//funktion.cpp

```
#include <iostream.h>
#include <funktion.h>

double Funktion::operator()(double x) const {
      return 0.;
}

void Funktion::ausgebenAuf(ostream& os) const {
      os << "???";
}

ostream& operator<< (ostream& os, const Funktion& f) {
      f.ausgebenAuf(os);
      return os;
}
```

//polynomfkt.cpp

```
#include <iostream.h>

#include <polynom.h>
#include <polynomfkt.h>

PolynomFunktion::PolynomFunktion(const Polynom& p) {
      mPolynom = p;
}

double PolynomFunktion::operator() (double x) const {
      return mPolynom(x);
}

void PolynomFunktion::ausgebenAuf(ostream& os) const {
      os << mPolynom;
}

Funktion* PolynomFunktion::kopie() const {
      return new PolynomFunktion(mPolynom);
}
```

//dekorierer.cpp

```
#include <iostream.h>

#include <funktion.h>
#include <dekorierer.h>

Dekorierer::Dekorierer(const Funktion& f) : mFunktion(f.kopie()) {
}

Dekorierer::Dekorierer(const Funktion* const f) : mFunktion(f->kopie()) {
}

Dekorierer::Dekorierer(const Dekorierer& d) : mFunktion(d.mFunktion->kopie()) {
}

double Dekorierer::operator() (double x) const {
      return mFunktion->operator()(x);
}

void Dekorierer::ausgebenAuf(ostream& os) const {
      mFunktion->ausgebenAuf(os);
}

Funktion* Dekorierer::kopie() const {
      return new Dekorierer(mFunktion->kopie());
}
```

```
const Dekorierer& Dekorierer::operator= (const Dekorierer& d) {
      if (this == &d) return *this;
      delete mFunktion;
      mFunktion = d.mFunktion->kopie();
      return *this;
}

Dekorierer::~Dekorierer() {
      delete mFunktion;
}
```

//betrag.cpp

```
#include <iostream.h>
#include <math.h>

#include <funktion.h>
#include <betrag.h>

Betrag::Betrag(const Funktion& f) : Dekorierer(f) {
}

double Betrag::operator() (double x) const {
      double innen = Dekorierer::operator()(x);
      return fabs(innen);
}

void Betrag::ausgebenAuf(ostream& os) const {
      os << "|";
      Dekorierer::ausgebenAuf(os);
      os << "|";
}

Funktion* Betrag::kopie() const {
      return new Betrag(*this);
}
```

//kehrwert.cpp

```
#include <iostream.h>
#include <assert.h>

#include <funktion.h>
#include <kehrwert.h>

Kehrwert::Kehrwert(const Funktion& f) : Dekorierer(f) {
}

double Kehrwert::operator() (double x) const {
      double nenner = Dekorierer::operator()(x);
      assert(nenner != 0.);
      return 1./nenner;
}

void Kehrwert::ausgebenAuf(ostream& os) const {
      os << "1/(";
      Dekorierer::ausgebenAuf(os);
      os << ")";
}

Funktion* Kehrwert::kopie() const {
      return new Kehrwert(*this);
}
```

//spiegelhor.cpp

```
#include <iostream.h>

#include <funktion.h>
#include <spiegelhor.h>

SpiegelungHorizontal::SpiegelungHorizontal(const Funktion& f, double versatz)
    : Dekorierer(f), mVersatz(versatz) {
}

double SpiegelungHorizontal::operator() (double x) const {
      double innen = Dekorierer::operator()(x);
      return mVersatz - innen;
}

void SpiegelungHorizontal::ausgebenAuf(ostream& os) const {
      os << mVersatz << "-(";
      Dekorierer::ausgebenAuf(os);
      os << ")";
}

Funktion* SpiegelungHorizontal::kopie() const {
      return new SpiegelungHorizontal(*this);
}
```