Российская федерация

Ханты-Мансийский автономный округ – Югра

Департамент образования и науки

Сургутский государственный университет ХМАО

Политехнический институт

Кафедра автоматики и компьютерных систем

**Пояснительная записка**

К курсовому проекту

по дисциплине «Структурное программирование»

Выполнил: Бельтюков М. О.

студент группы 609-22

Проверил: Гришмановский П. В.

доцент кафедры автоматики и компьютерных систем

Сургут

2023 г.

# Задание

Вариант 40 – Муравьи Лэнгтона. Требуется реализовать клеточный автомат “Муравьи Лэнгтона”.

# Аннотация

## Цель и задачи

Необходимо реализовать программу на языке программирования “C”, которая будет выводить клеточный автомат муравьи Лэнгтона. Таким образом можно выделить следующие задачи:

1. Изучить OpenGl для вывода муравьёв Лэнгтона
2. Определиться с тем, какие параметры нужно задавать для клеточного автомата
3. Реализовать основную программу, которая будет с помощью OpenGl выводить муравьёв Лэнгтона
4. Реализовать дополнительную программу, для изменения параметров, принимаемых основной программой

## Результаты

Реализованы 2 программы, одна для сохранения/загрузки параметров клеточного автомата, другая для вывода клеточного автомата.

Оглавление

[Задание 2](#_Toc138060545)

[Аннотация 3](#_Toc138060546)

[Цель и задачи 3](#_Toc138060547)

[Результаты 3](#_Toc138060548)

[Содержание **Ошибка! Закладка не определена.**](#_Toc138060549)

[Введение 6](#_Toc138060550)

[1 Этап “Анализ” 7](#_Toc138060551)

[1.1 Обзор аналогов 8](#_Toc138060552)

[1.1.1 Аналог 1 8](#_Toc138060553)

[1.1.2 Аналог 2 9](#_Toc138060554)

[1.1.3 Аналог 3 11](#_Toc138060555)

[1.2 Обобщение частной задачи 12](#_Toc138060556)

[2 Этап “Проектирование” 13](#_Toc138060557)

[2.1 Формальное описание задачи 13](#_Toc138060558)

[2.2 Поведенческая модель программы 13](#_Toc138060559)

[2.3 Структура файлов 14](#_Toc138060560)

[2.4 Представление данных в памяти при их обработке 14](#_Toc138060561)

[2.5 Основные алгоритмы 16](#_Toc138060562)

[2.5.1 Основные алгоритмы в программе menu 16](#_Toc138060563)

[2.5.2 Основные алгоритмы в программе Langton\_ant: 16](#_Toc138060564)

[3 Этап “ Разработка программного продукта” 18](#_Toc138060565)

[3.1 Программа меню 18](#_Toc138060566)

[3.1.1 структуры программы меню 18](#_Toc138060567)

[3.1.2 Функции программы меню 18](#_Toc138060568)

[3.1.3 Функция main 23](#_Toc138060569)

[3.2 Программа Langton\_ant 24](#_Toc138060570)

[3.2.1 Функции программы Langton\_ant 24](#_Toc138060571)

[4 Тестирование 27](#_Toc138060572)

[4.1 Программа menu 27](#_Toc138060573)

[4.1.1 Проверка возможности добавить муравья 27](#_Toc138060574)

[4.1.3 Проверка ввода параметров поля: 29](#_Toc138060575)

[4.1.4 Проверка функции, которая измененяет правила по индексу 30](#_Toc138060576)

[4.1.5 Сохранение в главный файл: 31](#_Toc138060577)

[4.1.6 Сохранение и загрузка параметров поля и муравьёв в сторонние файлы 33](#_Toc138060578)

[4.2 Программа Langton\_ant 34](#_Toc138060579)

[5 Источники 39](#_Toc138060580)

[6 Приложения 40](#_Toc138060581)

[Приложения А. Рисунки 40](#_Toc138060582)

[Приложения Б. Листинги 45](#_Toc138060583)

# Введение

Клеточный автомат - дискретная модель, представляющая собой сетку произвольной размерности, каждая клетка которой в каждый момент времени может принимать одно из конечного множества состояний, и определено правило перехода клеток из одного состояния в другое. [1] В данном курсовом проекте рассматривается клеточный автомат Муравьи Лэнгтона.

Двумерные клеточные автоматы используются для генерации случайных чисел. Клеточные автоматы предложены для использования в криптосистемах с открытым ключом. В этом случае односторонняя функция является результатом эволюции конечного клеточного автомата, первоначальное состояние которого сложно найти. По заданному правилу легко найти результат эволюции клеточного автомата, но очень сложно вычислить его предыдущие состояния. [7]

Процессоры на клеточных автоматах — физическая реализация идей клеточного автомата. Элементы процессора размещены на равномерной сетке одинаковых ячеек. Состояние ячеек определяются взаимодействием со смежными клетками-соседями. В свою очередь соседство может определяться по фон Нейману или по Муру. Один из таких процессоров имеет вид систолической матрицы. Взаимодействие частиц может быть реализовано с помощью электрического тока, магнетизма, вибрации (например, фононы), либо и использованием любого другого способа передачи информации. Передача информации может быть осуществлена несколькими способами, которые не предусматривают использования проводников для передачи информации между элементами. Такой способ устройства процессора очень отличается от большинства процессоров, используемых на сегодняшний день и построенных по принципу фон Неймана, в которых процессор разделен на несколько секций, которые могут взаимодействовать между собой с использованием непосредственно проводников. [7]

# 1 Этап “Анализ”

Познакомимся с муравьём Лэнгтона поближе. Он живёт на бесконечной плоскости, состоящей из белых клеток. У него есть два неиссякаемых ведёрка — одно с белой краской, другое с чёрной. Муравей перемещается по клеткам плоскости от одной клетки к другой. При этом он выполняет несложный алгоритм:

1. Если клетка белая, то муравей перекрашивает её в чёрный цвет, поворачивает на 90° направо (по часовой стрелке) и делает шаг вперёд.
2. Если клетка чёрная, то муравей перекрашивает её в белый цвет, поворачивает на 90° налево (против часовой стрелки) и делает шаг вперёд. [2]

Муравей Лэнгтона также может быть описан как клеточный автомат в котором почти всё поле покрашено в чёрно-белый цвет, а клетка с «муравьём» имеет один из восьми различных цветов, кодирующих соответственно все возможные комбинации чёрного/белого цвета клетки и направления движения муравья.

Расширение муравья Лэнгтона

Существует простое расширение муравья Лэнгтона, в котором используется более двух цветов клеток. Цвета изменяются циклически. Для таких муравьев существует также простая форма названия: для каждого следующего цвета используется буква *L* или *R* (*Л* и *П*), в зависимости от того, поворачивает ли муравей направо или налево. Таким образом, муравей Лэнгтона — это муравей *RL*.

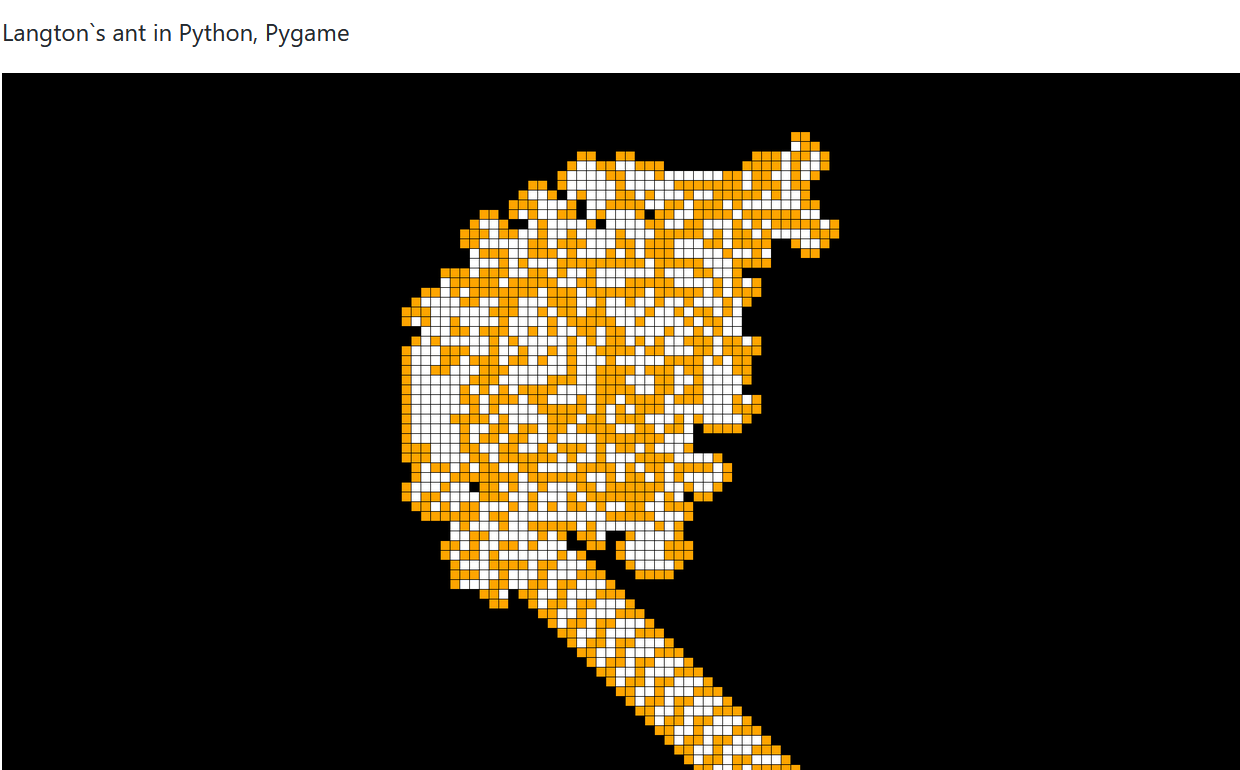
Некоторые из этих обобщенных муравьев Лэнгтона рисуют узоры, которые становятся все более симметричными. Один из простых примеров — муравей *RLLR*. Одно достаточное условие этого заключается в том, что имя муравья, рассматриваемое как циклический список, состоит из последовательных пар повторяющихся букв *LL* или *RR* (цикличность списка означает, что последняя буква может спариваться с первой).

Также добавлена буква N, которая означает, что муравей не будет поворачиваться и просто пройдёт вперёд. [3]

## Обзор аналогов

### 1.1.1 Аналог 1

исходный код Langton’s ant in Python [4]

Рисунок 1 – работа первого аналога

Плюсы этого аналога:

* Высокий уровень оптимизации.

Минусы:

* Нет возможности добавлять муравьёв через интерфейс
* Отсутствие взаимодействия с пользователем
* Без установленного python программу не запустить

Функции:

* Показывает муравьёв

### 1.1.2 Аналог 2

Langton’s ant [5]

