Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Сибирский государственный университет телекоммуникаций и
информатики»

(СибГУТИ)

Кафедра прикладной математики и кибернетики

Лабораторная работа №8
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:

Написать программу для декодирования и вывода на экран PCX файла.

Листинг программы:

```
#include <SDL2/SDL.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>

#pragma pack(1)
typedef struct {
    uint8_t Manufacturer;
    uint8_t Version;
    uint8_t Encoding;
    uint8_t BitsPerPixel;
    uint16_t XMin;
    uint16_t YMin;
    uint16_t XMax;
    uint16_t YMax;
    uint16_t HRes;
    uint16_t VRes;
    uint8_t Palette[48];
    uint8_t Reserved;
    uint8_t ColorPlanes;
    uint16_t BytesPerLine;
    uint16_t PaletteType;
    uint16_t HScreenSize;
    uint16_t VScreenSize;
    uint8_t Filler[54];
} PCXHead;
#pragma(pop)

typedef struct {
    uint8_t red;
    uint8_t green;
    uint8_t blue;
} Color;

typedef struct {
    PCXHead head;
    uint64_t width;
    uint64_t height;
    uint8_t* rastr;
    Color palette[256];
} Image;

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

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

```
    read_head(&image.head, file);

    if (image.head.Manufacturer != 0x0A || image.head.Encoding != 1 ||
image.head.BitsPerPixel != 8) {
        fprintf(stderr, "Error: Unsupported PCX file format.\n");
        exit(EXIT_FAILURE);
    }

    uint64_t palette_start_byte;
    if (image.head.ColorPlanes == 1 && image.head.BitsPerPixel == 8) {

        fseek(file, -769, SEEK_END);
        palette_start_byte = ftell(file);
        uint8_t palette_marker;
        fread(&palette_marker, sizeof(uint8_t), 1, file);
        if (palette_marker != 0x0C) {
            fprintf(stderr, "Error: Invalid palette marker.\n");
            exit(EXIT_FAILURE);
        }

        for (int i = 0; i < 256; i++) {
            fread(&image.palette[i].red, sizeof(uint8_t), 1, file);
            fread(&image.palette[i].green, sizeof(uint8_t), 1, file);
            fread(&image.palette[i].blue, sizeof(uint8_t), 1, file);
        }
    }

    image.width = image.head.XMax - image.head.XMin + 1;
    image.height = image.head.YMax - image.head.YMin + 1;

    uint64_t filesize = image.head.ColorPlanes *
image.head.BytesPerLine * (image.height + 1);

    image.rastr = (uint8_t*)malloc(filesize);

    if (!image.rastr) {
        fprintf(stderr, "Error: Memory allocation failed for image
raster.\n");
        exit(EXIT_FAILURE);
    }

    size_t index = 0;
    fseek(file, sizeof(PCXHead), SEEK_SET);
    uint8_t byte;
    while (index < filesize) {
        if (ftell(file) >= palette_start_byte) {
            break;
        }
        fread(&byte, sizeof(uint8_t), 1, file);
        if ((byte & 0xC0) == 0xC0) {
            int count = byte & 0x3F;
            fread(&byte, sizeof(uint8_t), 1, file);
            for (int i = 0; i < count; i++) {
                image.rastr[index++] = byte;
            }
        } else {
            image.rastr[index++] = byte;
```

```
        }
    }

    return image;
}


void display_image(Image image)
{
    SDL_Init(SDL_INIT_VIDEO);
    SDL_Window* window = SDL_CreateWindow("Test",
SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED,
        image.width, image.height, 0);

    if (!window) {
        fprintf(stderr, "Error: Failed to create window!\n");
        exit(EXIT_FAILURE);
    }

    SDL_Renderer* renderer = SDL_CreateRenderer(window, -1,
SDL_RENDERER_ACCELERATED);
    uint64_t rastr_index = 0;
    for (int y = 0; y < image.height; y++) {
        for (int x = 0; x < image.head.BytesPerLine; x++) {
            Color color = image.palette[image.rastr[rastr_index++]];
            SDL_SetRenderDrawColor(renderer, color.red, color.green,
color.blue, 0xFF);
            SDL_RenderDrawPoint(renderer, x, y);
        }
    }

    SDL_RenderPresent(renderer);

    bool running = true;
    SDL_Event event;
    while (running) {
        while (SDL_PollEvent(&event)) {
            if (event.type == SDL_QUIT) {
                running = false;
            }
        }
    }

