```cpp
// HFIntegrals1s.cpp (c) Sunday, October 27, 2024
// by James Pate Williams, Jr., BA, BS, MSWE, PhD
// Functions OverlapAB, KineticEnergyAB,
// NuclearAttractionAB, ElectronRepulsionABCD
// created from formulas in Appendix A of
// "Modern Quantum Chemistry Intrduction to
// Advanced Electronic Structure Theory" (c) 1996
// by Attila Szabo and Neil S. Ostlund
// Functions ERF, S, T, V, TwoE, translated from
// FORTRAN source code in Appendix B of above
// Functions ErrorFunction, nonexperfc are from
// "A Numerical Library in C for Scientists and
// Engineers" (c) 1996 by H. T. Lau, PhD
#include <stdio.h>
#include <fstream>
#include <iostream>
using namespace std;
// global character buffers
char buffer[16384] = { }, line[128] = { };
double fcoeff[4][4] =
{ { 0.0, 0.0, 0.0, 0.0 },
    { 0.0, 1.0, 0.0, 0.0},
    { 0.0, 0.678914, 0.430129, 0.0 },
    { 0.0, 0.444635, 0.535328, 0.154329 } };
double fexpon[4][4] =
{ { 0.0, 0.0, 0.0, 0.0 },
    { 0.0, 0.270950, 0.0, 0.0},
    { 0.0, 0.151623, 0.851819, 0.0 },
    { 0.0, 0.109818, 0.406771, 2.22766 } };
double coeffm[4][4], exponm[4][4];
// Euclidean distance norm_2
double Rab(double Ra[], double Rb[]) {
    double delta1 = Ra[0] - Rb[0];
    double delta2 = Ra[1] - Rb[1];
    double delta3 = Ra[2] - Rb[2];
    double norm_2 = sqrt(
        delta1 * delta1 +
        delta2 * delta2 +
        delta3 * delta3);
    return norm_2;
}
// Gaussian Type Orbital 1s overlap integral
// Appendix B translation from FORTRAN
double S(double a, double b, double rab2)
{
    // Calculates overlaps for unnormalized
    // Gaussian primitives
    double pi = 4.0 * atan(1.0);
    long double s = pow(pi / (a + b), 1.5);
    return s * exp(-a * b * rab2 / (a + b));
}
// Appendix B translation of FORTRAN
```

```
 53 double T(double a, double b, double rab2)
 54 {
 55     // Calculates the kinetic energy for
 56     // unnormalized Gaussian primitives
 57     double pi = 4.0 * atan(1.0);
 58     double t = a * b / (a + b);
 59     double u = 3.0 - 2.0 * a * b * rab2 / (a + b);
 60     double v = pow(pi / (a + b), 1.5);
 61     double w = exp(-a * b * rab2 / (a + b));
 62     return t * u * v * w;
 63 }
 64 // Appendix B translation
 65 double ERF(long double arg)
 66 {
 67     // Calculates the error function according
 68     // to a rational approximation from M.
 69     // Abramowitz and I. A. Stegun
 70     // Handbook of Mathematical Functions, Dover
 71     // Absolute error is less than 1.5*10^-7
 72     // Can be replaced by a built-in function
 73     // on some machines
 74     long double p = 0.3276911;
 75     long double a[] = {
 76         0.0,
 77         0.254829592, -0.284496736,
 78         1.421413741, -1.453152027,
 79         1.061405429 };
 80     long double t = 1.0 / (1.0 + p * arg);
 81     long double tn = t;
 82     long double poly = a[1] * tn;
 83
 84     for (int i = 2; i <= 5; i++)
 85     {
 86         tn *= t;
 87         poly = poly + a[i] * tn;
 88     }
 89
 90     return 1.0 - poly * exp(-arg * arg);
 91 }
 92 // Appendix B translation
 93 double F0A(double arg)
 94 {
 95     // Calculates the F function
 96     // F0 only (S-type orbitals)
 97
 98     double f0 = 0.0;
 99     double pi = 4.0 * atan(1.0);
100
101     if (arg < 1.0e-6)
102         goto Label10;
103
104     // F0 in terms of the error function
```

```
105
106     f0 = sqrt(pi / arg) * ERF(sqrt(arg)) / 2.0;
107     goto Label20;
108
109 Label10:
110
111     f0 = 1.0 - arg / 3.0;
112
113 Label20:
114
115     return f0;
116 }
117 // Appendix B translation
118 double V(double a, double b, double rab2,
119     double rcp2, double zc)
120 {
121     // Calculates unnormalized nuclear
122     // attraction integrals
123     double pi = 4.0 * atan(1.0);
124     double u = 2.0 * pi / (a + b);
125     double v = F0A((a + b) * rcp2);
126     double w = exp(-a * b * rab2 / (a + b));
127     return -u * v * w * zc;
128 }
129 // Appendix B translation
130 double TwoE(
131     double a, double b, double c, double d,
132     double rab2, double rcd2, double rpq2)
133 {
134     // Calculates two electron integrals for
135     // unnormalized primitives a, b, c, d are
136     // exponents etc. rab2 is the distance
137     // from center A to center B, etc.