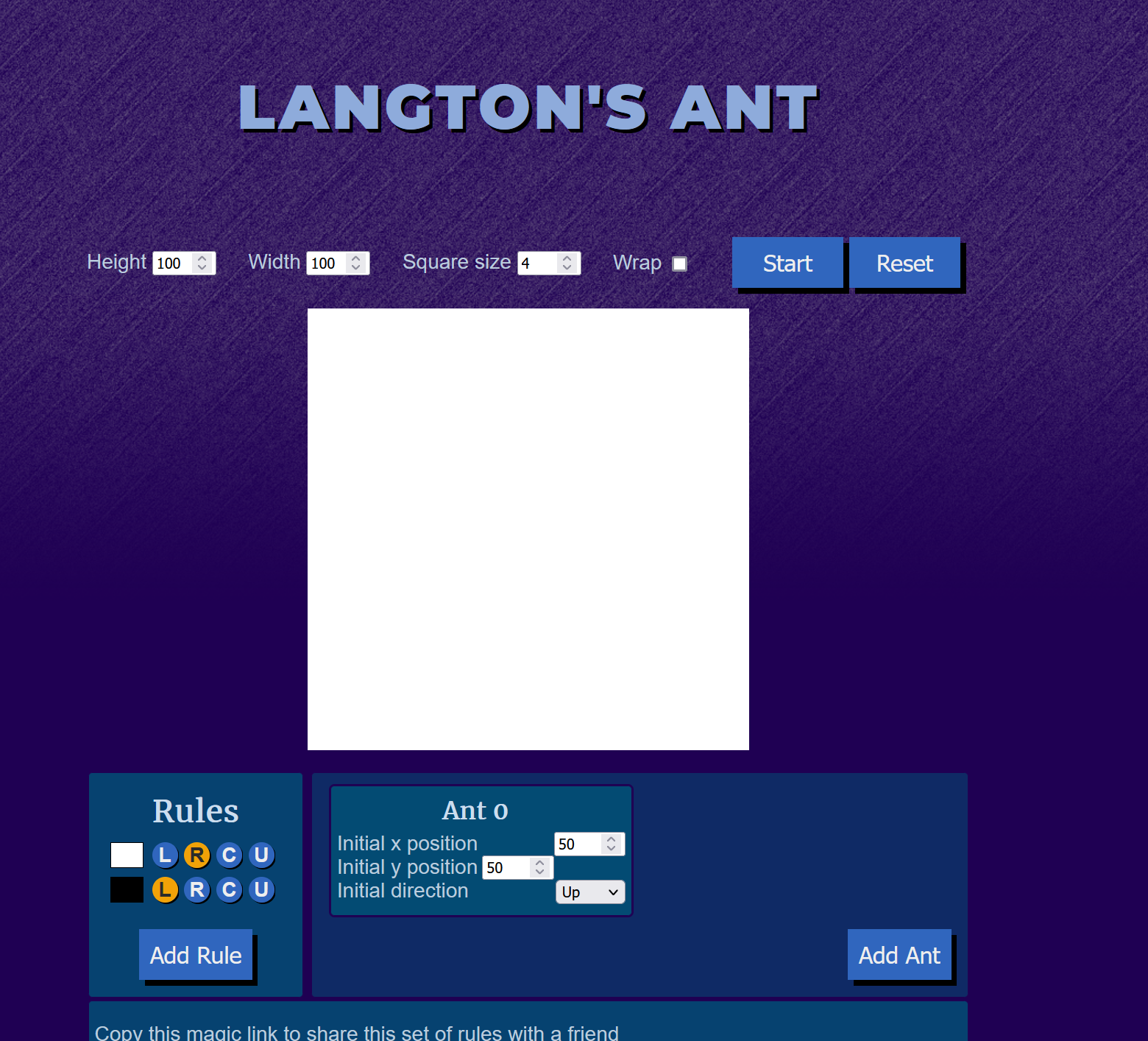


Рисунок 2 – интерфейс аналога 2

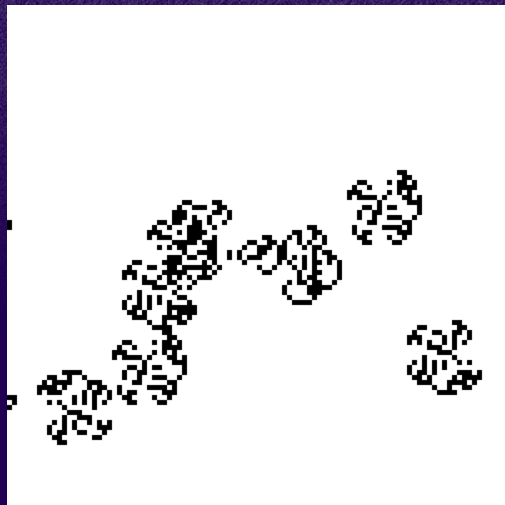


Рисунок 3 – Работа аналога 2

Плюсы этого аналога:

* Приятный интерфейс
* Не требует сторонних ресурсов (прошлый аналог требует установленного python на компьютер, а этот можно просто запустить в браузере)
* Хорошая оптимизация (Быстро выполняются итерации)
* Возможность изменять правила
* Справка о работе программы

Минусы этого аналога:

* Английский интерфейс

Функции:

* Задавать расположение муравьёв
* Показывать муравьёв
* Изменение правил движения муравьёв
* Задавать размер поля
* Задавать размер пикселя

### 1.1.3 Аналог 3

Langton’s ant [6]

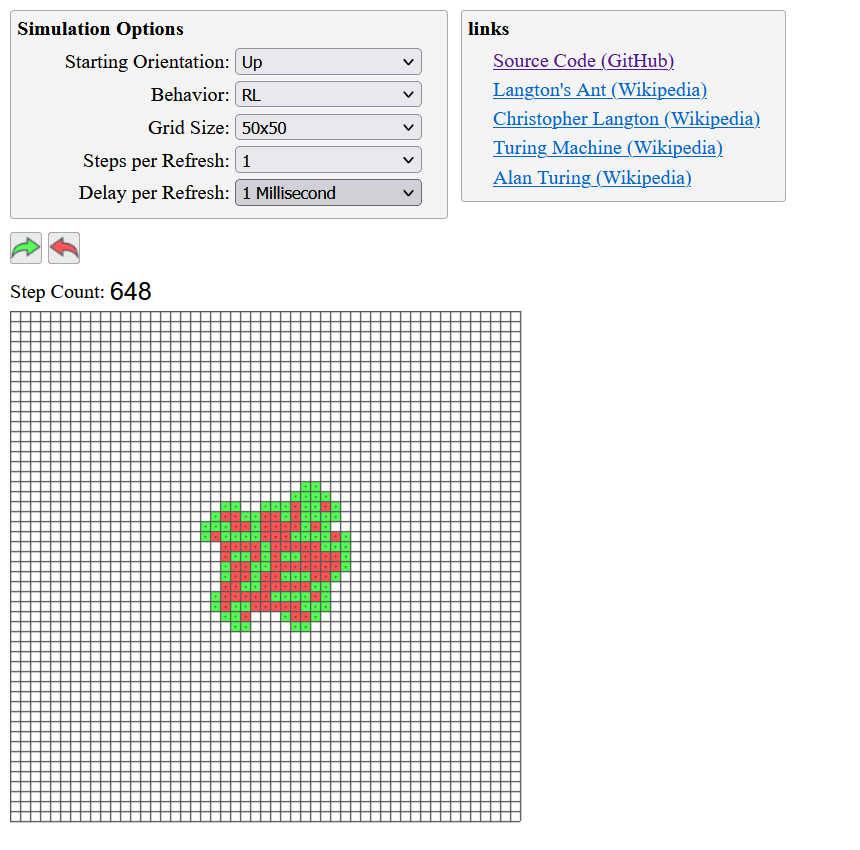


Рисунок 4 – интерфейс аналога 3

Плюсы этого аналога:

* Приятный интерфейс
* Не требует сторонних ресурсов (прошлый аналог требует установленного python на компьютер, а этот можно просто запустить в браузере)
* Хорошая оптимизация (Быстро выполняются итерации)
* Возможность изменять правила

Минусы этого аналога:

* Английский интерфейс
* Отсутствие документации

Функции: Заголовок

* Задавать расположение муравьёв
* Показывать муравьёв
* Изменение правил движения муравьёв
* Задавать размер поля

## 1.2 Обобщение частной задачи

Написать клеточный автомат “Муравьи Лэнгтона”.

Размер поля планируется, как константная величина, параметром для поля будет массив координат. Возможность редактирования будет достигнута благодаря записи данных в файл.

Задачи:

* Сохранение текущего набора правил, муравьёв, параметров поля и количества итераций.
* Возможность сохранять параметры поля в разные файлы, загрузка будет происходить из отдельного файла, в который можно будет загрузить данные через меню, также, как и в любые другие файлы.
* Интерактивный интерфейс с помощью меню. В меню можно заполнить параметры программы (координаты муравьёв, размер муравья, и т.д. (ниже указано подробнее)), сохранить параметры в 2 основных файла (параметры поля и муравьи), загрузить из файла параметры поля или муравьёв, сохранить параметры в другой файл, который не будет подтягивать основная программа.
* Реализацию нужно выполнить под Windows с использованием OpenGl
* Добавить шаблоны поведения муравьёв
* Добавить редактирование размеров пикселя
* Добавить различные цвета для муравьёв
* Реализовать клеточный автомат и пустить в личное, а потом и локальное тестирование.

# 2 Этап “Проектирование”

## 2.1 Формальное описание задачи

Программа по входным параметрам должна реализовывать клеточный автомат. Также программа должна хранить несколько файлов с входными параметрами.

## 2.2 Поведенческая модель программы

Диалог с пользователем происходит при помощи файла menu.exe – Файл открывает консоль и предлагает варианты действий на выбор

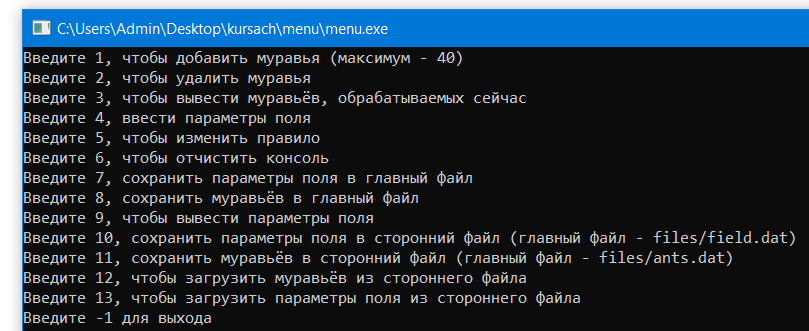


Рисунок 5 – Вид программы меню

Далее мы должны открыть файл Langton\_ant.exe и увидеть результат работы программы.

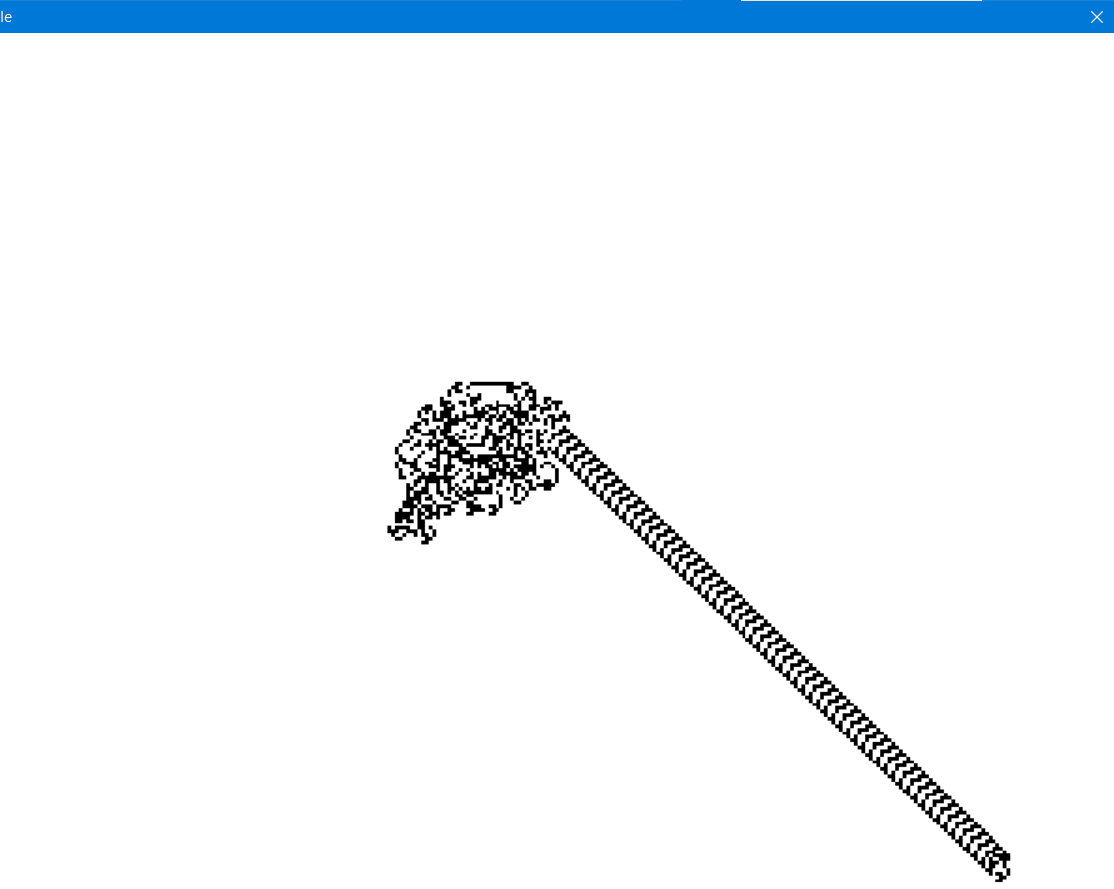


Рисунок 6 – Работа программы Langton\_ant.exe

## 2.3 Структура файлов

Загружать файлы можно откуда угодно, сохранять файлы можно куда угодно с любым расширением. Основные файлы находятся в папке files и имеют расширение **.dat** т.к. было использовано двоичное представление данных.

## 2.4 Представление данных в памяти при их обработке

Структуры данных, которые используются в обоих программах

typedef struct{

int rule;

int x;

int y;

}Ant;

Листинг 1 – Структура муравей

В структуре муравей 3 переменных: rule отвечает за то, куда смотрит муравей, x и y за его координаты.

typedef struct{

int color[20][3];

int rule[20];

int count\_rule;

int px;

}Field;

Листинг 2 – Структура поле

В структуре поле 4 переменных, color[20][3] Хранит набор цветов в формате rgb, rule[20] хранит набор правил, count\_rule это количество правил и px это размер блока в пикселях

Ant ants[40] = {0};

Field field;

int n=0, i=0, menu=100;

char name[100];

Листинг 3 – переменные в функции main программы menu

В меню для всех чисел используется тип int, для всех строк char\*.

