Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim Geisenheim

Fachbereich Design Informatik Medien
Studiengang Bachelor Allgemeine Informatik

Programmieren 3 AI

Klausuraufgaben

Sebastian Flothow

2012-02-08

Dieses Dokument enthält eine Sammlung potentieller Klausuraufgaben. Die abgedruckten Quelltexte können auch im Archiv `klausuraufgaben-src.tar.gz` heruntergeladen werden. Um die Organisation und Erzeugung dieses Dokuments zu vereinfachen sind die Quelltexte so verfasst dass jede Aufgabe in genau einer Quelltextdatei enthalten ist; dies führt hier zu Vorgehensweisen von denen man unter normalen Umständen absehen sollte (Dateinamen die nicht den enthaltenen Klassen entsprechen, mehrere Klassen in einer Datei, keine Header-Dateien). Die Aufgaben basieren teilweise auf Klausuraufgaben von Prof. Dr. Panitz.

1 Multithreading (Java)

Jedes Programm enthält ein Synchronisationsproblem. Erläutern Sie das Problem und beschreiben Sie wie es behoben werden kann.

19-007.java

```
1  class Pot {
2      private int content = 0;
3
4      public int getContent() {
5          return content;
6      }
7
8      public void put(int n) {
9          content += n;
10     }
```

```
11
12      public void take() {
13          if (content <= 0) {
14              throw new IllegalStateException("Topf ist leer");
15          }
16          content--;
17      }
18  }
19
20  class Savage implements Runnable {
21      private Pot pot;
22
23      public Savage(Pot p) {
24          pot = p;
25      }
26
27      public void run() {
28          while (pot.getContent() > 0) {
29              try {
30                  Thread.sleep(10);
31                  pot.take();
32              } catch (Exception e) {
33                  e.printStackTrace();
34              }
35          }
36      }
37  }
38
39  class Main {
40      public static void main(String[] args) {
41          Pot pot = new Pot();
42          pot.put(1000);
43          new Thread(new Savage(pot)).start();
44          new Thread(new Savage(pot)).start();
45      }
46  }
```

19-268.java

```
1   class A implements Runnable {
2       private Object o1;
3       private Object o2;
4
5       public A(Object o1, Object o2) {
```

```
        this.o1 = o1;
        this.o2 = o2;
    }

    public void run() {
        for (int i = 0; i < 100; i++) {
            synchronized (o1) {
                synchronized (o2) {
                    try {
                        System.out.println(
                            Thread.currentThread().getName()
                            + ": " + i);
                        Thread.sleep(100);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            }
        }
    }
}

class Main {
    public static void main(String[] args) {
        String x = "foo";
        String y = "bar";

        new Thread(new A(x, y)).start();
        new Thread(new A(y, x)).start();
    }
}
```

19-425.java

```
class A extends Thread {
    boolean keepGoing = true;

    public void run() {
        while (keepGoing) {
            /* ... */
        }
    }

    public void shutdown() {
```

```
        keepGoing = false;
    }
}

class Main {
    public static void main(String[] args) throws Exception {
        A a = new A();
        a.start();
        Thread.sleep(500);
        a.shutdown();
        System.out.println("Fertig");
    }
}
```

19-640.java

```
class Sequence {
    volatile private int n = 0;
    public int next() {
        return ++n;
    }
}

class Worker extends Thread {
    private Sequence s;
    public Worker(Sequence s) {
        this.s = s;
    }
    public void run() {
        for (int i = 0; i < 100; i++) {
            int seqno = s.next();
            /* ... */
        }
    }
}

class Main {
    private final static int N_THREADS = 100;
    public static void main(String[] args)
        throws InterruptedException {
        Worker[] w = new Worker[N_THREADS];
        Sequence s = new Sequence();
        for (int i = 0; i < N_THREADS; i++) {
            w[i] = new Worker(s);
```

```
29              w[i].start();
30          }
31          for (int i = 0; i < N_THREADS; i++) {
32              w[i].join();
33          }
34          System.out.println("Nächste Nummer: " + s.next());
35      }
36  }
```

2 Fehlersuche (C++)

Jeder Quelltext enthält einen Fehler, der die Übersetzung zu einem ausführbaren Programm verhindert. Erläutern Sie den Fehler und beschreiben Sie wie er behoben werden kann.

28-021.cpp

```
1   #include <iostream>
2
3   class A {
4   public:
5       int x;
6   };
7
8   class B : public A {
9   public:
10      bool check() const;
11  };
12
13  class C : public A {
14  public:
15      void increment();
16  };
17
18  class D : public B, public C {
19  public:
20      bool incrementAndCheck();
21  };
22
23  bool B::check() const {
24      return (x == 42);
25  }
26
27  void C::increment() {
```

```
    ++x;
}

bool D::incrementAndCheck() {
    increment();
    return check();
}

int main() {
    D d;
    d.x = 41;
    std::cout << d.incrementAndCheck() << std::endl;
    std::cout << d.x << std::endl;
    return 0;
}
```

28-054.cpp

```
template <typename A> class Box {
public:
    A x;
    Box(A x) : x(x) {}
}

int main() {
    new Box<int>(42);
}
```

28-071.cpp

```
class B {
public:
    int i;
    B(int i);
    B operator+(B& other);
};

B::B(int i) : i(i) {}

B B::operator+(B& other) {
    B result(i + other.i);
    return result;
}

int main() {
```

```
    B* b0 = new B(20);
    B* b1 = new B(22);
    B* b2 = b0 + b1;
    return 0;
}
```

28-072.cpp

```
#include <iostream>
using namespace std;

class A {
public:
    A(int x);
    int getX() const;
private:
    int x;
};

class B : A {
public:
    B(int x);
};

A::A(int x) : x(x) {}

int A::getX() const {
    return x;
}

B::B(int x) : A(x) {}

int main() {
    B b(105);
    cout << b.getX() << endl;
}
```

28-081.cpp

```
class A {
public:
    int i;
    A(int i);
};

