Задача 1. Распараллеливание известного алгоритма с анализом свойств полученного решения.

Исследование провёл студент группы 22207 Гордеев Никита

Дата выполнения работы 11.12.2022 (Вариант 3)

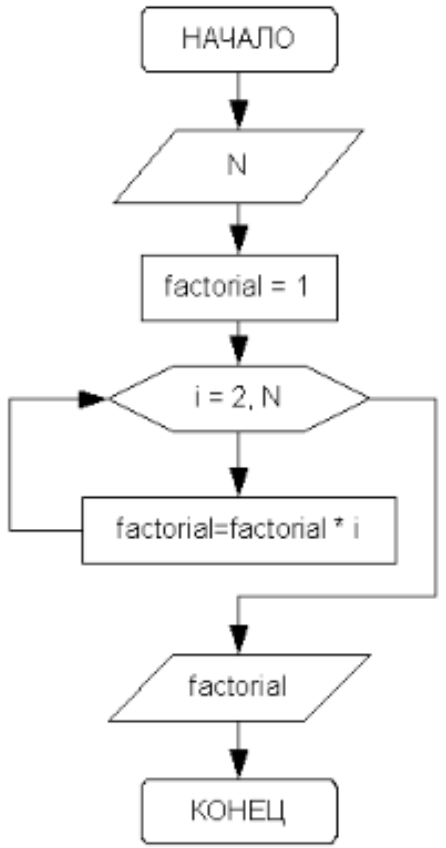
Постановка задач

1. **Взять известный алгоритм (последовательная версия).**
2. **Разработать его параллельную версию.**
3. **Выполнить анализ параллельной версии (плюсы и минусы в сравнении с последовательной).**

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n$$

Исходный алгоритм (факториал)

```
var k, n, f: integer;
// n - до какого числа вычислять факториал
input (n);
f := 1;
// вычисление факториала
for k := 1 to n do
f := f * k;
output (f);
end.
```



Анализ исходного алгоритма

Проблема:

- **Последовательное выполнение программы занимает много времени.**

Идея:

- **Вычисление чётных и нечётных цифр может быть реализовано независимо, для ускорения программы.**

Параллельная версия исходного алгоритма

```
begin
var m, o_p, e_p, n, f, i, j : integer;
input (m); n := m/2; // n - верхняя граница
co;
    // перемножаем чётные числа
    e_p := 1; for i := 1 to n do e_p := e_p * (2 * i);
    || |
    // перемножаем нечётные числа
    o_p: = 1; for j := 1 to (n - 1) do o_p: = o_p * (2 * j + 1);
    if odd (m) then o_p: = o_p * m; // если m было нечётным
oc;
f := e_p * o_p; output (f) // ответ
end.
```

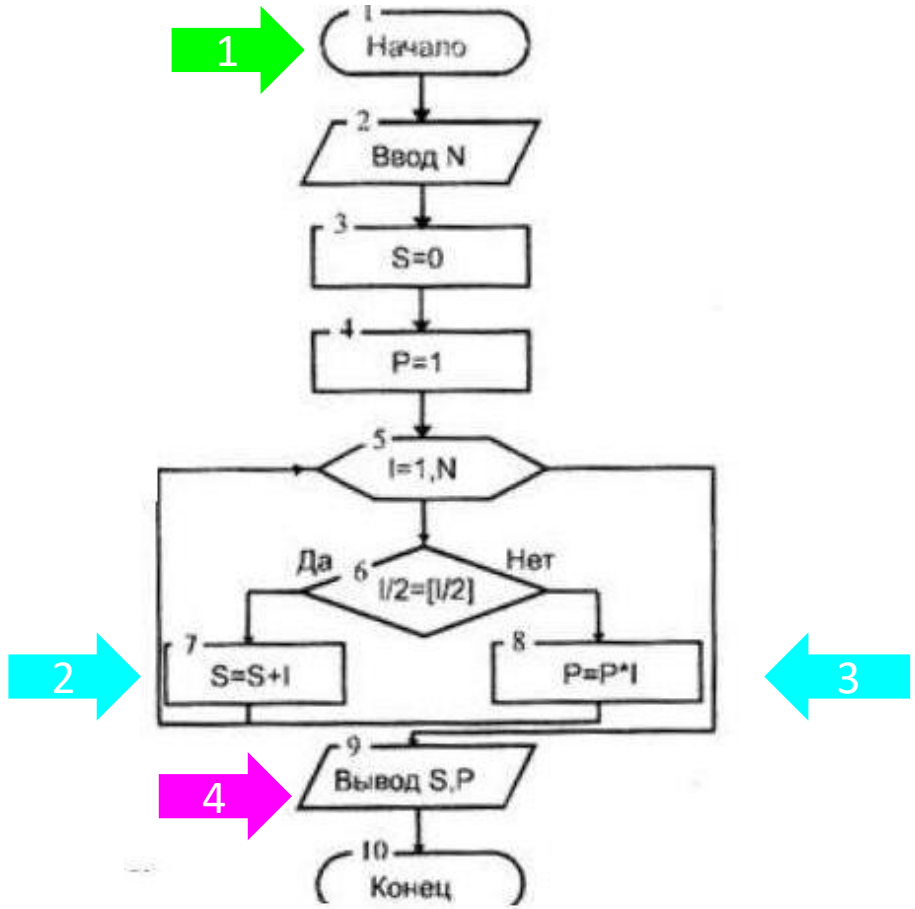
Пояснения по параллельной версии

Алгоритм:

Вычисление факториала N, как произведения двух чисел — произведения всех чётных чисел и произведения всех нечётных чисел

Программа:

- begin инициирует первый поток исполнения
- Команда co задаёт разделение данного потока на два (второй и третий)
  - even — произведение всех чётных числе из диапазона [1..N];
  - odd — произведение всех нечётных чисел из диапазона [1..N];
- Команда oc снова сливает два потока в один (четвёртый)



Сравнение программ

Один поток

Использование ядер:

- **Запуск команды осуществляется в порядке размещения её команд в памяти и их последовательное исполнение**

Время выполнения программы:

- **1 поток — N**

В несколько потоков

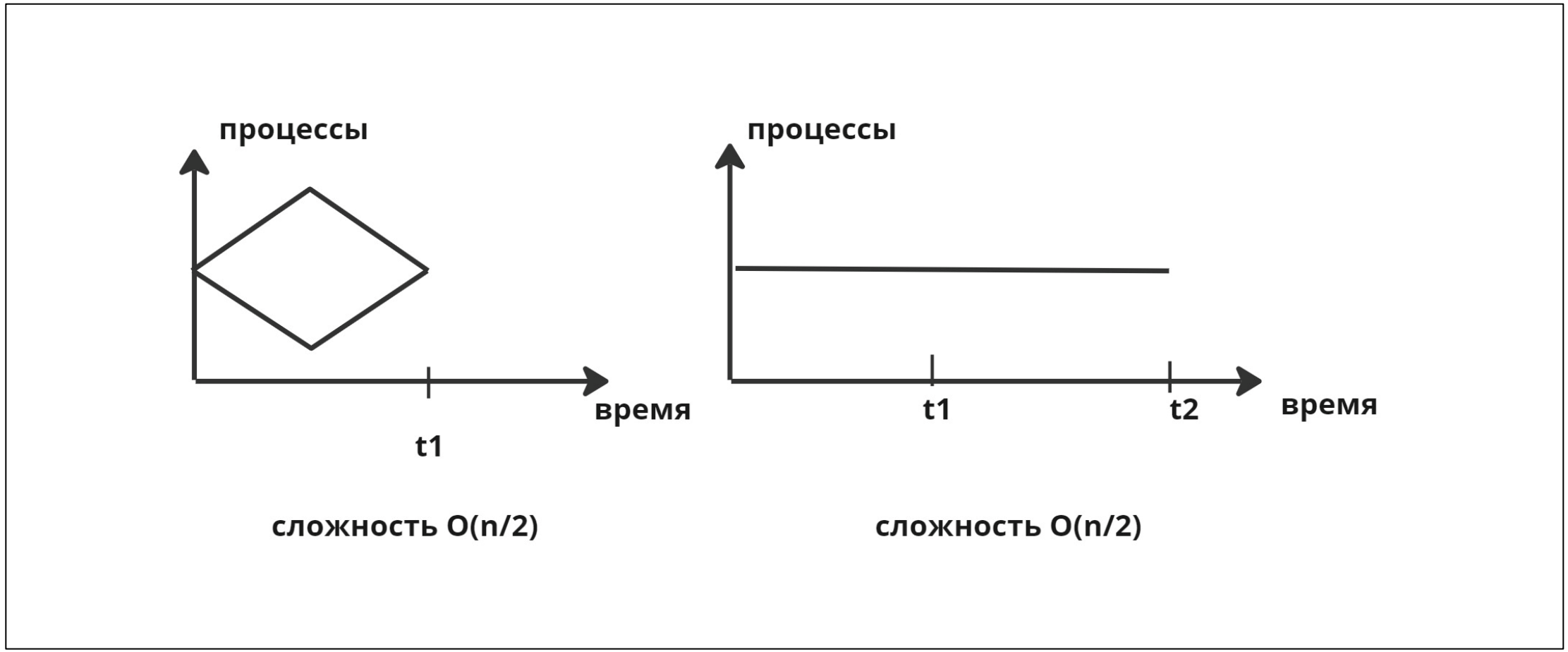
Использование ядер:

- **Первое ядро — планировщик**
- **Второе ядро — для 1го потока, затем для 2го, а в конце для 4го**
- **Третье ядро — для 3го**

Время выполнения программы:

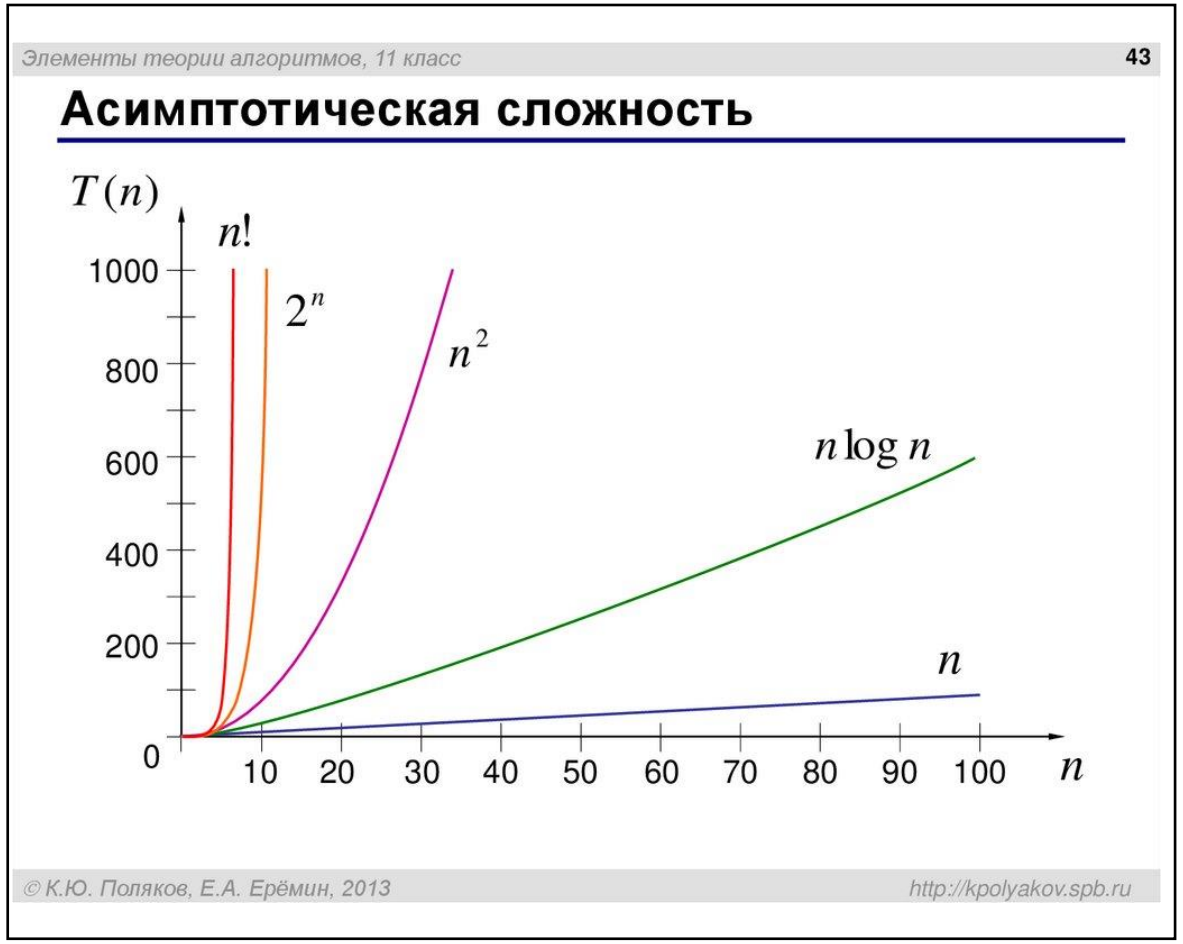
- **1-й поток 3 тика**
- **4-й поток — 2 тика**
- **2й и 3й — (2+4*(N/2))**

Схема работы



Анализ эффективности

- **Описанный алгоритм при реализации на трехъядерном процессоре способен дать ускорение примерно в 2 раза**
- **Сложность последовательной программы: O(n)**
- **Сложность параллельной программы: O(n/2)**



Анализ корректности

- **Из-за параллельности коллизий не возникает, т.к. каждый параллельный процесс записывает результат в свою переменную, т.е. процессы не пересекаются по переменным записи.**

Вывод

- **Параллельная программа работает корректно.**
- **Программа работает эффективнее последовательной версии, если**
  - **Вычисляемый факториал — большое число;**
  - **Данные вычисляются с помощью сети множества компьютеров.**
- **В ином случае эффективность примерно равна последовательной версии.**

Материалы

1. **Параллельное программирование среди других парадигм программирования // CYBERLENINKA URL: https://cyberleninka.ru/article/n/parallelnoe-programmirovanie-sredi-drugih-paradigm-programmirovaniya (дата обращения: 26.09.2022).**
2. **Оценка сложности алгоритмов // Хабр URL: https://habr.com/ru/post/104219 (дата обращения: 23.11.2022).**

Изменения

1. **Версия 2**
    1. **Изменено оформление титульного листа**
    2. **Добавлен слайд идея параллельности**
    3. **Сделал шрифт жирным, для лучшей читаемости на проекторе**
2. **Версия 3**
    1. **Фрагменты кода в виде снимков заменены текстом**
    2. **Добавлены комментарии в код**
    3. **Подробнее расписаны идеи параллельной и последовательной версии**