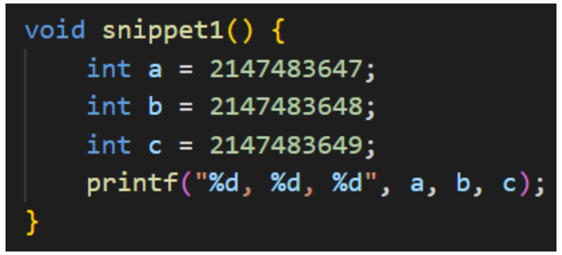
# Informatics II
# Exercise 01 - Week 2

26.02.2024

13:00 – 14:45, BIN-0-B.06

Mirjam Weibel (mirjamalexandra.weibel@uzh.ch)

## Task 1.1



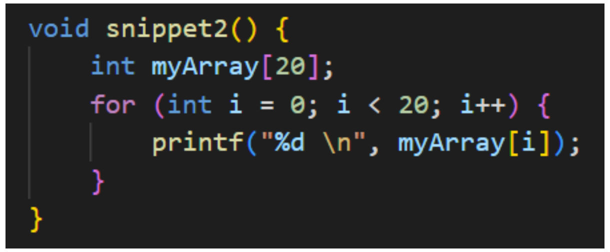
```c
void snippet1() {
    int a = 2147483647;
    int b = 2147483648;
    int c = 2147483649;
    printf("%d, %d, %d", a, b, c);
}
```

Expected output: 2147483647, 2147483648, 2147483649

Actual output/Observation:

Reason:

## Task 1.2



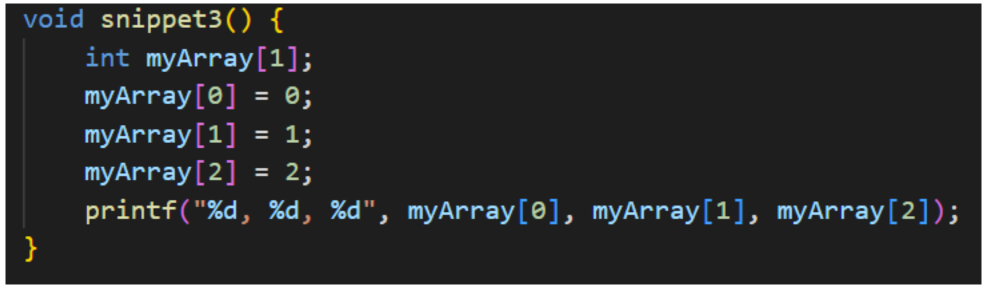
```c
void snippet2() {
    int myArray[20];
    for (int i = 0; i < 20; i++) {
        printf("%d \n", myArray[i]);
    }
}
```

Expected output: array filled with 20 0 values/empty array

Actual output/Observation:

Reason:

## Task 1.3



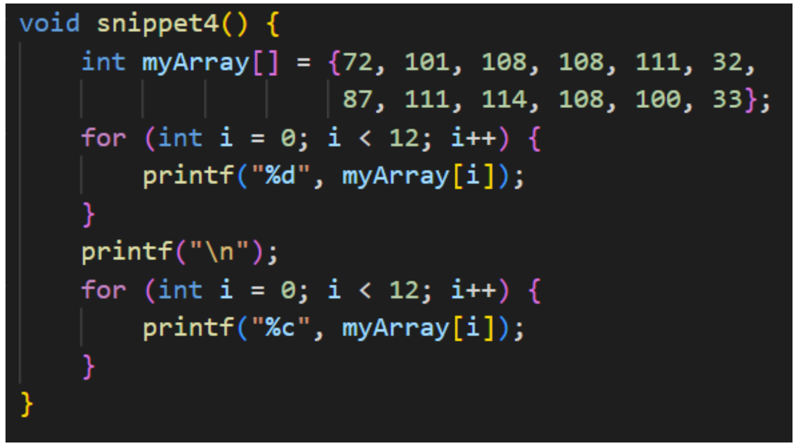
```
void snippet3() {
    int myArray[1];
    myArray[0] = 0;
    myArray[1] = 1;
    myArray[2] = 2;
    printf("%d, %d, %d", myArray[0], myArray[1], myArray[2]);
}
```

