**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Шадринский государственный педагогический университет»

Факультет информатики математики и физики

Кафедра программирования и автоматизации бизнес-процессов

«**Разработка на языке программирования Си утилиты обработки изображения в формате PPM P6**»

Курсовая работа по дисциплине «Проектный практикум»

направление подготовки 09.03.01 Информатика и вычислительная техника

образовательная программа « Программное обеспечение вычислительной техники и автоматизированных систем»

Квалификация - бакалавр

**Выполнил**:

Пономарев М.С.

Студент 330Б группы

очная форма обучения

**Научный Руководитель**:

Канд. Пед. Наук, доцент

Слинкин Д.А

“\_\_”\_\_\_\_\_\_\_\_\_\_\_\_2024 г.

оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Содержание

Оглавление

[Содержание 2](#_Toc168995908)

[Введение 3](#_Toc168995909)

[Теоретическая часть 4](#_Toc168995910)

[1. Формат PPM P6. Общая характеристика 4](#_Toc168995911)

[Плюсы формата PPM. 5](#_Toc168995912)

[Минусы формата PPM. 6](#_Toc168995913)

[2. Функции обработки изображения и методы реализации. 6](#_Toc168995914)

[Практическая часть 9](#_Toc168995915)

[1. Существующие программы. 9](#_Toc168995916)

[2. Этапы реализации утилиты 10](#_Toc168995917)

[2.1 Считывание и загрузка изображения 11](#_Toc168995918)

[2.2 Применение функций обработки. 12](#_Toc168995919)

[2.3 Сохранение изображения 15](#_Toc168995920)

[3. Взаимодействие пользователя с утилитой 15](#_Toc168995921)

[3.1 Взаимодействие через командную строку 15](#_Toc168995922)

[3.2 Взаимодействие через интерактивное меню 19](#_Toc168995923)

[Заключение 22](#_Toc168995924)

[Список используемой литературы 23](#_Toc168995925)

[Приложение 24](#_Toc168995926)

[image\_redakt.h 24](#_Toc168995927)

[image\_redakt.c 25](#_Toc168995928)

[redakt.c 31](#_Toc168995929)

# Введение

Редактор - это программа, предназначенная для изменения и обработки различных типов файлов или контента. Они стали неотъемлемой частью многих сфер, где требуется работа с текстом, графикой, аудио- и видеоматериалами. В данной курсовой работе рассматривается редактор изображений в формате PPM P6.

**Актуальность:** обработка изображений становится всё более востребованной. С увеличением объема графической информации возрастает потребность в программах, которые способны эффективно её обработать. Утилита на языке программирования СИ может быть полезной как для опытных разработчиков, так и для студентов.

**Цель:** Разработать утилиту для обработки изображения, которое будет иметь следующие функции: поворот изображения, масштабирование черно-белый фильтр, кадрирование.

**Задачи:**

1. Изучить спецификации формата изображений PPM P6.
2. Определить требуемые функции для редактирования изображения.
3. Реализовать рабочую программу на языке СИ со всеми требуемыми функциями редактирования изображений.
4. Протестировать и проверить работу всех функций.

# Теоретическая часть

## Формат PPM P6. Общая характеристика

Формат PPM (Portable Pixmap) представляет собой простой и широко поддерживаемый формат для хранения растровых изображений, который позволяет осуществлять быструю и эффективную обработку изображений без потерь качества. Файлы формата PPM делятся на два типа: P3 (ASCII-версия, текстовый формат) и P6 (бинарный формат). Оба этих формата содержат один и тот же набор информации но различаются способом представления этих данных.

P3 (текстовый) формат представляет данные каждого пикселя в виде ASCII-чисел, что занимает больше места и обрабатывается дольше из-за необходимости преобразования текста в двоичные данные в процессе чтения файла. P6 (бинарный) формат представляет цвет каждого пикселя в виде последовательности бинарных чисел, что уменьшает размер файла и ускоряет чтение и запись данных за счёт отсутствия промежуточных преобразований.

Благодаря простой структуре формата PPM файлы могут быть легко созданы и обработаны на любой платформе с помощью стандартных средств программирования, что делает их удобными для транспортировки между различными системами и программным обеспечением.

Также этот формат используется в академических и обучающих целях для изучения обработки изображений.

В отличие от современных форматов, которые используют сложное сжатие данных для уменьшения размера файла, PPM может привести к созданию относительно больших файлов при высоком разрешении изображения.

В целом, формат PPM P6 выступает в качестве идеального кандидата для операций, где время обработки и прямой доступ к данным изображения являются приоритетными, а размер файла и поддержка сжатия не являются основными ограничениями.

Файл в формате PPM P6 состоит из нескольких основных сегментов:

Магическое число: Первая строка файла начинается с символов P6, указывающих на то, что это файл формата PPM, версии P6.

Размеры изображения: Следующая строка указывает размеры изображения, сначала ширину в пикселях, затем высоту, разделенные пробелом.

Максимальное значение цвета: Третья строка определяет максимальное значение для компонентов цвета, обычно это значение равно 255. Эта величина позволяет определить, как интерпретировать последующие данные о цвете.

Данные о пикселях: После заголовка следуют данные о пикселях, начиная с верхнего левого угла. Каждый пиксель представлен тремя байтами для каналов RGB.

### Плюсы формата PPM.

* **Простота формата:** Одним из главных преимуществ формата PPM является его простота. Это дает возможность легко понять и реализовать поддержку формата как в новом, так и в существующем программном обеспечении.
* **Быстрая обработка данных:** Благодаря отсутствию сжатия, работы с данными в формате PPM производятся быстро, поскольку не требуется время на декомпрессию и повторное сжатие при обработке.
* **Универсальность:** Файлы PPM могут быть созданы и обработаны на любой платформе, снимая ограничения на использование в различных операционных системах и обеспечивая обмен данными между разнообразным программным обеспечением.
* **Прямой доступ к пикселям:** Формат позволяет легко получить доступ к каждому пикселю изображения для анализа или модификации, что делает его полезным для обучения и академических исследований по обработке и анализу изображений.
* **Поддержка высокого разрешения:** PPM поддерживает изображения с высоким разрешением без введения дополнительной сложности в структуру файла.

### Минусы формата PPM.

* **Размер файла:** Важным недостатком PPM является его тенденция к созданию больших файлов из-за отсутствия сжатия. Это может ограничивать его использование для очень больших изображений или при работе с ограниченным дисковым пространством.
* **Отсутствие поддержки прозрачности:** PPM не поддерживает прозрачность изображений, что ограничивает его применение в некоторых графических задачах, где требуется альфа-канал.
* **Ограниченная поддержка:** Хотя формат является универсальным, он не всегда непосредственно поддерживается в популярном графическом и редакторском ПО без установки дополнительных плагинов или конвертации в другие форматы.
* Безопасность: Простота формата делает его потенциально уязвимым для подмены данных или выполнения вредоносного кода, если процесс обработки не соответствует определенным мерам предосторожности.

## Функции обработки изображения и методы реализации.

**Кадрирование.**

Выделение интересующей области изображения и удаление ненужных краев.

