УДК 004

Создание чёрно-белого фильтра для изображения формата BMP на языке программирования C

Вихляев Дмитрий Романович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье содержится описание работы с 24 битным форматом BMP и создание чёрно-белого фильтра, реализованного с помощью языка программирования C. В результате исследования будет подробно описано создание программы и продемонстрирован результат её работы.
Ключевые слова: C, BMP, обработка изображения.

Creating a black-and-white filter for a BMP image in the C programming language

Vikhlyaev Dmitry Romanovich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article describes working with the 24-bit BMP format and creating a black-and-white filter implemented using the C programming language. As a result of the research, the creation of the program will be described in detail and the result of its work will be demonstrated.
Keywords: C, BMP, image processing.

1 Введение

1.1 Актуальность

Формат файла BMP способен хранить двумерные цифровые изображения, как монохромные, так и цветные, с различной глубиной цвета, альфа-каналами и цветовыми профилями. Является аппаратно-независимым растровым изображением (DIB), что разрешает перемещение растровых изображений с одного устройства на другое без дополнительных преобразований. Обычно формат используется для хранения изображений без сжатия, что означает отсутствие потерь. Несмотря на это некоторые стандарты имеют алгоритм сжатия RLE. Формат имеет более семи стандартов и включает глубину цвета от 1 до 64 бит на пиксель.

1.2Обзор исследований

З.В.Наймушина провела исследование обработки файлов в формате BMP [1]. В.А.Толстунови создал сглаживающий фильтр геометрического

среднего со степенными весами [2]. Е.В.Доманов провёл анализ качества работы фильтра Собеля на различных форматах изображения [3]. А.Д.Давлетов, Н.Г.Сокольвяк, А.С.Миронов описали алгоритм восстановления изображений с помощью фильтра Винера [4]. В.А.Шовин реализовал адаптивный фильтр изображений на базе нейронной сети [5].

1.3Цель исследования

Цель исследования – используя язык программирования С реализовать программу – чёрно-белый фильтр.

2 Материалы и методы

Для создания программы используется 24 битное растровое изображение формата BMP и язык программирования С.

3 Результаты и обсуждения

Файл BMP с точки зрения программиста представляет собой структурированный двоичный файл с прямым порядком байтов. Так как данный файл создавался под операционную систему windows, название типов данных соответствуют используемым в WinAPI, согласно документации Microsoft. (рис. 1).

```
typedef uint8_t  BYTE;
typedef uint32_t DWORD;
typedef int32_t  LONG;
typedef uint16_t WORD;
```

Рис. 1. Приведение типов языка С к WinAPI

Заголовок файла BMP (BITMAPFILEHEADER) имеет размер 14 байт и содержит информацию о типе и размере файла, а также о расположении в нем данных. С самого начала файл BMP обязан иметь 2 байта, символы «BM» в кодировке ASCII 0x4D42. Пиксельные данные могут находиться на произвольной позиции, об их нахождении указано поле OffBits. Следует обратить внимание на то, что с самого начала структуры сбивается выравнивание ячеек. В оперативной памяти данный заголовок распылается по чётным адресам, которые не кратны четырём, тогда 32-битные ячейки попадут на выровненные позиции (рис. 2).

```
typedef struct
{
    WORD   bfType;
    DWORD  bfSize;
    WORD   bfReserved1;
    WORD   bfReserved2;
    DWORD  bfOffBits;
} __attribute__((__packed__))
BITMAPFILEHEADER;
```

Рис. 2. Структура BITMAPFILEHEADER

Далее следует (BITMAPINFOHEADER), этот блок байтов сообщает приложению подробную информацию об изображении, которая будет использоваться для отображения изображения на экране. В разных версиях формата, данная структура имеет разную длину. В примере приведён основной заголовок доступный с 3-ей версии. Структура хранит: размер заголовка, ширина и высота изображения в пикселях, число плоскостей, глубина цвета, тип сжатия, размер изображения в байтах, горизонтальное и вертикальное разрешение, точки на метр, таблица цветов и число основных цветов. По стандарту сжатию поддаются изображения с 4 и 8-ми битами на пиксель. В данной статье будет использоваться пример с 24-мя битами на пиксель (рис. 3).

```
typedef struct
{
    DWORD  biSize;
    LONG   biWidth;
    LONG   biHeight;
    WORD   biPlanes;
    WORD   biBitCount;
    DWORD  biCompression;
    DWORD  biSizeImage;
    LONG   biXPelsPerMeter;
    LONG   biYPelsPerMeter;
    DWORD  biClrUsed;
    DWORD  biClrImportant;
} __attribute__((__packed__))
BITMAPINFOHEADER;
```

