

Распределение баллов:

Баллы		
№	Частичное решение (ЧР)	Полное решение (ПР)
1	3	5
2	10	25
3	10	20
4	0	10
5	10	15
6	15	25
7*	0	20

Лабораторная работа 1: Изучение простых преобразований изображений

Цель работы: изучить алгоритмы и реализовать программу выполняющую простые преобразования серых и цветных изображений в формате PNM.

Описание:

Программа должна поддерживать серые и цветные изображения (варианты PNM P5 и P6), самостоятельно определяя формат по содержимому.

Аргументы программе передаются через командную строку:

lab#.exe <имя_входного_файла> <имя_выходного_файла> <преобразование>

где <преобразование>:

0 - инверсия,

1 - зеркальное отражение по горизонтали,

2 - зеркальное отражение по вертикали,

3 - поворот на 90 градусов по часовой стрелке,

4 - поворот на 90 градусов против часовой стрелки.

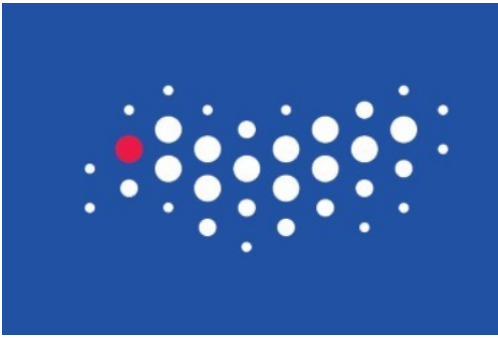
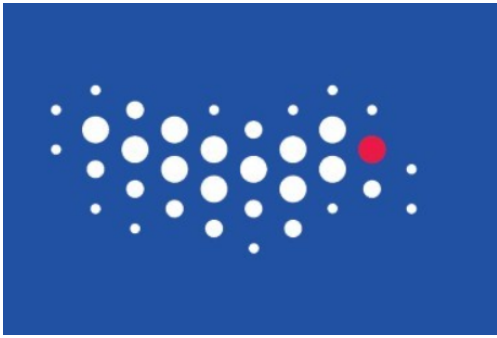
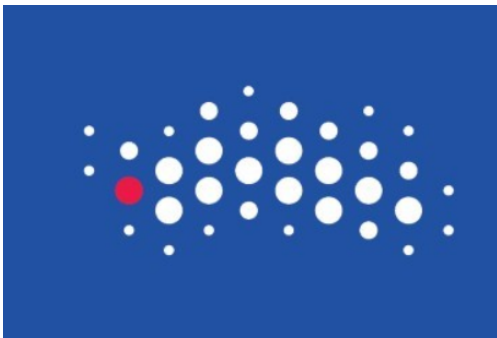
Программа должна быть написана на C/C++ и не использовать внешние библиотеки.

Частичное решение: работают преобразования 0-2; имена файлов и преобразование, возможно, написаны в исходном коде или читаются с консоли, а не берутся из командной строки.

Полное решение: всё работает + корректно выделяется и освобождается память, закрываются файлы, есть обработка ошибок: не удалось открыть файл, формат файла не поддерживается, не удалось выделить память.

Частая ошибка №1: путаются местами отражения по горизонтали и вертикали: отражение по горизонтали означает отзеркаливание по строкам (относительно центрального

столбца изображения), а не отзеркаливание относительно горизонтали (центральной строки изображения). Примеры в таблице ниже

	До преобразования	После преобразования
По горизонтали		
По вертикали		

Частая ошибка №2: если ваша прога попала в исключение и вы даже об этом говорите в консоль, то создается пустой файл с именем выходного (который передавался через аргументы командной строки). Это в принципе странно, ибо быть такого не должно, раз уж мы поругались в консоль, что все плохо..)

Лабораторная работа 2: Изучение алгоритмов отрисовки растровых линий с применением сглаживания и гамма-коррекции

Цель работы: изучить алгоритмы и реализовать программу, рисующую линию на изображении в формате PGM (P5) с учетом гамма-коррекции sRGB.

Описание:

Программа должна быть написана на C/C++ и не использовать внешние библиотеки.

Аргументы передаются через командную строку:

program.exe <имя_входного_файла> <имя_выходного_файла> <яркость_линии>
<толщина_линии> <x_начальный> <y_начальный> <x_конечный> <y_конечный>
<гамма>

где

- <яркость_линии>: целое число 0..255;
- <толщина_линии>: положительное дробное число;
- <x,y>: координаты внутри изображения, (0;0) соответствует левому верхнему углу, дробные числа (целые значения соответствуют центру пикселей).
- <гамма>: (optional) положительное вещественное число: гамма-коррекция с введенным значением в качестве гаммы. При его отсутствии используется sRGB.

Частичное решение: <толщина линии>=1, <гамма>=2.0, координаты начала и конца – целые числа, чёрный фон вместо данных исходного файла (размеры берутся из исходного файла).

Полное решение: всё работает (гамма + sRGB, толщина не только равная 1, фон из входного изображения) + корректно выделяется и освобождается память, закрываются файлы, есть обработка ошибок.

