Aufgabe 1 Sortieren dreier Zahlen

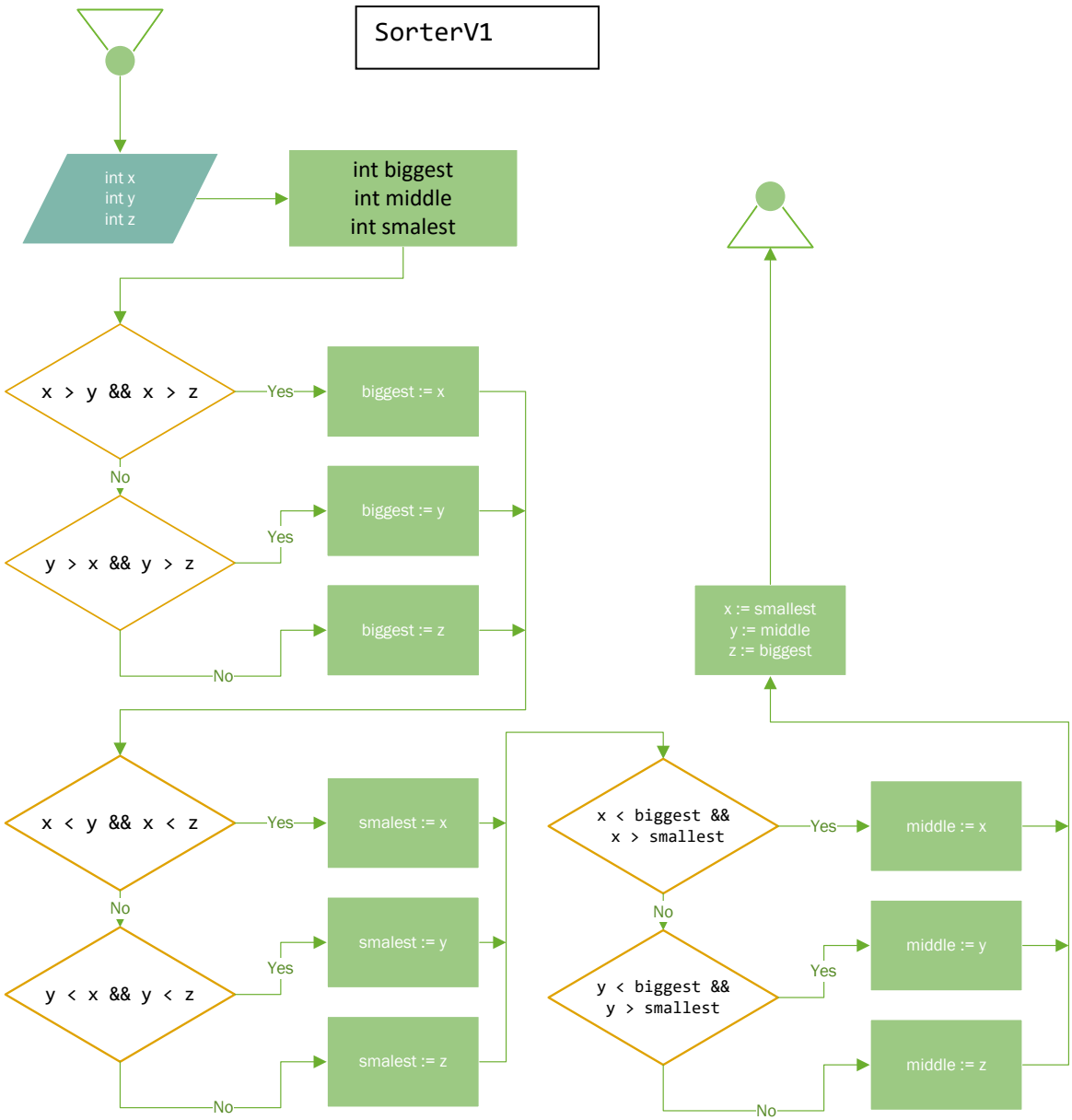
a) Pseudocode

```
1.  sorterV1(↓↑integer x, ↓↑integer y, ↓↑integer z)
2.      integer biggest
3.      integer middle
4.      integer smallest
5.
6.
7.      if (x > y && x > z) then
8.          biggest := x
9.      else
10.         if (y > x && y > z) then
11.             biggest := y
12.         else
13.             biggest := z
14.         end
15.     end
16.
17.     if (x < y && x < z) then
18.         smallest := x
19.     else
20.         if (y < x && y < z) then
21.             smallest := y
22.         else
23.             smallest := z
24.         end
25.     end
26.
27.     if (x < biggest && x > smallest) then
28.         middle := x
29.     else
30.         if (y < biggest && y > smallest) then
31.             middle := y
32.         else
33.             middle := z
34.         end
35.     end
36.
37.     x := smallest
38.     y := middle
39.     z := biggest
40.
41. end
```

