```cpp
#include <Eigene Pakete in C und Cplusplus/Cplusplus/Eingabe/eingabe.h>
#include <Eigene Pakete in C und Cplusplus/Cplusplus/Dreieck/dreieck.h>

char Text[1];

const char *Anzahl_der_Daten_Textdatei = "Aufgabe 3 Anzahl der Daten5.txt",*Ausgabe_Textdatei = "Ausgabe Aufgabe 3 test5.txt";

int main()
{
    eingabe Eingabe;
    dreieck Dreieck;

    Eingabe.~eingabe();
    Dreieck.~dreieck();

    Eingabe.eingabe_int_aus_Textdatei(Anzahl_der_Daten_Textdatei);
    Eingabe.ausgabe_Eingabe.var_Standard_Ausgabe.Anzahl_der_Daten = Eingabe.ausgabe_Eingabe.var_Standard_Ausgabe.Wert;

    Dreieck.Standardwerte();

    Dreieck.korrekte_Werte();

    Dreieck.korrekte_Punkte();

    return(0);
}

```

```
#ifndef AUSGABE_H
#define AUSGABE_H

#include <stdio.h>

#include <Eigene Pakete in C und Cplusplus/Cplusplus/Definition/varStandard.h>

class ausgabe
{
    public:
        varStandard var_Standard_Ausgabe;
        ausgabe();
        virtual ~ausgabe();
        void ausgabe_in_Textdatei(const char *Name_der_Textdatei,const char *Art,const char *Text);
        void speichern_in_Textdatei(const char *Name_der_Textdatei,const char *Text);
        void ausgabe_Text(const char *Text);
    protected:
    private:
        char *Art;
};

#endif // AUSGABE_H
```

```cpp
#include "ausgabe.h"

ausgabe::ausgabe()
{
    var_Standard_Ausgabe.~varStandard();
}

ausgabe::~ausgabe()
{
    //dtor
}

void ausgabe::ausgabe_in_Textdatei(const char *Name_der_Textdatei,const char *Art,const char *Text)
{
    FILE *aus_datei = fopen(Name_der_Textdatei,Art);

    fprintf(aus_datei,Text);
    fclose(aus_datei);
}

void ausgabe::speichern_in_Textdatei(const char *Name_der_Textdatei,const char *Text)
{
    Art = "w";

    if(var_Standard_Ausgabe.Anzahl_der_Daten != 0)
        Art = "a";
    var_Standard_Ausgabe.Anzahl_der_Daten++;
    ausgabe_in_Textdatei(Name_der_Textdatei,Art,Text);
}

void ausgabe::ausgabe_Text(const char *Text)
{
    printf("%s",Text);
}
```

```cpp
#ifndef DREIECK_H
#define DREIECK_H

#include <Eigene Pakete in C und Cplusplus/Cplusplus/Punkt/punkt.h>

class dreieck
{
    public:
        punkt Punkt_Dreieck;
        dreieck();
        virtual ~dreieck();
        void Standardwerte();
        void korrekte_Werte();
        void korrekte_Punkte();
    protected:
    private:
        int f_Anzahl_der_Verfeinerungen,p1_p2_Punkte_pruefen,p2_p3_Punkte_pruefen,p3_p1_Punkte_pruefen;
        int f_Anzahl_der_Dreiecke[2],Dreicke_pruefen[1];
        double x[4][3],y[4][3];
        point p1,p2,p3,p4,p5,p6;
        void f_p0_Koordinaten_setzen();
        void f_p1_Koordinaten_setzen();
        void f_p2_Koordinaten_setzen();
        void Anzahl_der_Verfeinerungen_setzen();
        void korrekte_Anzahl_der_Verfeinerungen();
        void Anzahl_der_Dreiecke_setzen();
        void korrekte_Anzahl_der_Dreiecke();
        void null_Koordinaten_setzen();
        void Punkte_setzen_1_Dreieck();
        void eins_Koordinaten_setzen();
        void Punkte_setzen_2_Dreiecke();
        void zwei_Koordinaten_setzen();
        void Punkte_setzen_3_Dreiecke();
        void drei_Koordinaten_setzen();
        void Punkte_setzen_4_Dreiecke();
        void Dreiecke_ueberpruefen();
};

#endif // DREIECK_H
```

```
#include "dreieck.h"

dreieck::dreieck()
{
    f_Anzahl_der_Verfeinerungen = 0;
    p1_p2_Punkte_pruefen = 0;
    p2_p3_Punkte_pruefen = 0;
    p3_p1_Punkte_pruefen = 0;
    Punkt_Dreieck.~punkt();