Expected output: error/cannot allocate values to indices outside of array bounds

Actual output/Observation:

Reason:

## Task 1.4



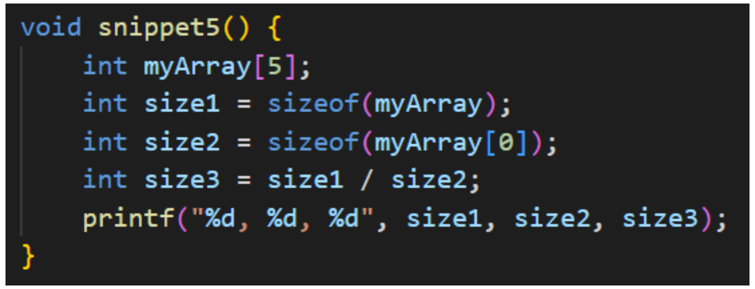
```
void snippet4() {
    int myArray[] = {72, 101, 108, 108, 111, 32,
                     87, 111, 114, 108, 100, 33};
    for (int i = 0; i < 12; i++) {
        printf("%d", myArray[i]);
    }
    printf("\n");
    for (int i = 0; i < 12; i++) {
        printf("%c", myArray[i]);
    }
}
```

Expected output: 72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100, 33 twice/error

Actual output/Observation:

Reason:

## Task 1.5



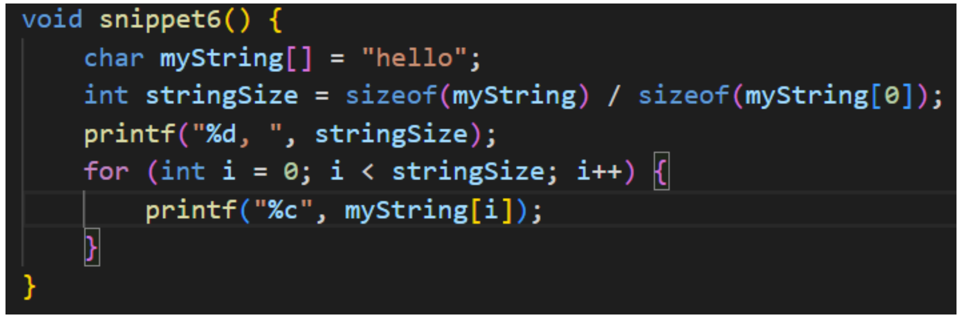
```c
void snippet5() {
    int myArray[5];
    int size1 = sizeof(myArray);
    int size2 = sizeof(myArray[0]);
    int size3 = size1 / size2;
    printf("%d, %d, %d", size1, size2, size3);
}
```

Expected output: 5, 1, 5

Actual output/Observation:

Reason:

## Task 1.6

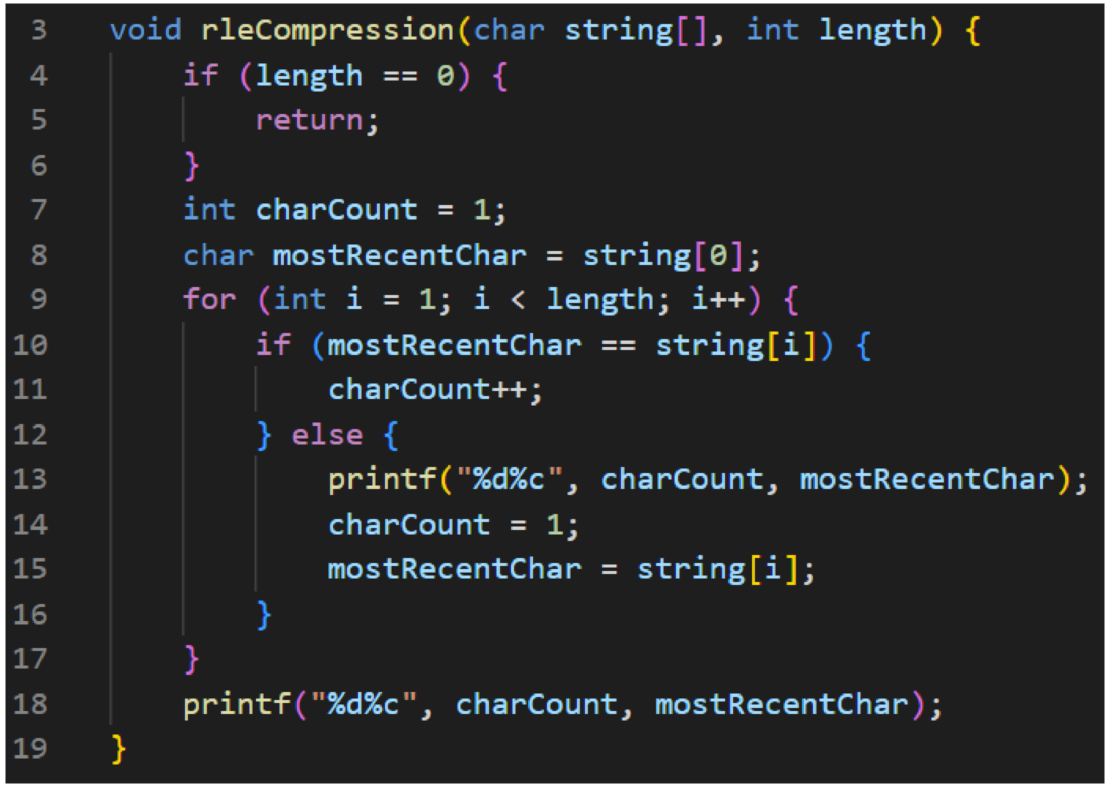

```
void snippet6() {
    char myString[] = "hello";
    int stringSize = sizeof(myString) / sizeof(myString[0]);
    printf("%d, ", stringSize);
    for (int i = 0; i < stringSize; i++) {
        printf("%c", myString[i]);
    }
}
```

Expected output: 5 hello

Actual output/Observation:

Reason:

## Task 2



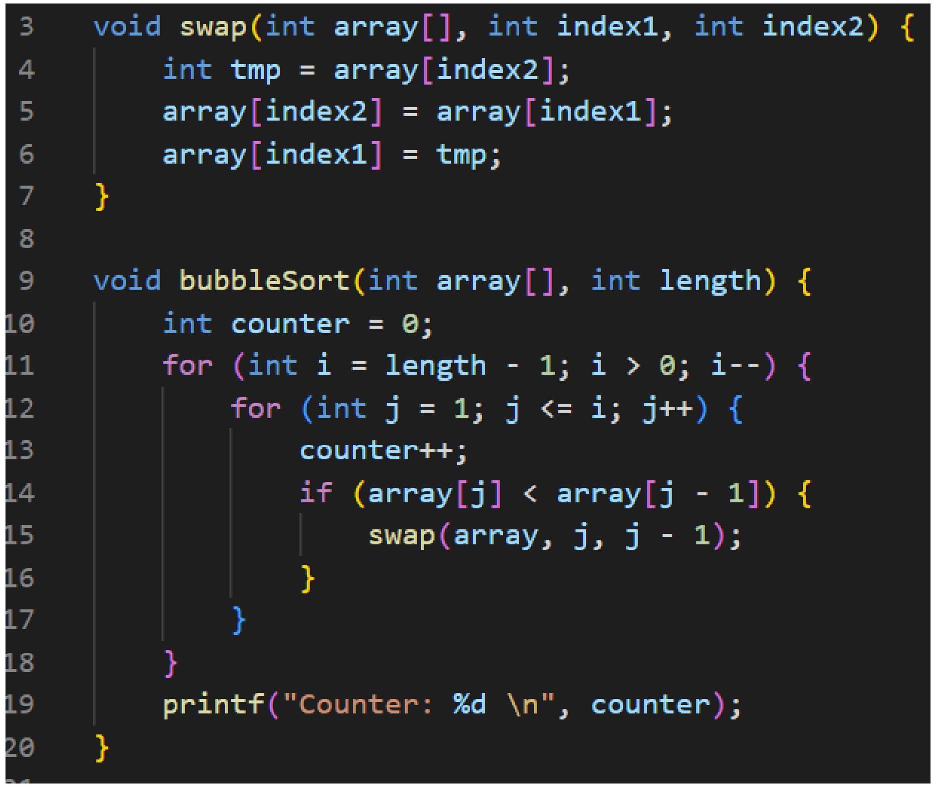
```c
void rleCompression(char string[], int length) {
    if (length == 0) {
        return;
    }
    int charCount = 1;
    char mostRecentChar = string[0];
    for (int i = 1; i < length; i++) {
        if (mostRecentChar == string[i]) {
            charCount++;
        } else {
            printf("%d%c", charCount, mostRecentChar);
            charCount = 1;
            mostRecentChar = string[i];
        }
    }
    printf("%d%c", charCount, mostRecentChar);
}
```

