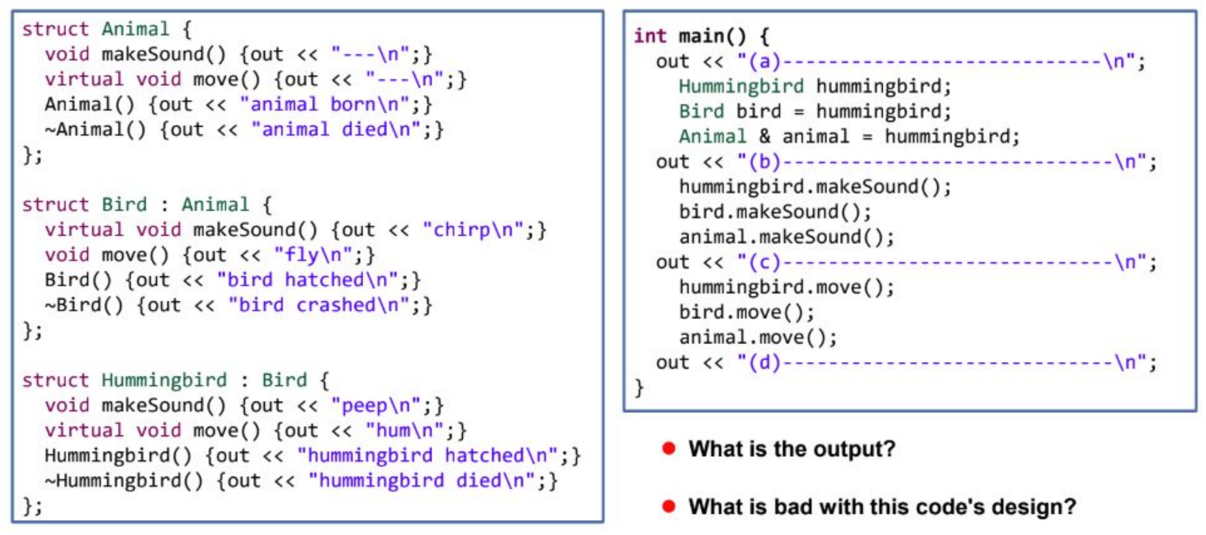
Dynamic Polymorphism

Beispiel aus dem Unterricht



```
struct Animal {
  void makeSound() {out << "---\n";}
  virtual void move() {out << "---\n";}
  Animal() {out << "animal born\n";}
  ~Animal() {out << "animal died\n";}
};

struct Bird : Animal {
  virtual void makeSound() {out << "chirp\n";}
  void move() {out << "fly\n";}
  Bird() {out << "bird hatched\n";}
  ~Bird() {out << "bird crashed\n";}
};

struct Hummingbird : Bird {
  void makeSound() {out << "peep\n";}
  virtual void move() {out << "hum\n";}
  Hummingbird() {out << "hummingbird hatched\n";}
  ~Hummingbird() {out << "hummingbird died\n";}
};
```

```
int main() {
  out << "(a)---------------------------\n";
    Hummingbird hummingbird;
    Bird bird = hummingbird;
    Animal & animal = hummingbird;
  out << "(b)---------------------------\n";
    hummingbird.makeSound();
    bird.makeSound();
    animal.makeSound();
  out << "(c)---------------------------\n";
    hummingbird.move();
    bird.move();
    animal.move();
  out << "(d)---------------------------\n";
}
```

- **What is the output?**
- **What is bad with this code's design?**

output:

(a):---

1.) **Animal born**
Bird hatched
Humminbird hatched

2.) **keine Ausgabe**

(da kein Copy-Konstruktor bei Bird)

