Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт по лабораторной работе №4 по курсу "Анализ алгоритмов"

Тема Параллельные вычисления

Студент Леонов В.В.

Группа ИУ7-56Б

Оценка (баллы) _____

Преподаватель Волкова Л.Л.

Москва — 2021 г.

Содержание

Введение

Многопоточность - это специализированная форма многозадачности, и многозадачность - это функция, которая позволяет вашему компьютеру одновременно запускать две или несколько программ. В общем, существует два типа многозадачности: основанные на процессах и потоки [1].

Многозадачность на основе процессов управляет одновременным выполнением программ. Многозадачность на основе потоков связана с одновременным выполнением частей одной и той же программы.

Многопоточная программа содержит две или несколько частей, которые могут запускаться одновременно. Каждая часть такой программы называется потоком, и каждый поток определяет отдельный путь выполнения.

Целью данной лабораторной работы являются изучение и реализация многопоточности на основе алгоритмов компьютерной графики, в частности алгоритма поворота двумерной фигуры, представленной в виде массива точек.

Для достижения указанной выше цели следует выполнить следующие задачи:

- изучить понятие параллельных вычислений и алгоритм поворота фигуры в двумерном растре;
- привести схемы рассматриваемого алгоритма в последовательном и параллельном вариантах;
- реализовать последовательный и параллельный алгоритм поворота фигуры в двумерном растре на одном из языков программирования, используя нативные потоки;
- провести сравнительный анализ реализаций алгоритма на основе экспериментальных данных;
- описание и обоснование полученных результатов в отчете о выполненной лабораторной работе.

1 Аналитическая часть

Задача поворота фигуры, представленной в виде массива точек в двумерном растре, является весьма актуальной, т.к. компьютерная графика стала неотъемлемой частью повседневной интернет-жизни человека, и существует потребность в быстром рендеринге изображения, например, при анимации поворота фигуры на двумерном растре. Как известно, в экранной плоскости изображение представляет из себя набор пикселей (точек). Очевидно, что какая-либо фигура - это тоже набор точек экранной плоскоти, и для поворота фигуры требуется над каждой ее точкой произвести преобразование для получения новой позиции.

Если использовать один поток для рендеринга изображения, то при большом количестве и сложности фигур изображение будет генерироваться ощутимо долго, что будет приносить человеку дискомфорт при восприятии, однако если распараллелить этот процесс, т.е. параллельно генерировать части изображения (т.к. эта операция выполняется независимо для каждой точки), то это может дать коллосальной прирост производительности.

1.1 Алгоритм поворота точек двумерного растра

Пусть необходимо повернуть точку $P(x, y)$ вокруг начала координат O на угол ϕ [2]. Изображение новой точки обозначим $P'(x', y')$. Всегда существуют четыре числа a, b, c, d такие, что новые координаты могут быть вычислены по значениям старых координат из следующей системы уравнений:

$$\begin{cases} x' = a \cdot x + b \cdot y \\ y' = c \cdot x + d \cdot y \end{cases} \tag{1.1}$$

Для получения значений a, b, c, d рассмотрим точку $P(x, y) = (1, 0)$.

Полагая $x = 1$ и $y = 0$ в уравнении 1.1, получим:

$$\begin{cases} x = a \\ y = c \end{cases} \tag{1.2}$$

Но в этом простом случае, значения x' и y' равны соответственно $\cos(\phi)$ и $\sin(\phi)$. Тогда имеем:

$$\begin{cases} a = \cos(\phi) \\ c = \sin(\phi) \end{cases} \tag{1.3}$$

Аналогичным образом рассматривая точку $P(x, y) = (0, 1)$, получим:

$$\begin{cases} b = -\sin(\phi) \\ d = \cos(\phi) \end{cases} \tag{1.4}$$

Тогда всесто системы уравнений 1.1 можно записать:

$$\begin{cases} x' = x \cdot \cos(\phi) - y \cdot \sin(\phi) \\ y' = x \cdot \sin(\phi) + y \cdot \cos(\phi) \end{cases} \tag{1.5}$$

