KU KONKUK UNIVERSITY

Team Presentation #5

- CFG Generator

TEAM [T2]

speaker 200811415 김영현
200811457 조성우
200811465 허준행

Project Overview

- First we take and analyze the team T5's project.
- If necessary, we have to modify and supplement it. (we will mark red)
- we do this project with the purpose of final implementation.

KU KONKUK UNIVERSITY

Contents

- Structured Analysis
- Structured Design
- Source Code
- Demonstration

KU KONKUK UNIVERSITY

Structured Analysis

Statement of Purpose [1/2]

CFG(Control Flow Graph) Generator

- Convert a C source code to CFG.
- CFG is printed in sorted text form according to each block's level.
  - Each block has block's name and major contents to be shown.
  - Each edge has a data of block to be connected.
  - Related block and edge are printed in the same line.
- Restriction of Input C source code
  - The C source code has 100~200 lines which includes main function.
  - It doesn't include pointers.
  - It is a single-file that doesn't have user defined header files.
  - Body of user defined functions should be ahead of main function.
  - Conditional statements should be wrapped in a brace({ }) at the next line of corresponding keyword.

KU KONKUK UNIVERSITY

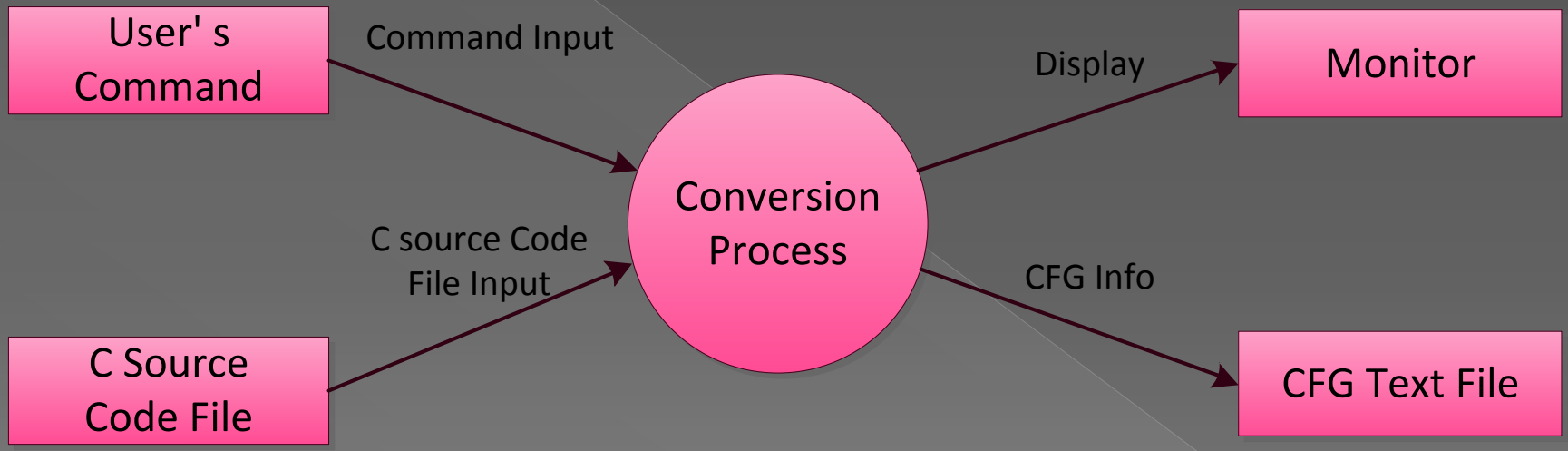
Statement of Purpose [2/2]

- When a user inputted incorrect command, the program show "help" that includes command syntax.
- When C source code inputted successfully, the program shows "success" message. Or in error case, the program shows "error" and terminates the program.
- Before the program converting CFG, shows "converting" message.
- After report generating process, the program shows the name of report file.
- Edges Recognition Algorithm, Basic Block Construction Algorithm, BFS Positioning Algorithm are used.

supplement

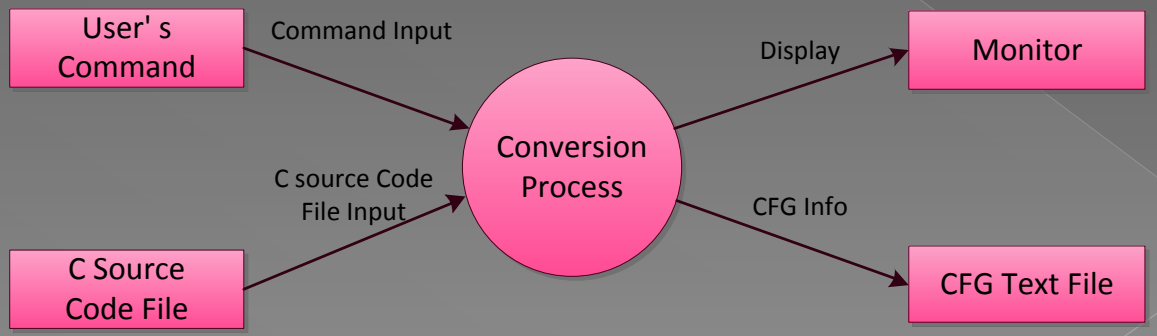
KU KONKUK UNIVERSITY

System Context Diagram



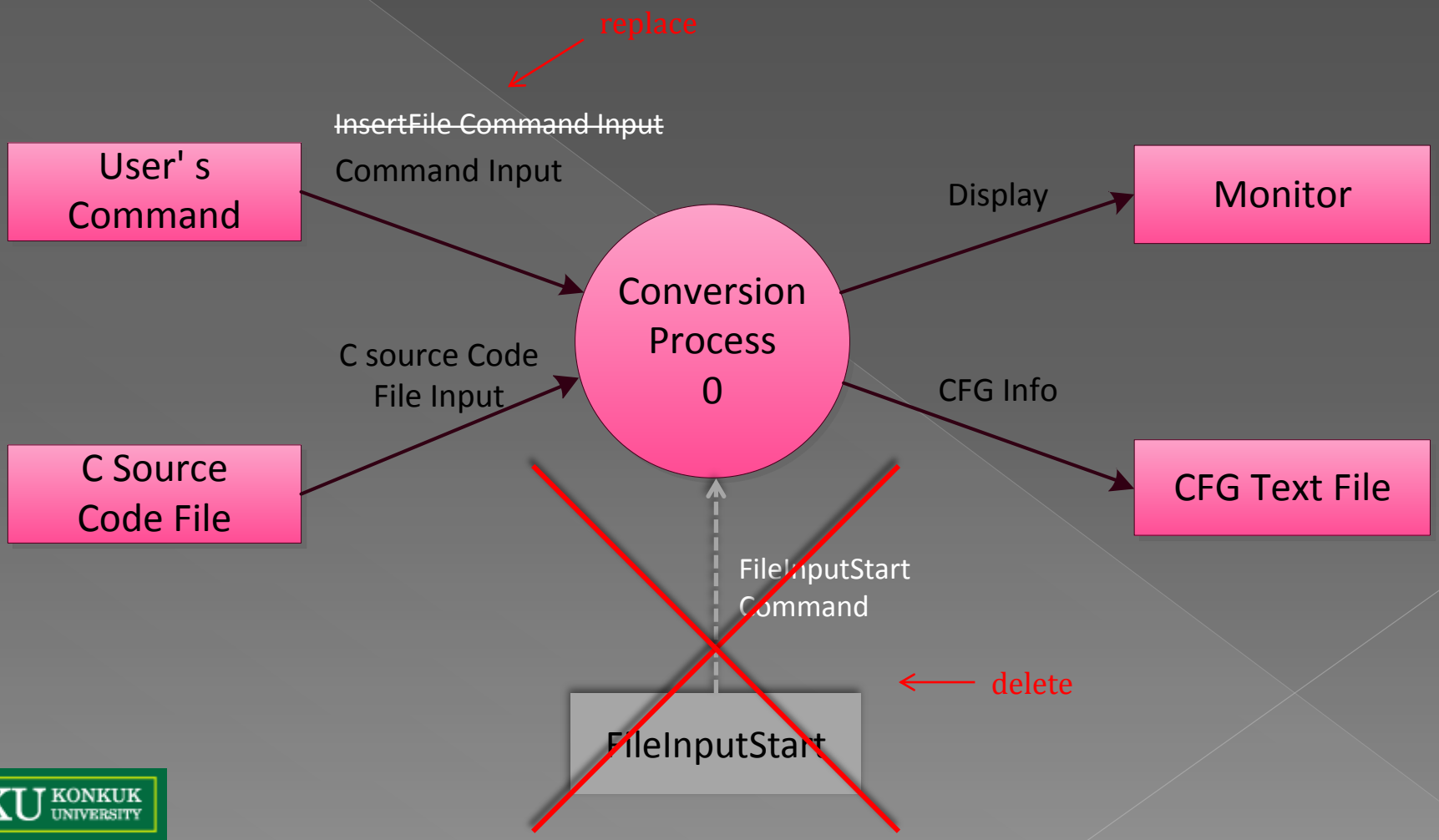
Event List

Input / Output Event	Description
Command Input	Receives a command from ***User's Command***
C Source Code File Input	Receives a C source code file
Display	Prints Commands' result and CFG's process to ***Monitor***
CFG Info	Prints converted CFG to ***CFG Text File***



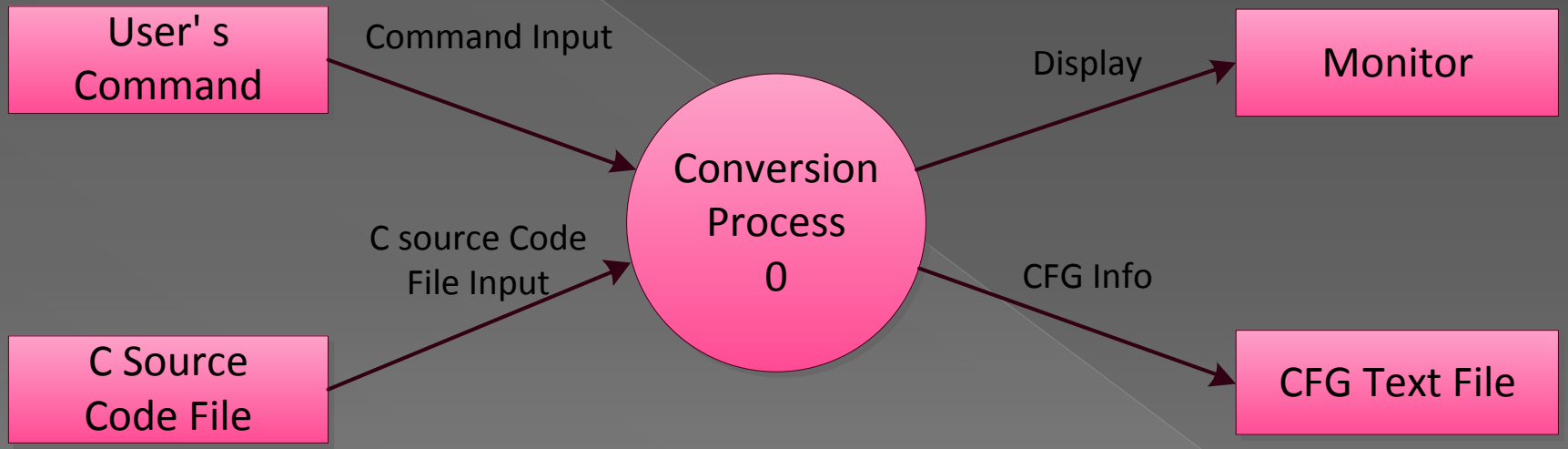
KU KONKUK UNIVERSITY

Data Flow Diagram – level 0



KU KONKUK UNIVERSITY

Data Flow Diagram – level 0

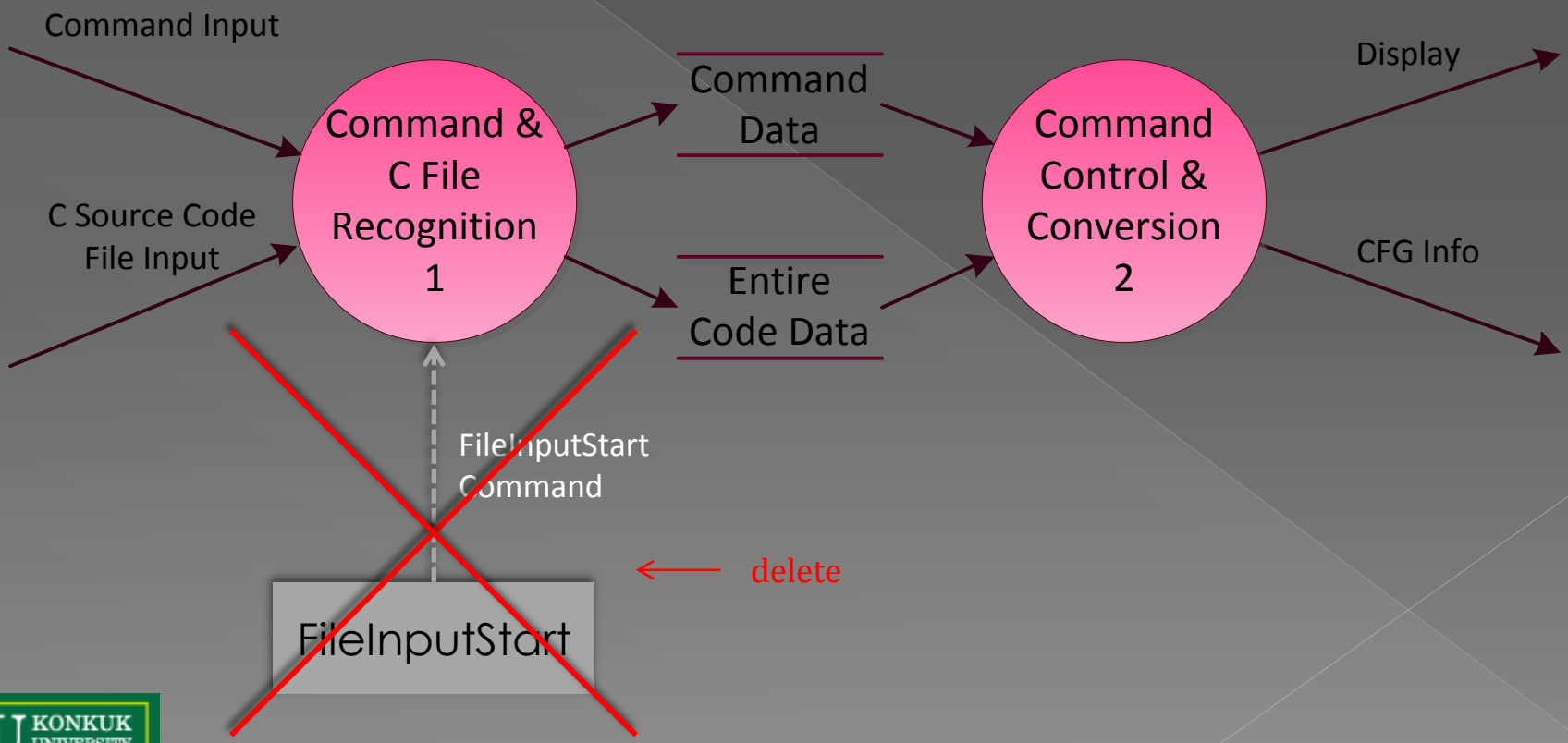


Data Dictionary – level 0

Data name	Description
Command Input	It is received command from ***User's Command***.
C Source Code File Input	It is inputted C source code file.
CFG Info	It is CFG data converted from inputted C source code.
Display	It is an output data contains system message caused by ***Command Info*** and CFG's process result to be sent to ***Monitor***.