3.) **keine Ausgabe**

(da Animal nur “neuen Namen” für hummingbird

(b):---

4.) **peep**

5.) **chirp**

6.) **---**

(c):---

7.)**hum**

8.)**fly**

9.)**hum**

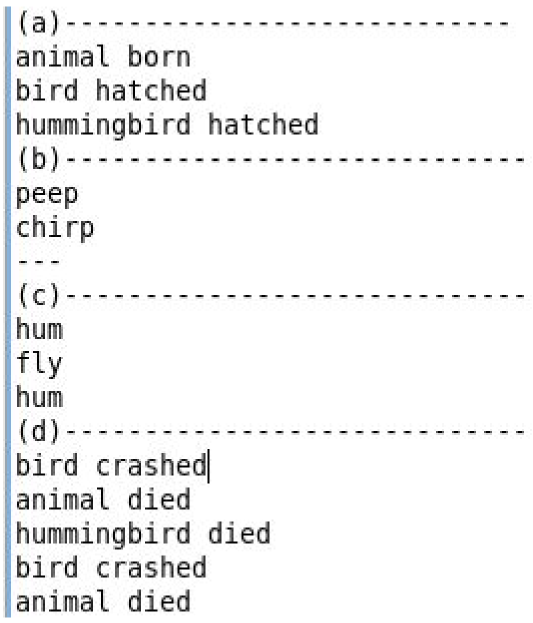
(d):---

10.)**keine Ausgabe**

Nur die Referenz wird aufgelöst, es wird nichts kaputt gemacht

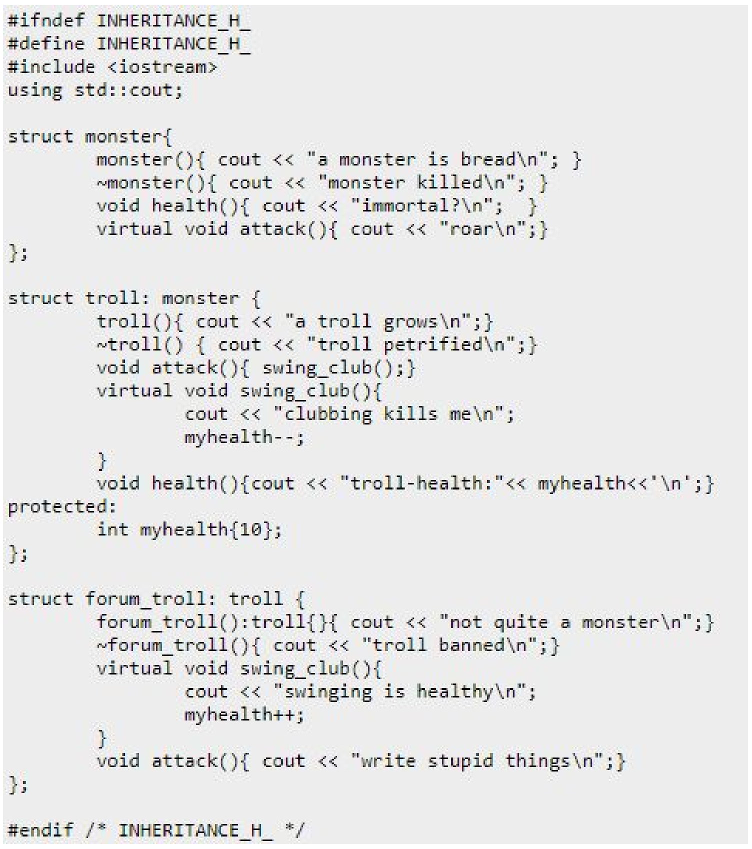
11.)**bird crashed**
animal died

12.) **hummingbird died**
bird crashed
animal died

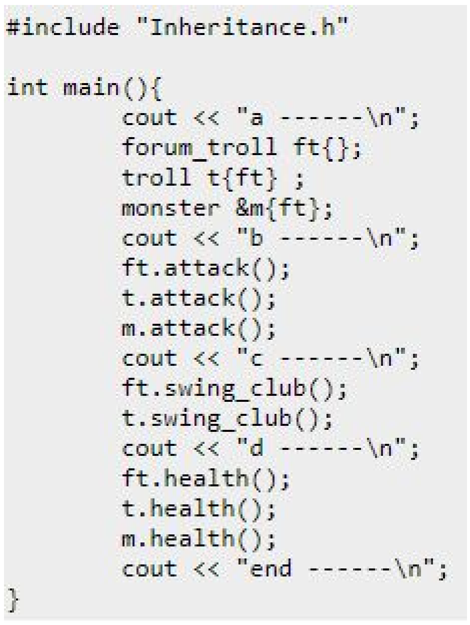


```
(a)---------------------------
animal born
bird hatched
hummingbird hatched
(b)---------------------------
peep
chirp
---
(c)---------------------------
hum
fly
hum
(d)---------------------------
bird crashed
animal died
hummingbird died
bird crashed
animal died
```

Beispiel aus der Übung

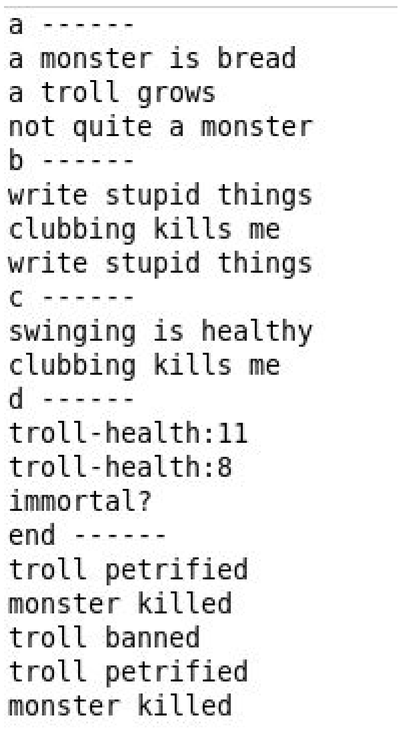


```
#ifndef INHERITANCE_H_
#define INHERITANCE_H_
#include <iostream>
using std::cout;

struct monster{
        monster(){ cout << "a monster is bread\n"; }
        ~monster(){ cout << "monster killed\n"; }
        void health(){ cout << "immortal?\n";  }
        virtual void attack(){ cout << "roar\n";}
};

struct troll: monster {
        troll(){ cout << "a troll grows\n";}
        ~troll() { cout << "troll petrified\n";}
        void attack(){ swing_club();}
        virtual void swing_club(){
                cout << "clubbing kills me\n";
                myhealth--;
        }
        void health(){cout << "troll-health:"<< myhealth<<'\n';}
protected:
        int myhealth{10};
};

struct forum_troll: troll {
        forum_troll():troll{}{ cout << "not quite a monster\n";}
        ~forum_troll(){ cout << "troll banned\n";}
        virtual void swing_club(){
                cout << "swinging is healthy\n";
                myhealth++;
        }
        void attack(){ cout << "write stupid things\n";}
};

#endif /* INHERITANCE_H_ */
```



