1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —

**Институт кибербезопасности и защиты информации**

**ЛАБОРАТОРНАЯ РАБОТА № 3**

1. «МЕХАНИЗМЫ МНОГОПОТОЧНОСТИ»
2. по дисциплине «Операционные системы»
3. Выполнил
4. студент гр. 4851003/90001 Горецкий И.А.

1. Руководитель
2. ассистент Крундышев В.М.
4. Санкт-Петербург
5. 2020

# цель работы

Изучение принципов разработки многопоточных программ, изучить программный интерфейс операционных систем для организации многопоточности, получить навыки организации взаимодействия потоков в многопоточных программах. ход работы

## Вычисления

Задача: посчитать количество представление числа в виде суммы натуральных чисел.

Решение: Данная задача решалась рекурсивно, так же как задача о сборе рюкзака. У каждого потока свой id, и каждый поток начинает разложение числа с номера своего id, до него же, те поток с id = 2 будет действовать следующим образом при разложении числа 6:

1)Получит первое разложение из 2 и 5 едениц: 2+1+1+1+1+1;

2) Заменит вторую цифру цифрой 2: 2+2+1+1

3) Заменит третью цифру цифрой 2: 2+2+2

4)Закончит свою работу тк все цифры достигли своего максимум = id = 2

В тоже время поток с id = 3, начнет разложение с цифры 3 и тремя единицами, на рисунке 1 представлена блок схема работы рекурсии

