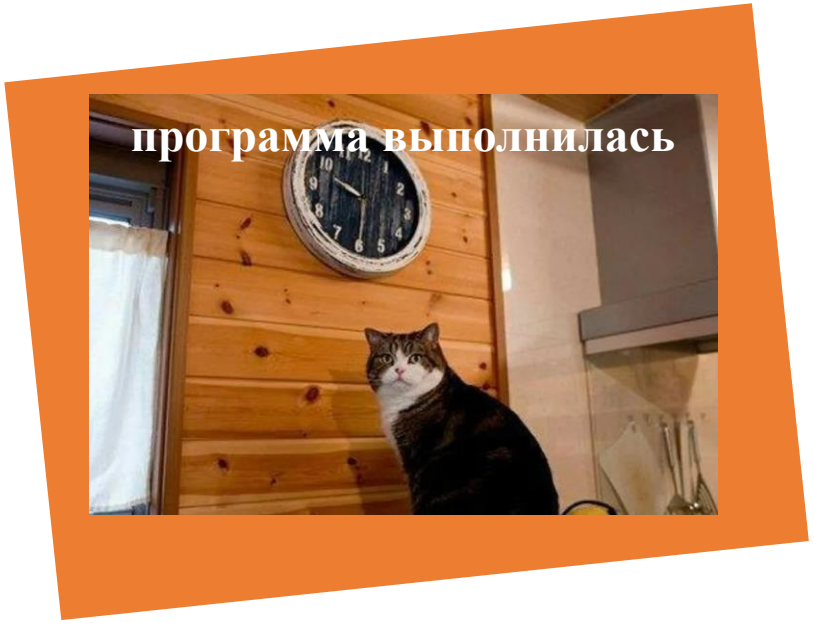
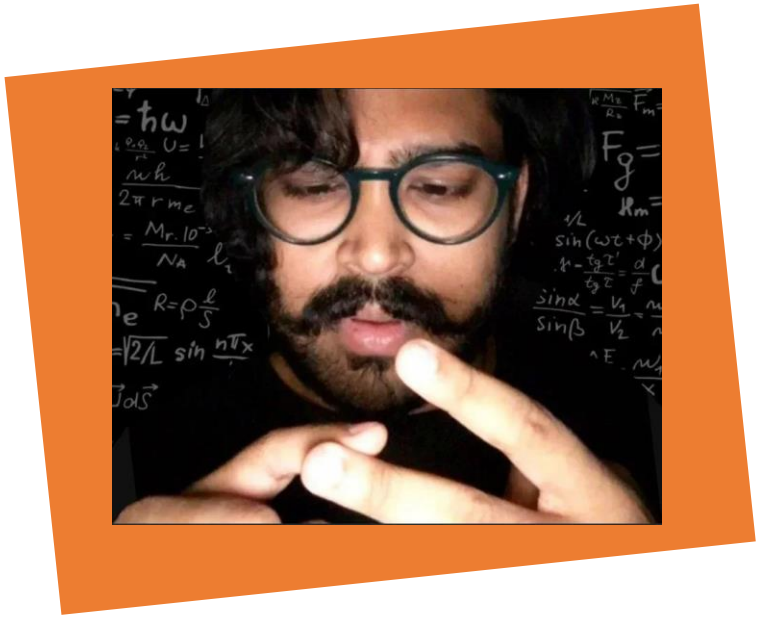
Введение в теорию сложности алгоритмов

10 класс

Как определить какой алгоритм эффективнее?

? Измерять время выполнения?



