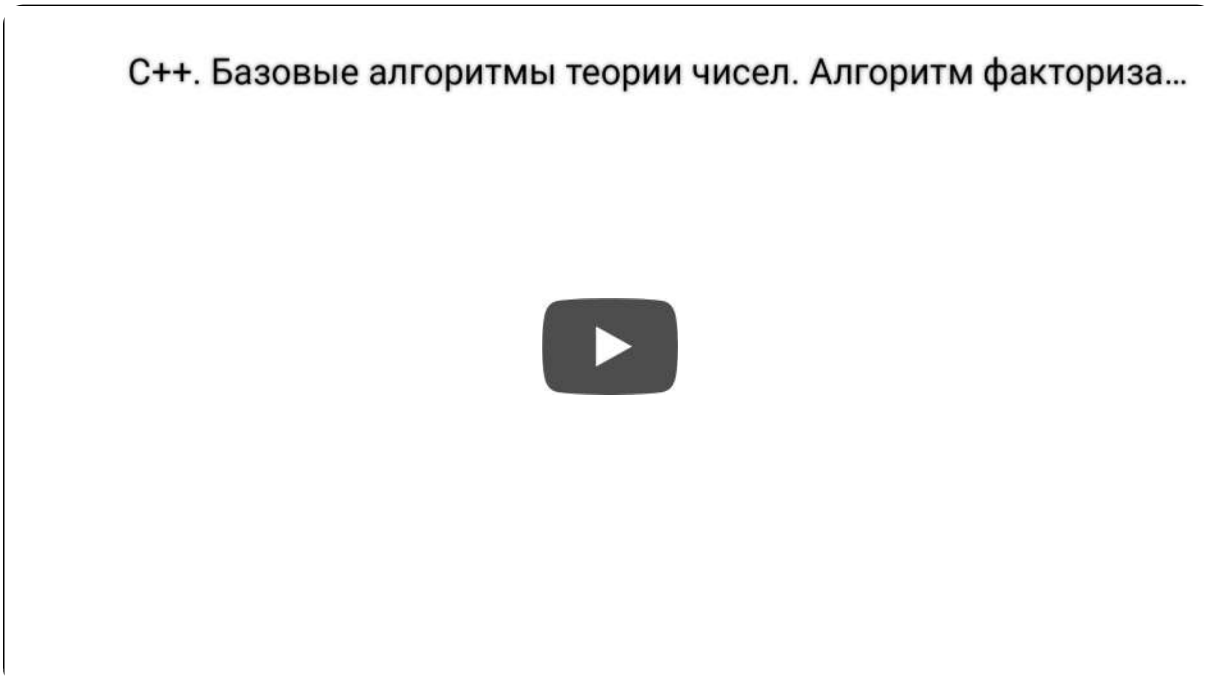
Алгоритм факторизации числа



Научимся искать разложение числа n на простые множители, то есть представление числа n в виде $n = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_s^{\alpha_s}$, где $p_1 < p_2 < \cdots < p_s$ – простые числа, а $\alpha_1, \alpha_2, \ldots, \alpha_s$ – натуральные числа. По основной теореме арифметики такое разложение всегда существует и единственно.

Будем перебирать в переменной d все числа от 2 до $\sqrt{n}$ и делить n на d до тех пор, пока оно делится. Заметим, что при таком подходе n всегда будет делиться только на простые числа, так как если в переменной d записано составное число, то это значит, что до этого мы уже разделили n на все простые числа, которые являются делителями d. Таким образом, мы нашли все простые делители числа n, не превосходящие $\sqrt{n}$. Возможно, у нас остался один простой делитель, больший $\sqrt{n}$ (если бы n содержало два таких делителя, то их произведение было бы больше n). Такой делитель хранится в переменной n после завершения перебора всех значений d.

Вычислительная сложность данного алгоритма – $O(\sqrt{n})$.

Также бывает полезным асимптотически оценить количество различных простых делителей $n - p_1, p_2 \ldots p_s$. Заметим, что деление на каждый делитель уменьшит число n минимум в два раза, а значит количество s различных простых делителей есть $O(\log n)$. Если мы будем учитывать каждый простой делитель столько раз, сколько он встречается в разложении, то есть рассмотрим сумму $\alpha_1 + \alpha_2 + \ldots + \alpha_s$, то её также можно оценить как $O(\log n)$.

Реализация

```
vector<int> factorization(int n) {
   vector<int> p;
   for (int d = 2; d * d <= n; ++d) {
       while (n % d == 0){
           p.push_back(d);
           n /= d;
       }
   }
   if (n > 1)
       p.push_back(n);
   return p;
}
```