```
*-----------------------------------Baby.cpp----------------------*
#include <stdio.h>

#include "Baby.h"   // Header nicht vergessen!

// Default Constructor Implementierung
Baby::Baby(){
    name = "mybaby1";
    alter = 0.0;
}

// Overload Constructor Implementierung
Baby::Baby(string name, float newAlter) {
    this->name = name;  // this-Zeiger auf Klassenattribut
    alter = newAlter;   // Ohne this-Zeiger
}

// Destructor Implementierung
Baby::~Baby() {
    cout << "ist tot. " << endl;
}

// Funktionen Implementierung
void Baby::schlafen(int dauer) {
    cout << "schlafe" << endl;
}

void Baby::trinken() {
    cout << "trinke jetzt" << endl;
}

void Baby::weinen() {
    cout << "weine jetzt" << endl;
}
*-----------------------------------Baby.cpp----------------------*
#include <stdio.h>

#include "Baby.h"   // Header nicht vergessen!

// Default Constructor Implementierung
Baby::Baby(){
    name = "mybaby1";
    alter = 0.0;
}

// Overload Constructor Implementierung
Baby::Baby(string name, float newAlter) {
    this->name = name;  // this-Zeiger auf Klassenattribut
    alter = newAlter;   // Ohne this-Zeiger
}

// Destructor Implementierung
```

```
Baby::~Baby() {
    cout << "ist tot. " << endl;
}

// Funktionen Implementierung
void Baby::schlafen(int dauer) {
    cout << "schlafe" << endl;
}

void Baby::trinken() {
    cout << "trinke jetzt" << endl;
}

void Baby::weinen() {
    cout << "weine jetzt" << endl;
}

*------------------------------main von Baby----------------------*

#include <cstdlib>
#include "Baby.h"   // Header Datei nicht vergessen

using namespace std;

/*
 * Kreiieren Sie ein Baby Kilian, 3 Monate alt, der zuerst weint,
 * dann zu trinken bekommt und anschließend 3 Stunden schläft.
 */
int main(int argc, char** argv) {

    // Aufruf des Overload Constructors:
    Baby Kilian("Kilian", 3.0);  // Baby Kilian wird kreiiert.

    Kilian.weinen();         // Kilian weint
    Kilian.trinken();        // Kilian trinke
    Kilian.schlafen(3.0);    // Kilian weint

    return 0;
}
```

%%
%%%%%%

```
*---------------------------Filehandling
 Main------------------------------*

//#include <cstdlib>
#include<iostream>
#include<stdio.h>

using namespace std;

int main() {
```

```
    FILE *exampleFile = NULL;
    exampleFile = fopen("./Numbers.txt", "w"); // Datei
 oeffnen zum Beschreiben.
    // Modus "w" für writable oder "a" für append.
    // Wenn File nicht existiert, dann wird zuerst erzeugt.

    // Erzeugen und Beschreiben einer Datei:
    const int MAX = 101;
    for (int i = 0; i <= MAX; i++) {
        fprintf(exampleFile, "%d; ", i * i);
    }
    fclose(exampleFile); // scliessen mit eof. Sonst Absturz bei !
feof(exampleFile) beim Lesen.


    // Auslesen einer Datei. Inhalt in einen Array abspeichern.
    exampleFile = fopen("./Numbers.txt", "r"); // Datei
 oeffnen zu lesen. Modus "r" für readable.
    int myArray[MAX] = {0};
    int i = 0;
    if (exampleFile != NULL) {  // Abfangen, dass die File vorher
 existiert.
        while (!feof(exampleFile) && i <= MAX) {   // Bis Ende der
 Datei lesen.
            fscanf(exampleFile, "%d; ", &myArray[i]); // Sucht
 nach ";" Zeichen und speichert die Eintraege in eine Array ab.
            printf("%d: %d \n", i, myArray[i]);   // Ausgabe
 auf Konsole zur Kontrolle
            i++;
        }
        fclose(exampleFile);   // Datei schliessen.
