Konstruktoren und Destruktoren
Initialisierung von Objekten

Breymann [15] Abschn. 3.3, S. 161ff

Nicht explizit in der Vorlesung behandelt
-- zum Selbststudium --

Software-Technik: Vom Programmierer zur erfolgreichen …

1. Von der Idee zur Software
2. Funktionen und Datenstrukturen
3. Organisation des Quellcodes
4. Werte- und Referenzsemantik
5. Entwurf von Klassen
6. Fehlersuche
7. Software-Entwicklung im Team
8. Abstrakte Datentypen: Einheit von Daten und Funktionalität
9. Vielgestaltigkeit (Polymorphie)
10. Entwurfsprinzipien für Software

Lehrbuch: 4.1
Kompendium, 3. Auflage: 8.10, 9.1
Kompendium, 4. Auflage: 8.1 ... 8.17

Anhang A: Die Familie der C-Sprachen

Anhang B: Grundlagen der C++ und der Java-Programmierung

Konstruktoren

Konstruktoren haben keinen Return-Wert, auch nicht void.

Methoden sorgen dafür, dass sich das Objekt immer in einem konsistenten Zustand befindet, was bei direktem Zugriff auf die Attribute nicht garantiert werden könnte.

Konstruktoren sorgen dafür, dass sich das Objekt auch bei der Entstehung in einem konsistenten/korrekten Zustand befindet.

Header Vektor.h

```
class  Vektor {
public:
Vektor(int s);
~Vektor();
private:
double* data;
int size;
};
```

Source Vektor.cxx

```
Vektor::Vektor(int s) {
   size = s;
   data = new double[s];
}

Vektor::~Vektor() {
   delete [] data;
   data = nullptr;
}
```

Standardkonstruktoren

Besitzt eine Klasse keinen Konstruktor, so erzeugt das System automatisch einen sogenannten Standardkonstruktor, der keine Aktion durchführt (außer die Standardkonstruktoren, der Attribute aufzurufen.

Die eingebauten Typen, wie double oder int besitzen keinen „echten“ Standardkonstruktor, sodass sie undefinierte Werte haben, obwohl manche Compiler sie mit 0 initialisieren. Man sollte sich aber nie darauf verlassen.

```
class  X {
private:
int size;          // undefinierter Wert
Guest g;           // Standardkonstruktor von Guest wird aufgerufen,
                   // muss vorhanden sein, sonst Compilerfehler
};
```

Standardkonstruktoren

Arrays können nur für Objekte erzeugt werden, die einen Standardkonstruktoren besitzen, unabhängig davon, ob automatisch vom System oder vom Entwickler spezifiziert.

```
class  X { public:   X();
};
class  Y { public:   Y(int);
};
class  Z { public:
   void print(int);
};
```

```
X arrX[10];  // O.K.
Y arrY[10];  // Syntaxfehler
Z arrZ[10];   // O.K.

vector<X> vecX; // O.K.
vector<Y> vecY; // O.K.

vector<X> vecX1(10); // O.K.
vector<Y> vecY1(44); // Fehler
```

Standardkonstruktoren mit Initialisierungslisten

```
class  Y {
public:   Y(int i) { k = i; }
      int k;
};
```

```
Y arrY[3]{ 0,1,3 };  // O.K.
   // zu wenig Argumente Syntaxfehler
   //Y arrY1[4]{ 0,1,3 };
   // zu viele Argumente Syntaxfehler
   //Y arrY2[4]{ 0,1,3,5,7 };

//   vector<Y> vecY(3) { 1,4,6 }; // Fehler

    Y* py = new Y[4]{ 4,3,5,6 }; // O.K.
   // zu wenig Argumente Syntaxfehler
   //Y* py1 = new Y[4]{ 4,3,5 };
```

Default-Initialisierungen

```
class  Vektor {
public:
Vektor() {defWert=0;}
Vektor(int s) {}
~Vektor();
int GetDefWert(){ return defWert;}

private:
int defWert=23;
};
```

```
int main() {
Vektor v(13);

cout << v.GetDefWert()
        << endl;
}
```

Ausgabe 23

Initialisierungen mit Initialisierungslisten

```
class  Vektor {
public:
double GetDefWert(){ return defWert;}

public:
double* data;
int size;
double defWert;
};
```

```
int main() {
Vektor v1 { nullptr, 16, 11.3 };
cout << v1.GetDefWert();}
```

Ausgabe 11.3

Clicker: Wie ist die Ausgabe?

```
class Vektor {
double GetDefWert(){ return defWert; }

private:
double* data;
int size;
double defWert = 14;
};
```

```
int main() {
Vektor v1 { nullptr, 13, 46.6};

cout << v1.GetDefWert();}
```

Bildschirmausgabe ?