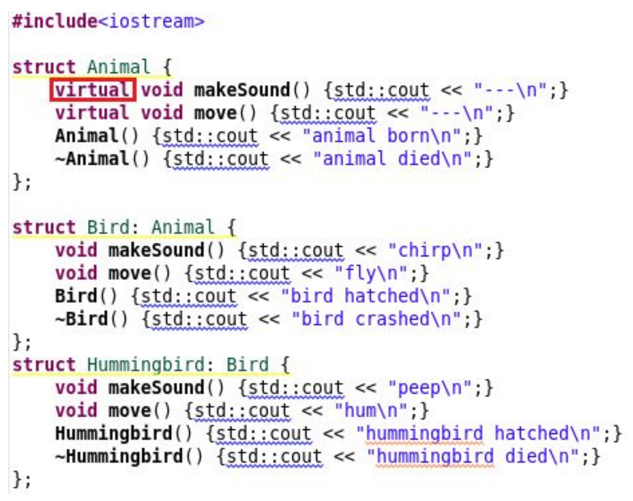
```
#include "Inheritance.h"

int main(){
        cout << "a ------\n";
        forum_troll ft{};
        troll t{ft} ;
        monster &m{ft};
        cout << "b ------\n";
        ft.attack();
        t.attack();
        m.attack();
        cout << "c ------\n";
        ft.swing_club();
        t.swing_club();
        cout << "d ------\n";
        ft.health();
        t.health();
        m.health();
        cout << "end ------\n";
}
```

output:
a -------
a monster is bread
a troll grows
not quite a monster
b -------drive.f
write stupid things
clubbing kills me //myhealt-- → myhealt = 9;
write stupid things
c -------
swinging is healthy //myhealth++ → myhealt = 11;
clubbing kills me //myhealt-- → myhealt = 8;
d -------
troll-health: 11
troll-health: 8
immortal?
end -------
troll petrified
monster killed
troll banned
troll petrified
monster killed

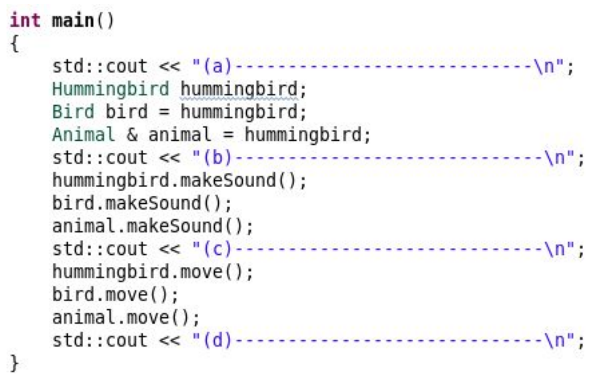


```
a ------
a monster is bread
a troll grows
not quite a monster
b ------
write stupid things
clubbing kills me
write stupid things
c ------
swinging is healthy
clubbing kills me
d ------
troll-health:11
troll-health:8
immortal?
end ------
troll petrified
monster killed
troll banned
troll petrified
monster killed
```

makeSound auch virtual



```cpp
#include<iostream>

struct Animal {
    virtual void makeSound() {std::cout << "---\n";}
    virtual void move() {std::cout << "---\n";}
    Animal() {std::cout << "animal born\n";}
    ~Animal() {std::cout << "animal died\n";}
};

struct Bird: Animal {
    void makeSound() {std::cout << "chirp\n";}
    void move() {std::cout << "fly\n";}
    Bird() {std::cout << "bird hatched\n";}
    ~Bird() {std::cout << "bird crashed\n";}
};
struct Hummingbird: Bird {
    void makeSound() {std::cout << "peep\n";}
    void move() {std::cout << "hum\n";}
    Hummingbird() {std::cout << "hummingbird hatched\n";}
    ~Hummingbird() {std::cout << "hummingbird died\n";}
};
```



```cpp
int main()
{
    std::cout << "(a)---------------------------\n";
    Hummingbird hummingbird;
    Bird bird = hummingbird;
    Animal & animal = hummingbird;
    std::cout << "(b)---------------------------\n";
    hummingbird.makeSound();
    bird.makeSound();
    animal.makeSound();
    std::cout << "(c)---------------------------\n";
    hummingbird.move();
    bird.move();
    animal.move();
    std::cout << "(d)---------------------------\n";
}
```

output:

(a):---

1.) **Animal born**
Bird hatched
Humminbird hatched

2.) **keine Ausgabe**
(da kein Copy-Konstruktor bei Bird)

3.) **keine Ausgabe**
(da Animal nur “neuen Namen” für hummingbird

(b):---

4.) **peep**
5.) **chirp**
6.) **peep**

(c):---

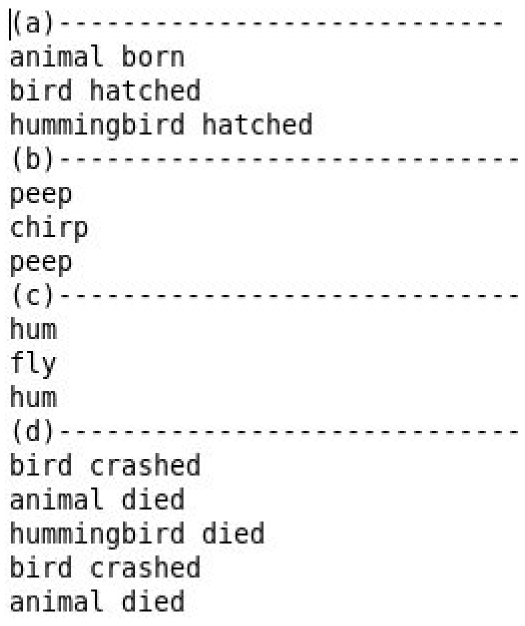
7.)**hum**
8.)**fly**
9.)**hum**

(d):---

10.)**keine Ausgabe**
Nur die Referenz wird aufgelöst, es wird nichts kaputt gemacht

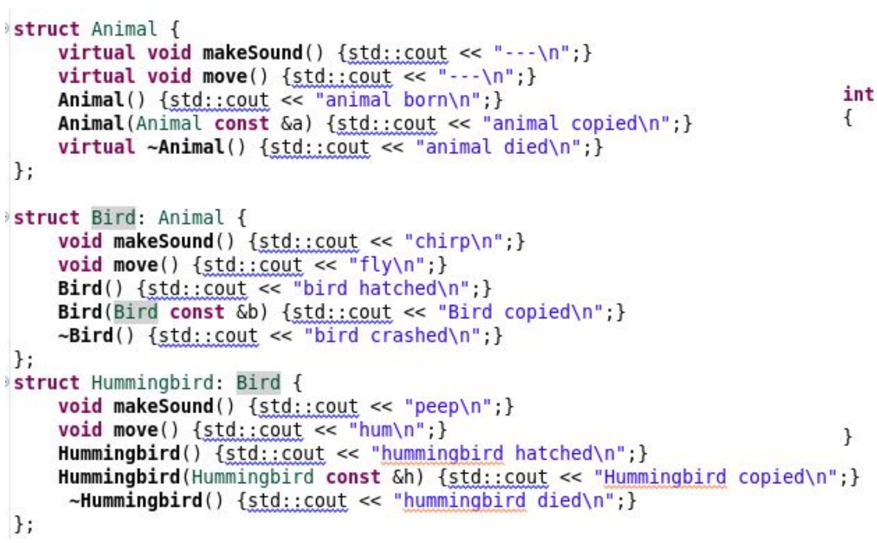
11.)**bird crashed**
animal died

12.) **hummingbird died**
bird crashed
animal died

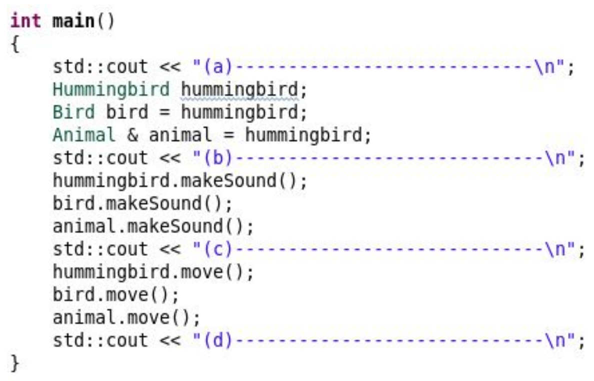


```
(a)---------------------------
animal born
bird hatched
hummingbird hatched
(b)---------------------------
peep
chirp
peep
(c)---------------------------
hum
fly
hum
(d)---------------------------
bird crashed
animal died
hummingbird died
bird crashed
animal died
```