Если программе передано значение, которое не поддерживается – следует сообщить об ошибке.

Коды возврата:

0 - ошибок нет

1 - произошла ошибка

В поток вывода ничего не выводится (printf, cout).

Сообщения об ошибках выводятся в поток вывода ошибок:

C: fprintf(stderr, "Error\n");

C++: std::cerr

Следующие параметры гарантировано не будут выходить за обусловленные значения:

- <яркость_линии> = целое число 0..255;
- <толщина_линии> = положительное вещественное число;
- width и height в файле - положительные целые значения;
- яркостных данных в файле ровно width * height;
- <x_начальный> <x_конечный> = [0..width];
- <y_начальный> <y_конечный> = [0..height];

Лабораторная работа 3: Изучение алгоритмов псевдотонирования изображений

Цель работы: изучить алгоритмы и реализовать программу, применяющий алгоритм дизеринга к изображению в формате PGM (P5) с учетом гамма-коррекции.

Описание:

Программа должна быть написана на C/C++ и не использовать внешние библиотеки.

Аргументы передаются через командную строку:

**program.exe <имя_входного_файла> <имя_выходного_файла> <градиент>
<дизеринг> <битность> <гамма>**

где

- <имя_входного_файла>, <имя_выходного_файла>: формат файлов: PGM P5; ширина и высота берутся из <имя_входного_файла>;
- <градиент>: 0 - используем входную картинку, 1 - рисуем горизонтальный градиент (0-255) (ширина и высота берутся из <имя_входного_файла>);
- <дизеринг> - алгоритм дизеринга:
 - 0 – Нет дизеринга;
 - 1 – Ordered (8x8);
 - 2 – Random;
 - 3 – Floyd–Steinberg;
 - 4 – Jarvis, Judice, Ninke;
 - 5 - Sierra (Sierra-3);
 - 6 - Atkinson;
 - 7 - Halftone (4x4, orthogonal);

- <битность> - битность результата дitherинга (1..8);
- <гамма>: 0 - sRGB гамма, иначе - обычная гамма с указанным значением.

Частичное решение:

- <градиент> = 1;
- <дизеринг> = 0..3;
- <битность> = 1..8;
- <гамма> = 1 (аналогично отсутствию гамма-коррекции)

+ корректно выделяется и освобождается память, закрываются файлы, есть обработка ошибок.

Полное решение: все остальное

Примеры преобразований 2 (random) и 3 (floyd-steinberg) при разных значениях гаммы и битностей: [dither](#)

Если программе передано значение, которое не поддерживается – следует сообщить об ошибке.

Коды возврата:

0 - ошибок нет

1 - произошла ошибка

В поток вывода ничего не выводится (printf, cout).

Сообщения об ошибках выводятся в поток вывода ошибок:

C: fprintf(stderr, "Error\n");

C++: std::cerr

Следующие параметры гарантировано не будут выходить за обусловленные значения:

- <градиент> = 0 или 1;
- <битность> = 1..8;
- width и height в файле - положительные целые значения;
- яркостных данных в файле ровно width * height;
- <гамма> - вещественная неотрицательная;

Лабораторная работа 4: Изучение цветовых пространств

Лекции: <https://vk.cc/asAPE1> и <https://vk.cc/asAPGB>

Цель работы: реализовать программу, которая позволяет проводить преобразования между цветовыми пространствами.

Входные и выходные данные могут быть как одним файлом ppm, так и набором из 3 ppm.

Описание:

Программа должна быть написана на C/C++ и не использовать внешние библиотеки.

Аргументы передаются через командную строку:

lab4.exe -f <from_color_space> -t <to_color_space> -i <count> <input_file_name> -o <count> <output_file_name> ,

где

- <color_space> - RGB / HSL / HSV / YCbCr.601 / YCbCr.709 / YCoCg / CMY
- <count> - 1 или 3
- <file_name>:

- для count=1 просто имя файла; формат ppm
- для count=3 шаблон имени вида <name.ext>, что соответствует файлам <name_1.ext>, <name_2.ext> и <name_3.ext> для каждого канала соответственно; формат pgm

Порядок аргументов (-f, -t, -i, -o) может быть произвольным.
Везде 8-битные данные и полный диапазон (**0..255, PC range**).

Полное решение: всё работает + корректно выделяется и освобождается память, закрываются файлы, есть обработка ошибок.
/* да, частичного решения нет */

Если программе передано значение, которое не поддерживается – следует сообщить об ошибке.

Коды возврата:
0 - ошибок нет
1 - произошла ошибка

В поток вывода ничего не выводится (printf, cout).
Сообщения об ошибках выводятся в поток вывода ошибок:
C: fprintf(stderr, "Error\n");
C++: std::cerr

Следующие параметры гарантировано не будут выходить за обусловленные значения:

- <count> = 1 или 3;
- width и height в файле - положительные целые значения;
- яркостных данных в файле ровно width * height;

Лабораторная работа 5: Изучение алгоритма настройки автояркости изображения

Лекция: <https://vk.cc/aulmUE>

Цель работы: реализовать программу, которая позволяет проводить настройку автояркости изображения в различных цветовых пространствах.