1. 46.6
2. 13
3. Compilerfehler
4. 14

Clicker: Wie ist die Ausgabe?

```
class Vektor {
double GetDefWert(){ return defWert; }

private:
double* data;
int size;
double defWert = 14;
};
```

```
int main() {
Vektor v1 { nullptr, 13, 46.6};

cout << v1.GetDefWert();}
```

Bildschirmausgabe ?

1. 46.6
2. 13
3. Compilerfehler
4. 14

Bildschirmausgabe ?

1. 46.6
2. 13
3. Compilerfehler
4. 14

Clicker: Wie ist die Ausgabe?

```
class Vektor {
public:
Vektor(int s, double defWert);
double GetDefWert(){ return defWert; }

private:
double* data;
int size;
double defWert ;
};
```

```
int main() {
Vektor v1 { nullptr, 13, 46.6};

cout << v1.GetDefWert();
}
```

Bildschirmausgabe ?

1. 46.6
2. 13
3. Compilerfehler
4. 67

Clicker: Wie ist die Ausgabe?

```
class Vektor {
public:
Vektor(int s, double defWert);
double GetDefWert(){ return defWert; }

private:
double* data;
int size;
double defWert ;
};
```

```
int main() {
Vektor v1 { nullptr, 13, 46.6};

cout << v1.GetDefWert();
}
```

Bildschirmausgabe ?

1. 46.6
2. 13
3. Compilerfehler, Attribute sind private
4. 67

Bildschirmausgabe ?

1. 46.6
2. 13
3. Compilerfehler
4. 67

Clicker: Wie ist die Ausgabe?

```
class Vektor {
public:
double GetDefWert(){ return defWert; }

public:
double* data;
int size;
double defWert ;
};
```

```
int main() {
Vektor v1 {13, 46.6};

cout << v1.GetDefWert();
}
```

Bildschirmausgabe ?

1. 14
2. 46.6
3. Compilerfehler
4. 67

Clicker: Wie ist die Ausgabe?

```
class Vektor {
public:
double GetDefWert(){ return defWert; }

public:
double* data;
int size;
double defWert ;
};
```

```
int main() {
Vektor v1 {13, 46.6};

cout << v1.GetDefWert();
}
```

Bildschirmausgabe ?
1. 14
2. 46.6
3. Compilerfehler, 13 kein Zeiger
4. 67

Bildschirmausgabe ?
1. 14
2. 46.6
3. Compilerfehler
4. 67

Clicker: Wie ist die Ausgabe?

```
class Vektor {
public:
double GetDefWert(){ return defWert; }

private:
double* data;
int size;
double defWert ;
};
```

```
int main() {
Vektor v1 { nullptr, 13};

cout << v1.GetDefWert();
}
```

Bildschirmausgabe ?

1. 13
2. 46.6
3. Compilerfehler
4. undefiniert

Clicker: Wie ist die Ausgabe?

```
class Vektor {
public:
double GetDefWert(){ return defWert; }

private:
double* data;
int size;
double defWert ;
};
```

```
int main() {
Vektor v1 { nullptr, 13};

cout << v1.GetDefWert();
}
```

Bildschirmausgabe ?
1. 13
2. 46.6
3. Compilerfehler, Member private
4. Undefiniert bzw. 0

Bildschirmausgabe ?
1. 13
2. 46.6
3. Compilerfehler
4. undefiniert

Clicker: Wie ist die Ausgabe?

```
class Vektor {
public:
double GetDefWert(){ return defWert; }

public:
double* data;
int size;
double defWert ;
};
```

```
int main() {
Vektor v1 { nullptr, 13};

cout << v1.GetDefWert();
}
```

Bildschirmausgabe ?

1. 13
2. 46.6
3. Compilerfehler
4. undefiniert

Clicker: Wie ist die Ausgabe?

```
class Vektor {
public:
double GetDefWert(){ return defWert; }

public:
double* data;
int size;
double defWert ;
};
```

```
int main() {
Vektor v1 { nullptr, 13};

cout << v1.GetDefWert();
}
```

Bildschirmausgabe ?
1. 13
2. 46.6
3. Compilerfehler, nur 2 Parameter
4. Undefiniert bzw. 0

Bildschirmausgabe ?
1. 13
2. 46.6
3. Compilerfehler
4. undefiniert

Clicker: Wie ist die Ausgabe?

```
class VektorSt {
public:
double GetDefWert()
    { return defWert; }
public:
double* data;
int size;
double defWert;
};
```

```
int main() {
VektorSt vst2{ 24, 16, 2.34 };
cout << vst2.GetDefWert() << endl;}
```