copyConstructor



```cpp
struct Animal {
    virtual void makeSound() {std::cout << "---\n";}
    virtual void move() {std::cout << "---\n";}
    Animal() {std::cout << "animal born\n";}
    Animal(Animal const &a) {std::cout << "animal copied\n";}
    virtual ~Animal() {std::cout << "animal died\n";}
};

struct Bird: Animal {
    void makeSound() {std::cout << "chirp\n";}
    void move() {std::cout << "fly\n";}
    Bird() {std::cout << "bird hatched\n";}
    Bird(Bird const &b) {std::cout << "Bird copied\n";}
    ~Bird() {std::cout << "bird crashed\n";}
};
struct Hummingbird: Bird {
    void makeSound() {std::cout << "peep\n";}
    void move() {std::cout << "hum\n";}
    Hummingbird() {std::cout << "hummingbird hatched\n";}
    Hummingbird(Hummingbird const &h) {std::cout << "Hummingbird copied\n";}
     ~Hummingbird() {std::cout << "hummingbird died\n";}
};
```



```cpp
int main()
{
    std::cout << "(a)---------------------------\n";
    Hummingbird hummingbird;
    Bird bird = hummingbird;
    Animal & animal = hummingbird;
    std::cout << "(b)----------------------------\n";
    hummingbird.makeSound();
    bird.makeSound();
    animal.makeSound();
    std::cout << "(c)----------------------------\n";
    hummingbird.move();
    bird.move();
    animal.move();
    std::cout << "(d)----------------------------\n";
}
```

output:

(a)----------------------------

animal born

bird hatched

hummingbird hatched

animal born

Bird copied

(b)----------------------------

peep

chirp

peep

(c)----------------------------

hum

fly

hum

(d)----------------------------

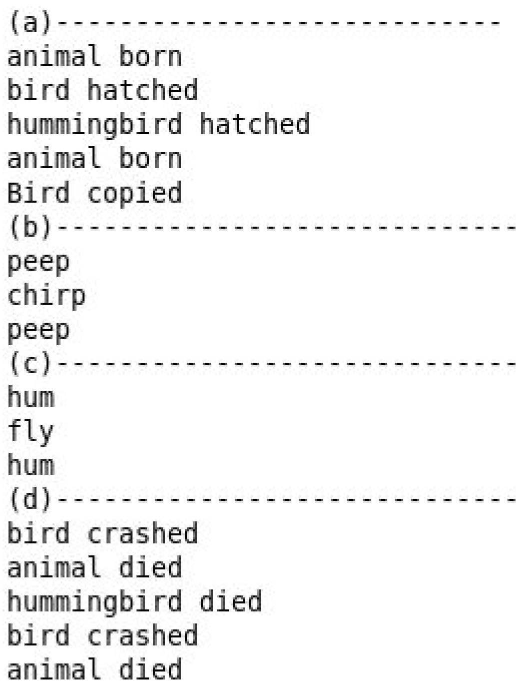
bird crashed

animal died

hummingbird died

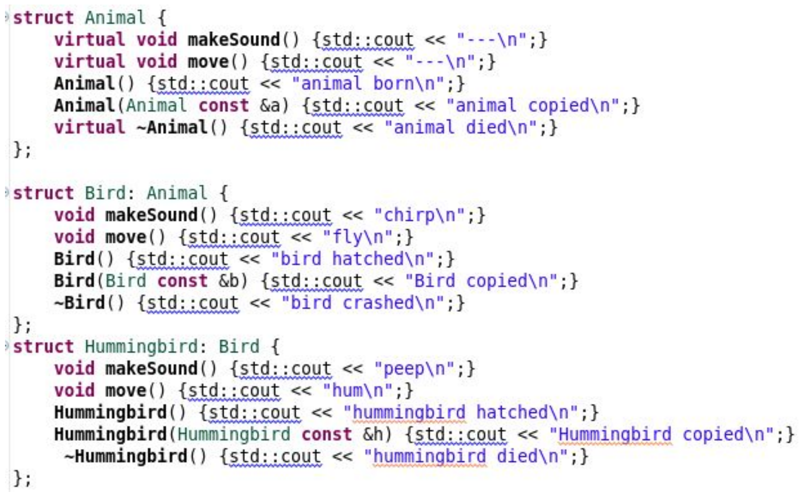
bird crashed

animal died

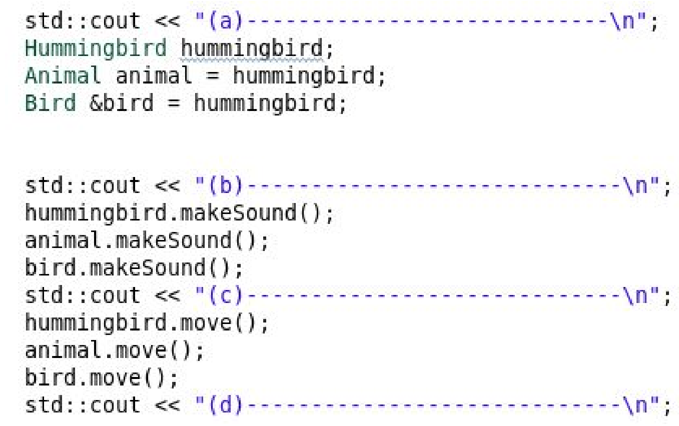


```
(a)---------------------------
animal born
bird hatched
hummingbird hatched
animal born
Bird copied
(b)----------------------------
peep
chirp
peep
(c)----------------------------
hum
fly
hum
(d)----------------------------
bird crashed
animal died
hummingbird died
bird crashed
animal died
```