    }

    return 0;
}

*----------------------------Filehandling: Tests-----------------*

#include <gtest/gtest.h>
#include <iostream>
using namespace std;

TEST(TS1,TC1){
    FILE *exampleFile = fopen("Numbers.txt", "r");
    int myArray[100] = {};
    int i = 0;
     while (!feof(exampleFile) && i<=100) {
            fscanf(exampleFile, "%d; ", &myArray[i]);  // Sucht
 nach ";" Zeichen
            i++;
        }
    EXPECT_EQ(myArray[0],0);
    fclose(exampleFile);
```

```
}

TEST(TS1,TC2){
    FILE *exampleFile = fopen("Numbers.txt", "r");
    int myArray[100] = {};
    int i = 0;
     while (!feof(exampleFile) && i<=100) {
            fscanf(exampleFile, "%d; ", &myArray[i]);  // Sucht
 nach ";" Zeichen
            i++;
        }
    EXPECT_EQ(myArray[50],2500);
    fclose(exampleFile);
}

TEST(TS1,TC3){
    FILE *exampleFile = fopen("Numbers.txt", "r");
    int myArray[100] = {};
    int i = 0;
     while (!feof(exampleFile) && i<=100) {
            fscanf(exampleFile, "%d; ", &myArray[i]);  // Sucht
 nach ";" Zeichen
            i++;
        }
    EXPECT_EQ(myArray[100],10000);
    fclose(exampleFile);
}

TEST(TS1,TC4){
    FILE *exampleFile = fopen("Numbers.txt", "r");
    int myArray[100] = {};
    int i = 0;
     while (!feof(exampleFile) && i<=100) {
            fscanf(exampleFile, "%d; ", &myArray[i]);  // Sucht
 nach ";" Zeichen
            i++;
        }
    EXPECT_EQ(myArray[100],10001);  // Failure
    fclose(exampleFile);
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%

/*****************************FSMBaby.h***************************/

#include "FSMBaby.h"
#include "Baby.h"

FSMBaby::FSMBaby() {
}

FSMBaby::FSMBaby(Baby baby, ZUSTAND zustand) {
    this->theBaby = baby;
```

```
    this->actualState = zustand;
}

FSMBaby::~FSMBaby() {
}


// Evaluierungsmethode um die Logik der FSM zu triggern:
void FSMBaby::evalFSMBaby(string command) {

    cout << "Anfangszustand des Kinds ist: " << actualState << endl;

    // Zustandsübergänge innerhalb von FSM (nach der Modellierung in
 der Vorlesung):
    switch (this->actualState) {
        case W:
            // input trinken und gehe in Zustand T
            if (command == "trinken") {
                theBaby.trinken();
                actualState = T; // = 2
            } else {
                cout << "geht leider nicht!" << endl;
            }
            break;
        case S:
            // input weinen und gehe in Zustand W
            if (command == "weinen") {
                theBaby.weinen();
                actualState = W; // = 1
            } else {
                cout << "geht leider nicht!" << endl;
            }
            break;
        case T:
            // input schlafen und gehe in Zustand S
            if (command == "schlafen") {
                theBaby.schlafen(3); // 3 h schlafen
                actualState = S; // = 0
            } else {
                cout << "geht leider nicht!" << endl;
            }
            break;
        default:
            cout << "geht nicht" << endl;
    }

    cout << "neuer Zustand ist: " << actualState << endl << endl;
}

*------------------ FSMBaby.cpp-----------------------*

/*****************************FSMBaby.h**************************/

#include "FSMBaby.h"
```

```
#include "Baby.h"

FSMBaby::FSMBaby() {
}

FSMBaby::FSMBaby(Baby baby, ZUSTAND zustand) {
    this->theBaby = baby;
    this->actualState = zustand;
}

FSMBaby::~FSMBaby() {
}


// Evaluierungsmethode um die Logik der FSM zu triggern:
void FSMBaby::evalFSMBaby(string command) {

    cout << "Anfangszustand des Kinds ist: " << actualState << endl;

    // Zustandsübergänge innerhalb von FSM (nach der Modellierung in
 der Vorlesung):
    switch (this->actualState) {
        case W:
            // input trinken und gehe in Zustand T
            if (command == "trinken") {
                theBaby.trinken();
                actualState = T; // = 2
            } else {
                cout << "geht leider nicht!" << endl;
            }
            break;
        case S:
            // input weinen und gehe in Zustand W
            if (command == "weinen") {
                theBaby.weinen();
                actualState = W; // = 1
            } else {
                cout << "geht leider nicht!" << endl;
            }
            break;
        case T:
            // input schlafen und gehe in Zustand S
            if (command == "schlafen") {
                theBaby.schlafen(3); // 3 h schlafen
                actualState = S; // = 0
            } else {
                cout << "geht leider nicht!" << endl;
            }
            break;
        default:
            cout << "geht nicht" << endl;
    }

    cout << "neuer Zustand ist: " << actualState << endl << endl;
```

```
}

*------------------------FSMBaby
 Main-------------------------------------*



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

#include "FSMBaby.h"        // FSM Header Datei nicht vergessen.

