Hochschule Deggendorf Dr. Peter Jüttner	
Vorlesung: Objektorientierte Programmierung	SS 2019
Übung 4	Termin 11.4.19

Klassen mit Kopierkonstruktor und Zuweisungsoperator

1. **Aufgabe: Kein parameterloser Konstruktor**
   In einer Klasse sollen die Attribute der Objekte immer mitttels Werten initialisiert werden, die als Parameter eines Konstruktors übergeben werden. Wie kann dies erreicht werden, bzw. wie kann erreicht werden, dass der parameterlose Konstruktor nicht außerhalb der Klasse bzw. gar nicht aufgerufen wird? Überprüfen Sie Ihre Überlegungen am Beispiel complex.

   Parameterloser Konstruktor kapseln, d.h. im private-Teil definieren

2. **Aufgabe: Kopierkonstruktor und Zuweisungsoperator**

   a.) Ändern Sie die Klasse Complex so ab, dass die Attribute für Real- und Imaginärteil immer auf dem Heap angelegt werden. Complex hat also zwei Attribute vom Typ double*. Verwenden Sie den Operator new um Speicherplatz auf dem Heap zu reservieren bzw. delete um den Speicher wieder frei zu geben. Passen Sie Konstruktoren, Destruktoren und andere Methoden an. Probieren Sie Ihre geänderte Klasse aus.

   b.) Legen Sie zwei Complex-Objekte c1 und c2 mit in c1 und c2 verschiedenen Werten für Real- und Imaginärteil an. Weisen Sie c2 an c1 zu (c1 = c2). Verändern Sie c1. Stellen Sie fest, ob sich auch c2 verändert hat.

   c.) Ergänzen Sie die geänderte Klasse Complex um einen selbst definierten Kopierkonstruktor und einen eigenen Zuweisungsoperator.
   Wiederholen Sie die Schritte aus Aufgabenteil b.

```
int zahl_complex = 0;

class complex
{ float *real;
  float *imag;

  public:
```

```
complex()
{ real = new float;
  imag = new float;
  *real = 0.0;
  *imag = 0.0;
  zahl_complex++;
  printf("parameterloser Konstruktor aufgerufen\n");
  printf("Anzahl Complex Objekte: %d\n",zahl_complex);
};

complex(float r, float i)
{ real = new float;
  imag = new float;
  *real = r;
  *imag = i;
  zahl_complex++;
  printf("Konstruktor mit 2 Parametern aufgerufen\n");
  printf("Anzahl Complex Objekte: %d\n",zahl_complex);
};

~complex()
{ delete real;
  delete imag;
  zahl_complex--;
  printf("Destruktor aufgerufen\n");
  printf("Anzahl Complex Objekte: %d\n",zahl_complex);
};
void ausgabe()
{ printf("%s%f\n","Realteil: ",*real);
  printf("%s%f\n","Imagteil: ",*imag);
};
void addiere(complex b)
{ *real = *real + *(b.real);
  *imag = *imag + *(b.imag);
};
void subtrahiere(complex b)
{ *real = *real - *(b.real);
  *imag = *real - *(b.imag);
};
void addiere_real(float r)
{ *real = *real + r;
};
void initialisiere(float r, float i)
{ *real = r;
  *imag = i;
};
void lesecomplex()
{ printf("Bitte Realteil eingeben:\n");
  scanf("%f",real);
  printf("Bitte Imaginaerteil eingeben:\n");
  scanf("%f",imag);
```

```
  };

  complex(const complex &c)  /* Copy Konstruktor */
  { real = new float;
    imag = new float;
    *real = *(c.real);
    *imag = *(c.imag);
  };
  complex& operator=(const complex& c) /* Zuweisungsoperator */
  { real = new float;
    imag = new float;
    *real = *(c.real);
    *imag = *(c.imag);
    return *this;
  };
};
```

d.) Erweitern Sie alle Kontruktoren um einen globalen Zähler für die konstruierten Ob jekte.

Lösung analog zum Übungsblatt 3