}

dreieck::~dreieck()
{
    //dtor
}

void dreieck::f_p0_Koordinaten_setzen()
{
    x[0][0] = 0.0;
    y[0][0] = 0.0;
    x[1][0] = 0.0;
    y[1][0] = 0.0;
    x[2][0] = 0.0;
    y[2][0] = 0.0;
    x[3][0] = 1.0;
    y[3][0] = 0.0;
}

void dreieck::f_p1_Koordinaten_setzen()
{
    x[0][1] = 1.0;
    y[0][1] = 0.0;
    x[1][1] = 1.0;
    y[1][1] = 0.0;
    x[2][1] = 0.5;
    y[2][1] = 1.0;
    x[3][1] = 0.5;
    y[3][1] = 1.0;
}

void dreieck::f_p2_Koordinaten_setzen()
{
    x[0][2] = 0.5;
    y[0][2] = 1.0;
    x[1][2] = 0.5;
    y[1][2] = -1.0;
    x[2][2] = 0.0;
    y[2][2] = 0.5;
    x[3][2] = 1.0;
    y[3][2] = 0.5;
}

void dreieck::Standardwerte()
{
    f_Anzahl_der_Verfeinerungen = 1;

    f_Anzahl_der_Dreiecke[0] = 1;
    f_Anzahl_der_Dreiecke[1] = 4;

    for(int i = 0;i < 3;i++)
    {
        switch(i)
        {
            case 0:f_p0_Koordinaten_setzen();
                   break;
            case 1:f_p1_Koordinaten_setzen();
```

```
                    break;
            case 2:f_p2_Koordinaten_setzen();
                    break;
        }
    }

    for(int i = 0;i < 4;i++)
    {
        for(int j = 0;j < 3;j++)
        {
            Punkt_Dreieck.var_Standard_Punkt.x = x[i][j];
            Punkt_Dreieck.var_Standard_Punkt.y = y[i][j];
            Punkt_Dreieck.set_point();
            Punkt_Dreieck.f_p[i][j] = Punkt_Dreieck.temp;
        }
    }
}

void dreieck::Anzahl_der_Verfeinerungen_setzen()
{
    Punkt_Dreieck.var_Standard_Punkt.Anzahl_der_Verfeinerungen = 0;
}

void dreieck::korrekte_Anzahl_der_Verfeinerungen()
{
    Anzahl_der_Verfeinerungen_setzen();
    if(Punkt_Dreieck.var_Standard_Punkt.Anzahl_der_Verfeinerungen <= 0)
        Punkt_Dreieck.var_Standard_Punkt.Anzahl_der_Verfeinerungen = f_Anzahl_der_Verfeinerungen;
}

void dreieck::Anzahl_der_Dreiecke_setzen()
{
    Punkt_Dreieck.var_Standard_Punkt.Anzahl_der_Dreiecke = 0;
}

void dreieck::korrekte_Anzahl_der_Dreiecke()
{
    Anzahl_der_Dreiecke_setzen();
    if(Punkt_Dreieck.var_Standard_Punkt.Anzahl_der_Dreiecke < 1)
        Punkt_Dreieck.var_Standard_Punkt.Anzahl_der_Dreiecke = f_Anzahl_der_Dreiecke[0];
    if(Punkt_Dreieck.var_Standard_Punkt.Anzahl_der_Dreiecke > 4)
        Punkt_Dreieck.var_Standard_Punkt.Anzahl_der_Dreiecke = f_Anzahl_der_Dreiecke[1];
}

void dreieck::korrekte_Werte()
{
    korrekte_Anzahl_der_Verfeinerungen();
    korrekte_Anzahl_der_Dreiecke();
}

void dreieck::null_Koordinaten_setzen()
{
    x[0][0] = 0.0;
    y[0][0] = 0.0;
    x[0][1] = 0.0;
    y[0][1] = 0.0;
    x[0][2] = 0.0;
    y[0][2] = 0.0;
}

void dreieck::Punkte_setzen_1_Dreieck()
{
    null_Koordinaten_setzen();
    for(int i = 0;i < 3;i++)
    {
        Punkt_Dreieck.var_Standard_Punkt.x = x[0][i];
```

```
133         Punkt_Dreieck.var_Standard_Punkt.y = y[0][i];
134         Punkt_Dreieck.set_point();
135         Punkt_Dreieck.p3[0][i] = Punkt_Dreieck.temp;
136     }
137 }
138 
139 void dreieck::eins_Koordinaten_setzen()
140 {
141     x[1][0] = 0.0;
142     y[1][0] = 0.0;
143     x[1][1] = 0.0;
144     y[1][1] = 0.0;
145     x[1][2] = 0.0;
146     y[1][2] = 0.0;
147 }
148 
149 void dreieck::Punkte_setzen_2_Dreiecke()
150 {
151     eins_Koordinaten_setzen();
152     Punkte_setzen_1_Dreieck();
153     for(int i = 0;i < 3;i++)
154     {
155         Punkt_Dreieck.var_Standard_Punkt.x = x[1][i];
156         Punkt_Dreieck.var_Standard_Punkt.y = y[1][i];
157         Punkt_Dreieck.set_point();
