МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский политехнический университет»
Московский Политех

Факультет информационных технологий
Кафедра «Инфокогнитивные технологии»

КУРСОВАЯ РАБОТА

по дисциплине «Языки программирования»

Тема курсовой работы
«Разработка компьютерной программы с GUI средствами языка C++»

Студент группы 221-329 Сафронов Евгений Максимович

(подпись студента)

Руководитель курсовой работы доц. Рысин М.Л.

(подпись руководителя)

Работа представлена к защите «14» января 2023 г.

Допущен к защите «14» января 2023 г.

Москва – 2023

МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский политехнический университет»

Московский Политех

Факультет информационных технологий
Кафедра «Инфокогнитивные технологии»

«Утверждаю»
Заведующий кафедрой
_____ Е.А. Пугова
«03» ноября 2022 г.

ЗАДАНИЕ
на выполнение курсовой работы по дисциплине
«Языки программирования»

Студент Сафронов Евгений Максимович Группа 221-329

Тема работы:
«Разработка компьютерной программы с GUI средствами языка C++»

Исходные данные:
Разработать компьютерную программу с графическим интерфейсом пользователя (GUI) в соответствии с общими требованиями и индивидуальным вариантом (см. приложение 1 к заданию) на ЯВУ C++.
Использовать IDE Visual Studio Community (или аналог) с установленным компонентом «Поддержка C++/CLI для средств сборки» (рабочая нагрузка «Разработка классических приложений на C++»).
Сформировать функциональную схему программы и техническое задание в соответствии с ГОСТ 19.201-78.
Представить отчет по курсовой работе в виде пояснительной записки.

Содержание пояснительной записки:
Титульный лист.
Задание на курсовую работу.
Список условных сокращений.
Оглавление.
Введение.
1. Математическая модель решения. Функциональная схема программы.
2. Техническое задание.
3. Код программы на языке C++.
4. Контрольный пример работы программы.
5. Заключение.
Список использованных источников.

Срок представления к защите курсовой работы: «14» января 2023 г.

Задание на курсовую работу выдал _____ (Рысин М.Л.)
«03» ноября 2023 г.

Задание на курсовую работу получил _____ (Сафронов Е.М.)

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Цель курсовой работы: разработать компьютерную игру с GUI – «Жизнь» средствами языка C++.

"Жизнь" — это клеточный автомат, созданный Джоном Конвеем в 1970 году. Игра интересна и сложна по ряду причин. Во-первых, она демонстрирует основные принципы нелинейной динамики, которые проявляются в различных системах, включая жизнь в реальном мире. Во-вторых, она предлагает множество возможностей для исследования, включая исследование различных типов паттернов, которые могут возникнуть в игре, и исследование влияния различных параметров на поведение системы. В-третьих, игра "Жизнь" является одним из первых и наиболее известных примеров клеточного автомата, что делает ее интересной для изучения истории компьютерной науки и искусства.

Состав пояснительной записки:

1. Введение
2. Математическая модель решения. Функциональная схема программы
3. Техническое задание
4. Код программы на языке c++
5. Контрольный пример работы программы
6. Заключение
7. Список из 5 источников информации

Задачи курсовой работы:

1. Разработать математическую модель решения вычислительной задачи, привести функциональную схему программы;
2. Сформировать техническое задание на разработку программы в соответствии с ГОСТ 19.201-78;
3. Реализовать код программы на языке высокого уровня C++, протестировать его и отладить;

4. Реализовать контрольный пример работы программы, начиная с открытия, показать все этапы работы приложения.

1. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ РЕШЕНИЯ. ФУНКЦИОНАЛЬНАЯ СХЕМА ПРОГРАММЫ

Формулировка индивидуального варианта 23(Задача №991): Игра моделирует жизнь поколений гипотетической колонии живых клеток, которые выживают, размножаются или погибают в соответствии со следующими правилами. Клетка выживает, если и только если она имеет двух или трёх соседей из восьми возможных. Если у клетки только один сосед или вовсе ни одного, она погибает в изоляции. Если клетка имеет четырех или более соседей, она погибает от перенаселения. В любой пустой позиции, у которой ровно три соседа, в следующем поколении появляется новая клетка.

Примечание к индивидуальному варианту: пользователь может задавать размер поля и расставлять живые клетки на этом поле; реализовать возможность игры пользователем и компьютером.

Описание алгоритмов:

Форма обладает элементами управления игрой и кнопками навигации, открывающими дополнительные окна: «Об игре», «Об авторе».

