

# Übungspaket 5

## Objektmodellierung in C++

---

### Übungsziele:

1. Modellieren eines Objektes in C++
2. Implementieren einer Klasse in C++

### Literatur:

C++-Skript<sup>1</sup>, Kapitel 19

### Semester:

Wintersemester 2020/21

### Betreuer:

Theo und Ralf

### Synopsis:

In diesem Übungspaket werden wir uns mit der Modellierung von Objekten in der Programmiersprache C++ beschäftigen. Dabei werden anfangs die einzelnen syntaktischen Elemente der Sprache C++ wiederholen. Im Anwendungsteil werden wir wiederum das Problem eines Eingabepuffers mit „unbegrenzter“ Länge bearbeiten. Diesmal werden wir aber ein C++ Programm entwickeln.

---

<sup>1</sup>[www.amd.e-technik.uni-rostock.de/ma/rs/lv/hoqt/script.pdf](http://www.amd.e-technik.uni-rostock.de/ma/rs/lv/hoqt/script.pdf)

# Teil I: Stoffwiederholung

---

## Aufgabe 1: Das Klassenkonzept in C++

Wie (Schlüsselwort) beginnt eine Klasse?	<input type="text"/>
Wie definiert man eine Klasse?	<input type="text"/>
Welche Zugriffsmodifikatoren kennst du?	<input type="text"/>
Wie spezifiziert „man“ Attribute?	<input type="text"/>
Wie spezifiziert „man“ Methoden?	<input type="text"/>
Wie definiert man Objekte?	<input type="text"/>
Wie heißt der Konstruktor?	<input type="text"/>
Wie heißt der Konstruktor der Klasse <code>C</code> ?	<input type="text"/>
Wie heißt der Destruktor?	<input type="text"/>
Wie heißt der Destruktor der Klasse <code>C</code> ?	<input type="text"/>
Welchen Präfix erhält eine Methode bei ihrer Implementierung?	<input type="text"/>
Notiere ein Beispiel	<input type="text"/>
Wie ruft man eine Methode auf?	<input type="text"/>
Notiere ein Beispiel	<input type="text"/>
Wie kann eine Methode auf die Attribute ihres Objektes zugreifen?	<input type="text"/>
Wer erzeugt diesen Zeiger?	<input type="text"/>
Wie erzeugt man Objekte dynamisch?	<input type="text"/>
Warum nimmt man nicht <code>malloc()</code> ?	<input type="text"/>
Wie gibt man diese Objekte wieder frei?	<input type="text"/>

## Aufgabe 2: Methodenaufruf

Erkläre mit eigenen Worten, wie in C++ eine Methoden aufgerufen wird. Erläutere dabei die Begriffe *Objekt*, *Methode* und *this-Zeiger*.

## Teil II: Quiz

---

### Aufgabe 1: Attribute, Methoden, Gültigkeitsbereiche

Für die anstehenden Quizzaufgaben betrachten wir folgendes C++ Programm:

```
1  #include <stdio.h>
2
3  class C {
4      private:
5          int i;
6          int j;
7      public:
8          void print();
9          void set( int i, int b );
10         C( int a );
11     };
12
13 C::C( int a )
14 {
15     i = a; j = a + 4;
16 }
17
18 void C::set( int i, int b )
19 {
20     i = j + b;
21     this->i = i; this->j = b - 1;
22 }
23
24 void C::print()
25 {
26     printf( "i=%2d j=%2d\n", i, j );
27 }
28
29 int main( int argc, char **argv )
30 {
31     C obj( 2 );
32     obj.print();
33     obj.set( 3, 4 );
34     obj.print();
35     return 0;
36 }
```

Schau dir das Programm gut an und versuche zu verstehen, wie es abgearbeitet wird.

Beantworte die folgenden Fragen zu obigem Programm.