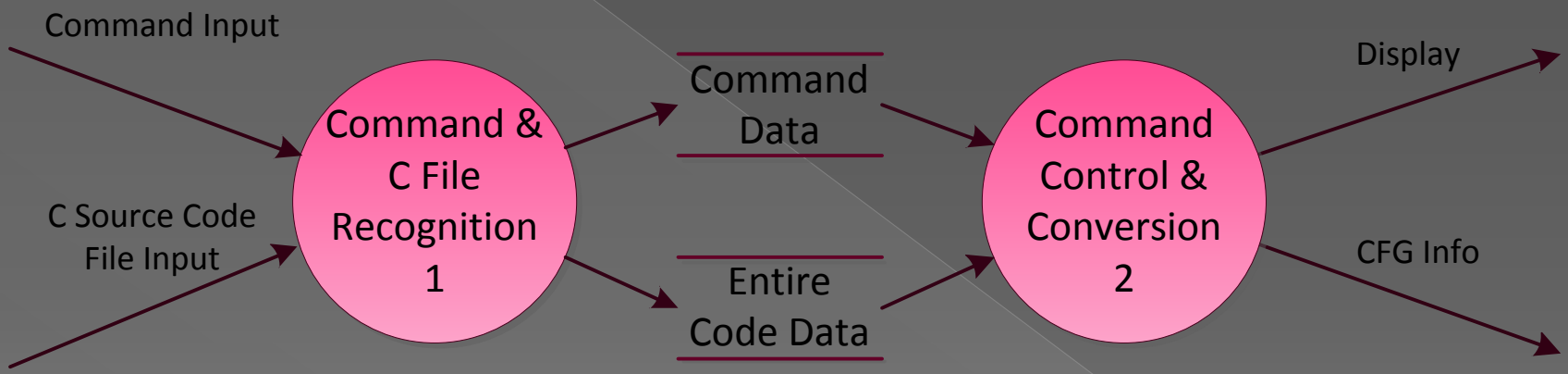
KU KONKUK UNIVERSITY

Data Flow Diagram – level 1



KU KONKUK UNIVERSITY

Data Flow Diagram – level 1



KU KONKUK UNIVERSITY

Data Dictionary – level 1

Command Data	Description
Correctness Info	A boolean value meaning correctness of received command. (True/False)

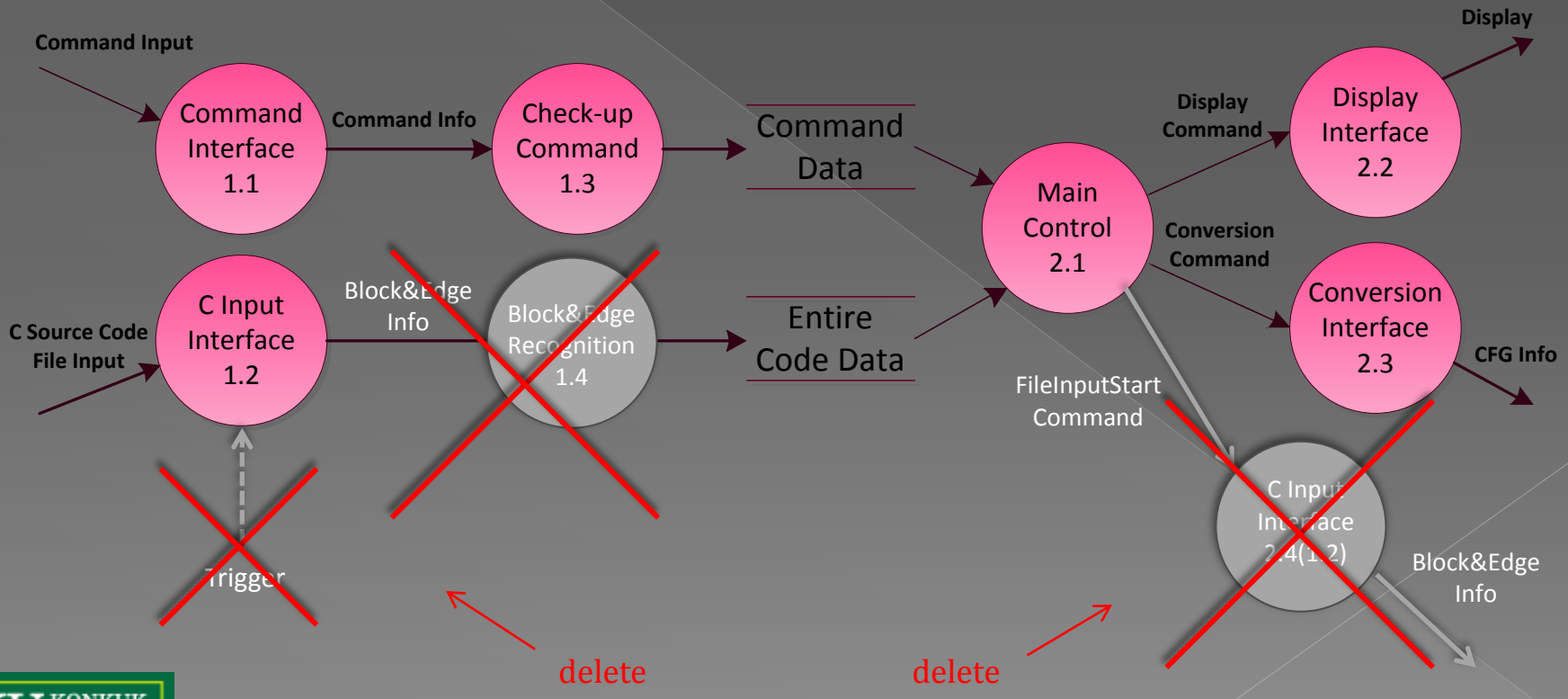
revise : exclude filename (string)

Entire Code Data	Description
Numbered Code Data	It consists of ***C Source Code File Input*** and each corresponding line number.

revise : from Block&Edge Data

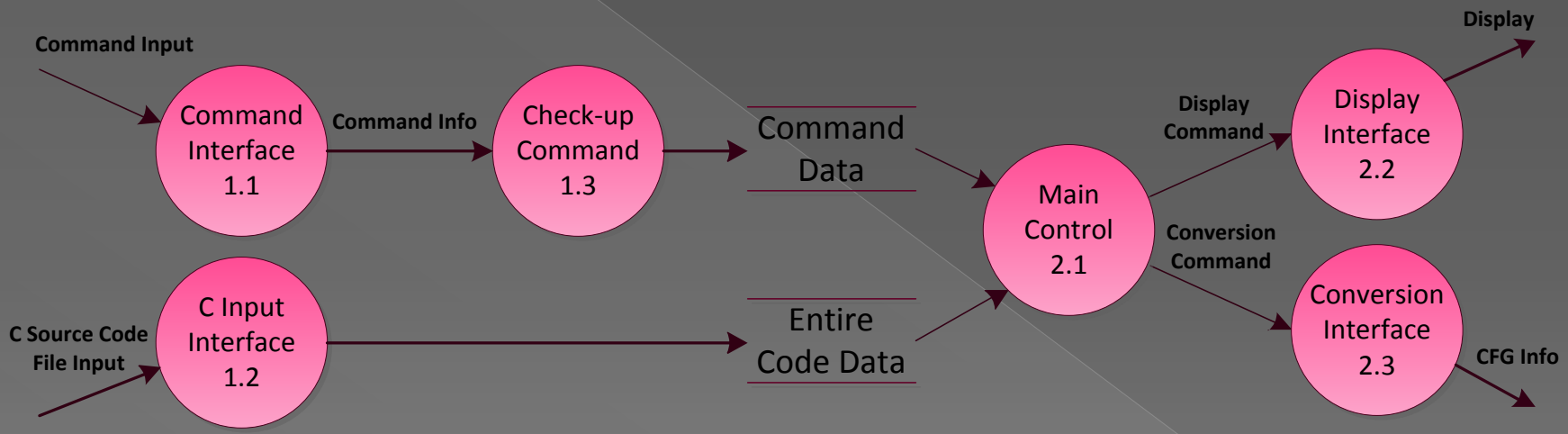
KU KONKUK UNIVERSITY

Data Flow Diagram – level 2



KU KONKUK UNIVERSITY

Data Flow Diagram – level 2



KU KONKUK UNIVERSITY

Data Dictionary – level 2

Data name	Description
Command Info	It is a processed ***Command Input*** for ***Check-up Command*** to check correctness of received command.
Display Command	A data that shows system message or program's progress to be sent to ***Display Interface***.
Conversion Command	A CFG conversion data converted from ***Parsed Data*** to be sent to ***Conversion Interface***.

KU KONKUK UNIVERSITY

Process Specification [1/3]

Reference No.	1.1
Name	Command Interface
Input	Command Input
Output	Command Info
Description	Receiving a ***Command Input*** of the ***User's Command***, converts it to ***Command Info*** that the system can make use of.

Reference No.	1.2
Name	C Input Interface
Input	C Source Code File Input
Output	Numbered Code Data
Description	Receiving a ***C Source Code File Input***, numbers off in order and save it to ***Entire Code Data***.

KU KONKUK UNIVERSITY

Process Specification [2/3]

Reference No.	1.3
Name	Check-up Command
Input	Command Info
Output	Correctness Info
Description	Checking correctness of ***Command Info***, assigns an boolean value and sends ***Correctness Info*** to ***Command Data***.

Reference No.	2.2
Name	Display Interface
Input	Display Commands
Output	Display
Description	Receiving all sorts of ***Display Commands*** from sub processes, sends a ***Display*** converted from them to ***Monitor*** to be printed.

KU KONKUK UNIVERSITY

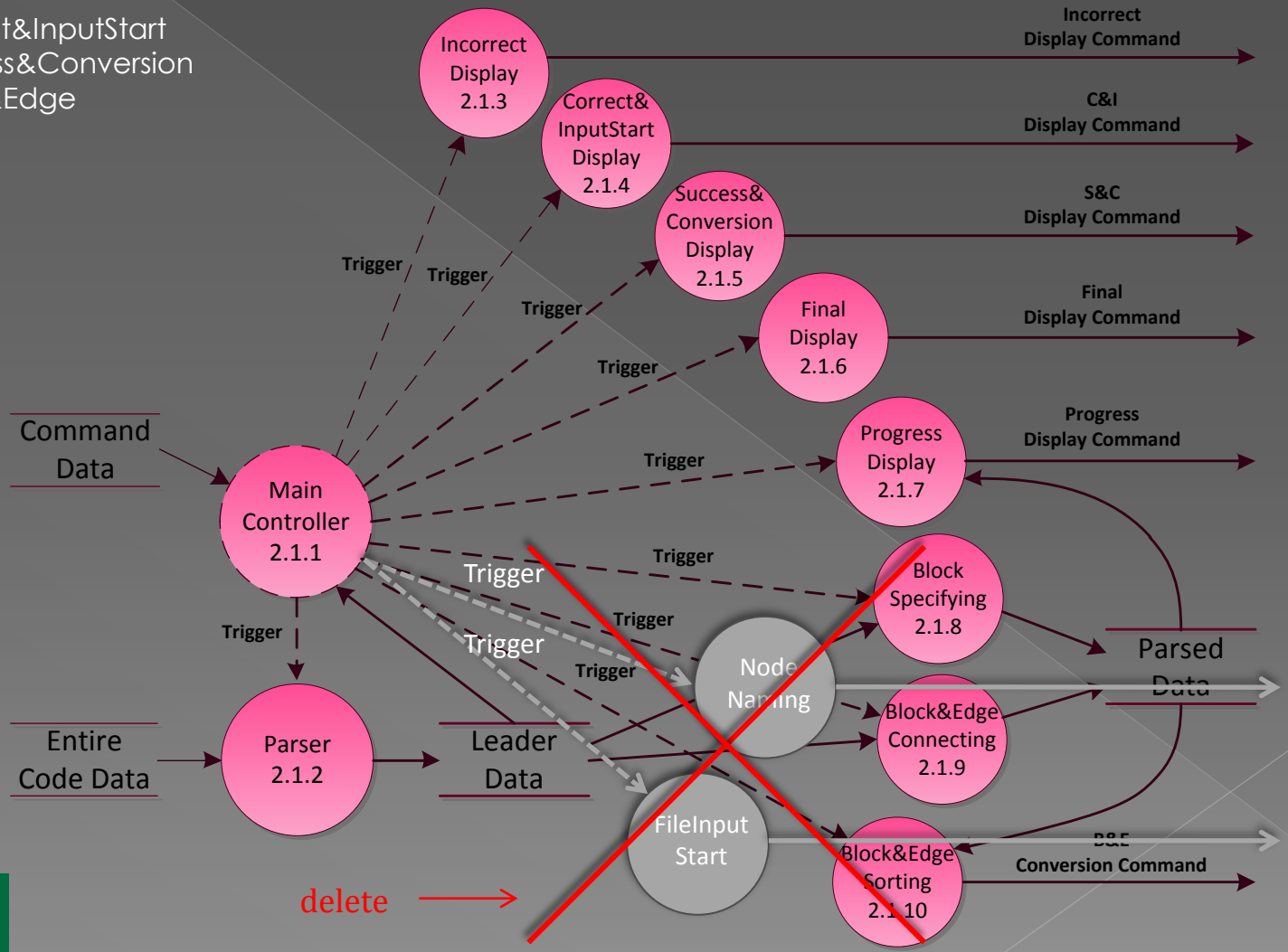
Process Specification [3/3]

Reference No.	2.3
Name	Conversion Interface
Input	Conversion Command
Output	CFG Info
Description	Receiving ***Conversion Command***, converts it to final output ***CFG Info***. It enables CFG Generator to report with a text file by sending ***CFG Info*** to ***CFG Text File***.

KU KONKUK UNIVERSITY

Data Flow Diagram – level 3

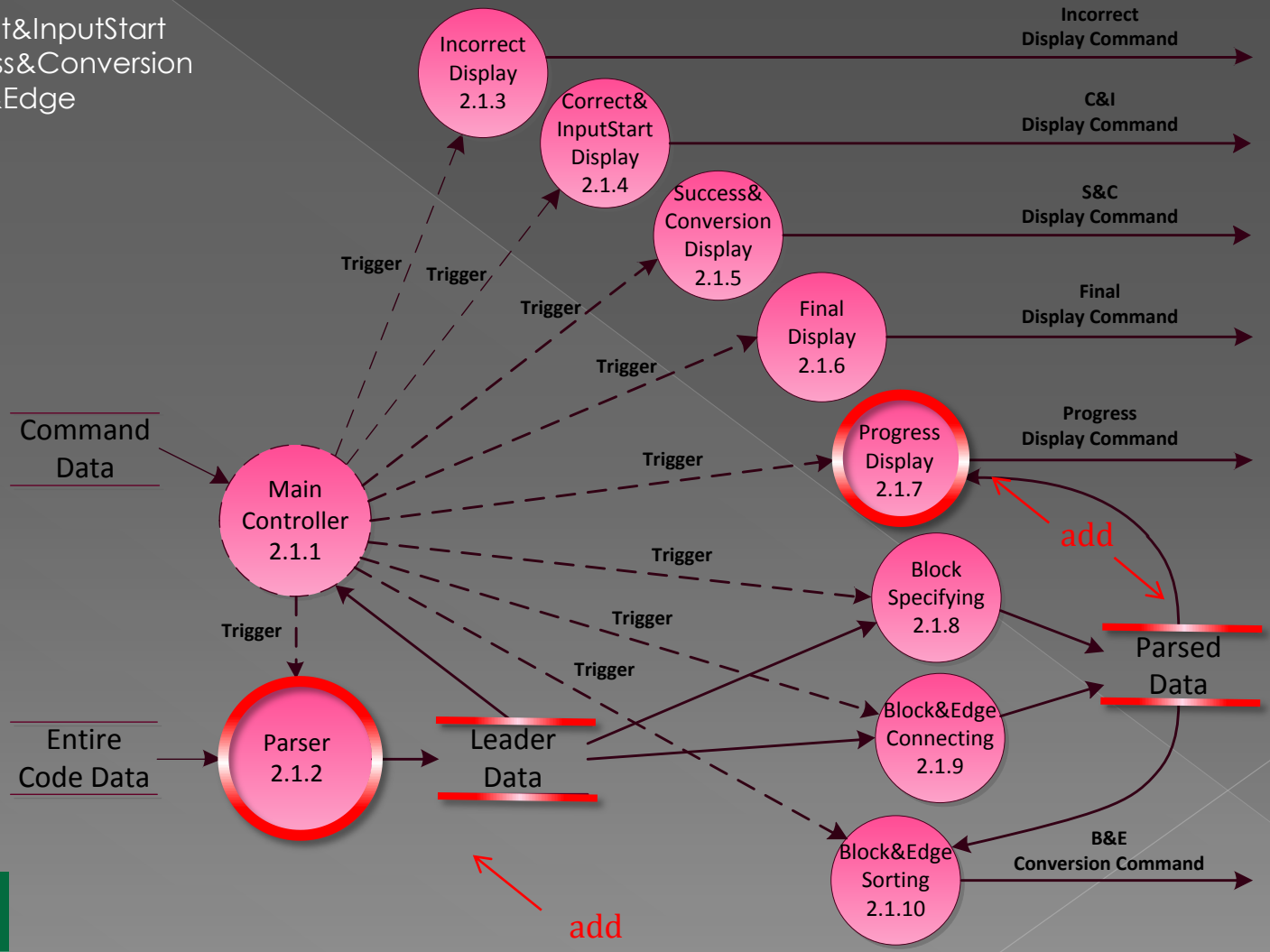
C&I : Correct&InputStart
S&C : Success&Conversion
B&E : Block&Edge



