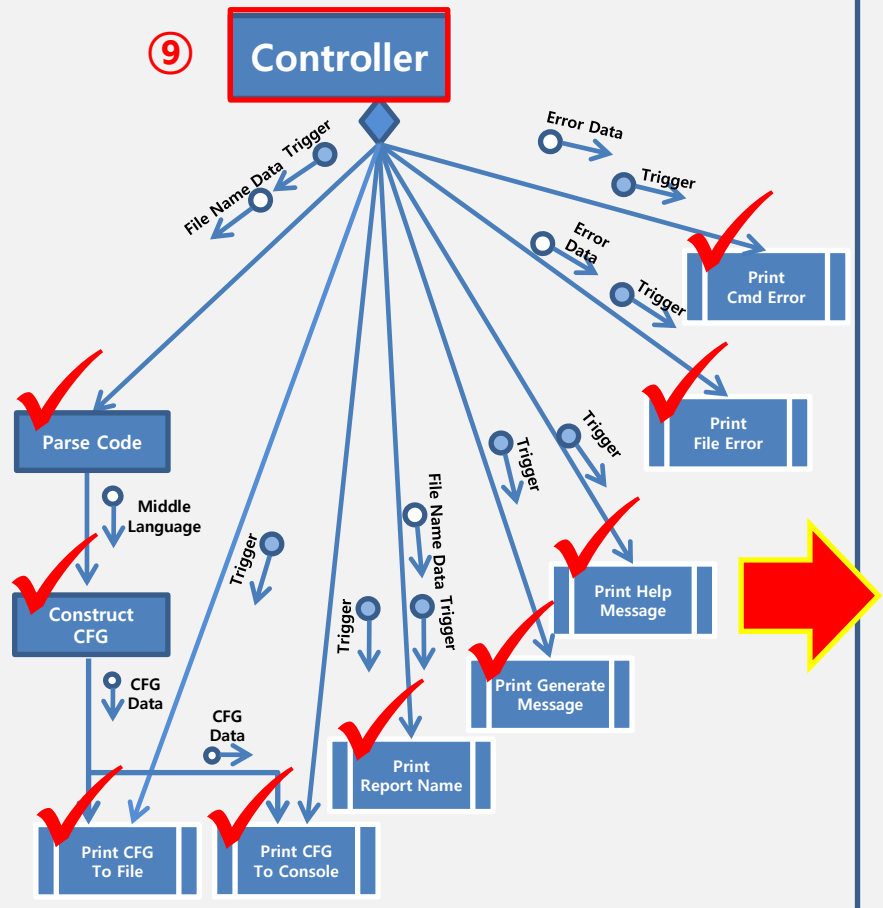
```
void PrintCFGtoConsole(CFGdata *cfg[])
{
        int j=0;
        int k=0;
        while(j<sum)
        {
                if(strcmp(cfg[j]->format, "FUNC"))
                {
                        printf("B : #%d\n",cfg[j]->cnum);
                        printf("%s %s\n",cfg[j]->block,cfg[j]->cname);
                        if(j==sum-1)
                        {
                                printf("E : End\n");
                                j++;
                        }
                        else
                        {
                                if(cfg[j]->cisback_e == 0)
                                {
                                        if(cfg[j]->jumpedge == -1)

                                ....Omitted....

                                printf("\n");
                        }
                }
                else
                {
                        printf("B : #%d\n",cfg[j]->cnum);
                        printf("%s %s\n",cfg[j]->block,cfg[j]->cname);
                        if(j==sum-1)
                        {
                                printf("E : End\n");
                                j++;
                        }
                        else
                        {
                                else if(cfg[j]->cblock[k]->edge != -1)

                                ....Omitted....

                                else
                                {
                                        printf("E : #%d->#%d, #%d->#%d\n",cfg[j]
[k]->backedge);
                                }
                                printf("\n");
                        }
                        k++;
                }
                j++;
        }
}
```

Print CFG data of main-function part.

Print CFG data of User-define-function part.

C Code explanations #9.