Welche Attribute sind in <code>C</code> definiert?	<input type="text"/>
Welche Methoden sind in <code>C</code> definiert?	<input type="text"/>
Welche Elemente in <code>C</code> sind <code>private</code> ?	<input type="text"/>
Elemente in <code>C</code> sind <code>public</code> ?	<input type="text"/>
Welche Werte haben <code>i</code> und <code>j</code> nach Zeile 31?	<input type="text"/>
Welche Werte haben <code>i</code> und <code>j</code> nach Zeile 33?	<input type="text"/>
Kann man in <code>main()</code> auf <code>i</code> und <code>j</code> zugreifen?	<input type="text"/>
Warum ist dies so	<input type="text"/>
Kann man in <code>main()</code> die Methoden aufrufen?	<input type="text"/>
Warum ist dies so	<input type="text"/>
Was ist <code>C( int a )</code> ?	<input type="text"/>

## Teil III: Fehlersuche

---

### Aufgabe 1: Praktische Fehlersuche

DR. CLASS hat gerade angefangen, in C++ zu programmieren. Leider hat er hier und da noch ein paar Tippfehler, bei deren Korrektur wir ihm helfen sollten. Sein erstes Programm sieht wie folgt aus:

```
1  #include <stdio.h>
2
3  cLass C {
4      private
5          int i;
6      Public:
7          void print();
8          void C( int a );
9      } My_Class;
10
11 C( int a )
12 {
13     (*this).i = a;
14 }
15
16 void C: :print()
17 {
18     printf( "i=%2d\n", this.i );
19 }
20
21 int main( int argc, char **argv )
22 {
23     CC obj( 2 );
24     obj.print();
25     printf( "i = %d\n", obj.i );
26     return 0;
27 }
```

# Teil IV: Anwendungen

---

In diesem Anwendungsteil lösen wir die selbe Aufgabe, die wir bereits in Übungspaket 4 gelöst haben. Dabei ging es um die Implementierung eines Puffers, der sich bei Bedarf vergrößert und somit nicht überlaufen kann. Diesmal entwickeln wir allerdings eine C++-Lösung.

## Aufgabe 1: Eingabepuffer beliebiger Länge in C++

### 1. Aufgabenstellung

An dieser Stelle verweisen wir auf Übungspaket 4, Aufgabe 1 des Anwendungsteils, da wir dort die Aufgabenstellung bereits detailliert besprochen haben.

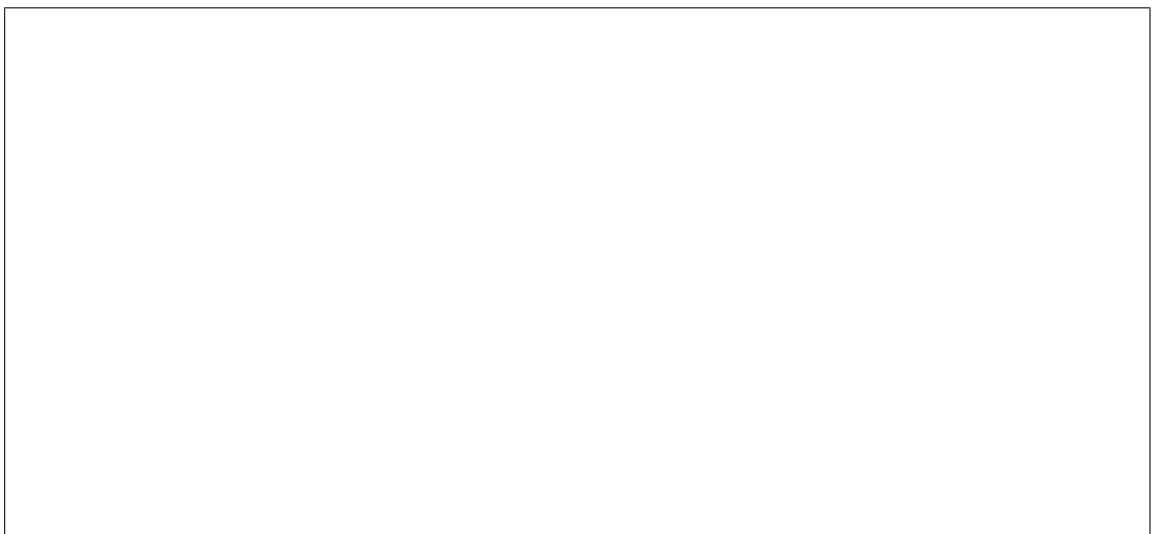
### 2. Entwurf

Im ersten Entwurfsschritt müssen wir definieren, welche Attribute und Methoden unsere C++-Klasse `CBUF` haben soll. Diese Klassendefinition werden wir in der Datei `cbuf.h` ablegen. Die Implementierung aller Methoden werden wir wie üblich in der korrespondierenden Datei `cbuf.cpp` plazieren.