copyConstructor - verändertes main



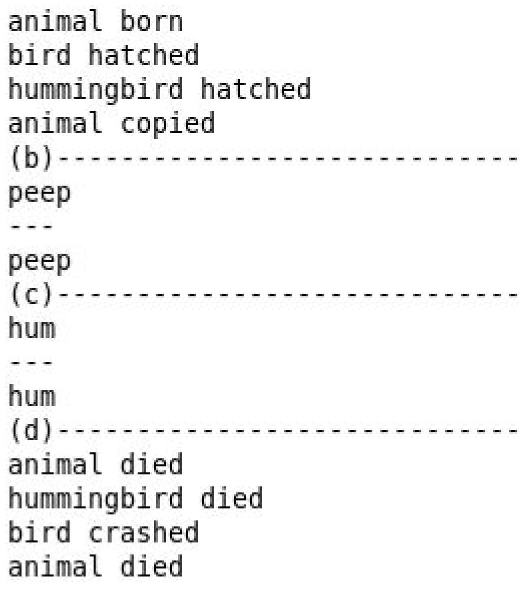
```
struct Animal {
    virtual void makeSound() {std::cout << "---\n";}
    virtual void move() {std::cout << "---\n";}
    Animal() {std::cout << "animal born\n";}
    Animal(Animal const &a) {std::cout << "animal copied\n";}
    virtual ~Animal() {std::cout << "animal died\n";}
};

struct Bird: Animal {
    void makeSound() {std::cout << "chirp\n";}
    void move() {std::cout << "fly\n";}
    Bird() {std::cout << "bird hatched\n";}
    Bird(Bird const &b) {std::cout << "Bird copied\n";}
    ~Bird() {std::cout << "bird crashed\n";}
};
struct Hummingbird: Bird {
    void makeSound() {std::cout << "peep\n";}
    void move() {std::cout << "hum\n";}
    Hummingbird() {std::cout << "hummingbird hatched\n";}
    Hummingbird(Hummingbird const &h) {std::cout << "Hummingbird copied\n";}
     ~Hummingbird() {std::cout << "hummingbird died\n";}
};
```



```
std::cout << "(a)----------------------------\n";
Hummingbird hummingbird;
Animal animal = hummingbird;
Bird &bird = hummingbird;


std::cout << "(b)----------------------------\n";
hummingbird.makeSound();
animal.makeSound();
bird.makeSound();
std::cout << "(c)----------------------------\n";
hummingbird.move();
animal.move();
bird.move();
std::cout << "(d)----------------------------\n";
```

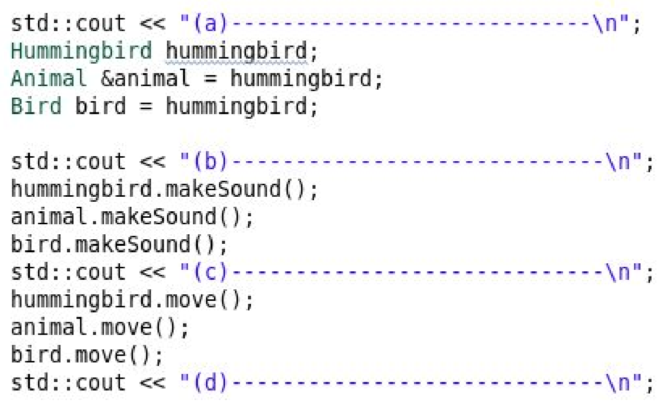
output:

```
(a)----------------------------
animal born
bird hatched
hummingbird hatched
animal copied
(b)----------------------------
peep
---
peep
(c)----------------------------
hum
---
hum
(d)----------------------------
animal died
hummingbird died
bird crashed
animal died
```

