Übungen zu Programmieren 2

Ostbayerische Technische Hochschule Regensburg

C++ Übung 6

Übungen zum Umgang mit Schleifen in C++

- Die aus C bekannte `for`-Schleife funktioniert auch in C++
  - Anwendungsbeispiel:

Schreiben Sie ein Programm zur Ausgabe des folgenden Musters

```
1******
12*****
123****
1234***
12345**
123456*

1234567
```

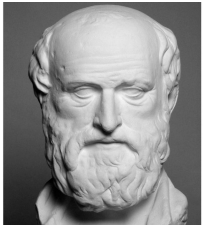
Sieb des Eratosthenes

- Alle zusammengesetzten Zahlen werden gestrichen

  gemäß: de.wikipedia.org/wiki/Sieb_des_Eratosthenes

Implementierung in C++

```
#include <cmath>
#include <vector>
#include <iostream>
std::vector<int> sieve(int n)  {  int sqrtN = (int)sqrt(n);
    std::vector<int> primes = std::vector<int>( );
    std::vector<bool> candidates(n + 1, true);
    for (int i = 2;  i <= sqrtN; ++i) {
        for (int j = i; i * j <= n;  ++j)
            candidates[i * j] = false;  }
    for (int i = 2; i <= n; ++i)
        if (candidates[i])  primes.push_back(i);
    return primes; }
```



- Ergänzen Sie die `sieve( )`-Funktion von der vorigen Folie um ein `main( )`-Programm zur Ausgabe der ersten 1000 Primzahlen
- Verwenden Sie dabei ein *range-based* `for`

- In dem Programm sind mehrere Optimierungen möglich:
  - für $n < 2$ kann ein leerer Vektor geliefert werden
  - in der `i`-ten Iteration sind die Zahlen `< i * i` bereits markiert
  - anstelle $i * j$ für jede Iteration zu berechnen, kann (in der inneren Schleife) `i` als Schrittweite gesetzt werden
  - gerade Zahlen müssen (in der dritten Schleife) *nicht* berücksichtigt werden
- Verbessern Sie den Code entsprechend den Vorgaben

Laufzeitmessung in C++

- Auf
  `http://en.cppreference.com/w/cpp/chrono/time_point`
  finden Sie eine Anleitung zur Messung der Laufzeit Ihres Codes

- Überprüfen Sie die Wirksamkeit Ihrer Optimierungen von der vorigen Folie durch eine Laufzeitmessung für die Berechnung der ersten 10000 Primzahlen