**Глава 1:**

Идеята на проекта е да се създаде редактор за изображения, който да поддържа няколко формата. Едно изображение трябва да може да се обработва по различни начини- ротиране, смяна на цвета и други. Самите изображения трябва да се четат от файл, като по този начин се извлича информацията за тях. Освен това, програмата трябва да работи със сесии, които поддържат масив от изображения и прилагат манипулации върху тях, както и с масив от тези сесии.

Задачите на разработка са да се създаде един родител от тип „Изображение” и негови наследници, които имат различни специфики. След това трябва да се създаде клас „Сесия”, който да работи със сответния масив от „Изображения”. И като последна стъпка трябва да бъде разработен „манипулаторът” за тези сесии;

За да може да се осъществи работа с изображенията, трябва да бъдат разработени функции за четене от файл и за записване, за да могат да бъдат записани съответните промени, стига потребителят да иска това. Други функции, които трябва да бъдат направени са за самите промени върху изображнието, както и такива, които да запазват стария му вид, за да може той да бъде възстановен, ако потребителят иска това.

За да се работи с класа, който осъществява сесията, се създава функция, която добавя изображение и други функции, изпозлващи тези на класа „Изображение”, за да променят всичко в сесията.

Класът с масив от сесии е разработен по аналогичен начин. Неговите функции използват тези на класа „Сесия”.

Целта на разработката е да осъществи изискваните задачи по правилен и с добра практика начин, по който да се постигне възможно най добра функционалност на проекта.

**Глава 2:**

Най-важни за кода са реализиране на функциите за четене от файла и записване с него, както и тези отговарящи,за промяната на външния вид на изображението – ротации, промяна на цветно или сиво изображение в черно-бяло, промяна на цветно и черно в сиво. Ключова част е че се работи с три вида изображения, но всеки от тях има своя версия и в двоичен, и в текстов файл.

Поставената задача не е твърде сложна, инетресна е изисква търсене на допълнителна информцаия специално за това, как се предствая едно изображение.

**Глава 3:**

3.1

Класовете са стандартни, с private, public, protected член-данни, функции за копиране и триене на и реализирани големи четворки, където е нужно. В проекта е създаден клас Image, който е родител на 6 наследника, за които са направени необходимите реализации. След това се създава създава Session с хетерогенен контейнер от тип Image. След него се реализира класът с масив от сесии и като последедн остава интерфейсът.

3.2

За класа Image – има абстрактни функции.

Код как една от тях е реализирана в .ppm наследника:

Grayscale: changeCopy();

const float r = 0.299F;

const float g = 0.587F;

const float b = 0.114F;

for (int i = 0; i < height; i++) {

for (int j = 0; j < width; j++) {

pixels[i][j].r = pixels[i][j].r \* r + pixels[i][j].g \* g + pixels[i][j].b \* b;

pixels[i][j].g = pixels[i][j].r \* r + pixels[i][j].g \* g + pixels[i][j].b \* b;

pixels[i][j].b = pixels[i][j].r \* r + pixels[i][j].g \* g + pixels[i][j].b \* b;

}

}

* Този формат се характерзира от три числа, описващи един пиксел и за да се постигне нужната промяна се умножава всяко число по дадена константа

3.3

За класа Session:

void Session::addImage(const char\* fileName) {

addToOperationArr(6);

char\* format = readFrFileFormat(fileName);

if (count == capacity) {

resize();

}

if (strcmp(format, "P6")==0) {

Image\* newObj = new PpmPlane(fileName);

newObj->open();

images[count++] = newObj->clone();

}

else if (strcmp(format, "P3")==0) {

Image\* newObj = new Ppm(fileName);

newObj->open();

images[count++] = newObj->clone();

}

else if (strcmp(format, "P2")==0) {

Image\* newObj = new Pgm(fileName);

newObj->open();

images[count++] = newObj->clone();

}

else if (strcmp(format, "P5")==0) {

Image\* newObj = new PlainPgm(fileName);

newObj->open();

images[count++] = newObj->clone();

}

else if (strcmp(format, "P1")==0) {

Image\* newObj = new Pbm(fileName);

newObj->open();

images[count++] = newObj->clone();

}

else if (strcmp(format, "P4")==0) {

Image\* newObj = new PlainPbm(fileName);

newObj->open();

images[count++] = newObj->clone();

}

std::cout << "Image" << " " << fileName << " added\n";

delete[]format;

}

-добавя изображение в зависимост от формата като го чете от подаден файл.

**Глава 4:**

Реализация на класове:

И трите основни класа имат голяма четврока и аналогични функции съдържащи се една в друга.

**Глава 5:**

По-голямата част от началните цели се изпълняват от написания проект;