```cpp
/*
 * Integrationsroutinen.cpp
 *
 *  Created on: 24.04.2017
 *      Author: mona
 */



#include <iostream>
#include <complex>
#include <cstdlib>
#include <vector>
#include <cmath>
#include <sstream>
#include <utility>
#include <math.h>
#include <fstream>
#include <functional>
using namespace std;


double F1(double x){
        if(x==0){
                        x=pow(10, -9);
                }

        double f1=exp(-x)/x;
        return f1;
}

double F2(double x){
        if(x==0){
                return 0;
        }

        double f2=x*sin(1/x);
        return f2;
}

double trapez(double a, double b, double N, double(*f)(double)){
        double h=(b-a)/N;
        double summe=(h/2)*(f(a)+f(b));
        for (int n=1; n<N; n++){
                summe=summe+h*f(a+n*h);
        };
        return summe;
}

double mittelpunkt(double a, double b, double N, double(*f)(double)){
        double h=(b-a)/N;
        double summe=0;
                for (int n=0; n<N; n++){
                        summe=summe+f(a+(h/2)+(n*h));
                };
        return h*summe;
}


double simpson(double a, double b, double N, double(*f)(double)){
        double h=(b-a)/N;
        double summe= f(a)+f(b);
        for (int n=1; n<N; n++){
                                if(n%2==0){summe=summe+2*f(a+h*n);}
                                else{summe=summe+4*f(a+h*n);}
        }
        return (h/3)*summe;
}


double eps=pow(10, -4);

int main(){

        int a1=1;
        int b1=100;
        int a2=0;
        int b2=1;
        double Delta = 1;
        double N=2;
        double links;
        double rechts;

        //Abspeichern der Daten I1 mit Trapezregel
        ofstream a;
        a.open("I1_Trapez.txt");
```

```
87:         a.precision(10);
88:         //
89:
90:         while(Delta>eps){
91:                 links=trapez(a1,b1,N/2,F1);
92:                 rechts=trapez(a1,b1,N,F1);
93:                 Delta=abs(rechts-links)/links;
94:                 a << N/2 << "\t" << links <<  "\n";
95:                 N=2*N;
96:         }
97:         a.close();
98:
99:
100:        //Abspeichern der Daten I2 mit Trapezregel
101:        ofstream b;
102:        b.open("I2_Trapez.txt");
103:        b.precision(10);
104:        //
105:
106:        N=2;
107:        Delta=1;
108:
109:        while(Delta>eps){
110:                links=trapez(a2,b2,N/2,F2);
111:                rechts=trapez(a2,b2,N,F2);
112:                Delta=abs(rechts-links)/links;
113:                b << N/2 << "\t" << links <<  "\n";
114:                N=2*N;
115:        }
116:        b.close();
117:
118:        //Abspeichern der Daten I1 mit Mittelpunktsregel
119:        ofstream c;
120:        c.open("I1_Mittelpunkt.txt");
121:        c.precision(10);
122:        //
123:
124:        N=2;
125:        Delta=1;
126:
127:        while(Delta>eps){
128:                links=mittelpunkt(a1,b1,N/2,F1);
129:                rechts=mittelpunkt(a1,b1,N,F1);
130:                Delta=abs(rechts-links)/links;
131:                c << N/2 << "\t" << links <<  "\n";
132:                N=2*N;
133:        }
134:        c.close();
135:
136:
137:        //Abspeichern der Daten I2 mit Mittelpunktsregel
138:        ofstream d;
139:        d.open("I2_Mittelpunkt.txt");
140:        d.precision(10);
141:        //
142:
143:        N=2;
144:        Delta=1;
145:
146:        while(Delta>eps){
147:                links=mittelpunkt(a2,b2,N/2,F2);
148:                rechts=mittelpunkt(a2,b2,N,F2);
149:                Delta=abs(rechts-links)/links;
150:                d << N/2 << "\t" << links <<  "\n";
151:                N=2*N;
152:        }
153:        d.close();
154:
155:        //Abspeichern der Daten I1 mit Simpsonregel
156:        ofstream g;
157:        g.open("I1_Simpson.txt");