KU KONKUK UNIVERSITY

Data Flow Diagram – level 3

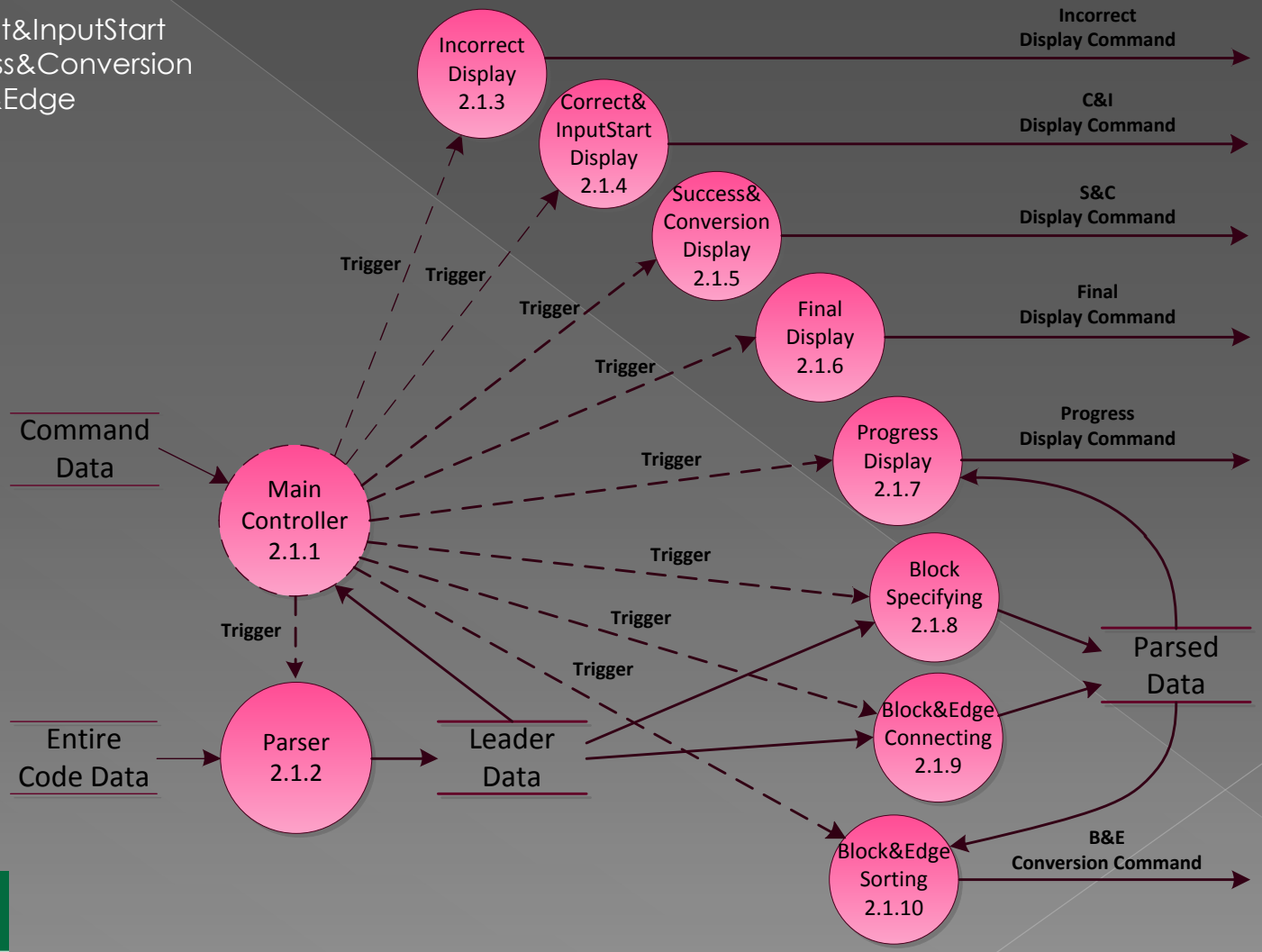
C&I : Correct&InputStart
S&C : Success&Conversion
B&E : Block&Edge



KU KONKUK UNIVERSITY

Data Flow Diagram – level 3

C&I : Correct&InputStart
S&C : Success&Conversion
B&E : Block&Edge



KU KONKUK UNIVERSITY

Data Dictionary – level 3 [1/2]

Leader Data	Description
Raw Block Data	A raw data of code which can be a block (C keyword, assignment statement...).
Raw Edge Group Data	A group data of all raw edge data being decided and conditioned when ***Raw Block Data*** is formed.

Parsed Data	Description
Block Data	It consists of block's name, ID number, range of source code, and major contents.
Edge Group Data	A group data of all parsed edges. Each edge data consists of edge's ID number, condition of control flow, source block's ID number, and destination block's ID number.

KU KONKUK UNIVERSITY

Data Dictionary – level 3 [2/2]

Data name	Description
Incorrect Display Command	A command data that contains error message meaning input of incorrect command and help message.
C&I Display Command	A command data that contains correct message meaning input of correct command and input start message meaning system will start to input of C source code file at once.
S&C Display Command	A command data that contains success message meaning system received input file successfully and conversion message meaning system will start to convert file to CFG at once.
Final Display Command	A command data that contains completion message meaning end of CFG converting and the name of report file.
Progress Display Command	A command data that contains execution order of C source code and progress.
B&E Conversion Command	A command data being sent to ***Conversion Interface*** to convert ***Parsed Data*** to final CFG.

Process Specification [1/5]

Reference No.	2.1.1
Name	Main Controller
Input	Leader Data, Correctness Info
Output	Trigger
Description	It triggers ***Parser*** and other processes which output ***Display Command*** or ***Conversion Command*** based on inputs (***Leader Data***, ***Correctness Info***).

Reference No.	2.1.2
Name	Parser
Input	Numbered Code Data, Trigger
Output	Leader Data
Description	Converting ***Numbered Code Data*** into ***Raw Block Data*** and ***Raw Edge Group Data***, stores them in the ***Leader Data***.

Process Specification [2/5]

C&I : Correct&InputStart

Reference No.	2.1.3
Name	Incorrect Display
Input	Trigger
Output	Incorrect Display Command
Description	When a user input incorrect command, sends ***Incorrect Display Command*** which consists of error message and help message to ***Display Interface***. Afterward, terminates program.

Reference No.	2.1.4
Name	Correct&InputStart Display
Input	Trigger
Output	C&I Display Command
Description	When a user input correct command, sends ***C&I Display Command*** which consists of correct message and input start message to ***Display Interface***.

Process Specification [3/5]

S&C : Success&Conversion

Reference No.	2.1.5
Name	Success&Conversion Display
Input	Trigger
Output	S&C Display Command
Description	When system receives input file successfully and ***Numbered Code Data*** is created, it sends ***S&C Display Command*** which consists of success message and conversion message to ***Display Interface***.

Reference No.	2.1.6
Name	Final Display
Input	Trigger
Output	Final Display Command
Description	After finishing parsing and sorting, sends ***Final Display Command*** which consists of completion message and the name of report file to ***Display Interface***. Afterward, terminates program.

Process Specification [4/5]

Reference No.	2.1.7
Name	Progress Display
Input	Parsed Data, Trigger
Output	Progress Display Command
Description	Receiving ***Parsed Data***, it sends ***Progress Display Command*** which shows execution order of C source code and progress to ***Display Interface***.

Reference No.	2.1.8
Name	Block Specifying
Input	Leader Data, Trigger
Output	Block Data
Description	Receiving incomplete ***Leader Data***, outputs ***Block Data*** specifying each block's concrete information.

Process Specification [5/5]

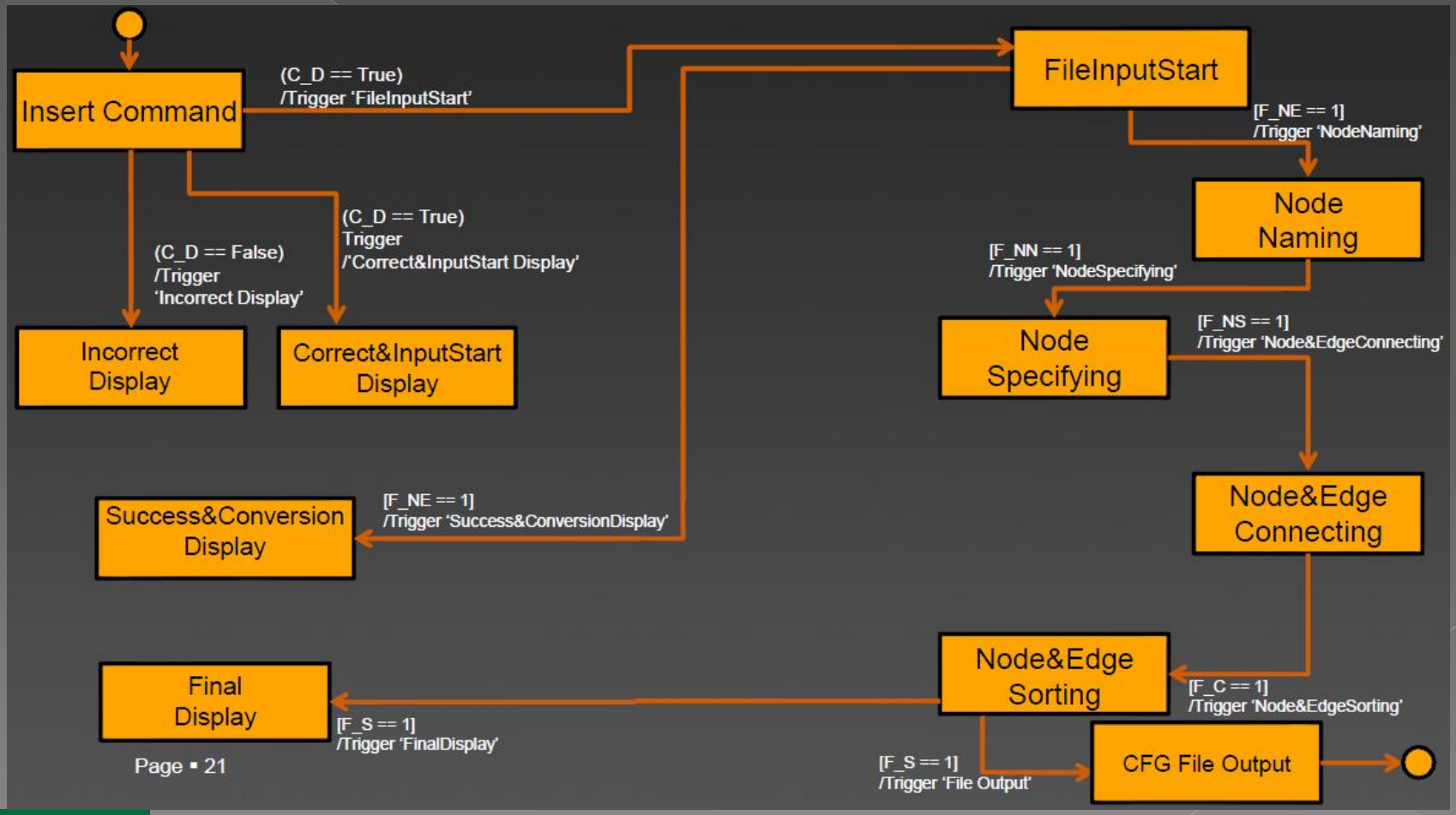
B&E : Block&Edge

Reference No.	2.1.9
Name	Block&Edge Connection
Input	Leader Data, Trigger
Output	Edge Group Data
Description	Receiving incomplete ***Leader Data***, outputs ***Edge Group Data*** whose each edge data corresponds with involved blocks.

Reference No.	2.1.10
Name	Block&Edge Sorting
Input	Parsed Data, Trigger
Output	B&E Conversion Command
Description	Receiving ***Parsed Data***, outputs ***B&E Conversion Command*** to be converted into final CFG in sorted text form according to each block's level.