158         Punkt_Dreieck.p3[1][i] = Punkt_Dreieck.temp;
159     }
160 }
161 
162 void dreieck::zwei_Koordinaten_setzen()
163 {
164     x[2][0] = 0.0;
165     y[2][0] = 0.0;
166     x[2][1] = 0.0;
167     y[2][1] = 0.0;
168     x[2][2] = 0.0;
169     y[2][2] = 0.0;
170 }
171 
172 void dreieck::Punkte_setzen_3_Dreiecke()
173 {
174     zwei_Koordinaten_setzen();
175     Punkte_setzen_2_Dreiecke();
176     for(int i = 0;i < 3;i++)
177     {
178         Punkt_Dreieck.var_Standard_Punkt.x = x[2][i];
179         Punkt_Dreieck.var_Standard_Punkt.y = y[2][i];
180         Punkt_Dreieck.set_point();
181         Punkt_Dreieck.p3[2][i] = Punkt_Dreieck.temp;
182     }
183 }
184 
185 void dreieck::drei_Koordinaten_setzen()
186 {
187     x[3][0] = 0.0;
188     y[3][0] = 0.0;
189     x[3][1] = 0.0;
190     y[3][1] = 0.0;
191     x[3][2] = 0.0;
192     y[3][2] = 0.0;
193 }
194 
195 void dreieck::Punkte_setzen_4_Dreiecke()
196 {
197     drei_Koordinaten_setzen();
198     Punkte_setzen_3_Dreiecke();
```

```
199     for(int i = 0;i < 3;i++)
200     {
201         Punkt_Dreieck.var_Standard_Punkt.x = x[3][i];
202         Punkt_Dreieck.var_Standard_Punkt.y = y[3][i];
203         Punkt_Dreieck.set_point();
204         Punkt_Dreieck.p3[3][i] = Punkt_Dreieck.temp;
205     }
206 }
207
208 void dreieck::Dreiecke_ueberpruefen()
209 {
210     for(int i = 0;i < Punkt_Dreieck.var_Standard_Punkt.Anzahl_der_Dreiecke;i++)
211         Dreicke_pruefen[i] = 0;
212     for(int i = 0;i < Punkt_Dreieck.var_Standard_Punkt.Anzahl_der_Dreiecke;i++)
213     {
214         p1 = Punkt_Dreieck.p3[i][0];
215         p2 = Punkt_Dreieck.p3[i][1];
216         p3 = Punkt_Dreieck.p3[i][2];
217
218         Punkt_Dreieck.temp = p1;
219         Punkt_Dreieck.temp1 = p2;
220         Punkt_Dreieck.Punkte_ueberpruefen();
221         p1_p2_Punkte_pruefen = Punkt_Dreieck.var_Standard_Punkt.Punkte_pruefen;
222
223         Punkt_Dreieck.temp = p2;
224         Punkt_Dreieck.temp1 = p3;
225         Punkt_Dreieck.Punkte_ueberpruefen();
226         p2_p3_Punkte_pruefen = Punkt_Dreieck.var_Standard_Punkt.Punkte_pruefen;
227
228         Punkt_Dreieck.temp = p3;
229         Punkt_Dreieck.temp1 = p1;
230         Punkt_Dreieck.Punkte_ueberpruefen();
231         p3_p1_Punkte_pruefen = Punkt_Dreieck.var_Standard_Punkt.Punkte_pruefen;
232
233         if((p1_p2_Punkte_pruefen == 0) || (p2_p3_Punkte_pruefen == 0) || (p3_p1_Punkte_pruefen == 0))
234             Dreicke_pruefen[i] = 1;
235     }
236     for(int i = 0;i < Punkt_Dreieck.var_Standard_Punkt.Anzahl_der_Dreiecke - 1;i++)
237     {
238         if(Dreicke_pruefen[i] == 0)
239         {
240             p1 = Punkt_Dreieck.p3[i][0];
241             p2 = Punkt_Dreieck.p3[i][1];
242             p3 = Punkt_Dreieck.p3[i][2];
243             for(int j = i + 1;i < Punkt_Dreieck.var_Standard_Punkt.Anzahl_der_Dreiecke;j++)
244             {
245                 if(Dreicke_pruefen[j] == 0)
246                 {
247                     p4 = Punkt_Dreieck.p3[j][0];
248                     p5 = Punkt_Dreieck.p3[j][1];
249                     p6 = Punkt_Dreieck.p3[j][2];
250                 }
251             }
252         }
253     }
254 }
255
256 void dreieck::korrekte_Punkte()
257 {
258     switch(Punkt_Dreieck.var_Standard_Punkt.Anzahl_der_Dreiecke)
259     {
260         case 1:Punkte_setzen_1_Dreieck();
261                break;
262         case 2:Punkte_setzen_2_Dreiecke();
263                break;