Система уравнений 1.6 описывает поворот вокруг точки O - начала системы координат, но часто нужно нужно выполнить поворот относительно заданной точки (x_c, y_c). Тогда система 1.6 примет следующий вид:

$$\begin{cases} x' = x_c + (x - x_c) \cdot \cos(\phi) - (y - y_c) \cdot \sin(\phi) \\ y' = y_c + (x - x_c) \cdot \sin(\phi) + (y - y_c) \cdot \cos(\phi) \end{cases} \tag{1.6}$$

Часто такие преобразования удобно представить в виде матричных преобразований:

$$M = \begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.7}$$

Таким образом, распараллеливание будет заключаться в том, что массив точек будет разбиваться на подмассивы, для каждого из которых независимо от других будет решаться задача преобразования.

1.2 Вывод

В данном разделе была проанализирована предметная область, установлена актуальность задачи, также был рассмотрен алгоритм задачи, которая будет подвергнута распараллеливанию.

2 Конструкторская часть

В данном разделе представлены схемы алгоритмов, описанных в аналитическом разделе.

2.1 Схемы алгоритмов

На рисунке 2.1 приведена схема последовательного алгоритма.

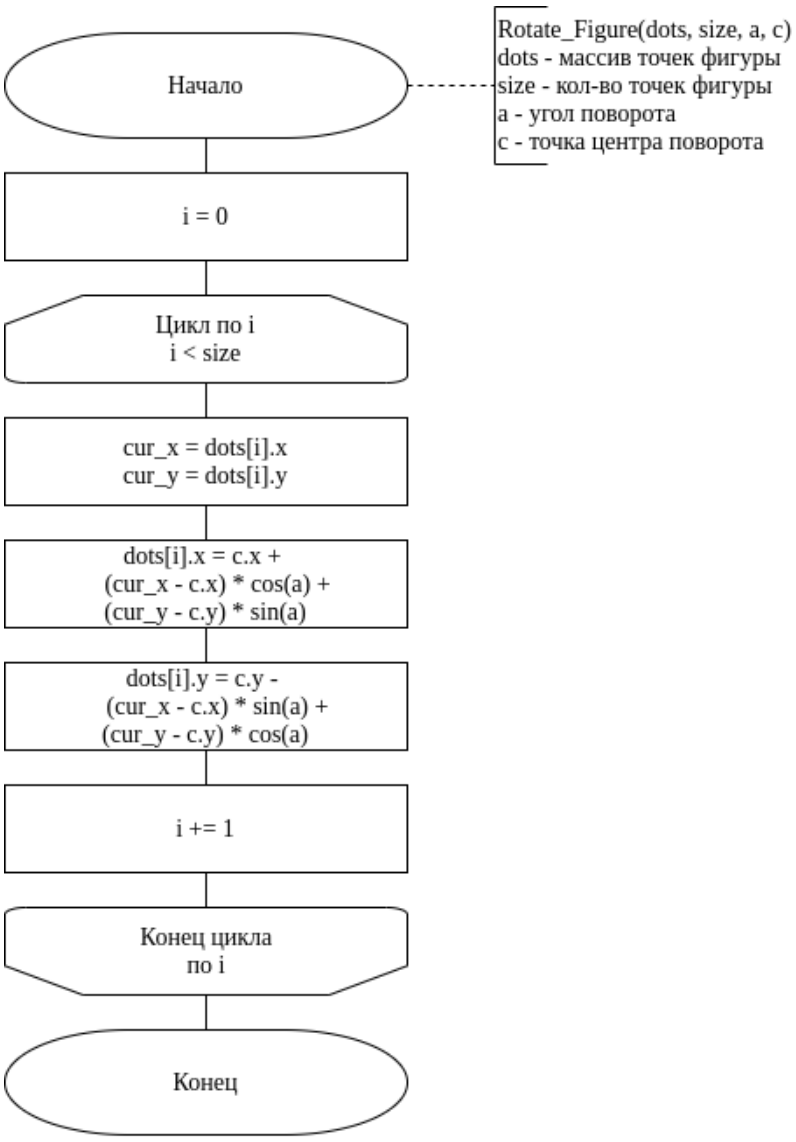

Рисунок 2.1 – Схема последовательного алгоритма поворота

На рисунке 2.2 приведена схема распараллеливания алгоритма поворота фигуры двумерного растра.