State Machine for Main Controller 2.1.1

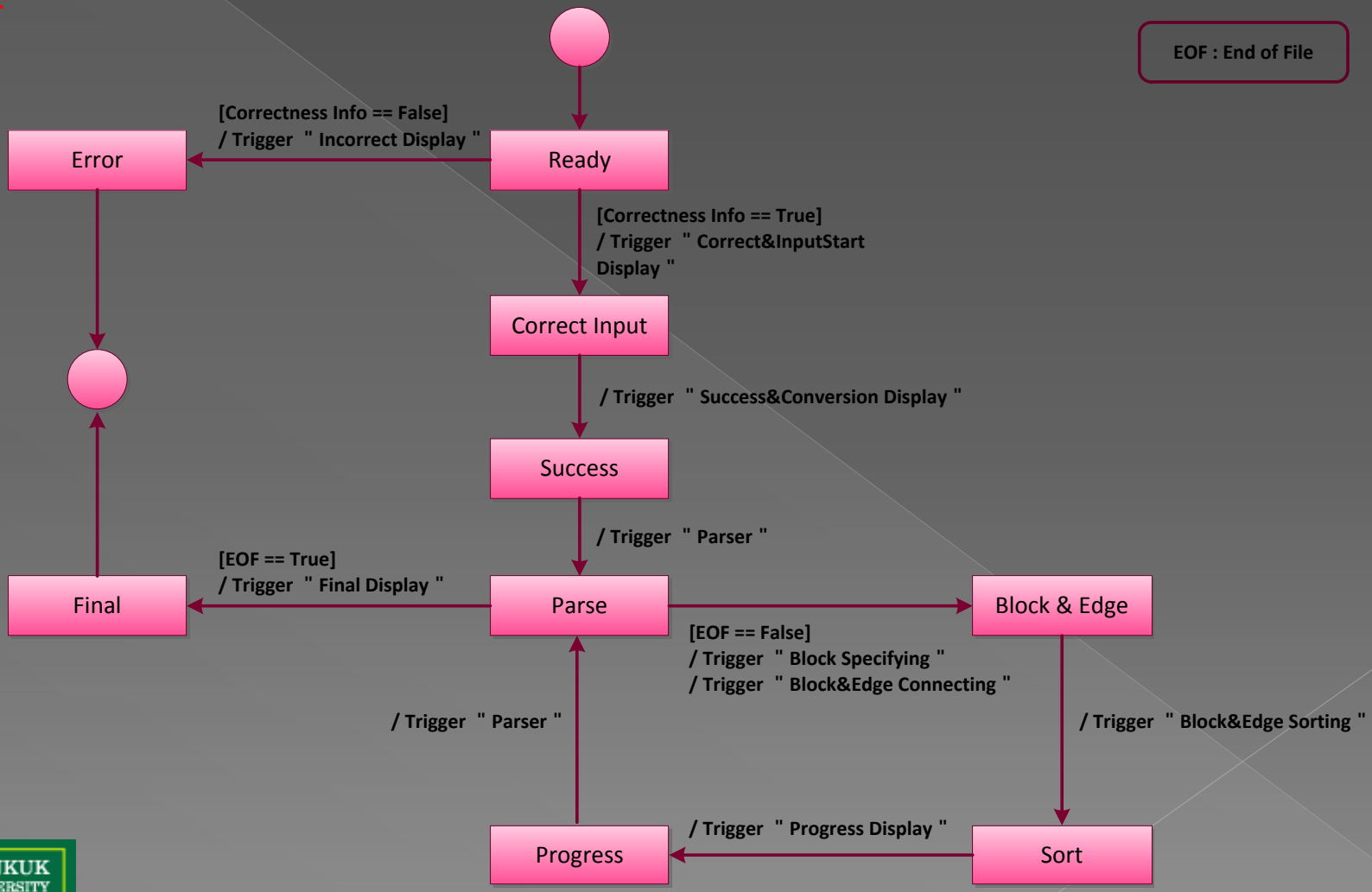
before



KU KONKUK UNIVERSITY

State Machine for Main Controller 2.1.1

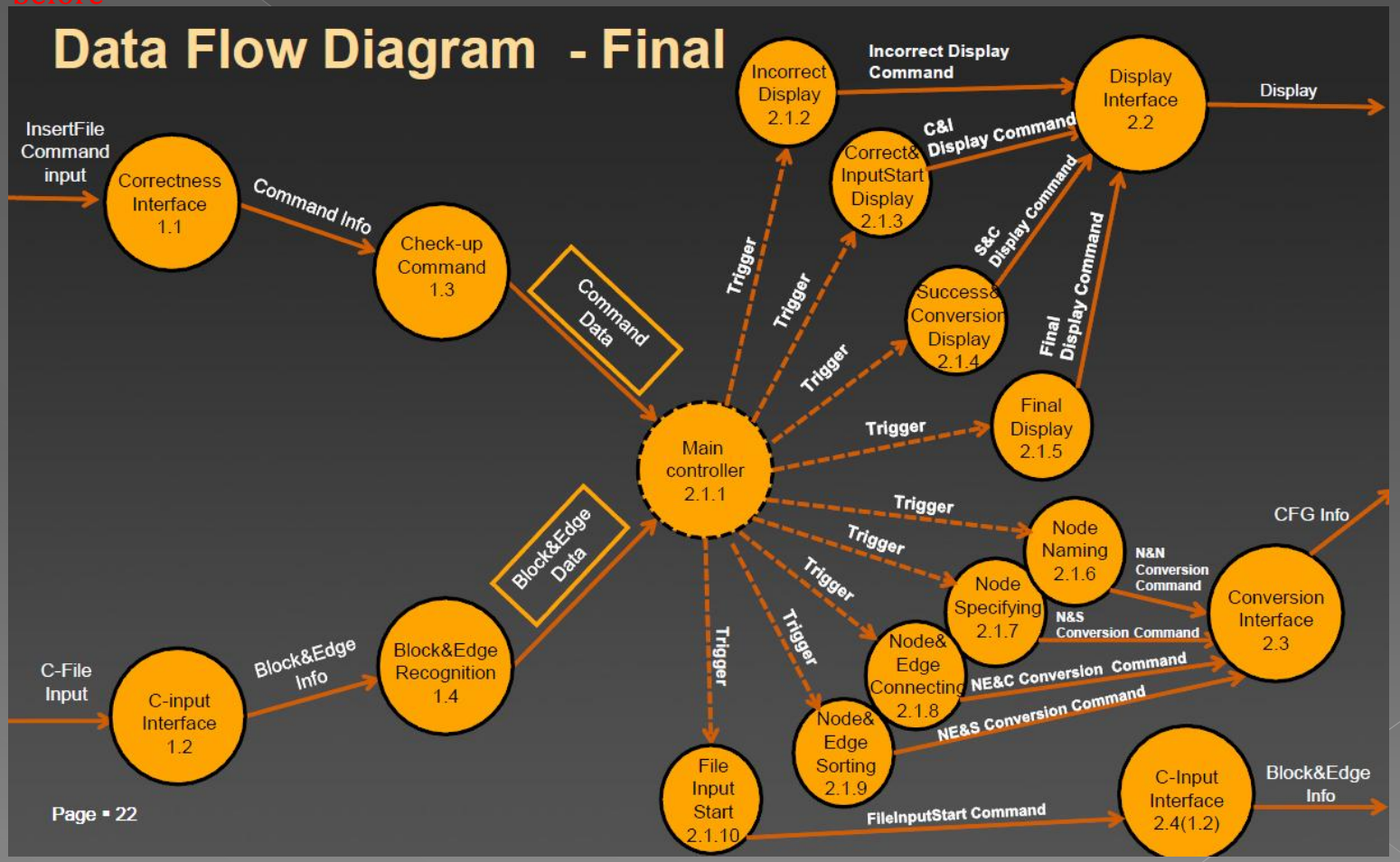
after



KU KONKUK UNIVERSITY

Data Flow Diagram - Overall

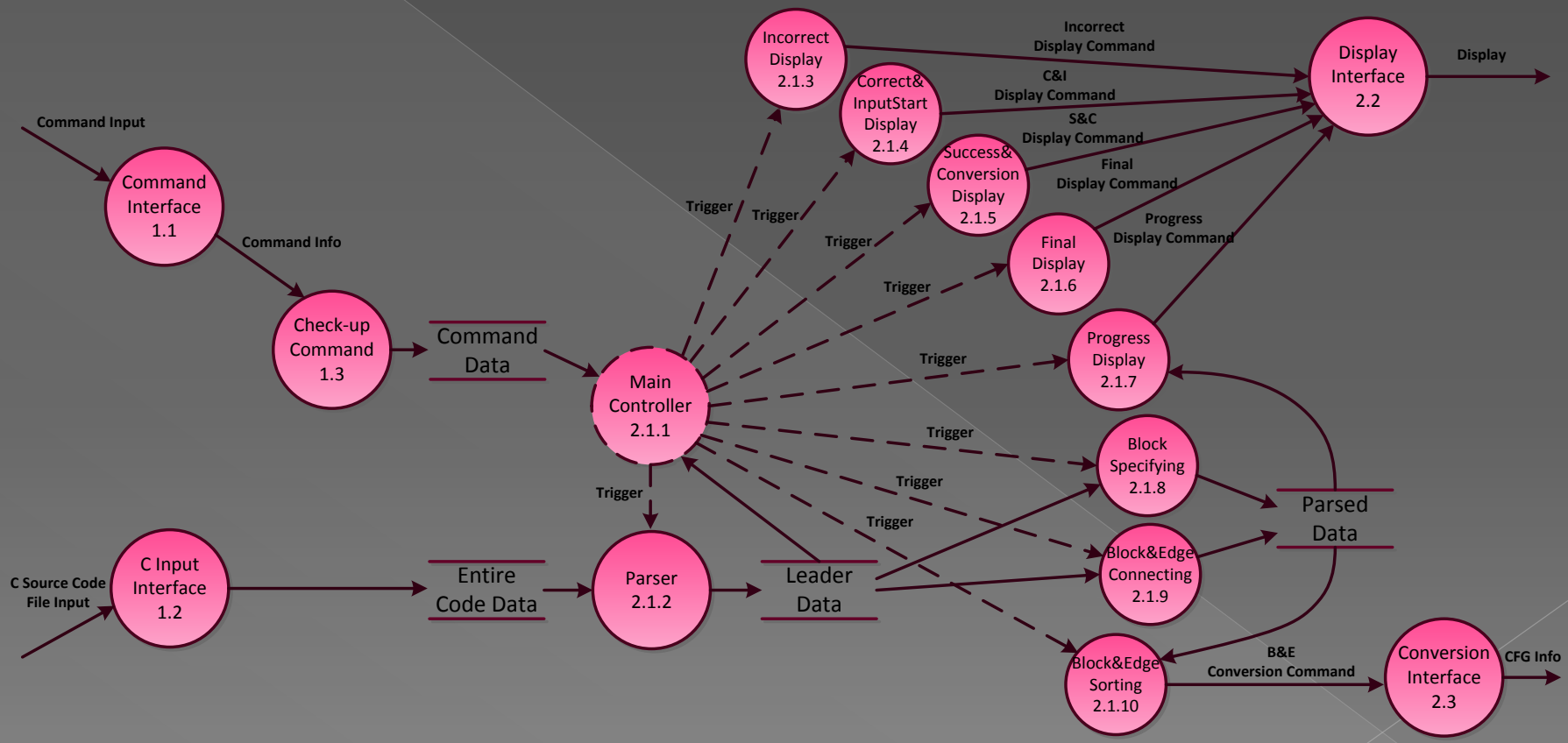
before



KU KONKUK UNIVERSITY

Data Flow Diagram - Overall

after



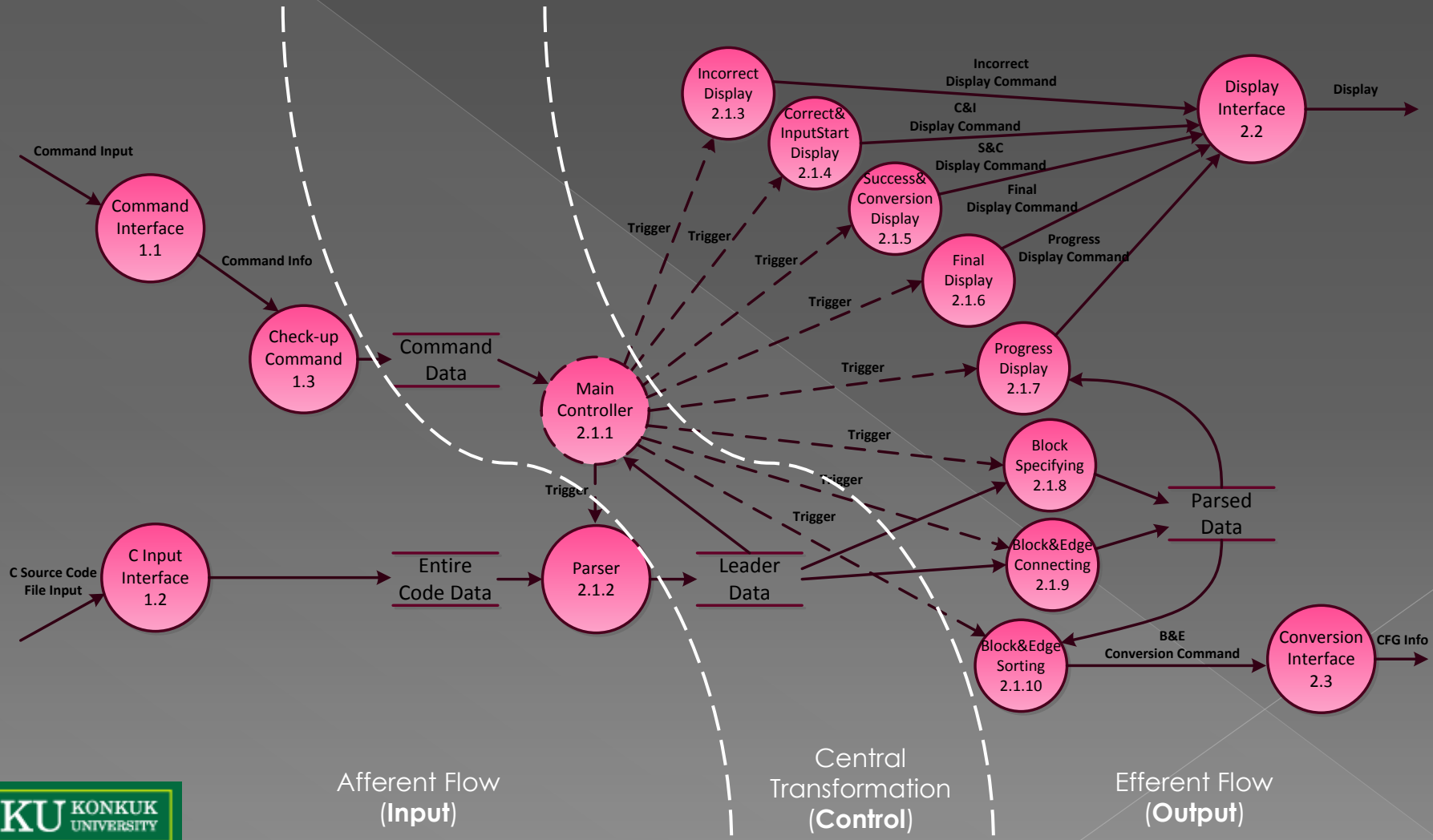
C&I : Correct&InputStart
S&C : Success&Conversion
B&E : Block&Edge

KU KONKUK UNIVERSITY

Structured Design

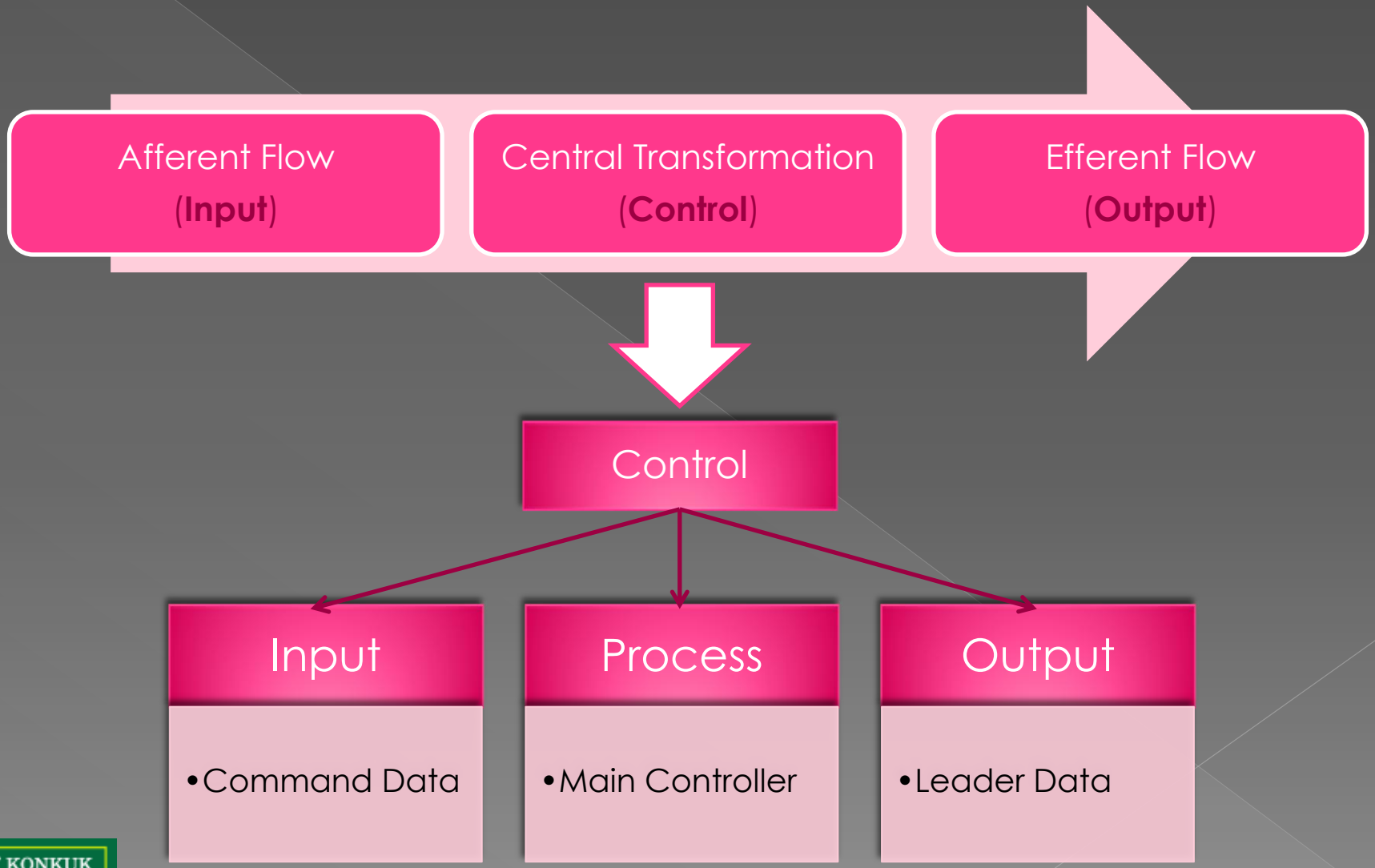
KU KONKUK UNIVERSITY

Structured Chart – Transform Analysis [1/2]



