Single Threaded Summation: Ranged Based for Loops

Explaining the solution for calculating the sum of a vector problem using ranged-based for loops in C++.

The obvious strategy is it to add the numbers in a range-based for loop like we did in the code below.

The summation takes place in line 27:

```cpp
// calculateWithLoop.cpp

#include <chrono>
#include <iostream>
#include <random>
#include <vector>

constexpr long long size = 100000000;

int main(){

  std::cout << std::endl;

  std::vector<int> randValues;
  randValues.reserve(size);

  // random values
  std::random_device seed;
  std::mt19937 engine(seed());
  std::uniform_int_distribution<> uniformDist(1, 10);
  for (long long i = 0 ; i < size ; ++i)
       randValues.push_back(uniformDist(engine));

  const auto sta = std::chrono::steady_clock::now();

  unsigned long long sum = {};
  for (auto n: randValues) sum += n;

  const std::chrono::duration<double> dur =
        std::chrono::steady_clock::now() - sta;

  std::cout << "Time for mySumition " << dur.count()
            << " seconds" << std::endl;
  std::cout << "Result: " << sum << std::endl;

  std::cout << std::endl;

}
```

You should not use loops explicitly. Most of the time you can use an algorithm from the Standard Template Library.