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler
4. undefiniert

Clicker: Wie ist die Ausgabe?

```
class VektorSt {
public:
double GetDefWert()
    { return defWert; }
public:
double* data;
int size;
double defWert;
};
```

```
int main() {
VektorSt vst2{ 24, 16, 2.34 };
cout << vst2.GetDefWert() << endl;}
```

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler
4. undefiniert

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler; int kann nicht in double* konvertiert werden
4. undefiniert

Clicker: Wie ist die Ausgabe?

```
class VektorSt {
public:
double GetDefWert()
    { return defWert; }
public:
double* data;
int size;
double defWert;
};
```

```
int main() {
VektorSt vst2{ 24, 16, 2.34 };
cout << vst2.GetDefWert() << endl;}
```

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler
4. undefiniert

Clicker: Wie ist die Ausgabe?

```
class VektorSt {
public:
double GetDefWert()
    { return defWert; }
public:
double* data;
int size;
double defWert;
};
```

```
int main() {
VektorSt vst2{ 24, 16, 2.34 };
cout << vst2.GetDefWert() << endl;}
```

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler
4. undefiniert

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler; int kann nicht in double* konvertiert werden
4. undefiniert

Clicker: Wie ist die Ausgabe?

```
class VektorSt {
public:
double GetDefWert()
    { return defWert; }
public:
double* data;
int size;
double defWert;
};
```

```
int main() {
VektorSt vst2{ 24, 16, 2.34 };
cout << vst2.GetDefWert() << endl;}
```

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler
4. Compiler Warnung

Clicker: Wie ist die Ausgabe?

```
class VektorSt {
public:
double GetDefWert()
    { return defWert; }
public:
double* data;
int size;
double defWert;
};
```

```
int main() {
VektorSt vst2{ 24, 16, 2.34 };
cout << vst2.GetDefWert() << endl;}
```

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler
4. Compiler Warnung

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler; int kann nicht in double* konvertiert werden
4. Compiler Warnung

Clicker: Wie ist die Ausgabe?

```
class VektorSt {
public:
double GetDefWert()
    { return defWert; }
public:
double* data;
int size;
double defWert;
};
```

```
int main() {
VektorSt vst3{ nullptr, 16.4, 2 };
cout << vst3.GetDefWert() << endl;}
```

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler
4. Compiler Warnung

Clicker: Wie ist die Ausgabe?

```
class VektorSt {
public:
double GetDefWert()
    { return defWert; }
public:
double* data;
int size;
double defWert;
};
```

```
int main() {
VektorSt vst3{ nullptr, 16.4, 2 };
cout << vst3.GetDefWert() << endl;}
```

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler
4. Compiler Warnung

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler;
4. Compiler Warnung: möglicher Datenverlust bei Konvertierung von double nach int

Clicker: Wie ist die Ausgabe?

```
class VektorSt {
public:
double GetDefWert()
    { return defWert; }
public:
double* data;
int size;
double defWert;
};
```

```
int main() {
VektorSt vst2{
   reinterpret_cast<double*>(24), 16, 2.34 };
cout << vst2.GetDefWert() << endl;}
```

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler
4. undefiniert

Clicker: Wie ist die Ausgabe?

```
class VektorSt {
public:
double GetDefWert()
    { return defWert; }
public:
double* data;
int size;
double defWert;
};
```

```
int main() {
VektorSt vst2{
   reinterpret_cast<double*>(24), 16, 2.34 };
cout << vst2.GetDefWert() << endl;}
```

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler
4. undefiniert

Bildschirmausgabe ?

1. 2
2. 2.34
3. Compilerfehler
4. undefiniert

Delegierender Konstruktor

```
class  DgtVec {
public:
DgtVec();
DgtVec(double dW);
DgtVec(int s, double dW);
double GetW(){return d;}
private:
int size; double d;
};
```

```
DgtVec::DgtVec() :
    DgtVec(8.2)   {  d = 1.3;  }
DgtVec::DgtVec(double dW) :
    DgtVec(6, dW) {
       size = 4; d = 5.2; }
DgtVec::DgtVec(int s, double dW){
   size = s;  d = dW; }
int main() {
DgtVec v(2.4);
cout << v.GetW() << endl; }
```

Bildschirmausgabe ?