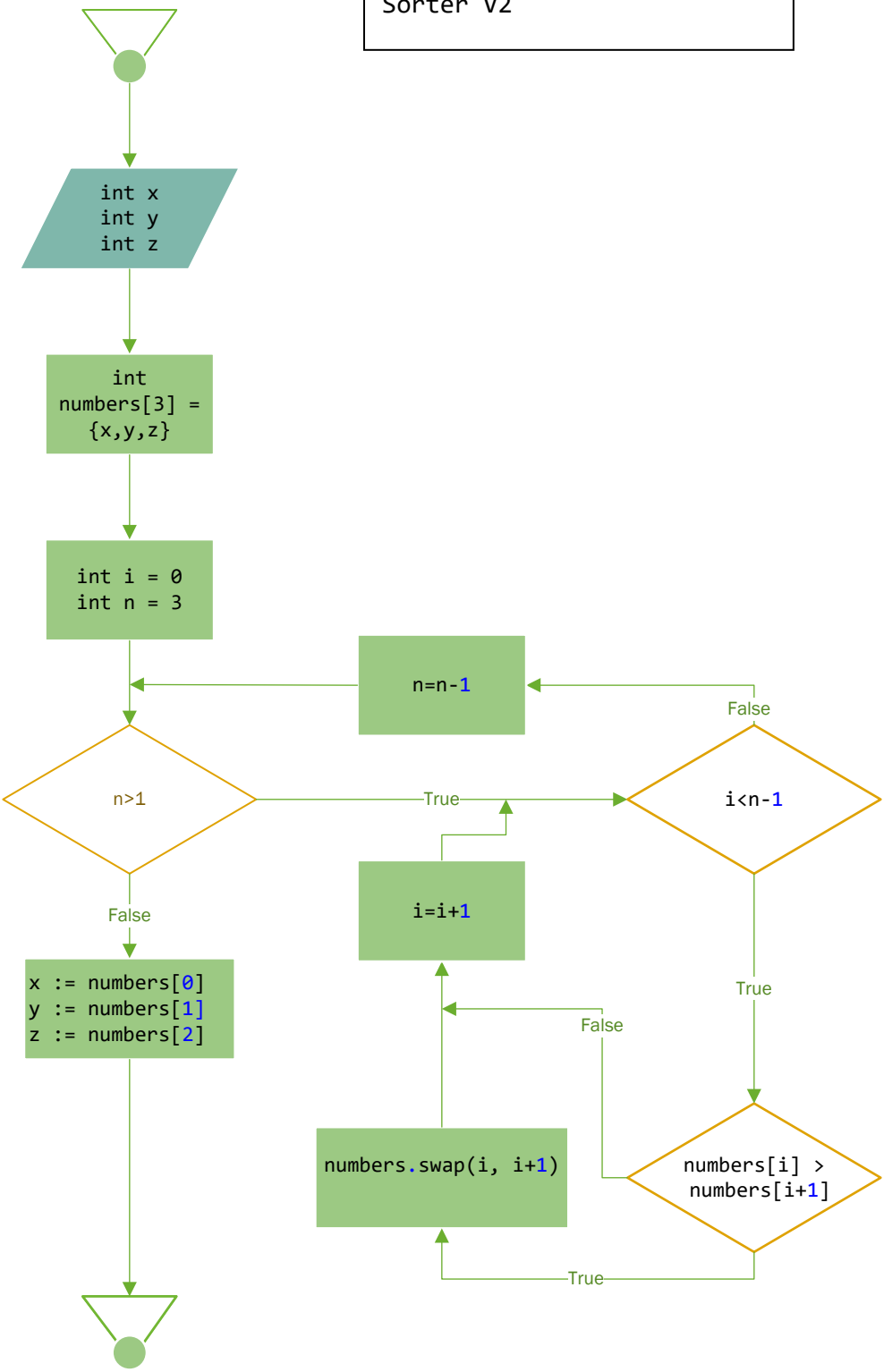
```
1.  sorterV2(↓↑integer x, ↓↑integer y, ↓↑integer z)
2.    integer numbers[3] = {x,y,z}
3.
4.    for (n=3; n>1; n=n-1)do
5.      for (i=0; i<n-1; i=i+1)do
6.        if (numbers[i] > numbers[i+1])then
7.          numbers.swap(i, i+1)
8.        end
9.      end
10.   end
11.
12.   x := numbers[0]
13.   y := numbers[1]
14.   z := numbers[2]
15.   end
```

```
1.  sorterV3(↓↑integer x, ↓↑integer y, ↓↑integer z)
2.      if(y > x) then
3.          Swap(x,y)
4.      end
5.
6.      if (z > y) then
7.          Swap(y,z)
8.      end
9.
10.     if (y > x) then
11.         Swap(x,y)
12.     end
13. end
```