Описание:

Программа должна быть написана на C/C++ и не использовать внешние библиотеки.

Аргументы передаются через командную строку:

lab5.exe <имя_входного_файла> <имя_выходного_файла> <преобразование> [<смещение> <множитель>],

где

- <преобразование>:
0 - применить указанные значения <смещение> и <множитель> в пространстве RGB к каждому каналу;
1 - применить указанные значения <смещение> и <множитель> в пространстве YCbCr.601 к каналу Y;
2 - автояркость в пространстве RGB: <смещение> и <множитель> вычисляются на основе минимального и максимального значений пикселей;
3 - аналогично 2 в пространстве YCbCr.601;

4 - автояркость в пространстве RGB: <смещение> и <множитель> вычисляются на основе минимального и максимального значений пикселей, после игнорирования 0.39% самых светлых и тёмных пикселей;
5 - аналогично 4 в пространстве YCbCr.601.

- <смещение> - целое число, только для преобразований 0 и 1 в диапазоне [-255..255];
- <множитель> - дробное положительное число, только для преобразований 0 и 1 в диапазоне [1/255..255].

Значение пикселя X изменяется по формуле: $(X - \text{<смещение>}) * \text{<множитель>}$.
YCbCr.601 в PC диапазоне: [0, 255].

Входные/выходные данные: PNM P5 или P6 (RGB).

Частичное решение: только преобразования 0-3 + корректно выделяется и освобождается память, закрываются файлы, есть обработка ошибок.

Полное решение: все остальное.

Если программе передано значение, которое не поддерживается – следует сообщить об ошибке.

Коды возврата:

0 - ошибок нет

1 - произошла ошибка

В поток вывода (printf, cout) выводится только следующая информация: для преобразований 2-5 найденные значения <смещение> и <множитель> в формате: "<смещение> <множитель>".

Сообщения об ошибках выводятся в поток вывода ошибок:

C: fprintf(stderr, "Error\n");

C++: std::cerr

Лабораторная работа 6: Изучение алгоритмов масштабирования изображений

Цель работы: реализовать программу, которая позволяет проводить масштабирование изображений.

Описание: Программа должна быть написана на C/C++ и не использовать внешние библиотеки.

Аргументы передаются через командную строку:

lab6.exe <input> <output> <width> <height> <dx> <dy> <gamma> <type> [<C>],

где:

- <input>, <output> - имена входного и выходного файлов в формате PNM P5 или P6.
- <width>, <height> - ширина и высота результирующего изображения, натуральные числа.

- `<dx>`, `<dy>` - смещение центра результата относительно центра исходного изображения, вещественные числа в единицах результирующего изображения.
- `<gamma>` - гамма-коррекция (0.0 = sRGB).
- `<type>` - способ масштабирования:
 - 0 – ближайшая точка (метод ближайшего соседа)
 - 1 – билинейное
 - 2 – Lanczos3
 - 3 – BC-сплайны. Для этого способа могут быть указаны ещё два параметра: B и C, по умолчанию 0 и 0.5 (Catmull-Rom).

Входные/выходные данные: PNM P5 или P6 (RGB).

Частичное решение:

- только серые файлы (P5),
- только на увеличение (масштаб $\geq 100\%$),
- гамма не учитывается (равна 1.0),
- смещения не учитываются, левый верхний угол результирующего изображения совпадает с исходным.

Полное решение: все остальное.

Если программе передано значение, которое не поддерживается – следует сообщить об ошибке.

Коды возврата:

0 - ошибок нет

1 - произошла ошибка

В поток вывода ничего не выводится (printf, cout).

Сообщения об ошибках выводятся в поток вывода ошибок:

C: `fprintf(stderr, "Error\n");`

C++: `std::cerr`

Подробнее про BC-сплайны:

1. <https://de.wikipedia.org/wiki/Mitchell-Netravali-Filter>
2. http://mentallandscape.com/Papers_siggraph88.pdf

Лабораторная работа 7: Изучение алгоритма повышения резкости

Цель работы: изучить алгоритм повышения резкости Contrast Adaptive Sharpening (с better diagonals, без масштабирования).

Описание:

Программа должна быть написана на C/C++ и не использовать внешние библиотеки.

Важно: Помимо реализации будет оцениваться изложение теории, представленной в отчете. Без раскрытия теоретического материала решение засчитано не будет.

Аргументы передаются через командную строку:

lab7.exe <input> <output> <sharpen>,

где sharpen - параметр резкости в диапазоне [0..1] (вещественное значение).

Входные/выходные данные: PNM P5 или P6 (RGB).

Полное решение: всё работает + корректно выделяется и освобождается память, закрываются файлы, есть обработка ошибок.

/* да, частичного решения нет */

Если программе передано значение, которое не поддерживается – следует сообщить об ошибке.

Коды возврата:

0 - ошибок нет

1 - произошла ошибка

В поток вывода ничего не выводится (printf, cout).

Сообщения об ошибках выводятся в поток вывода ошибок:

C: fprintf(stderr, "Error\n");

C++: std::cerr