Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Сибирский государственный университет телекоммуникаций и
информатики»

(СибГУТИ)

Кафедра прикладной математики и кибернетики

Лабораторная работа №2
по курсу программирование графических процессов

Выполнили:
студенты группы ИП-014

Гулая А.С.
Обухов А.И.
Малышев В.А.

Работу проверил:
доцент каф. ПМиК
Перцев И.В.

Новосибирск 2024 г.

Задание:

Преобразовать BMP файл, создав вокруг него рамку из пикселей рандомных цветов. Ширина рамки - 15 пикселей. Количество цветов – 256 и TrueColor. Изменить соответствующие поля в заголовке и сохранить файл под новым именем. Длина строки BMP файла выравнивается по 32-битовой границе, (4-мбайт), при необходимости к каждой строке в файле добавляются выравнивающие байты.

Листинг программы:

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>

#pragma pack(1)
typedef struct {
    uint16_t signature;
    uint32_t filesize;
    uint32_t reserved;
    uint32_t offset;
    uint32_t header_size;
    uint32_t width;
    uint32_t height;
    uint16_t planes;
    uint16_t bpp;
    uint32_t compression;
    uint32_t image_size;
    uint32_t x_pixels_per_m;
    uint32_t y_pixels_per_m;
    uint32_t colors_used;
    uint32_t colors_important;
} Head;

typedef struct {
    Head head;
    uint8_t* rastr;
    uint8_t* palette;
} Image;

void read_head(Head* head, FILE* file) { fread(head, sizeof(Head), 1,
file); }

Image read_image(FILE* file)
{
    Image image = { 0 };

    read_head(&image.head, file);

    fseek(file, sizeof(image.head), SEEK_SET);

    if (image.head.bpp <= 8) {
        int colors_number = 1 << image.head.bpp;
        image.palette = malloc(colors_number * 4 * sizeof(uint8_t));
        fread(image.palette, sizeof(uint8_t), colors_number * 4, file);
    }
```

```
    image.rastr = malloc(image.head.filesize);
    int bytes_per_pixel = image.head.bpp / 8;
    int row_size = image.head.width * bytes_per_pixel;

    uint32_t padding = (4 - image.head.width * bytes_per_pixel % 4) %
4;
    for (int i = 0; i < image.head.height; i++) {
        fread(&image.rastr[i * row_size], sizeof(uint8_t), row_size,
file);
        fseek(file, padding, SEEK_CUR);
    }

    fread(image.rastr, sizeof(uint8_t), image.head.filesize, file);

    return image;
}

void print_head(Head head)
{
    printf("signature: %s\n", (char*)&head.signature);
    printf("filesize: %d\n", head.filesize);
    printf("reserved: %d\n", head.reserved);
    printf("offset: %d\n", head.offset);
    printf("header_size: %d\n", head.header_size);
    printf("width: %d\n", head.width);
    printf("height: %d\n", head.height);
    printf("planes: %d\n", head.planes);
    printf("bpp: %d\n", head.bpp);
    printf("compression: %d\n", head.compression);
    printf("image_size: %d\n", head.image_size);
    printf("x_pixels_per_m: %d\n", head.x_pixels_per_m);
    printf("y_pixels_per_m: %d\n", head.y_pixels_per_m);
    printf("colors_used: %d\n", head.colors_used);
    printf("colors_important: %d\n", head.colors_important);
}

void write_image(FILE* file, Image image)
{
    fwrite(&image.head, sizeof(Head), 1, file);

    if (NULL != image.palette) {
        fwrite(image.palette, sizeof(uint8_t), (1 << image.head.bpp) *
4, file);
    }

    uint32_t padding = (4 - image.head.width * (image.head.bpp / 8) %
4) % 4;
    uint32_t bytes_per_pixel = image.head.bpp / 8;
    uint32_t row_size = image.head.width * bytes_per_pixel;

    for (int i = 0; i < image.head.height; i++) {
        fwrite(&image.rastr[i * row_size], sizeof(uint8_t), row_size,
file);
        fwrite(&padding, sizeof(uint8_t), padding, file);
    }
}
```

```
void make_border_for_image(Image* image)
{
    uint8_t bytes_per_pixel = image->head.bpp / 8;
    uint64_t old_row_size = image->head.width * bytes_per_pixel;

    uint8_t* new_rastr = calloc(old_row_size * image->head.height, 1);

    for (int i = 0; i <= image->head.height; i++) {
        for (int j = 0; j < old_row_size; j++) {
            if (j < 15 * bytes_per_pixel || j >= old_row_size - 15 *
bytes_per_pixel || i < 15 || i >= image->head.height - 15) {
                new_rastr[i * old_row_size + j] = rand() % 256;
            } else {
                new_rastr[i * old_row_size + j] = image->rastr[i *
old_row_size + j];
            }
        }
    }
    image->rastr = new_rastr;
}

int main(int argc, char* argv[])
{
    if (argc <= 2) {
        fprintf(stderr, "Usage: %s <input> <output>\n", argv[0]);
        return -1;
    }

    char* filename = argv[1];

    FILE* file = fopen(filename, "rb");
    if (NULL == file) {
        perror(filename);
        return -1;
    }

    Image image = read_image(file);

    print_head(image.head);

    make_border_for_image(&image);

    write_image(fopen(argv[2], "wb"), image);

    fclose(file);
    return 0;
}
```

Результат работы:

Рис. 1 цветной BMP файл

Рис. 2 Копия первого BMP файла с рамкой в 15 пикселей