CFG_Gen

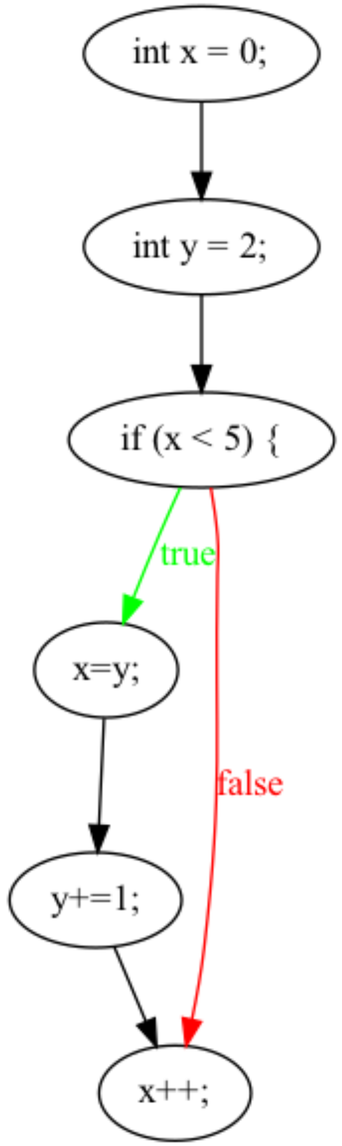
A tool to generate control flow graphs for C programs

Team 11

CFG

Control Flow Graphs

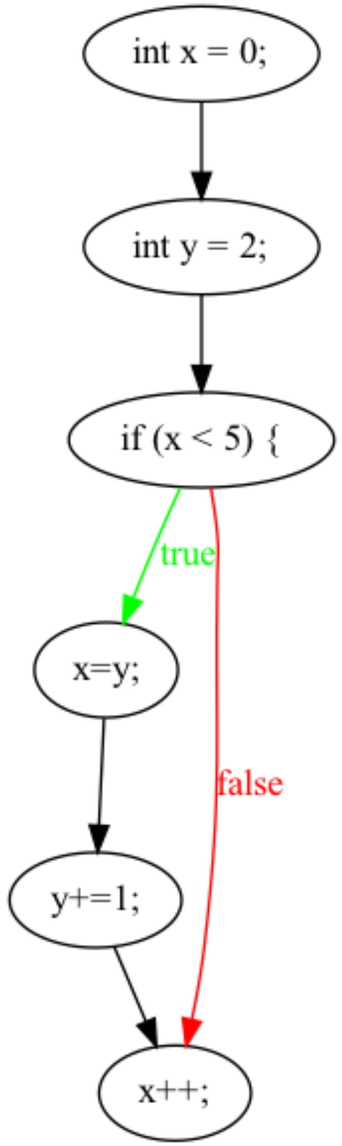
- directed graph in which the nodes represent basic blocks and the edges represent control flow paths
- basic block: linear sequence of program instructions having one entry point and one exit point.



Why CFGs?

Why CFGs?

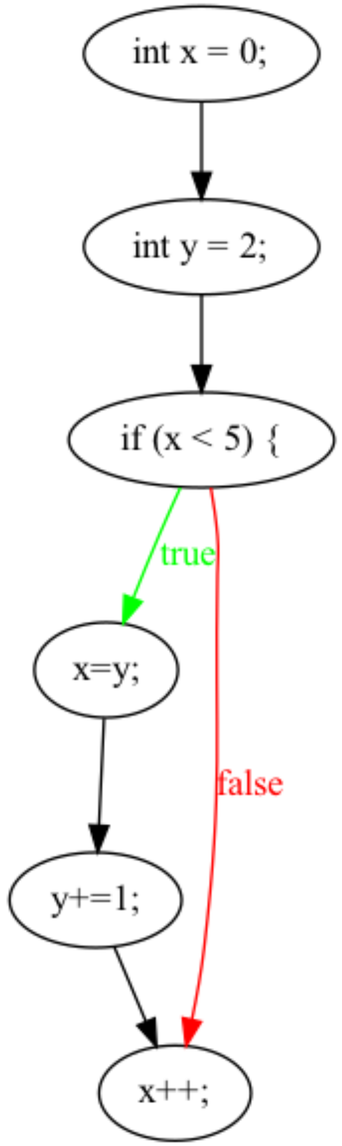
- It’s important to know how the program will behave in different situations
- Static analysis: doesn’t need to run the code.
- Gives insight into all paths that the program can traverse when executed.



Why CFG_Gen?

Why CFG_Gen?

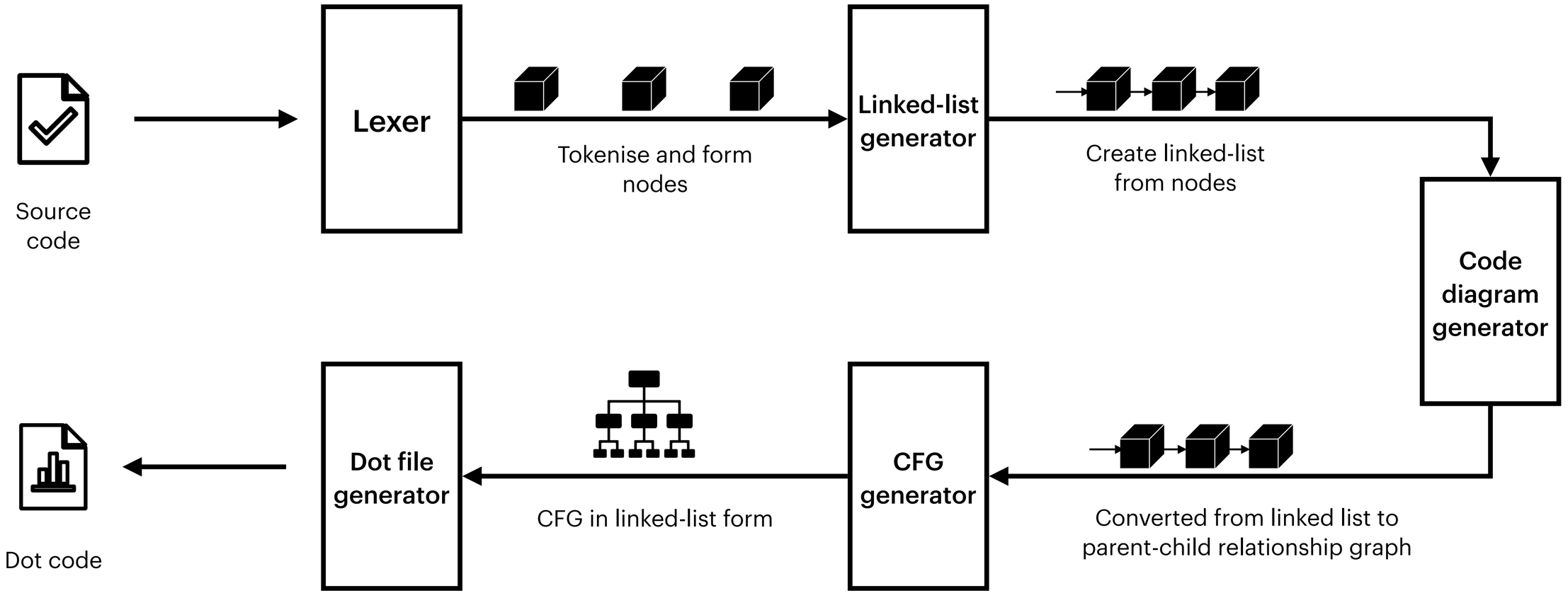
- Drawing CFGs by hand can be taxing for large and complex programs.
- A manual approach is prone to errors.
- The tool being presented draws the control flow graphs given syntactically correct programs.



Code Walkthrough

CFG_Gen

Structure



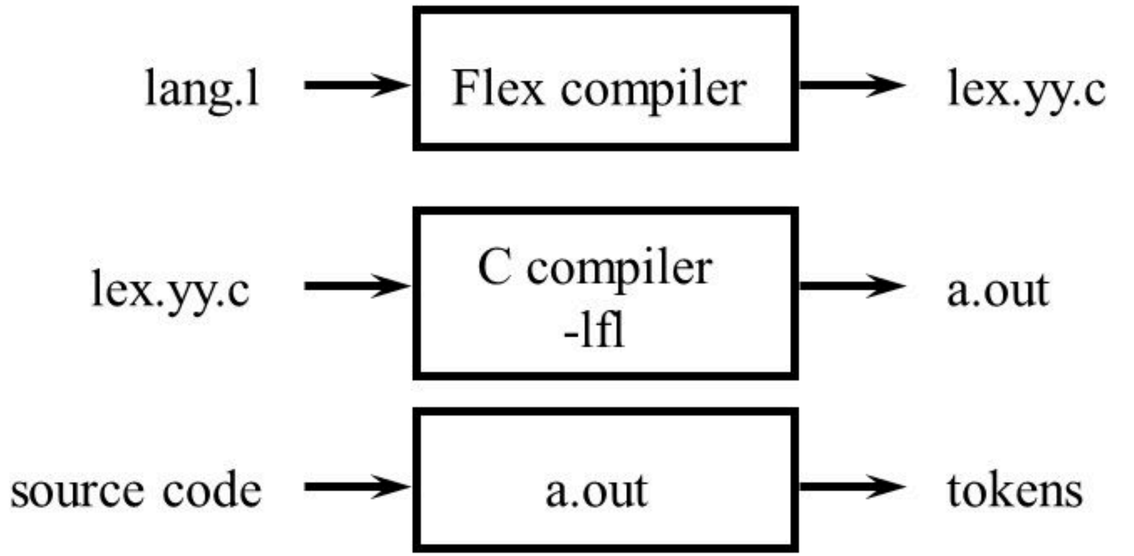
Scanner Generator

Scanner generator

- Have used Flex(fast lexical analyser generator), a fast lexical scanner generator.
- Regular expressions have been used to make patterns in flex file.
- The patterns have been used to detect different syntax.
- Macros have been used to communicate between the scanner and the linked list generator file.
- Special attention is provided to deal with scope, multiline statements and single line statements.

Lexer

A *language* for specifying lexical analyzers



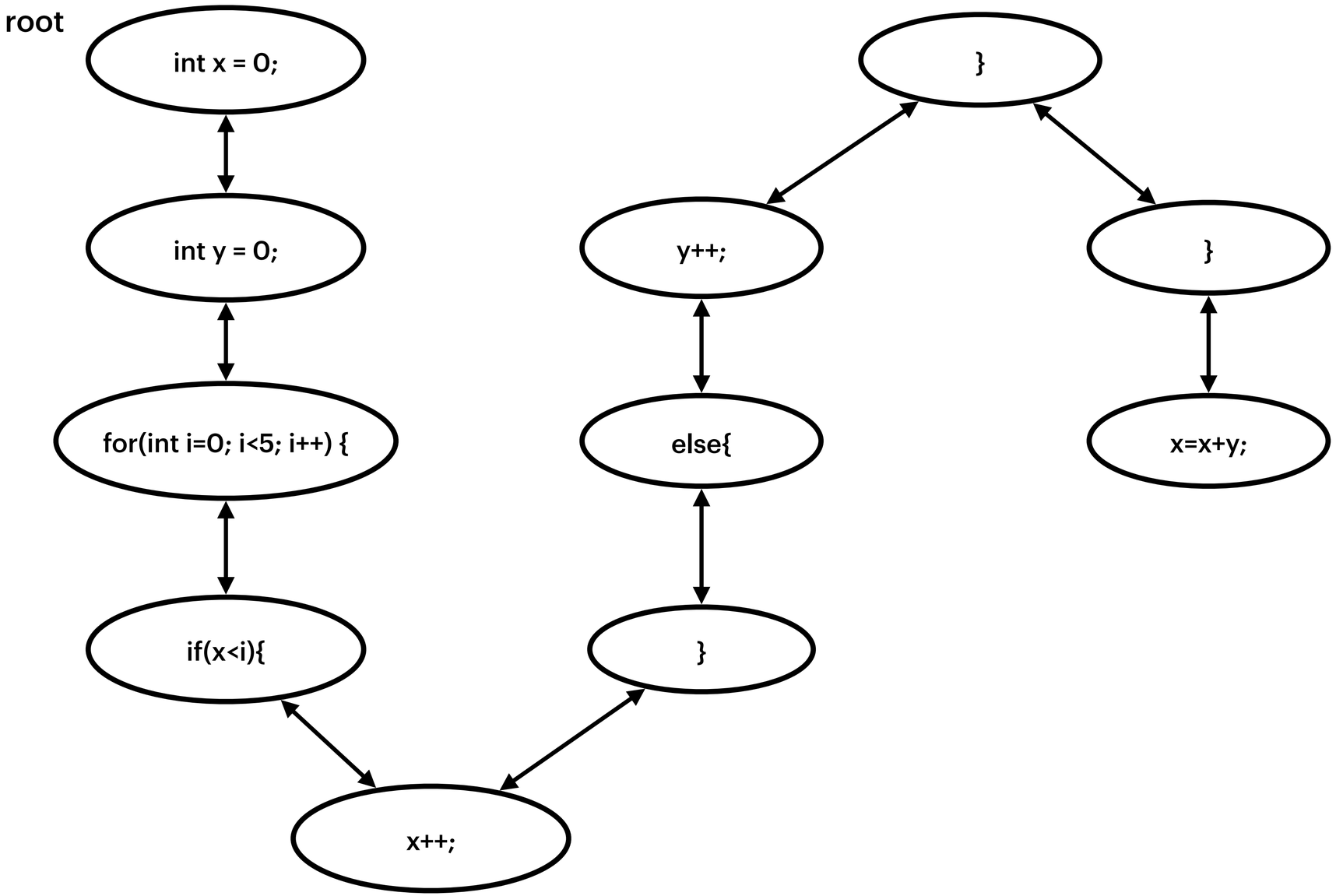
Linked list Generator

Linked list Generator

- Generates a different linked list for different functions as there will be a separate CFG for each function.
- Nodes are created based on the token class, which we receive from the lexer.
- We create a vector that comprises of the root nodes of each function, which is taken as the input for CDG (Code Diagram Generator).