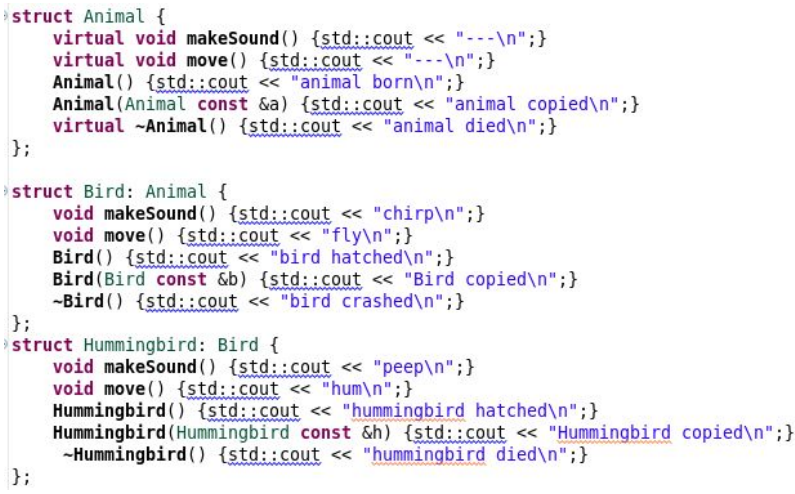


```
animal born
bird hatched
hummingbird hatched
animal copied
(b)----------------------------
peep
---
peep
(c)----------------------------
hum
---
hum
(d)----------------------------
animal died
hummingbird died
bird crashed
animal died
```

copyConstructor - verändertes main2



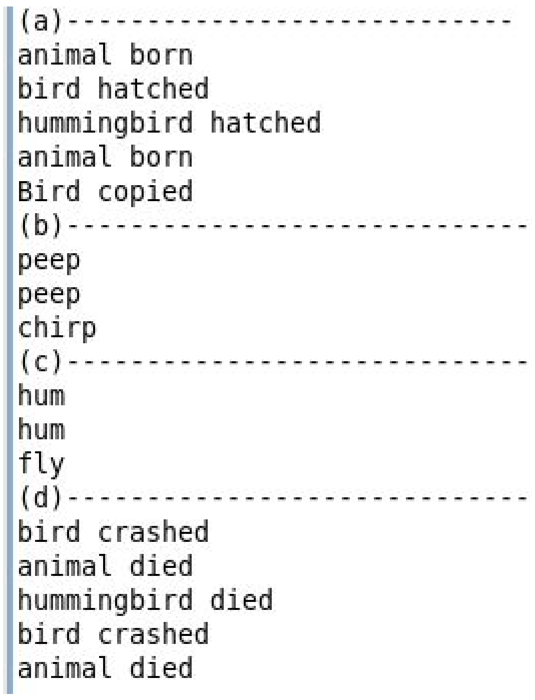
```
std::cout << "(a)----------------------------\n";
Hummingbird hummingbird;
Animal &animal = hummingbird;
Bird bird = hummingbird;

std::cout << "(b)-----------------------------\n";
hummingbird.makeSound();
animal.makeSound();
bird.makeSound();
std::cout << "(c)-----------------------------\n";
hummingbird.move();
animal.move();
bird.move();
std::cout << "(d)-----------------------------\n";
```



```
struct Animal {
    virtual void makeSound() {std::cout << "---\n";}
    virtual void move() {std::cout << "---\n";}
    Animal() {std::cout << "animal born\n";}
    Animal(Animal const &a) {std::cout << "animal copied\n";}
    virtual ~Animal() {std::cout << "animal died\n";}
};

struct Bird: Animal {
    void makeSound() {std::cout << "chirp\n";}
    void move() {std::cout << "fly\n";}
    Bird() {std::cout << "bird hatched\n";}
    Bird(Bird const &b) {std::cout << "Bird copied\n";}
    ~Bird() {std::cout << "bird crashed\n";}
};
struct Hummingbird: Bird {
    void makeSound() {std::cout << "peep\n";}
    void move() {std::cout << "hum\n";}
    Hummingbird() {std::cout << "hummingbird hatched\n";}
    Hummingbird(Hummingbird const &h) {std::cout << "Hummingbird copied\n";}
     ~Hummingbird() {std::cout << "hummingbird died\n";}
};
```

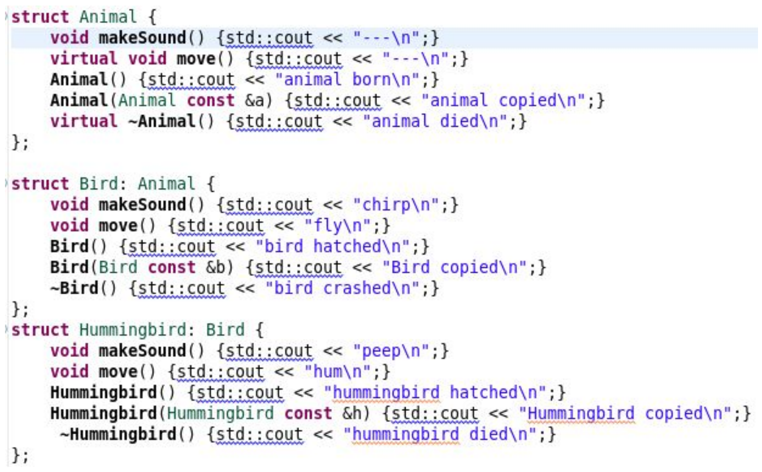
output:

```
(a)----------------------------
animal born
bird hatched
hummingbird hatched
animal born
bird copied
(b)-----------------------------
peep
peep
chirp
(c)-----------------------------
hum
hum
fly
(d)-----------------------------
bird crashed
animal died
hummingbird died
bird crashed
animal died
```

