Control Flow Graph (CFG) generator

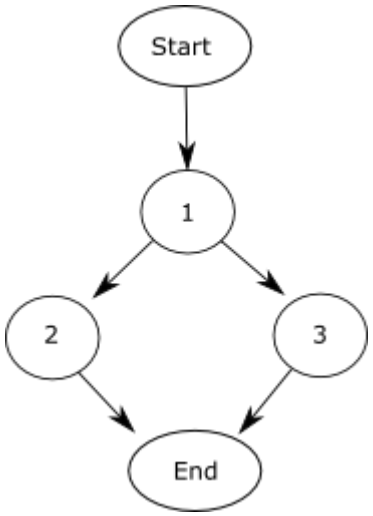
A반 T2 - 김우빈 (201011321)
임국현 (201011358)
박대규 (201011329)

목차

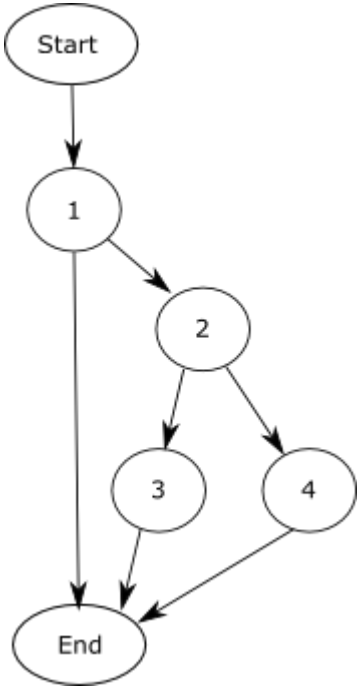
- CFG Introduce
- CFG Algorithm
- CFG Statement of Purpose