Structured Chart – Output Part

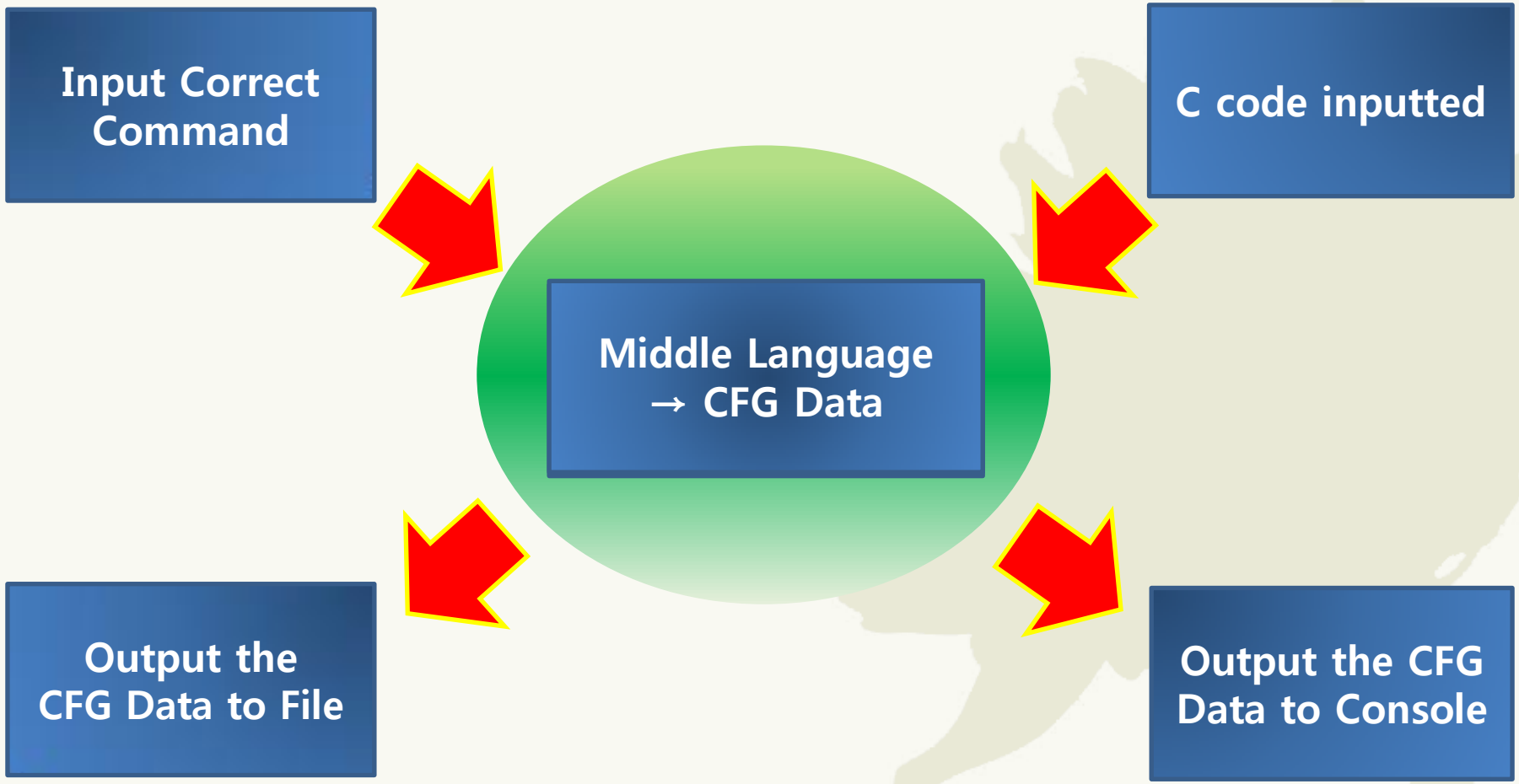


Programmed Code ⑨

```
void controller(int argc, char* argv[]){
        char *ED, *OO;
        FD filename;
        ED = determineErrorData(argc,argv);

        printCmdError(ED);
        if(!strcmp(ED, "CE")){
                printHelpMessage();
                return;
        }
        else{
                filename = determineFileName(argv);
                OO = determineOutputOptionData(argv);
                printFileError(ED);
                if(!strcmp(ED, "FE")){
                        printHelpMessage();
                        return;
                }