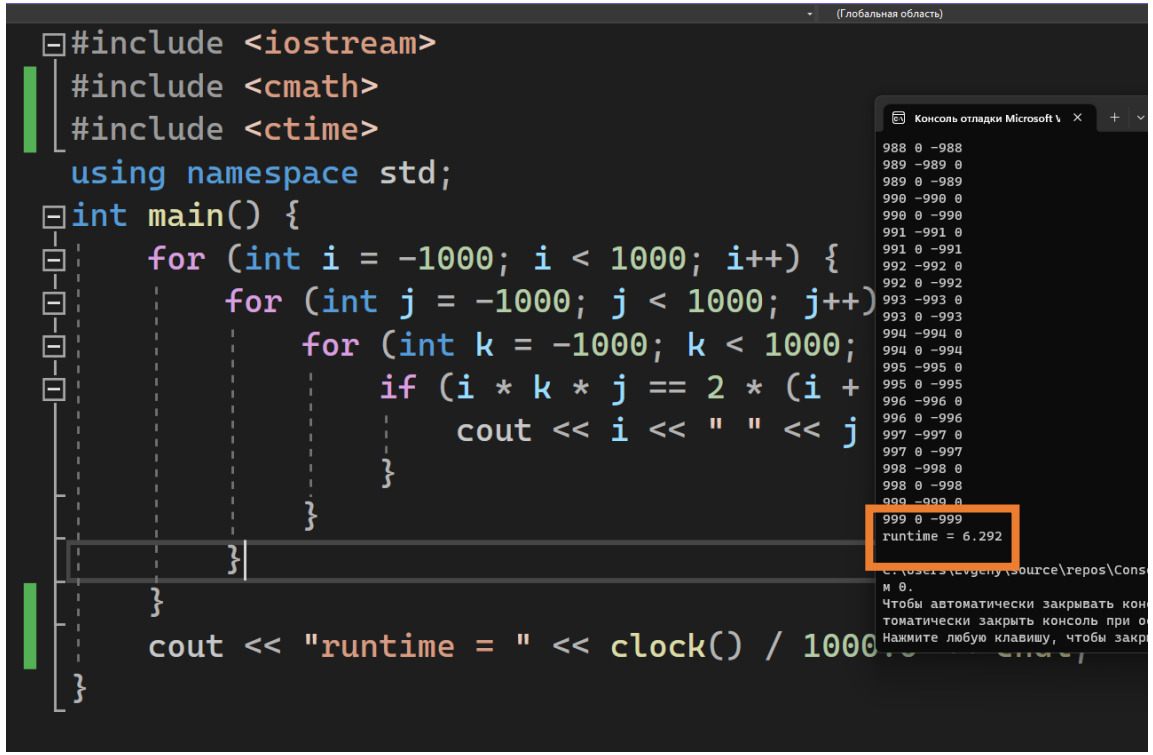


? Считать количество операций?