Ant ants;

Field field;

char color[1000][720] = {0};

int iter = 0, n = 0, j=0, width=0, height=0, i, flag=1;

Листинг 4 – переменные в программе Langton\_ant

В программе Langton\_ant.exe используется тип char \*\* для представления поля, где значение каждого элемента соответствует индексу цвета в структуре Field. Для представления цвета используется тип int <имя>[3] - формат rgb (прозрачность всегда равна 0).

## 2.5 Основные алгоритмы

### 2.5.1 Основные алгоритмы в программе menu

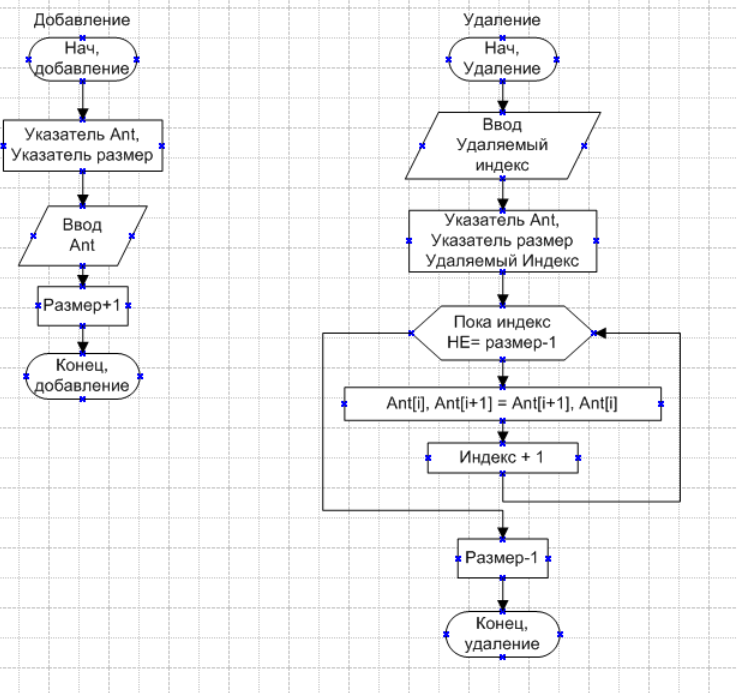


Рисунок 7 – Блок схемы добавления и удаления муравьёв

### 2.5.2 Основные алгоритмы в программе Langton\_ant:

#### Пояснения к блок схеме рисунка А.1 и А.1.1

В блок схеме вместо право/лево/вверх/вниз используем соответственно 1/2/3/4.

Также мы знаем куда смотрит муравей и в зависимости от заранее заданного правила для этого цвета муравей поворачивает в одну из 4х сторон. Отсюда оператор switch в switch-е для каждой ситуации муравей смотрит в определённую сторону и двигается по полю в одну из четырёх сторон (одна из координат изменяется на один)

# 3 Этап “ Разработка программного продукта”

## 3.1 Программа меню

Данная программа включает в себя папку files и компилируется в menu.exe файл.

Во всех функциях программы menu с вводом значений с клавиатуры реализована проверка ввода, то есть при вводе неправильных значений программа выводит сообщение “Неверно” и просит ввести данные заново.

### 3.1.1 структуры программы меню

1. typedef struct{
2. int rule;
3. int x;
4. int y; }Ant;
5. typedef struct{
6. int color[20][3];
7. int rule[20];
8. int count\_rule;
9. int px;
10. int iteration; }Field;

Листинг 5 – структуры в программе menu

В структуре Ant строк 1-4 заданы 3 переменные, отвечающие за направление муравья и координаты по x и по y. В структуре Field заданы 5 переменных, отвечающих за цвет в формате rgb, правила, количество правил (служебная переменная) размер квадрата, который красит муравей и количество повторений муравья.

### 3.1.2 Функции программы меню

1. void input(Ant \*inputed){
2. int x, y, r;
3. if (inputed == NULL){
4. return;}
5. printf("Введите координаты муравья и куда он смотрит (1 - право, 2 - лево, 3 - вверх, 4 - вниз)\n");
6. /\* Изначально вверх \*/
7. while (scanf("%d%d%d", &(x), &(y), &(r)) != 3
8. || r < 1 || r > 4){
9. printf("Неверно\n");
10. fflush(stdin);
11. }
12. inputed->x = x;
13. inputed->y = y;
14. inputed->rule = r;}

Листинг 5 – функция ввода одного муравья с консоли

1. void output(Ant \*outputed){
2. if (outputed == NULL){
3. return;
4. }
5. printf("(%d, %d)\n", outputed->x, outputed->y);
7. switch (outputed->rule){
8. case 1:
9. printf("Муравей смотрит вправо\n");
10. break;
11. case 2:
12. printf("Муравей смотрит влево\n");
13. break;
14. case 3:
15. printf("Муравей смотрит вверх\n");
16. break;
17. case 4:
18. printf("Муравей смотрит вниз\n");
19. break;
20. } }

Листинг 6 – функция, выводящая информацию об одном муравье

1. void add\_ant(Ant \*a, int \*n){
2. if (\*n > 39){
3. printf("Превышен лимит муравьёв\n");
4. return;
5. }
6. input(&a[\*n]);
7. (\*n)++;
8. }

Листинг 7 – функция, которая добавляет муравья

1. void del\_ant(Ant \*ants, int i, int \*n){
2. if (ants == NULL){
3. return;
4. }else if (\*n == 0){
5. printf("Нет муравьёв\n");
6. return;
7. }
8. i -= 1;
9. if (i < 0 || i > \*n - 1){
10. return;
11. }
12. Ant temp;
13. while (i != \*n - 1){
14. temp = ants[i+1];
15. ants[i+1] = ants[i];
16. ants[i] = temp;
17. i++;
18. }
19. (\*n)--;
20. }

Листинг 8 – функция, удаляющая муравья

1. void input\_rule(Field \*field, int i){
2. if (field){
3. int rule, r, g, b;
4. printf("Введите правило поведения муравья \n1: Право (-90)\n2: Лево (+90)\n3: Нейтрально(0)\n4: Обратно(180)\n");
5. while (scanf("%d", &rule) != 1 || rule < 1 || rule > 4){
6. printf("Неверно\n");
7. fflush(stdin);
8. }
9. (field->rule)[i] = rule;
10. printf("Введите цвет для этого правила в формате rgb\n");
11. while (scanf("%d%d%d", &r, &g, &b) != 3 ||
12. r > 255 || g > 255 || b > 255 ||
13. r < 0 || g < 0 || b < 0){
14. printf("Неверно\n");
15. fflush(stdin);}
16. (field->color)[i][0] = r;
17. (field->color)[i][1] = g;
18. (field->color)[i][2] = b;
19. }else return;
20. }

Листинг 9 – функция, для ввода одного правила и соответствующий этому правилу цвет.

1. void input\_field(Field \*field){
2. if (field){
3. int I;
4. printf("Введите размер пикселя\n");
5. while (scanf("%d", &(field->px)) != 1){
6. printf("Неверно\n");
7. fflush(stdin);
8. }
9. printf("Введите количество правил (Максимум - 20)\n");
10. while (scanf("%d", &(field->count\_rule)) != 1 || field->count\_rule > 20 || field->count\_rule < 1){
11. printf("Неверно\n");
12. fflush(stdin);
13. }
14. for (i=0;i<field->count\_rule;i++){
15. input\_rule(field, i);
16. }
17. printf("Введите количество итераций\n");
18. while(scanf("%d", &(field->iteration)) != 1){
19. printf("Неверно\n");
20. fflush(stdin);
21. }
22. }else return; }

Листинг 10 – функция для ввода информации о поле с консоли.

1. void change\_rule(Field \*field){
2. int i;
3. if (field && field->count\_rule != 0){
4. printf("Введите номер правила, которое хотите изменить\n");
5. while (scanf("%d", &i) != 1 || field->count\_rule < i || i < 1){
6. printf("Неверно\n");
7. fflush(stdin); }
8. i--;
9. input\_rule(field, i);
10. }else
11. return;}

Листинг 11 – функция, изменяющая ранее введённое правило.

1. void save\_file\_field\_main(Field \*field){
2. if (field){
3. int m;
4. FILE \*f;
5. f = fopen("files/field.dat", "wb");
6. if (f){
7. m = fwrite(field, sizeof(\*field), 1, f);
8. if (m != 0) printf("Успешное сохранение\n");
9. else printf("Сохранение не удалось\n");
10. fclose(f);
11. }else
12. printf("Сохранение не удалось\n");
13. }else
14. printf("Сохранение не удалось\n");}

Листинг 12 – функция для сохранения параметров поля в файл, из которого читает Langton\_ant.exe

Сохранение происходит в двоичном формате, по любому доступному пути.

1. void save\_file\_ants\_main(Ant \*ants, int n){
2. if (ants){
3. int m;
4. FILE \*f;
5. f = fopen("files/ants.dat", "wb");
6. if (f){
7. fwrite(&n, sizeof(n), 1, f);
8. m = fwrite(ants, sizeof(\*ants), n, f);
9. if (m != 0)
10. printf("Успешное сохранение\n");
11. else
12. printf("Сохранение не удалось\n");
13. fclose(f);
14. }else
15. printf("Сохранение не удалось\n");
16. }else
17. printf("Сохранение не удалось\n");}

Листинг 13 – функция для сохранения муравьёв в файл, из которого читает Langton\_ant.exe

Конкретнее сохранение происходит в файл по относительному пути files/ants.dat

1. void save\_file\_ants(Ant \*ants,char \*name, int n){
2. FILE \*f;
3. if (ants){
4. f = fopen(name, "wb");
5. if (f){
6. fwrite(&n, sizeof(n), 1, f);
7. if (fwrite(ants, sizeof(\*ants), n, f))
8. printf("Успешное сохранение\n");
9. else
10. printf("Сохранение не удалось\n");
11. fclose(f);
12. }else
13. printf("Сохранение не удалось\n");
14. }else
15. printf("Сохранение не удалось\n");
17. }

Листинг 14 – Сохранение муравьёв в файл с указанным путём (для дальнейшей загрузки)

1. void save\_file\_field(Field \*field, char \*name){
2. if (field && name){
3. int m;
4. FILE \*f;
5. f = fopen(name, "wb");
6. if (f){
7. m = fwrite(field, sizeof(\*field), 1, f);
8. if (m != 0)
9. printf("Успешное сохранение\n");
10. else
11. printf("Сохранение не удалось\n");
12. fclose(f);
13. }else
14. printf("Сохранение не удалось\n");
15. }else
16. printf("Сохранение не удалось\n"); }

Листинг 15 – Сохранение параметров поля в файл с указанным путём (для дальнейшей загрузки)

Конкретнее сохранение происходит в файл по относительному пути files/field.dat

1. void output\_rule(int rule, int \*color){
2. if (color){
3. switch (rule){
4. case 1:
5. printf("L(-90) ");
6. break;
7. case 2:
8. printf("R(90) ");
9. break;
10. case 3:
11. printf("N(0) ");
12. break;
13. case 4:
14. printf("U(180) ");
15. break;
16. }
17. printf("Цвет: (%d, %d, %d)\n", color[0], color[1], color[2]);
18. }else{
19. return;
20. }
21. }

Листинг 16 – Вывод в консоль данных о правиле и цвете этого правила

1. void output\_field(Field \*field){
2. if (field && count\_rule > 0){
3. int i;
4. printf("Размер пикселя - %d\n", field->px);
5. printf("Количество правил: %d\n", field->count\_rule);
6. for (i=0;i < field->count\_rule;i++){
7. printf("%d Правило: ", i+1);
8. output\_rule((field->rule)[i], (field->color)[i]);
9. }
10. printf("Количество итераций = %d\n", field->iteration);
11. }else{
12. printf("Нет данных\n");
13. return;
14. }
15. }

Листинг 17 – Вывод в консоль данных о поле

1. void load\_file\_ants(Ant \*ants, char \*name, int \*n){
2. if (ants && name){
3. FILE \*f;
4. f = fopen(name, "rb");
5. if (f){
6. fread(n, sizeof(\*n), 1, f);
7. fread(ants, sizeof(\*ants), 40, f);
8. if (\*n != 0){
9. printf("Успешная загрузка\n");
10. }
11. else
12. printf("Загрузка не удалась\n");
13. fclose(f);
14. }else
15. printf("Загрузка не удалась\n");
17. }else
18. printf("Загрузка не удалась\n");
19. }

Листинг 18 – функция, для загрузки муравьёв из файла по любому пути.

1. void load\_file\_field(Field \*field, char \*name){
2. if (field && name){
3. FILE \*f;
4. int m;
5. f = fopen(name, "rb");
6. if (f){
7. m = fread(field, sizeof(\*field), 1, f);
8. if (m != 0){
9. printf("Успешная загрузка\n");
10. }
11. else
12. printf("Загрузка не удалась\n");
13. fclose(f);
14. }else
15. printf("Загрузка не удалась\n");
17. }else
18. printf("Загрузка не удалась\n");
19. }

Листинг 19 – функция, для загрузки параметров поля из файла по любому пути.

