

```
// Vererbung 01
```

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

```
class X {
public:
    X () {i = 0;}
    void set (int ii) {
        cout << "set in X: i = " << ii << endl;
        i = ii;
    }
    int get () const {
        cout << "get in X" << endl;
        return i;
    }
    int increase () {return i = i * factor;}
private:
    int i;
    static const int factor;
};
```

```
int const X::factor = 11;
```

```
class Y {
public:
    X x;
    Y () {i = 0;}
    void set (int ii) {
        cout << "set in Y: i = " << ii << endl;
        i = ii;
    }
    int get () const {
        cout << "get in Y" << endl;
        return i;
    }
private:
    int i;
};
```

```
int main() {
    Y y;
    y.set (47);
    y.x.set (37);
```

```
return 0;
} //main
```

```
/* Ausgabe:
```

```
set in Y: i = 47
set in X: i = 37
```

```
*/
```

```
// Vererbung 02
```

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

```
struct X {
//public:      // nicht erforderlich
    X () {i = 0;}
    void set (int ii) {
        cout << "set in X: X.i = " << ii << endl;
        i = ii;
    }
    int get () const {
        cout << "get in X" << endl;
        return i;
    }
    int increase () {return i = i * factor;}
private:
    int i;
    static const int factor;
};
```

```
int const X::factor = 11;
```

```
struct Y {
    Y () {i = 0;}
    void set (int ii) {
        cout << "set in Y: Y.i = " << ii << endl;
        i = ii;
    }
    void xset (int ii) {      // Ergänzen der Schnittstelle
        x.set (ii); }
    int get () const {
        cout << "get in Y" << endl;
        return i; }
    int increase () {        // Ergänzen der Schnittstelle
        return x.increase (); }
private:
    int i;
    X x;                      // Ergänzung der Schnittstelle
                               // von y wird notwendig.
};
```

```
int main() {
    Y y;
    y.set (47);
    y.xset (37);
    y.increase ();
    return 0;
} //main
```

```
/* Ausgabe:
    set in Y: Y.i = 47
    set in X: X.i = 37
*/
```

```
// Vererbung 03
```

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

```
class X {
public:
    X () {i = 0;}
    void set (int ii) {
        cout << "set in X  X.i = " << ii << endl;
        i = ii;
    }
    int get () const {
        cout << "get in X  X.i = " << i << endl;
        return i;
    }
    int increase () {return i = i * factor;}
private:
    int i;
    static const int factor;
};
```

```
int const X::factor = 11;
```

```
class Y : public X {
public:
    Y() { i = 0; }
    int change () {
        i = increase ();
        return i;
    }
};
```

```
void set (int ii) {
    i = ii;
    X::set(ii); // Zugriff auf Basisklasse
}
private:
    int i;
};

int main() {
    cout << "sizeof(X) = " << sizeof(X) << endl;
    cout << "sizeof(Y) = "
        << sizeof(Y) << endl;
    Y D;
    D.change();
    D.get ();
    D.increase ();
    // Redefined functions hide base versions:
    D.set (12);
    return 0;
} //main
```

```
/* Ausgabe:
```

```
sizeof(X) = 4
sizeof(Y) = 8
get in X  X.i = 0
set in X  X.i = 12
```

```
*/
```

```
// Vererbung 03
```

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

```
class X {
public:
    X () {i = 0;}
    void set (int ii) {
        cout << "set in X  X.i = " << ii << endl;
        i = ii;
    }
    int get () const {
        cout << "get in X  X.i = " << i << endl;
        return i;
    }
    int increase () {return i = i * factor;}
private:
    int i;
    static const int factor;
};
```

```
int const X::factor = 11;
```

```
class Y : public X {
public:
    Y() { i = 0; }
    int change () {
        i = increase ();
        return i;
    }
};
```

```
void set (int ii) {
    i = ii;
    X::set(ii); // Zugriff auf Basisklasse
}
private:
    int i;
};

int main() {
    cout << "sizeof(X) = " << sizeof(X) << endl;
    cout << "sizeof(Y) = "
        << sizeof(Y) << endl;
    Y D;
    D.change();
    D.get ();
    D.increase ();
    // Redefined functions hide base versions:
    D.set (12);
    return 0;
} //main
```

```
/* Ausgabe:
```

```
sizeof(X) = 4
sizeof(Y) = 8
get in X  X.i = 0
set in X  X.i = 12
```

```
*/
```

// Vererbung 05

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

```
class Base {
public:
    Base (int) {cout << "Base Konstruktor\n"; }
    ~Base ()   {cout << "Base Destruktor\n"; }
};
```

```
class Member1 {
public:
    Member1 (int) {cout << "Member1 Konstruktor\n"; }
    ~Member1 ()  {cout << "Member1 Destruktor\n"; }
};
```

```
class Member2 {
public:
    Member2 (int) {cout << "Member2 Konstruktor\n"; }
    ~Member2 ()  {cout << "Member2 Destruktor\n"; }
};
```

```
class Derived1 : public Base {
    Member1 m1;
public:
    Derived1 (int) : m1 (1), Base (3) {
        cout << "Derived1 Konstruktor\n";
    }
    ~Derived1 () {
        cout << "Derived1 Destruktor\n";
    }
};
```

```
class Derived2 : public Derived1 {
    Member2 m2;
public:
    Derived2 () : Derived1 (6), m2 (7) {
        cout << "Derived2 Konstruktor\n";
    }
    ~Derived2 () {
        cout << "Derived2 Destruktor\n";
    }
};
```

```
int main () {
    Derived2 dd;
    return 0;
} //main
```

/* Ausgabe:

```
Base Konstruktor
Member1 Konstruktor
Derived1 Konstruktor
Member2 Konstruktor
Derived2 Konstruktor
Derived2 Destruktor
Member2 Destruktor
Derived1 Destruktor
Member1 Destruktor
Base Destruktor
```

***/**

```
// Vererbung 06
// Verdecken von Bezeichnern
```

```
class HH {
public:
    char d(char) const { return 'd';}
    float d(float) const { return 1.0; }
};
```

```
class BH : public HH {
public:
    class CC {};
    void d(CC) const {}
};
```

```
int main () {
    BH b;
    //b.d (1);    // Fehler, Funktion d aus BH
    //b.d ('x'); // Fehler, Funktion d aus BH
    return 0;
} //main
```

```
// Vererbung 06a
// Verdecken von Bezeichnern
#include <iostream>
using namespace std;
```

```
class HH {
public:
    char d(char) const { return 'd';}
    float d(float) const { return 1.0; }
};
```

```
class BH : public HH {
public:
    class CC {
    public:
        CC(int i) {
            cout << "CC konstruiert, i = "
                << i << endl;
        }
    };
    void d(CC) const {}
};
```

```
int main () {
    BH b;
    b.d (1);    // Typwandlung
    b.d ('x');  // Typwandlung
    return 0;
} //main
```

```
/* Ausgabe:
    CC konstruiert, i = 1
    CC konstruiert, i = 120
*/
```