138     double pi = 4.0 * atan(1.0);
139     double twoE1 = 2.0 * pow(pi, 2.5);
140     double twoE2 = (a + b) * (c + d);
141     double twoE3 = sqrt(a + b + c + d);
142     double twoE4 = F0A(twoE2 * rpq2 /
143         (a + b + c + d));
144     double twoE5 = exp(-a * b * rab2 / (a + b) -
145         c * d * rcd2 / (c + d));
146     return twoE1 / (twoE2 * twoE3) * twoE4 * twoE5;
147 }
148 // from Appendix A in textbook
149 double OverlapAB(
150     double Ra[], double Rb[],
151     double alpha, double beta) {
152     double pi = 4.0 * atan(1.0);
153     double factor0 = alpha + beta;
154     double factor1 = pow(pi / factor0, 1.5);
155     double factor2 = -alpha * beta / factor0;
156     double rab = Rab(Ra, Rb);
```

```
157     double rab2 = rab * rab;
158     return factor1 * exp(factor2 * rab2);
159 }
160 // See Lau, Chapter 6
161 double nonexperfc(double);
162 // See Lau, Chapter 6
163 void ErrorFunction(double x, double* erf, double* erfc) {
164     if (x > 26.0) {
165         *erf = 1.0;
166         *erfc = 0.0;
167         return;
168     }
169     else if (x < -5.5) {
170         *erf = -1.0;
171         *erfc = 2.0;
172         return;
173     }
174     else {
175         double absx, c, p, q;
176         absx = fabs(x);
177         if (absx <= 0.5) {
178             c = x * x;
179             p = ((-0.356098437018154e-1 * c + 0.699638348861914e1) * c +
180                 0.219792616182942e2) * c + 0.242667955230532e3;
181             q = ((c + 0.150827976304078e2) * c + 0.911649054045149e2) * c +
182                 0.215058875869861e3;
183             *erf = x * p / q;
184             *erfc = 1.0 - (*erf);
185         }
186         else {
187             *erfc = exp(-x * x) * nonexperfc(absx);
188             *erf = 1.0 - (*erfc);
189             if (x < 0.0) {
190                 *erf = -(*erf);
191                 *erfc = 2.0 - (*erfc);
192             }
193         }
194     }
195 }
196 // See Lau, Chapter 6
197 double nonexperfc(double x) {
198     double absx, erf, erfc, c, p, q;
199
200     absx = fabs(x);
201     if (absx <= 0.5) {
202         ErrorFunction(x, &erf, &erfc);
203         return exp(x * x) * erfc;
204     }
205     else if (absx < 4.0) {
206         c = absx;
207         p = ((((((-0.136864857382717e-6 * c + 0.564195517478974e0) * c +
208             0.721175825088309e1) * c + 0.431622272220567e2) * c +
```

```
209             0.152989285046940e3) * c + 0.339320816734344e3) * c +
210             0.451918953711873e3) * c + 0.300459261020162e3;
211         q = ((((((c + 0.127827273196294e2) * c + 0.770001529352295e2) * c +
212             0.277585444743988e3) * c + 0.638980264465631e3) * c +
213             0.931354094850610e3) * c + 0.790950925327898e3) * c +
214             0.300459260956983e3;
215         return ((x > 0.0) ? p / q : exp(x * x) * 2.0 - p / q);
216     }
217     else {
218         c = 1.0 / x / x;
219         p = (((0.223192459734185e-1 * c + 0.278661308609648e0) * c +
220             0.226956593539687e0) * c + 0.494730910623251e-1) * c +