### 3.1.3 Функция main

* В ней хранится текущее состояние муравьёв и поля.
* Функция просит пользователя ввести число, чтобы выбрать, какую функцию задействовать
* Оператором switch выбирается одно из действий

Листинг представлен в приложении Б.2

## 3.2 Программа Langton\_ant

### 3.2.1 Функции программы Langton\_ant

1. void load\_file\_ants(Ant \*ants, int \*n){
2. FILE \*f;
3. int m;
4. if (ants){
5. f = fopen("menu/files/ants.dat", "rb");
6. if (f){
7. fread(n, sizeof(\*n), 1, f);
8. \*n = fread(ants, sizeof(\*ants), 40, f);
10. fclose(f);
11. }else
12. return;
13. }else
14. return;
15. }

Листинг 20 – Функция для загрузки муравьёв из файла

1. void load\_file\_field(Field \*field){
2. FILE \*f;
3. if (field){
4. f = fopen("menu/files/field.dat", "rb");
5. if (f){
6. fread(field, sizeof(\*field), 1, f);
7. fclose(f);
8. }else
9. return;
10. }else
11. return;
12. }

Листинг 21 – Функция для загрузки параметров поля из файла

1. Ant ants[40] = {0};
2. Field field;
3. char color[1000][720] = {0};
4. int iter = 0, n = 0, j=0, width=0, height=0, i, flag=1;

Листинг 22 – Объявление переменных, загрузка в эти переменные данных из файла, задача цвета фона

Объявляются переменная структуры Ant в строке 106, как массив муравьёв, максимальное количество муравьёв в программе – 40. Массив color строки 108 обозначает максимальный размер поля, при размер поля в пикселях, в случае если программа будет загружать размер пикселя из файла, равный 1, будет использоваться весь массив, в ином случае будет использована лишь его часть. Далее задаются служебные переменные типа int

1. hWnd = CreateWindow (
2. "GLSample", "OpenGl Sample",
3. WS\_CAPTION | WS\_POPUPWINDOW | WS\_VISIBLE,
4. 0, 0, 1000, 720,
5. NULL, NULL, hInstance, NULL);

Листинг 23 – задание размеров поля (1000 на 720)

Стандартный код OpenGl, я только изменил параметры отвечающие за размер поля в строке 113

1. load\_file\_field(&field);
2. load\_file\_ants(ants, &n);
3. width = 1000 / field.px;
4. height = 720 / field.px;
5. for (i=0;i<n;i++){
6. ants[i].x /= field.px;
7. ants[i].y /= field.px;
8. }
9. glMatrixMode(GL\_PROJECTION);
10. glLoadIdentity();
11. glOrtho(0.0, width, 0.0, height, -1.0, 1.0);

Листинг 24 – загрузка данных из фалов, переход к дискретному пространству

В строках 115 и 116 происходит загрузка данных с помощью функций load\_file\_field и load\_file\_ants данные загружаются в переменные field, ants и в служебную переменную n, отвечающую за размер массива ants. Далее, в строках 117 и 118, делим переменные width и height на размер пикселя для задачи дискретной модели. Также мы приводим координаты всех муравьёв к дискретной модели в строках 119-122. В строке 125 мы изменяем границы координатных осей по x и по y из (-1;1) до (0;width) и (0;height).

Листинги 25, 26 и Б.1 выполняются каждую отрисовку окна OpenGl.

1. color[ants[j].x][ants[j].y] = (color[ants[j].x][ants[j].y] + 1) % field.count\_rule;

Листинг 25 – смена цвета в массиве

1. if (flag){
2. for (x=0;x<width;x++)
3. for(y=0;y<height;y++){
4. glBegin (GL\_QUADS);
5. glColor3f (field.color[color[x][y]][0], field.color[color[x][y]][1], field.color[color[x][y]][2]);
6. glVertex2f (1 + x, y);
8. glColor3f (field.color[color[x][y]][0], field.color[color[x][y]][1], field.color[color[x][y]][2]);
9. glVertex2f (x, y);
11. glColor3f (field.color[color[x][y]][0], field.color[color[x][y]][1], field.color[color[x][y]][2]);
12. glVertex2f (x, 1 + y);
14. glColor3f (field.color[color[x][y]][0], field.color[color[x][y]][1], field.color[color[x][y]][2]);
15. glVertex2f (1 + x, 1 + y);
16. glEnd ();
17. }
18. }

Листинг 26 – Отрисовка всего поля

Код в 26 листинге сопоставляет индекс цвета из переменной field и значение в массиве color для определения цвета. После с помощью функции glBegin и параметром GL\_QUADS мы начинаем рисовать квадрат, он рисуется с помощью функций glColor3f и glVertex2f в первую мы передаём цвет квадрата, во вторую мы задаём координаты 4 точек по часовой стрелке.

Последняя функция вынесена в Приложение Б.1. Стоит отметить, что ходы муравьёв идут линейно, по порядку. Например стоят два муравья на одной клетке, сначала ходит муравей, которого раньше добавили, ходит по цвету клетки А и меняет его на Б, следующий муравей будет ходит уже по правилу цвета Б.

# 4 Тестирование

## 4.1 Программа menu

### 4.1.1 Проверка возможности добавить муравья

Откроем файл menu.exe и следуя инструкции добавим муравья. Также попробуем ввести неправильные данные (смотрит в 255 или просто текст).

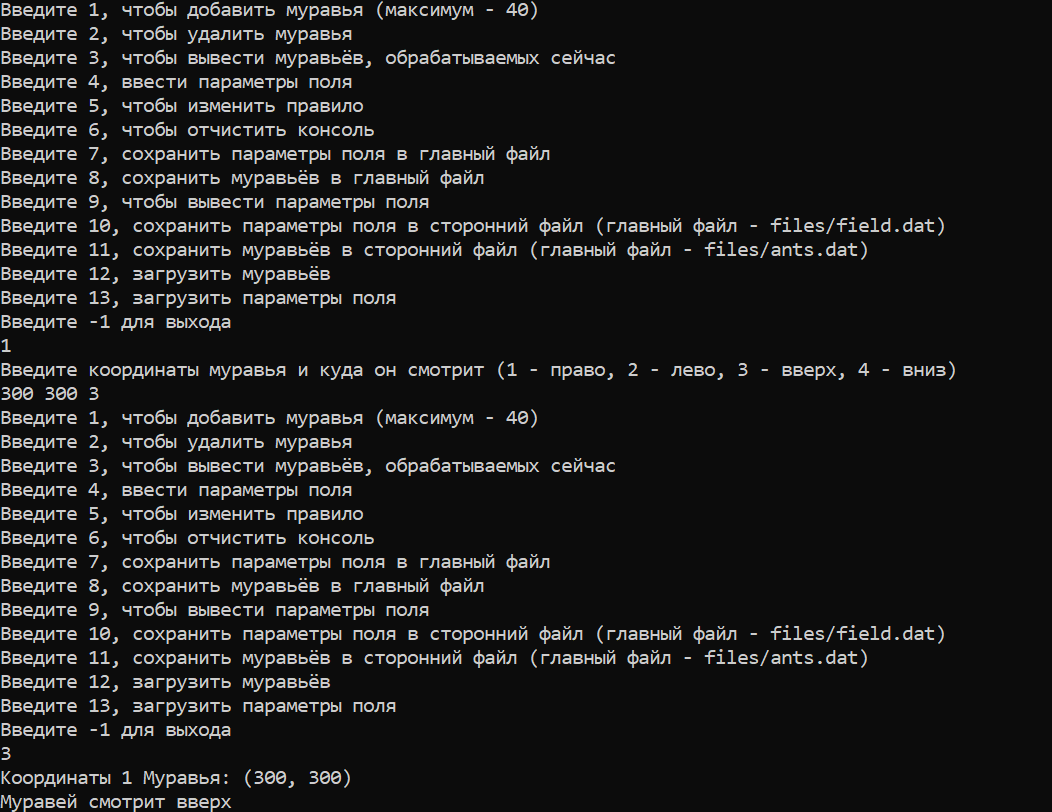


Рисунок 8 – начало программы menu и успешная попытка добавить муравья

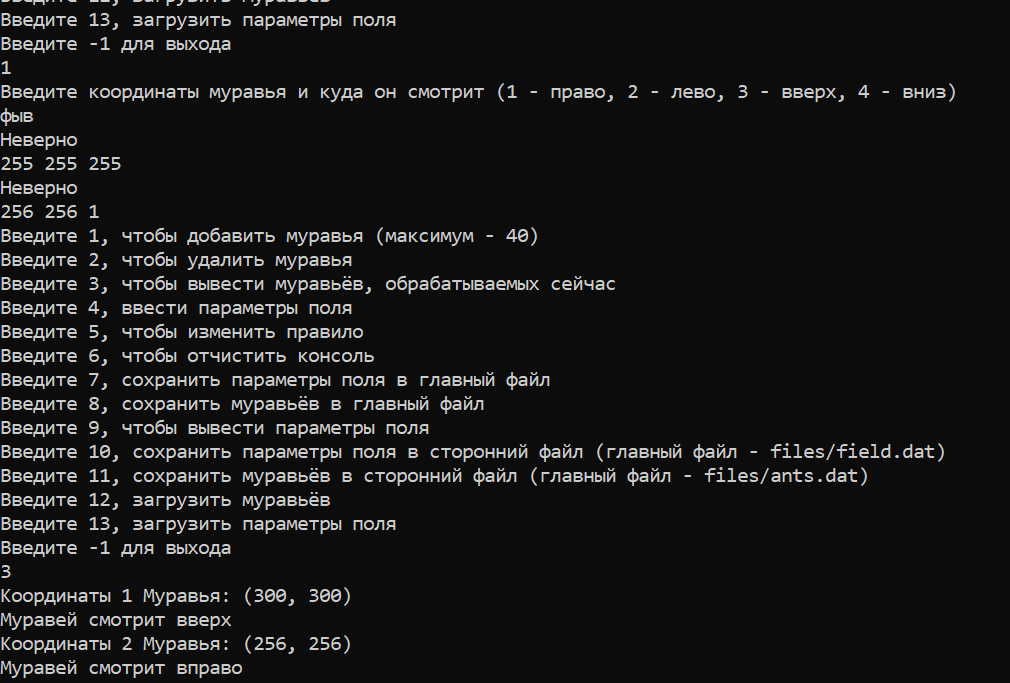


Рисунок 9 – Попытка неправильного ввода данных (правило может быть от 1 до 4)

4.1.2 Проверка возможности удалить муравья

Проделаем те же действия что и в п. 4.1.1 но с удалением.