Пользователь или программа определяет координаты (x, y) левого верхнего и правого нижнего углов области, которую необходимо сохранить. После этого из изображения удаляются все пиксели за пределами указанной области.

**Масштабирование.**

Изменение размера изображения без потери важных деталей или искажения пропорций.

Методы масштабирования:

-Ближайший сосед. Это самый простой и быстрый метод, который выбирает значение ближайшего пикселя к данной точке и применяет это значение для нового пикселя.

-Билинейная интерполяция. Учитывает ближайшие 2x2 пикселя и на основании их создает новый пиксель, линейно интерполируя значения по обоим осям. Этот метод лучше сохраняет плавность переходов между цветами, чем ближайший сосед, и подходит для умеренного масштабирования.

-Бикубическая интерполяция. Еще более продвинутый метод, который использует 16 ближайших пикселей для создания нового, интерполируя значения в обеих осях кубическим способом. Бикубическая интерполяция обеспечивает еще более гладкие градиенты и переходы между цветами, но требует больше вычислительных ресурсов, чем билинейная интерполяция.

**Поворот на 90 градусов.**

Изменение ориентации изображения на 90 градусов.

Простая операция реализуемая перестановкой пикселей.

**Черно-белый фильтр.**

Преобразование изображения в градации серого, что позволяет сосредоточить внимание на текстуре и форме объектов.

-Простое усреднение. Метод вычисляет среднее значение трех цветовых компонентов для каждого пикселя и присваивает это среднее значение каждому из цветовых каналов, делая пиксель оттенком серого.

-Линейное взвешенное преобразование. Принимает во внимание разные вклады цветов к общему восприятию яркости человеческим глазом. Этот метод ближе к тому, как человеческий глаз видит цвет.

-Метод оттенков. Сочетает в себе методы усреднения и использования яркостных коэффициентов. Он вычисляет разность между самым высоким и самым низким значениями из трех цветов и делит эту разность на 2, чтобы добавить её к усреднённому значению.

-Палитра максимального декодирования. При этом алгоритме для каждого пикселя используется только самая высокая (максимальное декодирование) или самая низкая (минимальное декодирование) из трех цветовых компонентов, что создает более контрастные черно-белые изображения.

-Одноканальный метод. Можно выбрать один цветной канал (красный, зеленый или синий) и использовать его значения для создания черно-белого изображения. Этот метод подчеркивает информацию в выбранном канале, но теряет информацию в других.

**Эффект размытия**

Преобразование изображения в менее детализированное.

-Простое размытие

Использует простую форму ядра, где все значения равны между собой. Процесс включает замену каждого пикселя усредненным значением его соседей в определенном радиусе.

-Гауссово размытие Применяет ядро, основанное на функции Гаусса, для достижения более мягкого и естественного эффекта размытия. Центральные пиксели имеют больший вес по сравнению с крайними, что обеспечивает плавный переход.

-Двунаправленное размытие

Учитывает как пространственные (расстояние между пикселями), так и радиометрические (различие в интенсивности) различия, сохраняя при этом острые границы. Используется для шумоподавления, сохраняя детали краев.

-Срединное размытие

Заменяет каждый пиксель медианным значением из его окрестностей. Эффективно удаляет шум типа "соль и перец", сохраняя границы объектов.

**Эффект негатива**

Эффект при котором цвета изображения инвертируются.

-Поэлементное инвертирование:

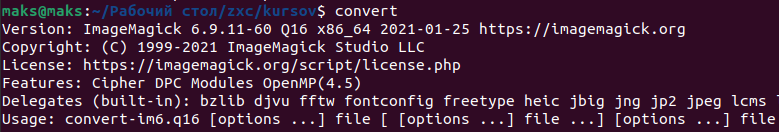
Этот подход подразумевает обработку отдельных пикселей изображения, где новое значение каждого цветового компонента пикселя (красного, зеленого, и синего) высчитывается как 255 минус текущее значение этого компонента.