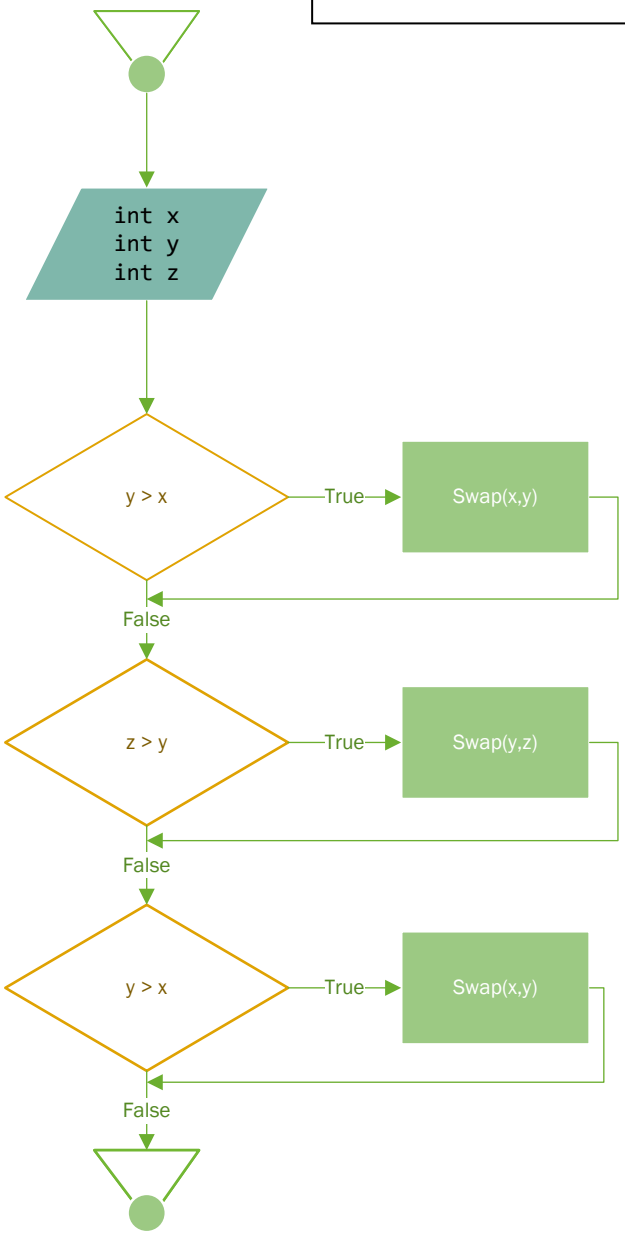
b) Ablaufdiagramm



Sorter V2


int x
int y
int z
int numbers[3] = {x,y,z}
int i = 0
int n = 3
n>1
True
False
n=n-1
i<n-1
False
True
i=i+1
False
True
x := numbers[0]
y := numbers[1]
z := numbers[2]
numbers.swap(i, i+1)
numbers[i] > numbers[i+1]

Sorter V3


int x
int y
int z
y > x
True
Swap(x,y)
False
z > y
True
Swap(y,z)
False
y > x
True
Swap(x,y)
False

