Blatt 13

Aufgabe 1

Für `fakultaet` wird `long` verwendet da `int` kleiner ist und long besser ist. Da e eine Kommazahl ist verwenden wir für `approximate_e` den Datentyp `double`.

```
#include <iostream>
#include <iomanip>
#include <print>
// Funktion zur Berechnung der Fakultaet k!
long fakultaet(int k)
{
    long result = 1;
    for (long 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;
    for (int k = 0; k < n; ++k)
    {
        summe += 1. / fakultaet(k);
    }
    return summe;
}
int main()
{
    int iterationen;
    std::print("Gib hier eine Zahl ein: ");
    std::cin >> iterationen;
    // Berechne e
    double e_approx = approximate_e(iterationen);
    // Ausgabe des Ergebnisses
    std::cout << "Annaeherung von e nach " << iterationen << " Iterationen:" <<
std::endl;
    std::println("Die Zahl ist: {:.8f}", e_approx);
    // Tipp: Nutze std::setprecision aus <iomanip>
    return 0;
}
```

Aufgabe 2

```cpp
#include <iostream>
#include <vector>
auto sum_below_limit(const std::vector<double> &vector, double limit) -> double
{
    auto result = 0.;
    for (auto i : vector)
    {
        if (i < limit)
        {
            result += i;
        }
    }
    return result;
}
int main()
{
    std::vector<double> v = {1.0, 2.0, 3.0, 4.0, 5.0, 42.0};
    std::cout << sum_below_limit(v, 4.0) << "\n";
    return 0;
}
```

Aufgabe 3

```cpp
#include <iostream>
#include <vector>
auto scalar_product(const std::vector<double> &vector1,
                    const std::vector<double> &vector2) -> double
{
    if (vector1.size() != vector2.size())
        throw std::runtime_error("Die Vectoren sind nicht gleich lang!");
    if (vector1.empty())
        throw std::runtime_error("Die Vectoren sind leer!");

    auto result = 0.;
    for (size_t i = 0; i < vector1.size(); ++i)
        result += vector1[i] * vector2[i];
    return result;
}
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};
    std::cout << scalar_product(v1, v2) << "\n";
    return 0;
}
```