Звіт до лабораторної роботи з паралельного програмування

*“Алгоритм обробки зображення”*

Гизили Євгенія, ТТП-3, 2018 р.

# Постановка задачі

В межах лабораторної роботи реалізував алгоритм перетворення з кольорової картинки в чорно-білу.

# Опис алгоритму

1. Зчитуємо bmp зображення в масив розміром 256 х 256 х 3, де 256 – це висота і ширина, а 3 – rgb зображеня.

2. Для кожно пікселя застосовуємо фунцію, яка обраховує тональність сірого на основі червоного, зеленого, блакитного.

3. Записуємо результат у файл.

# Послідовна реалізація

**char const \*inputFile = "data/in.bmp";**

**char const \*outputFile = "data/out.bmp";**

**unsigned char image[256][256][3];**

**clock\_t start, end;**

**readBotmapImage(inputFile, image);**

**start = clock();**

**char grayScale;**

**for (int i=0; i < 256; i++) {**

**for(int j = 0; j < 256; j ++) {**

**grayScale = toGrayScale(image[i][j][0], image[i][j][1], image[i][j][2]);**

**image[i][j][0] = grayScale;**

**image[i][j][1] = grayScale;**

**image[i][j][2] = grayScale;**

**}**

**}**

**end = clock();**

**printf("time: %d\n",end-start);**

**generateBitmapImage(image, 256, 256, outputFile);**

# Реалізація за допомогою OpenMP

**char const \*inputFile = "data/in.bmp";**

**char const \*outputFile = "data/out.bmp";**

**unsigned char image[256][256][3];**

**clock\_t start, end;**

**readBotmapImage(inputFile, image);**

**start = clock();**

**char grayScale;**

**#pragma omp parallel num\_threads(2)**

**{**

**#pragma omp for private(grayScale, image)**

**for (int i=0; i < 256; i++) {**

**for(int j = 0; j < 256; j ++) {**

**grayScale = toGrayScale(image[i][j][0], image[i][j][1], image[i][j][2]);**

**image[i][j][0] = grayScale;**

**image[i][j][1] = grayScale;**

**image[i][j][2] = grayScale;**

**}**

**}**

**}**

**end = clock();**

**printf("%d\n",end-start);**

**generateBitmapImage(image, 256, 256, outputFile);**

# MPI

**. // Initialize the MPI environment**

**MPI\_Init(NULL, NULL);**

**// Get the number of processes**

**int world\_size;**

**MPI\_Comm\_size(MPI\_COMM\_WORLD, &world\_size);**

**// Getting rank**

**int world\_rank;**

**MPI\_Comm\_rank(MPI\_COMM\_WORLD, &world\_rank);**

**if (world\_rank == 0) {**

**printf("master %i\n", world\_rank);**

**char const \*inputFile = "data/in.bmp";**

**char const \*outputFile = "data/out.bmp";**

**clock\_t start, end;**

**unsigned char pixel[3];**

**unsigned char image[256][256][3];**

**readBotmapImage(inputFile, image);**

**start = clock();**

**// master**

**MPI\_Send(image, SIZE\_PER\_NODE, MPI\_UNSIGNED\_CHAR, 1, 0, MPI\_COMM\_WORLD);**

**MPI\_Send(image[128], SIZE\_PER\_NODE, MPI\_UNSIGNED\_CHAR, 2, 0, MPI\_COMM\_WORLD);**

**MPI\_Recv(image, SIZE\_PER\_NODE, MPI\_UNSIGNED\_CHAR, 1, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);**

**MPI\_Recv(image[128], SIZE\_PER\_NODE, MPI\_UNSIGNED\_CHAR, 2, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);**

**end = clock();**

**printf("master time %d\n", end-start);**

**generateBitmapImage(image, 256, 256, outputFile);**

**} else {**

**printf("slave %i\n", world\_rank);**

**char grayScale;**

**unsigned char subimage[128][256][3];**

**MPI\_Recv(subimage, SIZE\_PER\_NODE, MPI\_UNSIGNED\_CHAR, 0, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);**

**for (int i = 0; i < 128; i++) {**

**for (int j = 0; j < 256; j++) {**

**grayScale = toGrayScale(subimage[i][j][0], subimage[i][j][1], subimage[i][j][2]);**

**subimage[i][j][0] = grayScale;**

**subimage[i][j][1] = grayScale;**

**subimage[i][j][2] = grayScale;**

**}**

**}**

**MPI\_Send(subimage, SIZE\_PER\_NODE, MPI\_UNSIGNED\_CHAR, 0, 0, MPI\_COMM\_WORLD);**

**}**

**MPI\_Finalize();**

# Результат

Без паралелізму: time: 3412  
OpenMP: time: 3075  
MPI: time: 1990