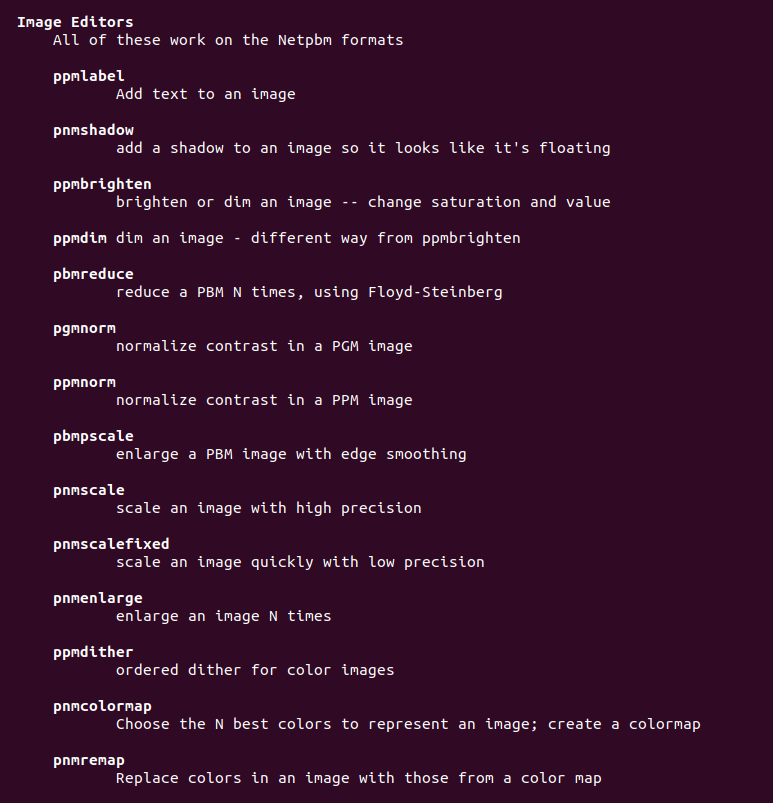
# Практическая часть

## Существующие программы.

**ImageMagick**

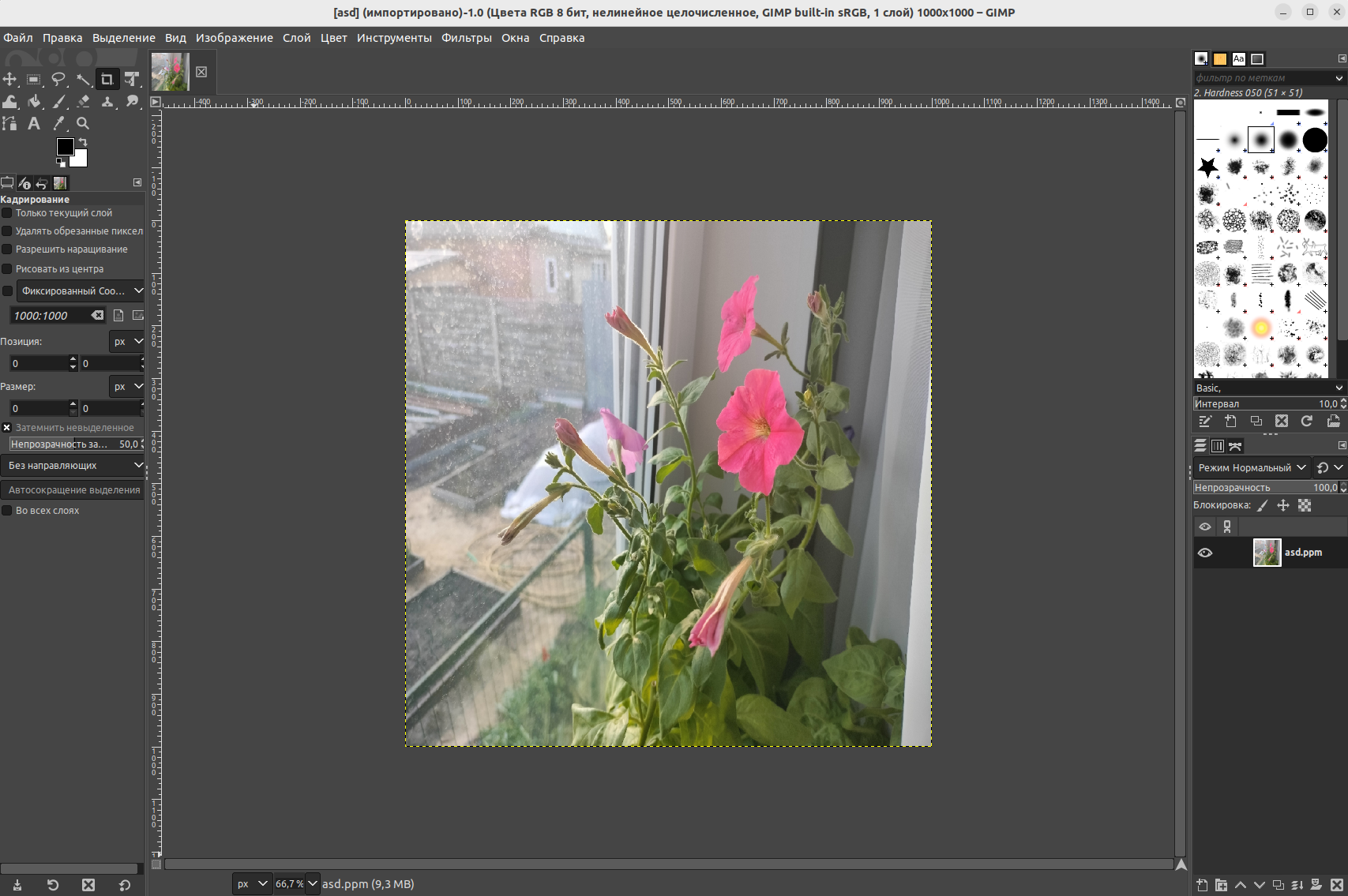
ImageMagick предоставляет широкий спектр возможностей для обработки изображений и поддерживает множество графических форматов. Она может быть использована для масштабирования, преобразования, наложения текста, вращения и многих других операций с помощью командной строки.

**Netpbm**

Netpbm – это набор простых утилит командной строки, которые могут использоваться для обработки изображений. Они могут конвертировать между различными графическими форматами, а также выполнять простые операции обработки, такие как масштабирование, поворот и изменение цветов.

**GIMP**

GIMP – это свободно распространяемый инструмент для таких задач, как ретуширование фотографий, композиция изображений и создание изображений. GIMP поддерживает автоматизацию с использованием сценариев на Scheme, Python и других скриптовых языках.

****

## Этапы реализации утилиты

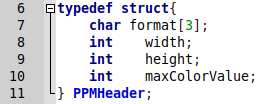
1. Считывание и загрузка изображения.

2. Применение функций обработки.

3. Сохранение изображения.

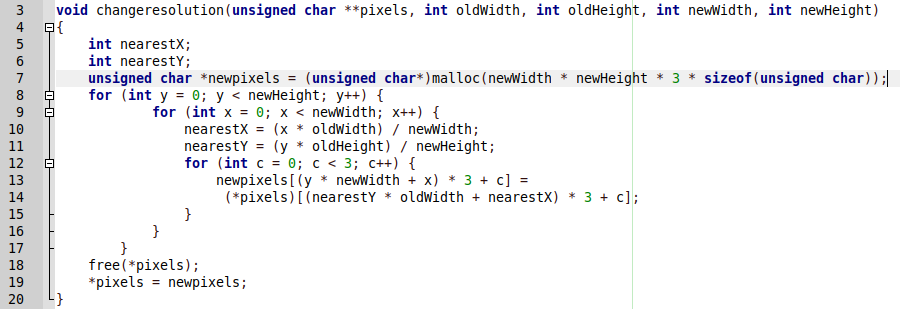
4. Взаимодействие пользователя с утилитой.

### Считывание и загрузка изображения

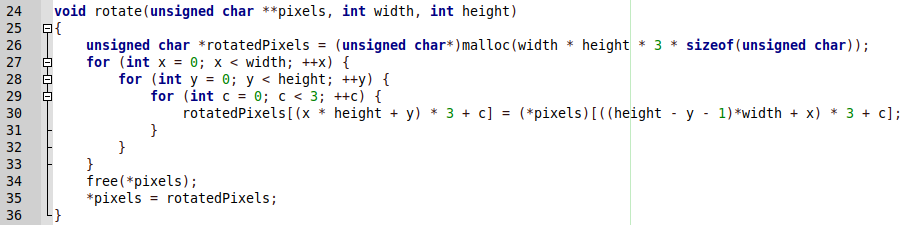
С помощью функции readimage считываются метаданные, которые записываются в структуру PPMHeader, и пиксели изображения.

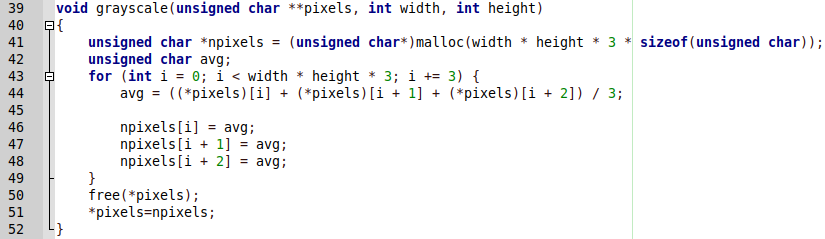
### Применение функций обработки.

**Фунция сhangeresolution**

Изменяет разрешение изображения с использованием метода ближайшего соседа.

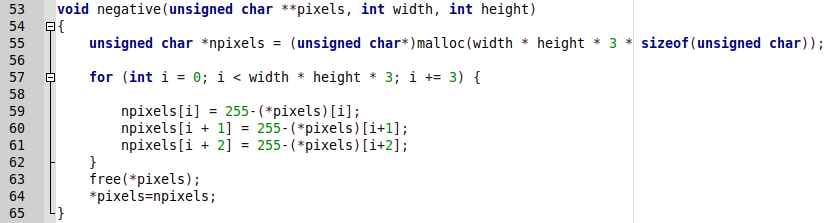
**Функция rotate**

Функция поворачивает изображение на 90 градусов по часовой стрелке.