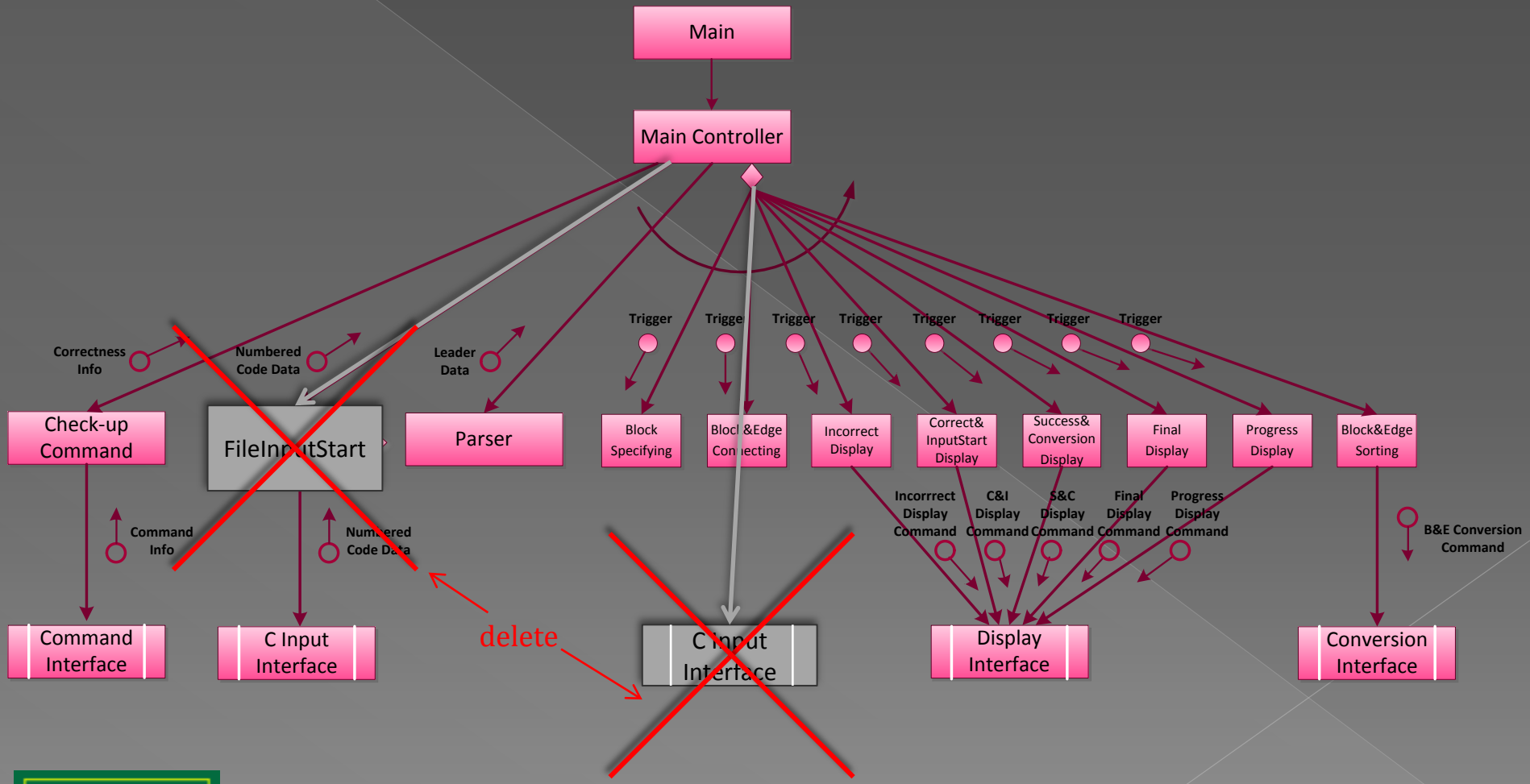
KU KONKUK UNIVERSITY

Structured Chart – Transform Analysis [2/2]



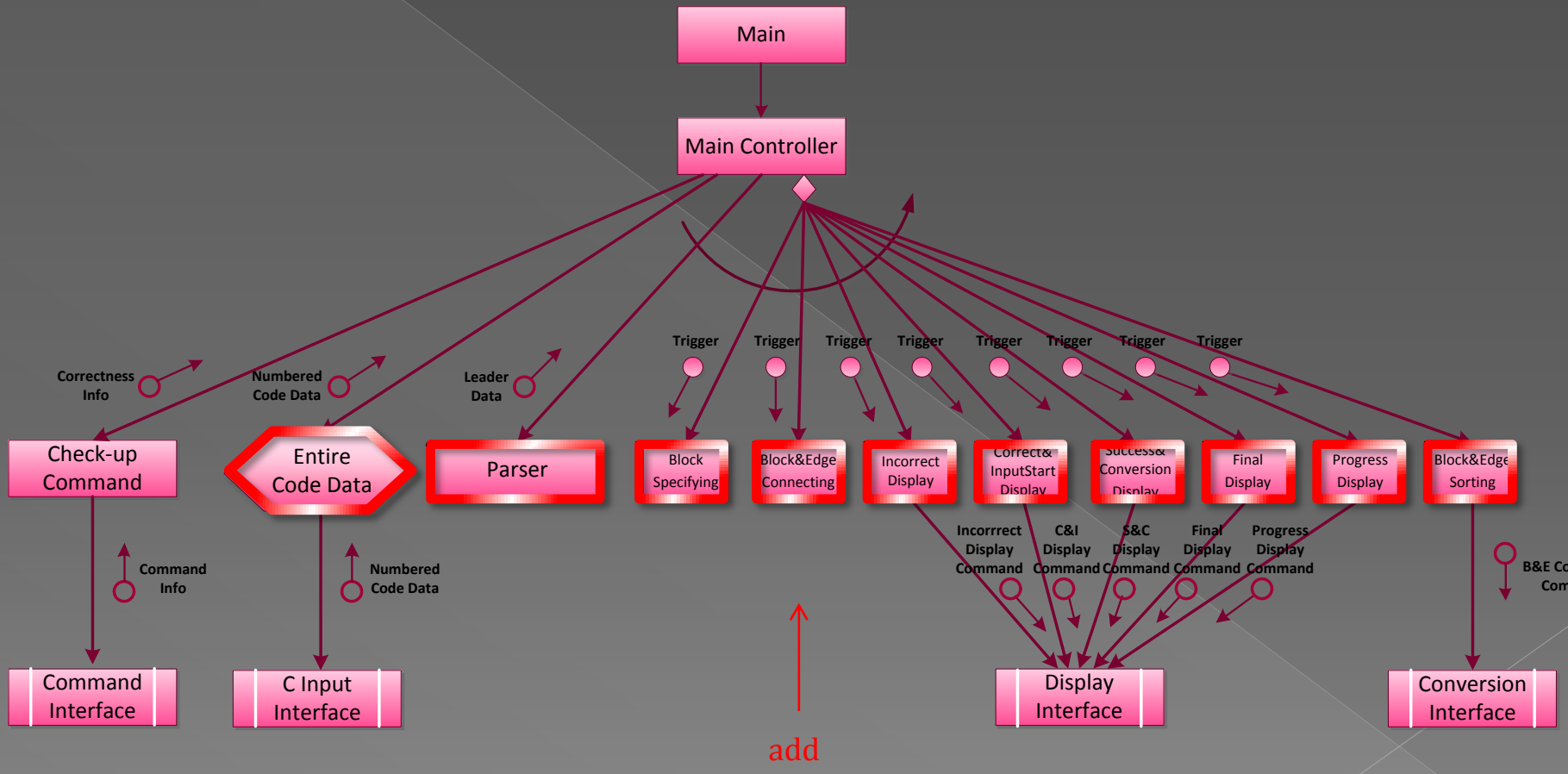
KU KONKUK UNIVERSITY

Structured Chart – CFG Generator



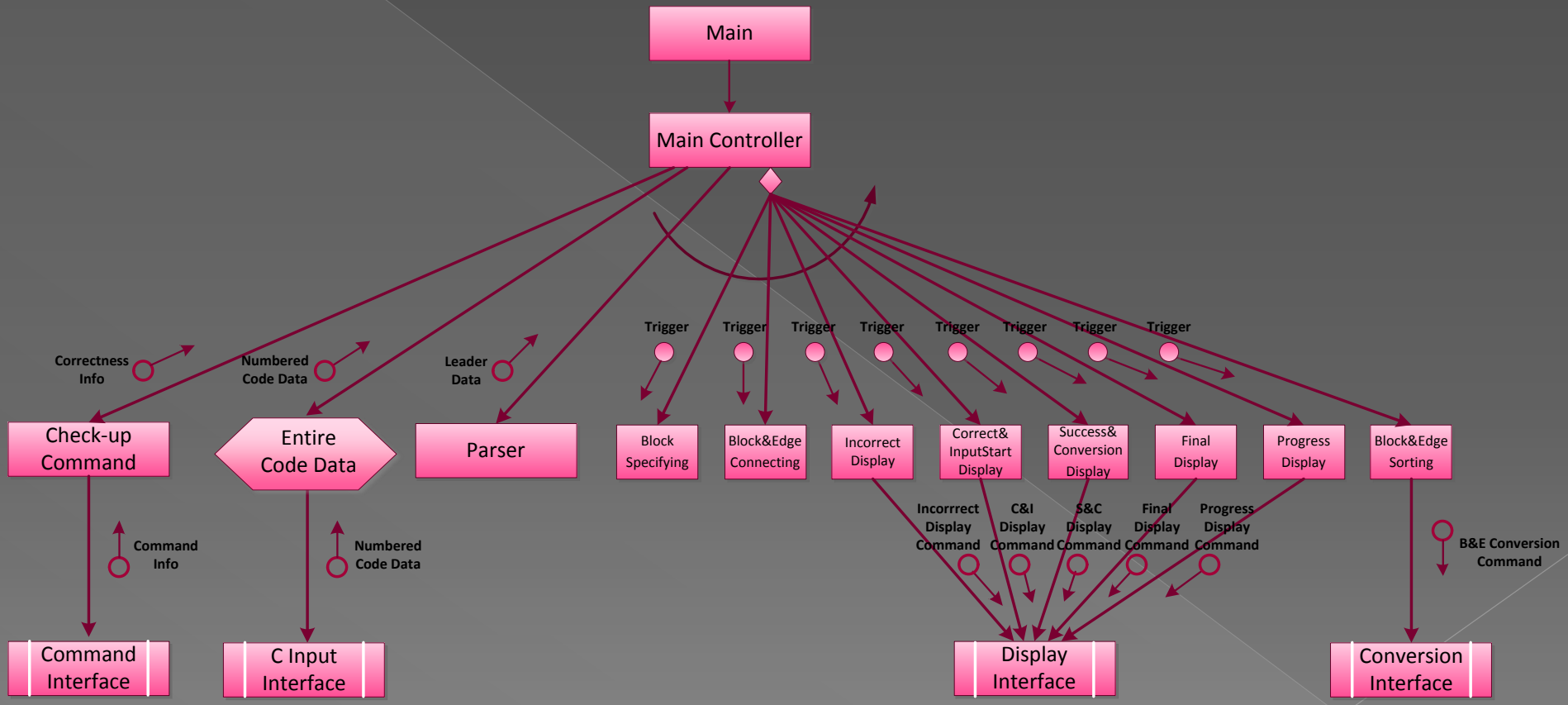
KU KONKUK UNIVERSITY

Structured Chart – CFG Generator



KU KONKUK UNIVERSITY

Structured Chart – CFG Generator



KU KONKUK UNIVERSITY

Structured Chart – Data Definition [1/2]

Data name	Description
Command Info	It is a processed ***Command Input*** for ***Check-up Command*** to check correctness of received command.
Correctness Info	A boolean value meaning correctness of received command. (True/False)
Numbered Code Data	It consists of ***C Source Code File Input*** and each corresponding line number.
Leader Data	It is a structure data { Raw Block Data, Raw Edge Group Data }
Incorrect Display Command	A command data that contains error message meaning input of incorrect command and help message.

KU KONKUK UNIVERSITY

Structured Chart – Data Definition [2/2]

Data name	Description
C&I Display Command	A command data that contains correct message meaning input of correct command and input start message meaning system will start to input of C source code file at once.
S&C Display Command	A command data that contains success message meaning system received input file successfully and conversion message meaning system will start to convert file to CFG at once.
Final Display Command	A command data that contains completion message meaning end of CFG converting and the name of report file.
Progress Display Command	A command data that contains execution order of C source code and progress.
B&E Conversion Command	A command data being sent to ***Conversion Interface*** to convert ***Parsed Data*** to final CFG.

Source Code

KU KONKUK UNIVERSITY

Development Environment

- Windows 7 Home edition K
- cygwin 1.7.9
- visual studio 2010 professional

KU KONKUK UNIVERSITY

Project Structure

- **Header**
  - data.h
  - input.h
  - output.h
  - _mingw.h / conio.h / tc.h / windows.h
- **Source**
  - input.c
  - output.c
  - main.c
  - tc.c

KU KONKUK UNIVERSITY

Proje

- Dat

```
11 enum
l_else_
l_break
12 enum
13 enum
15 // C
16 enum
PROGRES
```

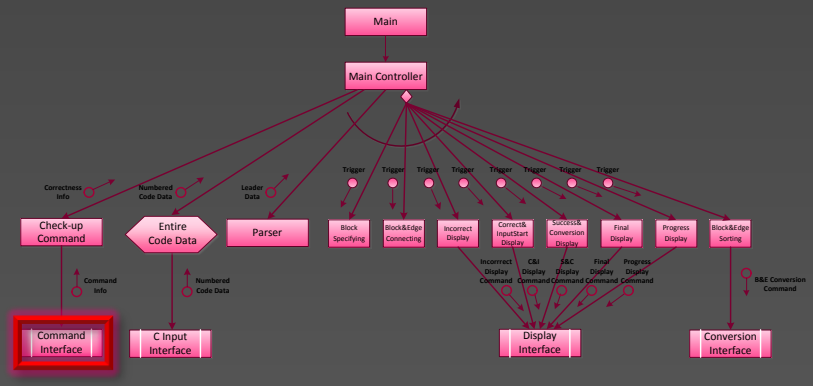
```
lo_while, l_if,
l_root,

t_b };

&Conversion.
AND_C, FINAL,
```

- st
- st
- st
- st
- st
- st

```
18 struct command_info
19 {
20 int command_count;
21 char** command_input;
22 };
24 struct entire_code_data
25 {
26 int line;
27 char** numbered_code_data;
28 };
30 struct leader_data {
31 enum leader keyword;
32 int block_start, block_end;
34 char* edge_condition; };
36 struct leader_data* parent;
37 int child_num;
39 struct leader_data** children;
40 int children_cap;
41 int children_size;
42 };
44 struct block_data {
45 enum block_type type;
46 int id;
47 int block_start, block_end;
48 enum leader keyword;
49 char* description;
50 };
52 struct edge_data {
53 enum edge_type type;
54 int id;
55 int source_block, destination_block;
56 char* description;
57 };
59 struct display_command {
60 enum display_command_type type;
61 char* message1;
62 char* message2;
64 char* block;
65 char** edge;
67 int edge_size;
68 int* edge_start;
69 int* edge_end;
71 int block_id;
72 };
```

KU KONKUK UNIVERSITY

Major Procedure

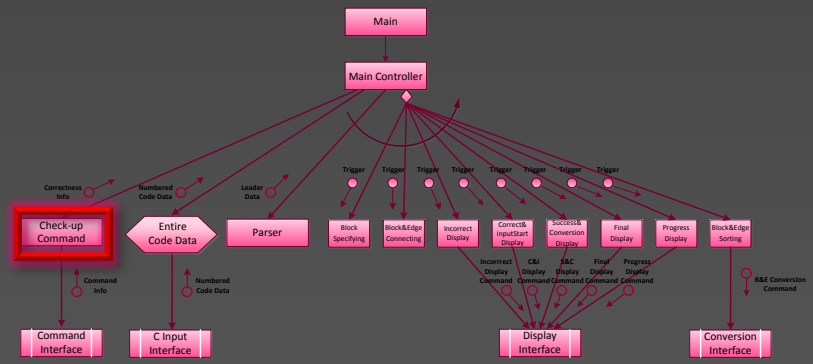


Reference No.	1.1
Name	Command Interface
Input	Command Input
Output	Command Info
Description	Receiving a ***Command Input*** of the ***User's Command***, converts it to ***Command Info*** that the system can make use of.

```
5 void command_interface(int argc, char **argv, struct command_info* c_info)
6 {
7 c_info->command_count = argc;
// 명령어 인자 개수 설정
8 c_info->command_input = argv;
// 명령어 인자 설정
9 }
```

KU KONKUK UNIVERSITY

Major Procedure

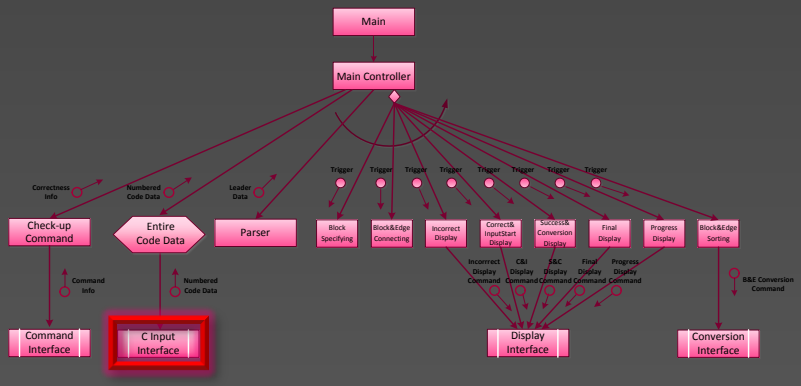


Reference No.	1.3
Name	Check-up Command
Input	Command Info
Output	Correctness Info
Description	Checking correctness of ***Command Info***, assigns an boolean value and sends ***Correctness Info*** to ***Command Data***.

```
86 if(c_info->command_count != 3) // 명령어 옵션 개
수 오류
87 return 0;
89 temp = strstr(c_info->command_input[1],".c");
// 소스파일 확장자 검사
91 if(NULL == temp) // 소스파일 확장자 오류
92 {
93 return 0;
94 }
95 else if(c_info->command_input[1] +
strlen(c_info->command_input[1]) - 2 != temp)
96 {
97 return 0; // ".c"는 찾았지만 맨 끝에 있지 않을경우
ex) asdf.cAA
98 }
100 temp = strstr(c_info-
>command_input[2],".txt"); // 출력파일 확장자 검사
102 if(NULL == temp) // 출력파일 확장자 오류
103 {
104 return 0;
105 }
106 else if(c_info->command_input[2] +
strlen(c_info->command_input[2]) -
strlen(".txt") != temp)
107 {
108 return 0; // ".txt"는 찾았지만 맨 끝에 있지 않을경
우 ex) asdf.txtAA
109 }
```

