Министр науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет информационных технологий и программирования

Лабораторная работа № 14

Игра жизнь

Выполнил студент группы № M3105
Пасичник Артем Аркадьевич
Подпись:

Проверил:
Повышев Владислав Вячеславович

Санкт-Петербург
2020

Лабораторная работ 14. Игра жизнь

Целью лабораторной работы является реализация игры “Жизнь” , позволяющая выводить поколение игры в монохромную картинку в формате BMP. Плоскость “вселенной” игры ограничена положительными координатами.

Лабораторная работы должна быть выполнена в виде консольного приложения принимающего в качестве аргументов следующие параметры:

1. ***--input input_file.bmp***
   Где input_file.bmp - монохромная картинка в формате bmp, хранящая начальную ситуация (первое поколение) игры
2. ***--output dir_name***
   Название директории для хранения поколений игры в виде монохромной картинки
3. ***--max_iter N***
   Максимальное число поколений которое может эмулировать программа. Необязательный параметр, по-умолчанию бесконечность
4. ***--dump_freq N***
   Частота с которой программа должно сохранять поколения виде картинки. Необязательный параметр, по-умолчанию равен 1

Программа должна предусматривать исключительные ситуации, которые могут возникать во время ее работы и корректно их обрабатывать.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <direct.h>

void GameLife(int **life, int hei, int wid) {
    int N;
    int **tmp = (int **) malloc(sizeof(int *) * hei);
    for (int i = 0; i < hei; ++i) {
        tmp[i] = (int *) malloc(sizeof(int) * wid);        //Создаем
временный массив пикселей
    }
    int x, y;
    int inf = 0;
    for (y = 0; y < hei; y++)
        for (x = 0; x < wid; x++)
            tmp[y][x] = life[y][x];         //Копируем life в temp
    for (y = 1; y < hei - 1; y++) {
        for (x = 1; x < wid - 1; x++) {
            N = life[y + 1][x - 1] + life[y + 1][x] + life[y + 1][x + 1] +
life[y][x - 1] + life[y][x + 1] +
                life[y - 1][x - 1] + life[y - 1][x] + life[y - 1][x + 1];
//Считаем кол-во соседей
            if (tmp[y][x] == 1) {
                if (N == 2) tmp[y][x] = life[y][x];
                if (N == 3) tmp[y][x] = life[y][x];    //Условие игры жизнь
                if (N > 3) tmp[y][x] = 0;
                if (N < 2) tmp[y][x] = 0;
            } else {
                if (N == 3) tmp[y][x] = 1;
            }
            N = 0;
        }
    }
    for (y = 0; y < hei; y++)
        for (x = 0; x < wid; x++) {
            if (life[y][x] == tmp[y][x])
                inf++;                      //Счетчик кол-ва одинаковых
пикселей
            life[y][x] = tmp[y][x];         //Обратно заполняем life
        }
    for (int i = 0; i < hei; ++i) {
        free(tmp[i]);
    }                                      //Освобождаем память
    free(tmp);
    if (inf == hei * wid)                  //Если все пиксели не изменились, то
завершаем работу
        exit(0);
}

struct Bmp {
    int Width;
    int Height;
    int Size;
};

int main(int argc, char *argv[]) {
    struct Bmp Image;
    unsigned char header[54];          //Массив хедеров
    int maxiter = 100, dumpfreq = 1;
    char *dirname = NULL;
    FILE *file = NULL;
```

```
    for (int i = 0; i < argc; i++) {
        if (!strcmp("--input", argv[i])) {
            file = fopen(argv[2], "rb");
        }
        if (!strcmp("--output", argv[i])) {
            dirname = argv[i + 1];
            _mkdir(dirname);
        }
        if (!strcmp("--max_iter", argv[i])) {
            maxiter = strtol(argv[i + 1], 0, 10);
        }
        if (!strcmp("--dump_freq", argv[i])) {
            dumpfreq = strtol(argv[i + 1], 0, 10);
        }
    }


    fread(header, sizeof(unsigned char), 54, file);
    Image.Width = header[21] * 256 * 256 * 256 + header[20] * 256 * 256 +
header[19] * 256 + header[18];
    Image.Height = header[25] * 256 * 256 * 256 + header[24] * 256 * 256 +
header[23] * 256 + header[22];
    Image.Size = header[5] * 256 * 256 * 256 + header[4] * 256 * 256 +
header[3] * 256 + header[2];
    unsigned char *imagebyte = (unsigned char *) malloc((Image.Size - 54) *
sizeof(unsigned char));
    fread(imagebyte, sizeof(unsigned char), Image.Size, file);

    int **img = (int **) malloc(Image.Height * sizeof(int *));
    for (int i = 0; i < Image.Height; i++)
        img[i] = (int *) malloc(Image.Width * sizeof(int));


    int k = 0;
    for (int i = Image.Height - 1; i >= 0; i--) {
        for (int j = 0; j < Image.Width; j++) {
            if (imagebyte[k] == 255)
                img[i][j] = 0;
            else
                img[i][j] = 1;
            k += 3;
        }
    }

    for (int l = 1; l <= maxiter; l++) {
        GameLife(img, Image.Height, Image.Width);

        if (l % dumpfreq != 0) continue;

        char way[256];
        char filename[16];
        strcpy(way, dirname);
        strcat(way, "\\");
        strcat(way, _itoa(l, filename, 10));   //Создаем путь к новому файлу
        strcat(way, ".bmp");
        FILE *life = fopen(way, "wb");
        fwrite(header, 1, 54, life);     //Записываем в него хедеры

        int m = 0;
        for (int i = Image.Height - 1; i >= 0; i--) {
            for (int j = 0; j < Image.Width; j++) {
                for (k = 0; k < 3; k++) {
                    if (img[i][j] == 1)
                        imagebyte[m] = 0;
```

```
                    else                                    //Преобразуем обратно
в битовое поле
                        imagebyte[m] = 255;
                    m++;
                }
            }
        }
        fwrite(imagebyte, sizeof(unsigned char), Image.Size, life);
//Записываем в файл
        fclose(life);
    }

    for (int i = 0; i < Image.Height; i++) {
        free(img[i]);
    }
    free(img);                                  //Освобождаем память
    free(imagebyte);
    return 0;
}
```