Вкладка «Игра «Жизнь»» содержит следующие элементы: кнопки: «Сохранить игру», «Загрузить игру», «Очистить поле», «Случайно заполнить поле», «Начать», «Об игре», «Об авторе», надпись отображающую количество строк в игровом поле, надпись отображающую количество столбцов в игровом поле, ползунок для изменения количества строк в игровом поле, ползунок для изменения количества столбцов в игровом поле, игровое поле.

При нажатии на кнопку «Начать», начинается автоматическая эволюция клеток в игровом поле в соответствии с правилами игры, надпись кнопки «Начать» изменяется на «Остановить», становятся недоступными для взаимодействия ползунки изменения размера игрового поля, игровое поле становится недоступным для размещения клеток пользователем.

При нажатии на игровое поле на нем появляется, либо исчезает (в зависимости от состояния клетки) клетка в месте, на которое нажал пользователь.

При нажатии на кнопку «Очистить поле» эволюция клеток останавливается (если она происходит в момент нажатия на кнопку), с игрового поля удаляются все живые клетки.

При нажатии на кнопку «Случайно заполнить поле»» эволюция клеток останавливается (если она происходит в момент нажатия на кнопку), игровое поле заполняется живыми клетками в случайных местах.

При движении ползунка изменения количества строк игрового поля с игрового поля удаляются все живые клетки, надпись, отображающая количество строк в игровом поле, изменяется в соответствии с выбранным ползунком значением, количество строк игрового поля изменяется в соответствии с выбранным ползунком значением.

При движении ползунка изменения количества столбцов игрового поля с игрового поля удаляются все живые клетки, надпись, отображающая количество столбцов в игровом поле, изменяется в соответствии с выбранным ползунком значением, количество столбцов игрового поля изменяется в соответствии с выбранным ползунком значением.

При нажатии кнопки «Сохранить игру» открывается диалоговое окно, предлагающее пользователю выбрать название файла сохранения игры и директорию его сохранения.

При нажатии кнопки «Загрузить игру» открывается диалоговое окно, предлагающее пользователю выбрать файл сохранения игры для загрузки.

При нажатии кнопки «Об игре» открывается окно, содержащее текст-описание задания и примечания.

При нажатии кнопки «Об авторе» открывается окно, содержащее фотографию автора, его ФИО, номер группы, год создания приложения, почту для связи.

Функциональная схема:

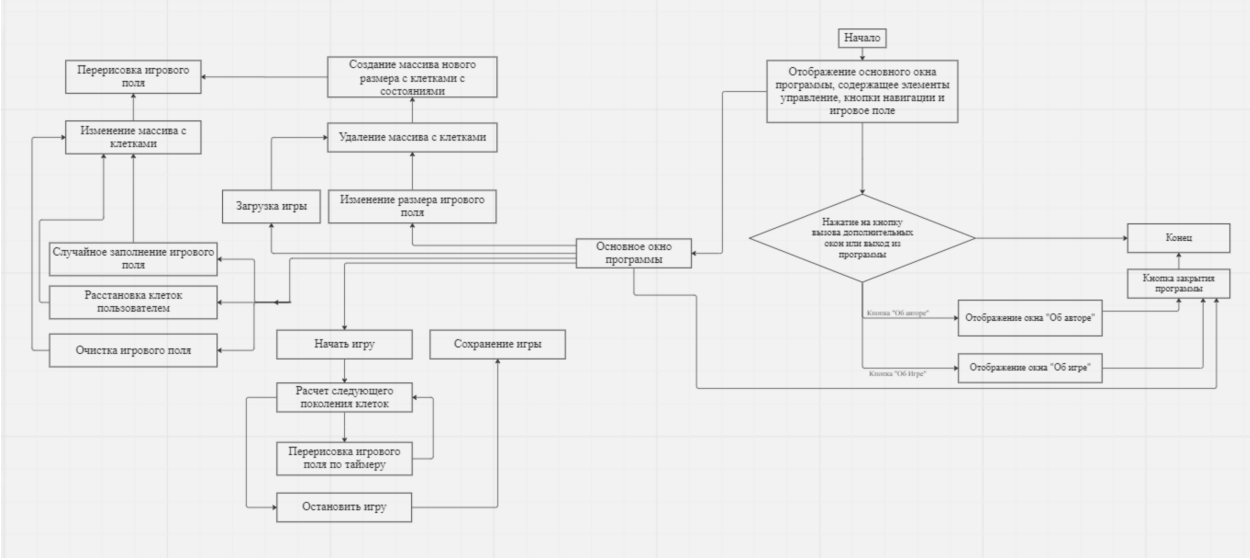

Рисунок 1. – Общий план функциональной схемы.