```

```
A::A(int i) : i(i) {}

int main() {
    A a;
    A b = 42;
    return 0;
}
```

28-090.cpp

```
class A {
private:
    int x;
public:
    void A(int x);
    int getX() const;
};

void A::A(int x) : x(x) {}

int A::getX() const {
    return x;
}

int main() {
    A a(23);
}
```

28-112.cpp

```
class A {
public:
    A(int x);
    virtual void modify()=0;
protected:
    int x;
};

class B : A {
public:
    B(int x);
    virtual void modify();
};

A::A(int x) : x(x) {}
```

```
16
17 B::B(int x) : A(x) {}
18
19 virtual void B::modify() {
20     x *= 2;
21 }
22
23 int main() {
24     B b(21);
25     b.modify();
26     return 0;
27 }
```

28-160.cpp

```
1  #include <string>
2  #include <iostream>
3  using namespace std;
4
5  class Person {
6  public:
7      string name;
8      string vorname;
9
10     Person(string name, string vorname);
11     string getFullName();
12 };
13
14 Person::Person(string name, string vorname) :
15     name(name), vorname(vorname) {}
16
17 string Person::getFullName() {
18     return vorname+" "+name;
19 }
20
21 template <typename T> T k(T x, T y) {
22     return x + y;
23 }
24 int main() {
25     char* n ="Zerlett";
26     string v ="Helmut";
27     cout << k(2, 40) << endl;
28     cout << k(Person(n, v), Person(n, v)).name << endl;
29 }
```

28-170.cpp

```cpp
#include <iostream>
using namespace std;

class Person {
public:
    string name;
    string vorname;
    Person(string name, string vorname);
    string getFullName();
};

Person::Person(string name, string vorname) :
    name(name), vorname(vorname) {}

string Person::getFullName(){
    return vorname+" "+name;
}

int main() {
    Person p("Andrak", "Manuel");
    cout << p->getFullName();
}
```

28-186.cpp

```cpp
#include <iostream>

class A {
private:
    int x;
public:
    A(int x);
    int getX();
};

A::A(int x) : x(x) {}

int A::getX() {
    return x;
}

int main() {
    A a;
```

```
    std::cout << a.getX() << std::endl;
}
```

28-243.cpp

```
int f(int& x) {
    x = x + 2;
    return x;
}

int main() {
    f(4);
    return 0;
}
```

28-381.cpp

```
#include <iostream>

template <typename T> class Box {
    Box(T x) : x(x) {};
    T get() const {
        return x;
    }
private:
    const T x;
};

int main() {
    Box<float> b(23.47);
    std::cout << b.get() << std::endl;
    return 0;
}
```

28-383.cpp

```
#include <iostream>

class StringBox {
public:
    std::string get() const;
    void set(const std::string& s);
private:
    std::string s;
};
```

```
10
11 std::string StringBox::get() const {
12     return s;
13 }
14
15 void StringBox::set(const std::string s) {
16     this->s = s;
17 }
18
19 int main() {
20     StringBox sb;
21     sb.set("Hello World");
22     std::cout << sb.get() << std::endl;
23     return 0;
24 }
```

28-410.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  class Person {
5  public:
6      string name;
7      string vorname;
8
9      Person(string name, string vorname);
10     string toString();
11 };
12
13 Person::Person(string name, string vorname) :
14     name(name), vorname(vorname) {}
15
16 string Person::toString() {
17     return vorname+" "+name;
18 }
19
20 template <typename T> T t(T x) {
21     cout << x.toString() << endl;
22     return x;
23 }
24
25 int t(int x) {
26     cout << x << endl;
```

```
27      return x;
28  }
29
30  int main() {
31      char* n = "Zerlett";
32      string v = "Helmut";
33      Person p = t(Person(n, v));
34      int i = t(42);
35      t(new Person(n, v));
36  }
```

28-423.cpp

```
1   #include <iostream>
2
3   class A1 {
4   public:
5       void sayIt();
6   };
7
8   class A2 {
9   public:
10      void sayIt();
11  };
12
13  class A3 : public A1, public A2 {
14  public:
15      void los();
16  };
17
18  void A1::sayIt() {
19      std::cout << "foo" << std::endl;
20  }
21
22  void A2::sayIt() {
23      std::cout << "bar" << std::endl;
24  }
25
26  void A3::los() {
27      this->sayIt();
28  }
29
30  int main() {
31      A3* a = new A3();
```

```
    a->los();
    return 0;
}
```

28-445.cpp

```
#include <iostream>

class A {
public:
    std::string getFinalMessage();
};

std::string getFinalMessage() {
    return "we apologise for the inconvenience";
}

int main() {
    std::cout << (new A())->getFinalMessage() << std::endl;
}
```

28-497.cpp

```
#include <iostream>
using namespace std;

class A {
public:
    A(int x);
    int getX() const;
private:
    int x;
};

A::A(int x) : x(x) {}

int A::getX() {
    return x;
}

int main() {
    cout << A(69).getX() << endl;
    return 0;
}
```

28-501.cpp

```
int main() {
    int i1 = 17;
    int* i2 = &i1;
    int& i3 = i1;
    (*i2) = i1 + i3;
    int& i4 = 2 * i1;
    return 0;
}
```

28-592.cpp

```
#include <iostream>

int f() {
    return 42;
}

int main() {
    int y = f();
    int& x = f();
    x=y;
}
```

28-623.cpp

```
#include <vector>
#include <complex>

int main() {
    std::vector<std::complex<double>> cmplvec;
    cmplvec.push_back(std::complex<double>(0, 1));
    cmplvec.push_back(std::complex<double>(17, 23));
    cmplvec.push_back(std::complex<double>(105, -42));
    cmplvec.push_back(666);
    return 0;
}
```

28-675.cpp

```
#include <iostream>

class Date {
    int year;
    int month;
```

```
6       int day;
7
8       Date(int year, int month, int day);
9   };
10
11  Date::Date(int year, int month, int day) :
12      year(year), month(month), day(day) {}
13
14  int main() {
15      Date d=Date(2004,1,26);;
16      std::cout << "starting main" << std::endl;
17  }
```

28-687.cpp