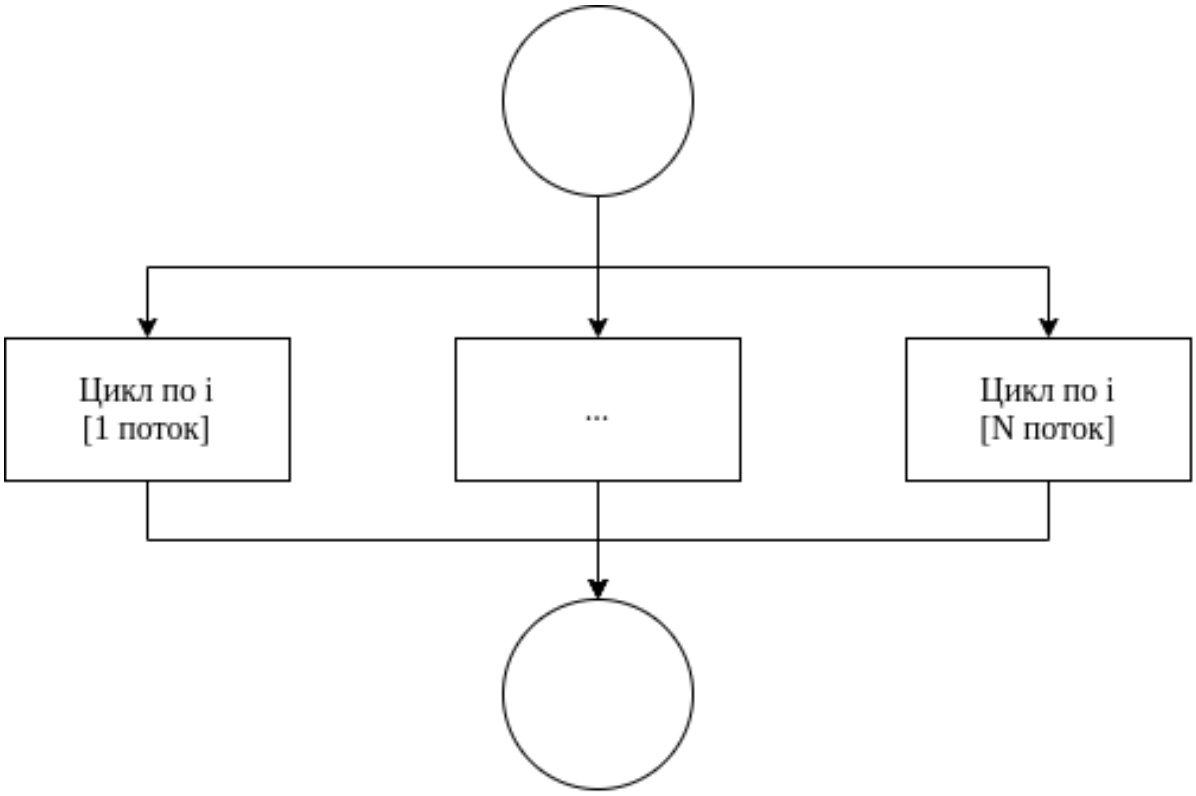

Рисунок 2.2 – Схема распараллеливания алгоритма поворота фигуры двумерного растра

На рисунке 2.3 приведена схема многопоточного алгоритма поворота фигуры двумерного растра.