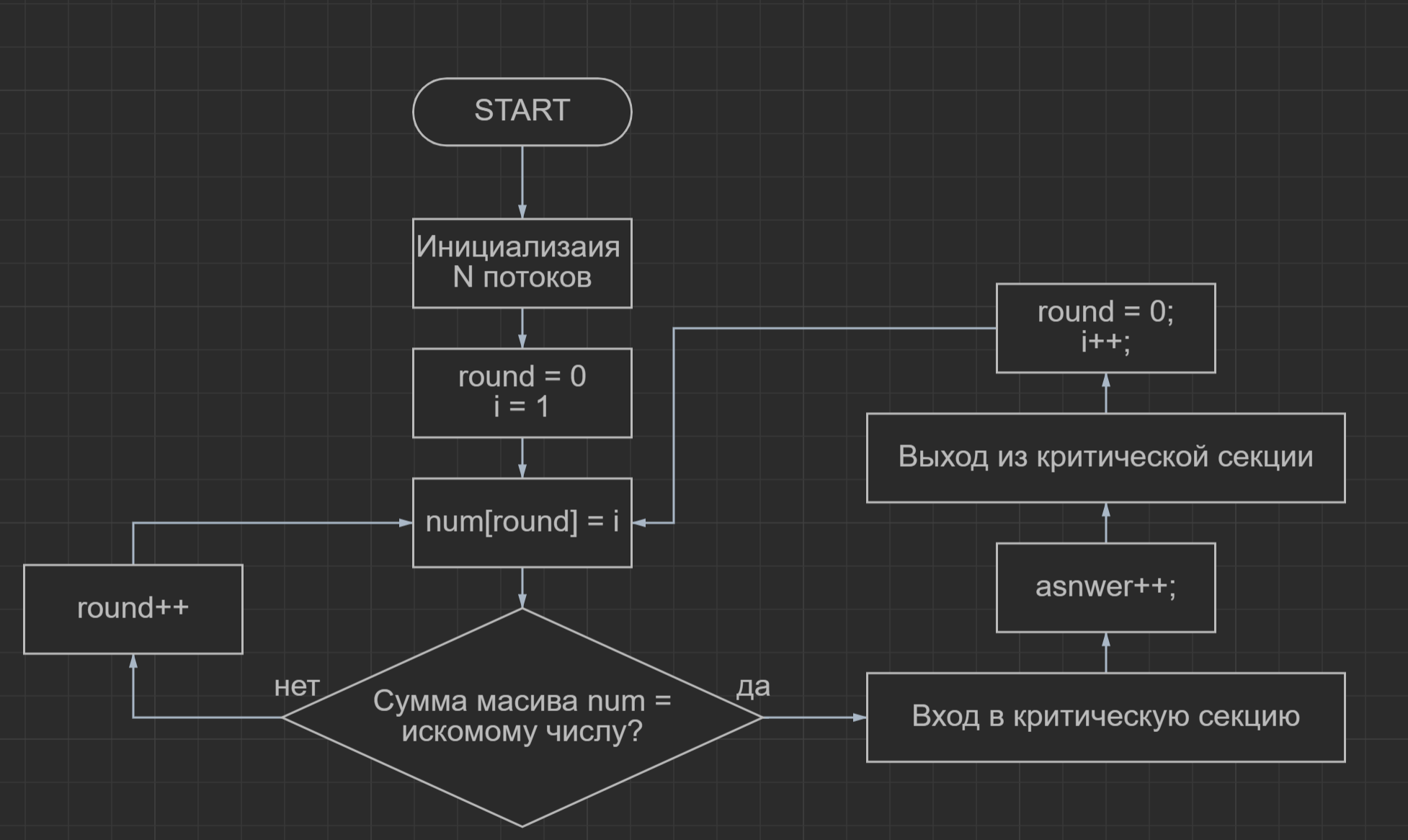


Рисунок 1 – Блок схема работы рекурсии

## QuickSort (MergeSort аналогично)

Задача: реализовать мульти поточную быструю сортировку

Решение: Данное задание решалось рекурсивно, а именно до начала работы алгоритма создавалась “задача”, состоящая из сходного массива. После этого потоки пытались завладеть мютексом, если у них это получалось они брали задачу из очереди и решали ее. Решением задачи является 2 подмассива первый состоял из меньших цифр, а второй из больших. После этого поток опять пытался завладеть мютексом, после чего добавлял эти две “задачи” в очередь. На рисунке 2 показан алгоритм работы.

