

// Beispiel nach Lippman

```
#include <iostream>
```

```
#include <string>
```

```
#include <cstdlib>
```

```
using namespace std;
```

```
enum Locs { ZOOANIMAL, BEAR, PANDA };
```

```
class Animal {
```

```
    friend void print (Animal*, ostream& = cout);
```

```
public:
```

```
    Animal (string s = "Animal");
```

```
    virtual ~Animal() {}
```

```
    void link (Animal*);
```

```
    virtual void print (ostream&);
```

```
    virtual void isA (ostream&);
```

```
protected:
```

```
    string    name;
```

```
    Animal*  next;
```

```
};
```

```
Animal::Animal (string s) : next (0) {
```

```
    name = s;
```

```
}
```

```
void Animal::link (Animal* za) {
```

```
    za->next = next;
```

```
    next = za;
```

```
}
```

```
void Animal::isA (ostream& os) {
```

```
    os << "Animal name: " << name << "\n";
```

```
}
```

```
void Animal::print( ostream& os ) {
```

```
    isA (os); // virtual invocation
```

```
}
```

```
ostream& operator << (ostream& os, Animal& za) {
```

```
    za.print (os);
```

```
    return os;
```

```
}
```

```
void print (Animal* pz, ostream& os) {
```

```
    while (pz) {
```

```
        pz->print (os);
```

```
        pz = pz->next;
```

```
    }
```

```
}
```

```
class Bear : public Animal {
```

```
public:
```

```
    Bear (string s = "Bear", Locs loc = BEAR,
```

```
          string sci = "Ursidae" );
```

```
    ~Bear() {}
```

```
    void print (ostream&);
```

```
    void isA (ostream&);
```

```
protected:
```

```
    string sciName; // scientific name
```

```
    Locs zooArea;
```

```
};
```

```

Bear::Bear (string s, Locs loc, string sci)
    : Animal (s), zooArea (loc) {
    sciName = sci;
}

void Bear::isA (ostream& os) {
    Animal::isA (os); // static invocation
    os << "\tscientific name:\t";
    os << sciName << "\n";
}

static string locTable[] = {
    "The entire animal display area",    // ZOOANIMAL
    "NorthWest: B1: area Brown",        // BEAR
    "NorthWest: B1.P: area BrownSpots" // PANDA
    // ... and so on
};

void Bear::print (ostream& os) {
    Animal::print (os); // static invocation
    os << "\tZoo Area Location:\n\t";
    os << locTable [zooArea] << "\n";
}

```

```

class Panda : public Bear {
public:
    Panda (string nm, int room, string s = "Panda",
           string sci = "Ailuropoda Melaoleuca",
           Locs loc = PANDA );
    void print (ostream&);
    void isA (ostream&);
    ~Panda() {}
protected:
    string indName; // name of individual animal
    int cell;
};

Panda::Panda (string nm, int room, string s,
               string sci, Locs loc )
    : Bear (s, loc, sci), cell (room) {
    indName = nm;
}

void Panda::isA (ostream& os) {
    Bear::isA (os);
    os << "\twe call our friend:\t";
    os << indName << "\n";
}

void Panda::print (ostream& os) {
    Bear::print (os);
    os << "\tRoom Location:\t";
    os << cell << "\n";
}

```

```
int main () {
    Animal TierA ("Heinrich");
    TierA.print (cout);
    Animal TierB ("Lisa");
    TierB.link (&TierA);
    print (&TierB, cout);

    cout << endl;
    Bear BaerA ("Bruno");
    TierA.link (&BaerA);
    print (&TierB, cout);

    cout << endl;
    Panda PA ("Chi", 8);
    Panda PB ("Chou", 3);
    PB.link (&PA);
    print (&PB, cout);

    system ("PAUSE");
    return 0;
} //main
```

```
/* Ausgabe:
```

```
Animal name: Heinrich
Animal name: Lisa
Animal name: Heinrich
```

```
Animal name: Lisa
Animal name: Heinrich
Animal name: Bruno
    scientific name:    Ursidae
    Zoo Area Location:
    NorthWest: B1: area Brown
```

```
Animal name: Panda
    scientific name:    Ailuropoda Melaoleuca
    we call our friend:    Chou
    Zoo Area Location:
    NorthWest: B1.P: area BrownSpots
    Room Location: 3
```

```
Animal name: Panda
    scientific name:    Ailuropoda Melaoleuca
    we call our friend:    Chi
    Zoo Area Location:
    NorthWest: B1.P: area BrownSpots
    Room Location: 8
```

```
*/
```