Example: Source Code -> Linked list input

```
int x = 0 ;
int y = 0 ;
for(int i=0; i<5;i++) {
  if ( x < i ) {
    x++;
  }
  else {
    y++;
  }
}
x=x+y;
```



Code Diagram Generator

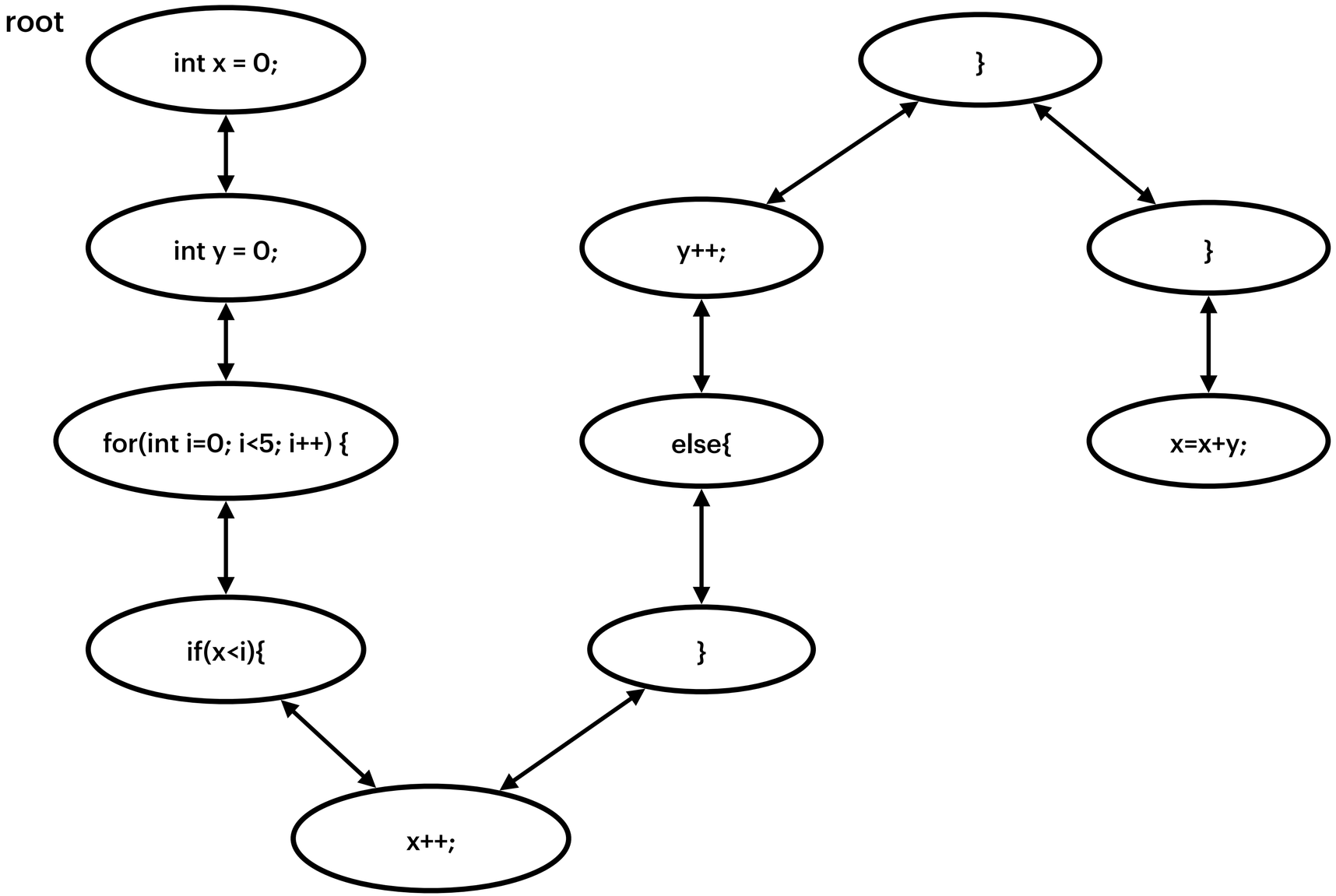
CDG Generator

- Takes a linked list of *Node* (every statement in the graph), which is returned by the *ll_generator*.
- For blocks like if, else if, for, while, switch, case: statements inside it are considered as children.
- This is maintained as a child-parent pointer in *Node of the first child*.

```
int x = 0 ;
int y = 0 ;
for(int i=0; i<5;i++) {
  if ( x < i ) {
    x++;
  }
  else {
    y++;
  }
}
x=x+y;
```

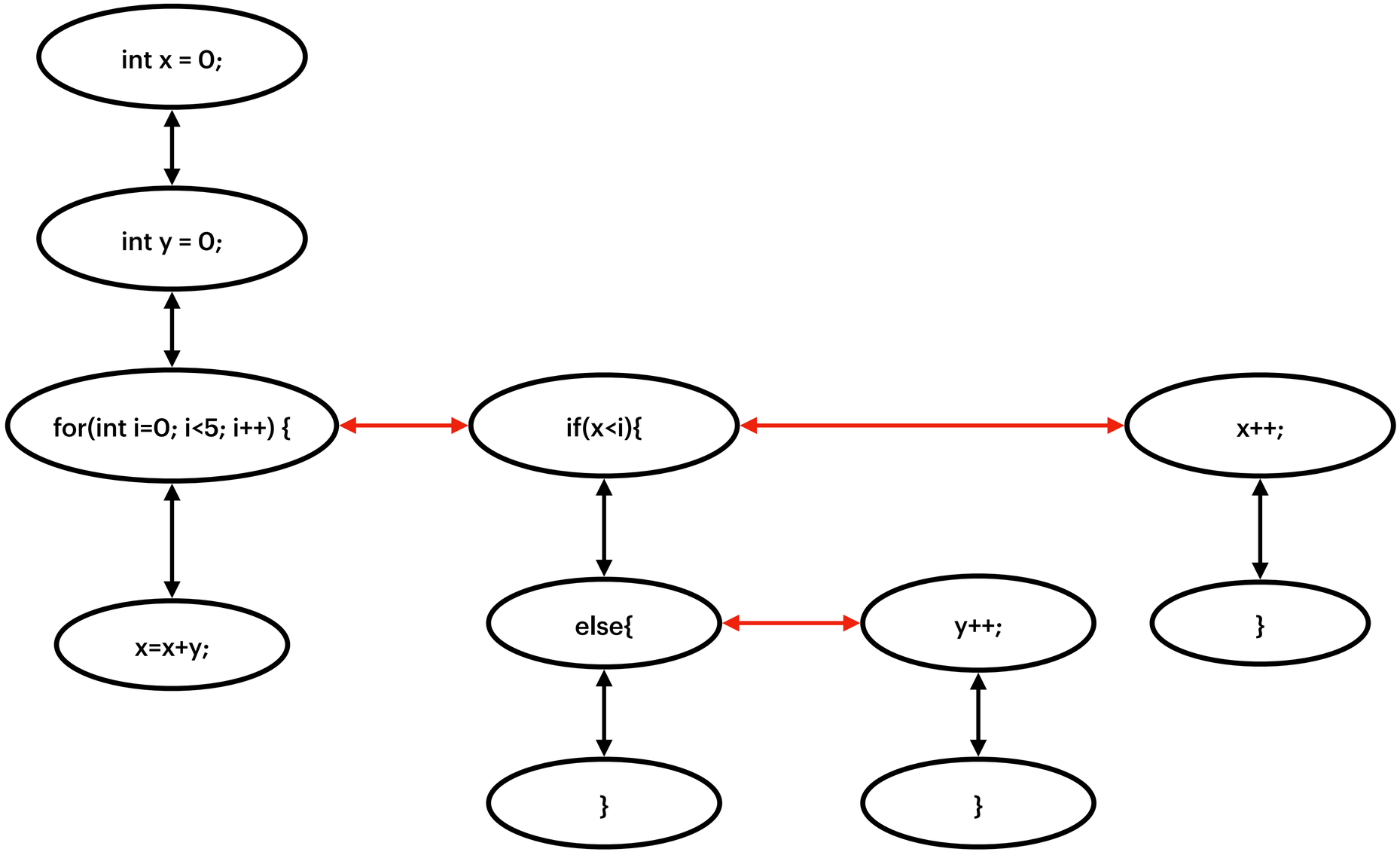
Example: Linked list input -> CDG output

```
int x = 0 ;
int y = 0 ;
for(int i=0; i<5;i++) {
  if ( x < i ) {
    x++;
  }
  else {
    y++;
  }
}
x=x+y;
```

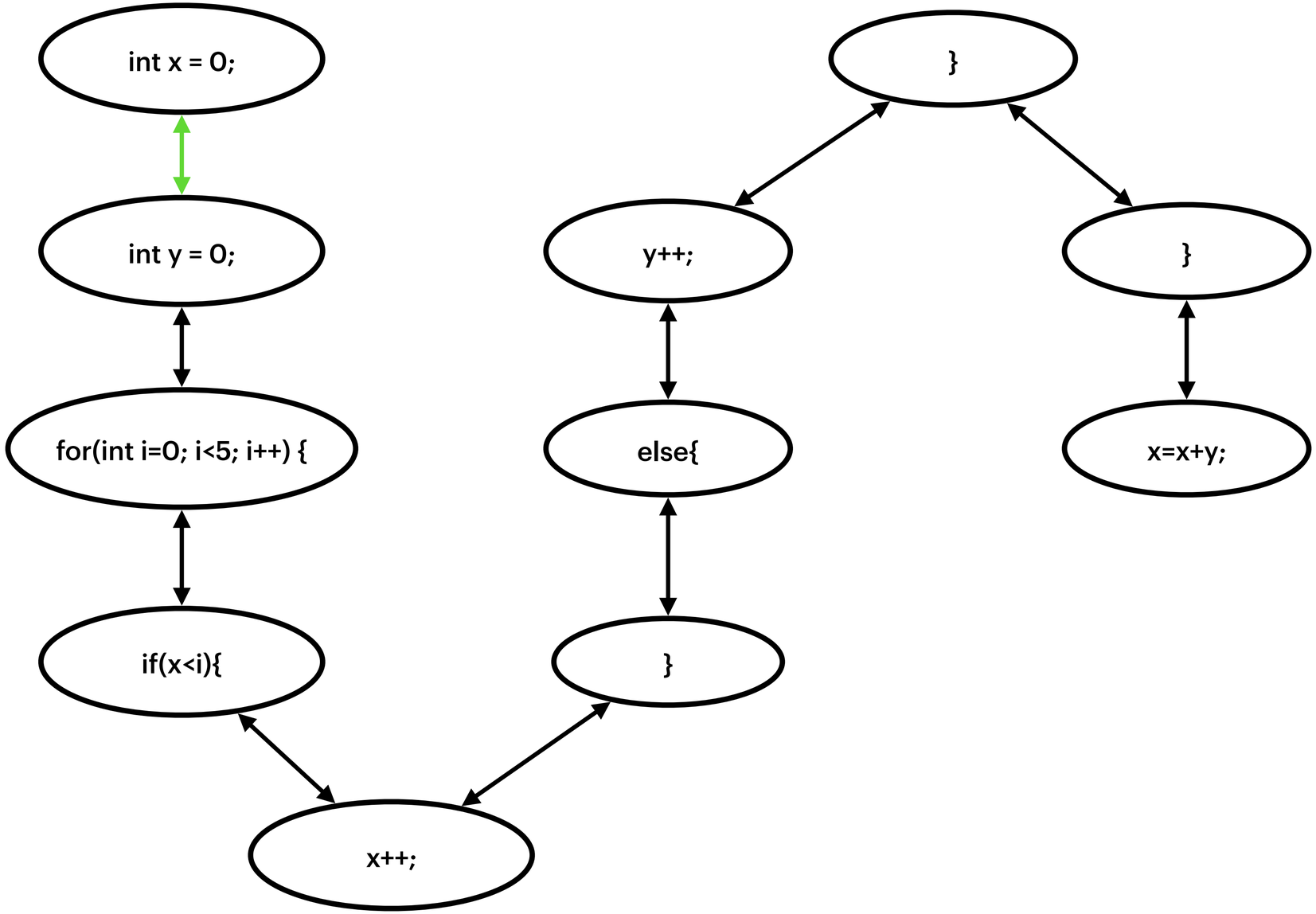


Example: Linked list input -> CDG output

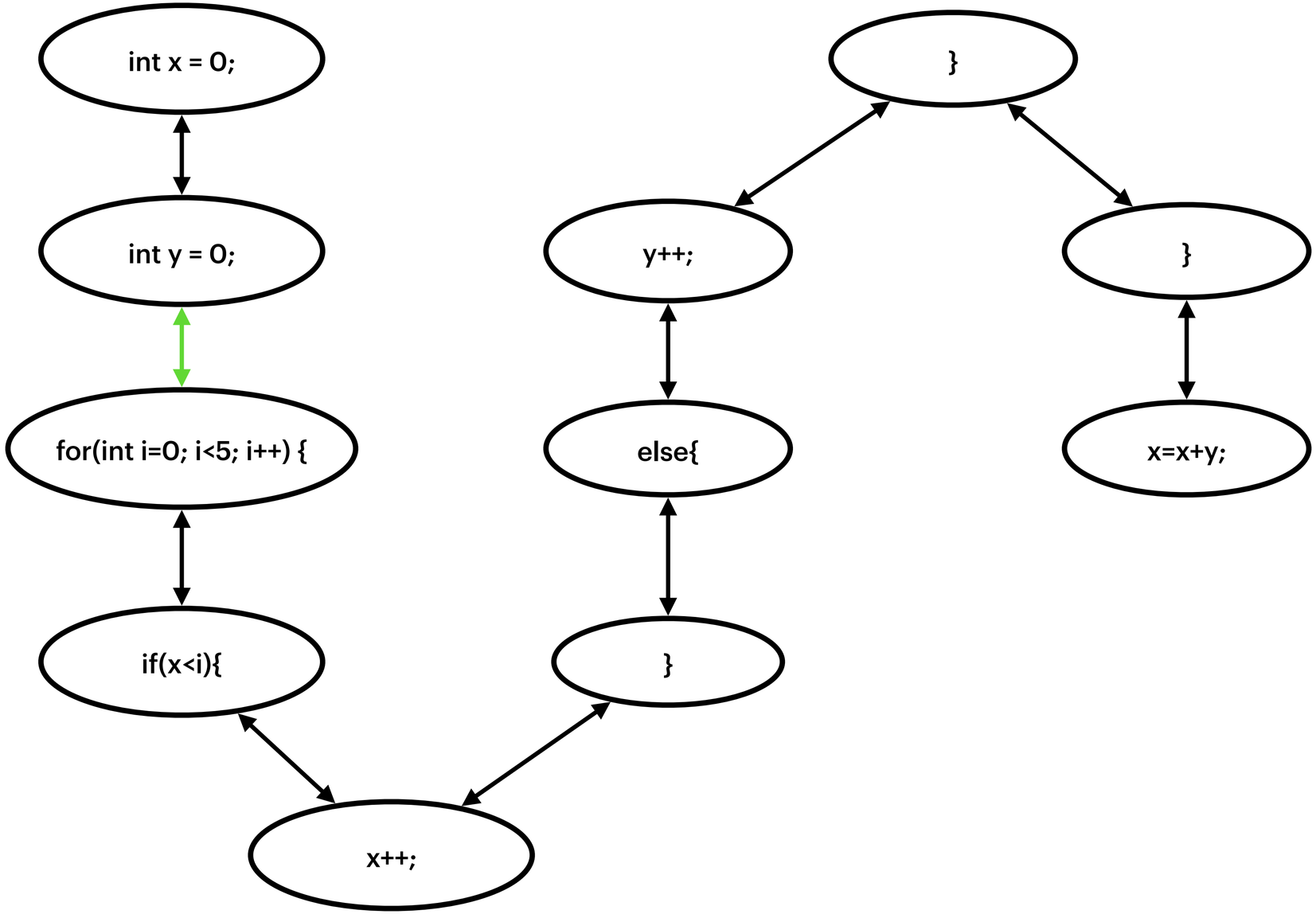
```
int x = 0 ;
int y = 0 ;
for(int i=0; i<5;i++) {
  if ( x < i ) {
    x++;
  }
  else {
    y++;
  }
}
x=x+y;
```