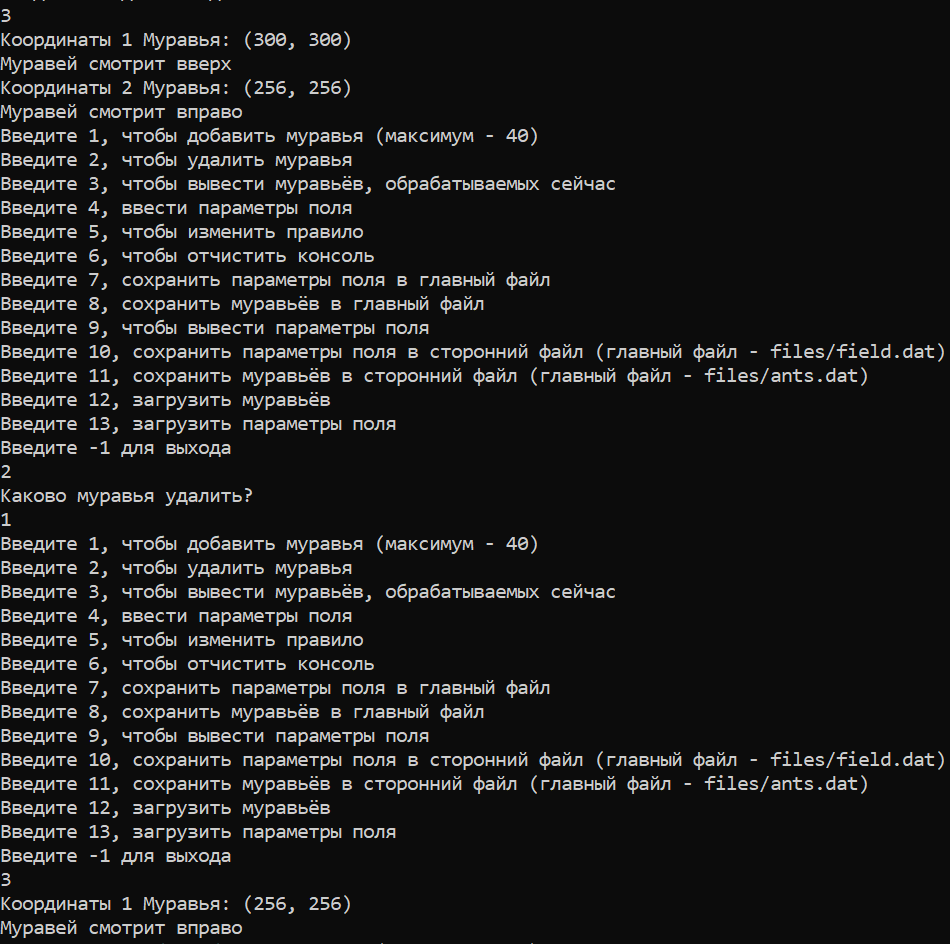


Рисунок 10 – Успешное удаление 1го муравья

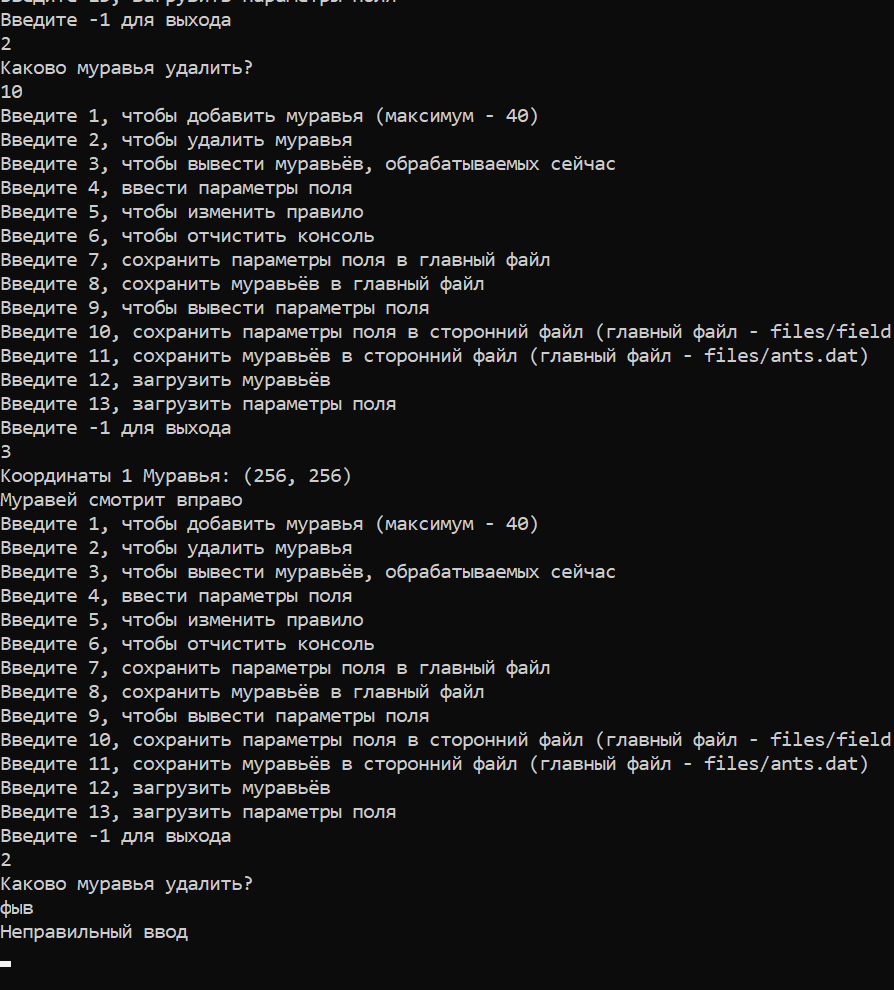


Рисунок 11 – реагирование на неправильный ввод (не существует 10 муравья на данный момент)

### 4.1.3 Проверка ввода параметров поля:

Проделаем те же действия что и в п. 4.1.1 но с параметрами поля.

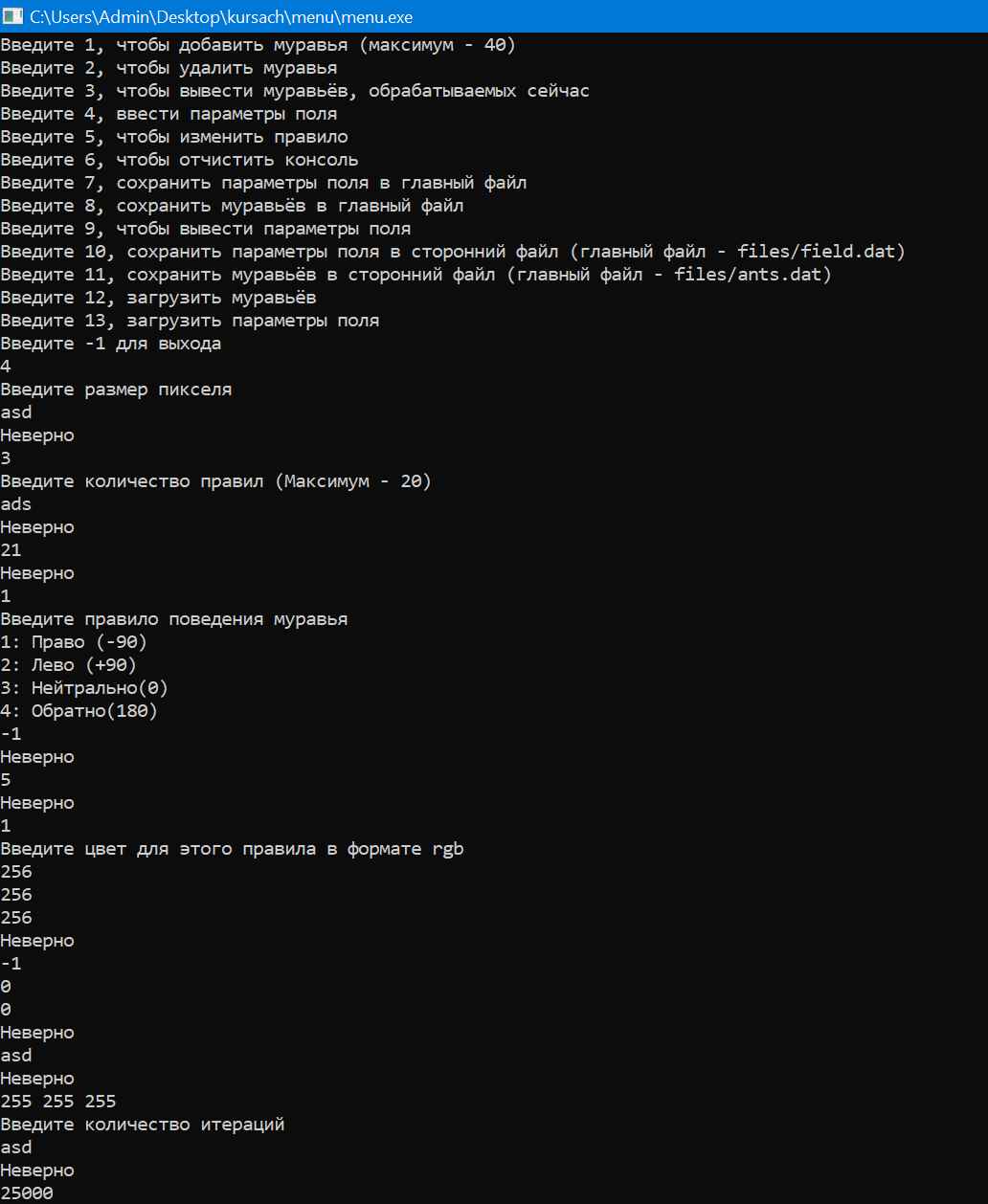


Рисунок 12 – Успешный ввод параметров поля, с проверкой на неправильный ввод.