CFG Introduce

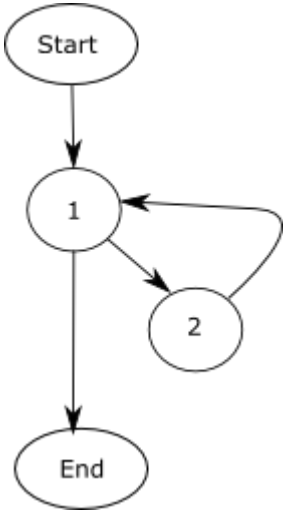
If Loop



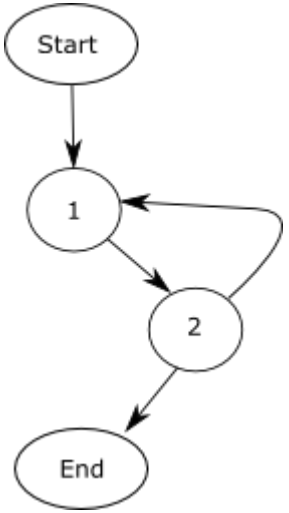
Nested If Loop



While Loop

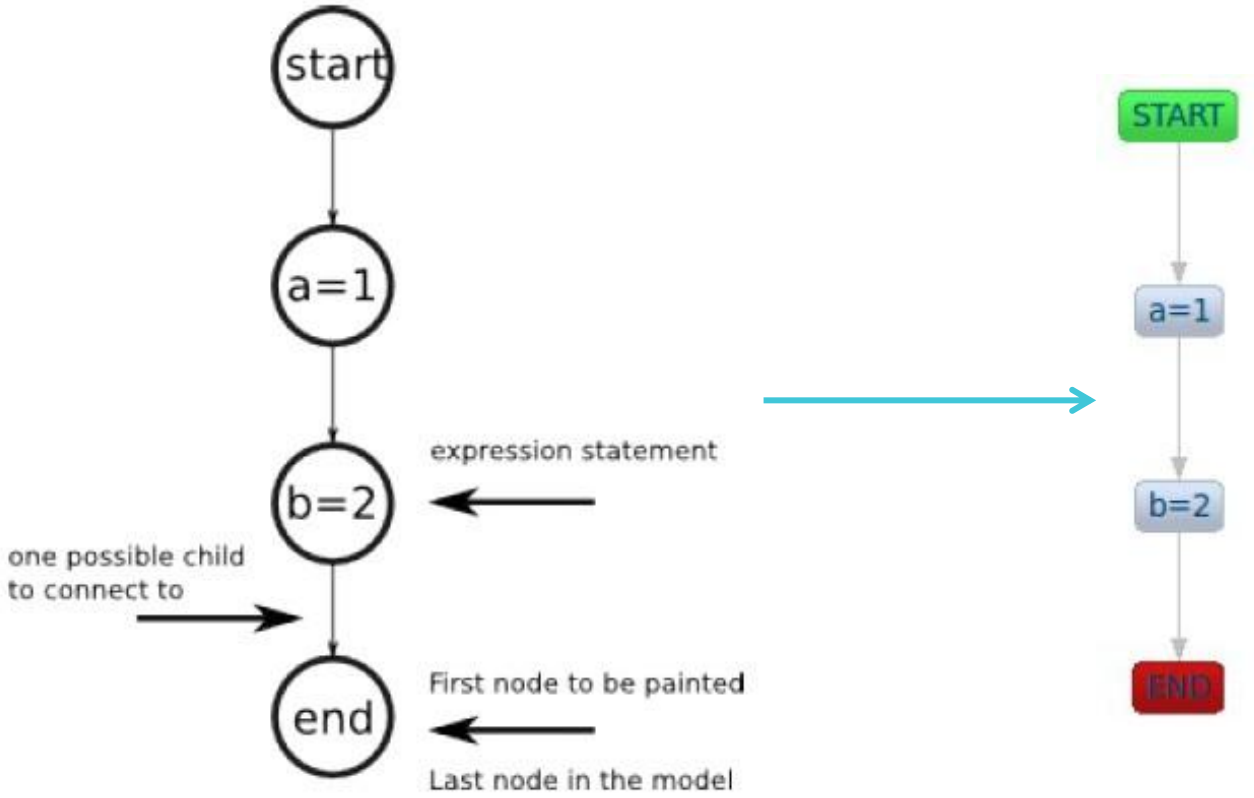


Do While Loop



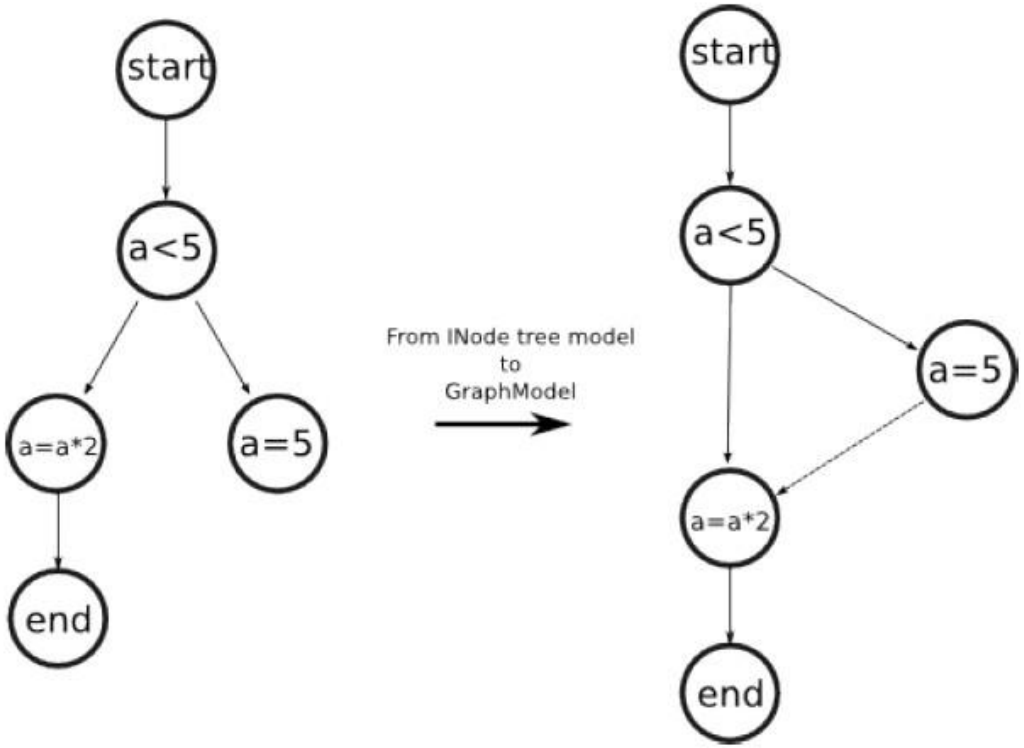
CFG Algorithm

```
void expression_statement(int b,int a) {
        a = 1;
        b = 2:
}
```