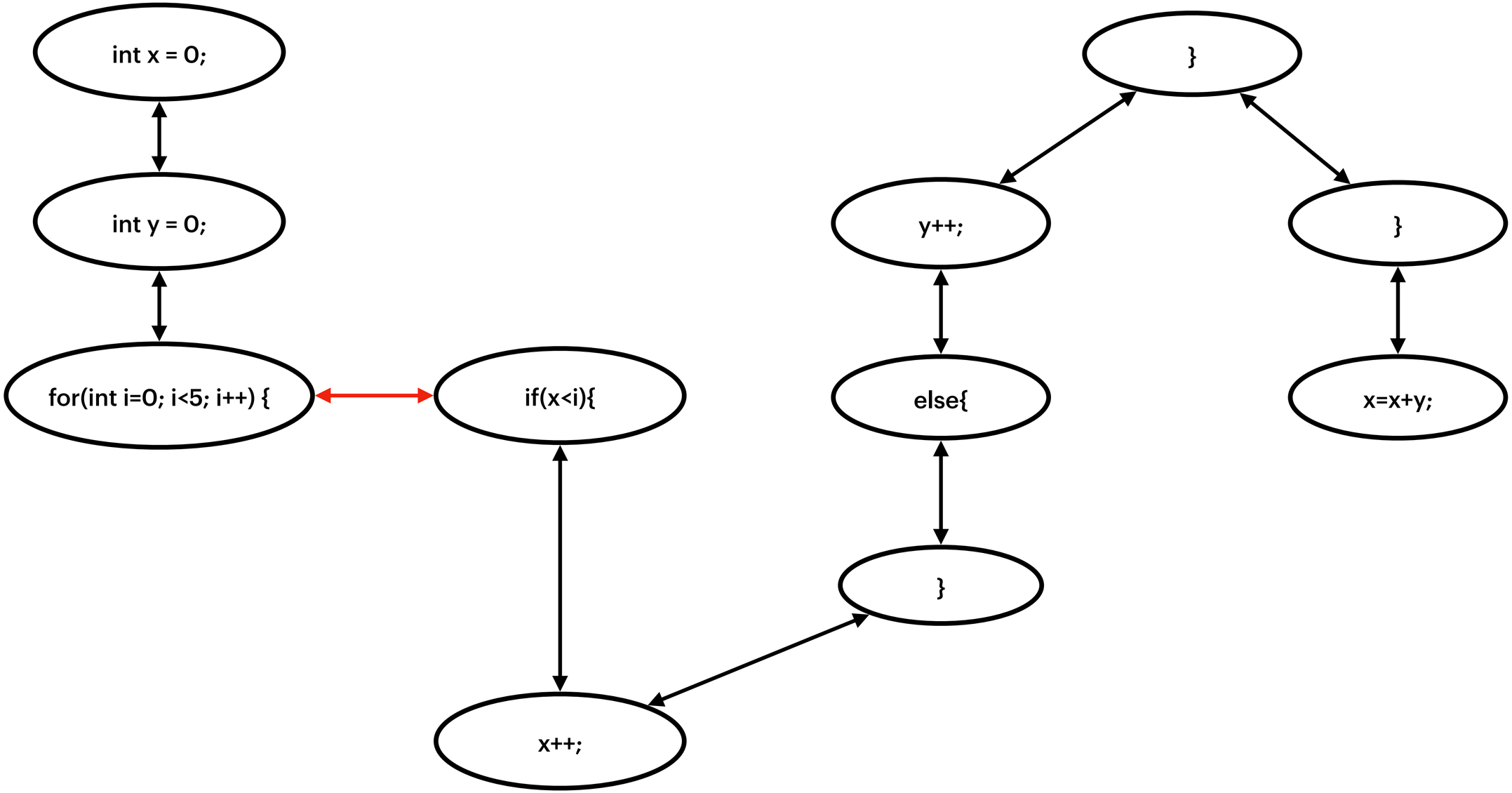
Example: Linked list input -> CDG output



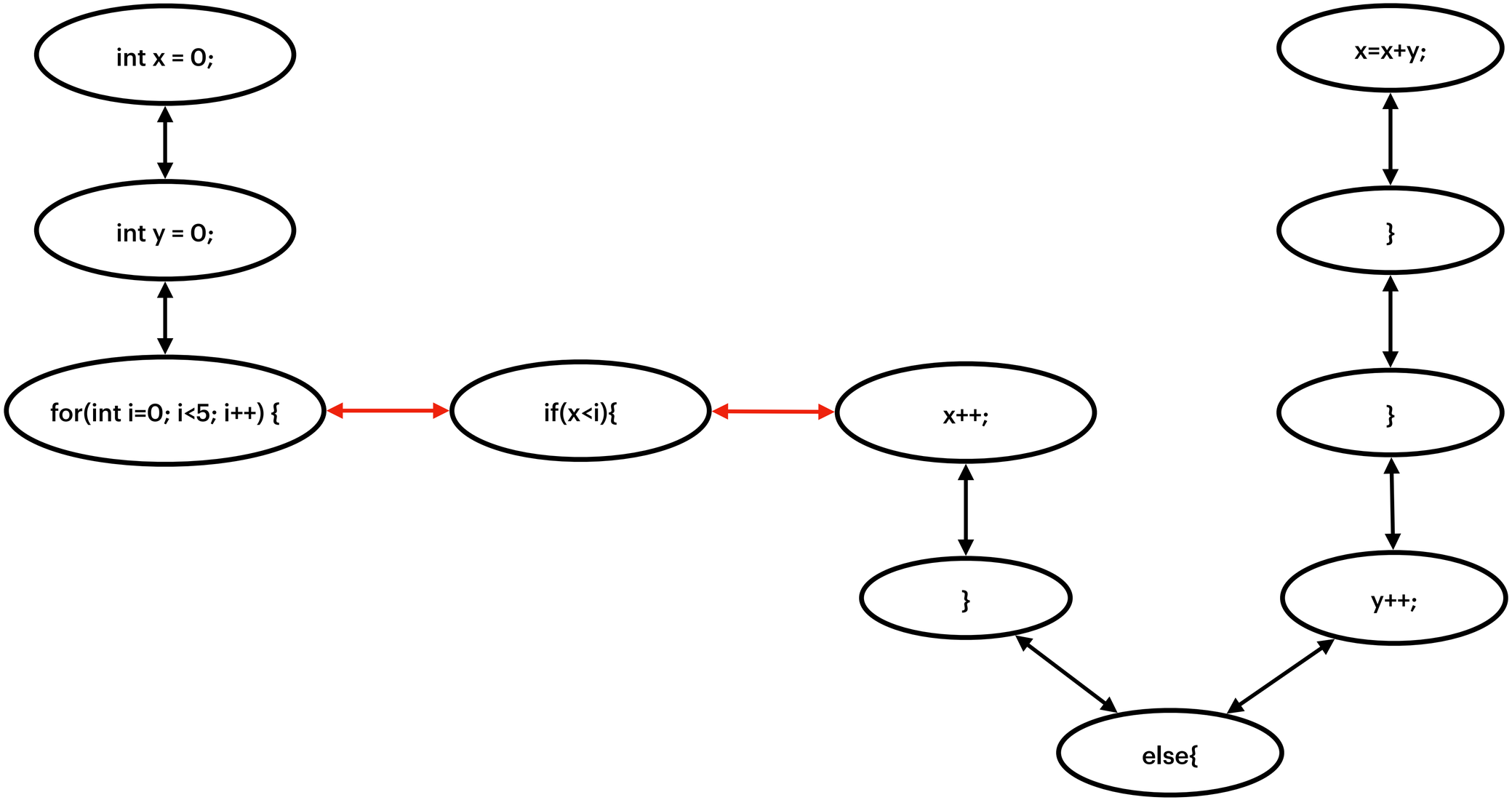
Example: Linked list input -> CDG output



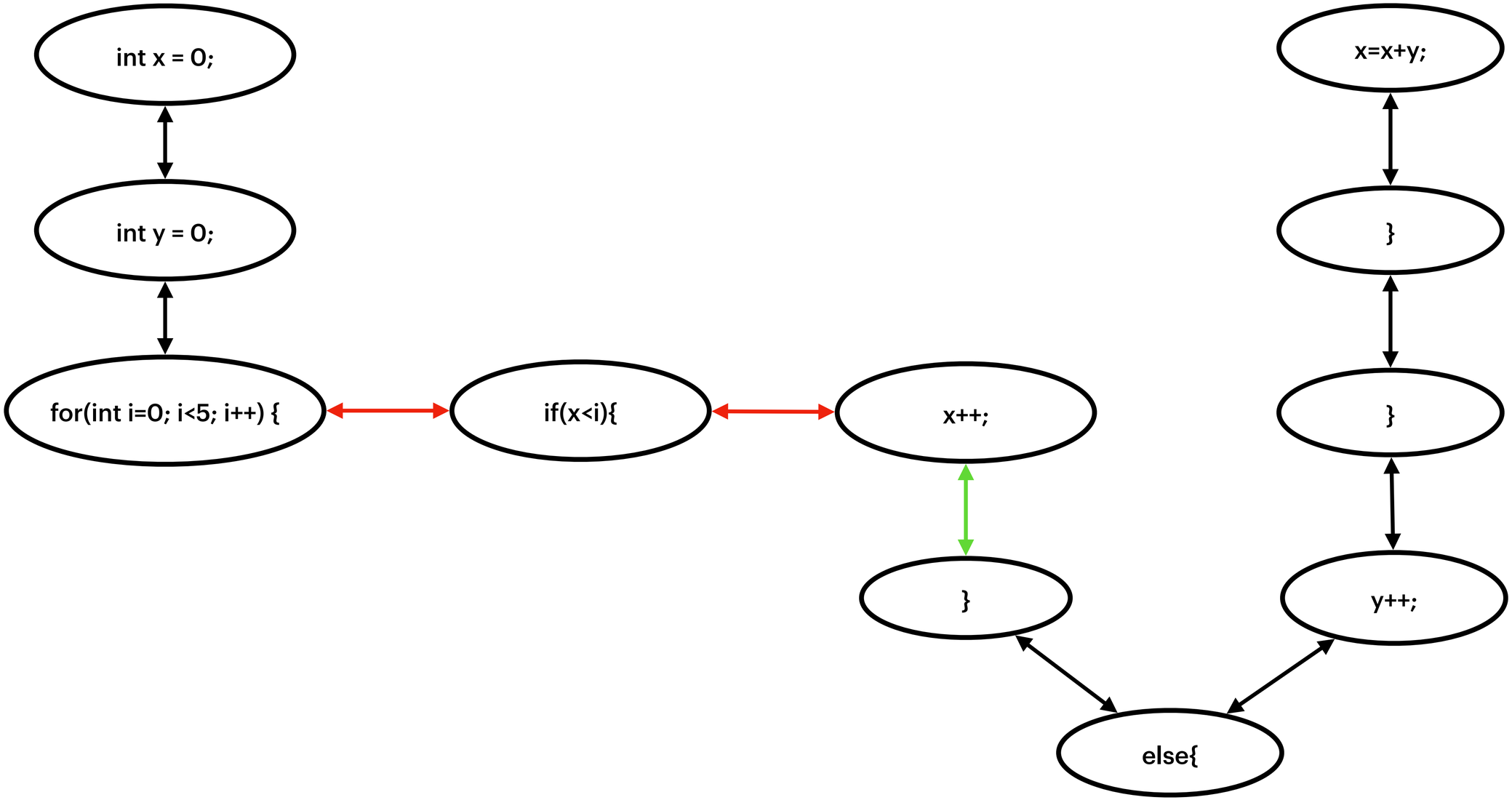
Example: Linked list input -> CDG output



