МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет автоматики и вычислительной техники

Кафедра вычислительной техники



Курсовая работа

По дисциплине: «Параллельное программирование»

|  |  |
| --- | --- |
| Факультет: АВТ  Группа: АВТ-118  Студент: Маркелов Н.Э. | Преподаватель: Зеленчук Н.А. |

Новосибирск

2023 г.

**Цель работы:**

Целью данной работы является изучение четырех технологий для параллельного вычисления: OpenMP, NVIDIA Cuda, OpenCL и MPI. Стоит выделить постановки задач:

1. Создать последовательный алгоритм, который будет выполняться на основном потоке;

2. Создать параллельный алгоритм, который будет выполнять параллельное вычисление;

3. Получить один и тот же результат от двух типов вычислений, замеряя их время и сохраняя в отдельный файл (при огромном выводе).

**Описание алгоритмов:**

1. **Параллельное программирование для систем с общей памятью с использованием технологии OpenMP.**

Задание: Решение системы линейных алгебраических уравнений заданного порядка с заданной точностью.

Последовательный алгоритм:

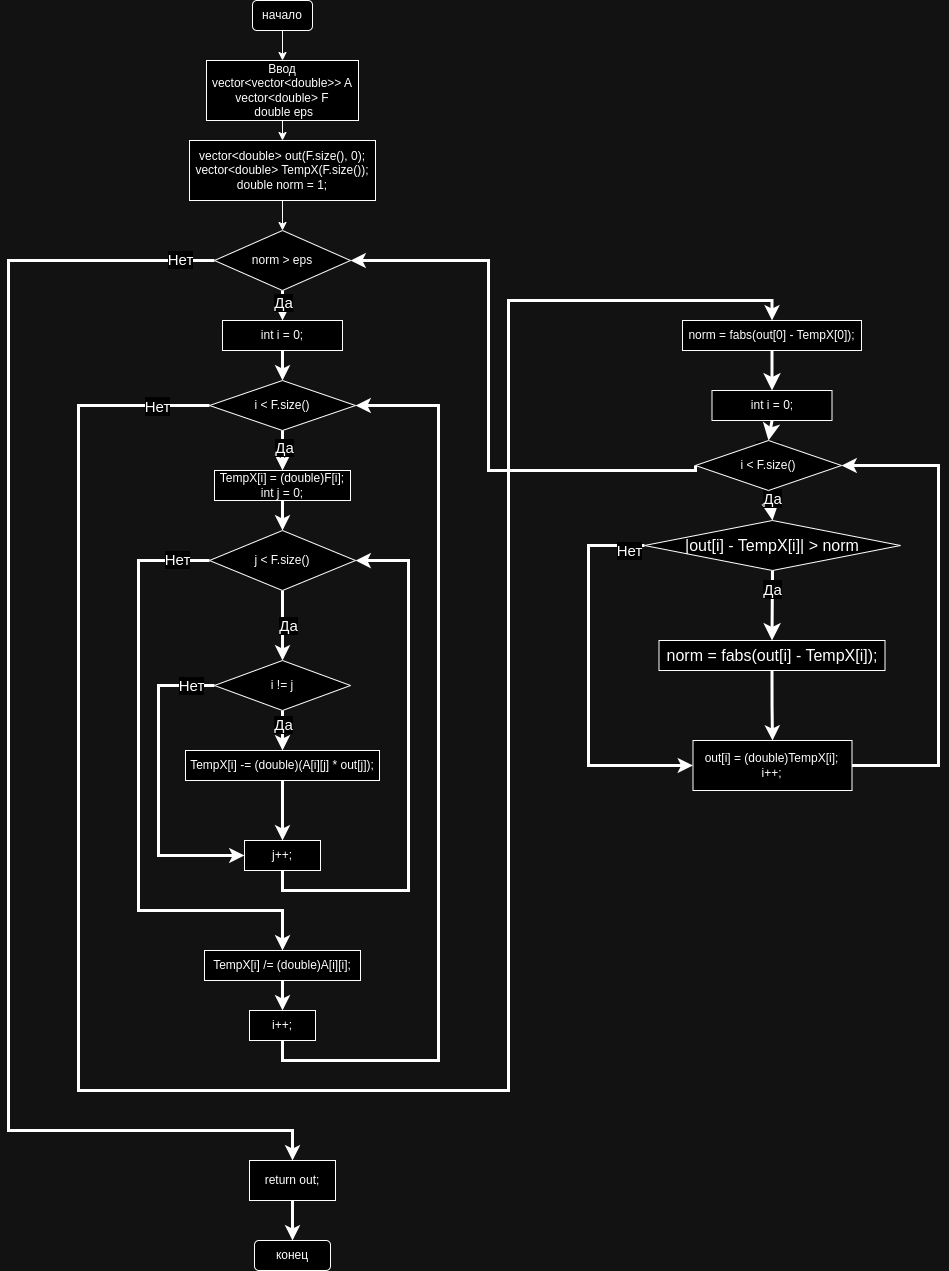


Рисунок 1. Блок-схема последовательного алгоритма.

Параллельный алгоритм:

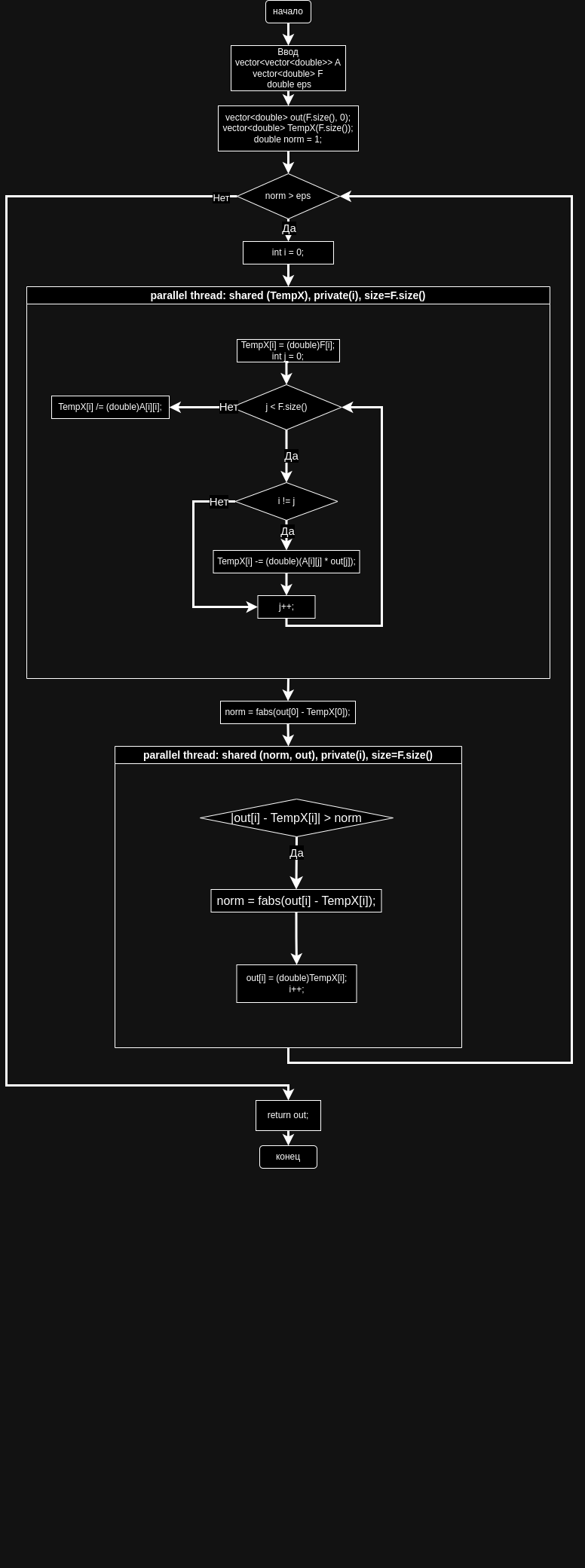


Рисунок 2. Блок-схема параллельного алгоритма.

1. **Параллельное программирование для графического процессора в среде NVIDIA CUDA.**

Задание: Вычислить методом последовательных приближений распределение значений температуры точек верхней грани параллелепипеда размером k \* m \* n, имеющего внутри полость в виде цилиндра. Теплопроводность материала не равна нулю. Нижняя грань параллелепипеда имеет постоянную температуру 0. Одна из сторон верхней грани имеет температуру T1, противоположная - температуру T2.

Последовательный алгоритм:

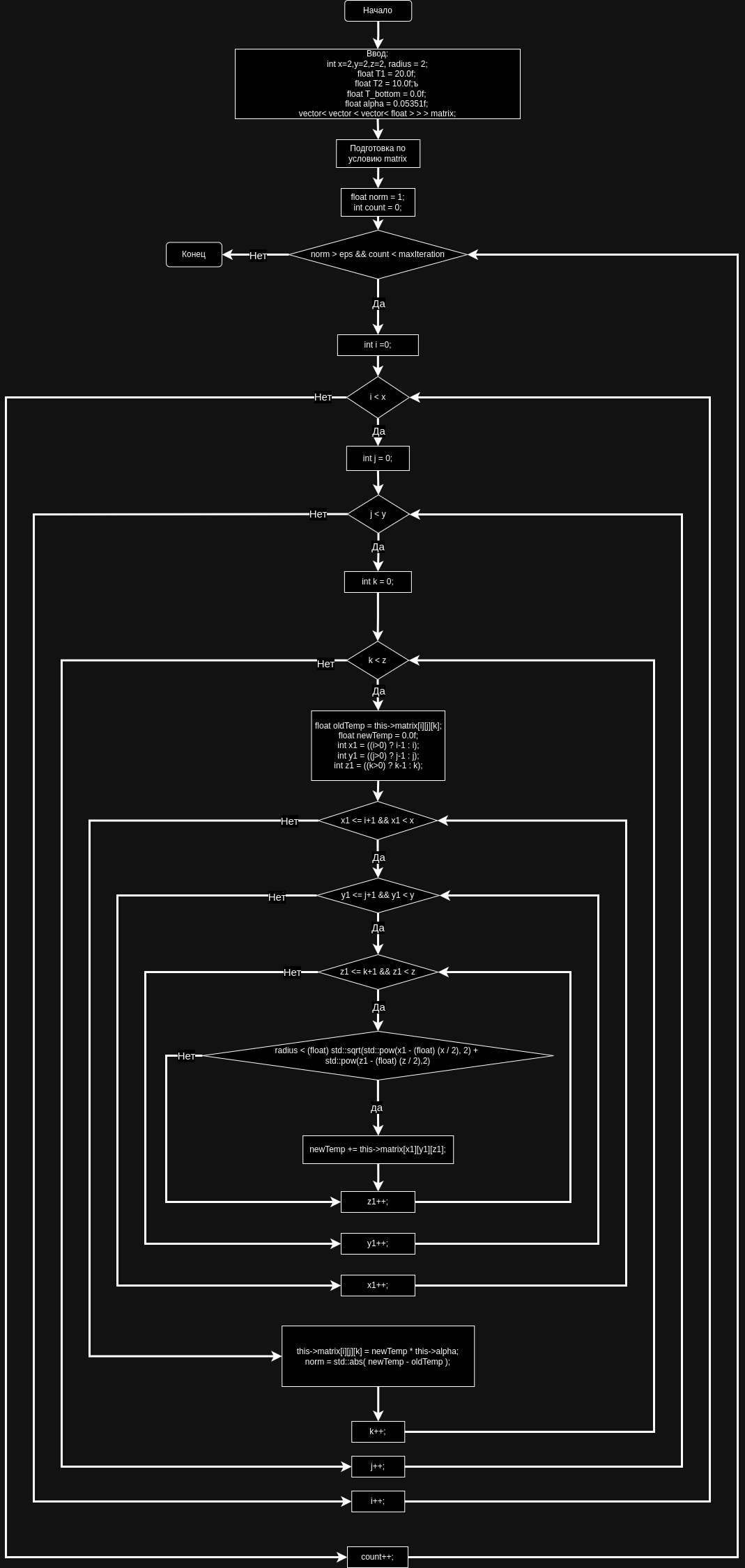


Рисунок 3. Блок-схема последовательного алгоритма.

Параллельный алгоритм:

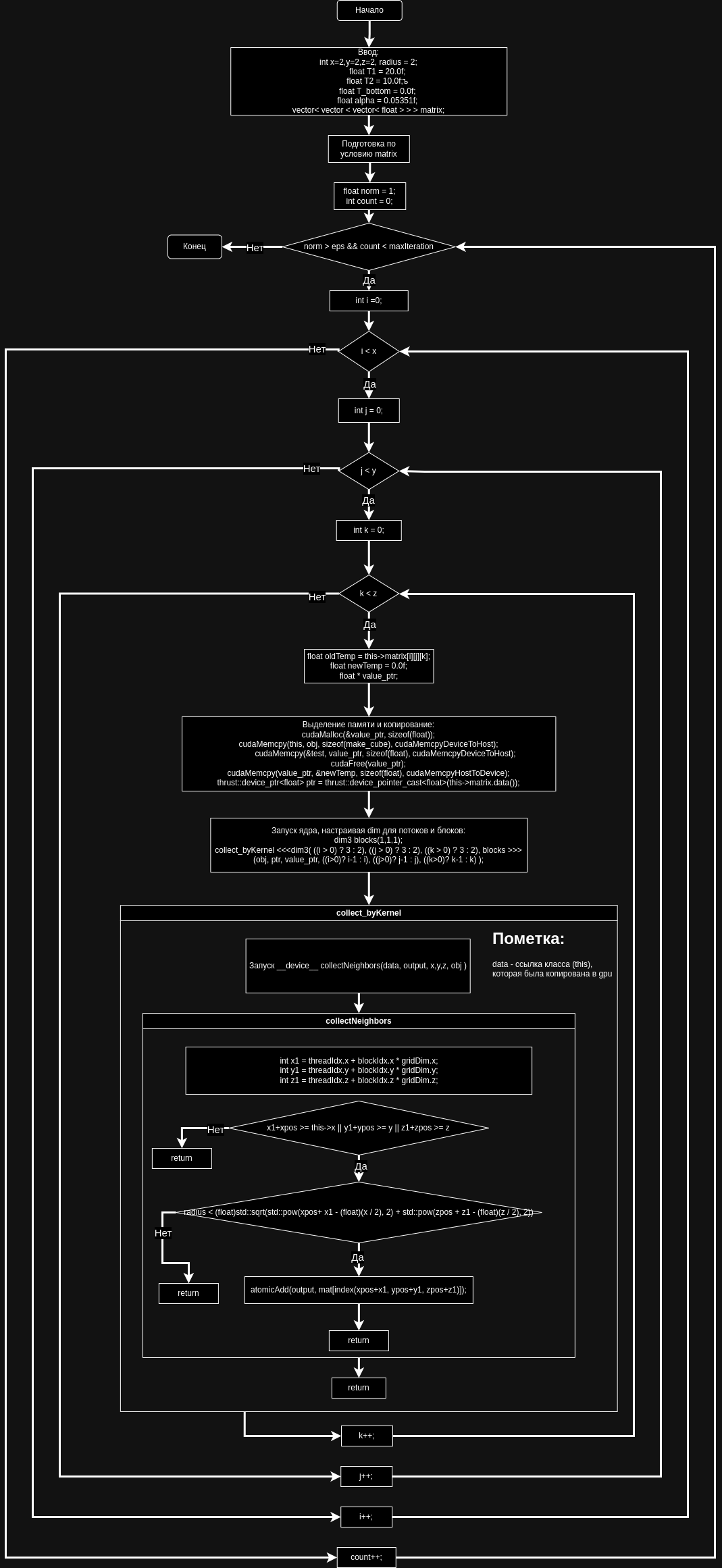


Рисунок 4. Блок-схема параллельного алгоритма.

1. Параллельноепрограммирование для гетерогенных вычислительных систем с использованием технологии OpenCL.

Задание: Вычислить методом последовательных приближений распределение значений температуры в точках верхней границы плоской прямоугольной области размером m \* n, имеющей внутри круглый вырез. Теплопроводность материала области конечна и не равна нулю. Левая и нижняя граница области имеют постоянную температуру 0. Правый верхний угол имеет постоянную температуру T, значения температур точек правой границы постоянны и равномерно убывают от T до 0. Все остальные точки в начальный момент имеют температуру Т3.

Последовательный алгоритм:

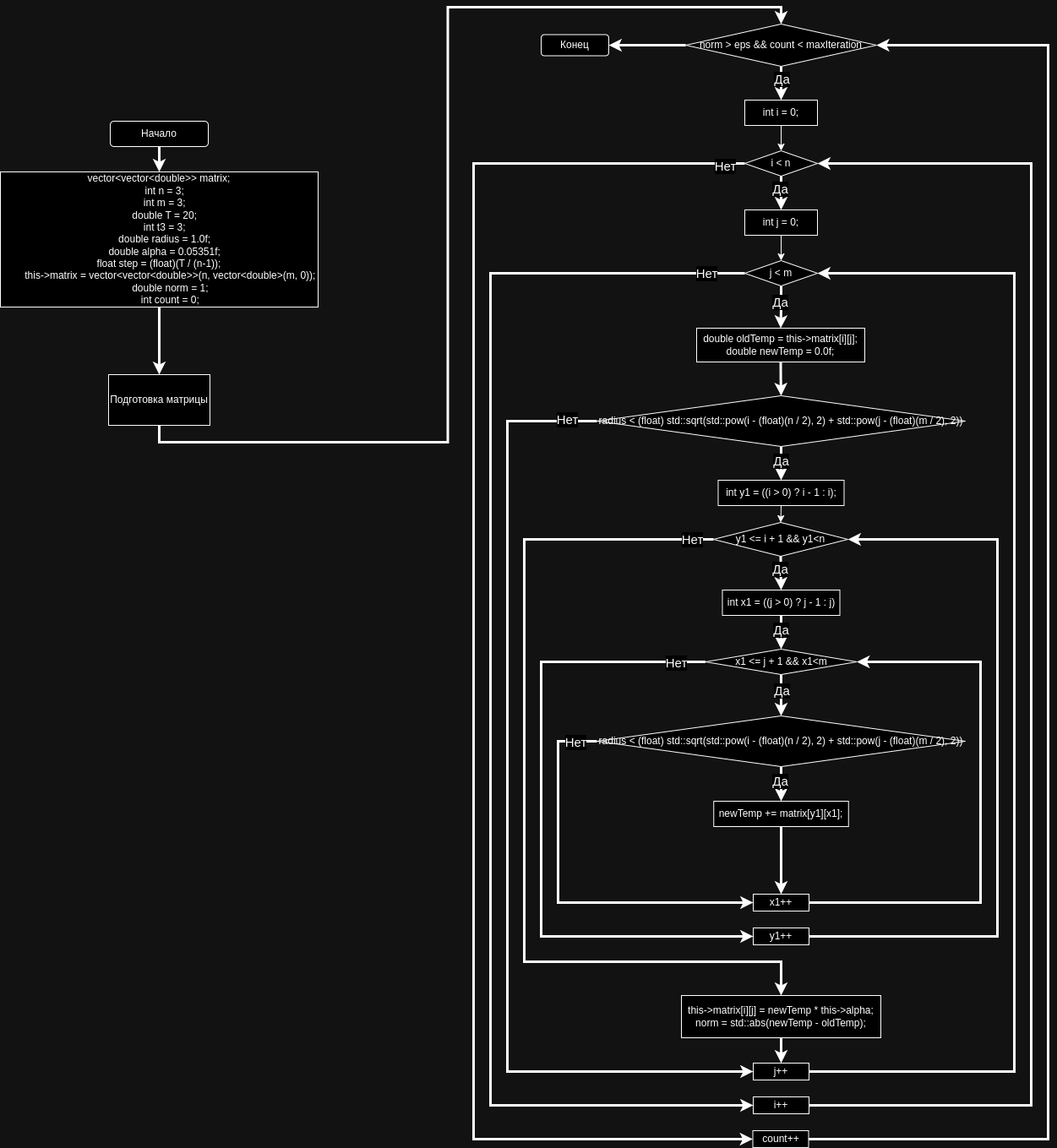


Рисунок 5. Блок-схема последовательного алгоритма.

Параллельный алгоритм:

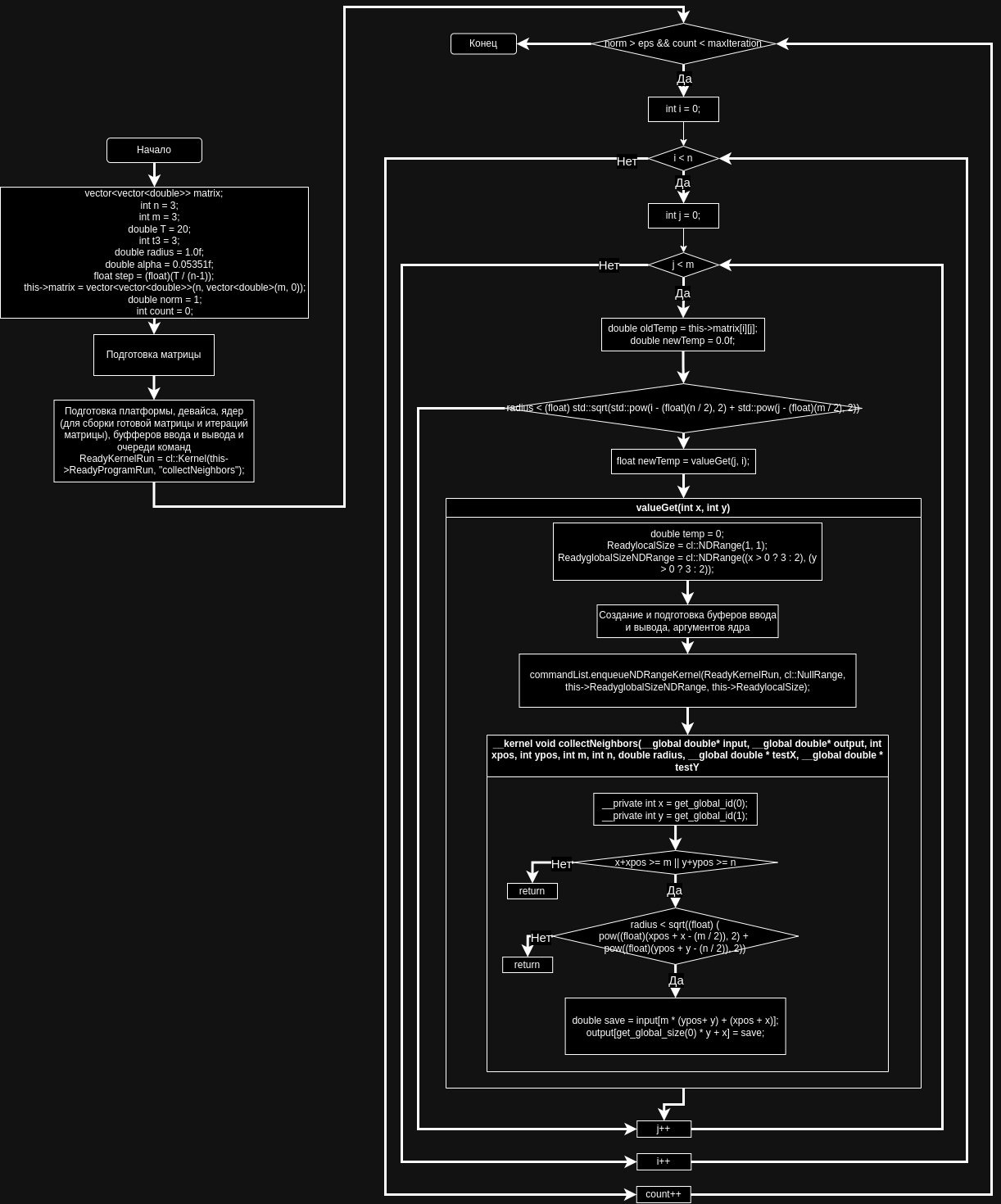


Рисунок 6. Блок-схема параллельного алгоритма.

1. Параллельное программирование с использованием технологии MPI и библиотеки профилирования MPE.

Задание: Найти максимальное число, меньшее заданного N, которое может быть представлено как сумма степеней 2, 3 и 4 простых чисел (минимальное такое число есть 28 = 22+23+24)

Последовательный алгоритм:

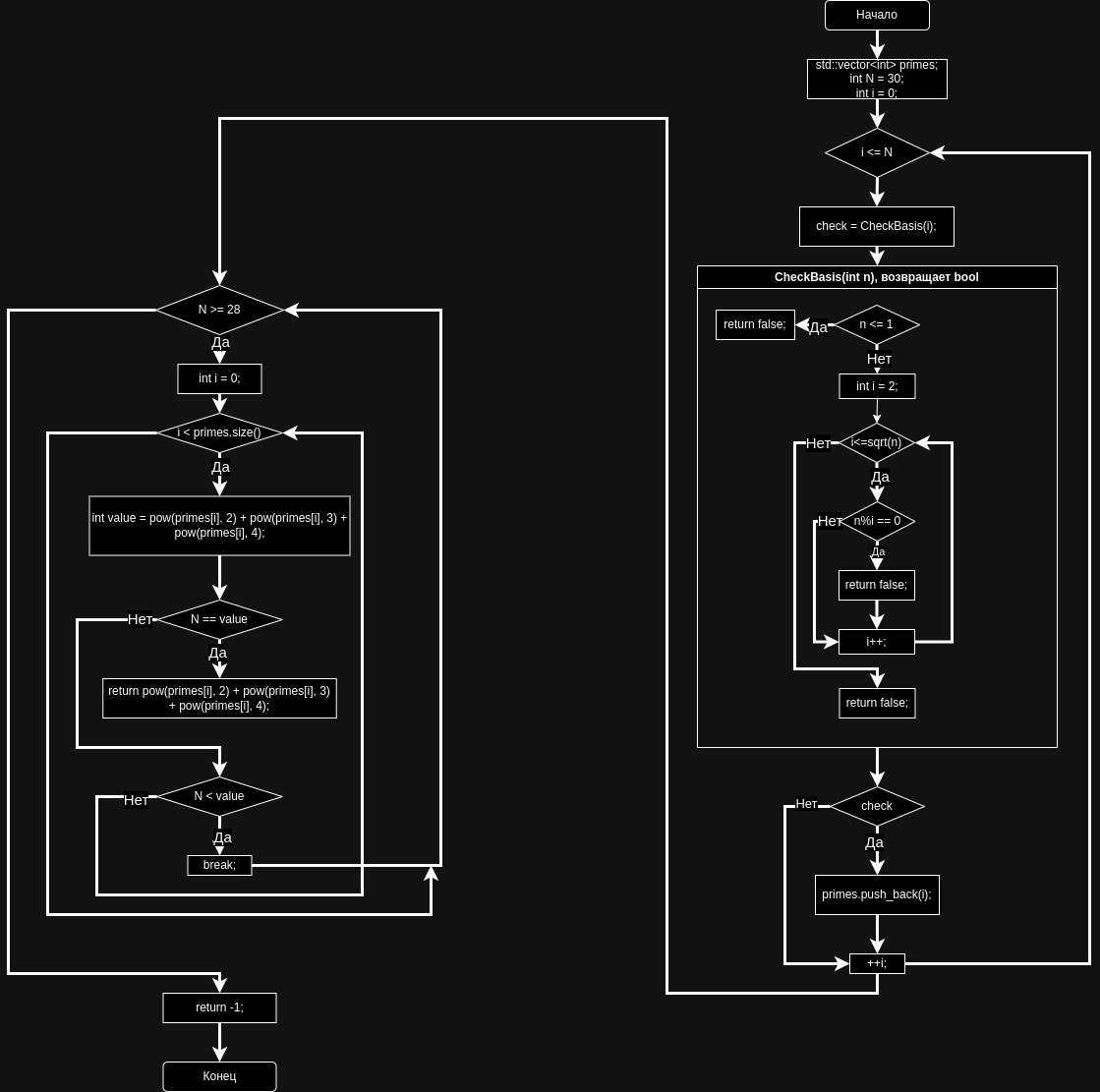


Рисунок 7. Блок-схема последовательного алгоритма.

Параллельный алгоритм:

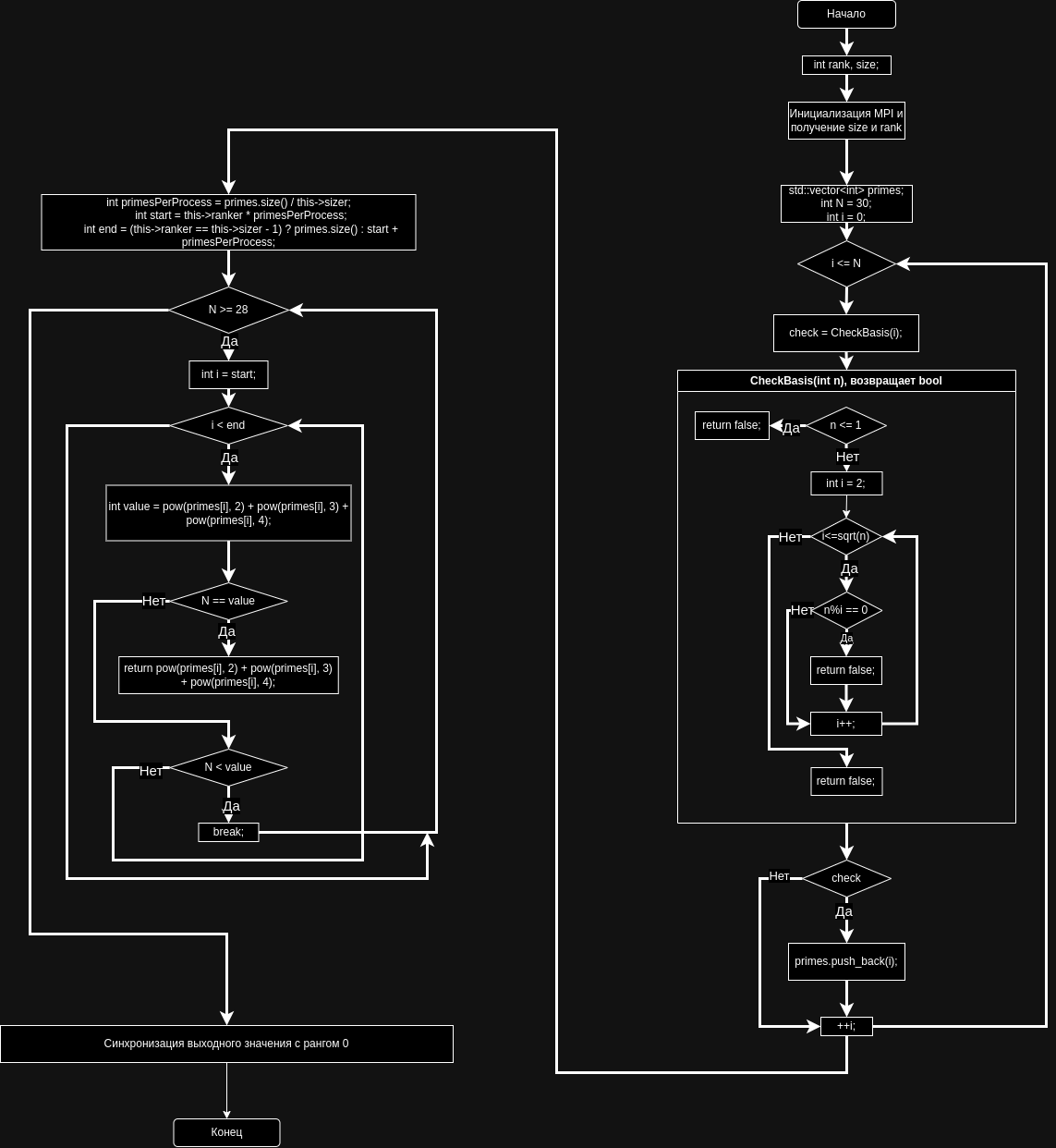


Рисунок 8. Блок-схема параллельного алгоритма.