```
1   #include <string>
2
3   template <typename A, typename B> class P {
4   public:
5       A fst;
6       B snd;
7       P(A x, B y) : fst(x), snd(y) {}
8       void swap() {
9           B x = snd;
10          snd = fst;
11          fst = x;
12      }
13  };
14
15  int main() {
16      P<std::string, std::string>* p1 =
17          new P<std::string, std::string>("hallo", "welt");
18      P<int,std::string>* p2 = new P<int,std::string>(17, "4");
19      p2->swap();
20      p1->swap();
21      return 0;
22  }
```

28-689.cpp

```
1   #include <iostream>
2
3   class A1 {
4   public:
5       virtual std::string toString()=0;
```

```
6       virtual int getAnswer();
7   };
8
9   class A2 : A1 {
10  public:
11      std::string toString();
12      virtual int getAnswer();
13  };
14
15  int A1::getAnswer() {
16      return 42;
17  }
18
19  std::string A2::toString() {
20      return A1::toString();
21  }
22
23  int A2::getAnswer() {
24      return A1::getAnswer();
25  }
26
27  int main() {
28      A2 a2;
29      return 0;
30  }
```

28-712.cpp

```
1   #include <iostream>
2
3   class Date {
4   public:
5       int year;
6       int month;
7       int day;
8
9       Date(int year, int month, int day);
10  };
11
12  Date::Date(int year, int month, int day) :
13      year(year), month(month), day(day) {}
14
15  int main(){
16      Date d;
```

```
17      std::cout << "starting main" << std::endl;
18      d = Date(2004, 1, 26);
19  }
```

28-777.cpp

```
1   #include <string>
2   #include <iostream>
3   using namespace std;
4
5   class Person {
6   public:
7       string name;
8       string vorname;
9
10      Person(string name, string vorname);
11      string getFullName();
12  };
13
14  Person::Person(string name ,string vorname) :
15      name(name), vorname(vorname) {}
16
17  string Person::getFullName() {
18      return vorname+" "+name;
19  }
20
21  int main() {
22      Person* p = new Person("Zerlet", "Helmut");
23      cout << p.getFullName();
24  }
```

28-872.cpp

```
1   #include <iostream>
2
3   class A {
4   public:
5       int answer();
6   };
7
8   class B {
9   public:
10      std::string message();
11  };
12
```

```
class C : public A, B {
public:
    int x;
    C();
};

int A::answer() {
    return 42;
}

std::string B::message() {
    return "we apologise for the inconvenience";
}

C::C() : x(46) {}

int main() {
    C c;
    std::cout << c.answer() << std::endl;
    std::cout << c.message() << std::endl;
    std::cout << (c.x / 2) << std::endl;
    return 0;
}
```

28-879.cpp

```
#include <iostream>

class StringBox {
public:
    std::string get() const;
    void set(const std::string& s);
private:
    std::string s;
};

std::string StringBox::get() {
    return s;
}

void StringBox::set(const std::string& s) {
    this->s = s;
}

```

```
int main() {
    StringBox sb;
    sb.set("Hello World");
    std::cout << sb.get() << std::endl;
    return 0;
}
```

28-915.cpp

```
#include <iostream>

namespace h2g2 {
    using namespace std;
    void answer() {
        cout << 42 << endl;
    }
}

int main() {
    h2g2:answer();
}
```

28-926.cpp

```
#include <string>
#include <iostream>
using namespace std;

class Person {
public:
    string name;
    string vorname;

    Person(string name, string vorname);
    string getFullName();
};

Person::Person(string name, string vorname) :
    name(name), vorname(vorname) {}

string Person::getFullName() {
    return vorname+" "+name;
}

int main() {
```

```
22      char* n = "Zerlett";
23      string v = "Helmut";
24      Person p = new Person(n, v);
25      cout << p.getFullName() << endl;
26      cout << p.vorname << endl;
27  }
```

28-935.cpp

```
1   int main() {
2       int xs = 42;
3       int& rXs = xs;
4       rXs = 15;
5       int&& rrXs = rXs;
6       rrXs = 17;
7   }
```

28-937.cpp

```
1   #include <iostream>
2
3   class A1 {
4   public:
5       virtual std::string toString()=0;
6       virtual int getAnswer();
7   };
8
9   int A1::getAnswer() {
10      return 42;
11  }
12
13  int main() {
14      A1* a1 = new A1();
15      std::cout << a1->getAnswer() << std::endl;
16      return 0;
17  }
```

3 Berechnen von Hand (C++)

Berechnen Sie die Ausgabe der folgenden Programme von Hand; verzeichnen Sie dabei auch nicht gefangene Exceptions. Erklären Sie kurz, wie es zu der Ausgabe kommt.

46-002.cpp

```cpp
#include <iostream>
using namespace std;

template <typename T> class Box {
public:
    T content;
    Box<T>(T x) : content(x) {}
};

int f(int& x) {
    x = x + 4;
    return x;
}

template <typename T> void fB(Box<T> b) {
    b.content = f(b.content);
}

int main(){
    int y = 42;
    int z = (f(y));
    Box<int> b(17);
    fB(b);

    cout << z << endl;
    cout << y << endl;
    cout << b.content << endl;
}
```

46-044.cpp

```cpp
#include <iostream>

class O {
public:
    void printMe();
};

class U : public O {
public:
    void printMe();
};

```

```
13 void O::printMe() {
14     std::cout << "O" << std::endl;
15 }
16
17 void U::printMe() {
18     std::cout << "U" << std::endl;
19 }
20
21 int main() {
22     U* u = new U();
23     O* o = u;
24     u->printMe();
25     o->printMe();
26     delete o;
27     return 0;
28 }
```

46-097.cpp

```
1  #include <iostream>
2
3  class O {
4  public:
5      int x;
6      O(int x);
7      virtual void print();
8  };
9
10 class U : public O {
11 public:
12     U(int x);
13     virtual void print();
14 };
15
16 O::O(int x) : x(x) {}
17
18 void O::print() {
19     std::cout << x << std::endl;
20 }
21
22 U::U(int x) : O(x) {}
23
24 void U::print() {
25     std::cout << (x * 2) << std::endl;
```