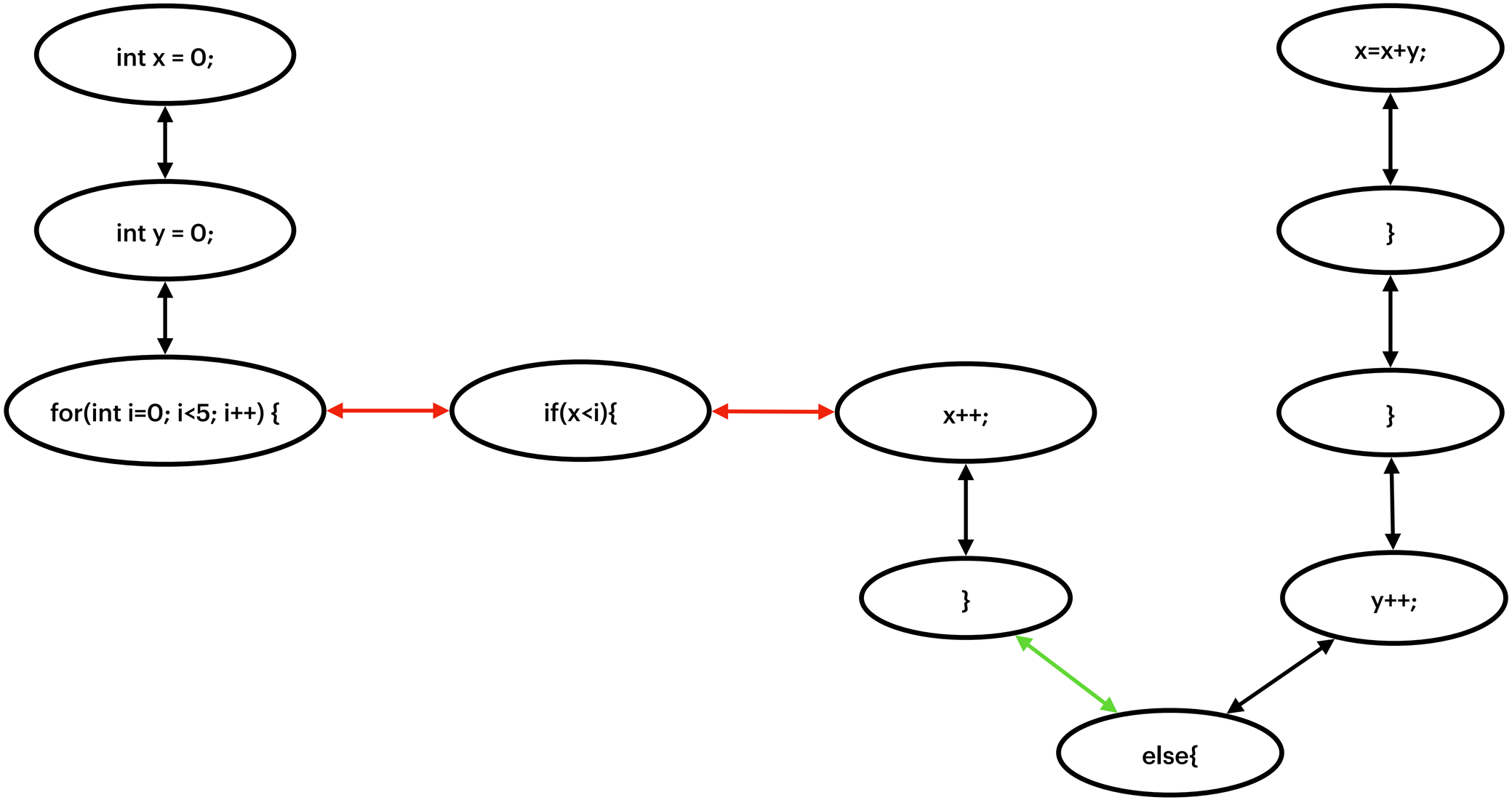
Example: Linked list input -> CDG output



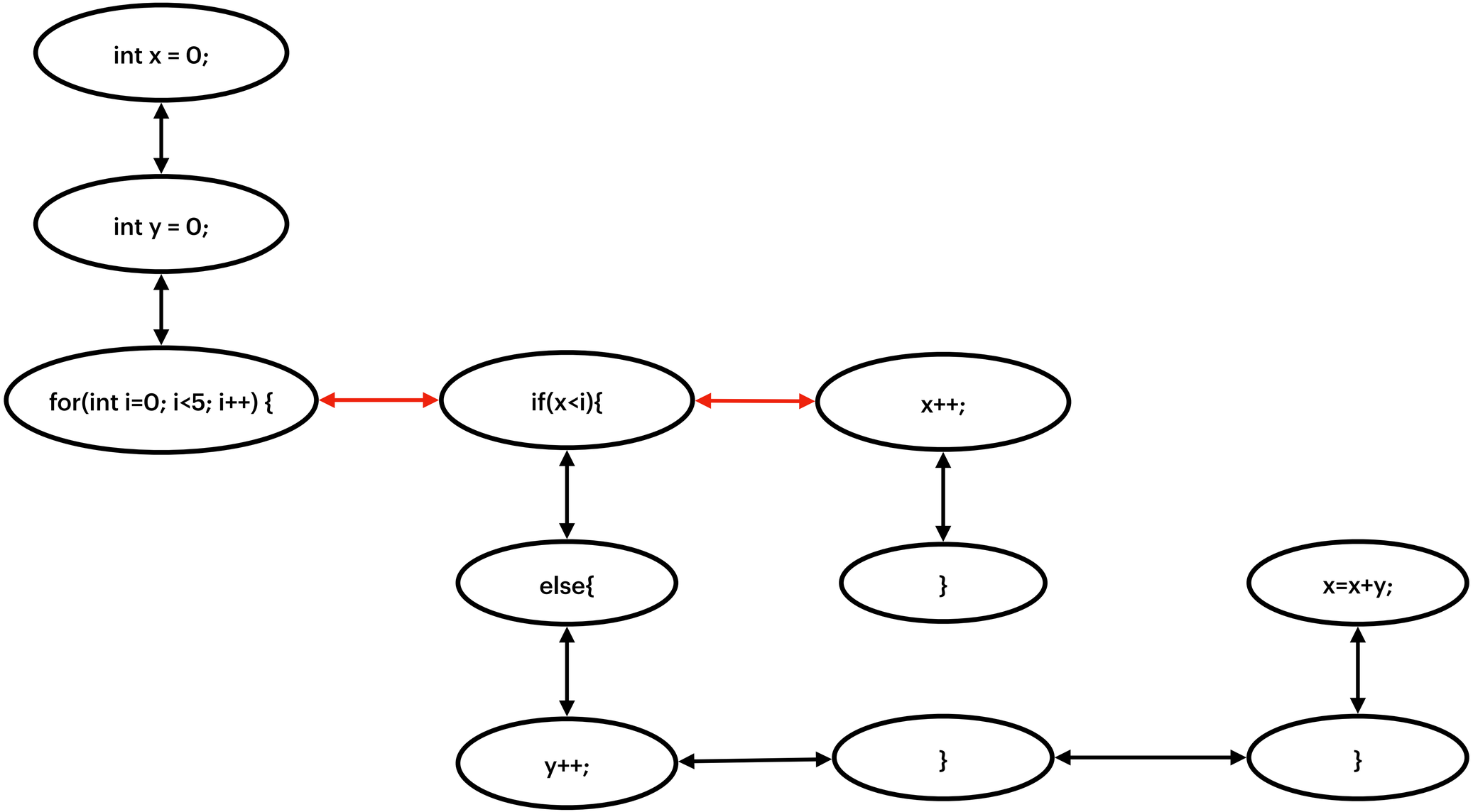
Example: Linked list input -> CDG output



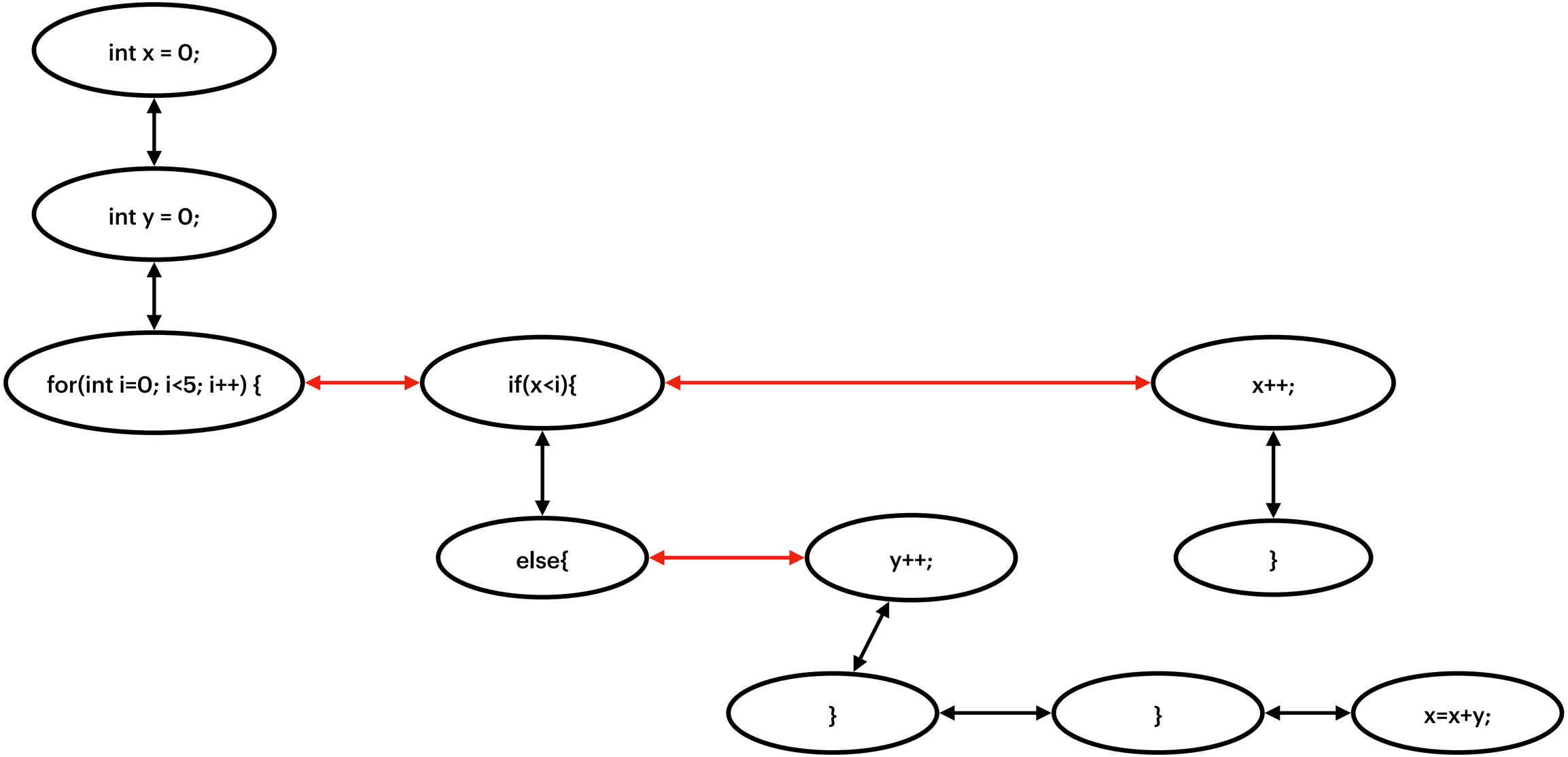
Example: Linked list input -> CDG output



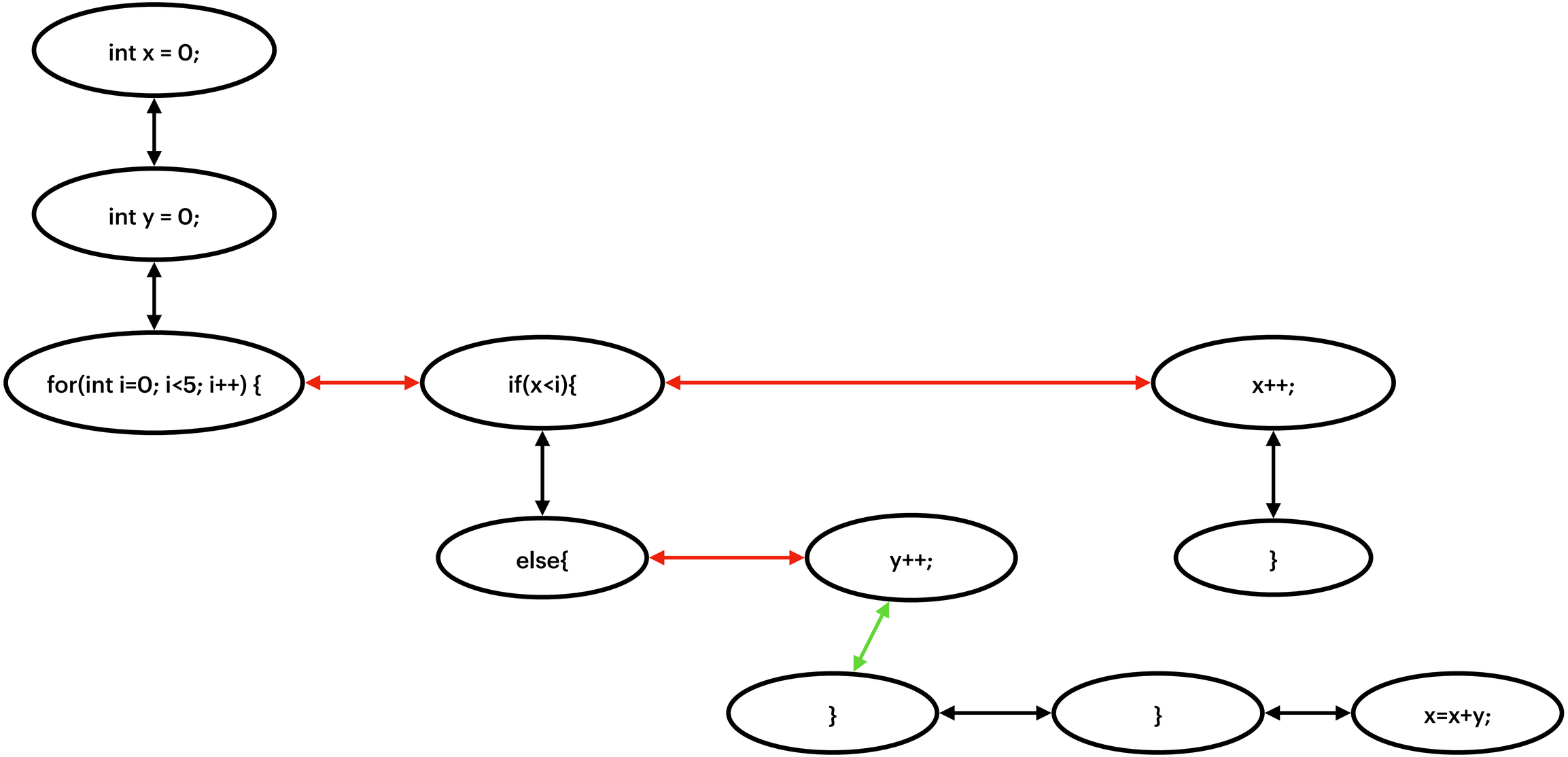
Example: Linked list input -> CDG output