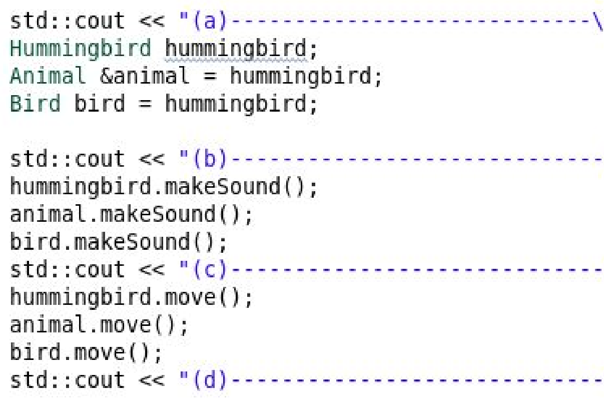


```
(a)----------------------------
animal born
bird hatched
hummingbird hatched
animal born
Bird copied
(b)-----------------------------
peep
peep
chirp
(c)-----------------------------
hum
hum
fly
(d)-----------------------------
bird crashed
animal died
hummingbird died
bird crashed
animal died
```

copyConstructor - verändertes main - makeSound non virtual

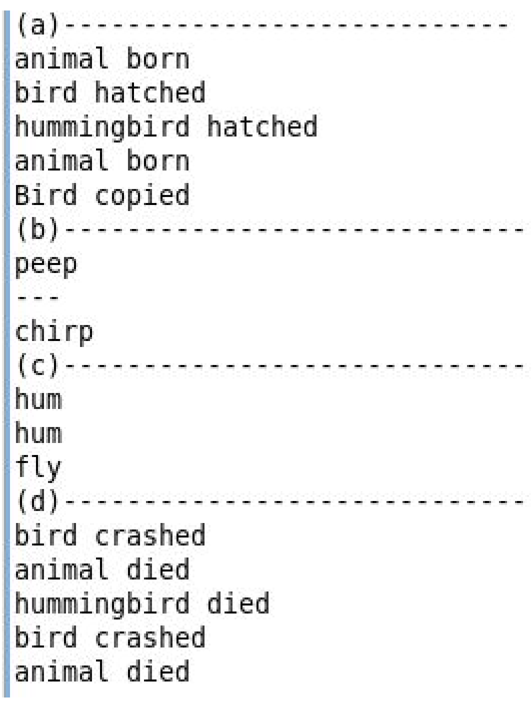


```cpp
struct Animal {
    void makeSound() {std::cout << "---\n";}
    virtual void move() {std::cout << "---\n";}
    Animal() {std::cout << "animal born\n";}
    Animal(Animal const &a) {std::cout << "animal copied\n";}
    virtual ~Animal() {std::cout << "animal died\n";}
};

struct Bird: Animal {
    void makeSound() {std::cout << "chirp\n";}
    void move() {std::cout << "fly\n";}
    Bird() {std::cout << "bird hatched\n";}
    Bird(Bird const &b) {std::cout << "Bird copied\n";}
    ~Bird() {std::cout << "bird crashed\n";}
};
struct Hummingbird: Bird {
    void makeSound() {std::cout << "peep\n";}
    void move() {std::cout << "hum\n";}
    Hummingbird() {std::cout << "hummingbird hatched\n";}
    Hummingbird(Hummingbird const &h) {std::cout << "Hummingbird copied\n";}
     ~Hummingbird() {std::cout << "hummingbird died\n";}
};
```



```cpp
std::cout << "(a)----------------------------\
Hummingbird hummingbird;
Animal &animal = hummingbird;
Bird bird = hummingbird;

std::cout << "(b)-----------------------------
hummingbird.makeSound();
animal.makeSound();
bird.makeSound();
std::cout << "(c)-----------------------------
hummingbird.move();
animal.move();
bird.move();
std::cout << "(d)-----------------------------
```

```
(a)----------------------------
animal born
bird hatched
hummingbird hatched
animal born
Bird copied
(b)-----------------------------
peep
---
chirp
(c)-----------------------------
hum
hum
fly
(d)-----------------------------
bird crashed
animal died
hummingbird died
bird crashed
animal died
```



```
(a)----------------------------
animal born
bird hatched
hummingbird hatched
animal born
Bird copied
(b)------------------------------
peep
---
chirp
(c)------------------------------
hum
hum
fly
(d)------------------------------
bird crashed
animal died
hummingbird died
bird crashed
animal died
```