Задачи:

1. **Ссылки 1:** Написать функцию `void swap(int& x, int& y)`, которая меняет значение двух переменных, используя ссылки. Вызовите эту функцию в функции main().

2. **header файлы:** Оформите весь код, описывающий класс Complex в виде одного заголовочного(header) файла. Подключите его к вашему .cpp файлу. Используйте стражи включений, чтобы избежать ошибки множественного включения.

3. **Ссылки 2:** Написать функцию `void normalize_complex(Complex& z)`, которая нормализует комплексное число z.

4. **Метод класса:** Написать метод класса Complex `void normalize()`.

5. **Конструкторы и деструкторы:** добавьте в соответствующие конструктор и деструктор класса Complex вывод на экран сообщений "Constructor" и "Destructor" соответственно. Создайте экземпляр класса Complex на стеке вызовов в функции main() и запустите программу. Создайте массив из 10-ти элементов класса Complex и запустите программу.

6. **new/delete:** Используёте malloc() и free() чтобы выделить память под экземпляр класса Complex в куче. Используйте операторы new и delete, чтобы создать экземпляр класса Complex в куче. Запустите программу. В чём различие между malloc() и new?

Задачи. Класс Image:

1. **Класс Image:** В папке `seminar05_class/code/` лежит код с описанием класса Image для работы с изображениями в формате .ppm. Изучите этот код и скомпилируйте его. Запустите получившуюся программу.

2. **Flip vertically:** Перепишите программу так, чтобы она загружала изображение и отзрекаливала его по вертикали. Используйте уже написанную функцию `filp_vertically()`.

3. **Sepia:** Добавьте функцию `void sepia()`, которая добавляет эффект сепии, изменяя цвета следующим образом:

$$\begin{aligned} r' &= 0.393 * r + 0.769 * g + 0.189 * b \\ g' &= 0.349 * r + 0.686 * g + 0.168 * b \\ b' &= 0.272 * r + 0.534 * g + 0.131 * b \end{aligned} \tag{1}$$

   Примените этот эффект на различные изображения из папки images.

4. **Новый конструктор:** Напишите новый конструктор `Image(int n, int m, Pixel color = Pixel(0, 0, 0))`, который будет создавать изображение цвета color и размера n на m.

5. **Set pixel:** Напишите методы класса `void set_pixel(int i, int j, unsigned char r0, unsigned char g0, unsigned char b0)` и `void set_pixel(int i, int j, Pixel color)`, которые устанавливают пиксель под координатами i и j в соответствующий цвет.

6. **Get pixel:** Напишите метод класса `Pixel get_pixel(int i, int j)`, которая возвращает цвет пикселя под координатами i и j.

7. **Белый шум:** Используйте новый конструктор `Image(int n, int m)` и функцию `set_pixel`, чтобы создавать изображение, каждый пиксель которого будет иметь случайный цвет. Случайное число от 0 до 255 можно получить, используя `rand() % 256` из библиотеки stdlib.h. Сохраните это изображение.

8. **Флаг Японии:** Используйте новый конструктор `Image(int n, int m)` и функцию `set_pixel`, чтобы создавать изображение флага Японии. Размеры 600x400 пикселей. Радиус круга – 100 пикселей.

9. **Задача об убегающей точке:** Предположим, что у нас есть комплексная функция $f(z) = z^2$. Выберем некоторое комплексное число z_0 и будем проводить следующие итерации: $z_1 = f(z_0), z_2 = f(z_1), ..., z_{n+1} = f(z_n)$. В зависимости от выбора точки z_0 эта последовательность либо разойдётся, либо останется в некоторой ограниченной области. Нужно найти все точки комплексной плоскости, которые не являются убегающими.

   Для функции $f(z) = z^2$ эта область тривиальна, но всё становится сложней для функции вида $f(z) = z^2 + c$, где c – некоторое комплексное число. Численно найдите область убегания для функций такого вида. Для этого создайте изображение размера 1000x1000, покрывающую область [-2:2]x[-2:2] на комплексной плоскости. Для каждой точки этой плоскости проведите N итераций и, в зависимости от результата, окрасьте пиксель в соответствующий цвет (цвет можно подобрать самим). Используйте классы Complex и Image

   Добавьте параметры командной строки: 2 вещественных числа, соответствующие комплексному числу c, и целое число итераций N. Программа должна создавать файл `julia.ppm`.

10. **Множество:** Зафиксируем теперь $z_0 = 0$ и будем менять c. Численно найдите все параметры c, для которых точка не является убегающей.