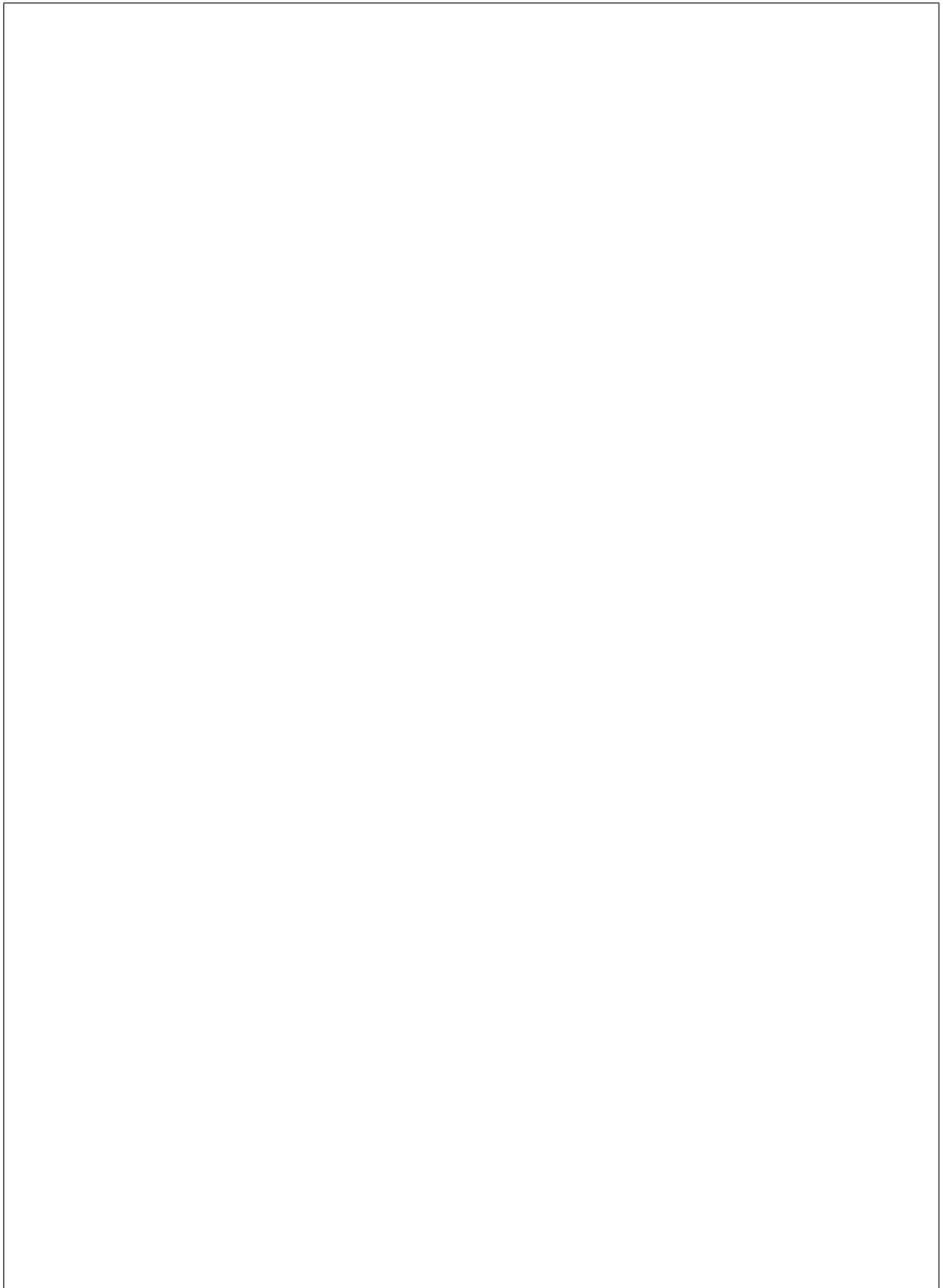
Wie im Skript beschrieben, werden sollten wir die Attribute `private` und die Methoden `public` spezifizieren. Die oben erwähnte C-Realisierung stellt einen guten Ausgangspunkt für diese Teilaufgabe dar:

### 3. Kodierung

Klassendefinition in `cbuf.h`:



**Klassenimplementierung in `cbuf.cpp` (sehr kompakt gesetzt):**



Ein Testprogramm `main.cpp`:

