Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа № 3

«Бинарные файлы»

Проверил: Выполнил:

асс. каф. ЭВМ ст. гр. 250505

И.Г. Скиба Бакунович Н.А.

МИНСК 2023

# 1 Цель и задачи лабораторной работы

Цель работы – освоить принципы работы с бинарными файлами в языке С.

**Задача:** написать программу, реализующую простую обработку изображения:

1. Варианты обработки – перевод изображения в негатив, перевод изображения в черно-белые цвета.
2. Название файла для обработки должно вводиться пользователем.
3. Если файл не подходит для программы, программа должна сообщить об ошибке и корректно завершиться.
4. Достаточно обрабатывать только структуры типа BITMAPINFOHEADER, только 24-bit изображения.
5. Поиск самого непопулярного среди коротких слов (В).
6. **Не использовать** WINAPI, структуры описывать самостоятельно.
7. Работу с пикселями реализовать через структуры.
8. Реализовать медианную фильтрацию.
9. Реализовать гамма-коррекцию.
10. Параметры преобразований изображений задаются в клавиатуры.

# 2 Результаты выполнения лабораторной работы

На рисунке 1.1 приведена блок-схема алгоритма для выполнения задания

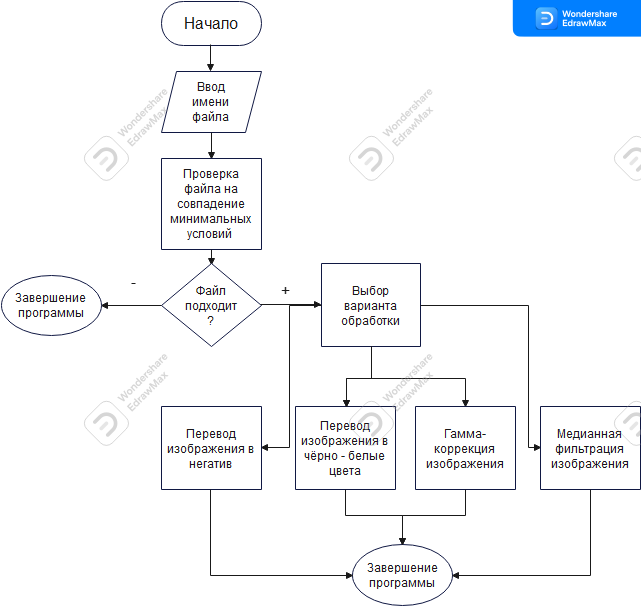


Рисунок 1.1 - Блок схема программы

**1.2** Листинг компьютерной программы

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include "Source.h"

#include <math.h>

#include <stdint.h>

int main()

{

FILE\* input\_file = fopen("korovka.bmp", "rb");

if (!input\_file) {

printf("Error: could not open input file.\n");

return -1;

}

BMPHeader header;

fread(&header, sizeof(header), 1, input\_file);

if (header.file\_header.type != 0x4D42) {

printf("Error: not a BMP file.\n");

fclose(input\_file);

return -1;

}

if (header.info\_header.bit\_count != 24) {

printf("Error: only 24-bit images are supported.\n");

fclose(input\_file);

return -1;

}

fseek(input\_file, header.file\_header.offset, SEEK\_SET);

int width = header.info\_header.width;

int height = header.info\_header.height;

Pixel\* pixels = malloc(width \* height \* sizeof(Pixel));

fread(pixels, sizeof(Pixel), width \* height, input\_file);

int chs = 0;

printf("Chose:\n");

printf("1. Negative\n");

printf("2. Black and white\n");

printf("3 --> Gammacorection\n");

input\_size(&chs);

switch (chs)

{

case 1:

{

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

pixels[i].red = 255 - pixels[i].red;

pixels[i].green = 255 - pixels[i].green;

pixels[i].blue = 255 - pixels[i].blue;

}

break;

}

case 2:

{

unsigned char average;

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

average = (pixels[i].red + pixels[i].green + pixels[i].blue) / 3;

pixels[i].red = average;

pixels[i].green = average;

pixels[i].blue = average;

}

break;

}

case 3:

{

float gamma = 0;

printf(":Input gamma cf:\n");

scanf("%f", &gamma);

printf("Gamma = %f\n", gamma);

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

pixels[i].red = (unsigned char)(pow((double)pixels[i].red / 255.0, gamma) \* 255.0 + 0.5);

pixels[i].green = (unsigned char)(pow((double)pixels[i].green / 255.0, gamma) \* 255.0 + 0.5);

pixels[i].blue = (unsigned char)(pow((double)pixels[i].blue / 255.0, gamma) \* 255.0 + 0.5);

}

Pixel\* pixels = malloc(width \* height \* sizeof(Pixel));

break;

}

}

FILE\* output\_file = fopen("output.bmp", "wb");

if (!output\_file) {

printf("Error: could not open output file.\n");

free(pixels);

fclose(input\_file);

return -1;

}

fwrite(&header, sizeof(header), 1, output\_file);

fseek(output\_file, header.file\_header.offset, SEEK\_SET);

fwrite(pixels, sizeof(Pixel), width \* height, output\_file);

fseek(output\_file, header.file\_header.offset, SEEK\_SET);

for (int y = height - 1; y >= 0; y--) {

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

Pixel\* p = &pixels[y \* width + x];

fread(p, sizeof(Pixel), 1, output\_file);

printf("\033[48;2;%d;%d;%dm ", p->red, p->green, p->blue);

}

printf("\033[0m\n");

fseek(output\_file, (4 - ((width \* 3) % 4)) % 4, SEEK\_CUR);

}

free(pixels);

fclose(input\_file);

fclose(output\_file);

return 0;

}

#pragma pack(push, 1)

typedef struct {

unsigned short type;

unsigned int size;

unsigned short reserved1;

unsigned short reserved2;

unsigned int offset;

} BMPFileHeader;

typedef struct {

unsigned int size;

int width;

int height;

unsigned short planes;

unsigned short bit\_count;

unsigned int compression;

unsigned int size\_image;

int x\_pels\_per\_meter;

int y\_pels\_per\_meter;

unsigned int clr\_used;

unsigned int clr\_important;

} BMPInfoHeader;

typedef struct {

BMPFileHeader file\_header;

BMPInfoHeader info\_header;

} BMPHeader;

typedef struct {

unsigned char blue;

unsigned char green;

unsigned char red;

} Pixel;

#pragma pack(pop)

void input\_size(int\* size);

void input\_size(int\* size)

{

long long a;

scanf\_s("%10lld", &a);

while (getchar() != '\n' || a < 1 || a > 3)

{

rewind(stdin);

printf("\nErorr: please try again:\n");

scanf\_s("%10lld", &a);

}

\*size = a;

}

**1.3** На рисунках 1.2 представлен скриншот выполнения программы

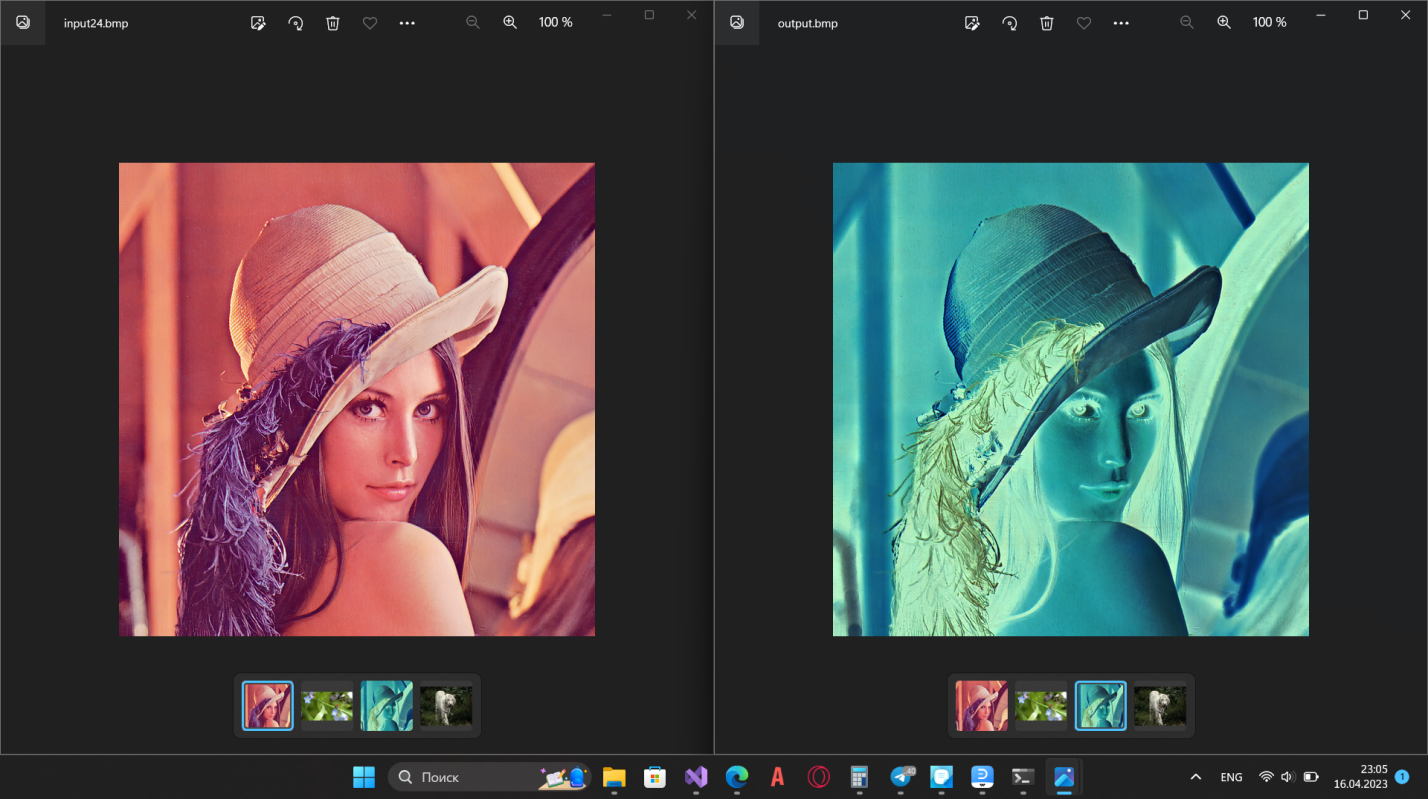


Рисунок 1.2 – Скрншот результата выполнения компьютерной программы