Aufgabe 2 Palindrom

```
1.  Palindromtest(↓string text, ↑bool result)
2.
3.      result := true
4.      integer n = text.length() - 1
5.      integer i = 0
6.
7.      while (result == true and i < n) do
8.          if(text[n] != text[i]) then
9.              result := false
10.         end
11.
12.         i := i+1
13.         n := n-1
14.     end
15. end
```

Aufgabe 3 Häufigkeitsanalyse

```
1.  VowelOccurencies(↓string text, ↑integer freqA, ↑integer freqE, ↑integer freqI, ↑inte
    ger freqO, ↑integer freqU)
2.
3.      integer i = 0
4.      integer textlength = text.length()
5.      integer Vowels[5] = {0,0,0,0,0}
6.
7.      while (i < textlength) do
8.          if (text[i] == 'A' || text[i] == 'a') then
9.              Vowels[0] = Vowels[0] +1
10.         end
11.         if (text[i] == 'E' || text[i] == 'e') then
12.             Vowels[1] = Vowels[1] +1
13.         end
14.         if (text[i] == 'I' || text[i] == 'i') then
15.             Vowels[2] = Vowels[2] +1
16.         end
17.         if (text[i] == 'O' || text[i] == 'o') then
18.             Vowels[3] = Vowels[3] +1
19.         end
20.         if (text[i] == 'U' || text[i] == 'u') then
21.             Vowels[4] = Vowels[4] +1
22.         end
23.         i = i + 1
24.     end
25.
26.     freqA := (Vowels[0] * 100) / textlength
27.     freqE := (Vowels[1] * 100) / textlength
28.     freqI := (Vowels[2] * 100) / textlength
29.     freqO := (Vowels[3] * 100) / textlength
30.     freqU := (Vowels[4] * 100) / textlength
31.
32. end
```

text = "Ich bin ein Test" textlength = 16

A	E	I	O	U	i	Überprüfung	Befehl
0	0	0	0	0	0	i < textlength	-
0	0	0	0	0	0	└ TRUE	-
0	0	0	0	0	0	Check Vowels	-
0	0	1	0	0	0	└ TRUE	Vowels[2] + 1
0	0	1	0	0	1	i < textlength	i + 1
0	0	1	0	0	1	└ TRUE	-
0	0	1	0	0	1	Check Vowels	-
0	0	1	0	0	2	i < textlength	i + 1
0	0	1	0	0	2	└ TRUE	-
0	0	1	0	0	2	Check Vowels	-
0	0	1	0	0	3	i < textlength	i + 1
0	0	1	0	0	3	└ TRUE	-
0	0	1	0	0	3	Check Vowels	-
0	0	1	0	0	4	i < textlength	i + 1
0	0	1	0	0	4	└ TRUE	-
0	0	1	0	0	4	Check Vowels	-
0	0	1	0	0	5	i < textlength	i + 1
0	0	1	0	0	5	└ TRUE	-
0	0	1	0	0	5	Check Vowels	-
0	0	2	0	0	5	└ TRUE	Vowels[2] + 1
0	0	2	0	0	6	i < textlength	i + 1
0	0	2	0	0	6	└ TRUE	-
0	0	2	0	0	6	Check Vowels	-
0	0	2	0	0	7	i < textlength	i + 1
0	0	2	0	0	7	└ TRUE	-
0	0	2	0	0	7	Check Vowels	-
0	0	2	0	0	8	i < textlength	i + 1
0	0	2	0	0	8	└ TRUE	-
0	0	2	0	0	8	Check Vowels	-
0	1	2	0	0	8	└ TRUE	Vowels[1] + 1
0	1	2	0	0	9	i < textlength	i + 1
0	1	2	0	0	9	└ TRUE	-
0	1	2	0	0	9	Check Vowels	-
0	1	3	0	0	9	└ TRUE	Vowels[2] + 1
0	1	3	0	0	10	i < textlength	i + 1
0	1	3	0	0	10	└ TRUE	-
0	1	3	0	0	10	Check Vowels	-
0	1	3	0	0	11	i < textlength	i + 1
0	1	3	0	0	11	└ TRUE	-
0	1	3	0	0	11	Check Vowels	-
0	1	3	0	0	12	i < textlength	i + 1
0	1	3	0	0	12	└ TRUE	-
0	1	3	0	0	12	Check Vowels	-
0	1	3	0	0	13	i < textlength	i + 1
0	1	3	0	0	13	└ TRUE	-
0	1	3	0	0	13	Check Vowels	-
0	2	3	0	0	13	└ TRUE	Vowels[1] + 1
0	2	3	0	0	14	i < textlength	i + 1
0	2	3	0	0	14	└ TRUE	-
0	2	3	0	0	14	Check Vowels	-
0	2	3	0	0	15	i < textlength	i + 1
0	2	3	0	0	15	└ TRUE	-
0	2	3	0	0	15	Check Vowels	-
0	2	3	0	0	16	i < textlength	i + 1
0	2	3	0	0	16	└ FALSE	-