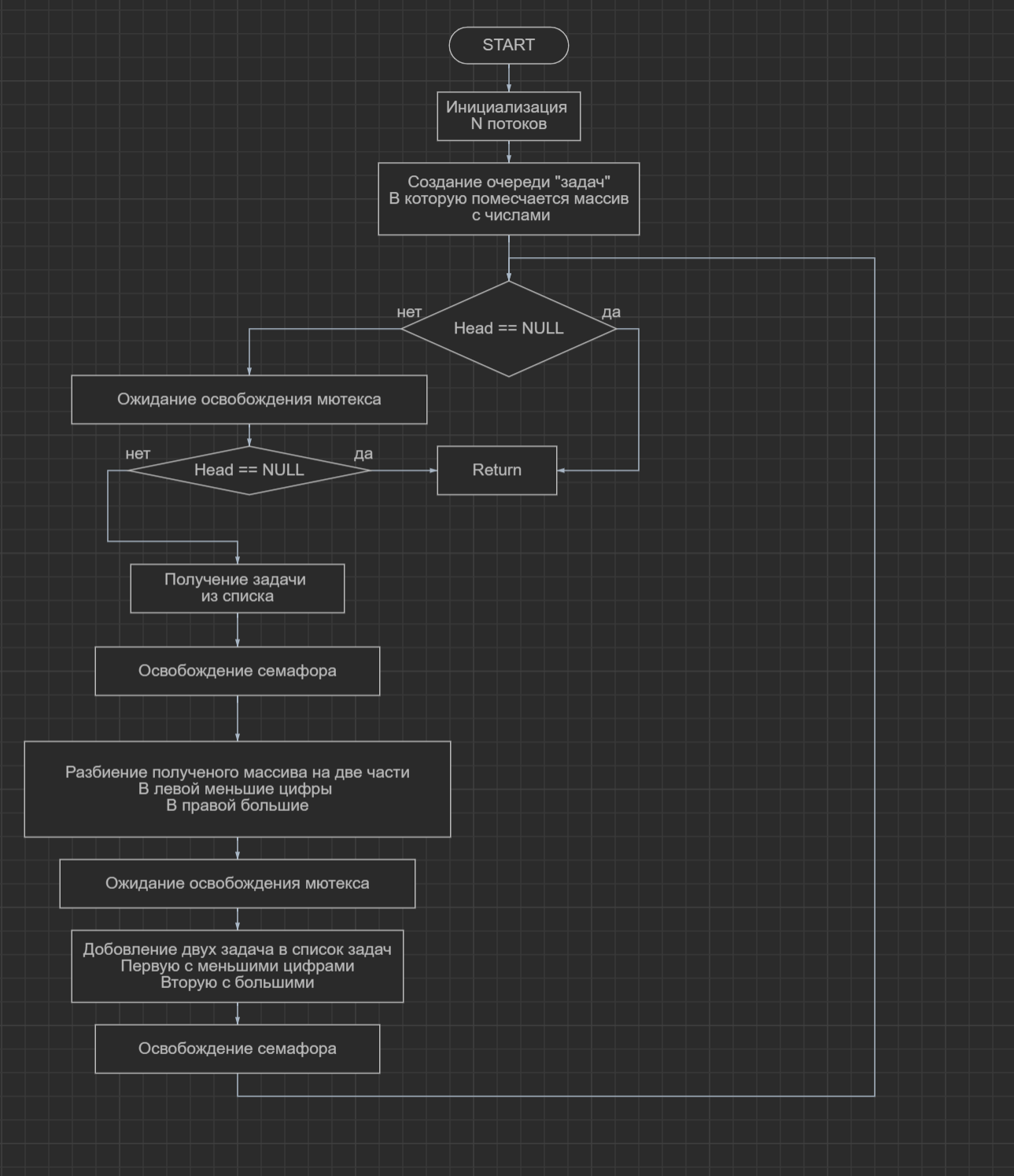


Рисунок 2 – Алгоритм работы быстрой сортировки

|  |  |  |
| --- | --- | --- |
| N | T | Время исполн-я (мс) |
| 10000 | 1 | 29 |
| 10000 | 4 | 61 |
| 1000000 | 1 | 1278 |
| 1000000 | 2 | 998 |
| 1000000 | 4 | 879 |
| 1000000 | 8 | 851 |
| 1000000 | 16 | 832 |
| 10000000 | 1 | 2564 |
| 10000000 | 2 | 1656 |
| 50000000 | 1 | 3821 |
| 50000000 | 8 | 1812 |
| 50000000 | 32 | 1545 |

## Философы

Задача: решить задачу обедающих философов

Решение: так как философы должны трапезничать парами, то за них должны отвечать два потока, когда остальные будут ожидать. Философы обедают в таком порядке: 13;24;35;14;25;31;24 и так далее. Так как всего 5 потоков то каждый поток отвечает за своего философа, дожидается своей очереди и вместе со вторым философом они вместе опускают семафор. На рисунке 3 показан алгоритм работы.

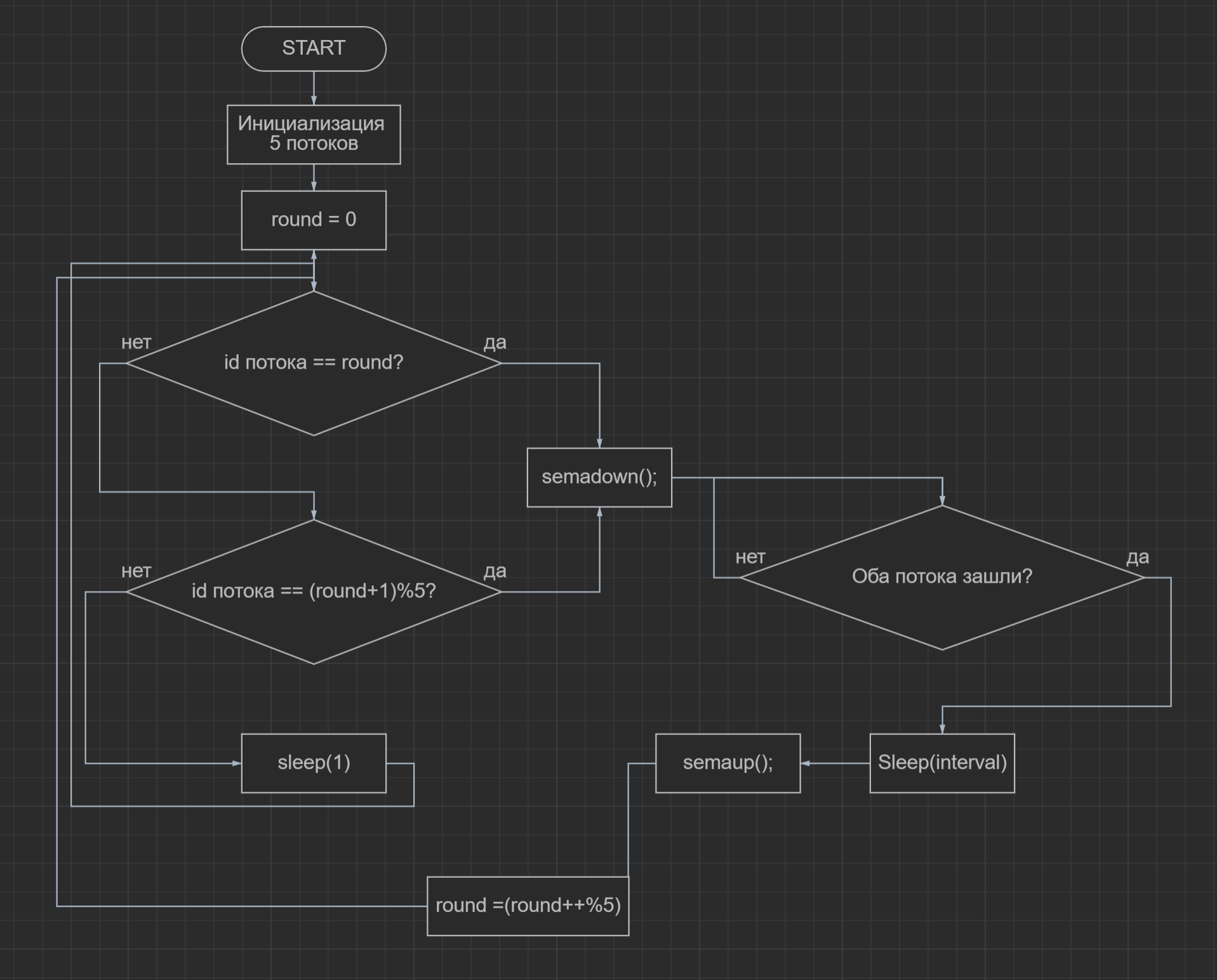


Рисунок 3 – Алгоритм решения задачи философов

## Шахматы

Задача: решить задачу о расстановке N ферзей

Решение: данная задача решается рекурсивным перебором всей доски (рисунок 4)