int main(int argc, char** argv) {
    Baby Kilian("Kilian", 3.0);  // Baby
 Kilian als Kontextobjekt erzeugen fuer die FSM.
    FSMBaby fsm(Kilian, W);      // Kreiieren des FSM-
Objekts fuer Baby Kilian.

    // Evaluierungslogik in der FSM einzeln aufrufen:
    fsm.evalFSMBaby("trinken");     // Kilian in Zustand T=2
    fsm.evalFSMBaby("schlafen");    // Kilian in Zustand S=0
    fsm.evalFSMBaby("weinen");      // Kilian in Zustand W=1
    fsm.evalFSMBaby("schlafen");    // nach
 Weinen sollte Kilian trinken.
                                    // Daher
 ist schlafen nicht zulässig.
    fsm.evalFSMBaby("weinen");      // immernoch nicht zulässig.
    fsm.evalFSMBaby("trinken");     // Kilian in Zustand S=0


    // Alternativ über
 eine While Schleife und Daten aus Konsole reinholen:
    string command;
    while (true) {          // Immerwieder nachfragen, was Baby machen
 soll.
        cout << "Kommando für das Baby bitte eingeben: " << endl;
        cin >> command;
        fsm.evalFSMBaby(command);   // Evaluierungslogik
 der FSM in der Schleife aufrufen.
    }
    return 0;
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

*--------------------- Interface ------------------------------*


*--------------------------------circle.h---------------------*

#include "Circle.h"

Circle::Circle() {
```

```
}

Circle::Circle(double radius){
    this->radius = radius;
}

Circle::~Circle() {
}

float Circle::getArea() {
    return (3.14*radius*radius);
}

*---------------------- circle.cpp -------------------------------*



#include "Circle.h"

Circle::Circle() {
}

Circle::Circle(double radius){
    this->radius = radius;
}

Circle::~Circle() {
}

float Circle::getArea() {
    return (3.14*radius*radius);
}

*--------------------------Rectangle.cpp-------------*

#include "Rectangle.h"

Rectangle::Rectangle(){
}

Rectangle::Rectangle(int hoehe, int breite) {
    this->hoehe = hoehe;
    this->breite = breite;
}

Rectangle::~Rectangle() {
}

float Rectangle::getArea() {
    return (breite * hoehe);
}

*----------------------Rectangle.h------------------*
```

```
#ifndef RECTANGLE_H
#define RECTANGLE_H

#include "IShape.h"

class Rectangle : public IShape {
public:
    Rectangle();
    Rectangle(int, int);
    ~Rectangle();
    float getArea();  // Interface Methode von IShape
private:
   int hoehe;
   int breite;
};


#endif /* RECTANGLE_H */

*-----------------------Ishape.h-----------------*

#ifndef ISHAPE_H
#define ISHAPE_H

class IShape{
public:
    virtual ~IShape() {};
    virtual float getArea() = 0;   // Pure Virtual Methode
};

#endif /* ISHAPE_H */

*-------------------------main von Ishape ------------------*

#include <cstdlib>
#include <iostream>
#include <stdio.h>
using namespace std;


// #include "IShape.h"
#include "Circle.h"
#include "Rectangle.h"

int main() {
   IShape *ptr = NULL;
   Rectangle rec(2,3);
   cout << "Retangle class: " << rec.getArea() << endl;

   ptr = &rec;
   cout << "Rectangle interface: " << ptr->getArea() << endl;

   Circle cir(2.3);
```

```
    ptr = &cir;
    cout << "Circle interface: " << ptr->getArea() << endl;

    return 0;
}


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%5

*------------------------------ List and vectors ---------------------
*

#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <list>
#include <vector>
using namespace std;

int main() {
    list<int> mylist;
    list<int>::iterator it;

    for (int i = 1; i <= 5; i++) {
        mylist.push_back(i);
    } // 1 2 3 4 5

    vector<int> myvector(2, 30); // [30 30]
