САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет

Шастин Егор

Отчёт по добавлению многопоточности в лабораторную работу №1 (обработка BMP-изображений).

студент 1 курса.

Санкт-Петербург

2025

1) Цель

Обновить реализацию лабораторной работы 1 (обработка изображений), добавив многопоточность для повышения производительности.

2) Описание изменений

```
auto worker = [&](int y_start, int y_end) {
    for (int y = y_start; y < y_end; ++y) {
        if (y == 0 || y == height - 1) continue;
        for (int x = 1; x < width - 1; ++x) {
            double red = 0.0, green = 0.0, blue = 0.0;

            for (int ky = -1; ky <= 1; ++ky) {
                for (int kx = -1; kx <= 1; ++kx) {
                    RGB pixel = bmp_file->file_data[(y + ky) * width + (x + kx)];
                    double weight = gaussianKernel[ky + 1][kx + 1];
                    blue += pixel.blue * weight;
                    green += pixel.green * weight;
                    red += pixel.red * weight;
                }
            }

            file_data[y * width + x] = RGB(
                static_cast<uint8_t>(std::min(std::max(red, 0.0), 255.0)),
                static_cast<uint8_t>(std::min(std::max(green, 0.0), 255.0)),
                static_cast<uint8_t>(std::min(std::max(blue, 0.0), 255.0))
            );
        }
    }
};

std::vector<std::thread> threads;
int block = height / thread_count;
for (int i = 0; i < thread_count; ++i) {
    int y_start = i * block;
    int y_end = (i == thread_count - 1) ? height : y_start + block;
    threads.emplace_back(worker, y_start, y_end);
}

for (auto& t : threads) t.join();
```

```
auto worker = [&](int start_row, int end_row) {
    for (int y = start_row; y < end_row; ++y) {
        for (uint32_t x = 0; x < dib_header.width; ++x) {
            uint32_t old_index = y * dib_header.width + x;
            uint32_t new_index = x * new_bmp_file->dib_header.width + (dib_header.height - 1 - y);
            new_bmp_file->file_data[new_index] = file_data[old_index];
        }
    }
};

int total_rows = dib_header.height;
int rows_per_thread = total_rows / thread_count;
std::vector<std::thread> threads;

for (int i = 0; i < thread_count; ++i) {
    int start_row = i * rows_per_thread;
    int end_row = (i == thread_count - 1) ? total_rows : start_row + rows_per_thread;
    threads.emplace_back(worker, start_row, end_row);
}

for (auto& t : threads) t.join();
```

```cpp
int thread_count = 4;
std::cout << "BMP file size:" << bmp_file.bmp_header.file_size << " byte" << "\n";
std::cout << "-> First task completed: file size printed." << "\n\n";

// Flip BMP contra clockwise
auto new_bmp_file_1 = bmp_file.flip_BMP_90_contra_clockwise(thread_count);
new_bmp_file_1->Save_BMP_File("BMP_contra_clockwise.bmp"); // Save new file 1
std::cout << "-> Second task completed: file flip contra clockwise." << "\n";

// Use filter Gausa for BMP contra clockwise
auto start1 = std::chrono::high_resolution_clock::now();
ApplyGaussianFilter(new_bmp_file_1.get(), 5, 3.0, thread_count);
auto end1 = std::chrono::high_resolution_clock::now();

std::chrono::duration<double> duration1 = end1 - start1;
std::cout << "Filter time: " << duration1.count() << " seconds\n";
```

> Методы flip_BMP_90_clockwise() и flip_BMP_90_contra_clockwise() были переписаны с использованием std::thread. Каждому потоку выделяется блок строк исходного изображения, которые он обрабатывает независимо от других потоков.

> Функция ApplyGaussianFilter() также модифицирована для параллельной обработки изображения.

> Количество потоков задаётся переменной thread_count (= 4).

3) Результат

> До добавления многопоточности: 0.041 секунды (картинка 1).

> После добавления многопоточности: 0.017 секунды (картинка 2).

> Скорость выполнения увеличилась примерно в 2.41 раза.

```
junior@junior:~/Lab$ make run
./bin/read_BMP example.bmp
BMP File read succesful.
BMP file size:818058 byte
-> First task completed: file size printed.

Image saved successfully to BMP_contra_clockwise.bmp
-> Second task completed: file flip contra clockwise.
Filter time: 0.0419439 seconds
Image saved successfully to BMP_contra_clockwise_filter.bmp
Image saved successfully to BMP_clockwise.bmp
-> Third task completed: file flip clockwise.
Filter time: 0.021253 seconds
Image saved successfully to BMP_clockwise_filter.bmp
Save result filter Gausa
-> Fourth task completed: used filter Gausa for fliping BMP file
```

(картинка 1)

```
junior@junior:~/Lab$ make run
./bin/read_BMP example.bmp
BMP File read succesful.
BMP file size:818058 byte
-> First task completed: file size printed.

Image saved successfully to BMP_contra_clockwise.bmp
-> Second task completed: file flip contra clockwise.
Filter time: 0.0177806 seconds
Image saved successfully to BMP_contra_clockwise_filter.bmp
Image saved successfully to BMP_clockwise.bmp
-> Third task completed: file flip clockwise.
Filter time: 0.00886915 seconds
Image saved successfully to BMP_clockwise_filter.bmp
Save result filter Gausa
-> Fourth task completed: used filter Gausa for fliping BMP file
```

(картинка 2)

4) Вывод

Цель работы достигнута. Добавление многопоточности позволило существенно повысить производительность программы, не переписывая основную архитектуру. Итоговое решение работает более чем в два раза быстрее при тех же входных данных и условиях запуска.