```
1  // Time-stamp: "24.02.05 19:14 ausgabe.cpp klaus@wachtler.de"
2
3  #include <iostream>
4  #include <ostream>
5
6  class A;
7  inline std::ostream &operator<<( std::ostream &s, const A &obj );
8
9  class A
10 {
11
12
13 public:
14
15   // Konstruktor
16   A()
17     : ersterwert( 0 )
18   {
19     std::cerr << "ctor A()" << std::endl;
20   }
21
22   A( int ersterwert )
23     : ersterwert( ersterwert )
24   {
25     std::cerr << "ctor A( " << ersterwert << " )"
26               << std::endl;
27   }
28
29   // copy-Konstruktor
30   A( const A &rechteSeite )
31     : ersterwert( rechteSeite.ersterwert )
32   {
33     std::cerr << "copy-ctor A( const A &rechteSeite )"
34               << " rechteSeite.ersterwert = "
35               << rechteSeite.ersterwert
36               << std::endl;
37   }
38
39   // Destruktor
40   virtual ~A()
41   {
42     std::cerr << "dtor ~A(), " << ersterwert << std::endl;
43   }
44
45   // Zuweisung A = A
46   // Rückgabewert ist eine REFERENZ auf die linke Seite
47   // (nicht const), und damit wieder als lvalue verwendbar.
48   A &operator=( const A &rechteSeite )
49   {
50     std::cerr << "Zuweisung A=A, ls=" << ersterwert
51               << ", rs=" << rechteSeite.ersterwert
52               << std::endl;
53
54     ersterwert = rechteSeite.ersterwert;
55     return *this;
56   }
57
58   // Ausgabe auf einen ostream
59   virtual void streamout( std::ostream & s ) const
60   {
61     s << "A=" << ersterwert;
62   }
63
64
65
66
67
```

```
68    // Kopie von sich selbst allokieren
69    virtual A *clone() const
70    {
71      std::cout << "A::clone()" << std::endl;
72      return new A(*this);
73    }
74
75    virtual void tuwas() const
76    {
77      std::cout << "A::tuwas: " << *this << std::endl;
78    }
79
80    void tuwasanderes() const
81    {
82      std::cout << "A::tuwasanderes: " << *this << std::endl;
83    }
84
85  protected:
86
87    // Membervariable (in den abgeleiteten Klassen kommen noch welche
88    // dazu):
89    int ersterwert;
90
91  }; // class A...
92
93
94  inline std::ostream &operator<<( std::ostream &s, const A &obj )
95  {
96    obj.streamout( s );
97    return s;
98  }
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
```

```
135 class B: public A
136 {
137 public:
138
139   // Konstruktor
140   B()
141     : A(),
142       zweiterwert( 0 )
143   {
144     std::cerr << "ctor B()" << std::endl;
145   }
146
147   B( int ersterwert, int zweiterwert )
148     : A( ersterwert ),
149       zweiterwert( zweiterwert )
150   {
151     std::cerr << "ctor B( "
152               << ersterwert
153               << ", "
154               << zweiterwert
155               << " )"
156               << std::endl;
157   }
158
159   // copy-Konstruktor
160   B( const B &rechteSeite )
161     : A( rechteSeite.ersterwert ),
162       zweiterwert( rechteSeite.zweiterwert )
163   {
164     std::cerr << "copy-ctor B( const B &rechteSeite )"
165               << "\n  rechteSeite.ersterwert = "
166               << rechteSeite.ersterwert
167               << ",\n  rechteSeite.zweiterwert = "
168               << rechteSeite.zweiterwert
169               << std::endl;
170   }
171
172   // Destruktor
173   virtual ~B()
174   {
175     std::cerr << "dtor ~B()"
176               << ", ersterwert=" << ersterwert
177               << ", zweiterwert=" << zweiterwert
178               << std::endl;
179   }
180
181   // Zuweisung B = B
182   // Rückgabewert ist eine REFERENZ auf die linke Seite
183   // (nicht const), und damit wieder als lvalue verwendbar.
184   B & operator=( const B &rechteSeite )
185   {
186     std::cerr << "Zuweisung B=B"
187               << ",\n  ls.ersterwert=" << ersterwert
188               << ",\n  ls.zweiterwert=" << zweiterwert
189               << ",\n  rs.ersterwert=" << rechteSeite.ersterwert
190               << ",\n  rs.zweiterwert=" << rechteSeite.zweiterwert
191               << std::endl;
192
193     this->A::operator=( rechteSeite );
194     zweiterwert = rechteSeite.zweiterwert;
195
196     return *this;
197   }
198
199
200
201
```

```
202   // Ausgabe auf einen ostream
203   virtual void streamout( std::ostream & s ) const
204   {
205     s << "B=";
206     A::streamout( s );
207     s << ", "  << zweiterwert;
208   }
209
210   // Kopie von sich selbst allokieren
211   virtual B *clone() const
212   {
213     std::cout << "B::clone()" << std::endl;
214     return new B(*this);
215   }
216
217   virtual void tuwas() const
218   {
219     std::cout << "B::tuwas: " << *this << std::endl;
220   }
221
222   void tuwasanderes() const
223   {
224     std::cout << "B::tuwasanderes: " << *this << std::endl;
225   }
226
227 protected:
228
229   // Membervariable
230   int zweiterwert;
231
232 }; // class B...
233
234 ////////////////////////////////////////////////////////////////////
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
```

```
269 // Testprogramm
270
271 void unterprogramm1( const A eina )
272 {
273   eina.tuwas();
274   eina.tuwasanderes();
275   std::cout << "unterprogramm1: eina = " << eina << std::endl;
276 }
277
278 void unterprogramm2( const A &eina )
279 {
280   eina.tuwas();
281   eina.tuwasanderes();
282   std::cout << "unterprogramm2: eina = " << eina << std::endl;
283   throw  "oioioi!";
284 }
285
286 #include <string>
287
288
289 int main( int nargs, char **args )
290 {
291   try
292   {
293     B einb( 30, 40 );
294
295     std::cout << "unterprogramm1:" << std::endl;
296     unterprogramm1( einb );
297
298     A *einanderesb_p = new B( einb );
299
300     std::cout << "unterprogramm2:" << std::endl;
301     unterprogramm2( *einanderesb_p );
302
303     delete einanderesb_p;
304   }
305   catch( std::string  fehler )
306   {
307     std::cerr << "Fehler in main(): " << fehler << "\n";
308   }
309   catch( const char *fehler )
310   {
311     std::cerr << "anderer Fehler in main(): " << fehler << "\n";
312   }
313   catch( ... )
314   {
315     std::cerr << "unbekannter Fehler in main()\n";
316   }
317 } // main( int nargs, char **args )
end
```