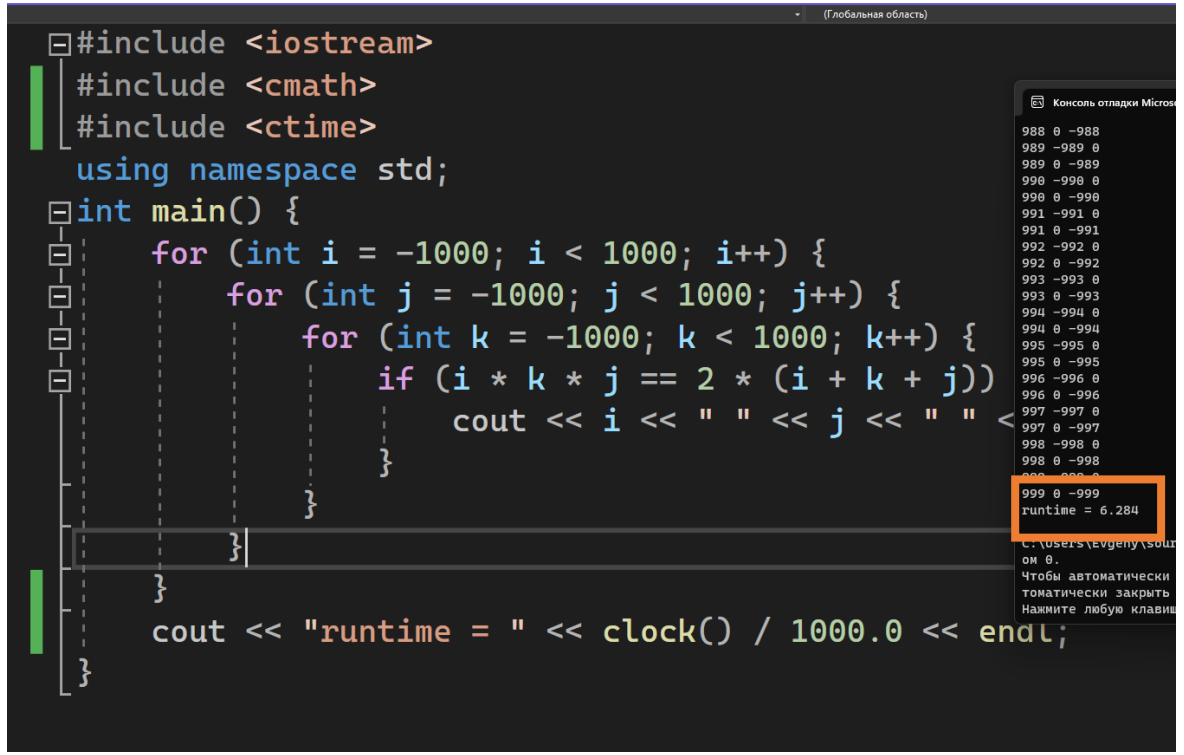
Что не так с подсчетом времени выполнения?

