

Министр науки и высшего образования Российской Федерации

**Федеральное государственное автономное образовательное учреждение высшего
образования**

«Национальный исследовательский университет ИТМО»

Факультет информационных технологий и программирования

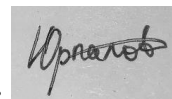
Лабораторная работа №14

Название работы: Игра жизнь.

Выполнил студент группы № М3105

Юрпалов Сергей Николаевич

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 ситуацию (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:

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

void imagetoarray(FILE *image, int **array, int height, int width){
    char c;
    for (int i = 0; i < height; i++){
        int bit = 0;
        while (bit < width){
            c = fgetc(image);
            for (int j = 0; j < 8; j++){
                int n = (c >> (7 - j)) % 2;
                array[i][bit] = n;
                bit++;
                if (bit == width)
                {
                    bit += 7 - j;
                }
            }
        }
    }
}

// Читаем bmp-файл побайтно, парсим байт на биты, записываем их в массив

int gamelife(int **oldgeneration, int height, int width){
    int newgeneration[height][width];
    for (int i = 0; i < height; i++){
        for (int j = 0; j < width; j++){
            newgeneration[i][j] = oldgeneration[i][j];
        }
    }
    for (int i = 0; i < height; i++){
        for (int j = 0; j < width; j++){
            int cellneighbour = 0;
            int right, left, top, bottom;
            if (j == width - 1){
                right = 0;
            }
            else{
                right = j + 1;
            }
            if (j == 0){
                left = width - 1;
            }
            else{
                left = j - 1;
            }
            if (i == 0){
                top = height - 1;
            }
            else{
                top = i - 1;
            }
            if (i == height - 1){
                bottom = 0;
            }
            else{
                bottom = i + 1;
            }
            cellneighbour += oldgeneration[top][left] + oldgeneration[top][j]
+ oldgeneration[top][right] + oldgeneration[i][left] +
oldgeneration[i][right] + oldgeneration[bottom][left] +
```

```

oldgeneration[bottom][j] + oldgeneration[bottom][right];
    if (oldgeneration[i][j] == 0 && cellneighbour == 3)
    {
        newgeneration[i][j] = 1;
    }
    else if (cellneighbour != 2 && cellneighbour != 3)
    {
        newgeneration[i][j] = 0;
    }
}
}
int changes = 0;
for (int i = 0; i < height; i++)
{
    for (int j = 0; j < width; j++)
    {
        if (oldgeneration[i][j] != newgeneration[i][j])
        {
            changes += 1;
        }
        oldgeneration[i][j] = newgeneration[i][j];
    }
}
if (changes > 0){
    return 1;
}
return 0;
}
// Считаем соседей клетки, выставляем значение данной в соответствии с
условиями игры. Если клетка крайняя, переназначаем соседей. Если поле игры
изменилось, возвращаем 1, иначе 0.

void arraytoimage(int **array, int height, int width, int iteration, char
*directory, char *input, int indexout)
{
    char newinput[50];
    char buffer[50];
    char c;
    FILE *image;
    FILE *imageout;
    image = fopen(input, "rb");
    strcpy(newinput, directory);
    strcat(newinput, "\\");
    strcat(newinput, itoa(iteration+1, buffer, 10));
    strcat(newinput, ".bmp");
    imageout = fopen(newinput, "wb");
    for (int i = 0; i < indexout; i++){
        c = fgetc(image);
        fputc(c, imageout);
    }
    for (int i = 0; i < height; i++){
        int j = 0;
        c = 0;
        while (j < width)
        {
            c = c << 1;
            c += array[i][j];
            j++;
            if (j % 8 == 0)
            {
                fputc(c, imageout);
                c = 0;
            }
        }
        else if (j == width)

```

```

        {
            int k = 8 - (j % 8);
            c = c << k;
            j += k;
            fputc(c, imageout);
        }
    }
    fclose(image);
    fclose(imageout);
}
// Формируем название файла, после записываем содержимое в таком же порядке,
как и при вводе.

int main(int argc, char* argv[]) {
    FILE *inputFile;
    char *outputDirectory;
    char *inputFileName;
    int amountofiter = 100;
    int freq = 1;
    for(int i = 1; i < argc; i++) {
        if (strcmp(argv[i], "--input") == 0){
            inputFileName = argv[i + 1];
        }
        else if(strcmp(argv[i], "--max_iter") == 0) {
            amountofiter = atoi(argv[i + 1]);
        }
        else if(strcmp(argv[i], "--dump_freq") == 0) {
            freq = atoi(argv[i + 1]);
        }
        else if(strcmp(argv[i], "--output") == 0) {
            outputDirectory = argv[i+1];
        }
    }
    inputFile = fopen(inputFileName, "r+b");
    char BMPINFO[54];
    int width, height, outputname;
    fread(BMPINFO, 1, 54, inputFile);
    outputname = BMPINFO[10] + BMPINFO[11] * 256 + BMPINFO[12] * 65536 +
BMPINFO[13] * 16777216;
    width = BMPINFO[18] + BMPINFO[19] * 256 + BMPINFO[20] * 65536 +
BMPINFO[21] * 16777216;
    height = BMPINFO[22] + BMPINFO[23] * 256 + BMPINFO[24] * 65536 +
BMPINFO[25] * 16777216;
    mkdir(outputDirectory);
    for (int i = 0; i < outputname - 54; i++){
        fgetc(inputFile);
    }
    int **currentgeneration;
    currentgeneration = (int**) malloc(height * sizeof(int*));
    for(int i = 0; i < height; i++) {
        currentgeneration[i] = (int *) malloc(width * sizeof(int));
    }
    imagetoarray(inputFile, currentgeneration, width, height);
    for (int s = 0; s < amountofiter; s++)
    {
        int result = gamelife(currentgeneration, height, width);
        if (result == 0){
            break;
        }
        if (s % freq == 0)
        {
            arraytoimage(currentgeneration, height, width, s,
outputDirectory, inputFileName, outputname);

```

```
    }  
}  
// Читаем bmp-файл в соответствии с форматом, записываем высоту изображения,  
его ширину и сдвиг его данных. Парсим входные данные. Запускаем игру в цикле  
с заданным количеством итераций. Если поле не изменилось (ф-я вернула 0),  
останавливаем игру.
```