221             0.299610707703542e-2;
222         q = (((c + 0.198733201817135e1) * c + 0.105167510706793e1) * c +
223             0.191308926107830e0) * c + 0.106209230528468e-1;
224         c = (c * (-p) / q + 0.564189583547756) / absx;
225         return ((x > 0.0) ? c : exp(x * x) * 2.0 - c);
226     }
227 }
228 // Appendix B translation
229 double F0L(double arg)
230 {
231     // Calculates the F function
232     // F0 only (S-type orbitals)
233 
234     double erf = 0, erfc = 0, f0 = 0;
235     double pi = 4.0 * atan(1.0);
236 
237     if (arg < 1.0e-6)
238         goto Label10;
239     // F0 in terms of the error function
240     ErrorFunction(sqrt(arg), &erf, &erfc);
241     f0 = sqrt(pi / arg) * erf / 2.0;
242     goto Label20;
243 Label10:
244     f0 = 1.0 - arg / 3.0;
245 Label20:
246     return f0;
247 }
248 // Appendix A formula
249 double KineticEnergyAB(
250     double Ra[], double Rb[],
251     double alpha, double beta)
252 {
253     double pi = 4.0 * atan(1.0);
254     double factor0 = alpha + beta;
255     double factor1 = 2.0 * alpha * beta / factor0;
256     double factor2 = pow(Rab(Ra, Rb), 2.0);
257     double factor3 = factor1 * factor2;
258     double factor4 = pow(pi / factor0, 1.5);
259     double factor5 = alpha * beta / factor0;
260     double factor6 = factor5 * (3.0 - factor3) * factor4;
```

```
261     double factor7 = (alpha * beta / factor0) * factor2;
262     return factor6 * exp(-factor7);
263 }
264 // Appendix A formula
265 double NuclearAttractionAB(
266     double Ra[], double Rb[],
267     double Rc[], double Rp[],
268     double alpha, double beta,
269     double Zc) {
270     double pi = 4.0 * atan(1.0);
271     double factor0 = alpha + beta;
272     double factor1 = -2.0 * pi / factor0 * Zc;
273     double factor2 = pow(Rab(Ra, Rb), 2.0);
274     double factor3 = -alpha * beta / factor0 * factor2;
275     double factor4 = factor1 * exp(factor3);
276     double factor5 = pow(Rab(Rp, Rc), 2.0);
277     double x = (alpha + beta) * factor5;
278     double fop = F0L(x);
279     return factor4 * fop;
280 }
281 // Appendix A formula
282 double ElectronRepulsionABCD(
283     double Ra[], double Rb[],
284     double Rc[], double Rd[],
285     double Rp[], double Rq[],
286     double alpha, double beta,
287     double gamma, double delta) {
288     double pi = 4.0 * atan(1.0);
289     double factor0 = 2.0 * pow(pi, 2.5);
290     double factor1 = alpha + beta;
291     double factor2 = gamma + delta;
292     double factor3 = sqrt(factor1 + factor2);
293     double factor4 = factor0 /
294         (factor1 * factor2 * factor3);
295     double factor5 = -alpha * beta / factor1;
296     double factor6 = pow(Rab(Ra, Rb), 2.0);
297     double factor7 = factor5 * factor6;
298     double factor8 = -gamma * delta / factor2;
299     double factor9 = pow(Rab(Rc, Rd), 2.0);
300     double factora = factor8 * factor9;
301     double expont1 = exp(factor7 + factora);
302     double x = 0, f0p = 0;
303     x = factor1 * factor2 / (factor3 * factor3) *