                else{
                        printGenerateMessage();
                        parseCode(filename);
                        if(!strcmp(OO,"PC")){
                                PrintCFGtoConsole(cfg);
                        }
                        PrintCFGtoFile(cfg,argv[2]);
                        printReportName(filename);
                }
        }
}

int main(int argc, char *argv[])
{
        controller(argc,argv);
        return 0;
}
```

Examples of the program execution.

Brief code flow of our CFG Generator.

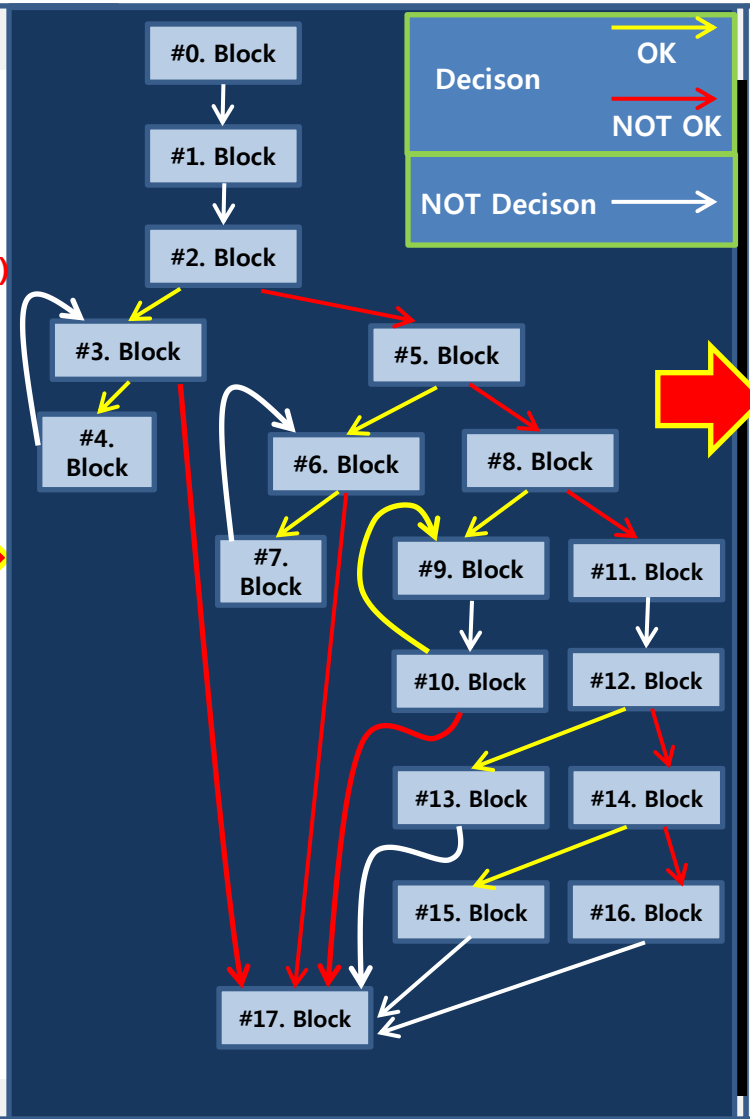


Example of execution #1.

-When statements are included in 'switch-case'.