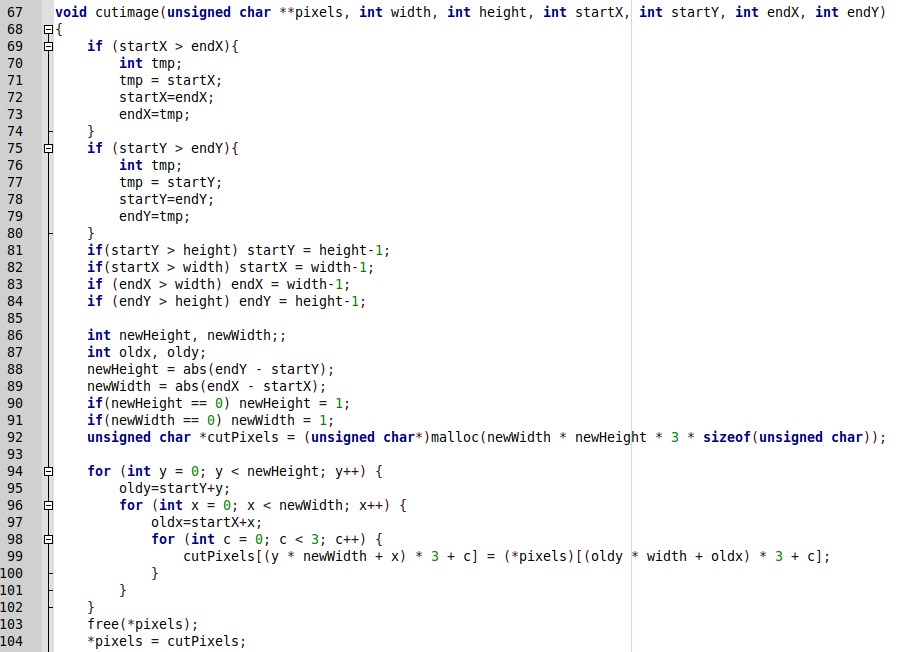
**Функция grayscale**

Функция преобразует цветное изображение в оттенки серого. Для каждого пикселя вычисляет среднее значение его цветовых компонент (RGB), заменяя ими исходные значения.

**Фунцкия negative**

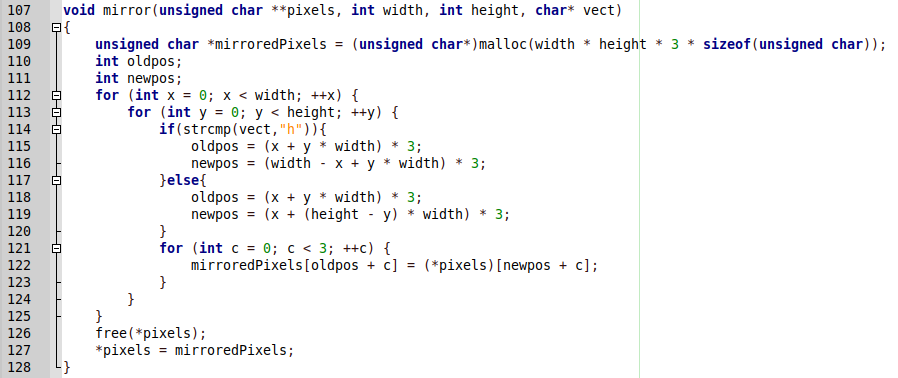
****

Эта функция создает негатив изображения. Для каждого пикселя вычисляет его негатив, вычитая его значение из 255.

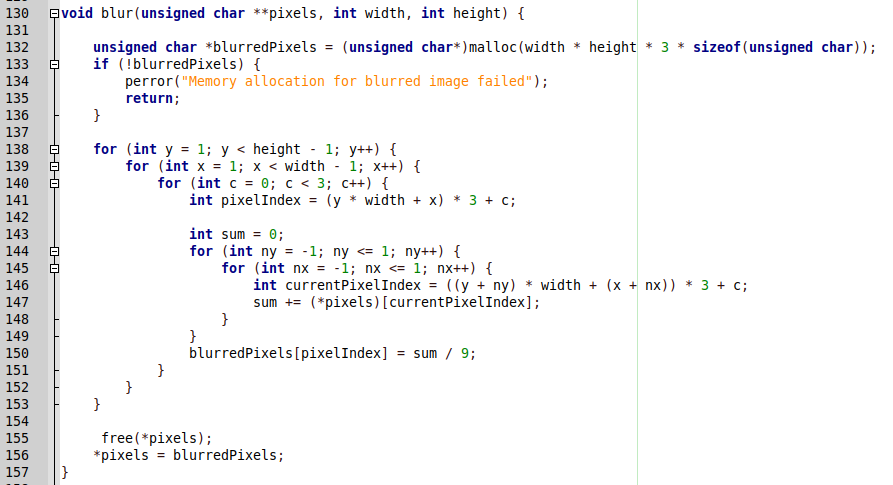
**Функция cutimmage**

Эта функция вырезает участок из изображения. Она принимает координаты начальной и конечной точек вырезаемого участка, пересчитывает координаты и копирует пиксели из старого массива в новый.

**Функция mirror**

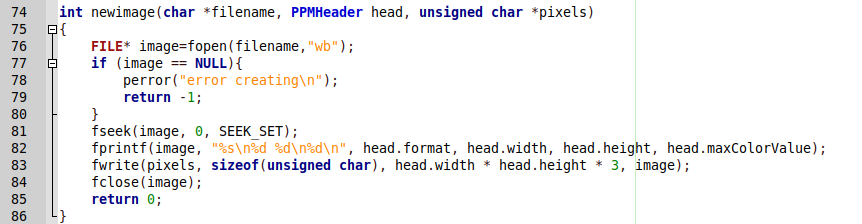
Эта функция зеркально отражает изображение. В зависимости от переданного параметра (vect), она может отразить изображение по горизонтали или вертикали.

**Функция blur**

****

Эта функция применяет размытие к изображению с использованием фильтра среднего значения.