304         pow(Rab(Rp, Rq), 2.0);
305     f0p = F0L(x);
306     return factor4 * expont1 * f0p;
307 }
308 // main function R, Za, Zb, zeta1, and zeta2 for H2
309 int main() {
310     int N = 1;
311     double pi = 4.0 * atan(1.0);
312     double R = 1.4011, Za = 1, Zb = 1;
```

```
313     double zeta1 = 1.24, zeta2 = 1.24;
314     double a1 = fexpon[1][1] * pow(zeta1, 2.0);
315     double d1 = fcoeff[1][1] * pow(2.0 * a1 / pi, 0.75);
316     double a2 = fexpon[1][1] * pow(zeta2, 2.0);
317     double d2 = fcoeff[1][1] * pow(2.0 * a2 / pi, 0.75);
318     double R2 = R * R;
319     double Ra[3] = { }, Rb[3] = { };
320     double Rc[3] = { }, Rd[3] = { };
321     double Rp[3] = { }, Rq[3] = { };
322     double rap = a2 * R / (a2 + a1);
323     double rbq = R - rap;
324     double raq = a2 * R / (a2 + a1);
325     double rpq = rap - raq;
326     double rap2 = rap * rap;
327     double rbp2 = rbq * rbq;
328     double raq2 = raq * raq;
329     double rbq2 = rbq * rbq;
330     double rpq2 = rpq * rpq;
331     // compute overlap integral
332     double s12 = S(zeta1, zeta2, R2);
333     // compute kinetic energies
334     double t11 = T(a1, a1, 0.0) * d1 * d1;
335     double t12 = T(a1, a2, R2) * d1 * d2;
336     double t22 = T(a2, a2, 0.0) * d2 * d2;
337     // compute attraction on center A
338     double v11a = V(a1, a1, 0.0, 0.0, Za) * d1 * d1;
339     double v12a = V(a1, a2, R2, rap2, Za) * d1 * d2;
340     double v22a = V(a2, a2, 0.0, R2, Za) * d2 * d2;
341     // compute attraction on center B
342     double v11b = V(a1, a1, 0.0, R2, Zb) * d1 * d1;
343     double v12b = V(a1, a2, R2, rbp2, Zb) * d1 * d2;
344     double v22b = V(a2, a2, 0.0, 0.0, Zb) * d2 * d2;
345     // compute two-electron integrals
346     double v1111 = TwoE(a1, a1, a1, a1,
347         0.0, 0.0, 0.0) * d1 * d1 * d1 * d1;
348     double v2111 = TwoE(a2, a1, a1, a1,
349         R2, 0.0, rap2) * d2 * d1 * d1 * d1;
350     double v2121 = TwoE(a2, a1, a2, a1,
351         R2, R2, rpq2) * d2 * d1 * d2 * d1;
352     double v2211 = TwoE(a2, a2, a1, a1,
353         0.0, 0.0, R2) * d2 * d2 * d1 * d1;
354     double v2221 = TwoE(a2, a2, a2, a1,
355         0.0, R2, rbq2) * d2 * d2 * d2 * d1;
356     double v2222 = TwoE(a2, a2, a2, a2,
357         0.0, 0.0, 0.0) * d2 * d2 * d2 * d2;
358     // appendix A computations
359     Ra[0] = Ra[1] = Ra[2] = Rb[1] = Rb[2] = 0;
360     Rb[0] = R;
361     // compute overlap integral
362     double sAB = OverlapAB(Ra, Rb, zeta1, zeta2);
363     // compute kinetic energy integrals
364     double T11 = d1 * d1 *
```

```
365         KineticEnergyAB(Ra, Ra, a1, a1);
366     double T12 = d1 * d2 *
367         KineticEnergyAB(Ra, Rb, a1, a2);
368     double T22 = d2 * d2 *
369         KineticEnergyAB(Rb, Rb, a2, a2);
370     // compute nuclear-electron attraction
371     Ra[0] = Ra[1] = Ra[2] = 0;
372     Rb[0] = Rb[1] = Rb[2] = 0;
373     Rc[0] = Rc[1] = Rc[2] = 0;
374     Rp[0] = Rp[1] = Rp[2] = 0;
375     double V11a = NuclearAttractionAB(
376         Ra, Rb, Rc, Rp, a1, a1, Za) * d1 * d1;
377     Ra[0] = Ra[1] = Ra[2] = 0;