① Input Source code

```
int main(void){ (0)
    int i = 0;
    int input;
    scanf("%d",input); (1)

    switch(input){
        case 1 : (2)
            for(i = 0; i<10; i++){ (3)
                printf("1"); (4)
            }
            break;
        case 2 : (5)
            while(i <10){ (6)
                printf("2"); (7)
                i++;
            }
            break;
        case 3 : (8)
            do{
                printf("3"); (9)
                i++;
            }while(j<10); (10)
            break;
        default : (11)
            if(i = 0){ (12)
                printf("4"); (13)
            }
            else if(i = 1){ (14)
                printf("5"); (15)
            }
            else{
                printf("6"); (16)
            }
    }
    return 0; (17)
}
```



③ Final CFG format

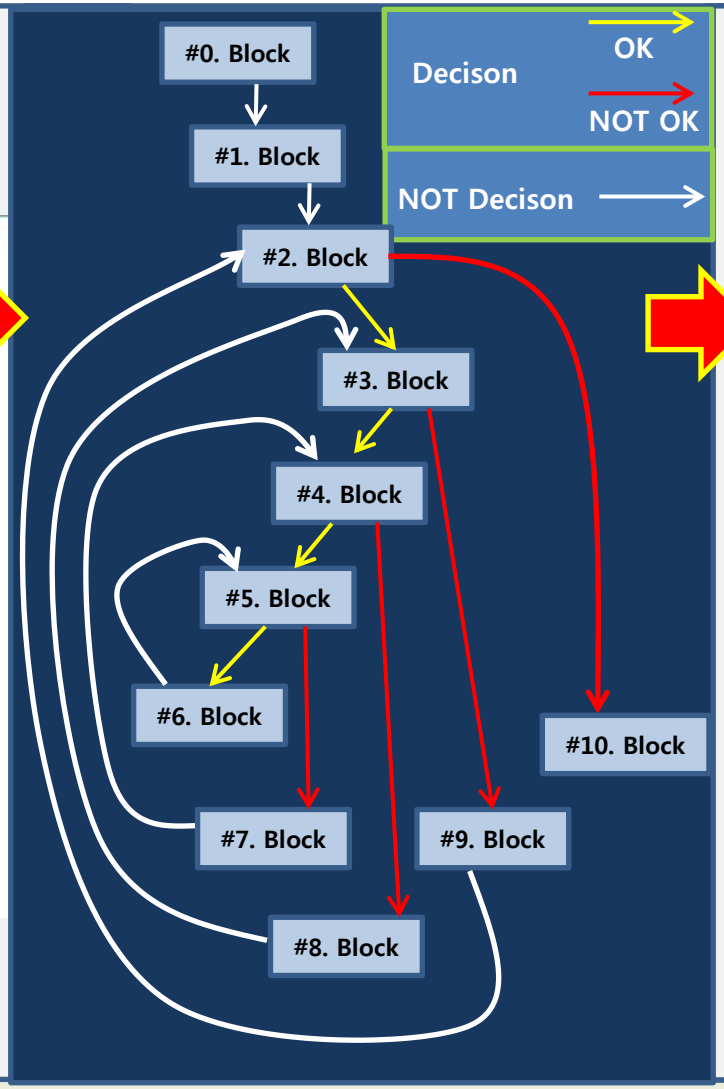
```
B : #0
Code :  int main(void) (0)
E : #0->#1

B : #1
Code :  int i = 0; int input; scanf("%d",input (1)
E : #1->#2

B : #2
Condition :  (input) 1 : (2)
E : #2->#3, #2->#5

B : #3
Condition :  (i = 0; i<10; i++) (3)
E : #3->#4, #3->#17

B : #4
Code :  printf("1"); (4)
E : #4->#3

B : #5
Condition :  (input) 2 : (5)
E : #5->#6, #5->#8

B : #6
Condition :  (i <10) (6)
E : #6->#7, #6->#17

B : #7
Code :  printf("2");  i++; (7)
E : #7->#6

B : #8
Condition :  (input) 3 : (8)
E : #8->#9, #8->#11

B : #9
Code :  printf("3");  i++; (9)
E : #9->#10

B : #10
Condition :  (j<10); (10)
E : #10->#17, #10->#9

B : #11
Condition :  default (11)
E : #11->#12

B : #12
Condition :  (i = 0) (12)
E : #12->#13, #12->#14

B : #13
Code :  printf("4"); (13)
E : #13->#17

B : #14
Condition :  (i = 1) (14)
E : #14->#15, #14->#16

B : #15
Code :  printf("5"); (15)
E : #15->#17

B : #16
Code :  printf("6"); (16)
E : #16->#17

B : #17
Code :  return 0; (17)
E : End
```

Example of execution #2.

