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.

Выполнила студентка группы № M31XX
Черных Арсений Игоревич
Подпись:

Проверил:
Повышев Владислав Вячеславович

Санкт-Петербург
2023

Текст задания

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

Лабораторная работа 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.

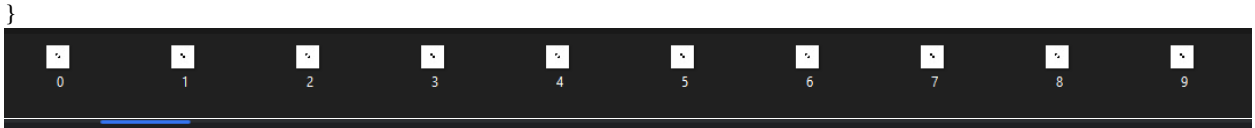
Решение с комментариями: 1) Создаю структуру с полями BMP, в которой хранятся ширина, высота, смещение, массив на 54 байта.
2) Функция Count считает кол-во соседей у текущей клетки и вычитает саму клетку, если она закрашена. 3) В функции converttoBMP я превращаю своё поле в картинку с RBG пикселями, поэтому беру по 3 элемента на пиксель. 4) Функцию memoryAllocation использую для выделения памяти под двумерный массив. 5) Функцией InformationForBMP я изменяю порядок, вместо убывания, у меня теперь возрастание. 6) В мейне проверяю на аргументы командной строки, читаю информацию из заголовка BMP, выполняю все итерации, сохраняю результаты и BMP-файлы и освобождаю память.

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

struct infoBMP {
    int height;
    int width;
    int size;
    int image_offset;
    unsigned char bmp_header[54];
};

int Count(unsigned char **arr, int x, int y) {
    int sum = 0;
    for (int i = x - 1; i <= x + 1; i++) {
        for (int j = y - 1; j <= y + 1; j++)
            sum += arr[i][j];
    }
    if (arr[x][y] == 1)
        sum--;
    return sum;
}

unsigned char *convertToBmp(unsigned char **matrix, int height, int width) {
    unsigned char *pixelData = (unsigned char *)malloc(height * width * 3 * sizeof(unsigned char)); // RGB 3 байта
    int index = 0;
    for (int i = height - 1; i >= 0; i--) {
        for (int j = 0; j < width; j++) {
            if (matrix[i][j] == 1) {
                pixelData[index] = 0;
                pixelData[index + 1] = 0;
                pixelData[index + 2] = 0;
            } else {
                pixelData[index] = 255;
                pixelData[index + 1] = 255;
                pixelData[index + 2] = 255;
            }
            index += 3;
        }
    }
    return pixelData;
}

unsigned char **memoryAllocation(int height, int width) {
    unsigned char **arr = (unsigned char **)malloc(height * sizeof(unsigned char *));
    if (arr == NULL)
        return NULL;
    for (int i = 0; i < height; i++) {
        arr[i] = (unsigned char *)malloc(width * sizeof(unsigned char));
        if (arr[i] == NULL)
```

```
            return NULL;
        }
        return arr;
    }

    void InformationForBMP(struct infoBMP informationBMP, FILE *image) {
        fread(informationBMP.bmp_header, sizeof(unsigned char), 54, image);
        informationBMP.image_offset = (informationBMP.bmp_header[0xD] << 24) | (informationBMP.bmp_header[0xC] << 16) |
                                      (informationBMP.bmp_header[0xB] << 8) | informationBMP.bmp_header[0xA];
        informationBMP.size = (informationBMP.bmp_header[0x5] << 24) | (informationBMP.bmp_header[0x4] << 16) |
                              (informationBMP.bmp_header[0x3] << 8) | informationBMP.bmp_header[0x2];
        informationBMP.width = (informationBMP.bmp_header[0x15] << 24) | (informationBMP.bmp_header[0x14] << 16) |
                               (informationBMP.bmp_header[0x13] << 8) | informationBMP.bmp_header[0x12];
        informationBMP.height = (informationBMP.bmp_header[0x19] << 24) | (informationBMP.bmp_header[0x18] << 16) |
                                (informationBMP.bmp_header[0x17] << 8) | informationBMP.bmp_header[0x16];
        printf("offset %d\n", informationBMP.image_offset);
        printf("size - %d\n", informationBMP.size);
        printf("height - %d\n", informationBMP.height);
        printf("width - %d\n", informationBMP.width);
        printf("\n");
    }

    int main(int argc, char *argv[]) {
        struct infoBMP informationBMP;

        if (argc < 4) {
            fprintf(stderr, "ERROR, no arguments");
            return 1;
        }

        int dump_freq = 1;
        int max_iter = 1;
        char *dirName;
        unsigned char **cur_gen;
        unsigned char **next_gen;