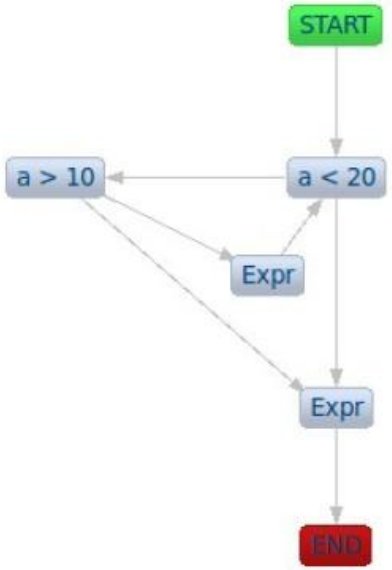
CFG Algorithm

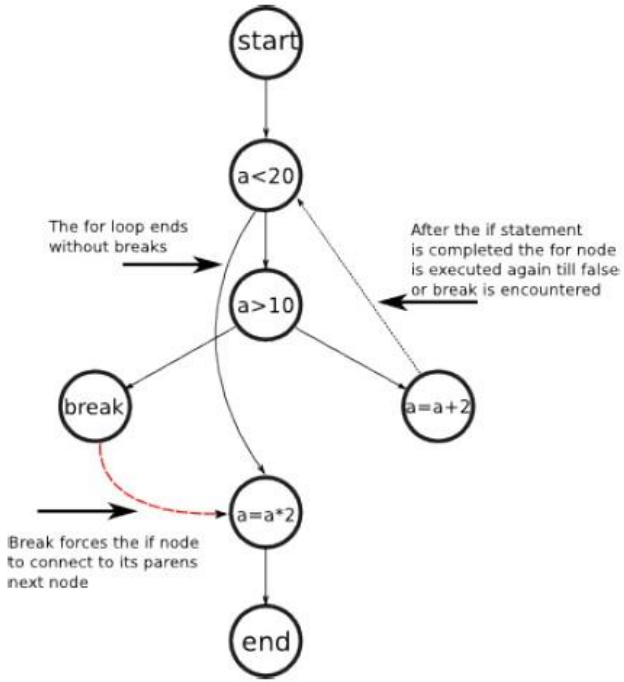
```
void ifTestMethod(int a) {
        if (a < 5) {
                a = 5;
        }
        a= a*2;
}
```



CFG Algorithm

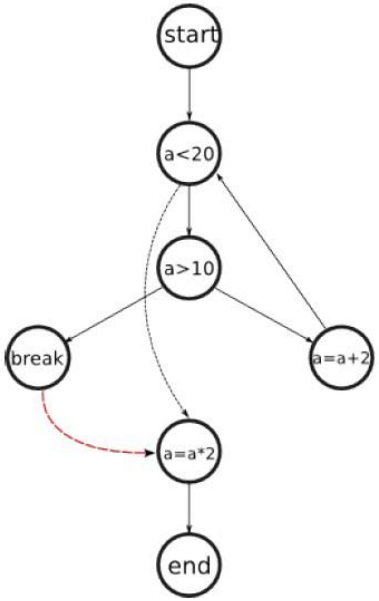
```
void forTestMethod(int a) {
        for (a = 1; a < 20;) {
                if (a > 10) {
                        break;
                } else {
                        a = a + 2;
                }
        }
        a= a*2;
}
```