-The repetitive case in overlapping statements.

① Input Source code

```
int main(void)                    (0)
{
    int i,j,k = 0;                (1)

    for(i=0;i<10;i++)             (2)
    {
        for(j=0;j<10;j++)         (3)
        {
            for(k=0;k<10;k++)     (4)
            {
                while(i = 0)      (5)
                {
                (6) printf("here0");
                }
                printf("here1");  (7)
            }
            printf("here2");      (8)
        }
        printf("here3");          (9)
    }
    printf("here4");
                                  (10)
    return 0;
}
```



③ Final CFG format

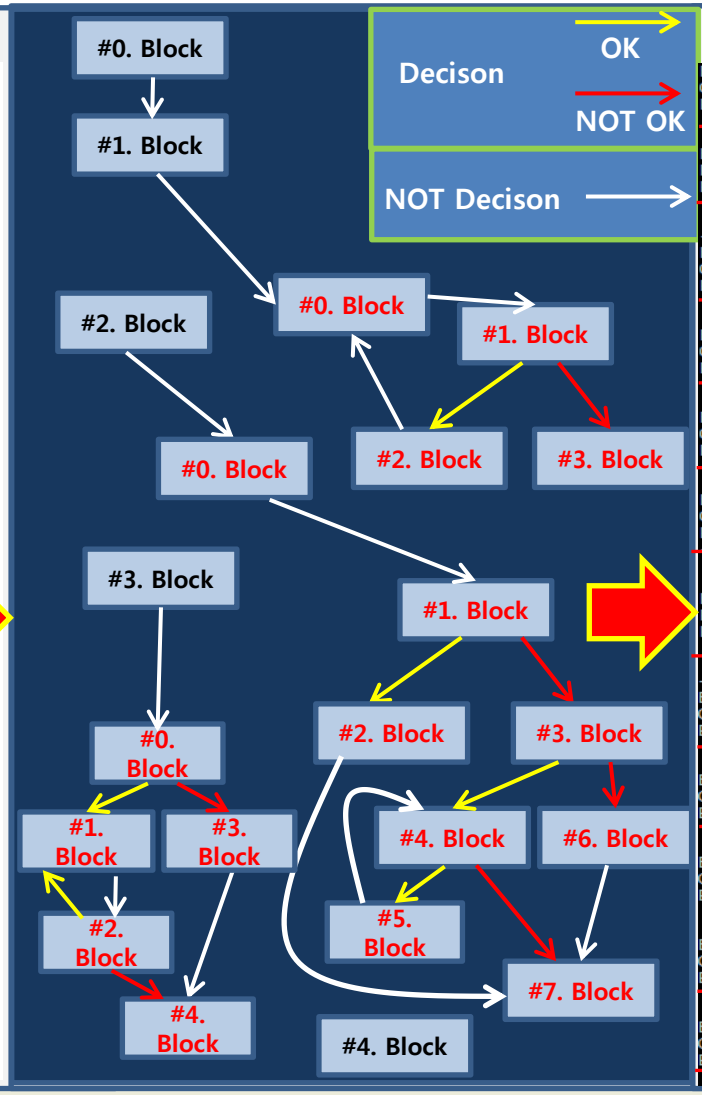
```
B : #0
Code :  int main(void)                    (0)
E : #0->#1

B : #1
Code :  int i,j,k = 0;                    (1)
E : #1->#2

B : #2
Condition :  (i=0;i<10;i++)               (2)
E : #2->#3, #2->#10

B : #3
Condition :  (j=0;j<10;j++)               (3)
E : #3->#4, #3->#9

B : #4
Condition :  (k=0;k<10;k++)               (4)
E : #4->#5, #4->#8

B : #5
Condition :  (i = 0)                      (5)
E : #5->#6, #5->#7

B : #6
Code :  printf("here0");                  (6)
E : #6->#5

B : #7
Code :  printf("here1");                  (7)
E : #7->#4

B : #8
Code :  printf("here2");                  (8)
E : #8->#3

B : #9
Code :  printf("here3");                  (9)
E : #9->#2

B : #10
Code :  printf("here4"); return 0;        (10)
E : End

Jin@Jin-PC ~
$
```

Example of execution #3.