### 4.1.4 Проверка функции, которая измененяет правила по индексу

Проделаем те же действия что и в п. 4.1.1 но с этой функцией

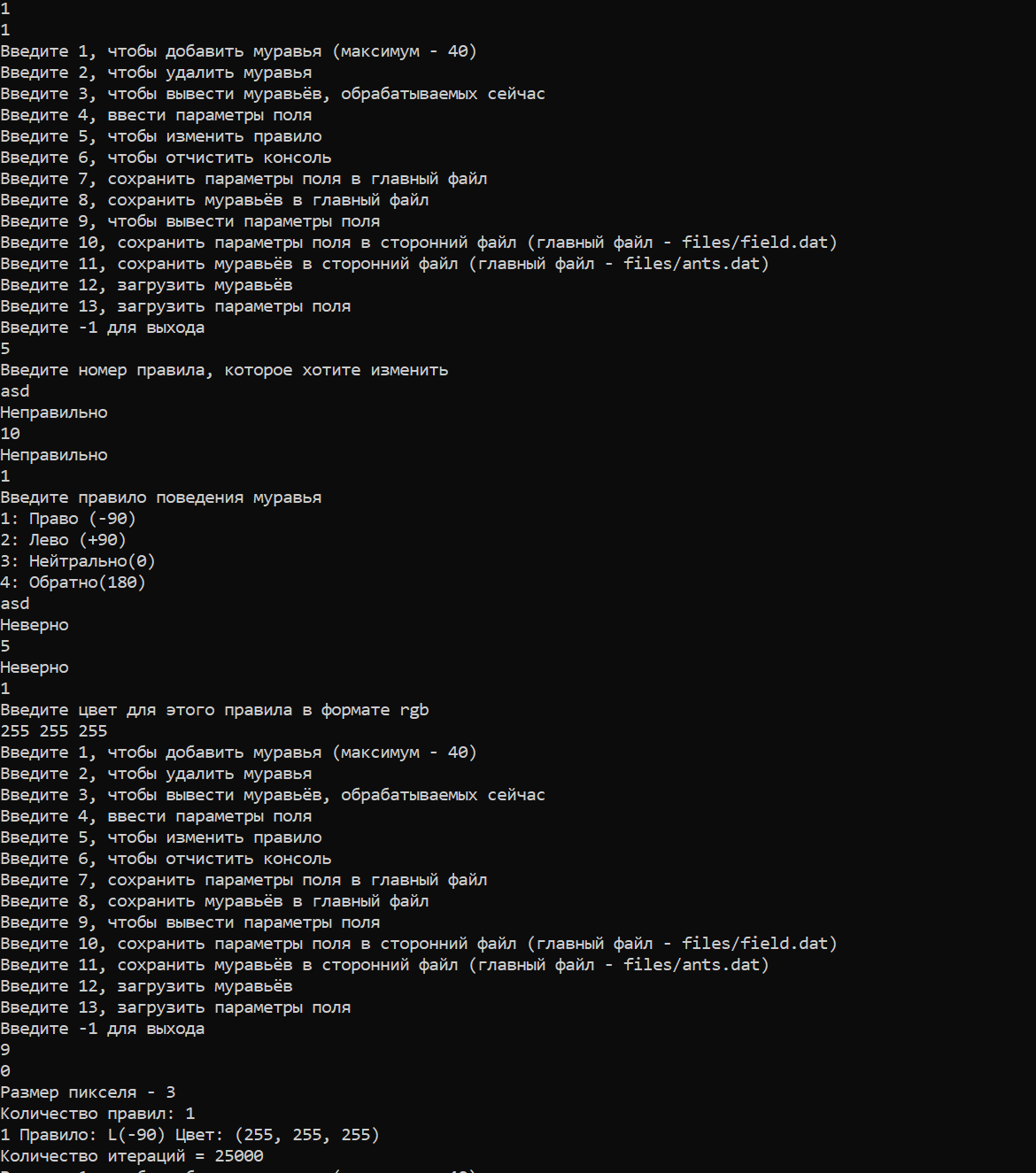


Рисунок 13 – Проверка функции изменения правила с неправильным вводом

### 4.1.5 Сохранение в главный файл:

Далее попробуем сохранять в файл, из которого читает программа Langton\_ant.exe

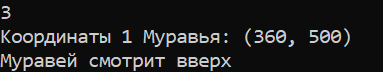


Рисунок 14 – сохраняемые муравьи

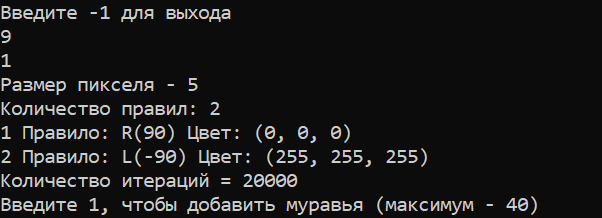


Рисунок 15 – параметры поля

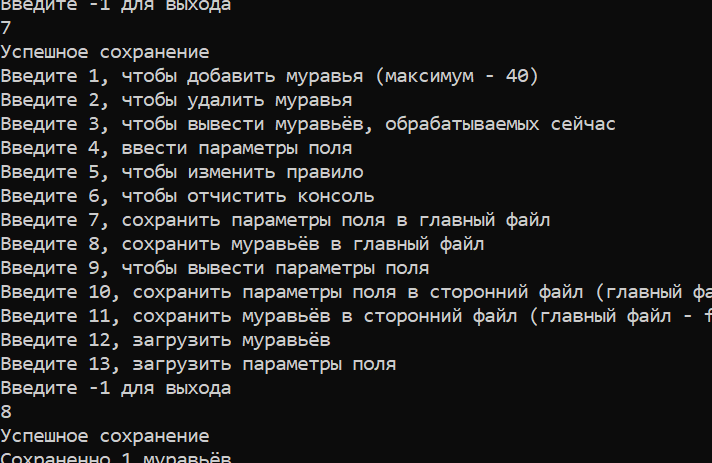


Рисунок 16 – сохранение

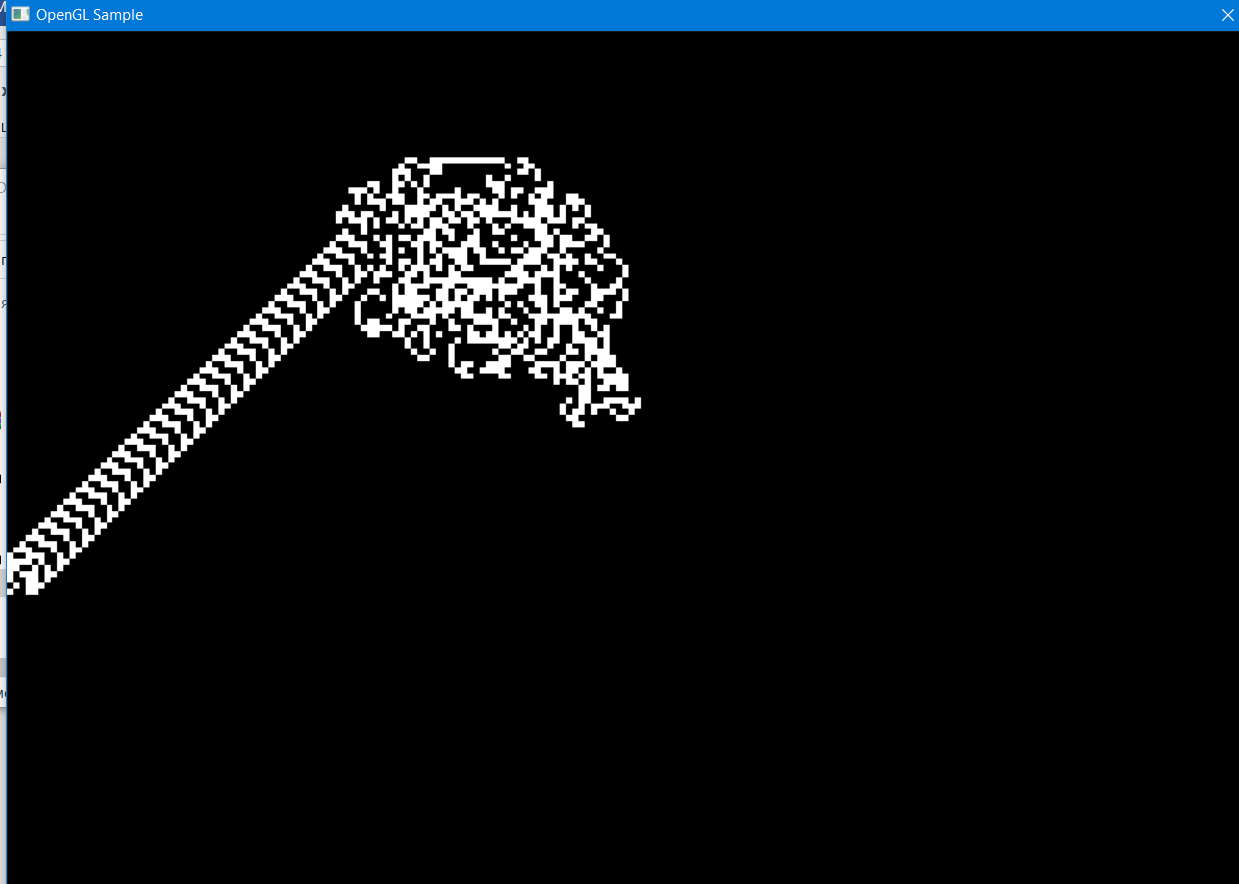


Рисунок 17 – запуск Langton\_ant.exe (соответствует введённому значению)

### 4.1.6 Сохранение и загрузка параметров поля и муравьёв в сторонние файлы

Повторим действия с п. 4.1.5

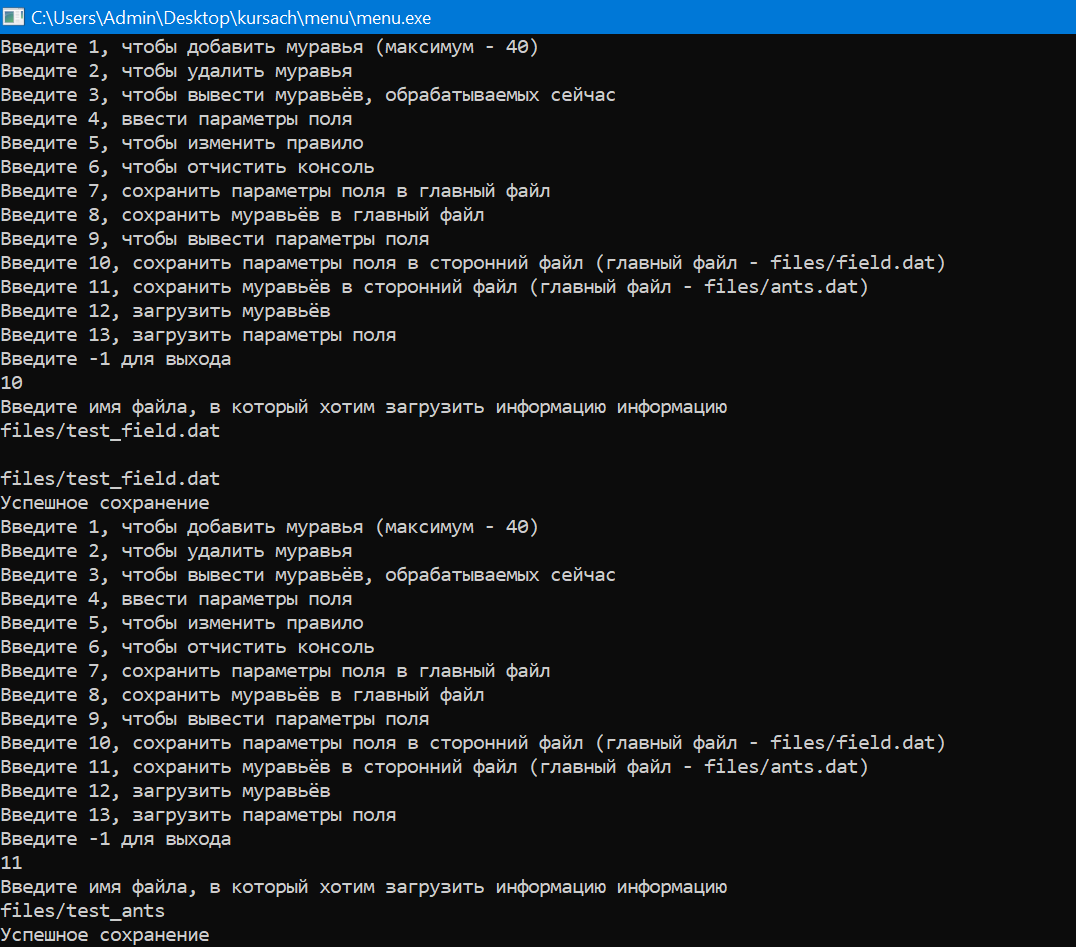


Рисунок 18 – сохранения файлов test\_field и test\_ants по пусти files/”имя\_файла”

Программа была запущена повторно, чтобы проверить, успешно ли программа считывает и сохраняет данные. Проверка находится в приложениях А.2.1 и А.2.2

## 4.2 Программа Langton\_ant

Для тестирования были загружены шаблоны поведения муравья

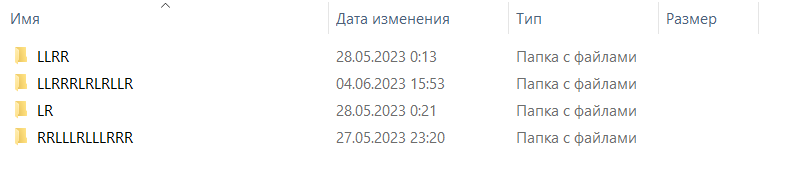


Рисунок 20 – Созданные шаблоны

Тестирование проводилось с использованием шаблона на рисунке 21 и собственным набором муравьёв с правилом RL.

Для тестирования был использован тот же набор правил, что и на рисунке 21, но отличались цвета. Суть шаблона в симметричном росте, который наблюдается на рисунке 22.

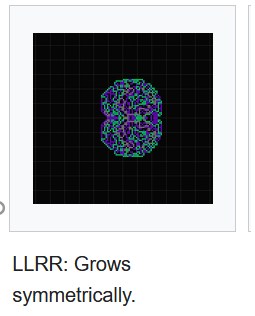


Рисунок 21 – Шаблон LLRR

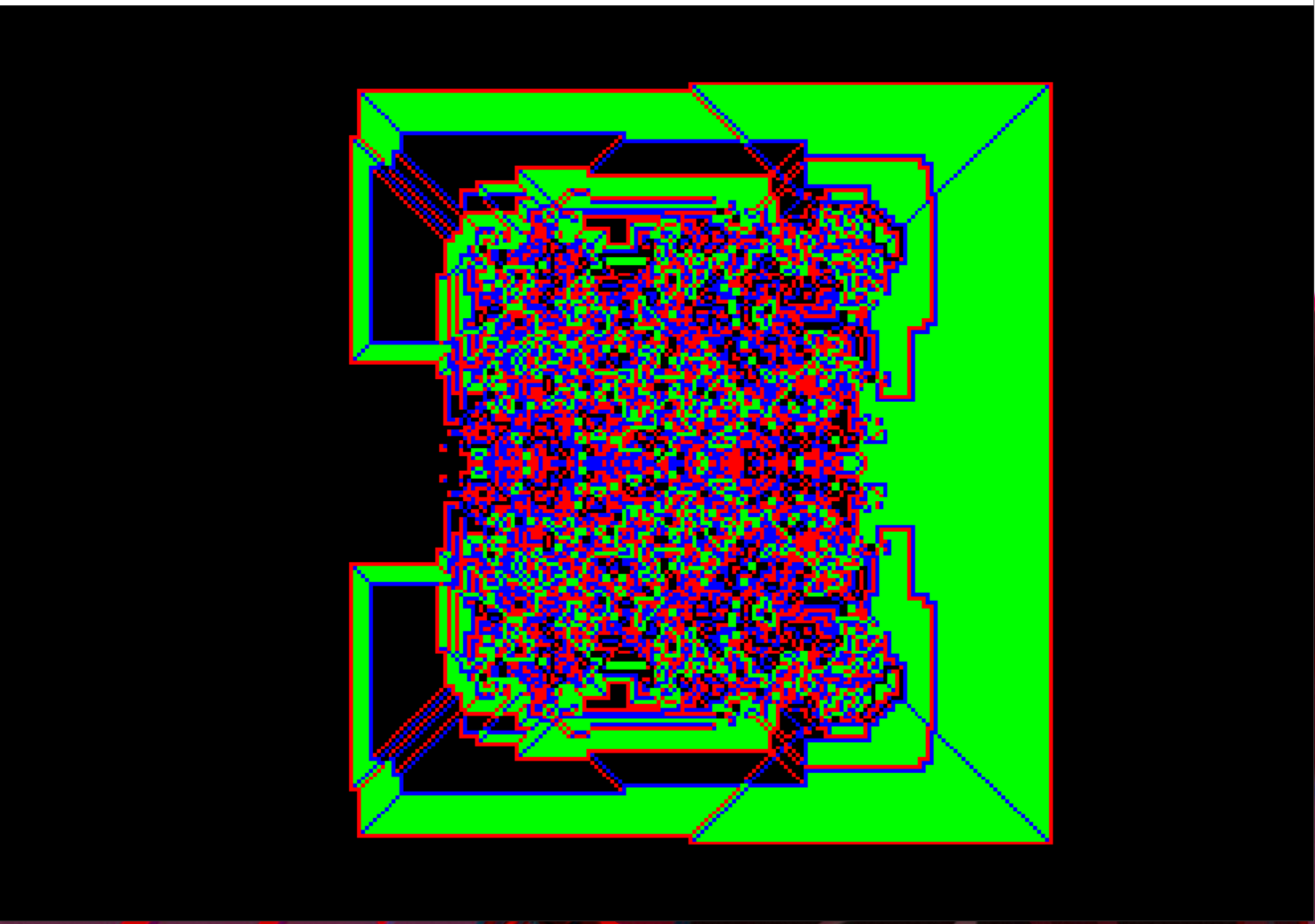


Рисунок 22 – процесс симметричного роста шаблона LLRR

Зададим 4 муравья, которые будут близко стоять с простыми правилами.

