```
    1: #include <iostream>
    2: #include <vector>                //  vector
    3: #include <cassert>
    4: using namespace std;
    5:
    6:
    7: /** \brief Berechnung von Folgengliedern
    8:  *         entsprechend der Formel
    9:  *         @f$ x_n = \frac{1}{n} \sum\limits_{k=1}^{n} \frac{2k+1}{2n
+1} \qquad\forall n=1,\ldots,nsize @f$.
   10:  *
   11:  * \param[in] nsize  Anzahl der zu berechnenden Glieder
   12:  * \return Vektor der Folgenglieder
   13:  *
   14:  */
   15: vector<double> folge(const int nsize)
   16: {
   17:     vector<double> x(nsize);
   18:
   19:     for (int n=1; n<=x.size(); ++n)
   20:     {
   21:         double tmp =0.0;             // nur innerhalb des n-Loops gue
ltig
   22:         for (int k=1; k<=n; ++k)
   23:         {
   24:             tmp += (2*k+1.0)/(2*n+1);
   25:         }
   26:         x[n-1] = tmp/n;                 // Indizierung von 0 bis (nsiz
e-1)
   27:     }
   28:     return x;
   29: }
   30:
   31: /** \brief Berechnung von Folgengliedern
   32:  *         entsprechend der Formel
   33:  *         @f$ x_n = \frac{1}{n} \sum\limits_{k=1}^{n} \frac{2k+1}{2n
+1} \qquad\forall n=1,\ldots,nsize @f$.
   34:  *
   35:  *         Es wird push_back() zur dynamischen VerlÃ¤ngerung des Vekt
ors benutzt.
   36:  *
   37:  * \param[in] nsize  Anzahl der zu berechnenden Glieder
   38:  * \return Vektor der Folgenglieder
   39:  *
   40:  */
   41: vector<double> folge_dynamisch(const int nsize)
   42: {
   43:     vector<double> x;                // Laenge 0
   44:
   45:     for (int n=1; n<=nsize; ++n)     // n i c h t    n<=x.size() !!
   46:     {
   47:         double tmp =0.0;             // nur innerhalb des n-Loops gue
ltig
   48:         for (int k=1; k<=n; ++k)
   49:         {
   50:             tmp += (2*k+1.0)/(2*n+1);
   51:         }
```

```
   52:         x.push_back(tmp/n);                 // Indizierung von 0 bis (
nsize-1)
   53:     }
   54:     return x;
   55: }
   56:
   57: /** \brief Berechnung von Folgengliedern
   58:  *         entsprechend der Formel
   59:  *         @f$ x_n = \frac{1}{n} \sum\limits_{k=1}^{n} \frac{2k+1}{2n
+1} \qquad\forall n=1,\ldots,nsize @f$.
   60:  *
   61:  *         Es wird push_back() zur dynamischen VerlÃ¤ngerung des Vekt
ors benutzt.
   62:  *
   63:  * \param[in] nsize  Anzahl der zu berechnenden Glieder
   64:  * \param[out] x     Vektor der Folgenglieder
   65:  *
   66:  */
   67: void folge_d2(const int nsize, vector<double> &x)
   68: {
   69:     x.clear();
   70:
   71:     for (int n=1; n<=nsize; ++n)      // n i c h t    n<=x.size() !!
   72:     {
   73:         double tmp =0.0;              // nur innerhalb des n-Loops gue
ltig
   74:         for (int k=1; k<=n; ++k)
   75:         {
   76:             tmp += (2*k+1.0)/(2*n+1);
   77:         }
   78:         x.push_back(tmp/n);                 // Indizierung von 0 bis (
nsize-1)
   79:     }
   80:     return;
   81: }
   82:
   83: int main()
   84: {
   85:     cout << "Hello world!" << endl;
   86:
   87:     int nn;
   88:     cout << "nsize = "; cin >> nn;
   89:
   90:     vector<double> a = folge(nn);
   91:     cout << "letzes Element: " << a.back();
   92:     cout << "   diff: " << a.back()-(nn+2.0)/(2*nn+1.0) << endl;
   93:
   94:     vector<double> b(2);
   95:     folge_d2(nn,b);
   96:     cout << "letzes Element: " << b.back();
   97:     cout << "   diff: " << b.back()-(nn+2.0)/(2*nn+1.0) << endl;
   98:     cout << " Laenge von  " << b.size() << endl;
   99:
  100:     return 0;
  101: }
```