```cpp
// Vorlesung  16.Maerz 2012
// Dynamische C++-Vektoren
// Vektor als Funktionsargument
#include <iostream>
#include <vector>                 // vector
#include <cmath>                  // sqrt()
using namespace std;

//-----------------------------------------------------------------------------
//  Drei Varianten Initialisierung eines C++-Vektors innerhalb einer Funktion
//-----------------------------------------------------------------------------
/** Speicherallokierung und Initialisierung eines C++-Vektors.
*  @param[in]  n Anzahl der Vektorelemente
*  @return     initialisierter Vektor
*  @warning Die Verwendung von at() ist sicher, aber verlangsamt den Code.
*/
vector<float> vec_init_1(const int n)
{
    vector<float> tmp(n);              // allokiert Speicher fuer n float-Zahlen
    for (int k=0; k<tmp.size(); ++k)
    {
        tmp.at(k) = sqrt(1.0+k);
    }
    return tmp;                        // nicht vergessen!
}


/** Speicherallokierung und Initialisierung eines C++-Vektors.
*  @param[in]  n Anzahl der Vektorelemente
*  @param[out] v initialisierter Vektor
*  @warning Daten auf dem ubergebenen Vekor @p v gehen verloren.
*/
void vec_init_2(const int n, vector<float>& v)
{
    v.resize(n);                       // allokiert Speicher fuer n float-Zahlen
    for (int k=0; k<v.size(); ++k)
    {
        v.at(k) = sqrt(1.0+k);
    }
}


/** Speicherallokierung und Initialisierung eines C++-Vektors.
*  Hier wird die Nutzung von push_back demonstriert, dies waere auch in
*  Funktion vec_init_1 so moeglich.
*  @param[in]  n Anzahl der Vektorelemente
*  @param[out] v initialisierter Vektor
*  @warning Daten auf dem ubergebenen Vekor @p v gehen verloren.
*/
void vec_init_3(const int n, vector<float>& v)
{
    v.clear();                         // kein Speicher allokiert
    for (int k=0; k<n; ++k)
    {
        v.push_back(sqrt(1.0+k));      // Element wird an das Ende angehaengt
    }
}

/** Ausgabe eines Vektors mit cout
*  @param[in]  v Vektor mt @p float- Elementen
*/
void print_vek(const vector<float>& v)
{
    for (size_t k=0; k<v.size();++k)
    {
        cout << "  " << v[k];
```

```
 67      }
 68      cout << endl;
 69  }
 70
 71  //---------------------------------------------------------------------------
 72
 73  int main()
 74  {
 75      cout << "Hello world!" << endl;
 76      int n;                                      //   variable Vektorlaenge
 77      cout << "Anzahl der Elemente = "; cin >> n;
 78
 79      {                                           //   Funktion 1 testen
 80          vector<float> v = vec_init_1(n);
 81          cout << v.front() << "   "  << v.at(n/2) <<  "   " << v.back() << endl <<
endl;
 82      }   // hier endet die Gueltigkeit von v
 83
 84      {                                           //   Funktion 2 testen
 85          vector<float> v;                        //   Vektor der Laenge 0
 86          vec_init_2(n,v);
 87          cout << v.front() << "   "  << v.at(n/2) <<  "   " << v.back() << endl <<
endl;
 88      }   // hier endet die Gueltigkeit von v
 89
 90      {                                           //   Funktion 2 testen
 91          vector<float> v(20, -1.2345);           //   20 Vektorelemente, jedes mit
-1.2345 initialisiert
 92          vec_init_3(n,v);
 93          cout << v.front() << "   "  << v.at(n/2) <<  "   " << v.back() << endl <<
endl;
 94
 95          print_vek(v);
 96      }   // hier endet die Gueltigkeit von v
 97
 98
 99      return 0;
100  }
```