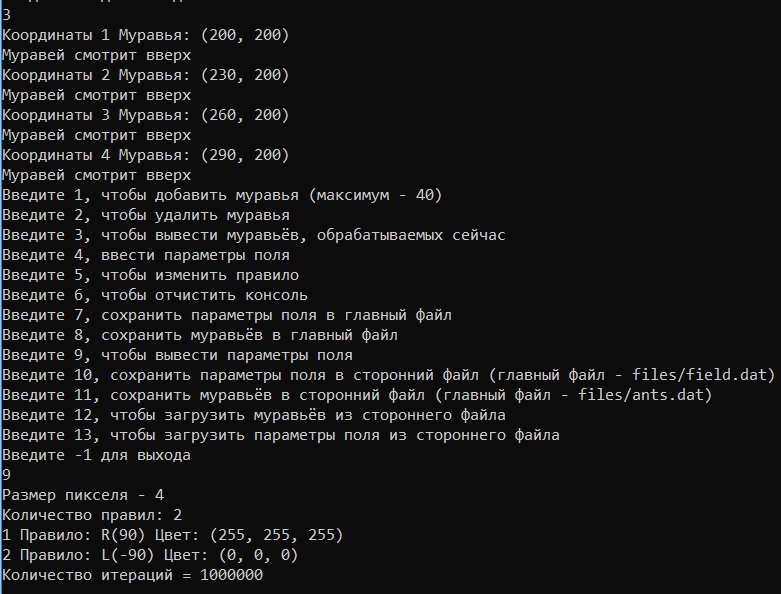


Рисунок 24 – Заполнение программы Langton\_ant.exe

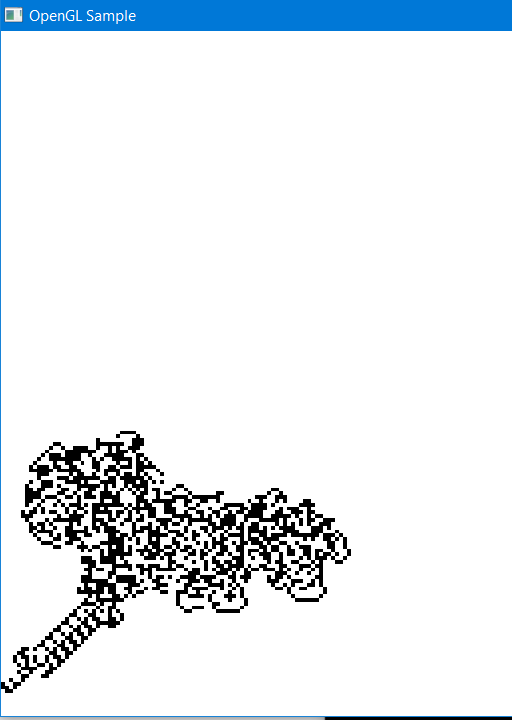


Рисунок 25 – результат работы нескольких муравьёв

# Заключение

Реализованы 2 программы на языке программирования “C”, одна из которых работает с файлами и отвечает за сохранение загрузку и изменение входных параметров для основной программы, которая запускает симуляцию. Программу можно улучшить заданием размера окна OpenGl, ускорением алгоритма и добавлением нескольких клеточных автоматов на выбор.

# 5 Источники

1. Высшая школа теоретической механики, простейшие клеточные автоматы - <http://tm.spbstu.ru/Простейшие_клеточные_автоматы>

2. Хабр, Муравей Лэнгтона — загадочный клеточный автомат - <https://habr.com/ru/post/599275/>