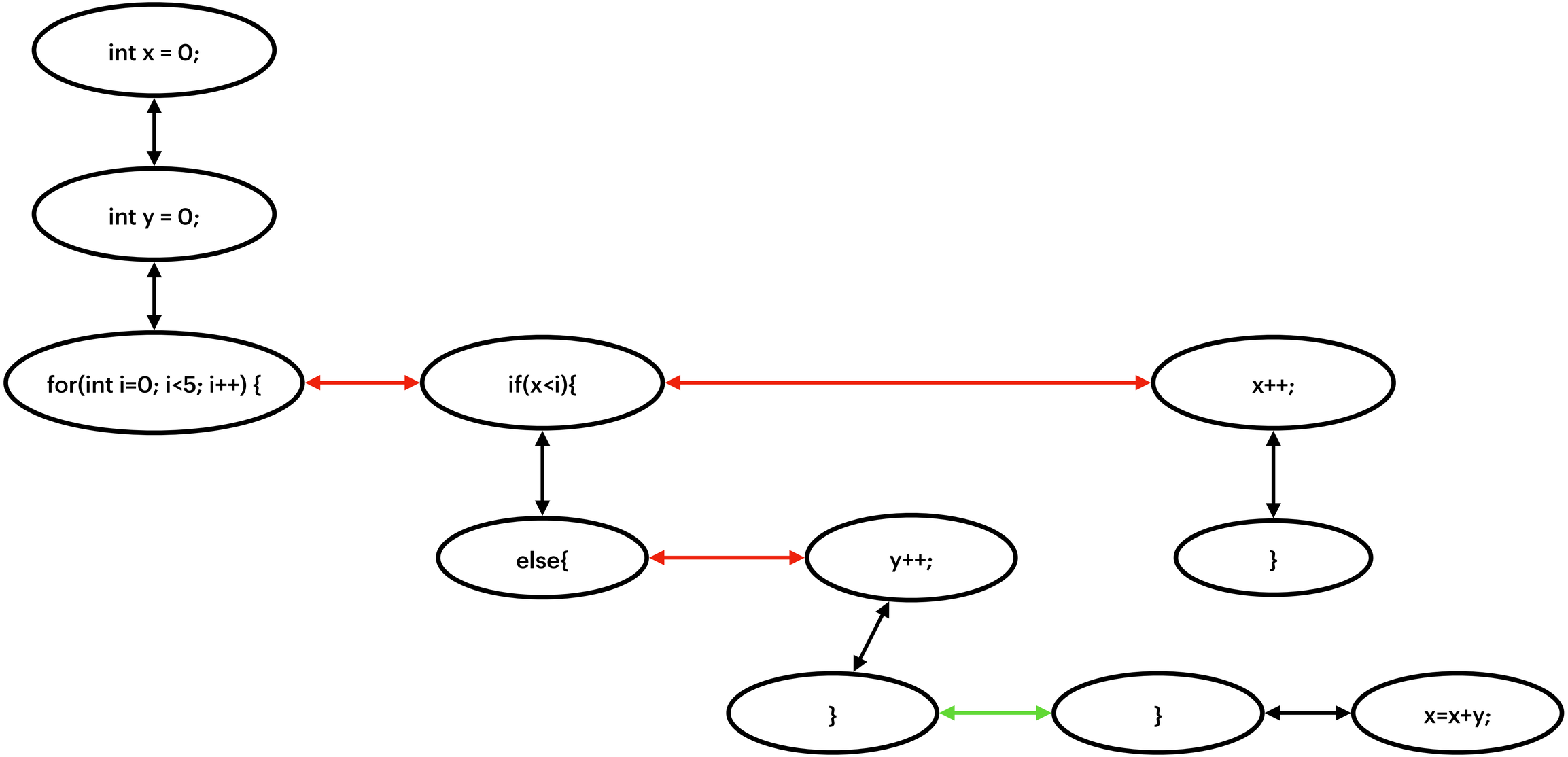
Example: Linked list input -> CDG output



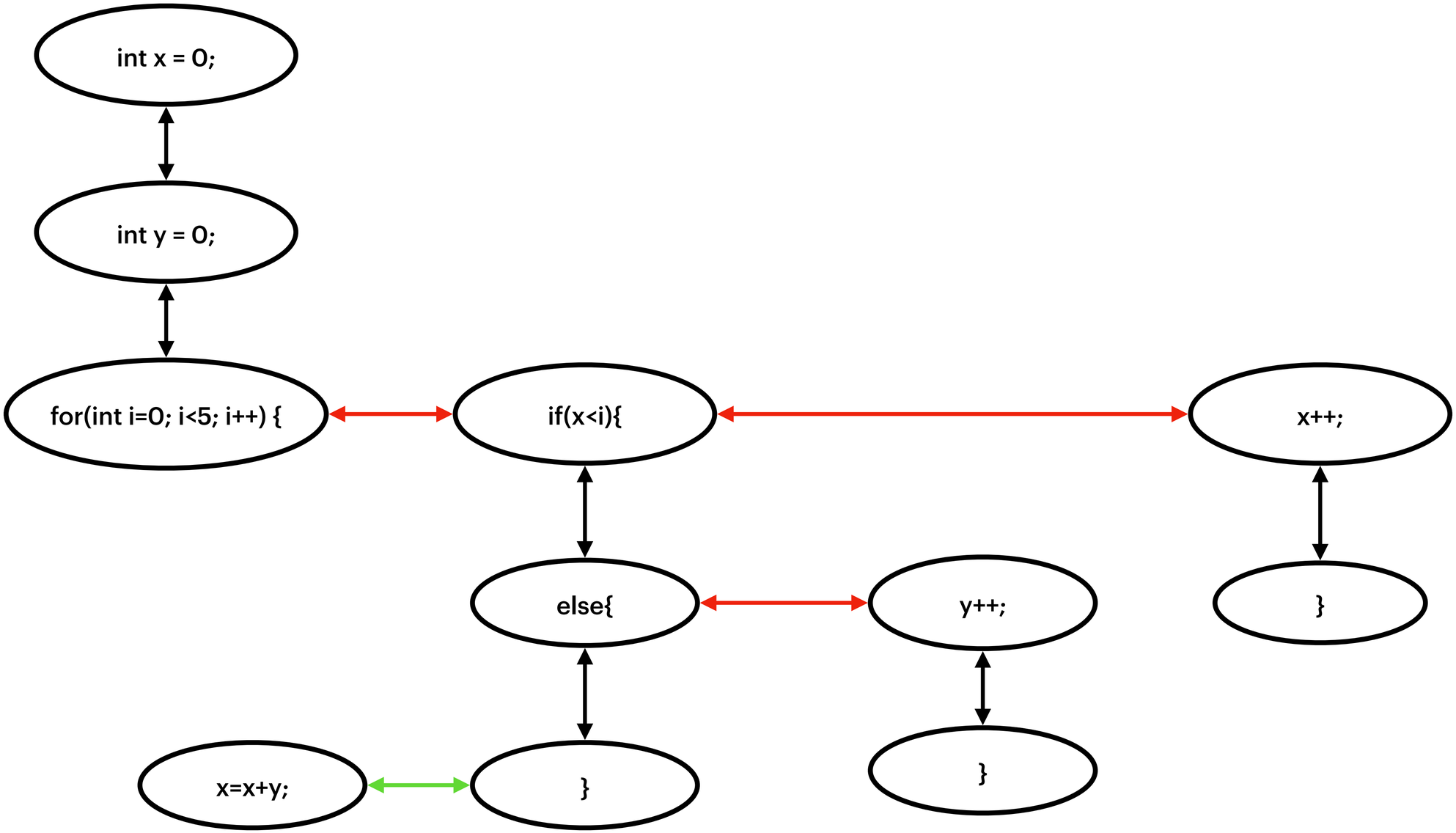
Example: Linked list input -> CDG output



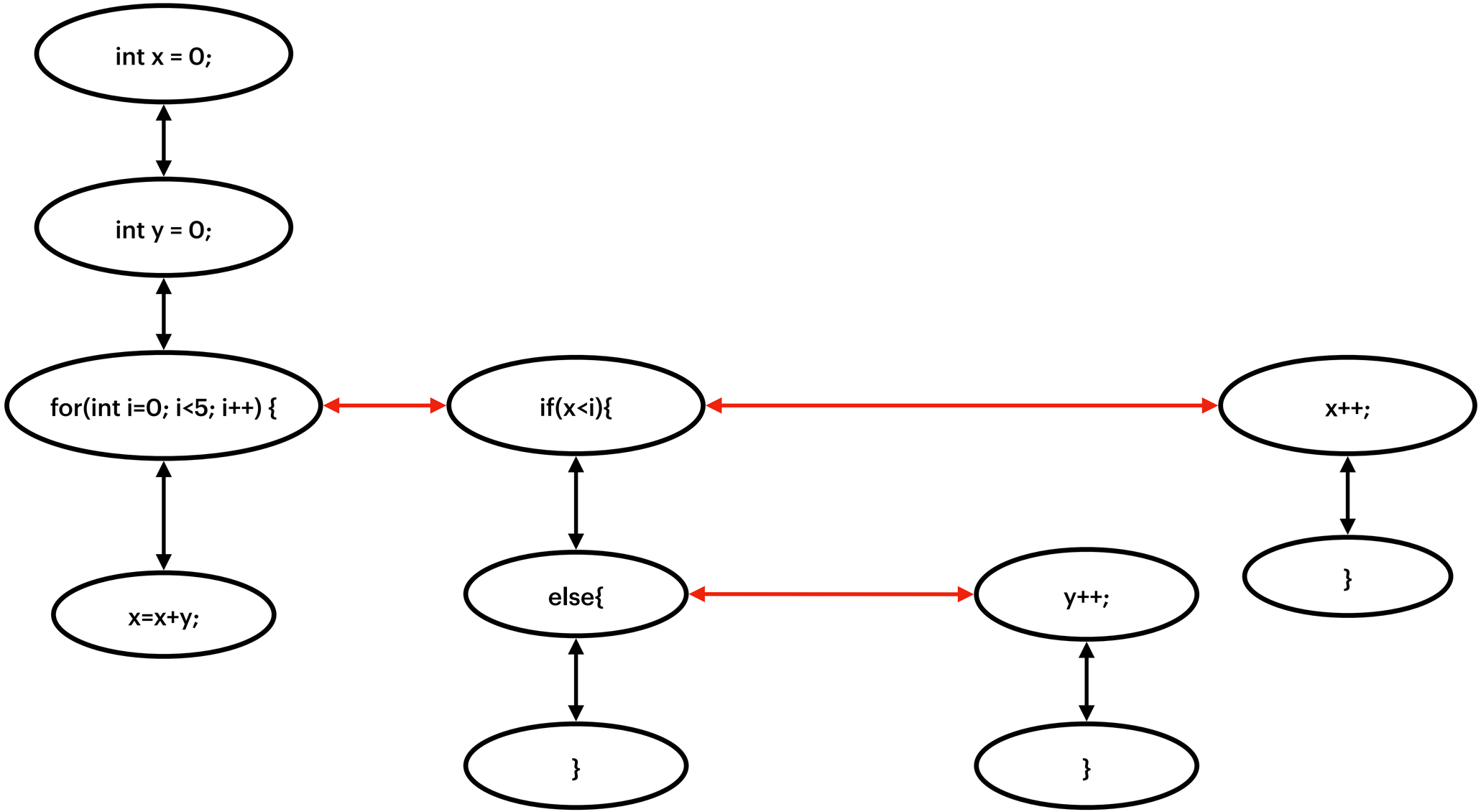
Example: Linked list input -> CDG output