freqA	freqE	freqI	freqO	freqU	-	-	-
0	0	0	0	0	-	-	(Vowels[0] * 100) / (textlength)
0	12	0	0	0	-	-	(Vowels[1] * 100) / (textlength)
0	12	18	0	0	-	-	(Vowels[2] * 100) / (textlength)
0	12	18	0	0	-	-	(Vowels[3] * 100) / (textlength)
0	12	18	0	0	-	-	(Vowels[4] * 100) / (textlength)

text = "AAII" textlength = 4

A	E	I	O	U	i	Überprüfung	Befehl
0	0	0	0	0	0	i < textlength	-
0	0	0	0	0	0	└ TRUE	-
0	0	0	0	0	0	Check Vowels	-
1	0	0	0	0	0	└ TRUE	Vowels[0] + 1
1	0	0	0	0	1	i < textlength	i + 1
1	0	0	0	0	1	└ TRUE	-
1	0	0	0	0	1	Check Vowels	-
2	0	0	0	0	1	└ TRUE	Vowels[0] + 1
2	0	0	0	0	2	i < textlength	i + 1
2	0	0	0	0	2	└ TRUE	-
2	0	0	0	0	2	Check Vowels	-
2	0	1	0	0	2	└ TRUE	Vowels[2] + 1
2	0	1	0	0	3	i < textlength	i + 1
2	0	1	0	0	3	└ TRUE	-
2	0	1	0	0	3	Check Vowels	-
2	0	2	0	0	3	└ TRUE	Vowels[2] + 1
2	0	2	0	0	4	i < textlength	i + 1
2	0	2	0	0	4	└ FALSE	-
freqA	**freqE**	**freqI**	**freqO**	**freqU**	-	-	-
50	0	0	0	0	-	-	(Vowels[0] * 100) / (textlength)
50	0	0	0	0	-	-	(Vowels[1] * 100) / (textlength)
50	0	50	0	0	-	-	(Vowels[2] * 100) / (textlength)
50	0	50	0	0	-	-	(Vowels[3] * 100) / (textlength)
50	0	50	0	0	-	-	(Vowels[4] * 100) / (textlength)

text = "GGGG" textlength = 4

A	E	I	O	U	i	Überprüfung	Befehl
0	0	0	0	0	0	i < textlength	-
0	0	0	0	0	0	└ TRUE	-
0	0	0	0	0	0	Check Vowels	-
0	0	0	0	0	1	i < textlength	i + 1
0	0	0	0	0	1	└ TRUE	-
0	0	0	0	0	1	Check Vowels	-
0	0	0	0	0	2	i < textlength	i + 1
0	0	0	0	0	2	└ TRUE	-
0	0	0	0	0	2	Check Vowels	-
0	0	0	0	0	3	i < textlength	i + 1
0	0	0	0	0	3	└ TRUE	-
0	0	0	0	0	3	Check Vowels	-
0	0	0	0	0	4	i < textlength	i + 1
0	0	0	0	0	4	└ FALSE	-
freqA	**freqE**	**freqI**	**freqO**	**freqU**	-	-	-
0	0	0	0	0	-	-	(Vowels[0] * 100) / (textlength)
0	0	0	0	0	-	-	(Vowels[1] * 100) / (textlength)
0	0	0	0	0	-	-	(Vowels[2] * 100) / (textlength)
0	0	0	0	0	-	-	(Vowels[3] * 100) / (textlength)
0	0	0	0	0	-	-	(Vowels[4] * 100) / (textlength)