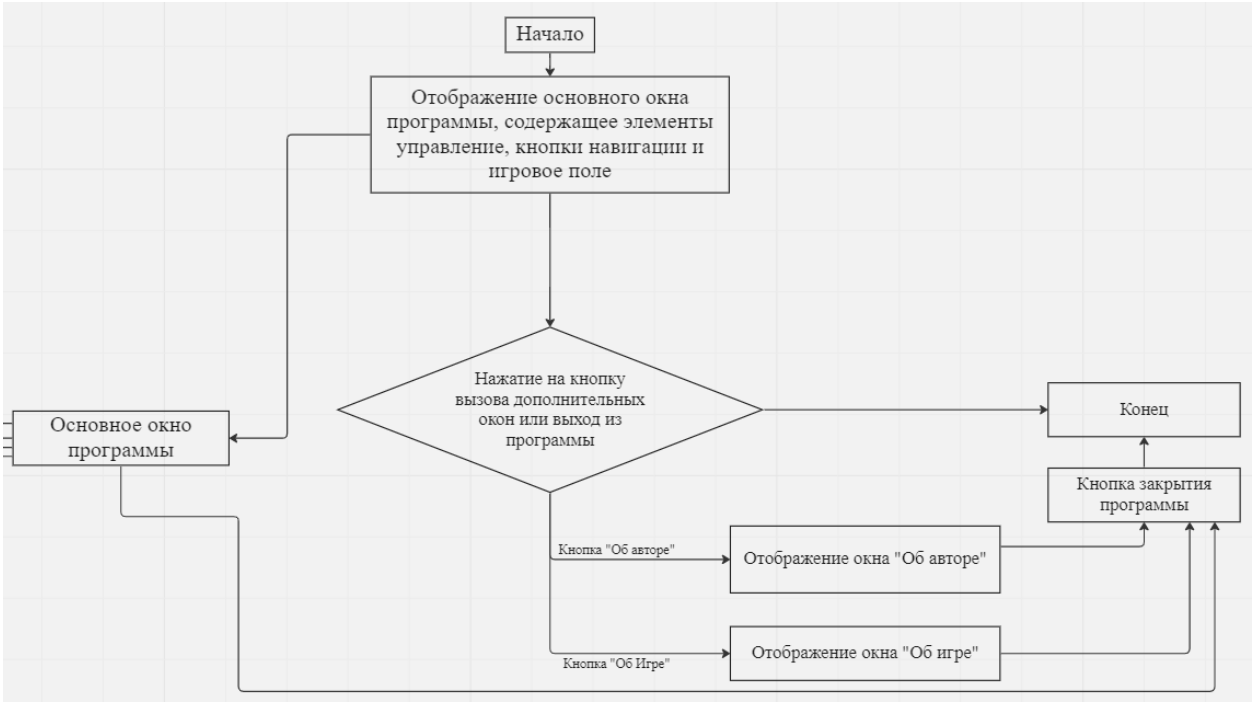

Рисунок 2. – Функциональная схема дополнительных окон.