378     Rb[0] = Rb[1] = Rb[2] = 0;
379     Rp[0] = Rp[1] = Rp[2] = 0;
380     Rb[0] = R;
381     Rp[0] = rap;
382     double V12a = NuclearAttractionAB(
383         Ra, Rb, Rc, Rp, a1, a2, Za) * d1 * d2;
384     Ra[0] = Ra[1] = Ra[2] = 0;
385     Rb[0] = Rb[1] = Rb[2] = 0;
386     Rp[0] = Rp[1] = Rp[2] = 0;
387     Rc[0] = R;
388     double V22a = NuclearAttractionAB(
389         Ra, Rb, Rc, Rp, a2, a2, Za) * d2 * d2;
390     Ra[0] = Ra[1] = Ra[2] = 0;
391     Rb[0] = Rb[1] = Rb[2] = 0;
392     Rp[0] = Rp[1] = Rp[2] = 0;
393     Rc[0] = R;
394     double V11b = NuclearAttractionAB(
395         Ra, Rb, Rc, Rp, a1, a1, Zb) * d1 * d1;
396     Ra[0] = Ra[1] = Ra[2] = 0;
397     Rb[0] = Rb[1] = Rb[2] = 0;
398     Rc[0] = Rc[1] = Rc[2] = 0;
399     Rp[0] = Rp[1] = Rp[2] = 0;
400     Rb[0] = R;
401     Rc[0] = rbq;
402     double V12b = NuclearAttractionAB(
403         Ra, Rb, Rc, Rp, a1, a2, Zb) * d1 * d2;
404     Ra[0] = Ra[1] = Ra[2] = 0;
405     Rb[0] = Rb[1] = Rb[2] = 0;
406     Rc[0] = Rc[1] = Rc[2] = 0;
407     Rp[0] = Rp[1] = Rp[2] = 0;
408     double V22b = NuclearAttractionAB(
409         Ra, Rb, Rc, Rp, a2, a2, Zb) * d2 * d2;
410     // compute two-electron integrals
411     Ra[0] = Ra[1] = Ra[2] = 0;
412     Rb[0] = Rb[1] = Rb[2] = 0;
413     Rc[0] = Rc[1] = Rc[2] = 0;
414     Rd[0] = Rd[1] = Rd[2] = 0;
415     Rp[0] = Rp[1] = Rp[2] = 0;
416     Rq[0] = Rq[1] = Rq[2] = 0;
```

```
417     double V1111 = ElectronRepulsionABCD(
418         Ra, Rb, Rc, Rd, Rp, Rq,
419         a1, a1, a1, a1) * d1 * d1 * d1 * d1;
420     Ra[0] = Ra[1] = Ra[2] = 0;
421     Rb[0] = Rb[1] = Rb[2] = 0;
422     Rc[0] = Rc[1] = Rc[2] = 0;
423     Rd[0] = Rd[1] = Rd[2] = 0;
424     Rp[0] = Rp[1] = Rp[2] = 0;
425     Rq[0] = Rq[1] = Rq[2] = 0;
426     Ra[0] = R;
427     Rp[0] = rap;
428     double V2111 = ElectronRepulsionABCD(
429         Ra, Rb, Rc, Rd, Rp, Rq,
430         a2, a1, a1, a1) * d2 * d1 * d1 * d1;
431     Ra[0] = Ra[1] = Ra[2] = 0;
432     Rb[0] = Rb[1] = Rb[2] = 0;
433     Rc[0] = Rc[1] = Rc[2] = 0;
434     Rd[0] = Rd[1] = Rd[2] = 0;
435     Rp[0] = Rp[1] = Rp[2] = 0;
436     Rq[0] = Rq[1] = Rq[2] = 0;
437     Ra[0] = R;
438     Rc[0] = R;
439     Rp[0] = rpq;
440     double V2121 = ElectronRepulsionABCD(
441         Ra, Rb, Rc, Rd, Rp, Rq,
442         a2, a1, a2, a1) * d2 * d1 * d2 * d1;
443     Ra[0] = Ra[1] = Ra[2] = 0;
444     Rb[0] = Rb[1] = Rb[2] = 0;
445     Rc[0] = Rc[1] = Rc[2] = 0;
446     Rd[0] = Rd[1] = Rd[2] = 0;
447     Rp[0] = Rp[1] = Rp[2] = 0;
448     Rq[0] = Rq[1] = Rq[2] = 0;
449     Rp[0] = R;
450     double V2211 = ElectronRepulsionABCD(
451         Ra, Rb, Rc, Rd, Rp, Rq,
452         a2, a2, a1, a1) * d2 * d2 * d1 * d1;
453     Ra[0] = Ra[1] = Ra[2] = 0;
454     Rb[0] = Rb[1] = Rb[2] = 0;
455     Rc[0] = Rc[1] = Rc[2] = 0;
456     Rd[0] = Rd[1] = Rd[2] = 0;
457     Rp[0] = Rp[1] = Rp[2] = 0;
458     Rq[0] = Rq[1] = Rq[2] = 0;
459     Rc[0] = R;
460     Rp[0] = rbq;
461     double V2221 = ElectronRepulsionABCD(
462         Ra, Rb, Rc, Rd, Rp, Rq,