1. Compilerfehler
2. 5.2
3. 2.4
4. 1.3

Delegierender Konstruktor

```
class  DgtVec {
public:
DgtVec();
DgtVec(double dW);
DgtVec(int s, double dW);
double GetW(){return d;}
private:
int size; double d;
};
```

```
DgtVec::DgtVec() :
    DgtVec(8.2)   {  d = 1.3;  }
DgtVec::DgtVec(double dW) :
    DgtVec(6, dW) {
       size = 4; d = 5.2; }
DgtVec::DgtVec(int s, double dW):
   size(s), d(dw) {}
int main() {
DgtVec v(2.4);
cout << v.GetW() << endl; }
```

Bildschirmausgabe ?

1. Compilerfehler
2. 5.2
3. 2.4
4. 1.3

Delegierender Konstruktor

```
class  DgtVec {
public:
DgtVec();
DgtVec(double dW);
DgtVec(int s, double dW);
double GetW(){return d;}
private:
int size; double d;
};
```

```
DgtVec::DgtVec() :
    DgtVec(8.2)   {  d = 1.3;  }
DgtVec::DgtVec(double dW) :
    DgtVec(6, dW) {
       size = 4; d = 5.2; }
DgtVec::DgtVec(int s, double dW){
   size = s;  d = dW; }
int main() {
DgtVec v(2.4);
cout << v.GetW() << endl; }
```

Bildschirmausgabe ?

1. Compilerfehler
2. 5.2
3. 2.4
4. 1.3

Bildschirmausgabe ?

1. Compilerfehler
2. 5.2
3. 2.4
4. 1.3

Delegierender Konstruktor

```
class  DgtVec {
public:
DgtVec();
DgtVec(double dW);
DgtVec(int s, double dW);
double GetW(){return d;}
private:
int size; double d;
};
```

```
DgtVec::DgtVec() :
    DgtVec(8.2)   {  d = 1.3;  }
DgtVec::DgtVec(double dW) :
    DgtVec(6, dW) {
       size = 4; d = 5.2; }
DgtVec::DgtVec(int s, double dW){
   size = s;  d = dW; }
int main() {
DgtVec v();
 cout << v.GetW() << endl; }
```

Bildschirmausgabe ?

1. Compilerfehler
2. 5.2
3. 2.4
4. 1.3

Delegierender Konstruktor

```
class  DgtVec {
public:
DgtVec();
DgtVec(double dW);
DgtVec(int s, double dW);
double GetW(){return d;}
private:
int size; double d;
};
```

```
DgtVec::DgtVec() :
    DgtVec(8.2)   {  d = 1.3;  }
DgtVec::DgtVec(double dW) :
    DgtVec(6, dW) {
       size = 4; d = 5.2; }
DgtVec::DgtVec(int s, double dW){
   size = s;  d = dW; }
int main() {
DgtVec v();
 cout << v.GetW() << endl; }
```

Bildschirmausgabe ?

1. Compilerfehler, v ist eine Funktion mit Rückgabewert DgtVec
2. 5.2
3. 2.4
4. 1.3

Bildschirmausgabe ?

1. Compilerfehler
2. 5.2
3. 2.4
4. 1.3

Delegierender Konstruktor

```
class  DgtVec {
public:
DgtVec();
DgtVec(double dW);
DgtVec(int s, double dW);
double GetW(){return d;}
private:
int size; double d;
};
```

```
DgtVec::DgtVec() :
   DgtVec(8.2)   {  d = 1.3;  }
DgtVec::DgtVec(double dW) :
   DgtVec(6, dW) {
      size = 4; d = 5.2; }
DgtVec::DgtVec(int s, double dW){
  size = s;  d = dW; }
int main() {
DgtVec v2;
 cout << v2.GetW() << endl; }
```

Bildschirmausgabe ?

1. Compilerfehler
2. 5.2
3. 2.4
4. 1.3

Delegierender Konstruktor

```
class  DgtVec {
public:
DgtVec();
DgtVec(double dW);
DgtVec(int s, double dW);
double GetW(){return d;}
private:
int size; double d;
};
```

```
DgtVec::DgtVec() :
    DgtVec(8.2)   {  d = 1.3;  }
DgtVec::DgtVec(double dW) :
    DgtVec(6, dW) {
       size = 4; d = 5.2; }
DgtVec::DgtVec(int s, double dW){
   size = s;  d = dW; }
int main() {
DgtVec v2;
 cout << v2.GetW() << endl; }
```

Bildschirmausgabe ?

1. Compilerfehler
2. 5.2
3. 2.4
4. 1.3

Bildschirmausgabe ?

1. Compilerfehler
2. 5.2
3. 2.4
4. 1.3