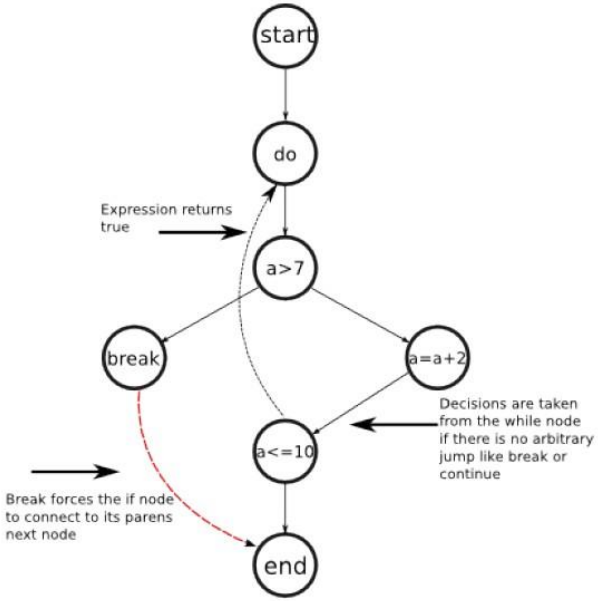


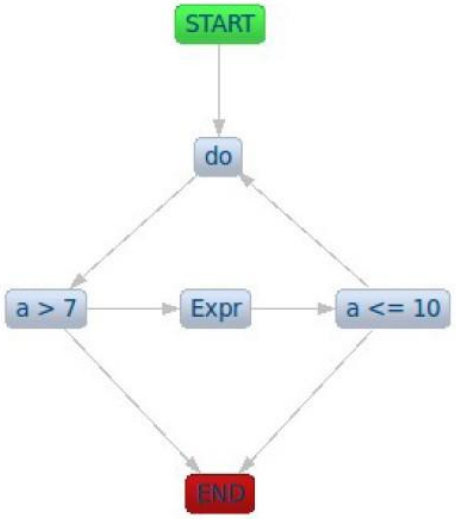


CFG Algorithm

```
void doWhileTestMethod(int a) {
        do {
                if (a > 7) {
                        break;
                } else {
                        a = a + 2;
                }
        } while (a <= 10);
}
```

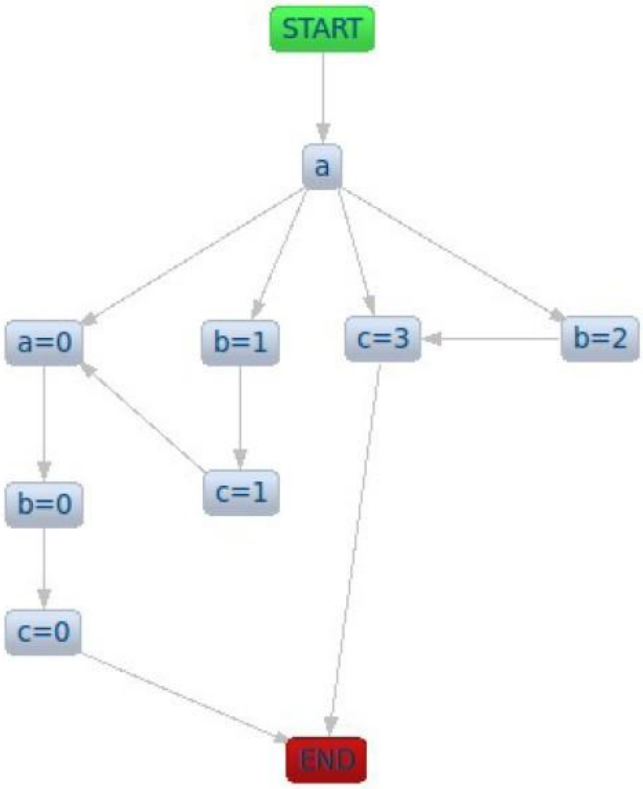






CFG Algorithm

```
void switchTestMethod(int a, int b, int c) {
        switch (a) {
                case 1:
                        b = 2;
                case 2:
                        c = 3;
                         break;
                case 3:
                        b = 1;
                        c = 1;
                default:
                        a = 0;
                        b = 0;
                        c = 0;
        }
}
```



CFG Algorithm

```
1  worklist = {start-block:Ø}
2  while (worklist)
3          remove element e from worklist
4          process-block(e.block, e.list)
5
6  process-block(block, counter list)
7          if block has been seen with counter list before
8                  break
9          for each instruction i in block
10                 decrement counters in counter list
11                 if i is a branch
12                         counter list = counter list + {i : branch-latency}
13                 if any counter in counter list = 0
14                         break for
15         if i is not at end of block
16                 create new block with remaining instructions in block
17                 add edge from block to new block
18         if no counter in counter list = 0
19                 let f = block's fall through block
20                 worklist = worklist + { f : counter list }
21         else
22                 let j = branch instruction in counter list with (counter = 0)
23                 for each target block t of instruction j
24                         add edge from block to target t
25                         worklist = worklist + { t : counter list - {j : 0}}
```

Statement Of Purpose

- 이 프로그램은 C소스 코드를 받아 CFG(Control Flow Graph)를 그리는 프로그램입니다.
- C소스 코드를 불러와 소스를 분석하여 소스코드의 구조를 파악합니다.
- 파악한 소스코드 구조를 바탕으로 전체적인 구조를 설계해 냅니다.
- 설계한 소스코드의 구조를 바탕으로 소스코드의 Graph를 그려냅니다.
- 이 프로그램의 핵심 요소는 소스코드를 분석하는 알고리즘설계 그리고 분석된 내용을 CFG로 그려내는 것 입니다.