```cpp
// Interpolation.cpp (c) Friday, October 25, 2024
// by James Pate Williams, Jr., BA, BS, MSWE, PhD
// See "Elementary Numerical Analysis: An Algorithmic
// Approach Third Edition" (c) 1980 by S. D. Conte
// and Carl de Boor Chapter 2 Interpolation by
// Polynomials

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

double f(double y) {
    return 1.0 / (1.0 + y * y);
}

void Example_2_4() {
    vector<double> d(20, 0), x(20, 0);

    std::cout << "n\tMaximum Error" << endl;

    for (int n = 2; n <= 16; n += 2) {
        int np1 = n + 1;
        double h = 10.0 / n, pnofy = 0;

        for (int i = 1; i <= np1; i++) {
            x[i] = (i - 1.0) * h - 5.0;
            d[i] = f(x[i]);
        }

        for (int k = 1; k <= n; k++) {
            for (int i = 1; i <= np1 - k; i++) {
                d[i] = (d[i + 1] - d[i]) /
                    (x[i + k] - x[i]);
            }
        }

        double errmax = 0.0;

        for (int j = 1; j <= 101; j++) {
            double y = (j - 1.0) / 10.0 - 5.0;
            pnofy = d[1];

            for (int k = 2; k <= np1; k++)
                pnofy = d[k] + (y - x[k]) * pnofy;

            double delta = fabs(f(y) - pnofy);
            errmax = fmax(fabs(f(y) - pnofy), errmax);
        }

        std::cout << scientific;
        std::cout << n << "\t" << errmax << endl;
    }
```

```
 53  }
 54
 55  double InterpolationInaFunctionTable(
 56      double xbar, vector<double> f,
 57      vector<double> x, int ntable,
 58      double tolerance, int& flag) {
 59      double table = 0;
 60      int next = 0, nextl = 0, nextr = 0;
 61      vector<double> a(ntable + 2, 0);
 62      vector<double> xk(ntable + 2, 0);
 63
 64      if (xbar >= x[1] && xbar <= x[ntable])
 65          for (next = 2; next <= ntable; next++)
 66              if (xbar <= x[next])
 67                  goto Label12;
 68      if (xbar < x[1])
 69          table = f[1];
 70      else
 71          table = f[ntable];
 72      cout << scientific;
 73      cout << xbar << " not in table range." << endl;
 74      flag = 3;
 75      return 0.0;
 76  Label12:
 77      xk[1] = x[next];
 78      nextl = next - 1;
 79      nextr = next + 1;
 80      a[1] = f[next];
 81      table = a[1];
 82
 83      double psik = 1.0;
 84      int kp1max = min(20, ntable);
 85
 86      for (int kp1 = 2; kp1 <= kp1max; kp1++) {
 87          if (nextl == 0) {
 88              next = nextr;
 89              nextr++;
 90          }
 91
 92          else if (nextr > ntable) {
 93              next = nextl;
 94              nextl--;
 95          }
 96
 97          else if (xbar - x[nextl] > x[nextr] - xbar) {
 98              next = nextr;
 99              nextr++;
100          }
101
102          else {
103              next = nextl;
104              nextl--;
```

```
105         }
106
107         xk[kp1] = x[next];
108         a[kp1] = f[next];
109
110         for (int j = kp1 - 1; j >= 1; j--)
111             a[j] = (a[j + 1] - a[j]) /
112             (xk[kp1] - xk[j]);
113
114         psik = psik * (xbar - xk[kp1 - 1]);
115
116         double error = a[1] * psik;
117
118         cout << scientific;
119         cout << kp1 << "\t" << xk[kp1] << "\t";
120         cout << table << "\t" << error << endl;
121
122         table += error;
123
124         if (fabs(error) < tolerance) {
125             flag = 1;
126             return table;
127         }
128     }
129
130     cout << "No convergence in " << kp1max << "steps.";
131     cout << endl;
132
133     return 0;
134 }
135
136 double F(double x) {
137     return x * log(x);
138 }
139
140 void NewtonForm(
141     int np1,
142     int npoint,
143     double x,
144     double dx,
145     vector<double> f,
146     vector<double> y) {
147     int n = np1 - 1;
148
149     for (int k = 1; k <= n; k++) {
150         double realk = k;
151         double flast = f[1];
152
153         for (int i = 1; i <= np1 - k; i++) {
154             double dy = y[i + k] - y[i];
155
156             if (dy == 0)
```

```
157                 f[i] = f[i + 1] / realk;
158             else {
159                 f[i] = (f[i + 1] - flast) / dy;
160                 flast = f[i + 1];
161             }
162         }
163
164         f[np1 - k + 1] = flast;
165     }
166
167     for (int j = 1; j <= npoint; j++) {
168         double pnofx = f[1];
169
170         for (int i = 2; i <= np1; i++)
171             pnofx = f[i] + (x - y[i]) * pnofx;
172
173         cout << scientific;
174         cout << j << "\t" << x << "\t" << pnofx;
175         cout << endl;
176
177         x += dx;
178     }
179 }
180
181 int main() {
182     Example_2_4();
183     cout << endl;
184
185     vector<double> f(52, 0), x(52, 0);
186
187     for (int i = 1; i <= 50; i++) {
188         x[i] = 0.1 * i;
189         f[i] = F(x[i]);
190     }
191
192     double xbar = 0.55;
193     int flag = 0;
194
195     double ybar = InterpolationInaFunctionTable(
196         xbar, f, x, 50, 1.0e-8, flag);
197
198     cout << endl;
199
200     double dz = 0.05, zz = 0.05;
201     int np1 = 51, npoint = 10;
202     vector<double> g(np1 + 1, 0);
203     vector<double> z(np1 + 1, 0);
204
205     for (int i = 1; i <= np1; i++) {
206         z[i] = i * 0.01;
207         g[i] = z[i] * sin(z[i]);
208     }
```

```
209
210     NewtonForm(np1, npoint, zz, dz, g, z);
211     return 0;
212 }
```