```
}

void print(O& o) {
    o.x = o.x + 4 ;
    o.print();
}

int main() {
    U u(17);
    print(u);
    std::cout << u.x << std::endl;
    O o = u;
    print(o);
    std::cout << o.x << std::endl;
    return 0;
}
```

46-099.cpp

```
#include <iostream>

class C {
public:
    int x;
    int y;
    C(int x);
    C(const C& other);
};

C::C(int x) : x(x), y(42) {}

C::C(const C& other) : x(other.x + 1), y(2 * other.y) {}

int main() {
    C c1 = 17;
    C c2 = c1;
    std::cout << c1.x << ", " << c1.y << std::endl;
    std::cout << c2.x << ", " << c2.y << std::endl;
}
```

46-107.cpp

```
#include <iostream>

class C {
```

```
public:
    int counter;
    C();
    int nextOdd();
};

C::C() : counter(0) {}

int C::nextOdd() {
    counter++;
    if (counter % 2 == 0) throw 42;
    return counter;
}

int main() {
    C c;
    try {
        std::cout << c.nextOdd() << std::endl;
        std::cout << c.nextOdd() << std::endl;
        std::cout << c.nextOdd() << std::endl;
        std::cout << c.nextOdd() << std::endl;
    } catch (std::string s) {
        std::cout << s << std::endl;
    } catch (int i) {
        if (i == 42) {
            std::cout << c.nextOdd() << std::endl;
        } else {
            std::cout << "unknown error: " << i << std::endl;
        }
    }
}
```

46-165.cpp

```
#include <string>
#include <iostream>
using namespace std;

class C1 {
public:
    string getDescription();
};

class C2 : public C1 {
```

```
11      string getDescription();
12  };
13
14  string C1::getDescription() {
15      return "C1";
16  }
17
18  string C2::getDescription() {
19      return "C2";
20  }
21
22  int main() {
23      C1* c1 = new C1();
24      C1* c2 = new C2();
25
26      cout << c1->getDescription() << endl;
27      cout << c2->getDescription() << endl;
28  }
```

46-203.cpp

```
1   #include <iostream>
2   using namespace std;
3
4   class C {
5   public:
6       int x;
7       C(int x);
8       virtual ~C();
9       int operator()(int y);
10  };
11
12  C::C(int x) : x(x) {}
13
14  C::~C() {
15      this->x = 42;
16      cout << x << endl;
17  }
18
19  int C::operator()(int y) {
20      x += y;
21      return x;
22  }
23
```

```
int main() {
    C* c1 = new C(17);
    C c2 = *c1;
    cout << c2(5) << endl;
    cout << (*c1)(5) << endl;
    cout << c1->x << endl;
    delete c1;
    cout << c2.x << endl;
    return 0;
}
```

46-272.cpp

```
#include <iostream>
using namespace std;

class C {
public:
    int x;
    C(int x);
    virtual void setX(int x);
    virtual int getX();
};

C::C(int x) : x(x) {}

void C::setX(int x) {
    this->x = x;
}

int C::getX() {
    return x;
}

int f(C c) {
    c.setX(2 * c.getX());
    return c.x;
}

int main() {
    C* c = new C(17);
    int x = f(*c);
    cout << x << endl;
    cout << c->getX() << endl;
```

```
    cout << (*c).x << endl;
    c->setX(42);
    cout << x << endl;
    cout << c->getX() << endl;
    cout << (*c).x << endl;
    delete c;
    return 0;
}
```

46-307.cpp

```
#include <iostream>

int main() {
    int i = 17;
    int* i1 = &i;
    int& i2 = i;
    i2=(*i1) + 4;
    int i3 = *i1 + i2;
    *i1 = 2 * i2;
    std::cout << i << std::endl;
    std::cout << i1 << std::endl;
    std::cout << i2 << std::endl;
    std::cout << i3 << std::endl;
    return 0;
}
```

46-330.cpp

```
#include <iostream>

int f(int& x) {
    int r = x;
    x = (x % 2 == 0) ? (x / 2) : (3 * x + 1);
    return r;
}

int main() {
    int i;
    for (i = 0; i < 5; i++);
        std::cout << f(i) << std::endl;
}
```

46-456.cpp

```cpp
#include <iostream>

template <int N> class A {
public:
    enum { v = N * A<N-1>::v };
};

template <> class A<0> {
public:
    enum { v = 1 };
};

int main() {
    std::cout << A<10>::v << std::endl;
    return 0;
}
```

46-476.cpp

```cpp
#include <iostream>

int f(int& x) {
    if (x < 0) {
        x = 0;
        throw 42;
    }
    int result = x * x;
    x = x + 1;
    return result;
}

int main() {
    int x = 0;
    int y = -1;
    int z = 4;
    try {
        x = f(z);
        std::cout << x << std::endl;
        std::cout << z << std::endl;
        x = f(y);
        std::cout << x << std::endl;
        std::cout << y << std::endl;
    } catch (std::string e) {
```

```
        x = 2;
    } catch (int i) {
        x = i;
    }
    std::cout << x << std::endl;
    std::cout << y << std::endl;
    return 0;
}
```

46-525.cpp

```
#include <iostream>
using namespace std;

class U {
public:
    int x;
    U(int x);
    virtual int getX();
    void setX(int x);
};

class L : public U {
public:
    L(int x);
    virtual int getX();
    void setX(int x);
};