-When the user-define-function has branches.

① Input Source code

```
int func1(int x){
    int i;  (0)
    for(i = 0; i<10; i++){ (1)
        x+=1  (2)
    }
    return x;  (3)
}
void func2(int y){
    int k = 10;  (0)
    switch(y){
    case 1:  (1)
        printf("hey\n");  (2)
        break;
    case 2:  (3)
        while(k>10){  (4)
            k--;  (5)
        }
        break;
    default :  (6)
        break;
    }
    return 0;  (7)
}
float func3(float z){
    if(z>0){  (0)
        do{
            z--;  (1)
        }while(k>1);  (2)
    }
    else{
        printf("end");  (3)
    }
    return z;  (4)
}
void main(void){  (0)
    func1(1);  (1)
    func2(3);  (2)
    func3(3.5);  (3)
    return 0;  (4)
}
```



③ Final CFG output format

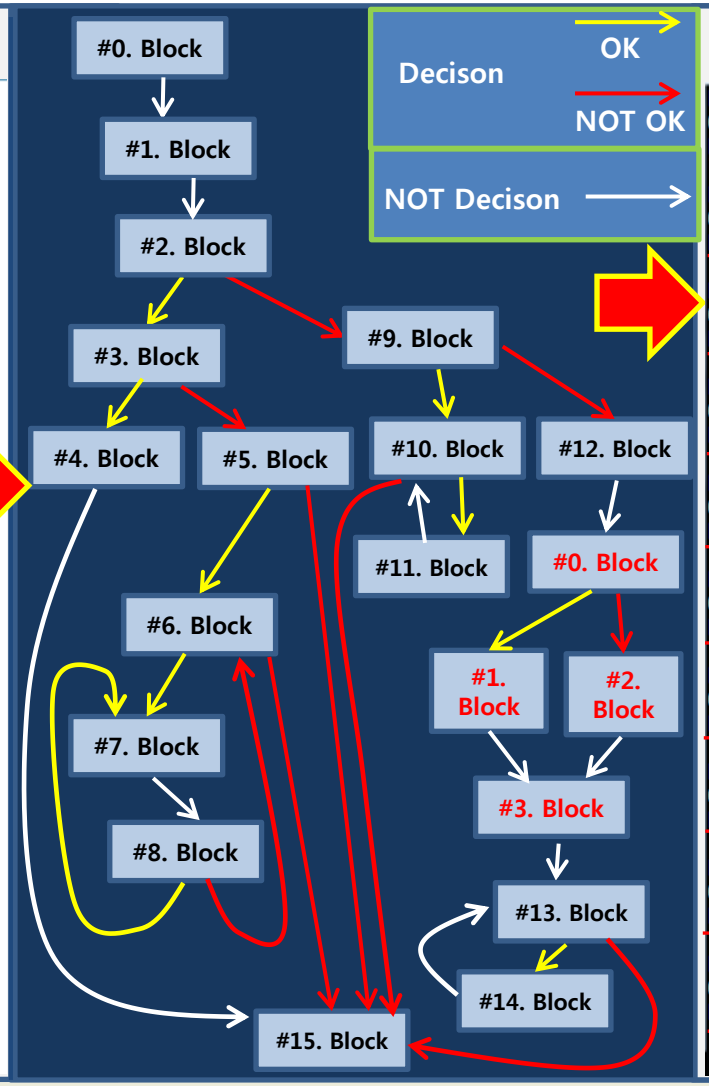
```
B : #0
Code :  void main(void)
E : #0->#1
(0)

B : #1
Function :  func1
E : #1->#2
(1)

-------Block in User_function---
B : #0
Code :  int i;
E : #0->#1
(0)

B : #1
Condition :  (i = 0; i<10;i++)
E : #1->#2, #1->#3
(1)

B : #2
Code :  x+=1
E : #2->#1
(2)

B : #3
Code :  return x;
E : End(User_function)
(3)

B : #2
Function :  func2
E : #2->#3
(2)

-------Block in User_function---
B : #0
Code :  int k = 10;
E : #0->#1
(0)

B : #1
Condition :  (y) 1:
E : #1->#2, #1->#3
(1)

B : #2
Code :  printf("hey\n");
E : #2->#7
(2)

B : #3
Condition :  (y) 2:
E : #3->#4, #3->#6
(3)

B : #4
Condition :  (k>10)
E : #4->#5, #4->#7
(4)

B : #5
Code :  k--;
E : #5->#4
(0)

B : #6
Condition :  default
E : #6->#7, #6->#7
(0)

B : #7
Code :  return 0;
E : End(User_function)
(0)

B : #3
Function :  func3
E : #3->#4
(3)

-------Block in User_function---
B : #0
Condition :  (z>0)
E : #0->#1, #0->#3
(0)

B : #1
Code :  z--;
E : #1->#2
(1)

B : #2
Condition :  (k>0);
E : #2->#4, #2->#1
(2)

B : #3
Code :  printf("end");
E : #3->#4
(3)

B : #4
Code :  return z;
E : End(User_function)
(4)

B : #4
Code :  return 0;
E : End
(4)

Jin@Jin-PC ~
```

