Exercise 1 (Control- and Data Flow Graphs) The code below is part of a behavioral specification in the hardware description language VHDL.

ENTITY myexample IS
  PORT(in1,in2,in3: IN INTEGER; com: IN BIT; clk: IN BIT; ...);
END myexample;

ARCHITECTURE behavior OF myexample IS BEGIN
  VARIABLE a,b,c,d,e,f,g: INTEGER;
  BEGIN
    WHILE (e > f) LOOP
      WAIT UNTIL clk'event AND (clk = '1');
      IF (com = '1') THEN
        e:=in1;
f:=in1;
a:=d+in2;
b:=c-in3;
g:=a+b;
d:=g+3;
      ELSE
        e:=in2;
f:=in3;
      END IF;
    END LOOP;
  END PROCESS;
END behavior;

The first part starting with the keyword ENTITY describes the inputs and outputs of a hardware block with the name myexample. The relevant inputs are in1, in2, in3 and com. The second part starting with the keyword ARCHITECTURE is the behavioral description of the hardware block and consists of one sequential PROCESS. The process defines the local variables a, b, c, d, e, f, g and executes a loop. Construct the

- Control flow graph (CFG) for the process
- Data flow graph (DFG) for the THEN and ELSE branches in the loop
Exercise 2 (Sequence Graphs) The following program (in C-like syntax) reads an array of \( n \) unsorted numbers \((a[0]...a[n-1])\), sorts the array by \textit{bubble-sort}, and outputs the sorted array. The function \texttt{swap} is called to exchange two array elements.

```c
bubble_sort {
    // read unsorted array a[0]...a[n-1]
    ...
    // sort array by bubble-sort
    do {
        sorted = TRUE;
        for (i=0; i<n-1; i++)
            if (a[i] > a[i+1]) {
                swap(&(a[i]), &(a[i+1]));
                sorted = FALSE;
            }
    } while (!sorted);
    // write sorted array a[0]...a[n-1]
    ...
}
swap(float *a, float *b) {
    float temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

Specify the control/data flow for this program by means of a hierarchical sequence graph.