U::U(int x) : x(x) {}

int U::getX() {
    return x;
}

void U::setX(int x) {
    this->x = x;
}

L::L(int x) : U(2 * x) {}

int L::getX() {
    return x * 3;
}
```

```
34
35 void L::setX(int x) {
36     this->x = 2 * x;
37 }
38
39 int main() {
40     L l = L(1);
41     U u = l;
42
43     U& uRe1 = l;
44     U& uRe2 = u;
45
46     L* low = &l;
47     U* up = low;
48
49     cout << l.x << " " << l.getX() << endl;
50     l.setX(10);
51     cout << l.x << " " << l.getX() << endl;
52
53     cout << u.x << " " << u.getX() << endl;
54     u.setX(10);
55     cout << u.x << " " << u.getX() << endl;
56
57     cout << uRe1.x << " " << uRe1.getX() << endl;
58     uRe1.setX(10);
59     cout << uRe1.x << " " << uRe1.getX() << endl;
60
61     cout << uRe2.x << " " << uRe2.getX() << endl;
62     uRe2.setX(100);
63     cout << uRe2.x << " " << uRe2.getX() << endl;
64
65     cout << low->x << " " << low->getX() << endl;
66     low->setX(1000);
67     cout << low->x << " " << low->getX() << endl;
68
69     cout << up->x << " " << up->getX() << endl;
70     up->setX(10000);
71     cout << up->x << " " << up->getX() << endl;
72
73     return 0;
74 }
```

46-631.cpp

```cpp
#include <iostream>

class O {
public:
    int i;
    O(int i);
    O changeThat(O& that);
};

O::O(int i) : i(i) {}

O O::changeThat(O& that) {
    O result = that;
    that.i = that.i + this->i;
    return result;
}

int main() {
    O o1(17);
    O o2(4);
    O o3 = o1.changeThat(o2);
    std::cout << o1.i << std::endl;
    std::cout << o2.i << std::endl;
    std::cout << o3.i << std::endl;
    return 0;
}
```

46-701.cpp

```cpp
#include <iostream>

int f1(int x) {
    int result = 0;
    for (int i = 0; i <= x; i++) {
        if (i == 3) throw result;
        if (x < 0) throw "negatives Argument in f1";
        result = result + i;
    }
    return result;
}

int f2(int i) {
    try {
```

```
        return f1(i);
    } catch (int i) {
        return i;
    }
}

int main() {
    std::cout << f2(12) << std::endl;
    return 0;
}
```

46-720.cpp

```
#include <iostream>
using namespace std;

template <class T> class Box {
public:
    T content;

    Box<T>(T x) : content(x) {};
};

template <typename T> void fB(Box<T> b) {
    T x = 2 * b.content;
    b = Box<T>(x);
    b.content = 42;
}

int main() {
    Box<int> b(17);
    fB(b);
    cout << b.content << endl;
    Box<int>& c = b;
    fB(c);
    cout << b.content << endl;
}
```

46-770.cpp

```
#include <string>
#include <iostream>
using namespace std;

class C1 {
```

```
public:
    virtual string getDescription();
};

class C2 : public C1 {
public:
    virtual string getDescription();
};

string C1::getDescription(){
    return "C1";
}

string C2::getDescription(){
    return "C2";
}

int main() {
    C1 c1;
    C2 c2;
    c1 = c2;
    C1* pC1 = &c2;

    cout << c1.getDescription() << endl;
    cout << c2.getDescription() << endl;
    cout << pC1->getDescription() << endl;
    pC1 = new C2();
    cout << pC1->getDescription() << endl;
}
```

46-775.cpp

```
#include <iostream>
using namespace std;

int f(int& x, int* y) {
    x = x + 2;
    int z = x;
    z = 2 * x;
    *y *= 3;
    return x* *y;
}

int main() {
```

```
13      int x = 5;
14      int y = 2;
15      cout << f(y, &x) << endl;
16      cout << x << " " << y << endl;
17      cout << f(x, &y) << endl;
18      cout << x << " " << y << endl;
19      return 0;
20  }
```

46-805.cpp

```
1   #include <iostream>
2
3   class O {
4   private:
5       std::string explanation;
6
7   public:
8       O(std::string explanation);
9       void explain();
10  };
11
12  class U : public O {
13  public:
14      U(std::string explanation);
15      void explain();
16  };
17
18  O::O(std::string explanation) : explanation(explanation) {}
19
20  void O::explain() {
21      std::cout << explanation << std::endl;
22  }
23
24  U::U(std::string explanation) : O(explanation) {}
25
26  void U::explain() {
27      std::cout << "Der Fehler ist: ";
28      O::explain();
29  }
30
31  int f(int i) {
32      if (i > 10) throw U("Argument zu groß");
33      if (i <= 0) return 1;
```

```
34      return i * f(i-1);
35  }
36
37  int main() {
38      try {
39          std::cout << f(5) << std::endl;
40          std::cout << f(-2) << std::endl;
41          std::cout << f(11) << std::endl;
42          std::cout << f(2) << std::endl;
43          std::cout << f(2) << std::endl;
44      } catch (O o) {
45          o.explain();
46      }
47      return 0;
48  }
```

46-850.cpp

```
1   #include <iostream>
2   using namespace std;
3
4   template <typename T> class Box {
5   public:
6       T content;
7       Box<T>(T& x) : content(x) {
8           x[1] = 'o';
9       }
10  };
11
12  int g(int& x) {
13      x = x + 4;
14      return x;
15  }
16
17  int f(int x) {
18      return g(x);
19  }
20
21  int main() {
22      string s = "hallo";
23      int y = 42;
24      int z = (f(y));
25      Box<string> b(s);
26      cout << s << endl;
```

```
    s[1] = 'e';
    cout << z << endl;
    cout << y << endl;
    cout << b.content << endl;
    cout << s << endl;
}
```

46-852.cpp

```
#include <iostream>
using namespace std;

class GeometricObject {
public:
    int hoehe;
    int breite;
    GeometricObject(int h, int b);
    virtual float flaeche();
    float umfang();
};

class Quadrat : public GeometricObject {
public:
    Quadrat(int h);
};

class Kreis : public Quadrat {
public:
    Kreis(int h);
    virtual float flaeche();
    virtual float umfang();
};