Example: Linked list input -> CDG output



Example: Linked list input -> CDG output



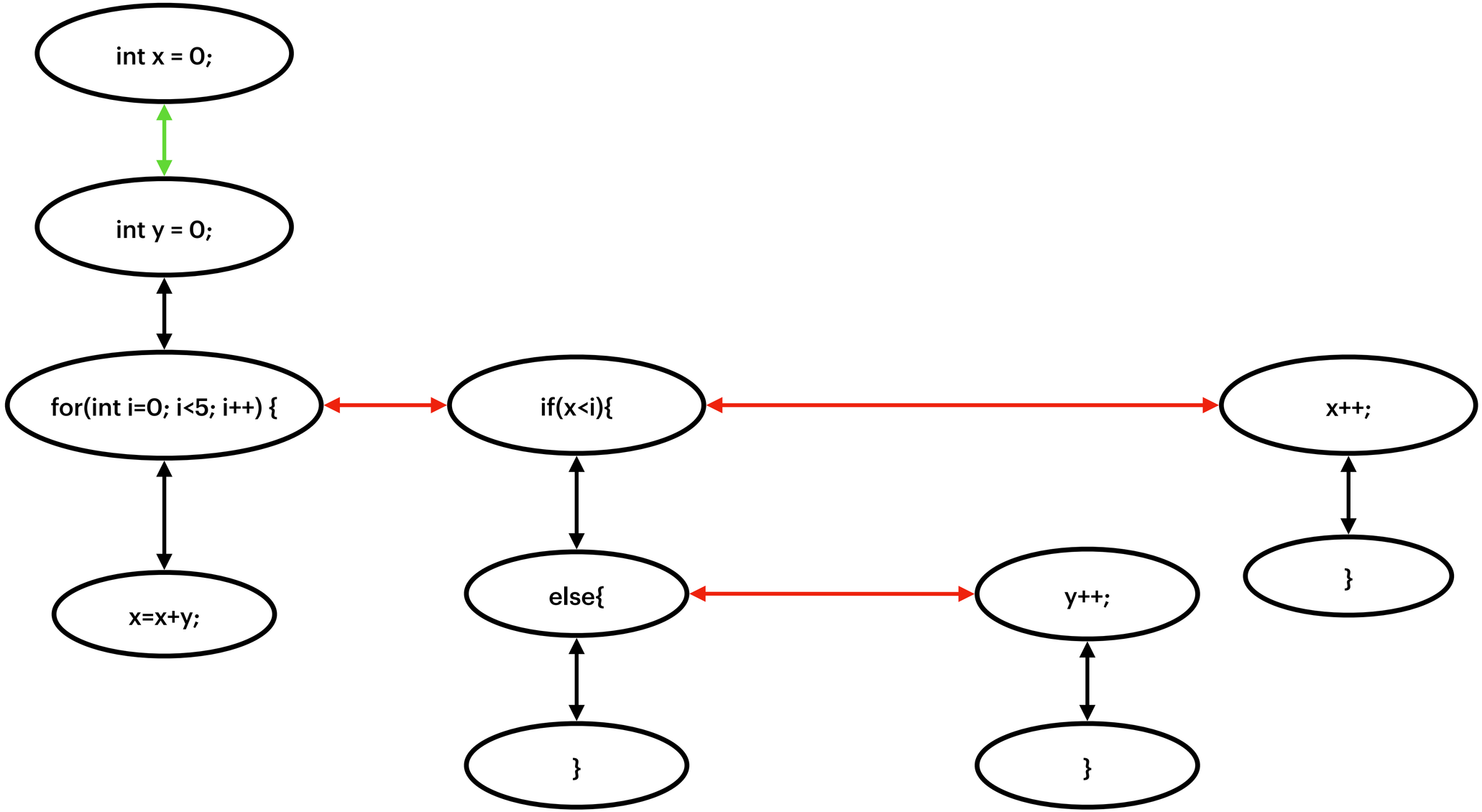
CFG Generator

CFG Generator

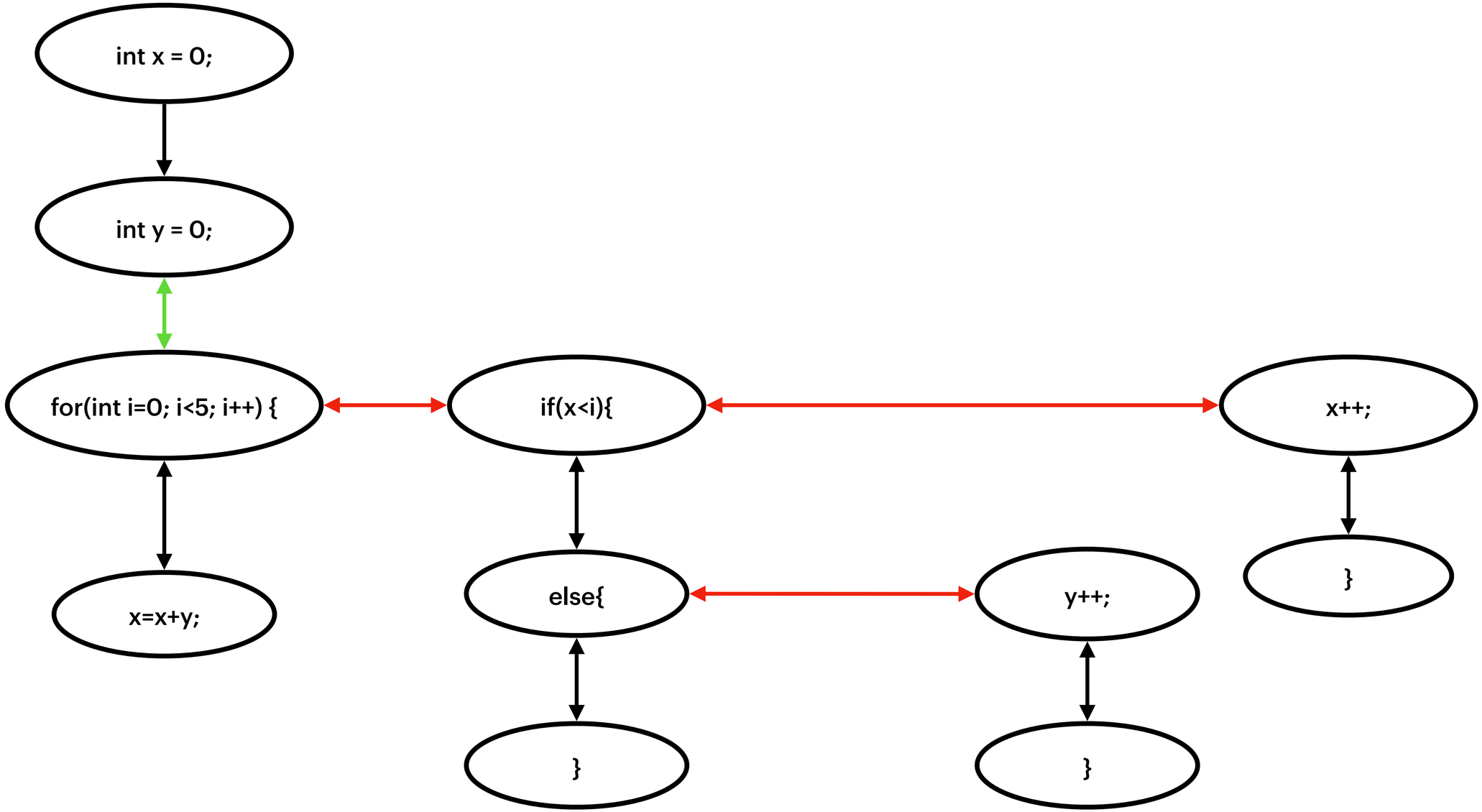
- Takes in input from code diagram generator.
- Uses the parent-child relationships to create flow arrows from each node.
- Example:

```
int x = 0 ;
int y = 0 ;
for(int i=0; i<5;i++) {
  if ( x < i ) {
    x++;
  }
  else {
    y++;
  }
}
x=x+y;
```