463         a2, a2, a2, a1) * d2 * d2 * d2 * d1;
464     Ra[0] = Ra[1] = Ra[2] = 0;
465     Rb[0] = Rb[1] = Rb[2] = 0;
466     Rc[0] = Rc[1] = Rc[2] = 0;
467     Rd[0] = Rd[1] = Rd[2] = 0;
468     Rp[0] = Rp[1] = Rp[2] = 0;
```

```
469     Rq[0] = Rq[1] = Rq[2] = 0;
470     double V2222 = ElectronRepulsionABCD(
471         Ra, Rb, Rc, Rd, Rp, Rq,
472         a2, a2, a2, a2) * d2 * d2 * d2 * d2;
473     ofstream ofile;
474     ofile.open("HFIntegralsResults.txt",
475         ofstream::out | ostream::ate | ostream::trunc);
476     sprintf_s(line, "STO-%ldG for Atomic Numbers %ld and %ld\n",
477         N, (int)Za, (int)Zb);
478     strcat_s(buffer, line);
479     sprintf_s(line, "R = Rab = %12.6e\n", R);
480     strcat_s(buffer, line);
481     sprintf_s(line, "Zeta 1 = %12.6e\t", zeta1);
482     strcat_s(buffer, line);
483     sprintf_s(line, "Zeta 2 = %12.6e\n\n", zeta2);
484     strcat_s(buffer, line);
485     sprintf_s(line, "Overlap integral = %12.6e\n\n", s12);
486     strcat_s(buffer, line);
487     sprintf_s(line, "Kinetic energy integrals:\n\n");
488     strcat_s(buffer, line);
489     sprintf_s(line, "t11 = %+12.6e\t", t11);
490     strcat_s(buffer, line);
491     sprintf_s(line, "t12 = %+12.6e\n", t12);
492     strcat_s(buffer, line);
493     sprintf_s(line, "t21 = %+12.6e\t", t12);
494     strcat_s(buffer, line);
495     sprintf_s(line, "t22 = %+12.6e\n\n", t22);
496     strcat_s(buffer, line);
497     sprintf_s(line, "Potential energy integrals on center A:\n\n");
498     strcat_s(buffer, line);
499     sprintf_s(line, "v11-A = %12.6e\t", v11a);
500     strcat_s(buffer, line);
501     sprintf_s(line, "v12-A = %12.6e\n", v12a);
502     strcat_s(buffer, line);
503     sprintf_s(line, "v21-A = %12.6e\t", v12a);
504     strcat_s(buffer, line);
505     sprintf_s(line, "v22-A = %12.6e\n\n", v22a);
506     strcat_s(buffer, line);
507     sprintf_s(line, "Potential energy integrals on center B:\n\n");
508     strcat_s(buffer, line);
509     sprintf_s(line, "v11-B = %12.6e\t", v11b);
510     strcat_s(buffer, line);
511     sprintf_s(line, "v12-B = %12.6e\n", v12b);
512     strcat_s(buffer, line);
513     sprintf_s(line, "v21-B = %12.6e\t", v12b);
514     strcat_s(buffer, line);
515     sprintf_s(line, "v22-B = %12.6e\n\n", v22b);
516     strcat_s(buffer, line);
517     sprintf_s(line, "Direct and exchange integrals:\n\n");
518     strcat_s(buffer, line);
519     sprintf_s(line, "v[1, 1, 1, 1] = %12.6e\t", v1111);
520     strcat_s(buffer, line);
```

```
521     sprintf_s(line, "v[2, 1, 1, 1] = %12.6e\n", v2111);
522     strcat_s(buffer, line);
523     sprintf_s(line, "v[2, 1, 2, 1] = %12.6e\t", v2121);
524     strcat_s(buffer, line);
525     sprintf_s(line, "v[2, 2, 1, 1] = %12.6e\n", v2211);
526     strcat_s(buffer, line);
527     sprintf_s(line, "v[2, 2, 2, 1] = %12.6e\t", v2221);
528     strcat_s(buffer, line);
529     sprintf_s(line, "v[2, 2, 2, 2] = %12.6e\n", v2222);
530     strcat_s(buffer, line);
531     sprintf_s(line, "\nOverlap integral = %12.6e\n\n", sAB);
532     strcat_s(buffer, line);