GeometricObject::GeometricObject(int h, int b) :
    hoehe(h), breite(b) {}

float GeometricObject::flaeche() {
    return hoehe*breite;
}

float GeometricObject::umfang() {
    return 2 * (hoehe + breite);
}

```

```
36 Quadrat::Quadrat(int h) : GeometricObject(h, h) {}
37
38 Kreis::Kreis(int h) : Quadrat(h) {}
39
40 float Kreis::flaeche() {
41     return 3.14 * breite * hoehe;
42 }
43
44 float Kreis::umfang() {
45     return 3.14 * breite;
46 }
47
48 int main() {
49     Quadrat geo0 = Quadrat(10);
50     GeometricObject& geo1 = geo0;
51     GeometricObject geo2 = Kreis(5);
52     GeometricObject* geo3 = &geo2;
53     geo0.breite = 20;
54     cout << geo1.flaeche() << endl;
55     cout << geo1.umfang() << endl;
56     cout << geo2.flaeche() << endl;
57     cout << geo2.umfang() << endl;
58     cout << geo3->flaeche() << endl;
59     cout << geo3->umfang() << endl;
60     return 0;
61 }
```

46-984.cpp

```
1  #include <string>
2  #include <iostream>
3  using namespace std;
4
5  class C1 {
6  public:
7      virtual string getDescription();
8  };
9
10 class C2 : public C1 {
11     string getDescription();
12 };
13
14 string C1::getDescription() {
15     return "C1";
```

```
}

string C2::getDescription() {
    return "C2";
}

int main() {
    C1* c1 = new C1();
    C1* c2 = new C2();

    cout << c1->getDescription() << endl;
    cout << c2->getDescription() << endl;
}
```

46-992.cpp

```
#include <iostream>

class C {
public:
    int counter;
    C();
    int next();
};

C::C() : counter(42) {}

int C::next() {
    counter = counter + 1;
    if (counter > 43) {
        counter = 0;
        throw 17;
    }
    return counter;
}

int main() {
    C c;
    try {
        std::cout << c.next() << std::endl;
        std::cout << c.next() << std::endl;
        std::cout << c.next() << std::endl;
        std::cout << c.next() << std::endl;
    } catch (std::string s) {
```

```
29          std::cout << s << std::endl;
30          std::cout << c.next() << std::endl;
31      } catch (int i) {
32          if (i == 42) {
33              std::cout << c.next() << std::endl;
34              std::cout << c.next() << std::endl;
35          } else {
36              std::cout << "unknown error: " << i << std::endl;
37              std::cout << c.next() << std::endl;
38          }
39      }
40  }
```

4 Eigenen Quelltext schreiben (C++)

57-002

Gegeben sei folgende Template-Klasse, die einen binären Suchbaum realisiert:

treenode.hpp

```
1  template <typename T> class TreeNode {
2  private:
3      T wert;
4      TreeNode<T>* leftChild;
5      TreeNode<T>* rightChild;
6  public:
7      TreeNode(T w) : wert(w), leftChild(0), rightChild(0) {}
8  };
```

Ergänzen sie die Klasse um die Methode `void preorder(std::vector<T>& result) const;` die die im Baum gespeicherten Werte in Präorder-Reihenfolge in `result` einfügt.

57-056

Schreiben Sie folgende Funktion im Stil der Algorithmen der Standardbibliothek:

57-056.hpp

```
1  template <typename Iterator, typename ElementType>
2  void wendeAn(Iterator anfang, Iterator ende,
3      void (*f)(ElementType));
```

Die Funktion soll die übergebene Funktion `f` auf alle Elemente, über die iteriert wird, anwenden. Schreiben Sie einen Test, in dem Sie die Funktion aufrufen.

57-100

Gegeben sei folgende Template-Klasse, die einen binären Suchbaum realisiert:

treenode.hpp

```
template <typename T> class TreeNode {
private:
    T wert;
    TreeNode<T>* leftChild;
    TreeNode<T>* rightChild;
public:
    TreeNode(T w) : wert(w), leftChild(0), rightChild(0) {}
};
```

Ergänzen sie die Klasse um die Methode `void insert(TreeNode<T>* node);` die den übergebenen Knoten korrekt sortiert in den Baum einfügt (ggf. vermittels eines rekursiven Aufrufs). Schreiben Sie dann die Methode `void insert(T wert);` um komfortableres Einfügen zu ermöglichen.

57-176

Schreiben Sie folgende Funktion im Stil der Algorithmen der Standardbibliothek:

57-176.hpp

```
template <typename Iterator, typename Key, typename Value>
Value lookUp(Iterator begin, Iterator end, Key key,
    Value default);
```

Dabei verweisen die Iteratoren `begin` und `end` auf eine Folge von Objekten des Typs `std::pair<Key,Value>`, die eine Abbildung von Schlüsseln auf Werte speichern. Die Klasse `std::pair` ist eine Template-Klasse mit öffentlichen Feldern `first` und `second`, deren Typen durch die Template-Parameter festgelegt werden. Die Funktion `lookUp` soll über die `std::pair<Key,Value>`-Objekte iterieren und jeweils das Feld `first` mit dem gesuchten Schlüssel `key` vergleichen; bei Übereinstimmung soll der Wert von `second` zurückgegeben wird. Falls keines der Paare den gesuchten Schlüssel enthält soll `default` zurückgegeben werden.

57-304

Schreiben Sie folgende Funktion im Stil der Algorithmen der Standardbibliothek:

57-304.hpp

```
template <typename Iterator, typename ElementType>
void drop(Iterator begin, Iterator end, int i,
    std::vector<ElementType>& result);
```

`drop` soll alle Elemente des Iteratorbereichs mit Ausnahme der `i` ersten in den Vektor `result` einfügen. Schreiben Sie zwei Beispielanwendungen der Funktion `drop`, einmal mit einem Iteratorbereich über einem Array, einmal über einer Deque.

57-405

Schreiben Sie folgende Funktion im Stil der Algorithmen der Standardbibliothek:

57-405.hpp