158:        g.precision(10);
159:        //
160:
161:        N=2;
162:        Delta=1;
163:
164:        while(Delta>eps){
165:                links=simpson(a1,b1,N/2,F1);
166:                rechts=simpson(a1,b1,N,F1);
167:                Delta=abs(rechts-links)/links;
168:                g << N/2 << "\t" << links <<  "\n";
169:                N=2*N;
170:        }
171:        g.close();
172:
```

```
173:
174:         //Abspeichern der Daten I2 mit Simpson
175:         ofstream k;
176:         k.open("I2_Simpson.txt");
177:         k.precision(10);
178:         //
179:
180:         N=2;
181:         Delta=1;
182:
183:         while(Delta>eps){
184:                 links=simpson(a2,b2,N/2,F2);
185:                 rechts=simpson(a2,b2,N,F2);
186:                 Delta=abs(rechts-links)/links;
187:                 k << N/2 << "\t" << links <<  "\n";
188:                 N=2*N;
189:         }
190:         k.close();
191:
192:
193:         return 0;
194: }
195:
196:
197:
198:
```

```cpp
/*
 * Punktladungen.cpp
 *
 *  Created on: 24.04.2017
 *      Author: mona
 */



#include <iostream>
#include <complex>
#include <cstdlib>
#include <vector>
#include <cmath>
#include <sstream>
#include <utility>
#include <math.h>
#include <fstream>
#include <functional>
using namespace std;

double Energie(double r, double a){
        double r1=sqrt(2*a*a-2*a*r+r*r);
        double r2=sqrt(2*a*a+2*a*r+r*r);
        double E=-2*((1/r1)+(1/r2));
        return E;
}

double zPunkt(double h, double r, double a){
        double f=Energie(r+h,a)-Energie(r-h,a);
        f=f/(2*h);
        return f;
}

double Kraft(double r, double a){
        double t1=sqrt(2*a*a-2*a*r+r*r)*(2*a*a-2*a*r+r*r); //entspricht ^3/2
        double t2=sqrt(2*a*a+2*a*r+r*r)*(2*a*a+2*a*r+r*r);
        t1=(-2*a+2*r)/t1;
        t2=(2*a+2*r)/t2;
        double F=t1+t2;
        return -F;
}

int main(){
        double a=1;
        double relFehler;
        //Abspeichern der Daten fÃ¼r die Energie
        ofstream b;
        b.open("Energie.txt");
        b.precision(10);
        //

        for(double n=-3*a; n<=3*a; n=n+0.001){
                b << n << "\t" << Energie(n,a) <<  "\n";
        }
        b.close();

        //Abspeichern der Daten
                ofstream c;
                c.open("Kraft_03a.txt");
                c.precision(10);
                //

                for(double n=-3*a; n<=3*a; n=n+0.001){
                        relFehler=(-zPunkt(0.3*a,n,a)-Kraft(n,a))/Kraft(n,a);
                        c << n << "\t" << -zPunkt(0.3*a,n,a) << "\t" << Kraft(n,a) << "\t" << relFehler
<<"\n";
                }
                c.close();

                //Abspeichern der Daten
                ofstream d;
                d.open("Kraft_E4a.txt");
                d.precision(10);
                //
                double h=pow(10,-4);
                for(double n=-3*a; n<=3*a; n=n+0.001){
                        relFehler=(-zPunkt(h*a,n,a)-Kraft(n,a))/Kraft(n,a);
                        d << n << "\t" << -zPunkt(h*a,n,a) << "\t" << Kraft(n,a)<< "\t" << relFehler  <<
 "\n";
                }
                d.close();

                //Abspeichern der Daten
                ofstream g;
                g.open("Kraft_E15a.txt");
```

```
   85:                 g.precision(10);
   86:                                 //
   87:                 h=pow(10,-15);
   88:                 for(double n=-3*a; n<=3*a; n=n+0.001){
   89:                         relFehler=(-zPunkt(h*a,n,a)-Kraft(n,a))/Kraft(n,a);
   90:                         g << n << "\t" << -zPunkt(h*a,n,a) << "\t" << Kraft(n,a) << "\t" << relFehler <<
"\n";
   91:                                 }
   92:                 g.close();
   93:
   94:
   95:         return 0;
   96: }
   97:
```