Один и тот же алгоритм может выполняться за разное время даже на одном компьютере от запуска к запуску, не говоря о разных компьютерах с разными процессорами.

6.292 с



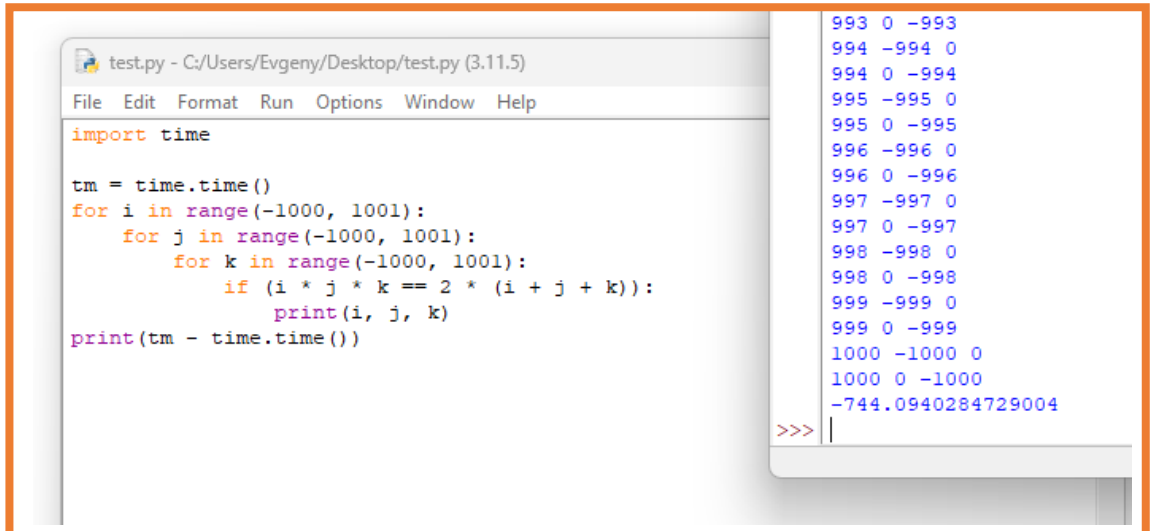
6.284 с



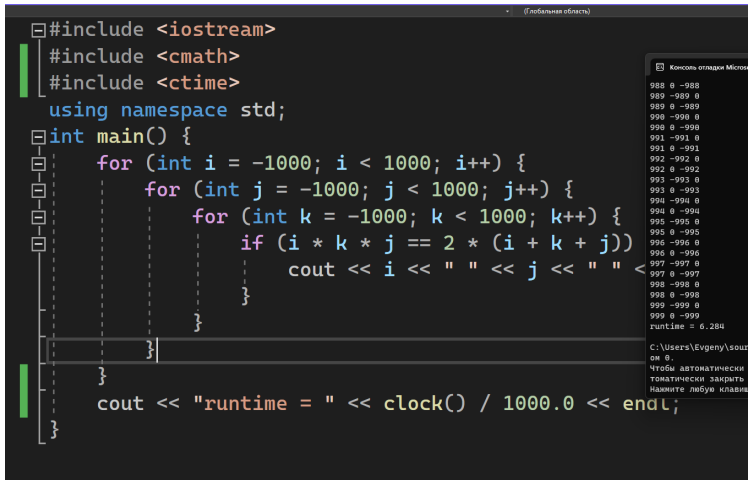
Что не так с подсчетом времени выполнения?

Один и тот же алгоритм в разных языках программирования выполняется за разное время, иногда за СУЩЕСТВЕННО разное время.

Python: ~ 744 с ~ 12.5 мин



C++: ~ 6.3 с



Параметры сборки, на которой проводились тесты:

Процессор: 12th Gen Intel(R) Core(TM) i7-12700KF (20 CPUs), ~3.6GHz
Память: 32768MB RAM

Подсчет количества операций

Договоримся, что считать будем следующие операции в программе, остальное считаем выполняющимся мгновенно:

- Присваивание значения переменной
- Сравнение двух значений
- Инкрементация значения
- Основные арифметические операции: сложение, умножение и т.д.

Посчитаем количество операций для конкретного алгоритма

```cpp
int main() {
    int n, a;
    int m = 0;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> a;
        if (a > m) {
            m = a;
        }
    }
}
```

Присваивание значения переменной

Сравнение двух значений

Инкрементация значения

Основные арифметические операции: сложение, умножение и т.д.

Наилучший и наихудший случаи

```cpp
int main() {
    int n, a;
    int m = 0;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> a;
        if (a > m) {
            m = a;
        }
    }
}
```

Подумайте, какой из данных случаев является наихудшим, а какой наилучшим?

4	4
1	4
2	3
3	2
4	1

Задание 1. Подсчитайте количество операций для данных алгоритмов. Определите наилучший и наихудший случай.

а)

```
int main() {
    for (int i = 0; i < 100; i++) {
        cout << i;
    }
}
```

б)

```
    int N;
    cin >> N;
    int s = 0;
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            s = s + i * j;
        }
    }
```

Задание 2. Определить асимптотическую сложность алгоритма с заданным в виде функции количеством операций.

а) $f(n) = 5n + 12$

б) $f(n) = 109$

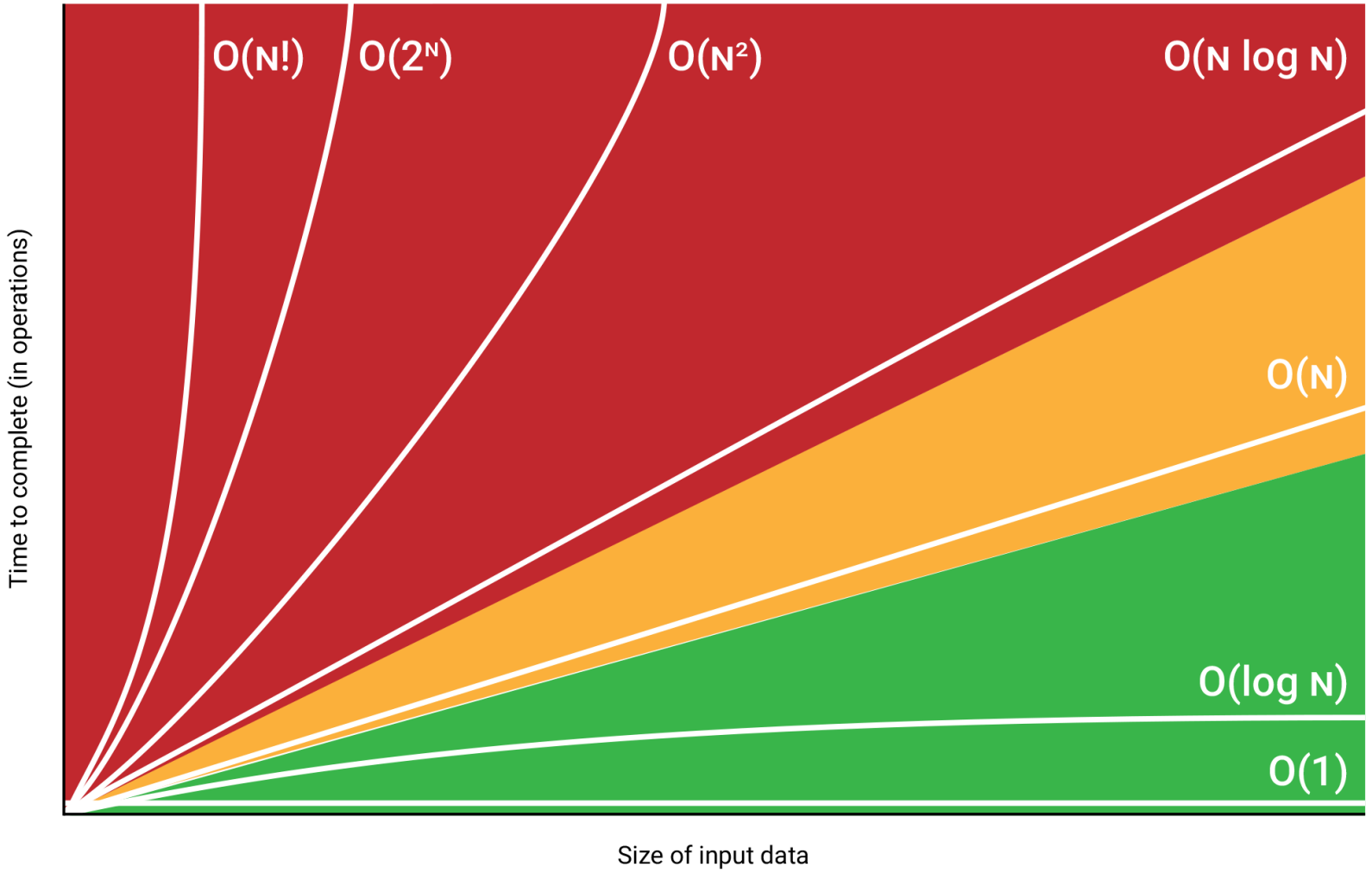
в) $f(n) = n^2 + 3n + 112$

г) $f(n) = n^3 + 1999n + 1337$

д) $f(n) = n + sqrt(n)$

Big O нотация

Факториальная	O(N!)	
Экспоненциальная	$O(2^N)$	
Квадратичная	$O(N^2)$	
Линейно-логарифмическая	O(N · log N)	
Линейная	O(N)	
Сублинейная	O(√N)	
Логарифмическая	O(log N)	
Константная	O(1)	



Примеры алгоритмов разной сложности

Константная:

```
int main() {
    for (int i = 0; i < 100; i++) {
        cout << i;
    }
}
```

Сублинейная:

```
int N;
cin >> N;
int s = 0;
for (int i = 0; i * i < N; i++) {
    cout << i;
}
```

Примеры алгоритмов разной сложности

Линейная:

```cpp
int N;
cin >> N;
int s = 0;
for (int i = 0; i < N; i++) {
    cout << i;
}
```

Квадратичная:

```cpp
int N;
cin >> N;
int s = 0;
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        s = s + i * j;
    }
}
```

Повторим изученное

#ОЦЕНКА СЛОЖНОСТИ АЛГОРИТМОВ

Домашнее задание

1. Выучить виды сложностей алгоритмов в нотации Big O

2. Знать разобранные на уроке примеры под различные виды сложности алгоритмов