Рис. 3. Структура BITMAPINFOHEADER

Структура пикселей в 3 версии BMP и 24 битами на пиксель хранит 3 цвета из стандарта RGB в порядке обратном названию, по одному байту на цвет (рис.4).

```
typedef struct
{
    BYTE  rgbtBlue;
    BYTE  rgbtGreen;
    BYTE  rgbtRed;
} __attribute__((__packed__))
RGBTRIPLE;
```

Рис. 4. Структура одного пикселя

При фильтровании изменению подвергаются только данные пикселей, заголовки можно просто скопировать. Прежде чем начать работу предварительно нужно убедиться, что исходный файл является файлом BMP нужной версии (рис. 5).

```
// Read infile's BITMAPFILEHEADER
BITMAPFILEHEADER bf;
fread(&bf, sizeof(BITMAPFILEHEADER), 1, inptr);

// Read infile's BITMAPINFOHEADER
BITMAPINFOHEADER bi;
fread(&bi, sizeof(BITMAPINFOHEADER), 1, inptr);

// Ensure infile is (likely) a 24-bit uncompressed BMP 4.0
if (bf.bfType != 0x4d42 || bf.bfOffBits != 54 || bi.biSize != 40 ||
    bi.biBitCount != 24 || bi.biCompression != 0)
{
    fclose(outptr);
    fclose(inptr);
    fprintf(stderr, "Unsupported file format.\n");
    return 6;
}
```

Рис. 5. Получение заголовков и проверка на валидность файла

Далее, создаётся двумерный массив, в котором будут храниться данные пикселей. Размеры массива соответствуют количеству пикселей по ширине и высоте, указанной в заголовке. Для значения, количество пикселей по высоте, следует брать абсолютное значение, так как формат поддерживает хранения сверху вниз и снизу вверх, что приводит к смене знака.

Формат изображения BMP предполагает, что каждая строка данных будет выровнена по границе четырех байтов или дополнена нулями, если это не так. Для изображения с разрешением 32 бита на пиксель условие выравнивания всегда выполняется. В случае изображений с 24 битами на пиксель выравнивание выполняется только в том случае, если ширина изображения делится на 4, в противном случае необходимо самостоятельно заполнить строки нулями. Для вычисления размера отступа для каждой строки, используется формула, пример одной из разнообразий формулы представлена в коде (рис. 6).

```
int height = abs(bi.biHeight);
int width = bi.biWidth;

// Allocate memory for image
RGBTRIPLE(*image)[width] = calloc(height, width * sizeof(RGBTRIPLE));
if (image == NULL)
{
    fprintf(stderr, "Not enough memory to store image.\n");
    fclose(outptr);
    fclose(inptr);
    return 7;
}

// Determine padding for scanlines
int padding = (4 - (width * sizeof(RGBTRIPLE)) % 4) % 4;

// Iterate over infile's scanlines
for (int i = 0; i < height; i++)
{
    // Read row into pixel array
    fread(image[i], sizeof(RGBTRIPLE), width, inptr);

    // Skip over padding
    fseek(inptr, padding, SEEK_CUR);
}
```

Рис. 6. Запись массива пикселей в оперативную память

Создание чёрно-белого изображения часто подразумевает собой приведение цветовой палитры к оттенкам серого цвета. Одним из самых простых способов реализации данного эффекта, является взятие среднего арифметического от значений каждого из цветов: красного, синего и зеленного. Затем присвоить каждому цвету получившееся значение (рис 7).

```
void grayscale(int height, int width, RGBTRIPLE image[height][width])
{
    for(int i=0;i<height;i++)
    {
        for(int j=0;j<width;j++)
        {
           BYTE gray =(image[i][j].rgbtBlue+image[i][j].rgbtGreen+image[i][j].rgbtRed)/3;
           image[i][j].rgbtBlue=gray;
           image[i][j].rgbtGreen=gray;
           image[i][j].rgbtRed=gray;
        }
    }
    return;
}
```

Рис. 7. Преобразование цветной в чёрно-белую палитру

После чего остаётся переписать заголовки, отфильтрованный массив пикселей и добавить отступы в новый файл (рис.8,9,10).

```
// Write outfile's BITMAPFILEHEADER
fwrite(&bf, sizeof(BITMAPFILEHEADER), 1, outptr);

// Write outfile's BITMAPINFOHEADER
fwrite(&bi, sizeof(BITMAPINFOHEADER), 1, outptr);

// Write new pixels to outfile
for (int i = 0; i < height; i++)
{
    // Write row to outfile
    fwrite(image[i], sizeof(RGBTRIPLE), width, outptr);

    // Write padding at end of row
    for (int k = 0; k < padding; k++)
    {
        fputc(0x00, outptr);
    }
}
```