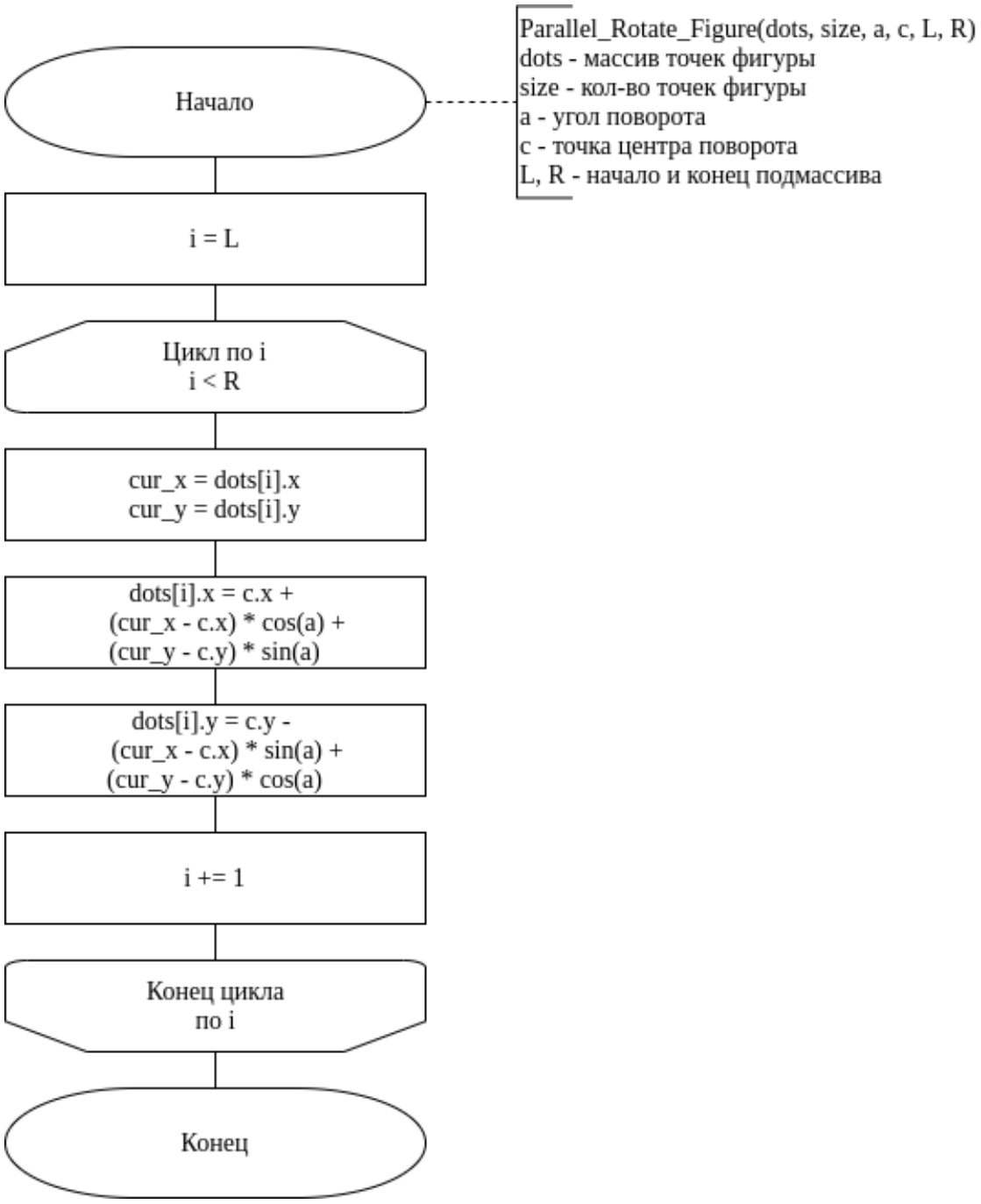

Рисунок 2.3 – Схема многопоточного алгоритма поворота фигуры двумерного растра

На рисунке 2.4 приведена схема создания потоков и запуска параллельных вычислений.

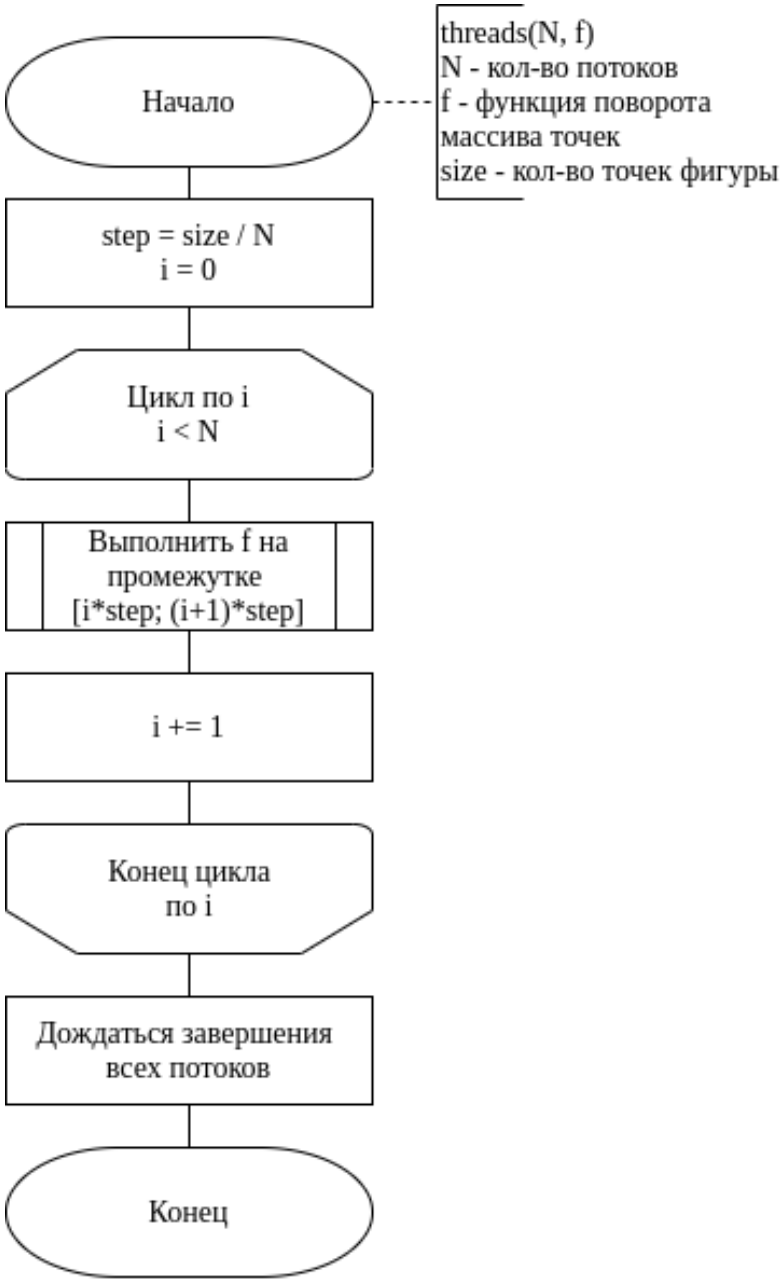

Рисунок 2.4 – Схема создания потоков и запуска параллельных вычислений

2.2 Вывод

На основе теоретических данных, полученных из аналитического раздела, были построены схемы требуемых алгоритмов.

3 Технологическая часть

В данном разделе приведены требования к программному обеспечению, средства реализации и листинги кода.

3.1 Требования к ПО

К программе предъявляется ряд требований:

- взаимодействие программы и пользователя реализованно в виде меню;
- пользователь может ввести фигуру в виде массива точек для поворота ручным способом или сгенерировать, заполнив случайными значениями;
- элементами массива могут быть вещественные числа;
- считается, что пользователь вводит корректные данные;
- на выходе — преобразованный массив точек фигуры двумерного растра;
- программа обладает автоматизированной системой сравнительного анализа скорости работы алгоритма при различном количестве потоков.

3.2 Средства реализации

В качестве языка программирования для реализации данной лабораторной работы был выбран многопоточный язык C++ [3]. Данный выбор обусловлен моим желанием расширить свои знания в области применения данного языка.

3.3 Листинг кода

В листингах 3.1–3.3 приведены реализации алгоритмов поворота, а также вспомогательные функции.

Листинг 3.1 – Многопоточная функция поворота фигуры

```
void thread_rotate(figure_t &figure, double angle, size_t start, size_t end)
{
    for (size_t i = start; i < end; i++)
        point_rotate(figure[i], angle);
}

void figure_rotate(figure_t &figure, double angle, size_t thrd_cnt)
{
    size_t elem_per_thrd = figure.size() / thrd_cnt;
    thread *thrds = new thread[thrd_cnt];

    for (size_t i = 0; i < thrd_cnt - 1; i++)
    {
        size_t start = i * elem_per_thrd;
        size_t finish = (i + 1) * elem_per_thrd;

        thrds[i] = thread(thread_rotate, ref(figure), angle, start, finish);
    }
    thrds[thrd_cnt - 1] = thread(thread_rotate, ref(figure), angle, (thrd_cnt - 1) *
        elem_per_thrd, figure.size());

    for (size_t i = 0; i < thrd_cnt; i++)
        thrds[i].join();
    delete[] thrds;
}
```

Листинг 3.2 – Функция поворота точки относительно другой

```
void point_rotate(point_t &point, double angle)
{
    double tmp_x = point.x;
    double angle_pi = degrees_to_rad(angle);

    point.x = point.x * cos(angle_pi) - point.y * sin(angle_pi);
    point.y = tmp_x * sin(angle_pi) + point.y * cos(angle_pi);
}
```

Листинг 3.3 – Функция для замера тиков процессора

```
uint64_t tick(void)
{
    uint32_t high, low;
    __asm__ __volatile__(
        "rdtsc\n"
        "movl %%edx, %0\n"
        "movl %%eax, %1\n"
        : "=r"(high), "=r"(low)::"%rax", "%rbx", "%rcx", "%rdx");

    uint64_t ticks = ((uint64_t)high << 32) | low;

    return ticks;
}
```

В таблице 3.1 приведены функциональные тесты для алгоритмов сортировки. Все тесты пройдены успешно (таблица 3.2).

Таблица 3.1 – Ожидаемый результат работы программы

Входной массив	Угол поворота	Ожидаемый результат
$[[1,2],[3,4],[5,6]]$	45	$[[-0.70,2.12],[-0.70,4.94],[-0.70,7.77]]$
$[[1,2],[3,4],[5,6]]$	360	$[[1,2],[3,4],[5,6]]$
$[[1,2],[3,4],[5,6]]$	180	$[[-1,-2],[-3,-4],[-5,-6]]$
$[[1,2],[3,4],[5,6]]$	-90	$[[2,-1],[4,-3],[6,-5]]$

Таблица 3.2 – Фактический результат работы программы

Входной массив	Угол поворота	Фактический результат
$[[1,2],[3,4],[5,6]]$	45	$[[-0.70,2.12],[-0.70,4.94],[-0.70,7.77]]$
$[[1,2],[3,4],[5,6]]$	360	$[[1,2],[3,4],[5,6]]$
$[[1,2],[3,4],[5,6]]$	180	$[[-1,-2],[-3,-4],[-5,-6]]$
$[[1,2],[3,4],[5,6]]$	-90	$[[2,-1],[4,-3],[6,-5]]$

3.4 Вывод

В данном разделе был реализован и протестирован спроектированный алгоритм поворота точек фигуры в двумерном растре.

4 Исследовательская часть

4.1 Пример работы

Демонстрация работы программы приведена на рисунке 4.1.

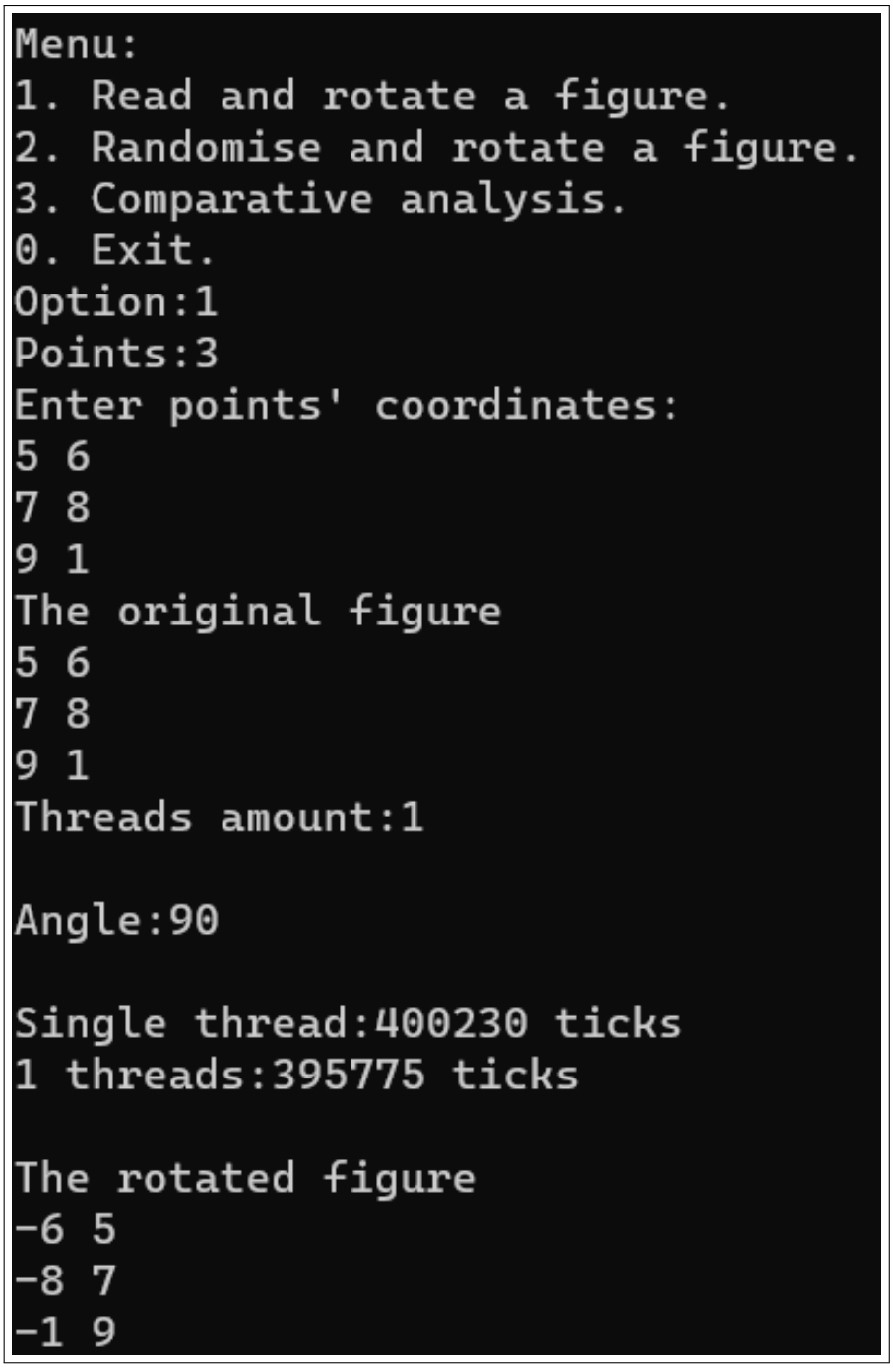

Рисунок 4.1 – Демонстрация работы алгоритма поворота

4.2 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

- Операционная система: Windows 10 64-bit [4].
- Память: 16 GB.
- Процессор: AMD Ryzen 5 4600H [5] @ 3.00 GHz.

Тестирование проводилось на ноутбуке при включённом режиме производительности. Во время тестирования ноутбук был нагружен только системными процессами.

4.3 Время выполнения алгоритмов

В таблице 4.1 и на рисунке 4.2 приведены зависимости времени работы алгоритмов для различного количества потоков (количество точек соответствует разрешению 1920x1080).

Таблица 4.1 – Результат времени работы алгоритма при различном количестве потоков

Количество потоков	Время, тики
1	1039794938
2	738458153
4	380418539
8	139540901
16	95889809
32	67919394

В таблице 4.2 и на рисунке 4.3 приведены зависимости времени работы алгоритмов для различного количества точек.