KU KONKUK UNIVERSITY

Major Procedure

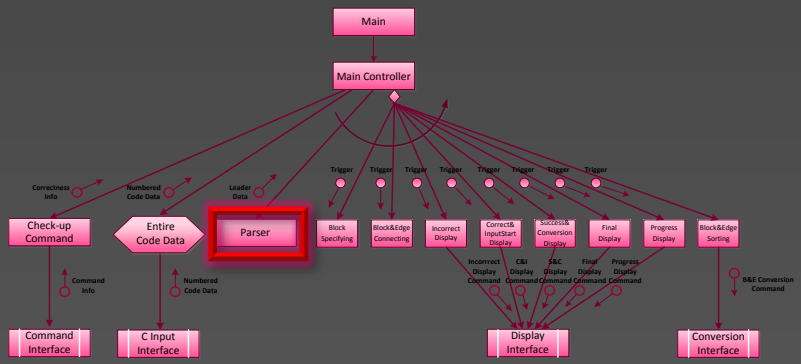


Reference No.	1.2
Name	C Input Interface
Input	C Source Code File Input
Output	Numbered Code Data
Description	Receiving a ***C Source Code File Input***, numbers off in order and save it to ***Entire Code Data***.

```
29 while( (ch=fgetc(file)) != EOF ) // 라인 수 카운트
30 {
31 if(ch == '\n') 32 entire_code->line++;
33 }
44 while( (ch=fgetc(file)) != EOF ) // 라인 당 문자 수 카운트
45 {
46 if(ch == '\n') 48 i++;
50 else 52 line_characters[i]++;
54 }
59 for(i=0;i<entire_code->line;i++) // numbered_code_data 생성
60 {
61 entire_code->numbered_code_data[i] = (char*)malloc(sizeof(char)*(line_characters[i]+1));
63 fgets(entire_code->numbered_code_data[i],line_characters[i]+2,file);
66 entire_code->numbered_code_data[i][(int)strlen(entire_code->numbered_code_data[i])] = '\0';
67 }
70 entire_code->numbered_code_data[entire_code->line] = (char*)malloc(sizeof(char)*4)
71 entire_code->numbered_code_data[entire_code->line] = "EOF";
```

KU KONKUK UNIVERSITY

Major Procedure

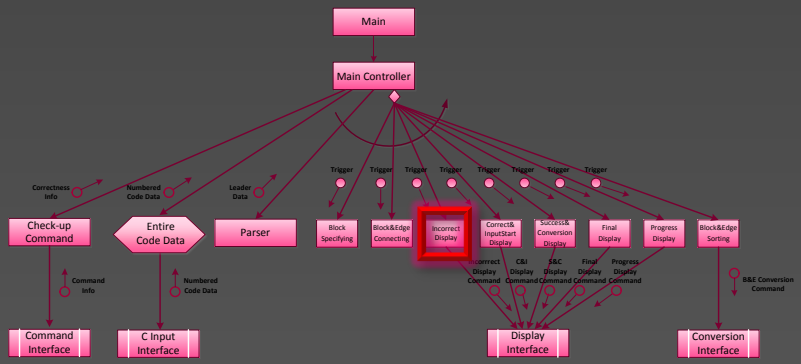


Reference No.	2.1.2
Name	Parser
Input	Numbered Code Data, Trigger
Output	Leader Data
Description	Converting ***Numbered Code Data*** into ***Raw Block Data*** and ***Raw Edge Group Data***, stores them in the ***Leader Data***.

```
129 static int cur_line = 0; //어디까지 읽었는지를 저장하는 코드
130 static struct leader_data* scope = NULL;
131 static char* block_in_condition = NULL;
//case문에 대한 조건을 저장하는 변수 - switch문에서만 쓰인다. 132 int type;
133 struct leader_data* rbd = NULL;
134 do
135 {
136 type = check_line(codeinfo[cur_line++]);
137 }
138 while(type == 0); //블록이 생성될때까지 체크한다. - 주석이나 중괄호를 무시한다.
141 //타입에 따른 구조체 동적할당 및 초기화와 트리구조를 구현한다.
142 switch(type)
143 {
144 case 1:
...
...중략
...
326 case -1: //파일의 끝일경우이다.
327 rbd = NULL;
328 break;
329 }
```

KU KONKUK UNIVERSITY

Major Procedure

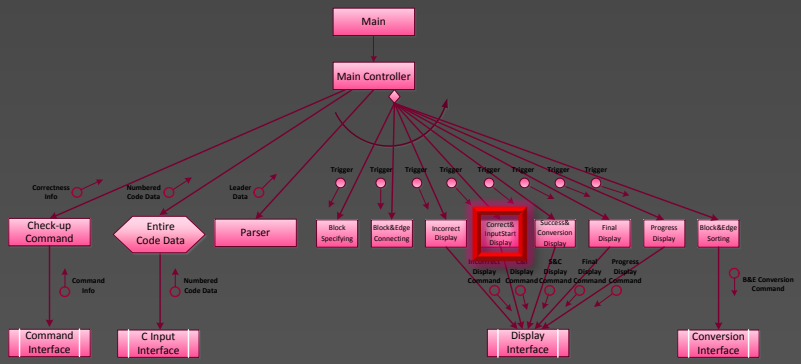


Reference No.	2.1.3
Name	Incorrect Display
Input	Trigger
Output	Incorrect Display Command
Description	When a user input incorrect command, sends ***Incorrect Display Command*** which consists of error message and help message to ***Display Interface***. Afterward, terminates program.

```
8 void
incorrect_display(struct
display_command* d_command)
9 {
10 d_command->type =
INCORRECT;
11 d_command->message1 =
"\n>> error : incorrect
command";
12 d_command->message2 =
"\n>> command syntax :
$ ./program Inputcode.c
Result.txt";
13 }
```

KU KONKUK UNIVERSITY

Major Procedure

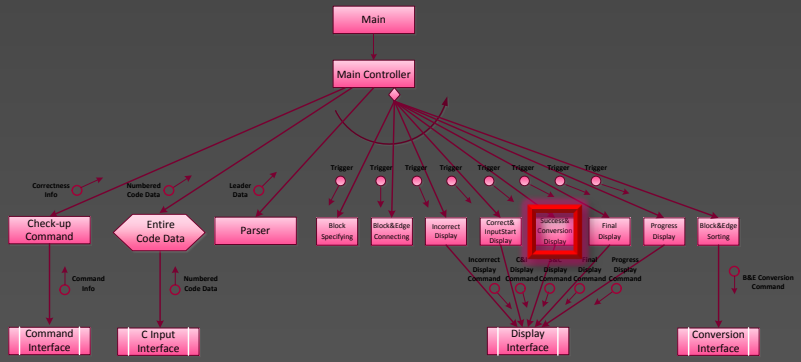


Reference No.	2.1.4
Name	Correct&InputStart Display
Input	Trigger
Output	C&I Display Command
Description	When a user input correct command, sends ***C&I Display Command*** which consists of correct message and input start message to ***Display Interface***.

```
18 void
correct_and_inputstart_disp
lay(struct display_command*
d_command)
19 {
20 d_command->type =
C_AND_I;
21 d_command->message1 =
"\n>> system : received
correct command";
22 d_command->message2 =
"\n>> system : start to
input";
23 }
```

KU KONKUK UNIVERSITY

Major Procedure

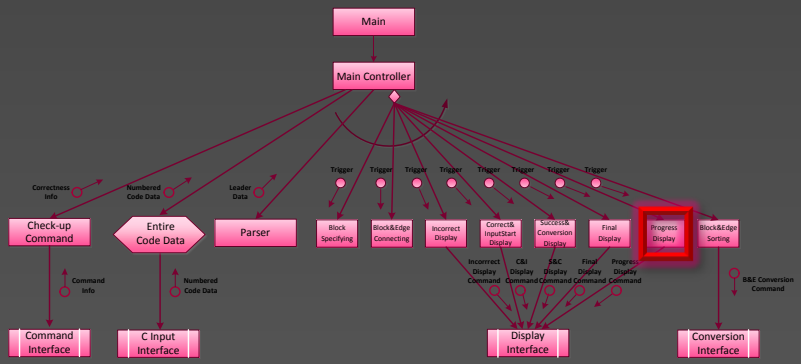


Reference No.	2.1.5
Name	Success&Conversion Display
Input	Trigger
Output	S&C Display Command
Description	When system receives input file successfully and ***Numbered Code Data*** is created, it sends ***S&C Display Command*** which consists of success message and conversion message to ***Display Interface***.

```
28 void success_and_conversion_display(struct display_command* d_command)
29 {
30 d_command->type = S_AND_C;
31 d_command->message1 = "\n>> system : received input file successfully";
32 d_command->message2 = "\n>> system : start to convert";
33 }
```

KU KONKUK UNIVERSITY

Major Procedure

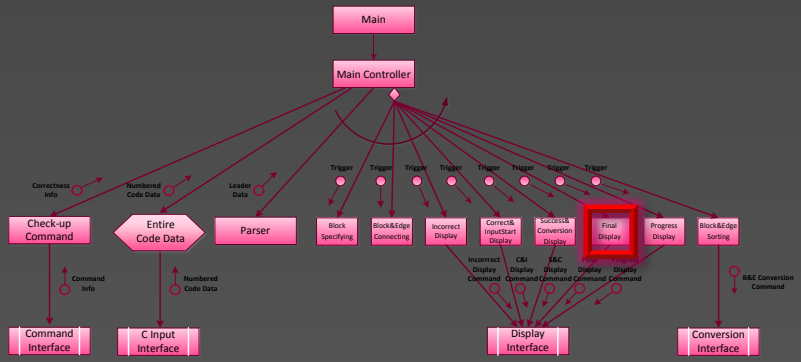


Reference No.	2.1.7
Name	Progress Display
Input	Parsed Data, Trigger
Output	Progress Display Command
Description	Receiving ***Parsed Data***, it sends ***Progress Display Command*** which shows execution order of C source code and progress to ***Display Interface***.

```
48 void progress_display(struct
display_command* d_command)
49 {
50 int i,j;
51 static int pre_size =0;
52 d_command->type = PROGRESS;
53 d_command->block = block_list[blist_size-1]-
>description;
54 d_command->block_id = block_list[blist_size-
1]->id;
56 for(i=pre_size,j=0; i<elist_size; i++,j++)
57 {
58 d_command->edge_start[j] = edge_list[i]-
>source_block;
59 d_command->edge_end[j] = edge_list[i]-
>destination_block;
60 d_command->edge[j] = edge_list[i]-
>description;
61 }
62 d_command->edge_size = j;
63 pre_size = elist_size;
64 }
```

KU KONKUK UNIVERSITY

Major Procedure

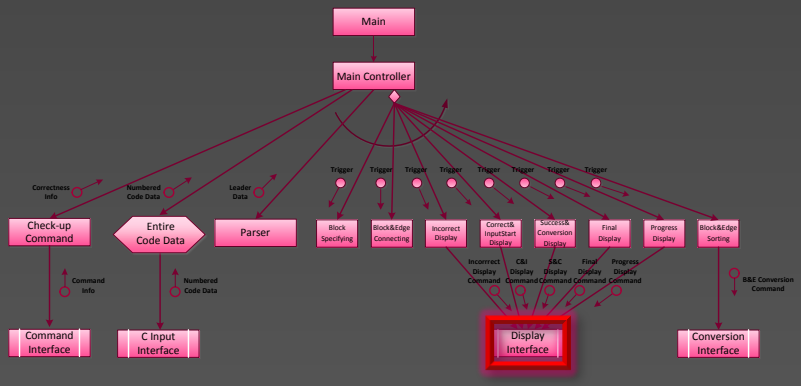


Reference No.	2.1.6
Name	Final Display
Input	Trigger
Output	Final Display Command
Description	After finishing parsing and sorting, sends ***Final Display Command*** which consists of completion message and the name of report file to ***Display Interface***. Afterward, terminates program.

```
38 void
final_display(struct
display_command* d_command,
char* filename)
39 {
40 d_command->type = FINAL;
41 d_command->message1 =
"\n>> system : complete
conversion. report filename
is";
42 d_command->message2 =
filename;
43 }
```

KU KONKUK UNIVERSITY

Major Procedure

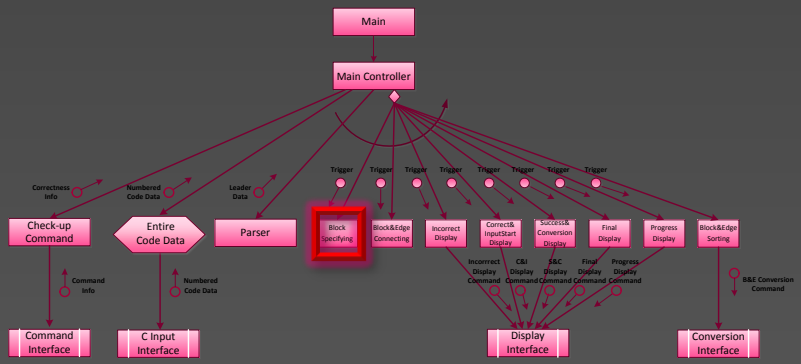


Reference No.	2.2
Name	Display Interface
Input	Display Commands
Output	Display
Description	Receiving all sorts of ***Display Commands*** from sub processes, sends a ***Display*** converted from them to ***Monitor*** to be printed.

```
72 if(d_command->type != PROGRESS) // 나머지 Display Command
처리
73 {
74 setcolor(YELLOW); 75 printf("%s\n", d_command->message1);
76 printf("%s\n", d_command->message2); 77 setcolor(WHITE);
78 printf("\n");
79 }
80 else // Progress Display Command 처리
81 {
82 for(i=0; i<d_command->edge_size; i++)
83 {
85 if(d_command->edge_start[i] < d_command->edge_end[i])
86 {
87 setcolor(LIGHTMAGENTA); printf("+[edge] ");
setcolor(WHITE); printf("source : ");
88 setcolor(LIGHTCYAN); printf("%d", d_command-
>edge_start[i]+1); setcolor(WHITE); printf(", destination :
");
89 setcolor(LIGHTRED); printf("%d", d_command-
>edge_end[i]+1); setcolor(WHITE);
90 }
91 else
… 중략
98 printf(", description : %s \n\n", d_command->edge[i]);
99 }
101 setcolor(LIGHTGREEN); printf("+[block] ");
setcolor(WHITE);
102 printf("id : %d, description : %s \n",d_command-
>block_id+1, d_command->block);
```

KU KONKUK UNIVERSITY

Major Procedure

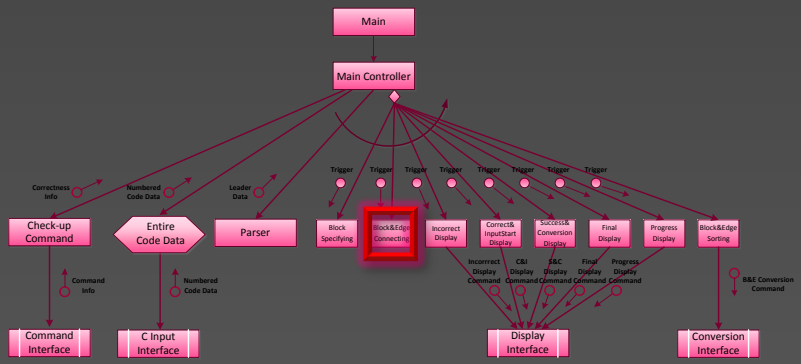


Reference No.	2.1.8
Name	Block Specifying
Input	Leader Data, Trigger
Output	Block Data
Description	Receiving incomplete ***Leader Data***, outputs ***Block Data*** specifying each block's concrete information.

```
112 block->block_start = input-
>block_start;
113 block->block_end = input->block_end;
114 block->keyword = input->keyword;
115 block->id = bid++;
116 block->description = codeinfo[input-
>block_start];
118 switch(block->keyword)
119 {
120 case l_function:
121 block->type = entry_b;
122 break;
123 case l_return:
124 block->type = exit_b;
125 break;
126 case l_while:
127 case l_for:
128 case l_do_while:
129 block->type = loop_header;
130 break;
131 default:
132 block->type = basic;
133 break;
134 }
```