Example of execution #4.

-The combinational case.

① Input Source code

```
int user_func(int x, int y)
{
    if(x > y){ (0)
        x+=y; (1)
    }
    else{
        x+=y; (2)
    }
    return x+y; (3)
}
int main(){ (0)
    int i, j, k = 0; (1)
    if(i != 0){ (2)
        switch(j){
            case 1: (3)
                (4) printf("hey\n");
                break;
            case 2: (5)
                (6) for(k = 0; k <10; k++){
                    do{ (7)
                        printf("yes\n");
                    }while(k<4); (8)
                }
                break;
        }
    }
    else if(i>0){ (9)
        for(j=0;j<10;j++){ (10)
            printf("ho\n"); (11)
        }
    }
    else{
        user_func(i,j); (12)
        while(k>0){ (13)
            k--; (14)
        }
    }
    return 0; (15)
}
```



③ Final CFG output format

```
B : #0
Code :  int main()  (0)
E : #0->#1

B : #1
Code :  int i, j, k = 0;  (1)
E : #1->#2

B : #2
Condition :  (i != 0)  (2)
E : #2->#3, #2->#9

B : #3
Condition :  (j) 1:  (3)
E : #3->#4, #3->#5

B : #4
Code :  printf("hey\n");  (4)
E : #4->#9

B : #5
Condition :  (j) 2:  (5)
E : #5->#6

B : #6
Condition :  (k = 0; k <10; k++)  (6)
E : #6->#7, #6->#9

B : #7
Code :  printf("yes\n");  (7)
E : #7->#8

B : #8
Condition :  (k<4);  (8)
E : #8->#15, #8->#7

B : #9
Condition :  (i>0)  (9)
E : #9->#10, #9->#12

B : #10
Condition :  (j=0;j<10;j++)  (10)
E : #10->#11, #10->#15

B : #11
Code :  printf("ho\n");  (11)
E : #11->#10

B : #12
Function :  user_func  (12)
E : #12->#13

-------Block in User_function-
B : #0
Condition :  (x > y)  (0)
E : #0->#1, #0->#2

B : #1
Code :  x+=y;  (1)
E : #1->#3

B : #2
Code :  x*=y;  (2)
E : #2->#3

B : #3
Code :  return x+y;  (3)
E : End(User_function)
------------------------

B : #13
Condition :  (k>0)  (13)
E : #13->#14, #13->#15

B : #14
Code :  k--;  (14)
E : #14->#13

B : #15
Code :  return 0;  (15)
E : End
```

Thanks a lot!
Any Questions?