    myvector.push_back(10); // [ 30 30 10]
    cout << "vektor: " << myvector.at(2) << endl;

    it = mylist.begin();
    advance(it, 2);  // Iterator verschoben mit advance()
    mylist.insert(it, 10); // 1 2 10 3 4 5

    for (int i = 0; i < myvector.size(); i++) {
        mylist.insert(it, myvector.at(i));
    }

    for (it = mylist.begin(); it != mylist.end(); it++) {
        cout << *it << "; " << endl;
    } // 1 2 10 30 30 10 3 4 5

    return 0;
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

*---------------------- Vererbung ---------------------------------*

*----------------------Adult.h------------------*
```

```
#include "Adult.h"

Adult::Adult() {
}

Adult::Adult(bool geschlecht) {

}

Adult::~Adult() {
    printf("Adult ist gestorben \n");
}

void Adult::arbeiten(int dauer) {
    if(this-> == true){
    printf("mein geschlecht ist: Männlich \n");
    }
          else{
             printf("mein geschlecht ist: Weiblich \n");

            }

}

void Adult::essen() {

}


*---------------------------------- Adult.cpp ------------------*


#include "Adult.h"

Adult::Adult() {
}

Adult::Adult(bool geschlecht) {

}

Adult::~Adult() {
    printf("Adult ist gestorben \n");
}

void Adult::arbeiten(int dauer) {
    if(this-> == true){
    printf("mein geschlecht ist: Männlich \n");
    }
          else{
             printf("mein geschlecht ist: Weiblich \n");

            }
```

```
}

void Adult::essen() {

}


*-------------------------Baby.cpp-----------------------*


#include "Baby.h"

Baby::Baby() {
    printf("Baby ist geboren \n");
}

Baby::Baby(string name, float alter) : Person(name, alter) {
    printf("Baby ist geboren \n");
}


Baby::~Baby() {
    printf("Baby ist tod \n");
}

bool Baby::krabbeln() {

    return false;

}

void Baby::weinen() {

}


*-------------------------baby.h---------------------*

#ifndef BABY_H
#define BABY_H
#include "Person.h"

using namespace std;

class Baby : public Person {
public:
    Baby();
    Baby(string name, float alter);
    virtual ~Baby();
    void weinen();
    bool krabbeln();

private:
```

```
};

#endif /* BABY_H */

*--------------------------- Person.cpp -----------------*

#include "Person.h"

Person::Person() {
    printf(" Person ist geboren \n");
}

Person::Person(string name, float alter) {
        printf("mein name ist: %s \n", name.c_str());
        printf("mein alter ist: %.2f \n", alter);
}


Person::~Person() {
    printf("Person ist tod \n");
}




void Person::schlafen(int dauer) {
    printf("Ich schlafe jetzt %d stunden \n", dauer);

}

bool Person::trinken() {

    return false;
}

*---------------------- Person.h --------------------*

#ifndef PERSON_H
#define PERSON_H
#include <stdlib.h>
#include <stdio.h>
#include <string>

using namespace std;

class Person {
public:
    Person();
    Person(string name, float alter);
    bool trinken();
    void schlafen(int dauer);
    virtual ~Person();

protected:
```

```
    string name;
    float alter;

private:
    bool geschlecht;


};

#endif /* PERSON_H */

*------------------------- main von Vererbung ----------------*

#include <cstdlib>
#include "Person.h"
#include "Baby.h"
#include "Adult.h"

using namespace std;

/*
 *
 */
int main(int argc, char** argv) {
    Baby bae("jeff", 2);
    bae.schlafen(5);
    Adult ahmed(false);


    return 0;
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Error using dbstatus
Error: File: C:\PR2\PR2CodefuerKlausur.m Line: 1 Column: 1
Invalid use of operator.
```

Published with MATLAB® R2018b