**Результаты серий вычислений:**

1. Параллельное программирование для систем с общей памятью с использованием технологии OpenMP.

Для сравнения был подан первоначальный ввод данных:

Insert random matrix (y/n): y

Insert size of square matrix: 150

Insert max values: 20

Insert eps: 0.01

Для получения результатов введем разные размеры квадратной матрицы:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер матрицы | 10 | 50 | 100 | 1000 | 2000 | 5000 | 10000 | 20000 |
| Метод | Время в секундах | | | | | | | |
| Линейный | ~0 | ~0 | 0.001 | 0.008 | 0.033 | 0.21 | 0.85 | 3.45 |
| Параллельный | 0.008 | 0.01 | 0.0005 | 0.001 | 0.004 | 0.051 | 0.11 | 0.46 |

Как видно, момент прироста ускорения вычисления на размере матрицы ~1000. Это может быть связано с тем, что процессору удобно вычислять линейно на очень маленьких размерах, ведь во время компиляции идет оптимизация процесса. Для параллельного вычисления требуется чуть больше времени на подготовку запуска, распараллеливания, ожидания завершения и вывода результата.

1. Параллельное программирование для графического процессора в среде NVidia CUDA.

Для сравнения был подан первоначальный ввод данных:

Run test? (y/n): n

Show iteration? (y/n): n

Show GPU information? (y/n): n

Enter alpha: 0.05351

Enter eps: 0.01

Enter maximum of iterations: 1

Для получения результатов введем разные размеры кубической матрицы радиус которой равен 3 с 1 итерацией:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер матрицы | 10 | 20 | 35 | 50 | 65 | 85 | 100 | 110 |
| Метод | Время в секундах | | | | | | | |
| Линейный | 0,55 | 0,36 | 1,87 | 6,88 | 15,14 | 34,27 | 55,12 | 73,36 |
| Параллельный | 0,44 | 0,44 | 2,35 | 5,7 | 12,35 | 27,31 | 44,409 | 58,75 |

Как видно, момент прироста ускорения вычисления на размере матрицы ~35-50. Это связано с тем, что основа алгоритма – последовательный метод, но параллельная сумма температур всех соседей. Первоначальный провал в скорости может быть связан с тем, что перед выполнением запуска ядра требуется время на подготовку и запуск (копирование на GPU и обратно, запуск из global в device и т.д.).

1. Параллельноепрограммирование для гетерогенных вычислительных систем с использованием технологии OpenCL.

Для сравнения был подан первоначальный ввод данных:

Run test? (y/n): n

Show GPU information? (y/n): n

Enter temperature (T, t3): 20 20

Enter alpha: 0.05351

Enter eps: 0.01

Enter maximum of iterations: 1

Для получения результатов введем разные размеры квадратной матрицы радиус которой равен 3 с 1 итерацией:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер матрицы | 10 | 20 | 35 | 50 | 65 | 85 | 100 | 110 |
| Метод | Время в секундах | | | | | | | |
| Линейный | 0,29 | 0,14 | 0,56 | 1,19 | 2,21 | 3,42 | 5,31 | 6,43 |
| Параллельный | 0,32 | 0,19 | 0,53 | 1,29 | 2,15 | 3,84 | 4,97 | 5,7 |

Как видно, момент прироста ускорения вычисления на размере матрицы ~35-65. Это связано с тем, что основа алгоритма – последовательный метод, но параллельная сумма температур всех соседей. Первоначальный провал в скорости может быть связан с тем, что перед выполнением запуска ядра требуется время на подготовку и запуск (копирование на GPU и обратно, запуск ядра и т.д.).

1. Параллельное программирование с использованием технологии MPI и библиотеки профилирования MPE.

Для получения результатов введем разные значения N:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | 10 | 1000 | 2000 | 3000 | 10000 | 20000 | 30000 |
| Метод | Время в секундах | | | | | | |
| Линейный | 0,43 | 0,65 | 0,27 | 0,15 | 0,19 | 0,27 | 0,58 |
| Параллельный | 0,60 | 0,12 | 0,15 | 0,20 | 0,11 | 0,20 | 0,53 |

В основе параллельного алгоритма лежит получение ранга и размера и выполнение вычисления так, чтобы его каждый ранг имели равное кол-во итераций на процесс.

**Выводы:**

Была произведена работа параллельного вычисления на четырех разных технологиях, где присутствовали вычисления как на GPU, так и на CPU. Можно сделать вывод, что самым легкодоступным был OpenMP, ведь технология позволяет с легкостью распараллелить алгоритм без особых знаний механизма работы технологии. Но более производительным показал себя CUDA, алгоритм вычисления был хоть и не самым лучшим (как и само его использование), но он довольно удобен в инструментарии, в отличие от OpenCL, которому приходится собирать ядро для работы.