533     sprintf_s(line, "Kinetic energy integrals:\n\n");
534     strcat_s(buffer, line);
535     sprintf_s(line, "T11 = %+12.6e\t", T11);
536     strcat_s(buffer, line);
537     sprintf_s(line, "T12 = %+12.6e\n", T12);
538     strcat_s(buffer, line);
539     sprintf_s(line, "T21 = %+12.6e\t", T12);
540     strcat_s(buffer, line);
541     sprintf_s(line, "T22 = %+12.6e\n\n", T22);
542     strcat_s(buffer, line);
543     sprintf_s(line, "Potential energy integrals on center A:\n\n");
544     strcat_s(buffer, line);
545     sprintf_s(line, "V11-A = %12.6e\t", V11a);
546     strcat_s(buffer, line);
547     sprintf_s(line, "V12-A = %12.6e\n", V12a);
548     strcat_s(buffer, line);
549     sprintf_s(line, "V21-A = %12.6e\t", V12a);
550     strcat_s(buffer, line);
551     sprintf_s(line, "V22-A = %12.6e\n\n", V22a);
552     strcat_s(buffer, line);
553     sprintf_s(line, "Potential energy integrals on center B:\n\n");
554     strcat_s(buffer, line);
555     sprintf_s(line, "V11-B = %12.6e\t", V11b);
556     strcat_s(buffer, line);
557     sprintf_s(line, "V12-B = %12.6e\n", V12b);
558     strcat_s(buffer, line);
559     sprintf_s(line, "V21-B = %12.6e\t", V12b);
560     strcat_s(buffer, line);
561     sprintf_s(line, "V22-B = %12.6e\n\n", V22b);
562     strcat_s(buffer, line);
563     sprintf_s(line, "Direct and exchange integrals:\n\n");
564     strcat_s(buffer, line);
565     sprintf_s(line, "V[1, 1, 1, 1] = %12.6e\t", V1111);
566     strcat_s(buffer, line);
567     sprintf_s(line, "V[2, 1, 1, 1] = %12.6e\n", V2111);
568     strcat_s(buffer, line);
569     sprintf_s(line, "V[2, 1, 2, 1] = %12.6e\t", V2121);
570     strcat_s(buffer, line);
571     sprintf_s(line, "V[2, 2, 1, 1] = %12.6e\n", V2211);
572     strcat_s(buffer, line);
```

```
573     sprintf_s(line, "V[2, 2, 2, 1] = %12.6e\t", V2221);
574     strcat_s(buffer, line);
575     sprintf_s(line, "V[2, 2, 2, 2] = %12.6e\n\n", V2222);
576     strcat_s(buffer, line);
577     sprintf_s(line, "Comparison of Two Error Functions\n");
578     strcat_s(buffer, line);
579     sprintf_s(line, "ERF from Abramowitz and Stegan\n");
580     strcat_s(buffer, line);
581     sprintf_s(line, "erf from C Translation of NUMAL by H. T. Lau\n");
582     strcat_s(buffer, line);
583     sprintf_s(line, "Difference below is the percent difference\n\n");
584     strcat_s(buffer, line);
585     sprintf_s(line, "x\t\tERF\t\terf\t\tDifference\n\n");
586     strcat_s(buffer, line);
587     double h = 1.0 / 10.0, x = 0;
588     for (int i = 0; i <= 10; i++)
589     {
590         double erf = 0, erfc = 0;
591         ErrorFunction(x, &erf, &erfc);
592         double erfa = ERF(x);
593         double num = fabs(erf - erfa);
594         double den = 0.5 * (erf + erfa);
595         double dif = 100.0 * num / den;
596         if (x == 0) dif = 0;
597         sprintf_s(line, "%lf\t%lf\t%lf\t%lf\n",
598             x, erfa, erf, dif);
599         strcat_s(buffer, line);
600         x += h;
601     }
602     ofile << buffer << endl;
603     ofile.close();
604     return 0;
605 }
```