### Сохранение изображения

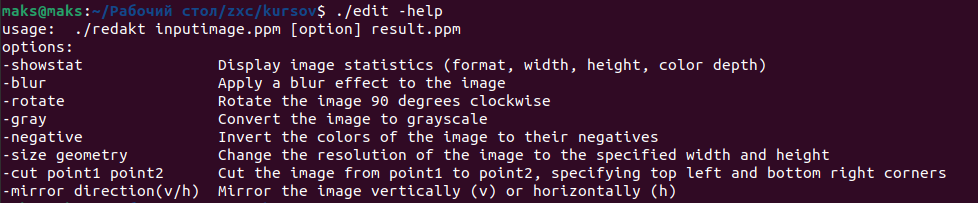
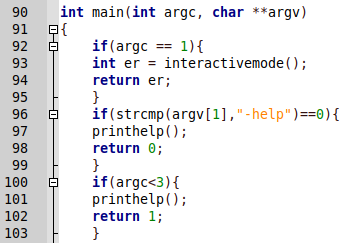
После применения функций обработки, изображение сохраняется при помощи функции newimage, которая принимает название файла, обновленный заголовок файла и обновленный массив пикселей

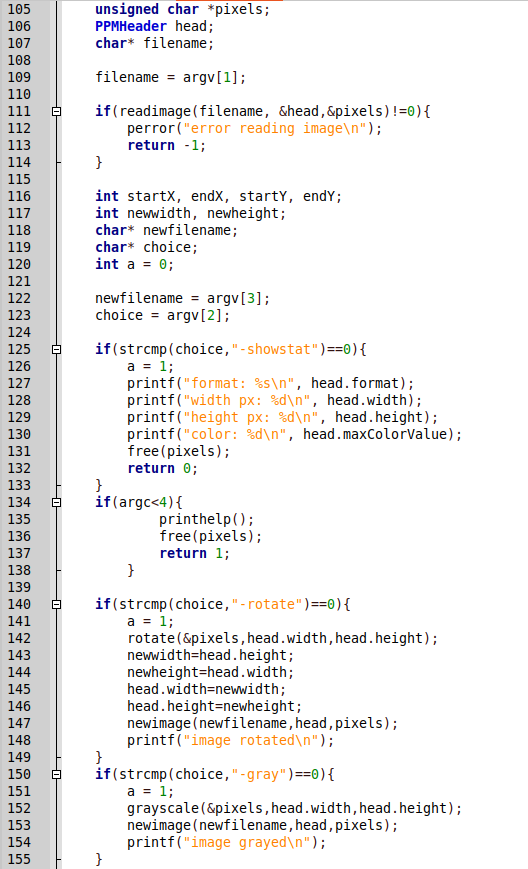
## Взаимодействие пользователя с утилитой

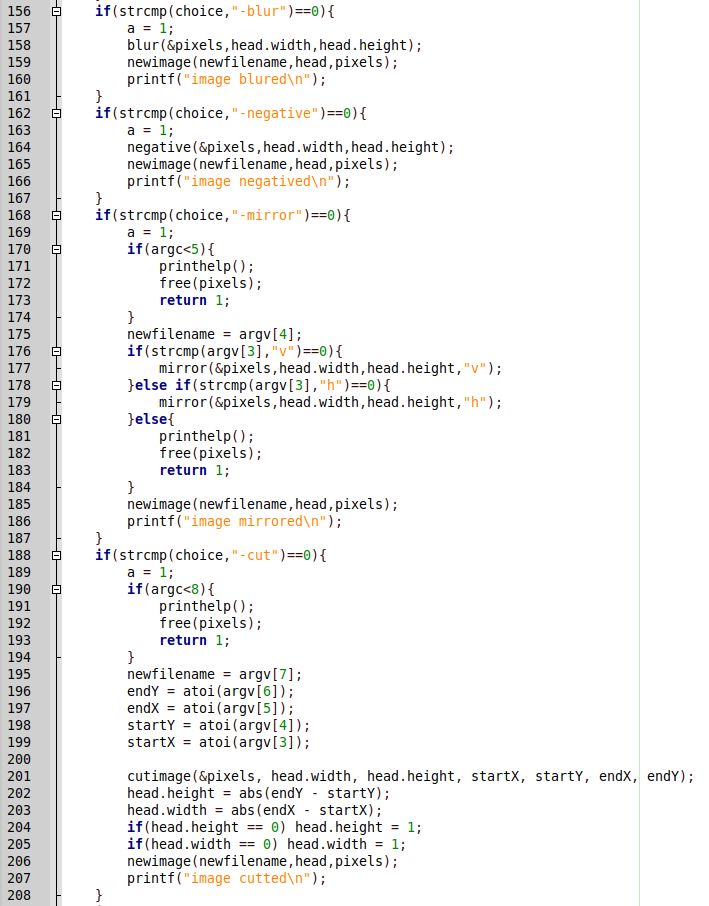
Для взаимодействия с утилитой предусмотрено два способа.

### Взаимодействие через командную строку

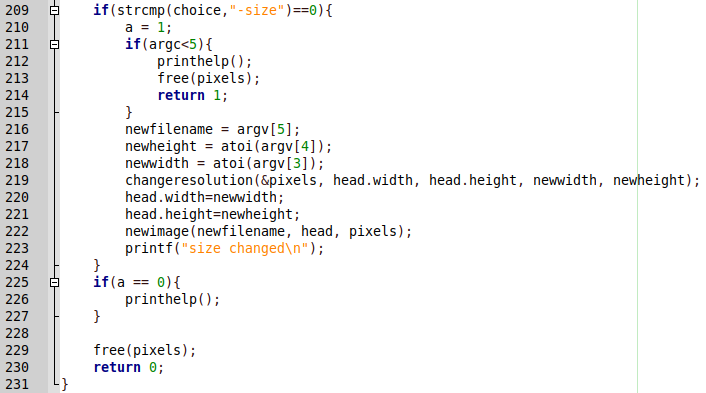
Для взаимодействия через командную строку пользователю надо ввести название картинки, которую он хочет редактировать, опцию и название результирующего файла.

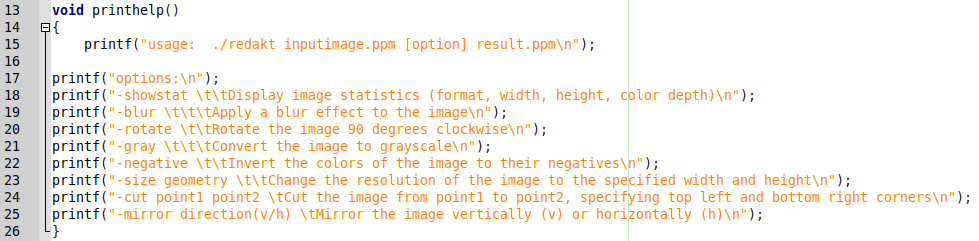


Переменная «a» нужна, чтобы определять сработала какая либо из опций или нет.

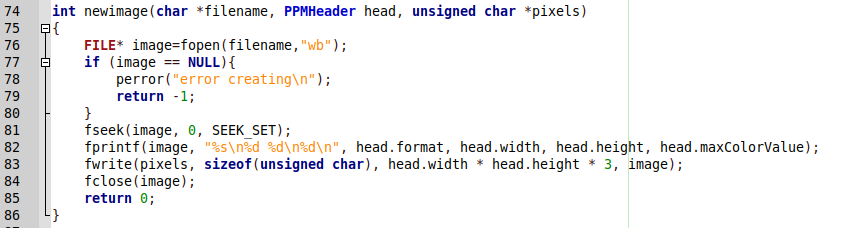


Функция atoi в строках 196-199 преобразуют строковые значения в int.

Если ни одна из функций не сработала и переменная «a» осталась равна нулю срабатывает функция printhelp.