AAABBAA -> 3A2B2A

## Task 3.1 Bubble Sort

- Array [1, 3, 4, 2, 8, 9, 5, 6, 7]

## Task 3.1 Bubble Sort



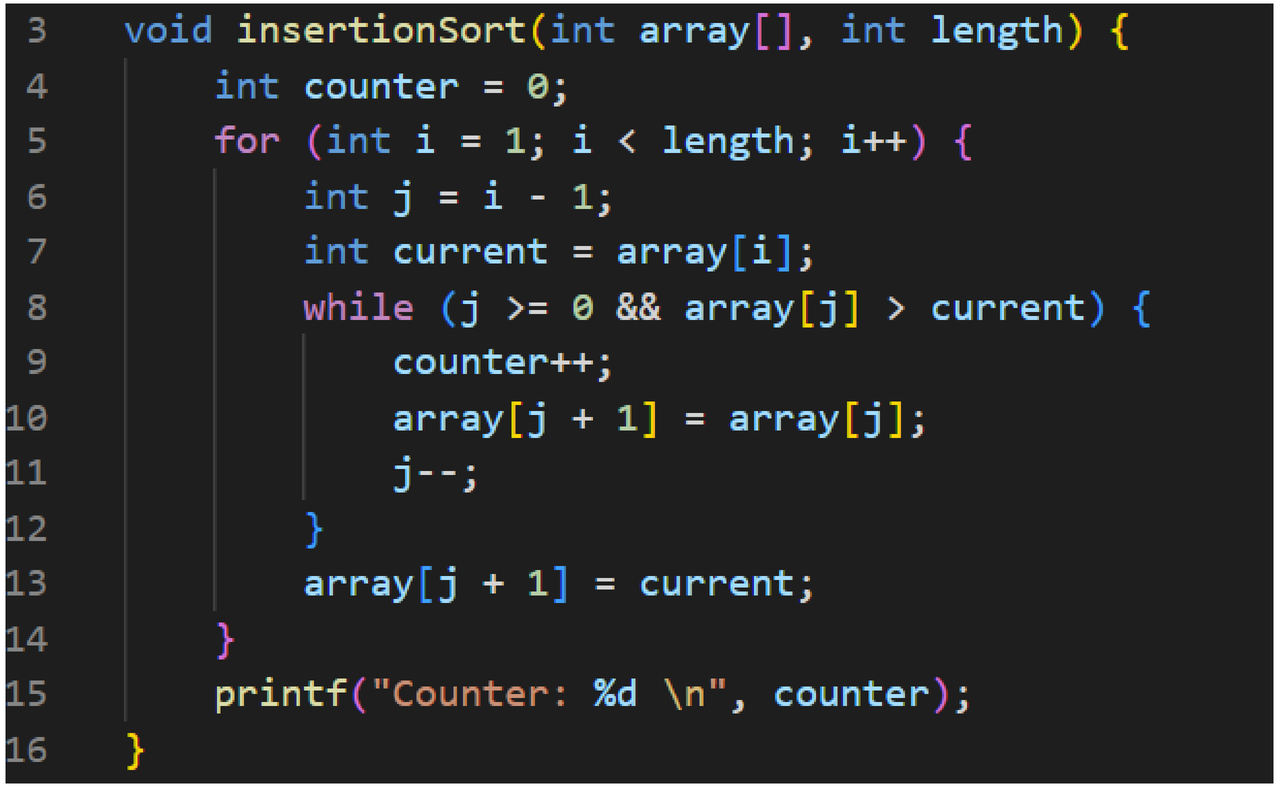
```c
void swap(int array[], int index1, int index2) {
    int tmp = array[index2];
    array[index2] = array[index1];
    array[index1] = tmp;
}

void bubbleSort(int array[], int length) {
    int counter = 0;
    for (int i = length - 1; i > 0; i--) {
        for (int j = 1; j <= i; j++) {
            counter++;
            if (array[j] < array[j - 1]) {
                swap(array, j, j - 1);
            }
        }
    }
    printf("Counter: %d \n", counter);
}
```