Рис. 8. Запись отфильтрованных данных в новый файл

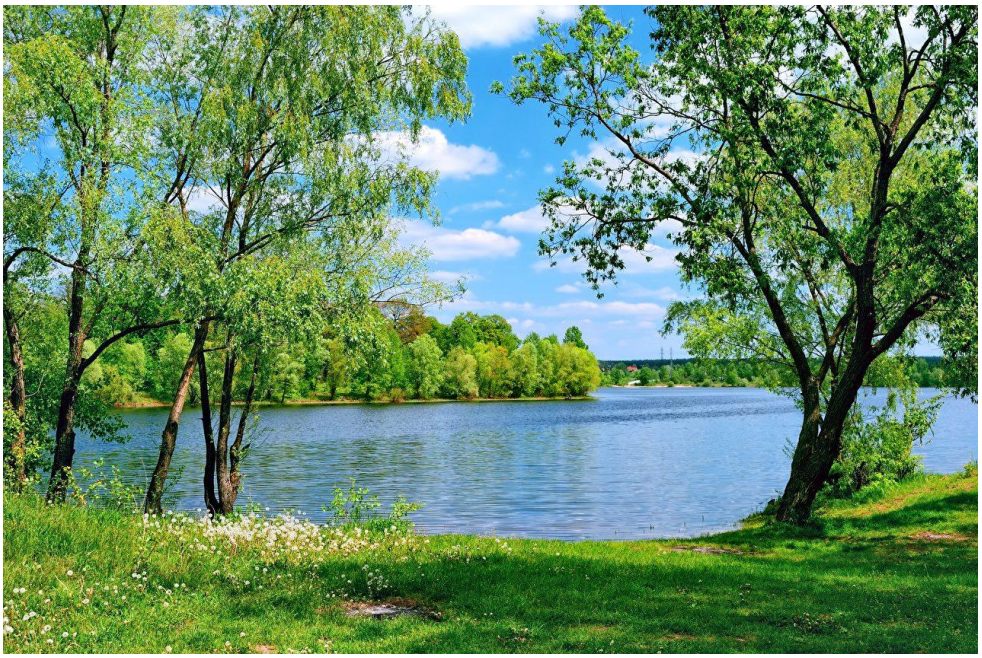

Рис. 9. Исходное изображение формата BMP

Рис. 10. Изображение, полученное после наложения фильтра

Выводы

Таким образом, было описано низкоуровневое содержание файла BMP и реализация фильтра, преобразующего полноцветное 24 битное растровое изображение в соответствующее чёрно-белое изображение.

Библиографический список

1. Наймушина З.В. Обработка файлов в формате BMP // Мир современной науки. 2021. № 4 (68). С. 11-19.
2. Толстунов В.А. Сглаживающий фильтр геометрического среднего со степенными весами // Актуальные проблемы гуманитарных и естественных наук. 2014. № 4-1. С. 107-112.
3. Доманов Е.В. Анализ качества работы фильтра Собеля на различных форматах изображения // Научный альманах. 2016. № 11-2 (25). С. 308-311.
4. Давлетов А.Д., Сокольвяк Н.Г., Миронов А.С. Восстановление изображений с помощью фильтра Винера // В сборнике: Информационные технологии XXI века. Сборник научный трудов. Отв. ред. В.В. Воронин. Хабаровск, 2018. С. 159-163.
5. Шовин В.А. Адаптивный фильтр изображений на базе нейронной сети // В книге: Математическое и компьютерное моделирование. Сборник материалов VI Международной научной конференции, посвященной памяти Б.А. Рогозина. 2018. С. 106-107.
6. Червяков Н.И., Ляхов П.А., Калита Д.И., Попова Н.В. О математических моделях фильтров для цифровой обработки изображений // В сборнике: Основные направления развития научного потенциала в свете современных исследований: теория и практика. материалы одиннадцатой

международной заочной научной конференции. Северо-Кавказский федеральный университет. 2017. С. 238-241.
7. Каменева А.Е., Горбунова А.В., Артамонов А.А. Пространственная фильтрация изображения на примере сглаживающего фильтра // В сборнике: Научное сообщество студентов. Сборник материалов X Международной студенческой научно-практической конференции. 2016. С. 119-121.