3. Википедия, Муравей Лэнгтона –

<https://ru.wikipedia.org/wiki/Муравей_Лэнгтона>

4. Гитхаб, StanislavPetrovV

<https://github.com/StanislavPetrovV/Langton-ant>

5. JOSEPH PETITTI, персональный сайт

<https://josephpetitti.com/ant>

6. Langton’s ant

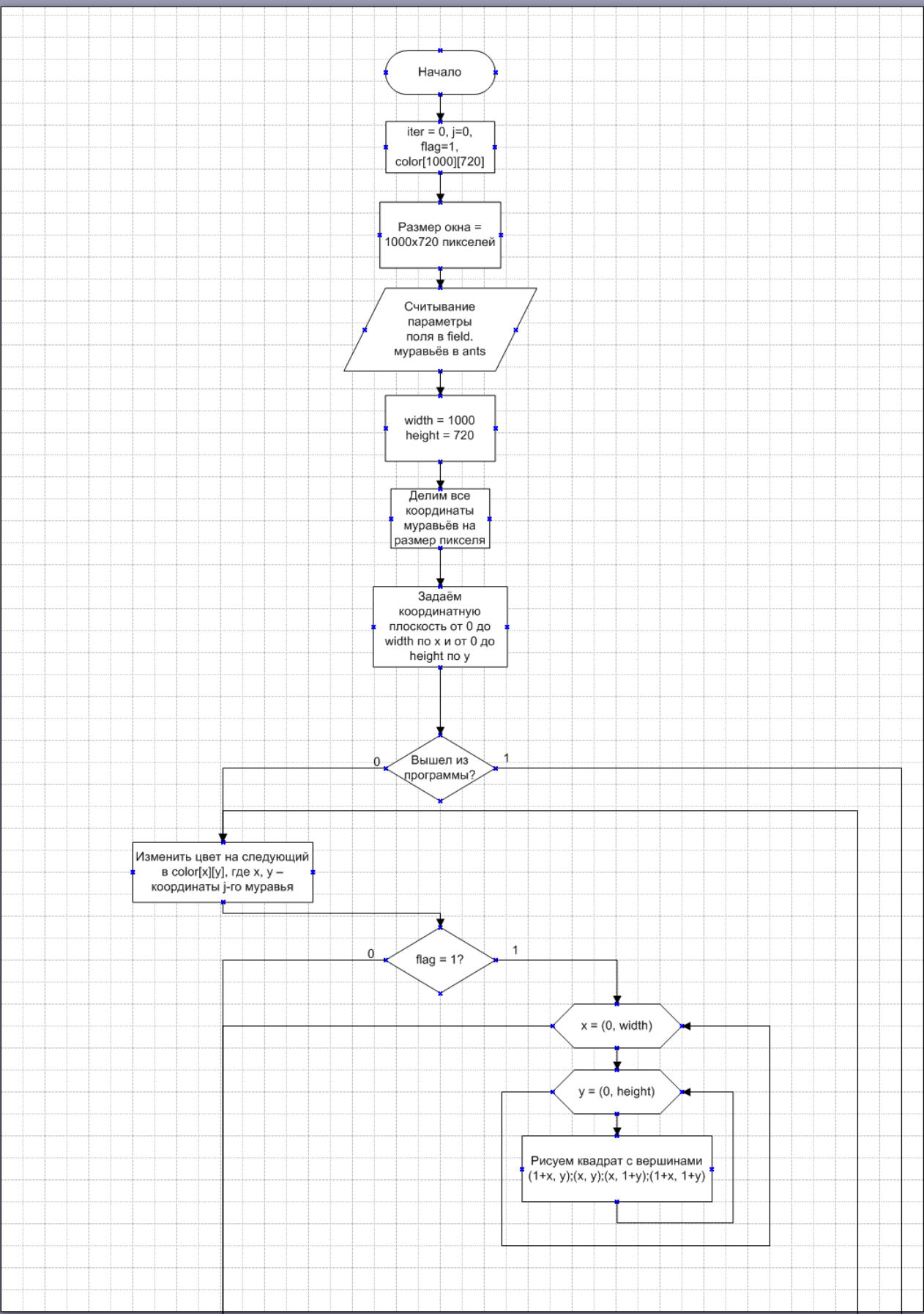
<http://www.langtonant.com/>

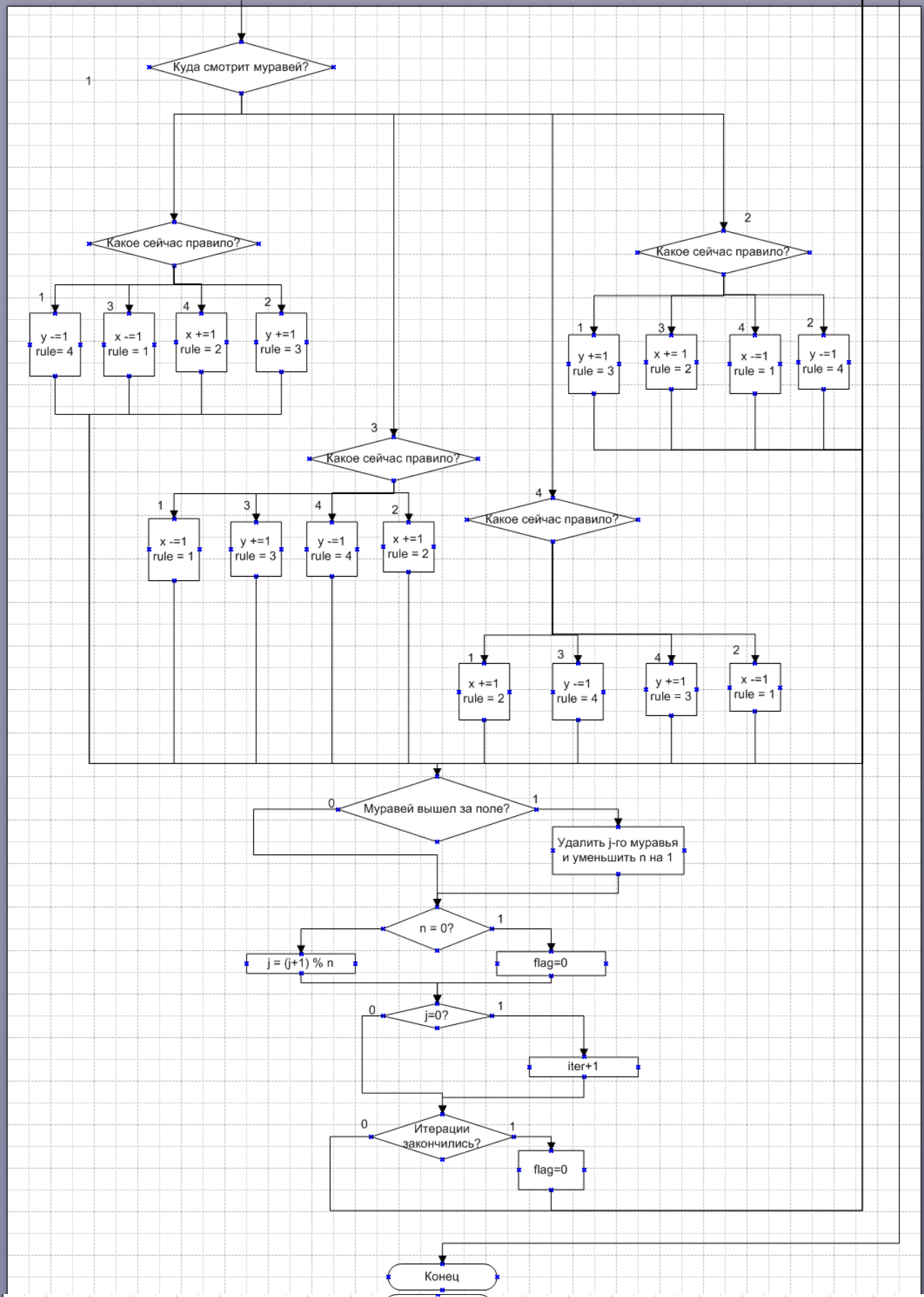
7. Википедия, Клеточный автомат

<https://ru.wikipedia.org/wiki/Клеточный_автомат>

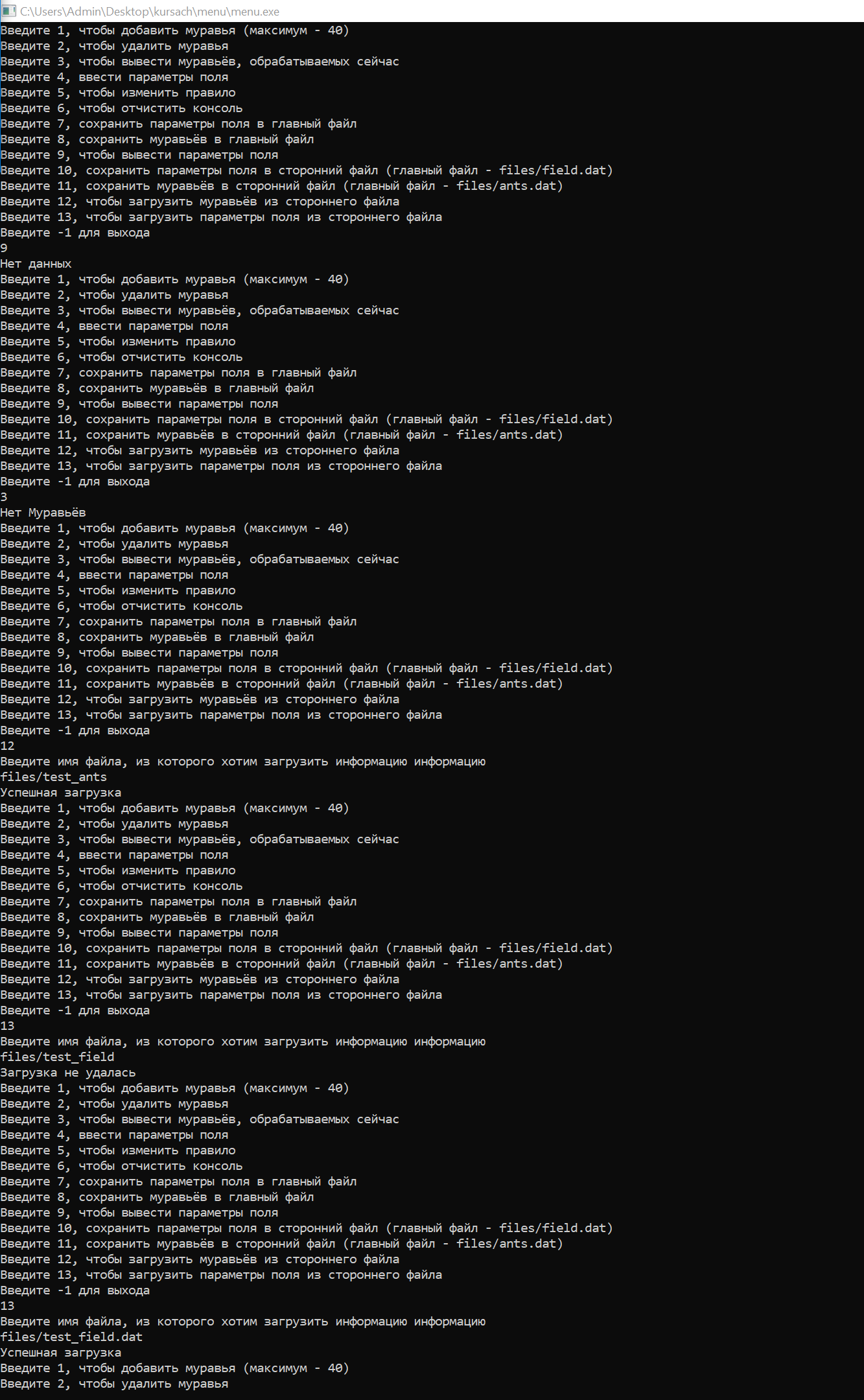
# 6 Приложения

## Приложения А. Рисунки

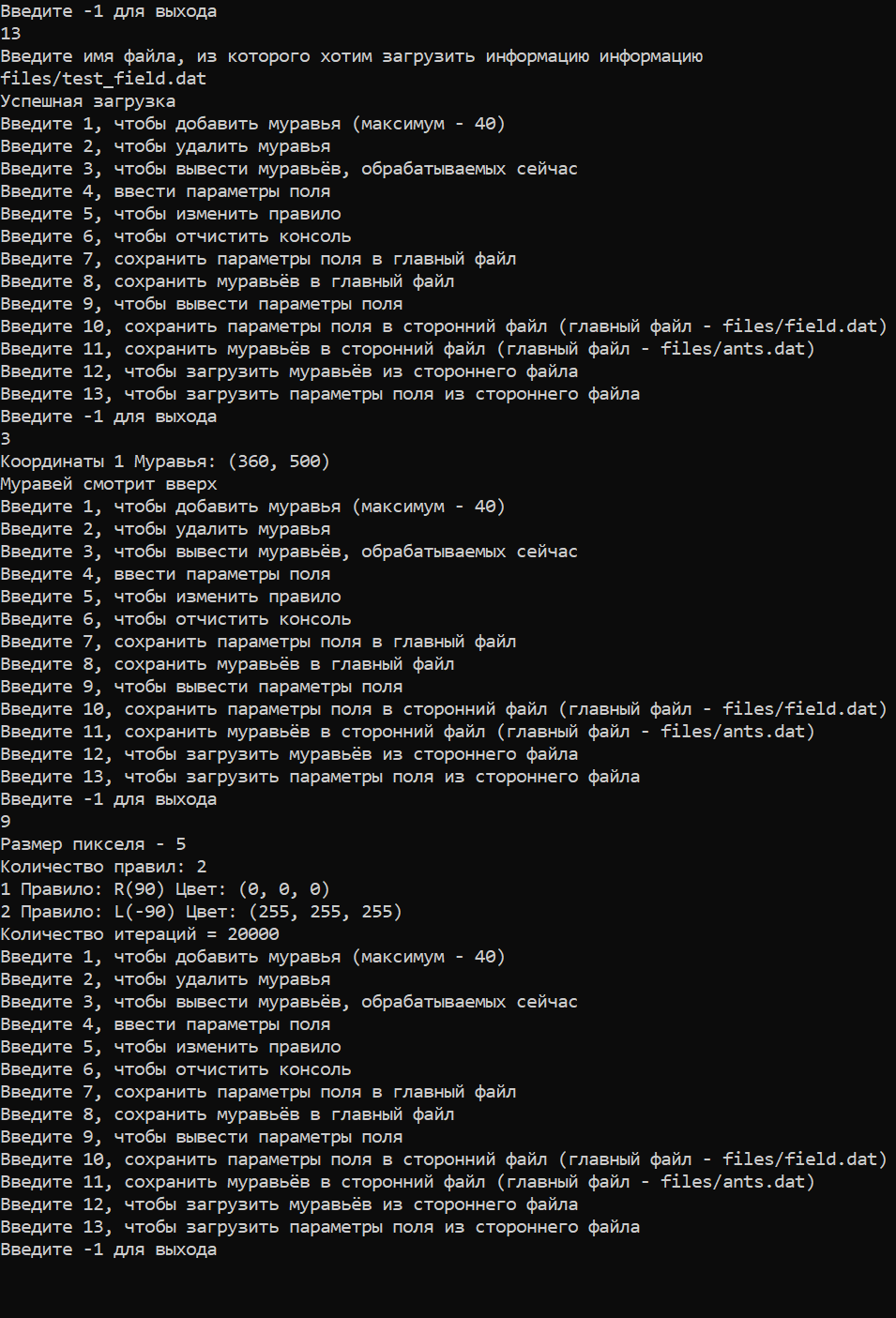
Приложение А.1.1 – блок-схема алгоритма работы Langton\_ant



Приложение А.1.2 – блок-схема алгоритма работы Langton\_ant



Приложение А.2.1 – результат загрузки из файла



Приложение А.2.2 - результат загрузки из файла

## Приложения Б. Листинги

1. switch (ants[j].rule){
2. case 1:
3. switch(field.rule[color[ants[j].x][ants[j].y]]){
4. case 1:
5. ants[j].y -= 1;
6. ants[j].rule = 4;
7. break;
8. case 2:
9. ants[j].y += 1 ;
10. ants[j].rule = 3;
11. break;
12. case 3:
13. ants[j].x -= 1 ;
14. ants[j].rule = 1;
15. break;
16. case 4:
17. ants[j].x += 1 ;
18. ants[j].rule = 2;
19. break;
20. }
21. break;
22. case 2:
23. switch(field.rule[color[ants[j].x][ants[j].y]]){
24. case 1:
25. ants[j].y += 1 ;
26. ants[j].rule = 3;
27. break;
28. case 2:
29. ants[j].y -= 1 ;
30. ants[j].rule = 4;
31. break;
32. case 3:
33. ants[j].x += 1 ;
34. ants[j].rule = 2;
35. break;
36. case 4:
37. ants[j].x -= 1 ;
38. ants[j].rule = 1;
39. break;
40. }
41. break;
42. case 3:
43. switch(field.rule[color[ants[j].x][ants[j].y]]){
44. case 1:
45. ants[j].x -= 1 ;
46. ants[j].rule = 1;
47. break;
48. case 2:
49. ants[j].x += 1 ;
50. ants[j].rule = 2;
51. break;
52. case 3:
53. ants[j].y += 1 ;
54. ants[j].rule = 3;
55. break;
56. case 4:
57. ants[j].y -= 1 ;
58. ants[j].rule = 4;
59. break;
60. }
61. break;
62. case 4:
63. switch(field.rule[color[ants[j].x][ants[j].y]]){
64. case 1:
65. ants[j].x += 1 ;
66. ants[j].rule = 2;
67. break;
68. case 2:
69. ants[j].x -= 1 ;
70. ants[j].rule = 1;
71. break;
72. case 3:
73. ants[j].y -= 1 ;
74. ants[j].rule = 4;
75. break;
76. case 4:
77. ants[j].y += 1 ;
78. ants[j].rule = 3;
79. break;
80. }
81. break;
82. }

Приложение Б.1 – куда двинется муравей и куда будет смотреть после итерации

1. int main(){
2. Ant ants[40] = {0};
3. Field field = {0};
4. int n=0, i=0, menu=100;
5. char name[100];
6. system("chcp 1251 > nul");
7. while (menu){
9. printf("Введите 1, чтобы добавить муравья (максимум - 40)\n");
11. printf("Введите 2, чтобы удалить муравья\n");
13. printf("Введите 3, чтобы вывести муравьёв, обрабатываемых сейчас\n");
15. printf("Введите 4, ввести параметры поля\n");
17. printf("Введите 5, чтобы изменить правило\n");
19. printf("Введите 6, чтобы отчистить консоль\n");
21. printf("Введите 7, сохранить параметры поля в главный файл\n");
23. printf("Введите 8, сохранить муравьёв в главный файл\n");
25. printf("Введите 9, чтобы вывести параметры поля\n");
27. printf("Введите 10, сохранить параметры поля в сторонний файл (главный файл - files/field.dat)\n");
29. printf("Введите 11, сохранить муравьёв в сторонний файл (главный файл - files/ants.dat)\n");
31. printf("Введите 12, чтобы загрузить муравьёв из стороннего файла\n");
33. printf("Введите 13, чтобы загрузить параметры поля из стороннего файла\n");
35. printf("Введите -1 для выхода\n");
37. while (scanf("%d", &menu) != 1){
38. printf("Неправильный ввод\n");
39. fflush(stdin);
40. }
42. switch(menu){
43. case 1: /\* Добавить муравья\*/
44. add\_ant(ants, &n);
45. break;
47. case 2: /\* Удалить элемент \*/
48. printf("Каково муравья удалить?\n");
49. while (scanf("%d", &i) != 1){
50. printf("Неправильный ввод\n");
51. fflush(stdin);
52. }
53. del\_ant(ants, i, &n);
54. break;
56. case 3: /\* Вывод \*/
57. if (n == 0){
58. printf("Нет Муравьёв\n");
59. }
60. for (i=0;i<n;i++){
61. printf("Координаты %d Муравья: ", i + 1);
62. output(&ants[i]);
63. }
64. break;
66. case 4:
67. input\_field(&field);
68. break;
70. case 5:
71. change\_rule(&field);
72. break;
74. case 6:
75. system("cls");
76. break;
78. case 7:
79. save\_file\_field\_main(&field);
80. break;
81. case 8:
82. save\_file\_ants\_main(ants, n);
83. printf("Сохраненно %d муравьёв\n", n);
84. break;
86. case 9:
87. output\_field(&field);
88. break;
89. case 10:
90. printf("Введите имя файла, в который хотим загрузить информацию информацию\n");
91. while (scanf("%s", name) != 1){
92. printf("Wrong input, try again\n");
93. }
94. printf("\n%s\n", name);
95. save\_file\_field(&field, name);
96. break;
98. case 11:
99. printf("Введите имя файла, в который хотим загрузить информацию информацию\n");
100. while (scanf("%s", name) != 1){
101. printf("Wrong input, try again\n");
102. }
103. save\_file\_ants(ants, name, n);
104. break;
105. case 12:
106. printf("Введите имя файла, из которого хотим загрузить информацию информацию\n");
107. while (scanf("%s", name) != 1){
108. printf("Wrong input, try again\n");
109. }
110. load\_file\_ants(ants, name, &n);
111. break;
112. case 13:
113. printf("Введите имя файла, из которого хотим загрузить информацию информацию\n");
114. while (scanf("%s", name) != 1){
115. printf("Wrong input, try again\n");
116. }
117. load\_file\_field(&field, name);
118. break;
119. case 14:
120. field.iteration = 1000000;
121. break;
122. case -1:
123. printf("Спасибо, что воспользовались\n");
124. system("pause");
125. return 0;
126. default:
127. printf("Пока нет такой функции(\n");
128. break;
130. }
132. }
133. system("pause");
134. return 0;
135. }

Приложение Б.2 – главная функция программы menu