```
template <typename Iterator>
void printAll(Iterator anfang, Iterator ende);
```

Die Funktion soll alle Elemente des Iterationsbereichs auf die Konsole ausdrucken. Schreiben Sie einen Test, in dem Sie die Funktion aufrufen.

57-491

Schreiben Sie folgende Funktion im Stil der Algorithmen der Standardbibliothek:

57-491.hpp

```
temnplate <typename Iterator, typename ElementType>
void reverse(Iterator anfang, Iterator ende,
    std::deque<ElementType>& result);
```

Die Funktion soll alle Elemente des Iterationsbereichs in umgekehrter Reihenfolge in die übergebene Deque einfügen.

57-524

Schreiben Sie folgende Funktion im Stil der Algorithmen der Standardbibliothek:

57-524.hpp

```
template <typename Iterator, typename ElemType>
ElemType groesstes(Iterator begin, Iterator end,
    bool(*groesser)(ElemType, ElemType));
```

`groesstes` soll das nach der übergebenen Vergleichsfunktion größte Element im Iteratorbereich zurückgeben. Falls der Iteratorbereich kein Element enthält, soll die Funktion eine Ausnahme werfen. Schreiben Sie eine Beispielanwendung Ihrer Funktion, die für eine Sammlung von Strings den längsten dieser Strings ermittelt.

57-544

Gegeben sei folgende Template-Klasse, die einen binären Suchbaum realisiert:

treenode.hpp

```
template <typename T> class TreeNode {
private:
    T wert;
    TreeNode<T>* leftChild;
    TreeNode<T>* rightChild;
public:
    TreeNode(T w) : wert(w), leftChild(0), rightChild(0) {}
};
```

Ergänzen sie die Klasse um die Methode `void postorder(std::vector<T>& result) const;` die die im Baum gespeicherten Werte in Postorder-Reihenfolge in `result` einfügt.

57-587

Schreiben Sie folgende Funktion im Stil der Algorithmen der Standardbibliothek:

57-587.hpp

```
template <typename Iterator, typename ElementType>
int containsHowOften(Iterator anfang, Iterator ende,
    ElementType element);
```

Die Funktion soll zurückgeben, wie oft das übergebene Element im Iterationsbereich enthalten ist. Schreiben Sie einen Test, in dem Sie die Funktion aufrufen.

57-589

Schreiben Sie folgende Funktion im Stil der Algorithmen der Standardbibliothek:

57-589.hpp

```
template <typename Iterator, typename ElementType>
void fuerAlle(Iterator anfang, Iterator ende,
    ElementType (*f)(ElementType));
```

Die Funktion soll auf alle Elemente des Iterationsbereichs die Funktion `f` anwenden und die Elemente durch das Ergebnis dieser Funktionsanwendung ersetzen. Schreiben Sie einen Test, in dem Sie die Funktion aufrufen.

57-628

Schreiben Sie folgende Funktion im Stil der Algorithmen der Standardbibliothek:

57-628.hpp

```
template <typename Iterator, typename Value>
void values(Iterator begin, Iterator end,
    std::vector<Value>& result);
```

Dabei verweisen die Iteratoren `begin` und `end` auf eine Folge von Objekten des Typs `std::pair<Key,Value>`, die eine Abbildung von Schlüsseln auf Werte speichern. Die Klasse `std::pair` ist eine Template-Klasse mit öffentlichen Feldern `first` und `second`, deren Typen durch die Template-Parameter festgelegt werden. Die Funktion `values` soll über die `std::pair<Key,Value>`-Objekte iterieren und jeweils den Wert des Felds `second` in `result` speichern.

57-748

Gegeben sei folgende Template-Klasse, die einen binären Suchbaum realisiert:

treenode.hpp

```
template <typename T> class TreeNode {
private:
    T wert;
    TreeNode<T>* leftChild;
    TreeNode<T>* rightChild;
public:
    TreeNode(T w) : wert(w), leftChild(0), rightChild(0) {}
};
```

Schreiben Sie einen geeigneten Destruktor für die Klasse.

57-764

Schreiben Sie folgende Funktion im Stil der Algorithmen der Standardbibliothek:

57-764.hpp

```
template <typename Iterator, typename ElementType>
ElementType verknuepfe(Iterator anfang, Iterator ende,
    ElementType (*f)(ElementType, ElementType),
    ElementType startwert);
```

Die Funktion soll mit der übergebenen Funktion `f` alle Elemente des Iterationsbereichs aufsummieren, d.h. für eine Iteration über $x_1, x_2, \ldots, x_n$ soll das Ergebnis zurückgegeben werden, das $f(\ldots f(f(startwert, x_1), x2), \ldots, x_n)$ entspricht.

57-840

Gegeben sei folgende Template-Klasse, die einen binären Suchbaum realisiert:

treenode.hpp

```
template <typename T> class TreeNode {
private:
    T wert;
    TreeNode<T>* leftChild;
    TreeNode<T>* rightChild;
public:
    TreeNode(T w) : wert(w), leftChild(0), rightChild(0) {}
};
```

Ergänzen sie die Klasse um die Methode `bool isLeaf() const;` die genau dann *true* zurückgibt wenn der Knoten keine Kinder hat.

57-845

Gegeben sei folgende Template-Klasse, die einen binären Suchbaum realisiert:

treenode.hpp

```
template <typename T> class TreeNode {
private:
    T wert;
    TreeNode<T>* leftChild;
    TreeNode<T>* rightChild;
public:
    TreeNode(T w) : wert(w), leftChild(0), rightChild(0) {}
};
```

Ergänzen sie die Klasse um die Methode `bool contains(T t) const;` die zurückgibt ob der Baum den Wert `t` enthält. Nutzen Sie dabei die Sortierung des Baums, um unnötigen Suchaufwand zu vermeiden.

57-856

Schreiben Sie folgende Funktion im Stil der Algorithmen der Standardbibliothek:

57-856.hpp

```
template <typename Iterator, typename ElementType>
boolean contains(Iterator anfang, Iterator ende,
    ElementType element);
```

Die Funktion soll zurückgeben, ob das übergebene Element im Iterationsbereich enthalten ist. Schreiben Sie einen Test, in dem Sie die Funktion aufrufen.

57-943

Gegeben sei folgende Template-Klasse, die einen binären Suchbaum realisiert:

treenode.hpp