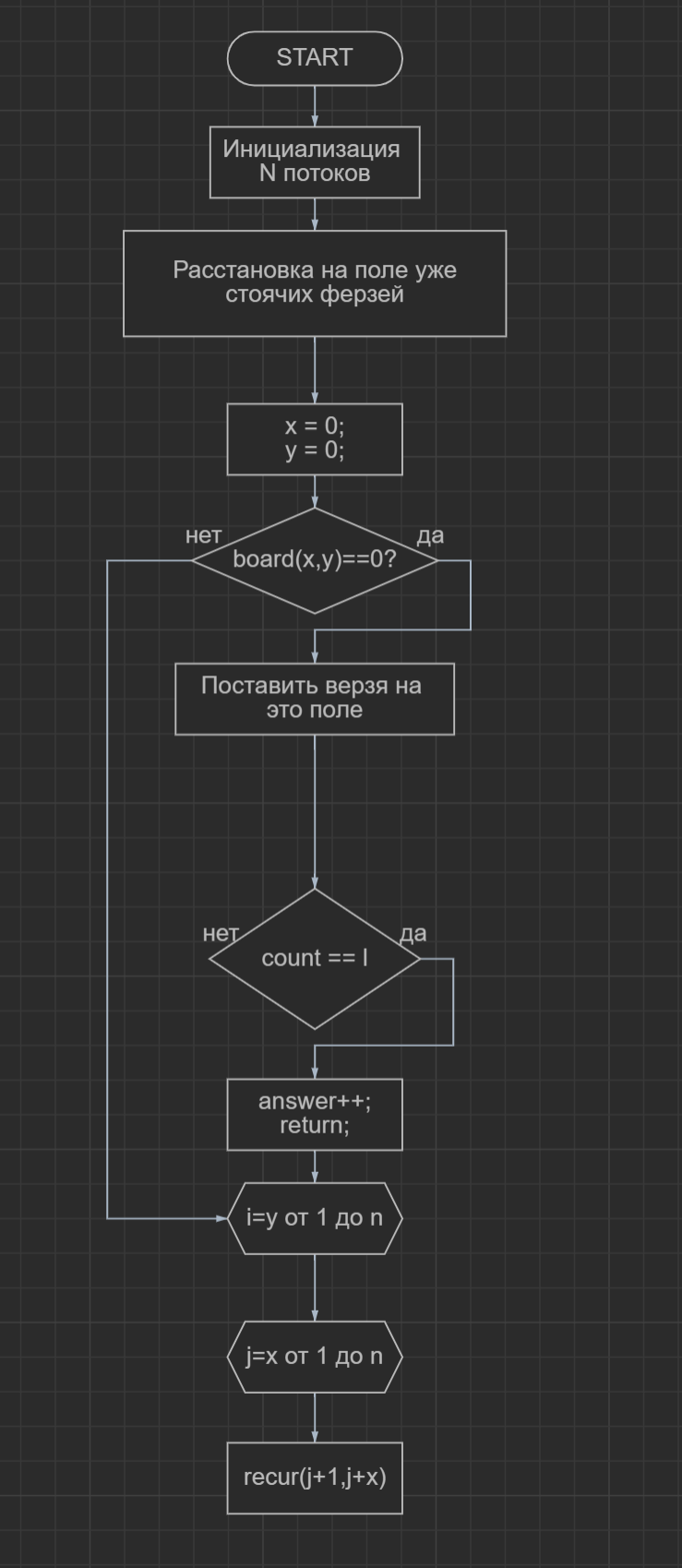


Рисунок 4 – Алгоритм решение задачи о ферзях

# Вывод

В ходе лабораторной работы, были изучены примитивы синхронизаций и многопоточность. В тестах с сортировками можно заметить что при малой длине массив (N<106), многопоточность вместо ускорения программы наоборот ее замедлила, но при больших N, можно наблюдать очень хорошую прибавку к скорости сортировки массива. Это связано с тем, что работа процессора по переключению между потоками, не окупается при малом количестве вычислений. Так же при большом числе потоков (T>4), наблюдалось ухудшение ускорения, вместо двухкратного получалось ~1,2, это связано с тем что PC на котором тестировались данные программы обладал 4-ех ядерным процессом, и каждое ядро за такт может обработать только один поток, поэтому при больших T, может даже начаться замедление работы программы.