## Task 3.2 Insertion Sort

- Array [1, 3, 4, 2, 8, 9, 5, 6, 7]

## Task 3.2 Insertion Sort



```c
void insertionSort(int array[], int length) {
    int counter = 0;
    for (int i = 1; i < length; i++) {
        int j = i - 1;
        int current = array[i];
        while (j >= 0 && array[j] > current) {
            counter++;
            array[j + 1] = array[j];
            j--;
        }
        array[j + 1] = current;
    }
    printf("Counter: %d \n", counter);
}
```

## Nice visualization tool for sorting algorithms

https://www.hackerearth.com/practice/algorithms/sorting/selection-sort/visualize/

## Task 3.3 Bubble Sort vs. Insertion Sort

- Which one executes inner loop more often with the example array?
- Why?

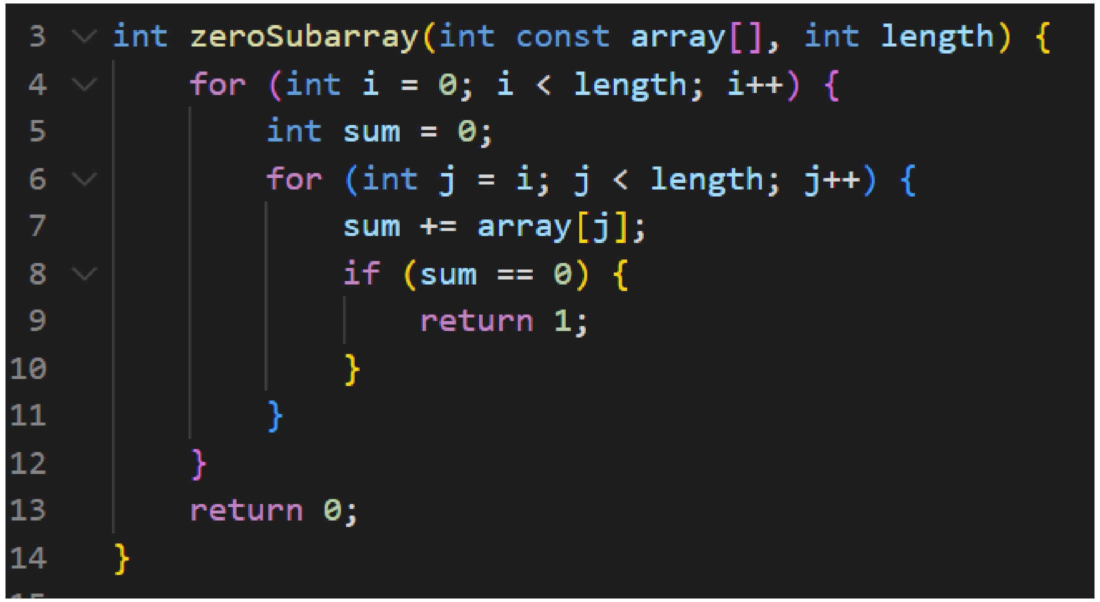
## Task 3.4 Bubble Sort vs. Insertion Sort

- Worst case (most executions of inner loop) for Bubble Sort?
- Worst case (most executions of inner loop) for Insertion Sort?

# Task 3.5 Runtime for Bubble sort

- How often is the innermost loop run for a list with 100,000 elements?

## Task 4



```
int zeroSubarray(int const array[], int length) {
    for (int i = 0; i < length; i++) {
        int sum = 0;
        for (int j = i; j < length; j++) {
            sum += array[j];
            if (sum == 0) {
                return 1;
            }
        }
    }
    return 0;
}
```

Inner loop

Outer loop

| Sum | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |

Array = [3, -2, 4, 2, 1, -5]