KU KONKUK UNIVERSITY

Major Procedure

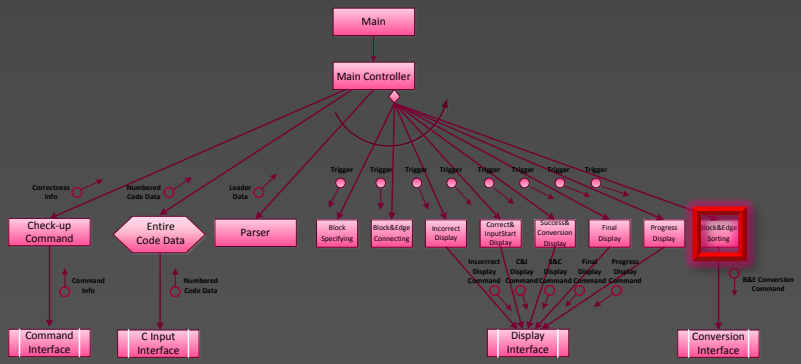


Reference No.	2.1.9
Name	Block&Edge Connection
Input	Leader Data, Trigger
Output	Edge Group Data
Description	Receiving incomplete ***Leader Data***, outputs ***Edge Group Data*** whose each edge data corresponds with involved blocks.

```
143 struct leader_data* parent =
input->parent;
144 struct leader_data*
brother=NULL;
147 //부모와연결
148 if(input->parent->keyword !=
l_root)
149 {
150
link_parent2(parent,block,input-
>child_num,input->edge_condition);
151 }
152 if(input->child_num != 0)
153 {
154 brother = input->parent-
>children[input->child_num-1];
155 //형제와연결
156 link_brother(brother,block);
158 //형제가 반복문일 경우 백엣지 연결
159 add_back(brother);
160 }
```

KU KONKUK UNIVERSITY

Major Procedure

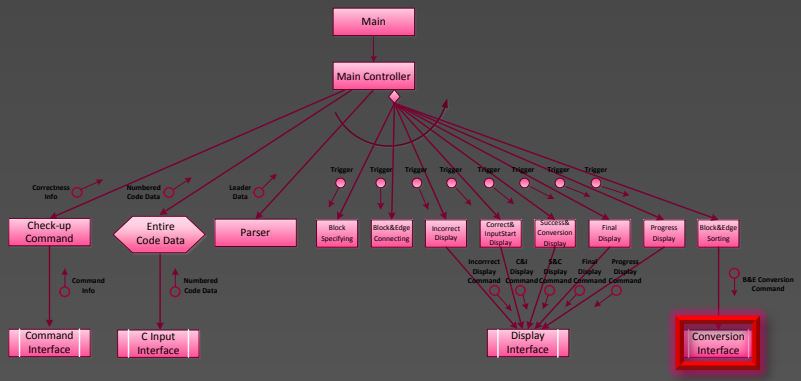


Reference No.	2.1.10
Name	Block&Edge Sorting
Input	Parsed Data, Trigger
Output	B&E Conversion Command
Description	Receiving ***Parsed Data***, outputs ***B&E Conversion Command*** to be converted into final CFG in sorted text form according to each block's level.

```
207 while(temp[index] != NULL)
208 {
209 for(i=0;i<temp[index]->children_size;i++)
210 {
211 j=0;
212 while(temp[j++]!=0);
213 temp[--j] = temp[index]->children[i];
215 }
216 if(temp[index]->keyword != l_root)
217 {
218 result = search_block(temp[index]-
>block_start, temp[index]->block_end);
219 result_block_list[rblist_size++] =
block_list[result];
220 }
221 for(i=0;i<elist_size;i++)
222 {
223 if(edge_list[i]->source_block == result)
224 {
225 result_edge_list[relist_size++] =
edge_list[i];
226 }
227 }
228 index++;
229 }
```

KU KONKUK UNIVERSITY

Major Procedure



Reference No.	2.3
Name	Conversion Interface
Input	Conversion Command
Output	CFG Info
Description	Receiving ***Conversion Command***, converts it to final output ***CFG Info***. It enables CFG Generator to report with a text file by sending ***CFG Info*** to ***CFG Text File***.

```
253 for(i=0; i<rblist_size; i++)
254 {
255 fprintf(report," %d / ",
result_block_list[i]->id+1);
257 switch(result_block_list[i]->type)
258 {
259 case basic:
260 fprintf(report,"Basic block / ");
261 break;
262 case loop_header:
263 fprintf(report,"Loop header / ");
264 break;
265 case entry_b:
266 fprintf(report,"Entry block / ");
267 break;
268 case exit_b:
269 fprintf(report,"Exit block / ");
270 break;
271 }
273 fprintf(report,"%d / %d / ",
result_block_list[i]->block_start+1,
result_block_list[i]->block_end+1);
274 fprintf(report,"%s\r\n",
result_block_list[i]->description);
275 }
```

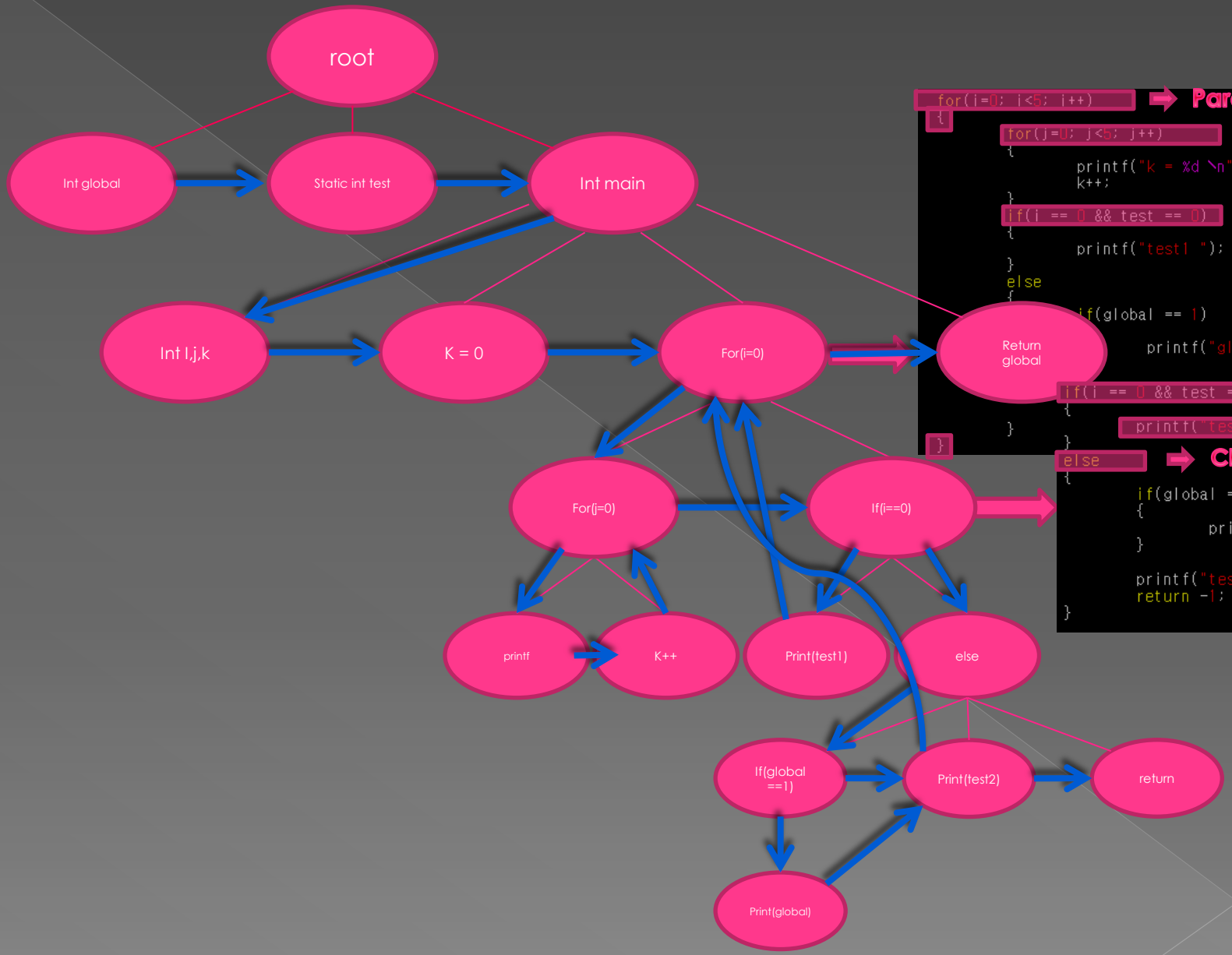
KU KONKUK UNIVERSITY

Converting Algorithm

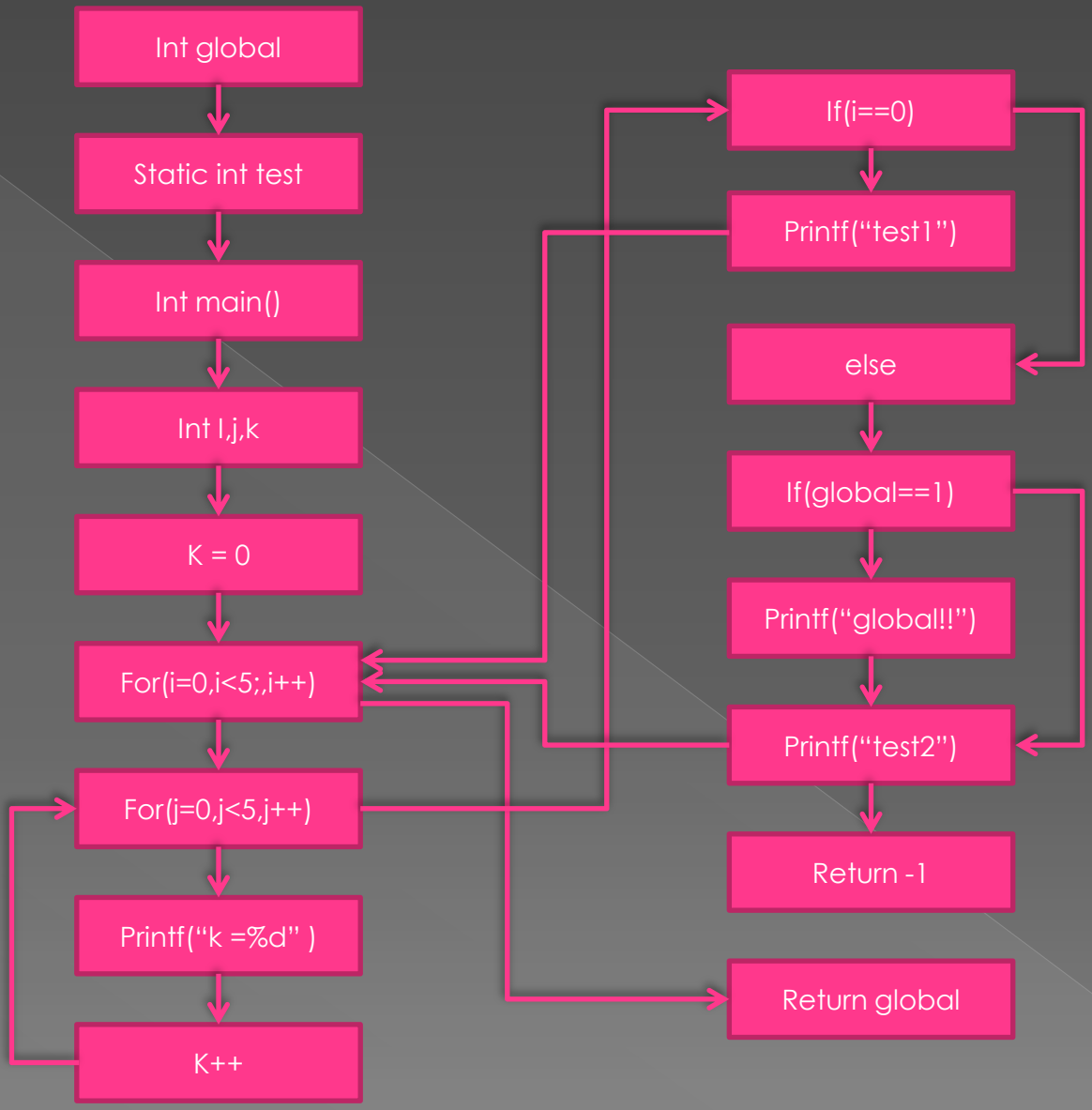
KU KONKUK UNIVERSITY

```c
#include <stdio.h>
#include <stdlib.h>

int global = 1;
static int test = 0;

int main(int argc)
{
        int i, j, k;
        k = 0;

        for(i=0; i<5; i++)
        {
                for(j=0; j<5; j++)
                {
                        printf("k = %d \n", k);
                        k++;
                }
                if(i == 0 && test == 0)
                {
                        printf("test1 ");
                }
                else
                {
                        if(global == 1)
                        {
                                printf("global 1!!!!\n");
                        }

                        printf("test2 ");
                        return -1;
                }
        }
        return global;
}
```

Block !!!
Block !!!
Block !!!
Block !!!
Block !!!
Block !!!
Block !!!
Block !!!
Block !!!
Block !!!
Block !!!
Block !!!
Block !!!
Block !!!
Block !!!