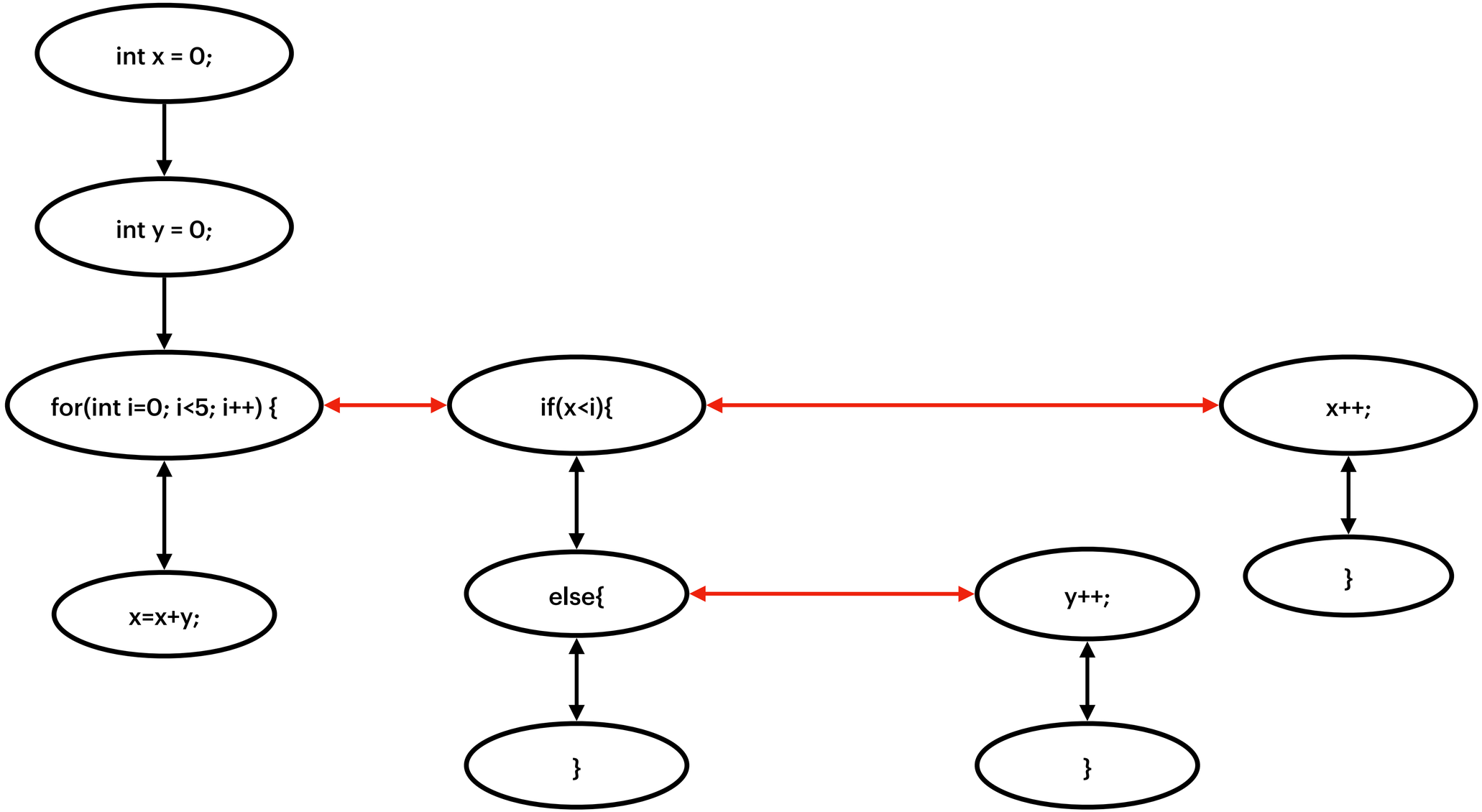
Example: CDG input -> CFG output



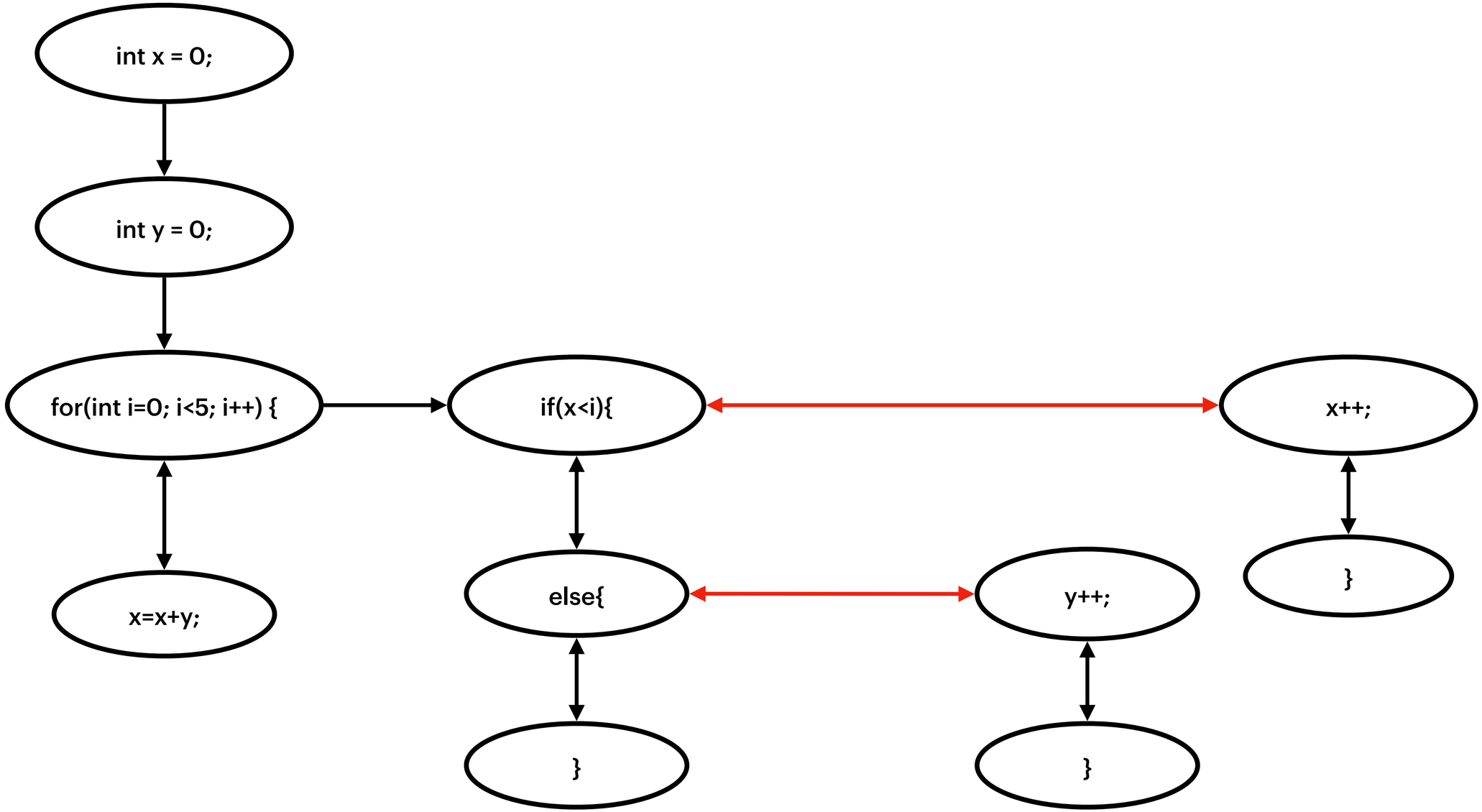
Example: CDG input -> CFG output



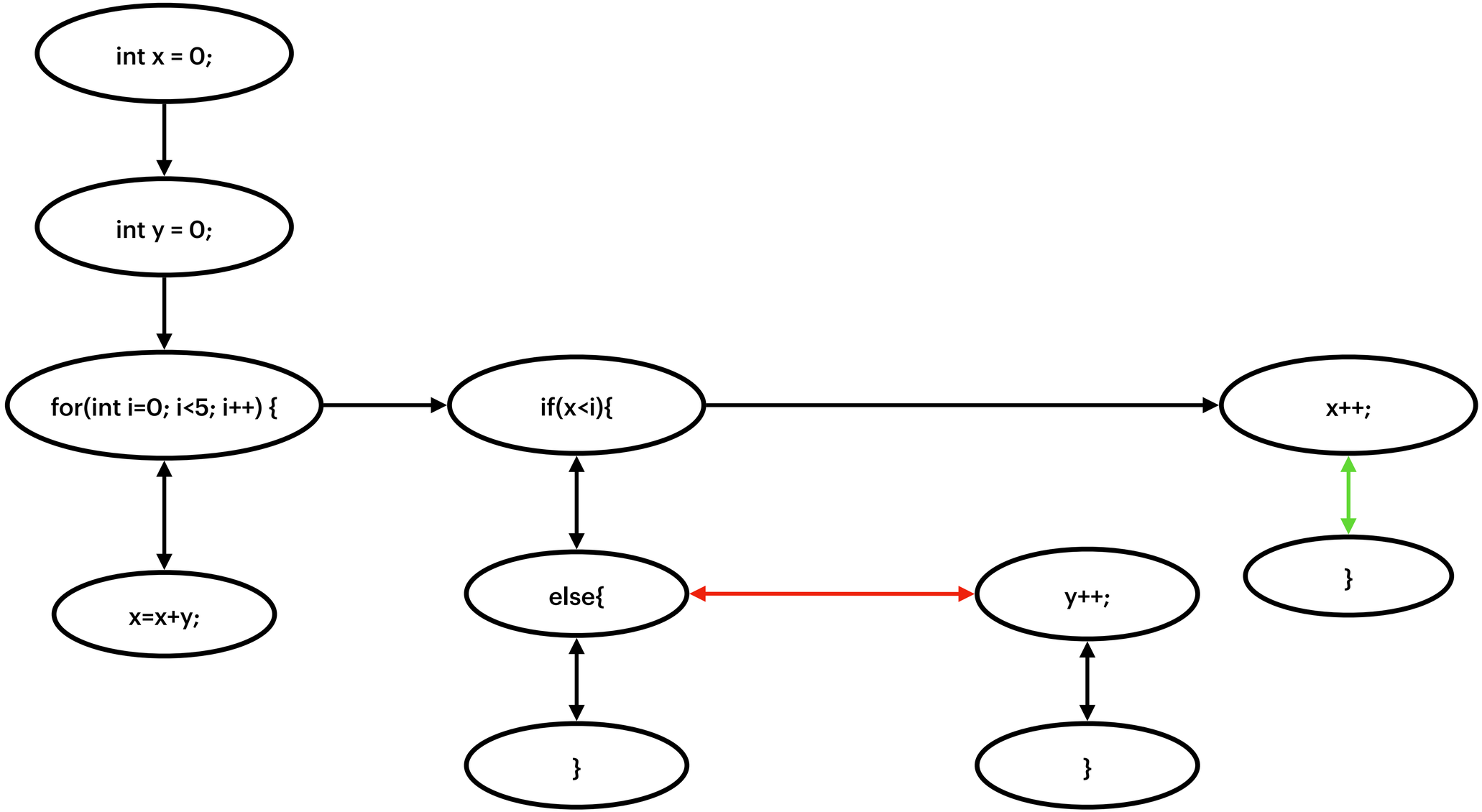
Example: CDG input -> CFG output



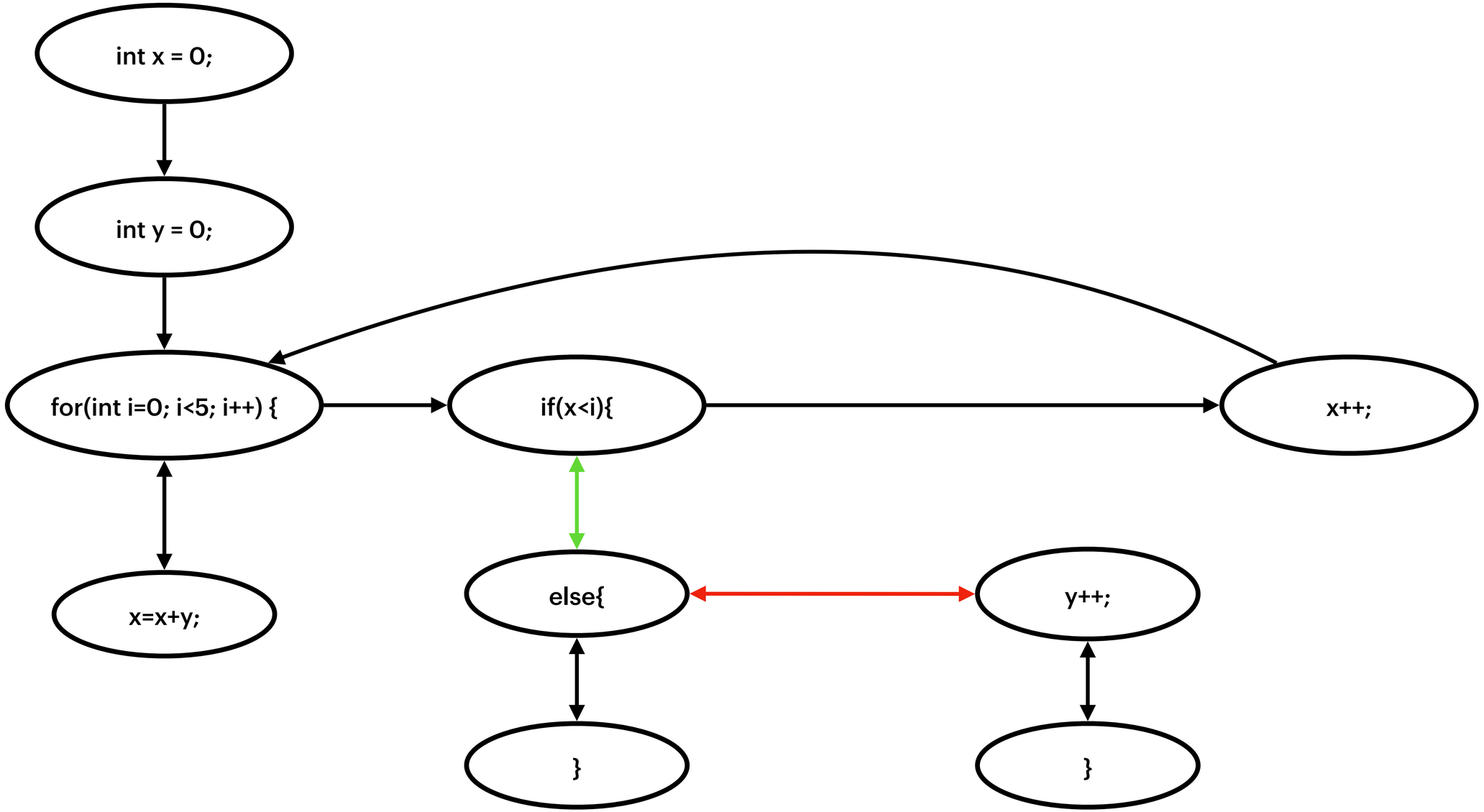
Example: CDG input -> CFG output



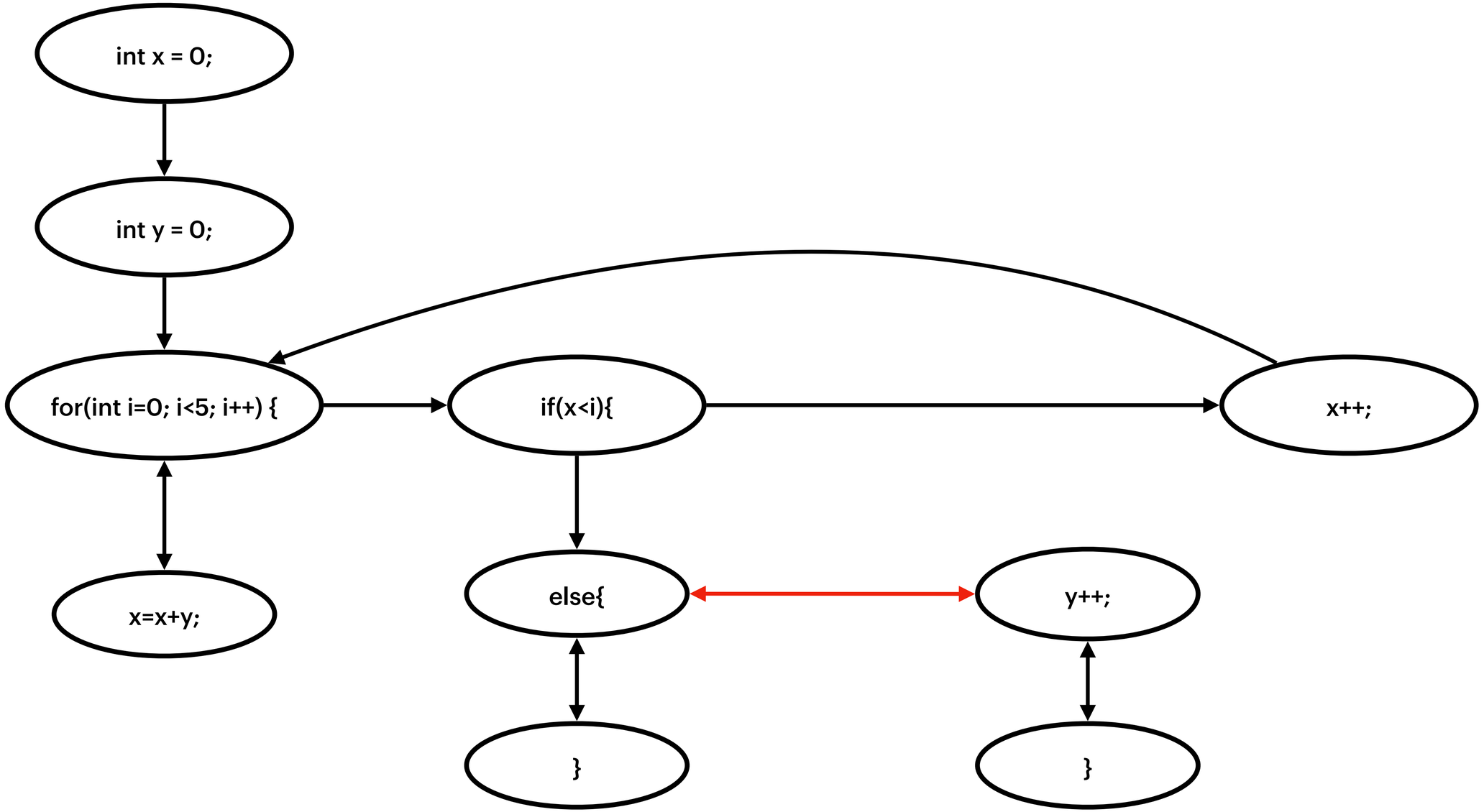
Example: CDG input -> CFG output



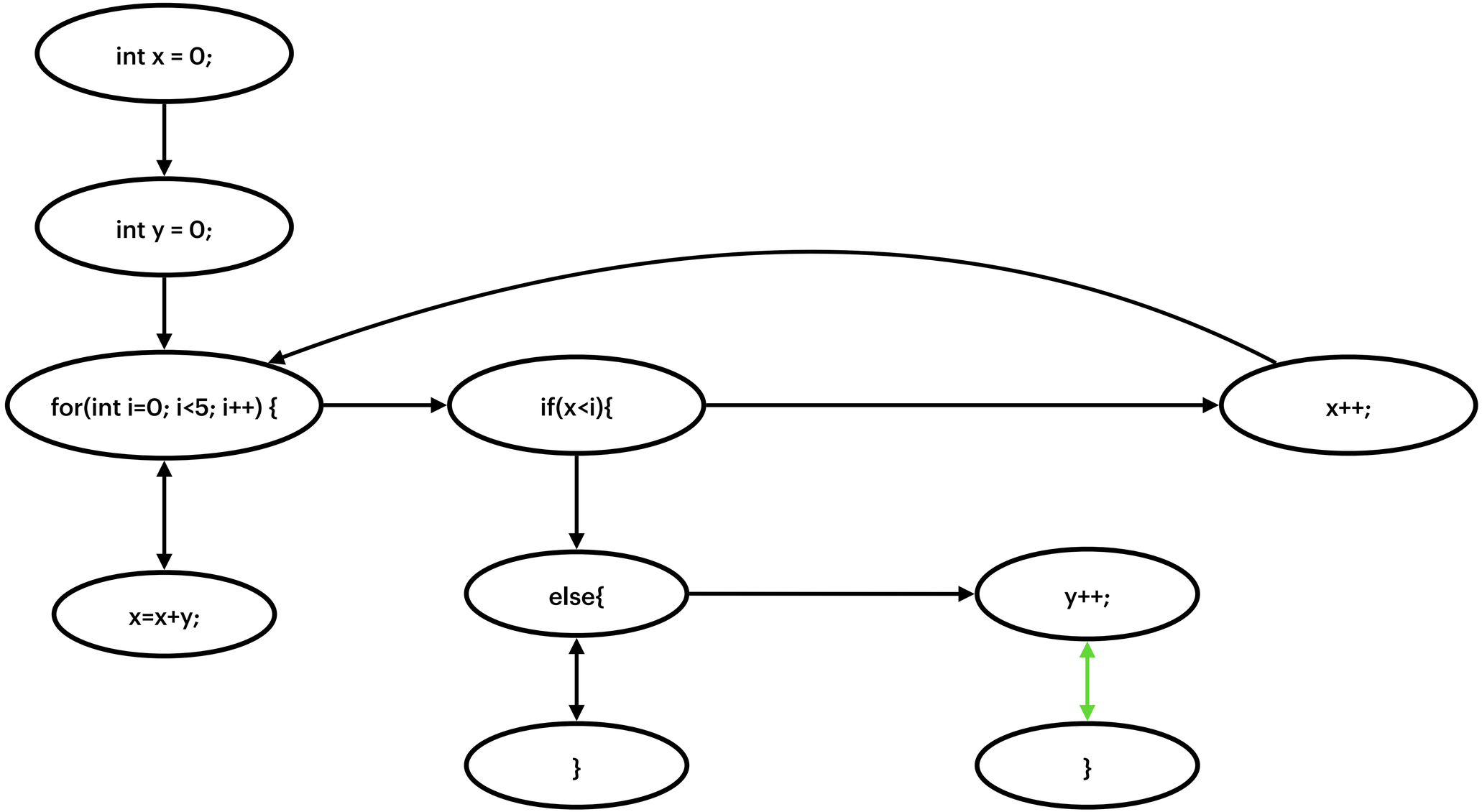
Example: CDG input -> CFG output



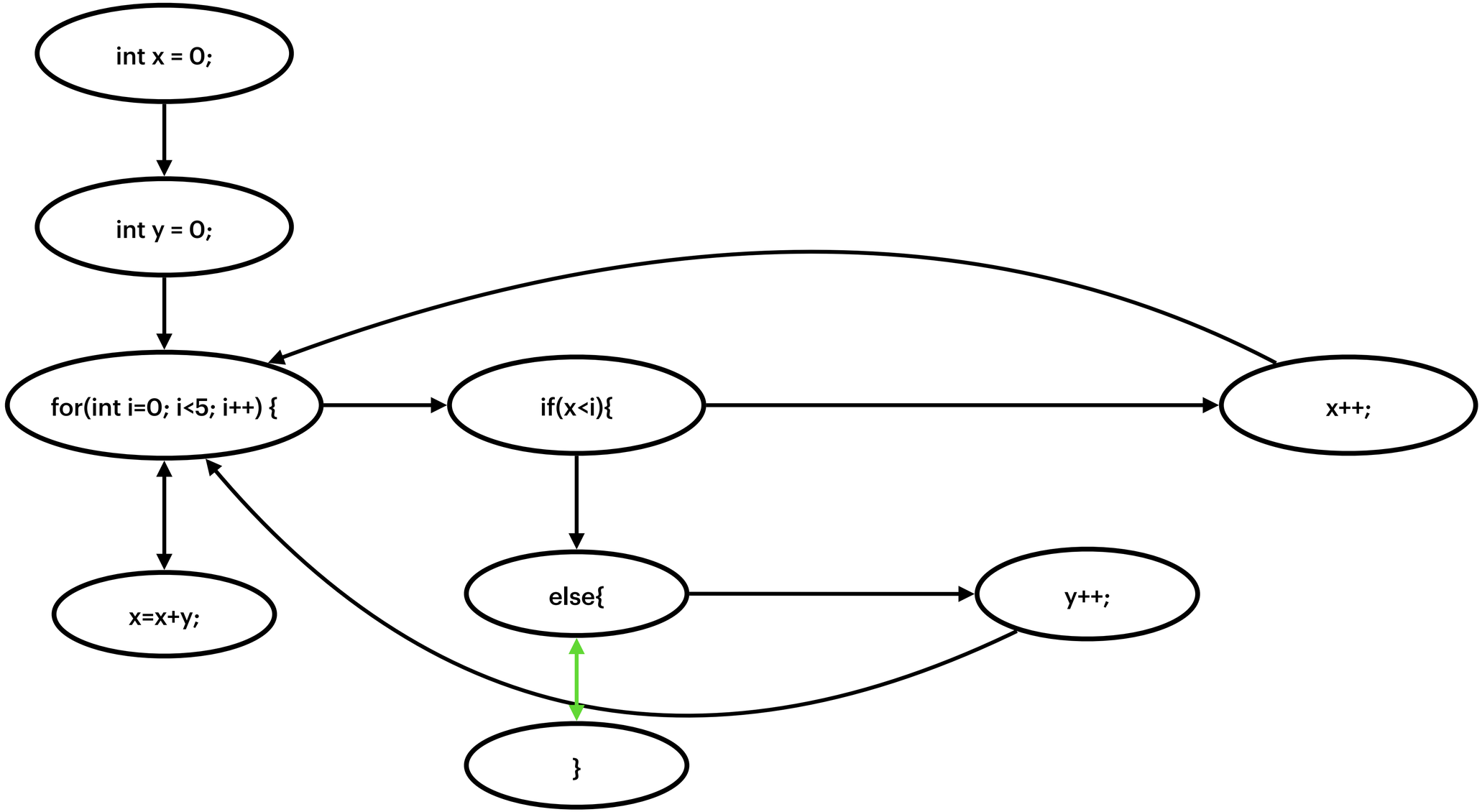
Example: CDG input -> CFG output



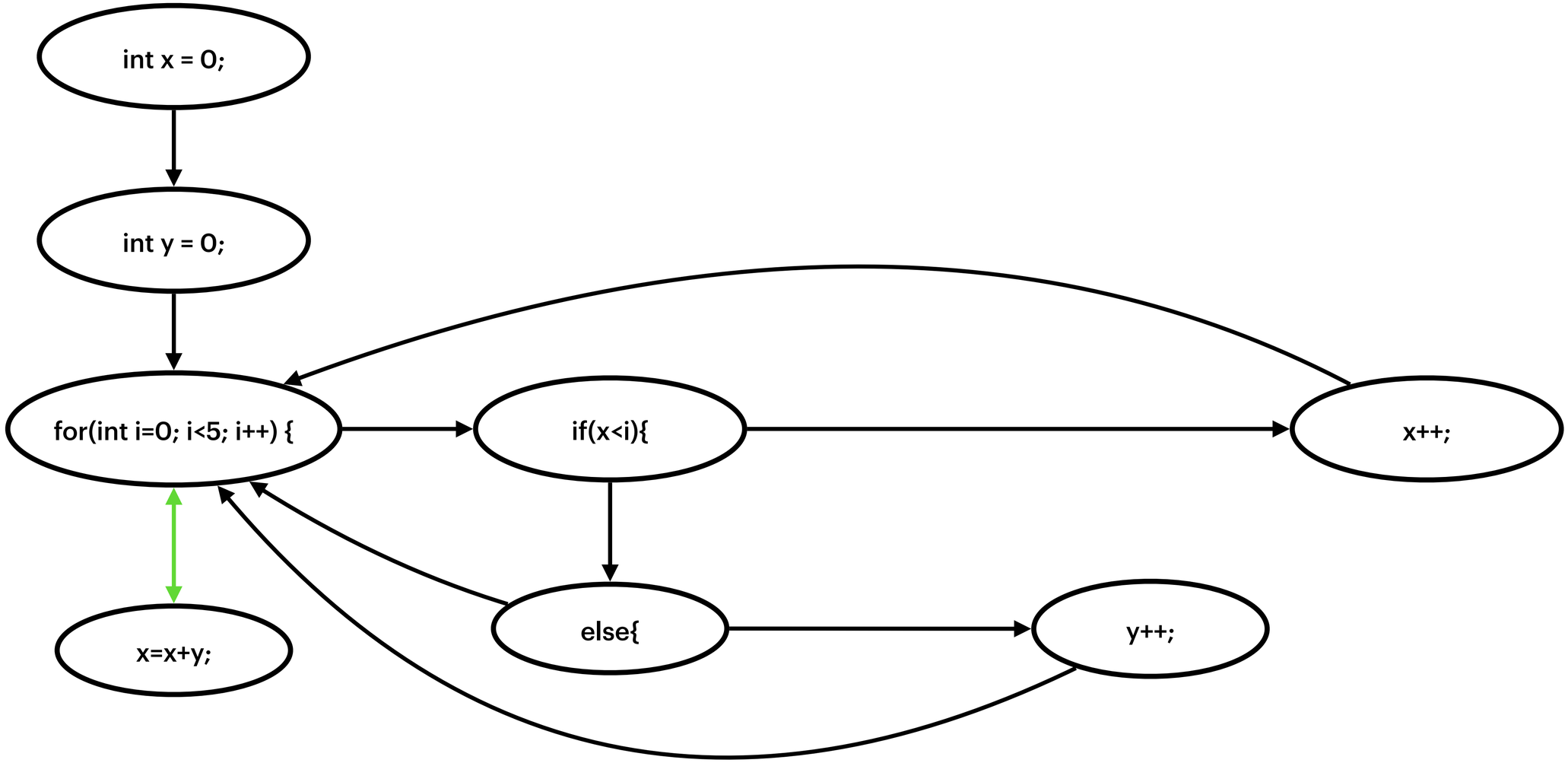
Example: CDG input -> CFG output



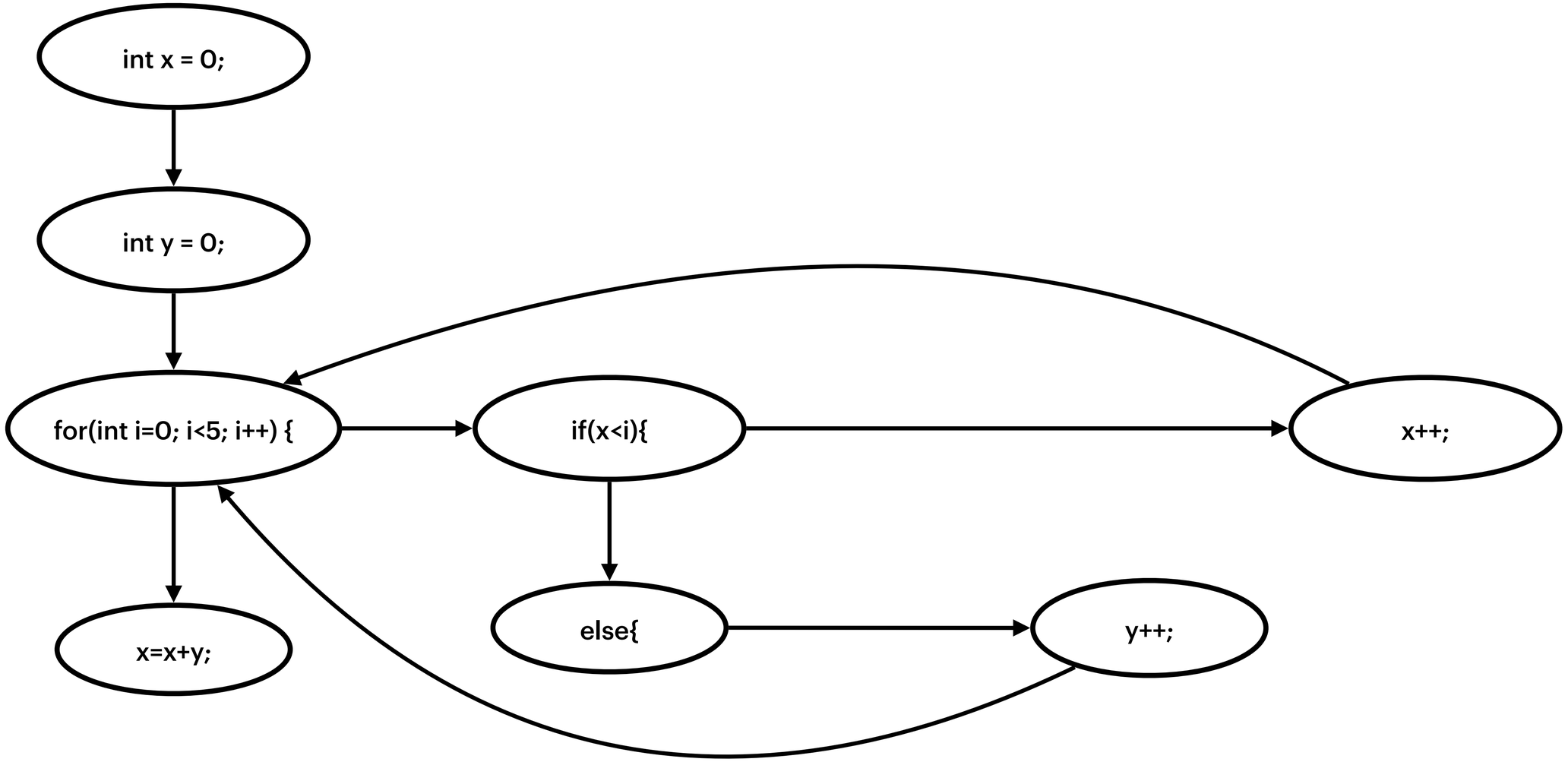
Example: CDG input -> CFG output



Example: CDG input -> CFG output



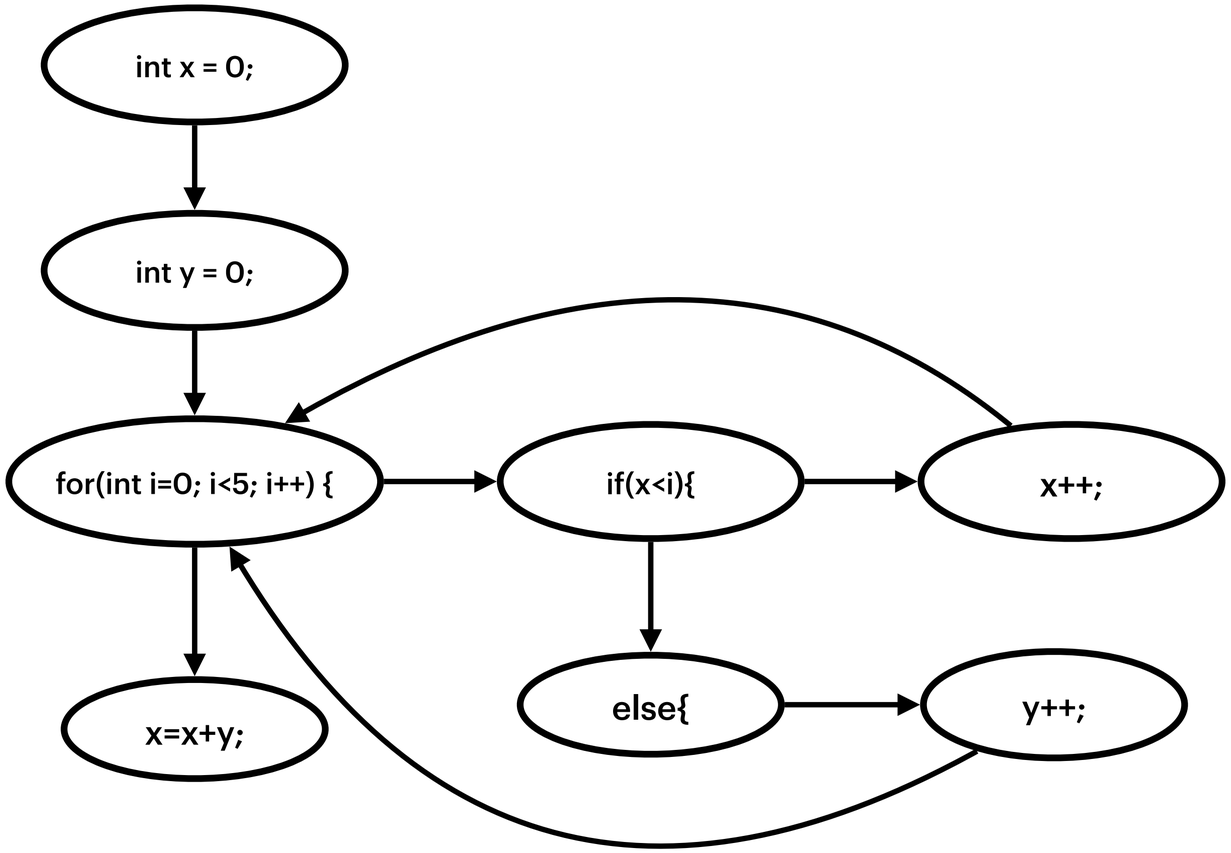
Example: CDG input -> CFG output



Dot file generator

- The previously generated control flow graph is converted into a dot file.
- A->B means that a node with the label "A" points to a node with the label "B" .
- Nodes can be specified before being used (optional).

Dot file generator




```
digraph example {
  8 [label="int x = 0 ;"];
  14 [label="int y = 0 ;"];
  20 [label="for(int i=0; i<5;i++) {"];
  26 [label="if ( x < i ) {"];
  72 [label="x=x+y;"];
  36 [label="x++;"];
  48 [label="else {"];
  54 [label="y++;"];
  66 [label="}"];
  74 [label="}"];
  26 -> 36 [label= "true" color="green" fontcolor="green"];
  26 -> 48 [label= "false" color="red" fontcolor="red"];
  36 -> 20 [label= "end loop"];
  8 -> 14;
  14 -> 20;
  20 -> 26;
  20 -> 72;
  48 -> 54;
  48 -> 66;
  54 -> 20 [label= "end loop"];
  72 -> 74;
}
```

“Within the perfect situation, not everything is perfect.”

Gene Simmons

Cases not handled

- Labels and go-to
- Do-while
- Struct definition
- Function call

Thank you!