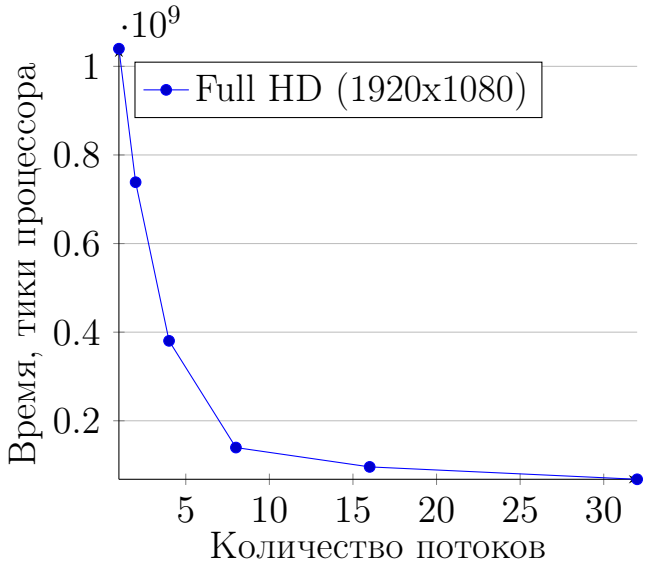

Рисунок 4.2 – Сравнение алгоритмов на различных количествах потоков

Таблица 4.2 – Результат времени работы алгоритма при различном количестве точек

Количество точек	1 поток	12 потоков
4000	941572	1818340
16000	2733740	1960482
64000	10062319	3676082
256000	38439025	7296946
512000	76648487	10203703
2048000	303856737	29792272

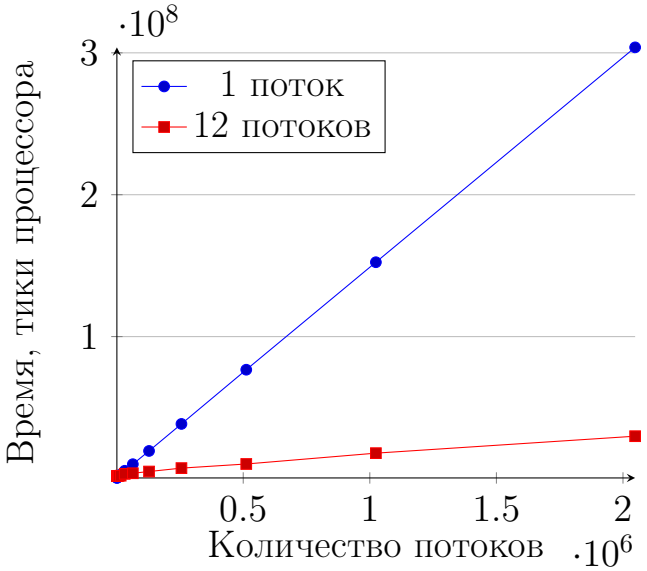

Рисунок 4.3 – Сравнение алгоритмов при различном количестве элементов

4.4 Вывод

Исходя из полученных эксперементальных данных можно сделать следующие выводы.

Использование многопоточности однозначно может дать существенный выигрыш по времени работы алгоритма. В алгоритме поворота фигуры двумерного растра использование многопоточности позволило добиться 15-кратного увеличения производительности.

Следует отметить, что использование многопоточности на небольшом количестве данных может быть неэффективно, т.к. накладные расходы на создание потоков будут превышать выигрыш от параллельных вычислений.

Заключение

В ходе выполнения работы были выполнены все поставленные задачи и изучены методы динамического программирования с реализацей многопоточности на основе алгоритма поворота фигуры двумерного растра.

Использование многопоточности однозначно может дать существенный выигрыш по времени работы алгоритма. В алгоритме поворота фигуры двумерного растра использование многопоточности позволило добиться 15-кратного увеличения производительности.

Следует отметить, что использование многопоточности на небольшом количестве данных может быть неэффективно, т.к. накладные расходы на создание потоков будут превышать выигрыш от параллельных вычислений.

Список литературы

[1] Williams A. C++ Concurrency in Action: Practical Multithreading. Manning Publications Co., 2019.

[2] Rotating a point about another point [Электронный ресурс]. Режим доступа: `https://stackoverflow.com/questions/2259476/` (дата обращения: 26.10.2021).

[3] cppreference.com [Электронный ресурс]. Режим доступа: `https://en.cppreference.com/w/` (дата обращения: 26.10.2021).

[4] Explore Windows 10. [Электронный ресурс]. Режим доступа: `https://www.microsoft.com/en-us/windows/` (дата обращения: 02.10.2021).

[5] AMD Processors [Электронный ресурс]. Режим доступа: `https://www.amd.com/en/products/apu/amd-ryzen-5-4600h` (дата обращения: 02.10.2021).