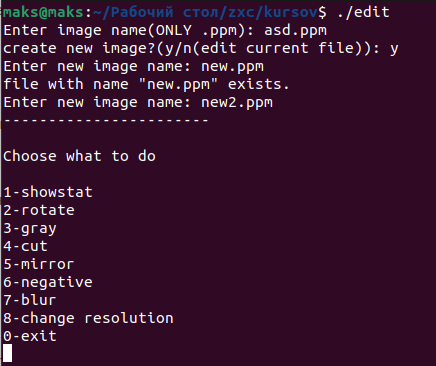
Также эта функция выводится в случае ввода неверных аргументов или если ввести опцию -help.

Функция newimage перезаписывает или создает новое изображение с обновленными данными.

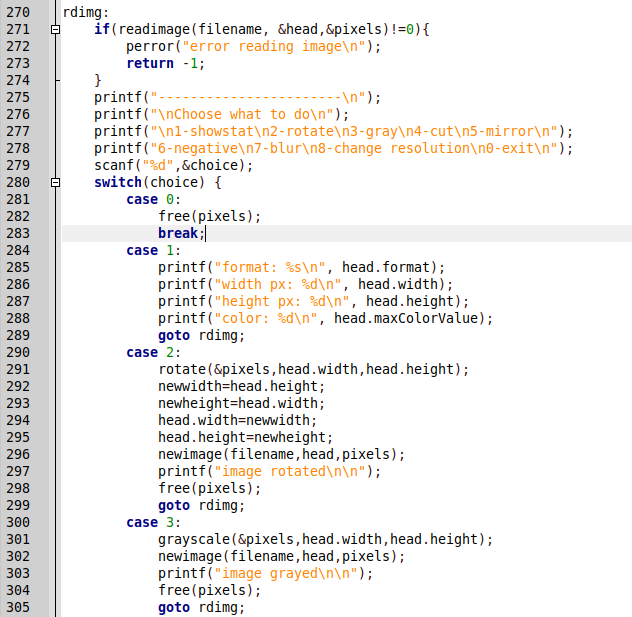


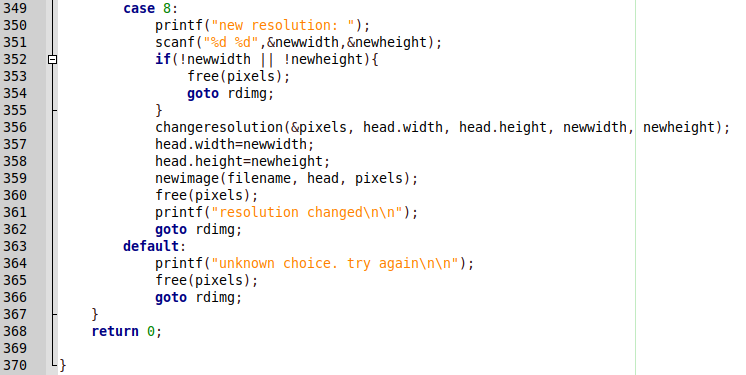
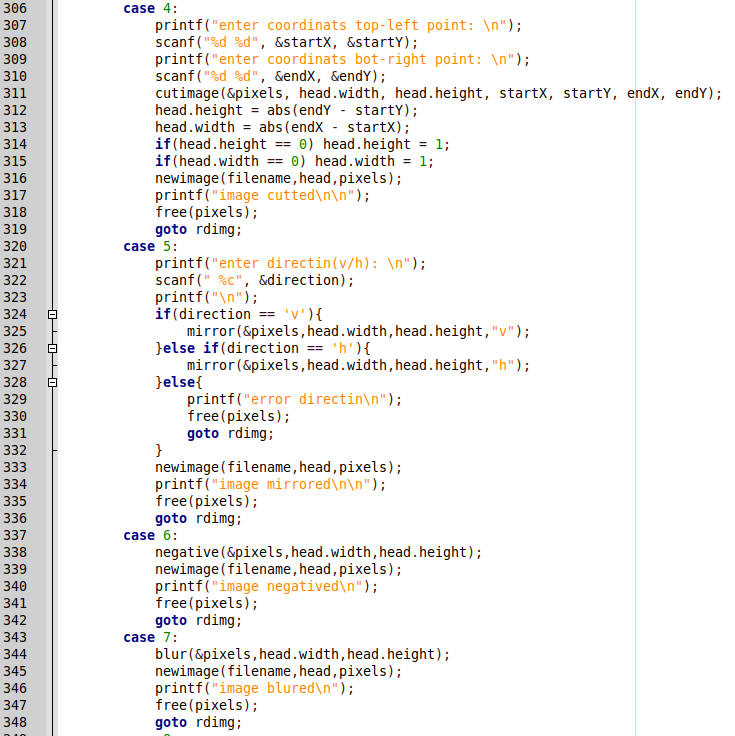
### Взаимодействие через интерактивное меню

В случае если программа запускается без агрументов, то утилита запустит интерактивное меню.

Все функции обработки изображения и функции чтения и создания изображений идентичны прошлому режиму. Отличие лишь во взаимодействии утилиты и пользователя.

В случае, когда пользователь выбрал создать новый файл, изначальный файл копируется с указанным именем и далее работа ведется именно с ним.

Точка rdimg позволяет программе вернуться на момент чтения изображения после редактирования либо после какой-либо ошибки.



# Заключение

В ходе работы успешно была реализована утилита для обработки изображения в формате PPM P6, были выполнены все поставленные цели и задачи.

Работа утилиты была протестирована успешно и она готова к использованию пользователями.

# Список используемой литературы

1. https://netpbm.sourceforge.net/doc/ppm.html
2. https://homepages.inf.ed.ac.uk/rbf/BOOKS/PHILLIPS/cips2ed.pdf
3. https://imagemagick.org/
4. https://netpbm.sourceforge.net/
5. <https://www.gimp.org/>

# Приложение

## image\_redakt.h

#ifndef IMAGE\_REDAKT\_H

#define IMAGE\_REDAKT\_H

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

// Объявления функций

void changeresolution(unsigned char \*\*pixels, int oldWidth, int oldHeight, int newWidth, int newHeight);

void rotate(unsigned char \*\*pixels, int width, int height);

void grayscale(unsigned char \*\*pixels, int width, int height);

void negative(unsigned char \*\*pixels, int width, int height);

void cutimage(unsigned char \*\*pixels, int width, int height, int startX, int startY, int endX, int endY);

void mirror(unsigned char \*\*pixels, int width, int height, char\* vect);

void blur(unsigned char \*\*pixels, int width, int height);

#endif // IMAGE\_REDAKT\_H

## image\_redakt.c

#include "image\_redakt.h"

void changeresolution(unsigned char \*\*pixels, int oldWidth, int oldHeight, int newWidth, int newHeight)

{

int nearestX;

int nearestY;

unsigned char \*newpixels = (unsigned char\*)malloc(newWidth \* newHeight \* 3 \* sizeof(unsigned char));

for (int y = 0; y < newHeight; y++) {

for (int x = 0; x < newWidth; x++) {

nearestX = (x \* oldWidth) / newWidth;

nearestY = (y \* oldHeight) / newHeight;

for (int c = 0; c < 3; c++) {

newpixels[(y \* newWidth + x) \* 3 + c] = (\*pixels)[(nearestY \* oldWidth + nearestX) \* 3 + c];

}

}

}

free(\*pixels);

