Blatt 13

Vincent Kümmerle und Elvis Gnaglo

24. Januar 2026

1 Berechnung der Eulerschen Zahl

```cpp
#include <iostream>
#include <iomanip>
// 1. for-Schleife zur Berechnung der Fakultaet k!
long fakultaet(int k) {
    long result = 1; // long, da mit 64-Bit mehr Platz fuer groessere Zahlen
    for (int i = 1; i <= k; ++i) {
        result *= i;}
    return result;
}
// Funktion zur Annaeherung von e durch n Terme
double approximate_e(int n) {
    double summe = 0.0; // double, da Bruchzahlen mit bis zu 16 Nachkommastellen
    for (int k = 0; k < n; ++k) {
        // 2. Berechnung: 1 / k!
        summe += 1.0 / fakultaet(k);}
    return summe;
}
int main() {
    int iterationen = 12;
    // Berechne e
    double e_approx = approximate_e(iterationen);

    // 4. Ausgabe des Ergebnisses
    std::cout << "Annaeherung von e nach " << iterationen << " Iterationen:" << std::endl;
    std::cout << std::fixed << std::setprecision(8) << e_approx << std::endl;
    return 0;
} // Ausgabe: 2.71828183
```

2 Vektoren I

```cpp
#include <iostream>
#include <vector>

double sum_below_limit(const std::vector<double>& vector, double
    limit) {
    double sum = 0.0;
    for (double element : vector) {
        if (element < limit) {
            sum += element;
        }
    }
    return   sum;
}

int main()
{
    std::vector<double> v = { 1.0, 2.0, 3.0, 4.0, 5.0, 42.0, 100.00,
       300.00 };
    std::cout << sum_below_limit(v, 4.0) << "\n";
    return 0;
}
```

3 Vektoren II

```cpp
#include <iostream>
#include <vector>

double scalar_product(const std::vector<double>& vector1,
    const std::vector<double>& vector2) {
    double sum = 0.0;

    for (size_t i = 0; i < vector1.size(); ++i) {
        sum += vector1[i] * vector2[i];
    }

    return sum;
}

int main()
{
    std::vector<double> v1 = { 1.0, 2.0, 3.0, 4.0 };
    std::vector<double> v2 = { 1.0, -1.0, -1.0, 1.0 };
```

```
19      std::cout << scalar_product(v1, v2) << "\n";
20      return 0;
21  }
```