Structured Analysis and Structured Design

ST, SST and CFG

By Jesse Ong Pho &
Bjarke D. Larsen

Software Testing

- Software testing is the investigation that is conducted to check a software for errors and abundances or the lack of some important feature or code.

- Test techniques include, but are not limited to, the process of executing a program or application to check and/or find software bugs.

- Software testing is the process of validating and verifying that a software:
  1. Meets the requirements that was set for its design and development
  2. Works as expected.
  3. Can be implemented with the right characteristics.

- Most testing is done after the requirements are set up. These requirements serve as the bar that the software has to be able to serve under.

Software Testing

System Structure Testing

- The objective of software structural testing is to challenge the decisions made by a given program. These tests are done with test cases that are based on the structure and logic of the design and source code.
- Structural testing is done at three levels; unit, integration and system level.
- Structural testing assures the program’s statements and decisions are fully exercised by code execution.
- At the unit level, structural testing should find and eliminate ”dead code” that can not be reached for execution.
- The integration level testing ensures that the different parts of the program works together as a unit.
- The system level testing checks the entire systems and ensures that there is no loose ends.

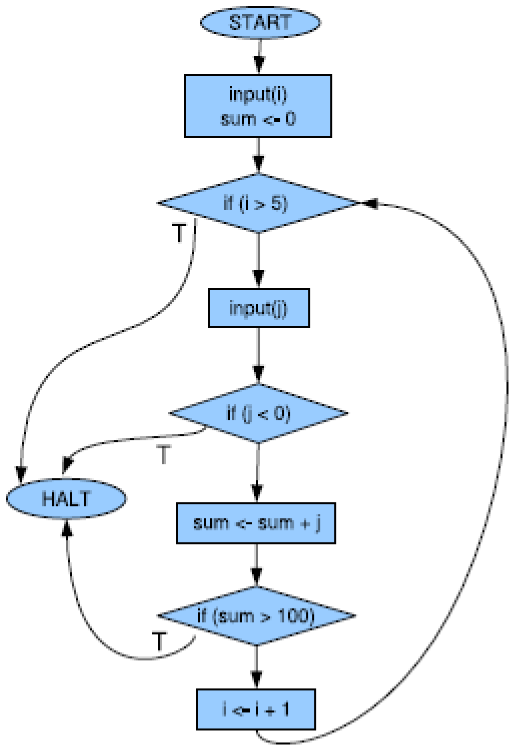
Control Flow Diagram

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 de CFG
- The output is a CFG
- The first block of the CFG is created automatically("Start" block)

Control Flow Diagram



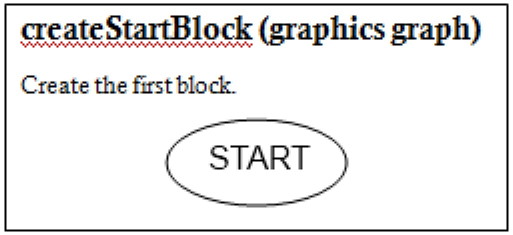
```
START
input(i)
sum := 0
loop : if (i > 5) goto end
input(j)
if (j < 0) goto end
sum := sum + j
if (sum > 100) goto end
i := i +1
goto loop
end : HALT
```

Algorithm

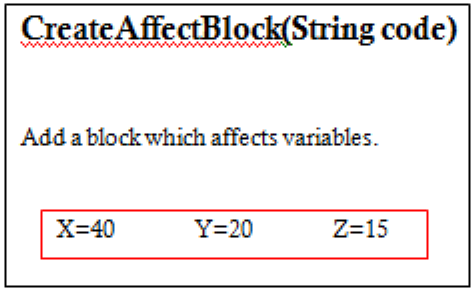
```
Variable
graph graphic;                                        //The output graphic
sourceCode file;
line String;
open(sourceCode);                                     //open the file including the source code
createStartBlock (graph);
While ( sourceCode != EOF ){                          //EOF : End of File
    switch (line){                                    //Analyse of the line
            case Affectation : createAffectBlock(sourceCode);
            Break;
            case Condition : createConditionBlock(sourceCode);
            Break;
            case loop : createConditionBlock(sourceCode, loop);
            break;
    }
    DrawArrow(graph);                                 //Draw the arrow after each blocks

    line = nextline(sourceCode)                       //go to the next line
}
createEndBlock(graph);
close(sourceCode);
End
```

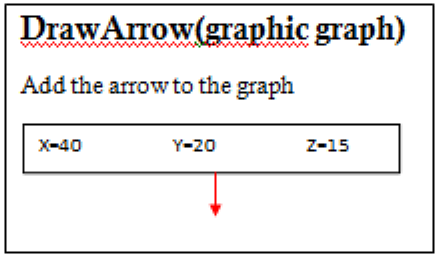
Functions


createStartBlock (graphics graph)

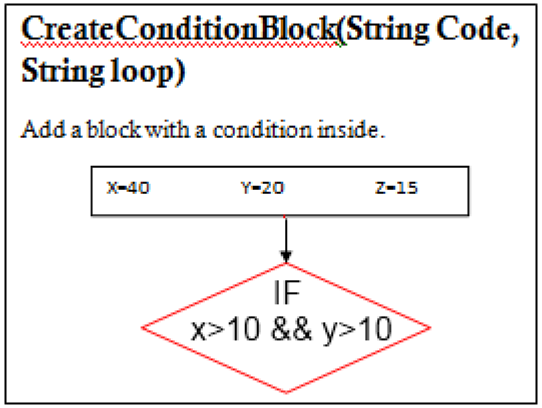
Create the first block.


CreateAffectBlock(String code)

Add a block which affects variables.


DrawArrow(graphic graph)

Add the arrow to the graph


CreateConditionBlock(String Code, String loop)

Add a block with a condition inside.

Bye bye ☺