\*pixels = newpixels;

}

void rotate(unsigned char \*\*pixels, int width, int height)

{

unsigned char \*rotatedPixels = (unsigned char\*)malloc(width \* height \* 3 \* sizeof(unsigned char));

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

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

for (int c = 0; c < 3; ++c) {

rotatedPixels[(x \* height + y) \* 3 + c] = (\*pixels)[((height - y - 1)\*width + x) \* 3 + c];

}

}

}

free(\*pixels);

\*pixels = rotatedPixels;

}

void grayscale(unsigned char \*\*pixels, int width, int height)

{

unsigned char \*npixels = (unsigned char\*)malloc(width \* height \* 3 \* sizeof(unsigned char));

unsigned char avg;

for (int i = 0; i < width \* height \* 3; i += 3) {

avg = ((\*pixels)[i] + (\*pixels)[i + 1] + (\*pixels)[i + 2]) / 3;

npixels[i] = avg;

npixels[i + 1] = avg;

npixels[i + 2] = avg;

}

free(\*pixels);

\*pixels=npixels;

}

void negative(unsigned char \*\*pixels, int width, int height)

{

unsigned char \*npixels = (unsigned char\*)malloc(width \* height \* 3 \* sizeof(unsigned char));

for (int i = 0; i < width \* height \* 3; i += 3) {

npixels[i] = 255-(\*pixels)[i];

npixels[i + 1] = 255-(\*pixels)[i+1];

npixels[i + 2] = 255-(\*pixels)[i+2];

}

free(\*pixels);

\*pixels=npixels;

}

void cutimage(unsigned char \*\*pixels, int width, int height, int startX, int startY, int endX, int endY)

{

if (startX > endX){

int tmp;

tmp = startX;

startX=endX;

endX=tmp;

}

if (startY > endY){

int tmp;

tmp = startY;

startY=endY;

endY=tmp;

}

if(startY > height) startY = height-1;

if(startX > width) startX = width-1;

if (endX > width) endX = width-1;

if (endY > height) endY = height-1;

int newHeight, newWidth;;

int oldx, oldy;

newHeight = abs(endY - startY);

newWidth = abs(endX - startX);

if(newHeight == 0) newHeight = 1;

if(newWidth == 0) newWidth = 1;

unsigned char \*cutPixels = (unsigned char\*)malloc(newWidth \* newHeight \* 3 \* sizeof(unsigned char));

for (int y = 0; y < newHeight; y++) {

oldy=startY+y;

for (int x = 0; x < newWidth; x++) {

oldx=startX+x;

for (int c = 0; c < 3; c++) {

cutPixels[(y \* newWidth + x) \* 3 + c] = (\*pixels)[(oldy \* width + oldx) \* 3 + c];

}

}

}

free(\*pixels);

\*pixels = cutPixels;

}

void mirror(unsigned char \*\*pixels, int width, int height, char\* vect)

{

unsigned char \*mirroredPixels = (unsigned char\*)malloc(width \* height \* 3 \* sizeof(unsigned char));

int oldpos;

int newpos;

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

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

if(strcmp(vect,"h")){

oldpos = (x + y \* width) \* 3;

newpos = (width - x + y \* width) \* 3;

}else{

oldpos = (x + y \* width) \* 3;

newpos = (x + (height - y) \* width) \* 3;

}

for (int c = 0; c < 3; ++c) {

mirroredPixels[oldpos + c] = (\*pixels)[newpos + c];

}

}

}

free(\*pixels);

\*pixels = mirroredPixels;

}

void blur(unsigned char \*\*pixels, int width, int height) {

unsigned char \*blurredPixels = (unsigned char\*)malloc(width \* height \* 3 \* sizeof(unsigned char));

if (!blurredPixels) {

perror("Memory allocation for blurred image failed");

return;

}

for (int y = 1; y < height - 1; y++) {

for (int x = 1; x < width - 1; x++) {

for (int c = 0; c < 3; c++) {

int pixelIndex = (y \* width + x) \* 3 + c;

int sum = 0;

for (int ny = -1; ny <= 1; ny++) {

for (int nx = -1; nx <= 1; nx++) {

int currentPixelIndex = ((y + ny) \* width + (x + nx)) \* 3 + c;

sum += (\*pixels)[currentPixelIndex];

}

}

blurredPixels[pixelIndex] = sum / 9;

}

}

}

free(\*pixels);

\*pixels = blurredPixels;

}

## redakt.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include "image\_redakt.h"

typedef struct{

char format[3];

int width;

int height;

int maxColorValue;

} PPMHeader;

void printhelp()

{

printf("usage: ./redakt inputimage.ppm [option] result.ppm\n");

printf("options:\n");

printf("-showstat \t\tDisplay image statistics (format, width, height, color depth)\n");

printf("-blur \t\t\tApply a blur effect to the image\n");

printf("-rotate \t\tRotate the image 90 degrees clockwise\n");

printf("-gray \t\t\tConvert the image to grayscale\n");

printf("-negative \t\tInvert the colors of the image to their negatives\n");

printf("-size geometry \t\tChange the resolution of the image to the specified width and height\n");

printf("-cut point1 point2 \tCut the image from point1 to point2, specifying top left and bottom right corners\n");

printf("-mirror direction(v/h) \tMirror the image vertically (v) or horizontally (h)\n");

}

int fileExists(const char \*filename) {

return access(filename, F\_OK) != -1;;

}

int readimage(char \*filename, PPMHeader \*head, unsigned char \*\*pixels)

{

FILE\* image = fopen(filename, "r+b");

if (image == NULL) {

return -1;

}

//Считывание магического числа и проверка на соответствие формата

fscanf(image, " %s", head->format);

if(strcmp(head->format,"P6")!=0){

fclose(image);

printf("error format.\n use format PPM P6\n");

return -1;

}

char c;

//Пропуск возможного комментария

fseek(image,1,SEEK\_CUR);

if((c = fgetc(image))=='#'){

while ((c = fgetc(image)) != EOF && c != '\n');

}else fseek(image,-1,SEEK\_CUR);

//Считывается разрешение картинки и максимальное значение цвета

fscanf(image, "%d %d", &head->width, &head->height);

fscanf(image, "%d", &head->maxColorValue);

//Выделение памяти под массив с пикселями

\*pixels = (unsigned char \*)malloc(head->width \* head->height \* 3 \* sizeof(unsigned char));

if (\*pixels == NULL) {

perror("Memory allocation failed");

fclose(image);

return -1;

}

//Считывание пикселей

fseek(image,+1,SEEK\_CUR);

fread(\*pixels, sizeof(unsigned char), head->width \* head->height \* 3, image);

fclose(image);

return 0;

}

int newimage(char \*filename, PPMHeader head, unsigned char \*pixels)

{

FILE\* image=fopen(filename,"wb");

if (image == NULL){

perror("error creating\n");

return -1;

}

fseek(image, 0, SEEK\_SET);

fprintf(image, "%s\n%d %d\n%d\n", head.format, head.width, head.height, head.maxColorValue);

fwrite(pixels, sizeof(unsigned char), head.width \* head.height \* 3, image);

fclose(image);

return 0;