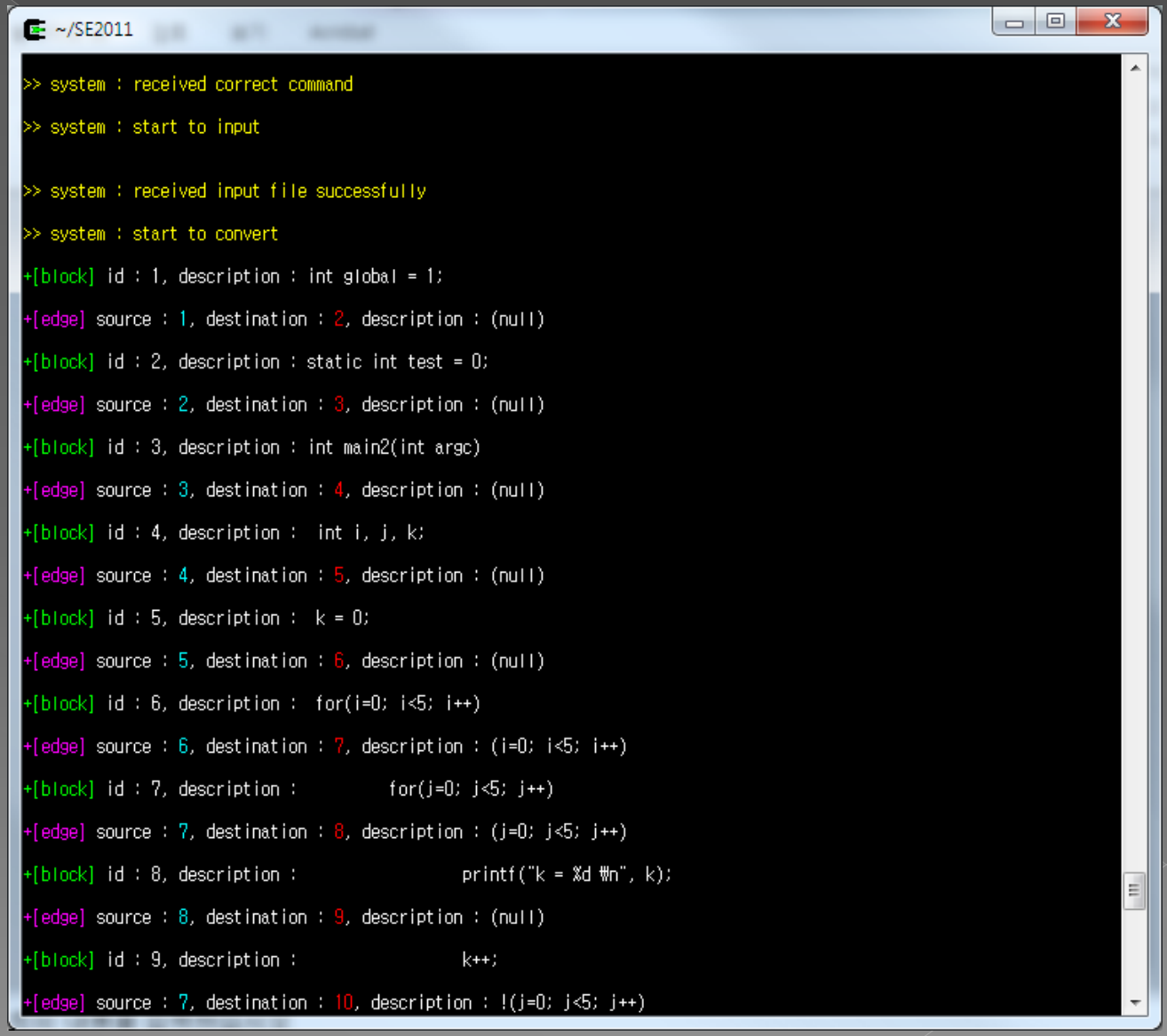


KU KONKUK UNIVERSITY



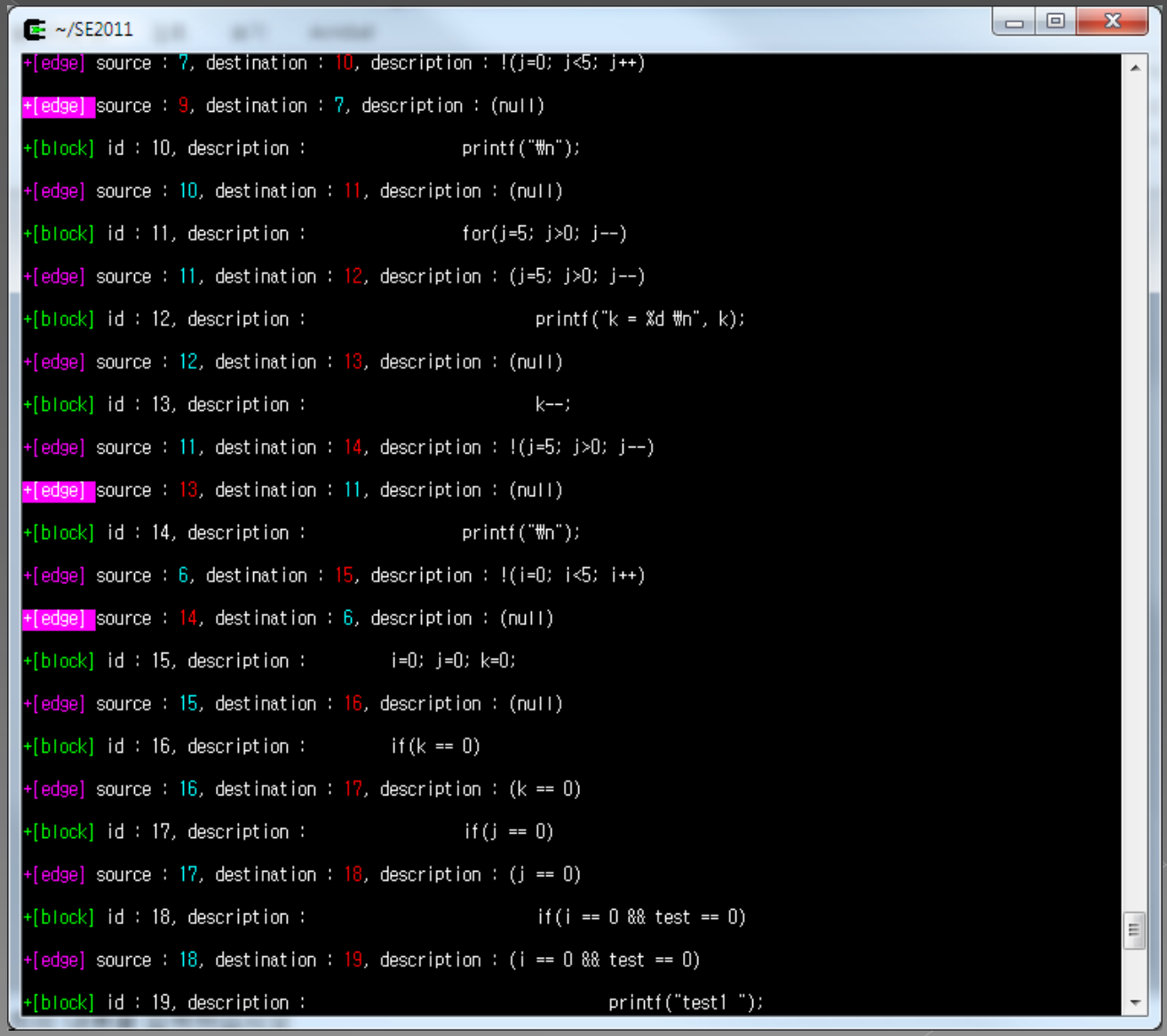
KU KONKUK UNIVERSITY

Program Screenshot



KU KONKUK UNIVERSITY

Program Screenshot

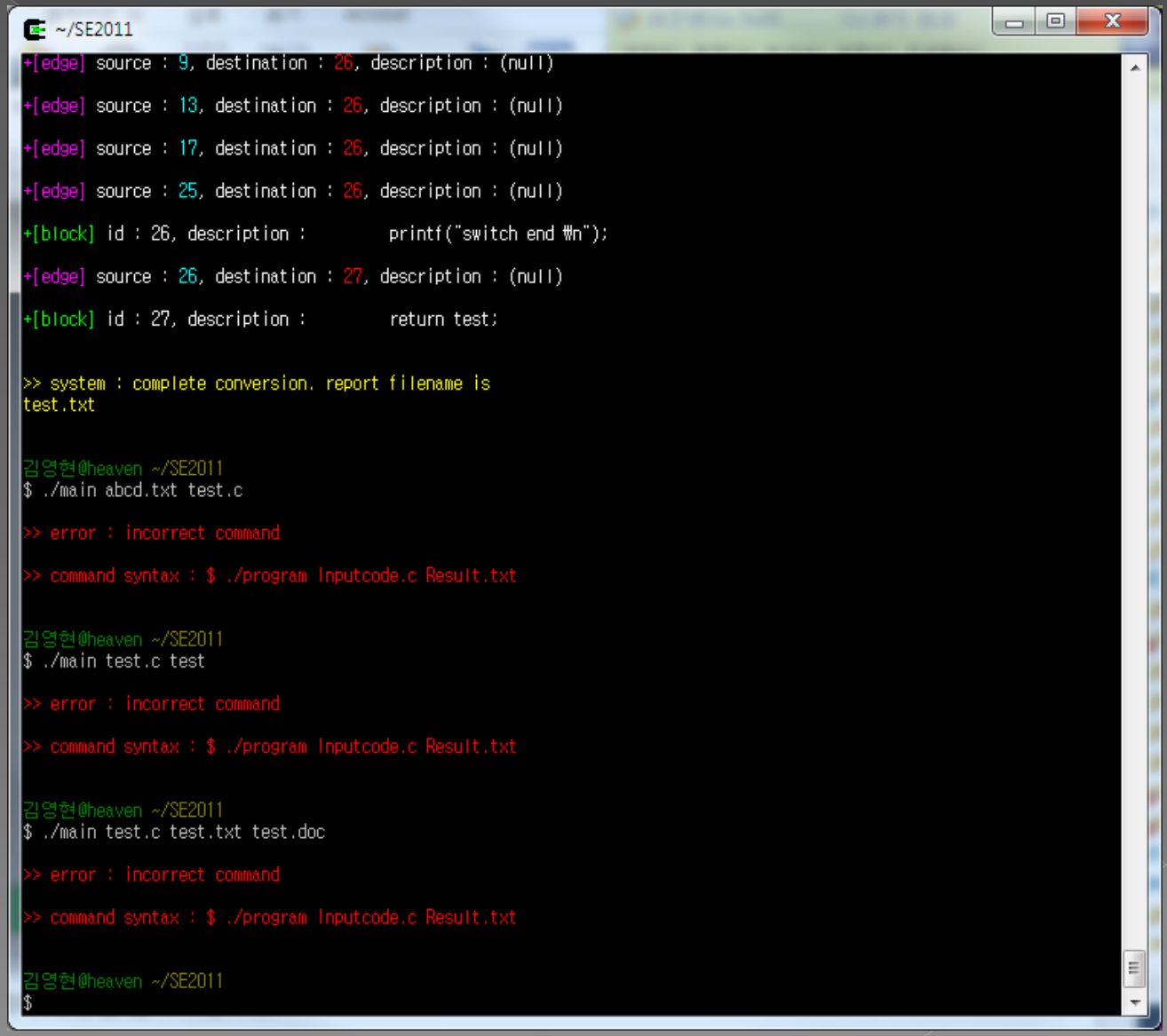


KU KONKUK UNIVERSITY

Program Screenshot

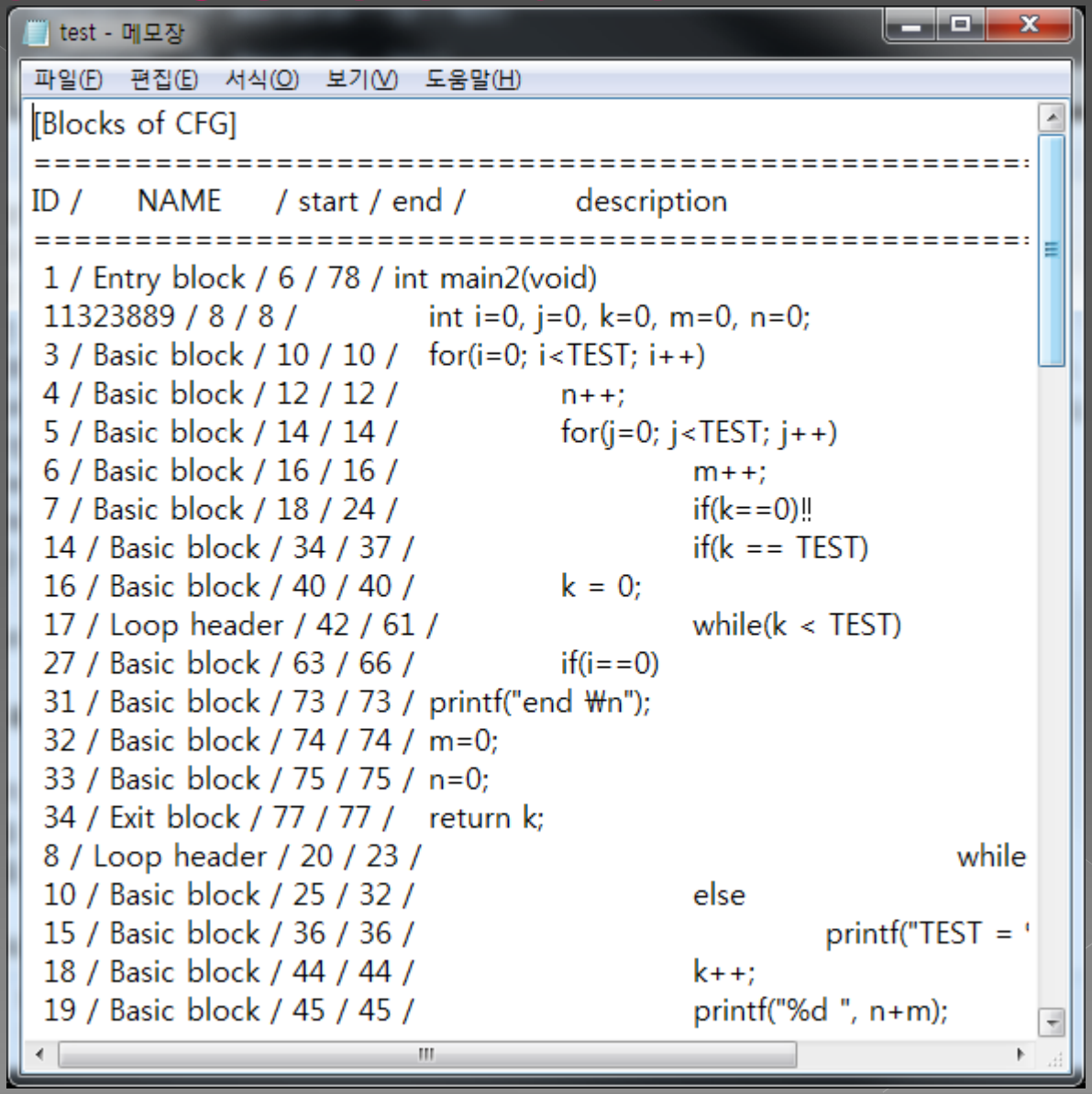
KU KONKUK UNIVERSITY

Program Screenshot



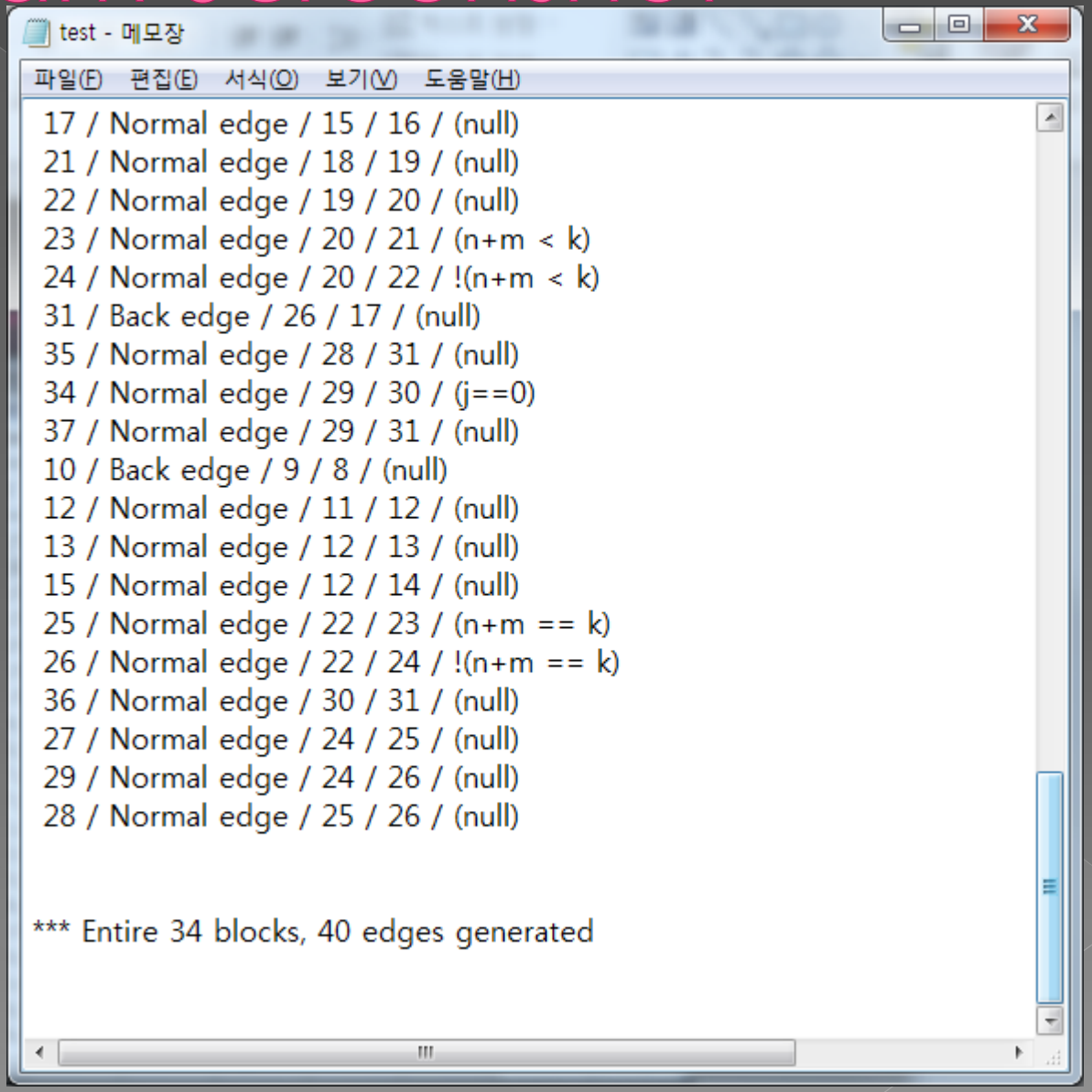
KU KONKUK UNIVERSITY

Program Screenshot



KU KONKUK UNIVERSITY

Program Screenshot



KU KONKUK UNIVERSITY

Demonstration

KU KONKUK UNIVERSITY

Demonstration

KU KONKUK UNIVERSITY

Question?

KU KONKUK UNIVERSITY