        FILE *image;
        for (int i = 1; i < argc; i += 2) {
            if (strcmp(argv[i], "--input") == 0) {
                image = fopen(argv[i + 1], "rb");
                if (image == NULL) {
                    printf("Can't open");
                    return 0;
                }
            } else if (strcmp(argv[i], "--output") == 0)
                dirName = argv[i + 1];
            else if (strcmp(argv[i], "--max_iter") == 0)
                max_iter = strtol(argv[i + 1], NULL, 10);
            else if (strcmp(argv[i], "--dump_freq") == 0)
                dump_freq = strtol(argv[i + 1], NULL, 10);
        }

        InformationForBMP(informationBMP, image);

        cur_gen = memoryAllocation(informationBMP.height, informationBMP.width * 3);
        next_gen = memoryAllocation(informationBMP.height, informationBMP.width * 3);

        fseek(image, informationBMP.image_offset + (informationBMP.width * 3), SEEK_SET);

        unsigned char buffer[3];
        for (int i = informationBMP.height - 1; i >= 0; i--) {
            for (int j = 0; j < informationBMP.width; j++) {
                buffer[0] = fgetc(image);
                buffer[1] = fgetc(image);
                buffer[2] = fgetc(image);
                if (buffer[0] != 0 && buffer[1] != 0 && buffer[2] != 0)
```

```
            cur_gen[i][j] = 0;
        else
            cur_gen[i][j] = 1;
    }
}
char fileName[999];
char directory[256];
unsigned char *pixelInfo;
for (int i = 0; i < informationBMP.height; i++) {
    for (int j = 0; j < informationBMP.width; j++)
        next_gen[i][j] = cur_gen[i][j];
}

int countOfNeighbors;
for (int gameIteration = 0; gameIteration < max_iter; gameIteration++) {
    for (int i = 1; i < informationBMP.height - 1; i++) {
        for (int j = 1; j < informationBMP.width - 1; j++) {
            countOfNeighbors = Count(cur_gen, i, j);

            if (cur_gen[i][j] == 0 && countOfNeighbors == 3)
                next_gen[i][j] = 1;
            else if (cur_gen[i][j] == 1) {
                if (countOfNeighbors < 2 || countOfNeighbors > 3)
                    next_gen[i][j] = 0;
            }
        }
    }

    for (int i = 0; i < informationBMP.height; i++) {
        for (int j = 0; j < informationBMP.width; j++)
            cur_gen[i][j] = next_gen[i][j];
    }

    pixelInfo = convertToBmp(cur_gen, informationBMP.height, informationBMP.width);

    if (gameIteration % dump_freq == 0) {
        sprintf(fileName, "%d", gameIteration);
        strcpy(directory, dirName);
        strcat(directory, "/");
        strcat(directory, fileName);
        strcat(directory, ".bmp");

        FILE *new_bmp = fopen(directory, "wb");

        if (new_bmp != NULL)
            printf("File %d created\n", gameIteration);
        else
            printf("File can't be open %d\n", gameIteration);

        fwrite(informationBMP.bmp_header, 1, 54, new_bmp);
        fwrite(pixelInfo, 1, 3 * informationBMP.width * informationBMP.height, new_bmp);
        fclose(new_bmp);
    }
}

for (int i = 0; i < informationBMP.height; i++) {
    free(cur_gen[i]);
    free(next_gen[i]);
}
free(cur_gen);
free(next_gen);
free(pixelInfo);

fclose(image);

return 0;
```

}




```
PS C:\Users\chern\CLionProjects\C>  ./main --input q.bmp  --max_iter 20 --output game
offset 247726080
size - 674627584
height - 0
width - 1073741760

File 0 created
File 1 created
File 2 created
File 3 created
File 4 created
File 5 created
File 6 created
File 7 created
File 8 created
File 9 created
File 10 created
File 11 created
File 12 created
File 13 created
File 14 created
File 15 created
File 16 created
File 17 created
File 18 created
File 19 created
PS C:\Users\chern\CLionProjects\C>
```