    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);
    SDL_Quit();
}

void print_header(PCXHead head)
{
    printf("Manufacturer: %d\n", head.Manufacturer);
    printf("Version: %d\n", head.Version);
    printf("Encoding: %d\n", head.Encoding);
    printf("Bits Per Pixel: %d\n", head.BitsPerPixel);
    printf("Bytes Per Line: %d\n", head.BytesPerLine);
    printf("Planes: %d\n", head.ColorPlanes);
    printf("X Min: %d\n", head.XMin);
    printf("Y Min: %d\n", head.YMin);
```

```
    printf("X Max: %d\n", head.XMax);
    printf("Y Max: %d\n", head.YMax);
    printf("H Res: %d\n", head.HRes);
    printf("V Res: %d\n", head.VRes);
    printf("Palette: ");
    for (int i = 0; i < 48; i++) {
        printf("%d ", head.Palette[i]);
    }
    printf("\n");
}

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

    char* filename = argv[1];

    FILE* file = fopen(filename, "rb");
    if (!file) {
        perror(filename);
        return EXIT_FAILURE;
    }

    Image image = read_image(file);

    print_header(image.head);

    fclose(file);

    display_image(image);

    free(image.rastr);

    return EXIT_SUCCESS;
}
```

Результат работы:

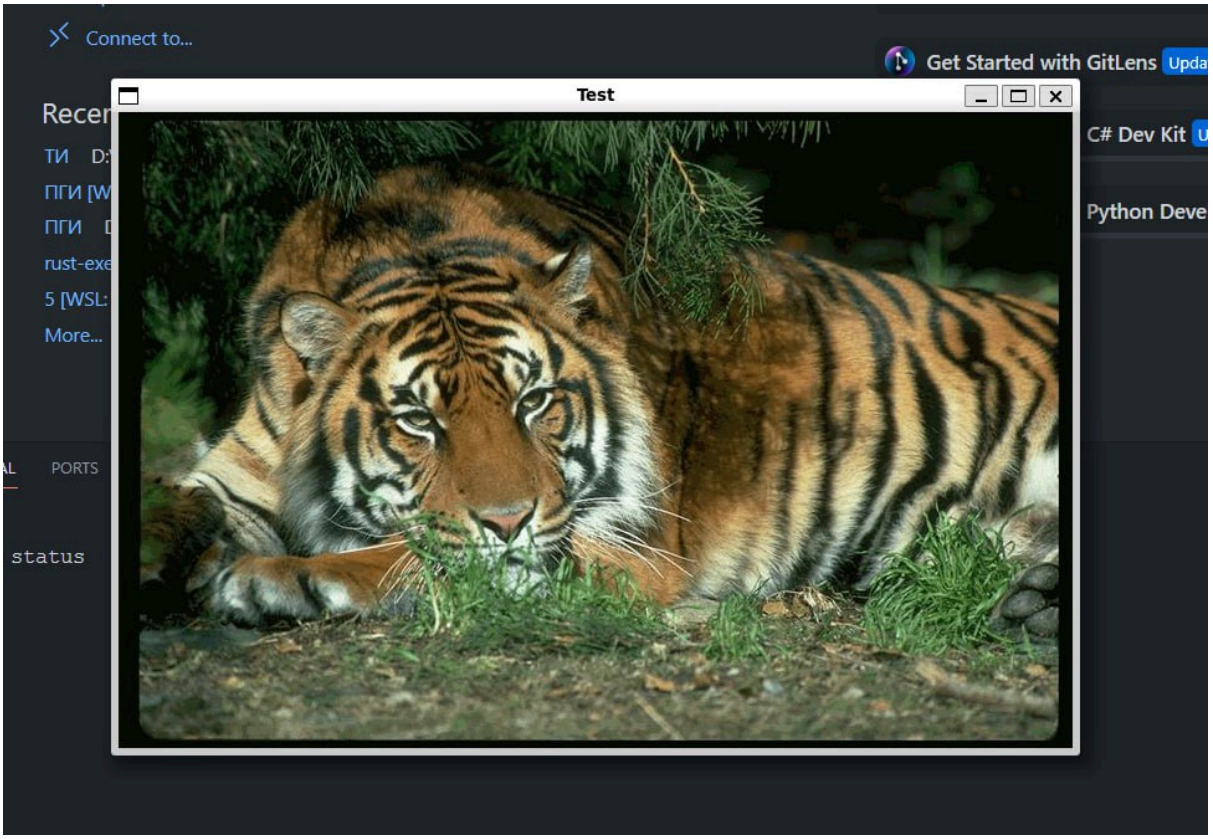

Рис. 1 декодированный PCX файл