```
template <typename T> class TreeNode {
private:
    T wert;
    TreeNode<T>* leftChild;
    TreeNode<T>* rightChild;
public:
    TreeNode(T w) : wert(w), leftChild(0), rightChild(0) {}
};
```

Ergänzen sie die Klasse um die Methode `void inorder(std::vector<T>& result) const;` die die im Baum gespeicherten Werte in Inorder-Reihenfolge in `result` einfügt.

57-949

Gegeben sei folgende Template-Klasse, die einen binären Suchbaum realisiert:

treenode.hpp

```
template <typename T> class TreeNode {
private:
    T wert;
    TreeNode<T>* leftChild;
    TreeNode<T>* rightChild;
public:
    TreeNode(T w) : wert(w), leftChild(0), rightChild(0) {}
};
```

Ergänzen sie die Klasse um die Methode `T max() const;` die den größten Wert im Baum zurückgibt.

57-969

Gegeben sei folgende Template-Klasse, die einen binären Suchbaum realisiert:

treenode.hpp

```
template <typename T> class TreeNode {
private:
    T wert;
    TreeNode<T>* leftChild;
    TreeNode<T>* rightChild;
public:
    TreeNode(T w) : wert(w), leftChild(0), rightChild(0) {}
```

```
};
```

Ergänzen sie die Klasse um die Methode `int count() const;` die die Anzahl der Knoten des im Knoten gewurzelten Baumes zurückgibt.

57-996

Gegeben sei folgende Template-Klasse, die einen binären Suchbaum realisiert:

treenode.hpp

```
template <typename T> class TreeNode {
private:
    T wert;
    TreeNode<T>* leftChild;
    TreeNode<T>* rightChild;
public:
    TreeNode(T w) : wert(w), leftChild(0), rightChild(0) {}
};
```

Ergänzen sie die Klasse um die Methode `int height() const;` die die Höhe des im Knoten gewurzelten Baums zurückgibt.

5 Wissensfragen

79-053 Was bewirkt in Java das Schlüsselwort `volatile`? Für welche Situationen eignet es sich, für welche nicht?

79-146 Welchen Unterschied macht es in C++, ob man im Konstruktor den Feldern eines Objekts ihre Werte in der Initialisierungsliste oder im Rumpf des Konstruktors zuweist?

79-156 Was ist bei der Programmierung überladener Zuweisungsoperateren in C++ zu beachten?

79-173 Warum sollten Methoden, die das Objekt nicht verändern, in C++ als `const` deklariert werden?

79-279 Was passiert technisch, wenn ein Objekt in C++ nicht als Pointer oder Referenz an eine Methode übergeben wird? Was folgt daraus für die Polymorphie?

79-290 Das Sternsymbol `*` hat in C++ mehrere Bedeutungen. Welche?

79-317 Das Exception-Handling von C++ kennt (im Gegensatz zu Java) keine `finally`-Klausel. Warum wird sie nicht benötigt?

79-329 Was versteht man in C++ unter einem Default-Konstruktor?

79-352 Was versteht man bei C++ unter RAII (resource acquisition is initialization)? Inwiefern vereinfacht es das Schreiben korrekten Codes? Warum kann das Konzept in Java nicht angewendet werden?

79-354 Was versteht man bei C++ unter dem Name Mangling? Warum ist es notwendig?

79-439 Was versteht man in C++ unter einem Smart Pointer?

79-516 Welcher Zusammenhang besteht in C++ zwischen Destruktor, Kopierkonstruktor und Zuweisungsoperator?

79-533 Wenn Sie in C++ eine Methode nicht als `virtual` markieren, findet für diese kein Late Binding statt. Worin kann dabei der Vorteil liegen?

79-555 Auf welchem Konzept der Programmiersprache C basieren die Iteratoren der Standardbibliothek von C++?

79-583 Welche Arten der Instanziierung von Templates gibt es in C++? Wie unterscheiden sie sich in der Organisation der Quelltexte, und welche Vor- und Nachteile resultieren daraus?

79-593 Warum benötigt C++ im Gegensatz zu Java kein gesondertes Konzept für Interfaces?

79-634 Warum sollte in C++ der Datentyp einer `catch`-Klausel immer ein Referenztyp sein?

79-659 Was passiert zur Laufzeit beim Aufruf einer virtuellen Methode in einem C++-Programm?

79-686 Wann sollte ein Destruktor in C++ als `virtual` markiert werden?

79-687 Wie kann das Schlüsselwort `explicit` in C++ verwendet werden, und was bewirkt es? Wann ist sein Einsatz sinnvoll?

79-720 Java kennt das Schlüsselwort `instanceof`. Was ist das Äquivalent dazu in C++?

79-723 Welche Voraussetzungen müssen erfüllt sein, damit Polymorphie bei einem Methodenaufruf in C++ zum Tragen kommen kann?

79-785 Welche Datentypen können in C++ als Exception geworfen werden?

79-834 Was sagen Sie zu der Idee, Konstruktoren in C++ mit `virtual` zu attributieren?

79-856 Nennen Sie ein Beispiel für Operatorüberladung in der Standardbibliothek von C++.

79-880 Wie werden in C++ Template-Klassen vom Compiler übersetzt? Wie unterscheidet sich dies von den Generics in Java?

79-886 Das Und-Zeichen `&` hat in C++ mehrere Bedeutungen. Welche?

79-895 Welche Operationen unterstützen alle Iteratoren der Standardbibliothek von C++?

79-947 Aus welchen Schritten besteht das Übersetzen eines C++-Quelltexts zu einem ausführbaren Programm? Was passiert bei den einzelnen Schritten?

79-950 Welchen Zweck haben Kopierkonstruktoren in C++? Wann kommen sie zum Einsatz? Worauf ist bei ihrer Programmierung zu achten?

79-969 Wann ist es in C++ nötig, für eine Klasse einen Destruktor explizit zu programmieren?