EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

WOLFGANG FUHL

PROGRAMMING IN C++

SHEET 4

Submission date: 16.04.2021 until 8:00 am

4.1 Recursion vs. Iteration (10 Points)

C/C++

Often complex problems can be captured very elegantly with recursively formulated rules, as shown in the example of the *tower of Hanoi* in the lecture.
In C and C++, the maximum *depth* (number of self-calls) of the recursion is limited. Exceeding this limit results in a *stack overflow*, which only causes the program to crash in the best case. This task is about investigating this limit and learning to translate recursions into loops.

a) Read the program in Listing 1.

```
1  #include<stdio.h>
2
3  unsigned long endless(unsigned long n) {
4      printf("n = %lu\n", n);
5      return endless(n+1);
6  }
7
8  int main(void)
9  {
10     unsigned long x = endless(1);
11     printf("Recursion terminated successfully, result is: x = %lu\n", x);
12     return 0;
13 }
```

Listing 1: Endlos Rekursion

- What would happen if the *depth of recursion* were not constrained? Will line 11 ever be reached?
- Compile and run the program. How many times will `endless()` be executed? Write down your result!
- Add the line

```
char memoryOnStack[1000];
```

  before the `return` statement in `endless` and execute the program. What has changed?
- Delete the newly inserted line again and run the program a second time. Do you get the same result as before?

b) Replace the function `endless` with the function `recursive` given in Listing 2. Find out what this function computes and then create a function `unsigned long iterative(unsigned long n)` that performs the same task using a loop (without recursion).

```
unsigned long recursive(unsigned long n) {
    if (n < 10) {
        return n;
    }
    return n % 10 + recursive(n / 10);
}
```

Listing 2: Recursion to Iteration

4.2 Depths of the recursion (6 Points)

C/C++

Bei der Verwendung von rekursiven Funktionen gibt es einige Fallstricke. Geben Sie für die folgenden Beispiele die Zeilennummer an, in denen ein Fehler vorliegt und beschreiben Sie den Fehler. Welche Veränderungen muss man vornehmen, damit das Programm *korrekt* funktioniert?
There are some pitfalls when using recursive functions. For the following examples, specify the line number where there is an error and describe the error. What changes must be made to make the program work *correctly*?

```
 1  double functionA(int n) {
 2      int m = n + 12;
 3      m += functionA(n-1) * 0.5;
 4      return m;
 5  }
 6
 7
 8  int main() {
 9      functionA(2);
10  }
```

Listing 3: Example a)

```
1  double functionB(int n) {
2      if (n == 0)
3          return 42;
4      return 2 + functionB(n);
5  }
6
7  int main() {
8      functionB(2);
9  }
```

Listing 4: Example b)

The following program is functional, but still needs improvement. Describe why this program is not optimal.

```
1  double functionC(int n) {
2      if (n == 0)
3          return 0;
4      return functionC(n-1) + functionC(n-2) + functionC(n-1);
5  }
6
7  int main() {
8      functionC(10048);
9  }
```

Listing 5: Example c)

4.3 Real world objects (8 Points)

Objects often represent things from the real world. They have properties *(attributes)* that describe the state of the object and functions *(methods)* for manipulating the object state.
Define the two real world objects *printer* and *RollerSkates*. Specify at least 4 attributes and 4 methods for each object.

4.4 Animalfarm (16 Points)

C++

In this task, you implement a class hierarchy according to the figure 1.

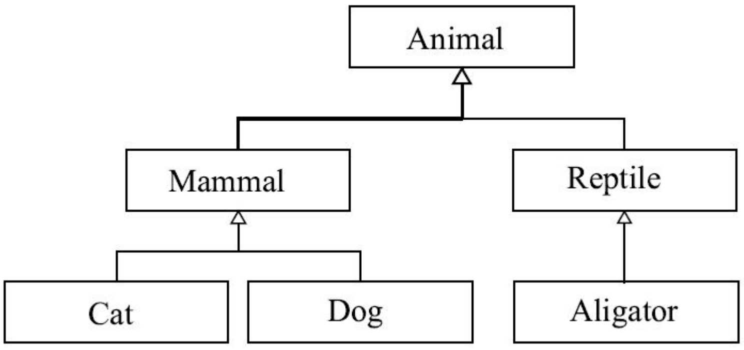

Abbildung 1: Class diagram inheritance

a) Create a `.cpp` and a `.h` file for each of the classes and implement the class bodies in the corresponding header files. Implement the inheritances from the class diagram (all are public inheritances).

b) Add the following private attributes with meaningful types to the classes.

- `Animal`: `age`
- `Mammal`: `hairLength`
- `Cat`: `eyeColor`
- `Dog`: `kilosFoodPerDay`
- `Reptile`: `hasFeet`
- `Aligator`: `numberOfTeeth`

c) Write a constructor for each class (respectively), which initialize its own attributes from suitable parameters. If necessary, call constructors of the parent classes and pass their parameters.

d) Write get methods for all added attributes.

e) Write a set method in the corresponding class for all attributes that can be changed.

f) Extend the constructors and write destructors of all classes so that they output the class name.

g) Implement a `main.cpp` file with a `main` function. This should create two cats, one dog and three alligators on the heap.

h) Create (while the animal objects exist) a new `Animal` pointer array that points to all animal objects. Use this array to output the age of all animals using a loop.