}

int interactivemode();

int main(int argc, char \*\*argv)

{

if(argc == 1){

int er = interactivemode();

return er;

}

if(strcmp(argv[1],"-help")==0){

printhelp();

return 0;

}

if(argc<3){

printhelp();

return 1;

}

unsigned char \*pixels;

PPMHeader head;

char\* filename;

filename = argv[1];

if(readimage(filename, &head,&pixels)!=0){

perror("error reading image\n");

return -1;

}

int startX, endX, startY, endY;

int newwidth, newheight;

char\* newfilename;

char\* choice;

int a = 0;

newfilename = argv[3];

choice = argv[2];

if(strcmp(choice,"-showstat")==0){

a = 1;

printf("format: %s\n", head.format);

printf("width px: %d\n", head.width);

printf("height px: %d\n", head.height);

printf("color: %d\n", head.maxColorValue);

free(pixels);

return 0;

}

if(argc<4){

printhelp();

free(pixels);

return 1;

}

if(strcmp(choice,"-rotate")==0){

a = 1;

rotate(&pixels,head.width,head.height);

newwidth=head.height;

newheight=head.width;

head.width=newwidth;

head.height=newheight;

newimage(newfilename,head,pixels);

printf("image rotated\n");

}

if(strcmp(choice,"-gray")==0){

a = 1;

grayscale(&pixels,head.width,head.height);

newimage(newfilename,head,pixels);

printf("image grayed\n");

}

if(strcmp(choice,"-blur")==0){

a = 1;

blur(&pixels,head.width,head.height);

newimage(newfilename,head,pixels);

printf("image blured\n");

}

if(strcmp(choice,"-negative")==0){

a = 1;

negative(&pixels,head.width,head.height);

newimage(newfilename,head,pixels);

printf("image negatived\n");

}

if(strcmp(choice,"-mirror")==0){

a = 1;

if(argc<5){

printhelp();

free(pixels);

return 1;

}

newfilename = argv[4];

if(strcmp(argv[3],"v")==0){

mirror(&pixels,head.width,head.height,"v");

}else if(strcmp(argv[3],"h")==0){

mirror(&pixels,head.width,head.height,"h");

}else{

printhelp();

free(pixels);

return 1;

}

newimage(newfilename,head,pixels);

printf("image mirrored\n");

}

if(strcmp(choice,"-cut")==0){

a = 1;

if(argc<8){

printhelp();

free(pixels);

return 1;

}

newfilename = argv[7];

endY = atoi(argv[6]);

endX = atoi(argv[5]);

startY = atoi(argv[4]);

startX = atoi(argv[3]);

cutimage(&pixels, head.width, head.height, startX, startY, endX, endY);

head.height = abs(endY - startY);

head.width = abs(endX - startX);

if(head.height == 0) head.height = 1;

if(head.width == 0) head.width = 1;

newimage(newfilename,head,pixels);

printf("image cutted\n");

}

if(strcmp(choice,"-size")==0){

a = 1;

if(argc<5){

printhelp();

free(pixels);

return 1;

}

newfilename = argv[5];

newheight = atoi(argv[4]);

newwidth = atoi(argv[3]);

changeresolution(&pixels, head.width, head.height, newwidth, newheight);

head.width=newwidth;

head.height=newheight;

newimage(newfilename, head, pixels);

printf("size changed\n");

}

if(a == 0){

printhelp();

}

free(pixels);

return 0;

}

int interactivemode()

{

int startX, endX, startY, endY;

int newwidth, newheight;

unsigned char \*pixels;

char newfilename[20];

char filename[20];

char direction;

PPMHeader head;

int choice=0;

char ch;

printf("Enter image name(ONLY .ppm): ");

scanf("%s", filename);

if(readimage(filename, &head,&pixels)!=0){

perror("error reading image\n");

return -1;

}

printf("create new image?(y/n(edit current file)): ");

scanf(" %c", &ch);

if(ch == 'y' || ch == 'Y'){

int created=0;

while(created==0){

printf("Enter new image name: ");

scanf("%s", newfilename);

if(fileExists(newfilename)){

printf("file with name \"%s\" exists.\n", newfilename);

}else{

created=1;

newimage(newfilename, head, pixels);

strcpy(filename, newfilename);

}

}

}

rdimg:

if(readimage(filename, &head,&pixels)!=0){

perror("error reading image\n");

return -1;

}

printf("-----------------------\n");

printf("\nChoose what to do\n");

printf("\n1-showstat\n2-rotate\n3-gray\n4-cut\n5-mirror\n");

printf("6-negative\n7-blur\n8-change resolution\n0-exit\n");

scanf("%d",&choice);

switch(choice) {

case 0:

free(pixels);

break;

case 1:

printf("format: %s\n", head.format);

printf("width px: %d\n", head.width);

printf("height px: %d\n", head.height);

printf("color: %d\n", head.maxColorValue);

goto rdimg;

case 2:

rotate(&pixels,head.width,head.height);

newwidth=head.height;

newheight=head.width;

head.width=newwidth;

head.height=newheight;

newimage(filename,head,pixels);

printf("image rotated\n\n");

free(pixels);

goto rdimg;

case 3:

grayscale(&pixels,head.width,head.height);

newimage(filename,head,pixels);

printf("image grayed\n\n");

free(pixels);

goto rdimg;

case 4:

printf("enter coordinats top-left point: \n");

scanf("%d %d", &startX, &startY);

printf("enter coordinats bot-right point: \n");

scanf("%d %d", &endX, &endY);

cutimage(&pixels, head.width, head.height, startX, startY, endX, endY);

head.height = abs(endY - startY);

head.width = abs(endX - startX);

if(head.height == 0) head.height = 1;

if(head.width == 0) head.width = 1;

newimage(filename,head,pixels);

printf("image cutted\n\n");

free(pixels);

goto rdimg;

case 5:

printf("enter directin(v/h): \n");

scanf(" %c", &direction);

printf("\n");

if(direction == 'v'){

mirror(&pixels,head.width,head.height,"v");

}else if(direction == 'h'){

mirror(&pixels,head.width,head.height,"h");

}else{

printf("error directin\n");

free(pixels);

goto rdimg;

}

newimage(filename,head,pixels);

printf("image mirrored\n\n");

free(pixels);

goto rdimg;

case 6:

negative(&pixels,head.width,head.height);

newimage(filename,head,pixels);

printf("image negatived\n");

free(pixels);

goto rdimg;

case 7:

blur(&pixels,head.width,head.height);

newimage(filename,head,pixels);

printf("image blured\n");

free(pixels);

goto rdimg;

case 8:

printf("new resolution: ");

scanf("%d %d",&newwidth,&newheight);

if(!newwidth || !newheight){

free(pixels);

goto rdimg;

}

changeresolution(&pixels, head.width, head.height, newwidth, newheight);

head.width=newwidth;

head.height=newheight;

newimage(filename, head, pixels);

free(pixels);

printf("resolution changed\n\n");

goto rdimg;

default:

printf("unknown choice. try again\n\n");

free(pixels);

goto rdimg;

}

return 0;

}