264         case 3:Punkte_setzen_3_Dreiecke();
```

```
265                 break;
266         case 4:Punkte_setzen_4_Dreiecke();
267                 break;
268     }
269 }
```

```h
#ifndef EINGABE_H
#define EINGABE_H

#include <Eigene Pakete in C und Cplusplus/Cplusplus/Definition/konstanten.h>
#include <Eigene Pakete in C und Cplusplus/Cplusplus/Ausgabe/ausgabe.h>

class eingabe
{
    public:
        ausgabe ausgabe_Eingabe;
        konstanten konstanten_Eingabe;
        eingabe();
        virtual ~eingabe();
        void eingabe_int_aus_Textdatei(const char *Name_der_Textdatei);
    protected:
    private:
        char Text[1];
};

#endif // EINGABE_H
```

```cpp
#include "eingabe.h"

eingabe::eingabe()
{
    ausgabe_Eingabe.~ausgabe();
    konstanten_Eingabe.~konstanten();
}

eingabe::~eingabe()
{
    //dtor
}

void eingabe::eingabe_int_aus_Textdatei(const char *Name_der_Textdatei)
{
    ausgabe_Eingabe.var_Standard_Ausgabe.Wert = 0;

    FILE *in_datei = fopen(Name_der_Textdatei,"r");

    if(in_datei == 0)
    {
        sprintf(Text,"Das Problem: Die Eingabe-Datei %s existiert nicht!%s",Name_der_Textdatei,
konstanten_Eingabe.Schraegstrichn);
        ausgabe_Eingabe.ausgabe_Text(Text);
        sprintf(Text,"%d%s",ausgabe_Eingabe.var_Standard_Ausgabe.Wert,konstanten_Eingabe.Schraegstrichn);
        ausgabe_Eingabe.ausgabe_in_Textdatei(Name_der_Textdatei,"w",Text);
    }
    fscanf(in_datei,"%d",&ausgabe_Eingabe.var_Standard_Ausgabe.Wert);
    fclose(in_datei);
}
```

```
#ifndef KONSTANTEN_H
#define KONSTANTEN_H


class konstanten
{
    public:
        char *Schraegstrichn;
        konstanten();
        virtual ~konstanten();
    protected:
    private:
};

#endif // KONSTANTEN_H
```

```cpp
#include "konstanten.h"

konstanten::konstanten()
{
    Schraegstrichn = "\n";
}

konstanten::~konstanten()
{
    //dtor
}
```

```
#ifndef PUNKT_H
#define PUNKT_H

#include <Eigene Pakete in C und Cplusplus/Cplusplus/Definition/varStandard.h>

typedef struct
    {
        double x;
        double y;
    }point;

class punkt
{
    public:
        varStandard var_Standard_Punkt;
        point temp,temp1;
        point f_p[4][3],p3[1][3];
        punkt();
        virtual ~punkt();
        void set_point();
        void Punkte_ueberpruefen();
    protected:
    private:
};

#endif // PUNKT_H
```

```cpp
#include "punkt.h"

punkt::punkt()
{
    var_Standard_Punkt.~varStandard();
}

punkt::~punkt()
{
    //dtor
}

void punkt::set_point()
{
    temp.x = var_Standard_Punkt.x;
    temp.y = var_Standard_Punkt.y;
}

void punkt::Punkte_ueberpruefen()
{
    var_Standard_Punkt.Punkte_pruefen = 0;
    if((temp.x != temp1.x) || (temp.y != temp1.y))
        var_Standard_Punkt.Punkte_pruefen = 1;
}
```

```
1  #ifndef VARSTANDARD_H
2  #define VARSTANDARD_H
3
4  class varStandard
5  {
6      public:
7          int Zahl1,Zahl2,ggt,kgv,Anzahl_der_Daten,Wert,Punkte_pruefen,Anzahl_der_Verfeinerungen,
Anzahl_der_Dreiecke;
8          double x,y;
9          varStandard();
10         virtual ~varStandard();
11     protected:
12     private:
13 };
14
15 #endif // VARSTANDARD_H
```

```cpp
#include "varStandard.h"

varStandard::varStandard()
{
    Zahl1 = 0;
    Zahl2 = 0;
    ggt = 0;
    kgv = 0;
    Anzahl_der_Daten = 0;
    Wert = 0;
    Punkte_pruefen = 0;
    Anzahl_der_Verfeinerungen = 0;
    Anzahl_der_Dreiecke = 0;
    x = 0.0;
    y = 0.0;
}

varStandard::~varStandard()
{
    //dtor
}
```