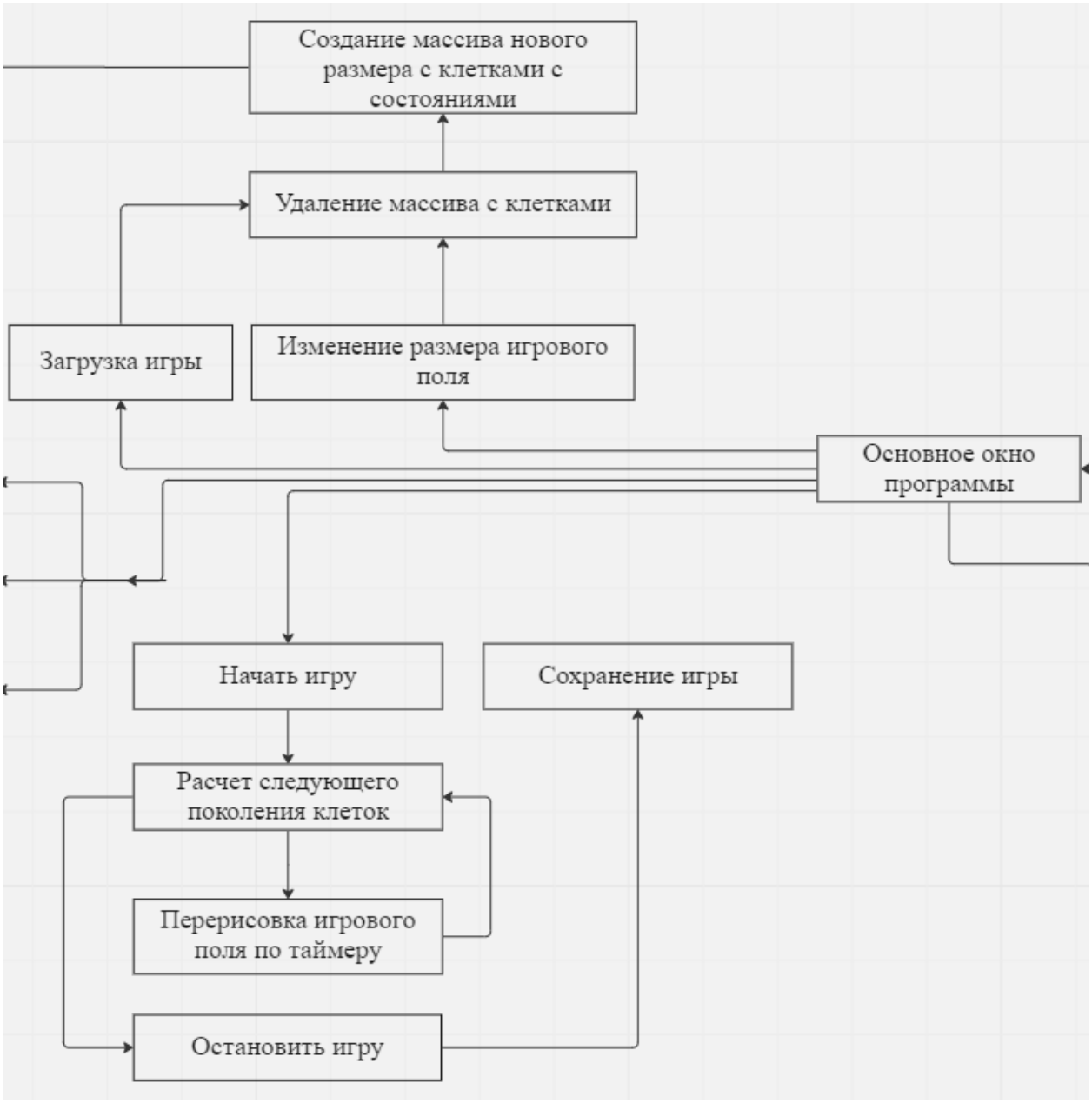
Создание массива нового размера с клетками с состояниями
Удаление массива с клетками
Загрузка игры
Изменение размера игрового поля
Основное окно программы
Начать игру
Сохранение игры
Расчет следующего поколения клеток
Перерисовка игрового поля по таймеру
Остановить игру

Рисунок 3. – Функциональная схема основного окна программы часть 1.

Рисунок 4. – Функциональная схема основного окна программы часть 2.

Выводы по этапу работы: функциональная схема показывает, что основная работа будет проводиться над основным окном программы. В целом программа обладает достаточно обширным ветвлением.

2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1. Введение.
   1.1. Наименование программы – «Игра «Жизнь»».
   1.2. Краткая характеристика области применения:
   программа «Игра «Жизнь»» - игра для одного игрока с управлением при помощи компьютерной мыши и возможностью автоматического режима игры. Описание игры:
2. Основания для разработки.
   2.1. Основанием для разработки является выдача курсовой работы по дисциплине «Языки программирования».
   Дата выдачи – 03.11.2022. Дата защиты: 16.01.2023.
   2.2. Тема курсовой работы: «Разработка компьютерной программы с GUI средствами языка C++».
3. Назначение разработки.
   3.1. Функциональное назначение:
   Программа является игрой для одного игрока с управлением при помощи компьютерной мыши и возможностью автоматического режима игры.
   3.2. Эксплуатационное назначение:
   Программа выполняет досуговую функцию для пользователя.
4. Требования к программе.
   4.1. Требования к функциональным характеристикам:
   - Программа должна быть приложением Windows, с главным и (при необходимости вспомогательными окнами, меню, кнопками и другими элементами оконных форм для взаимодействия с пользователем;
   - Интерфейс программы – на русском языке;
   - Программа должна реализовывать алгоритмы компьютерной графики с элементами анимации, а также (по желанию студента) звуковыми эффектами;

- Программа должна задействовать манипулятор «мышь» для взаимодействия с интерфейсом;
- Программа должна позволят сохранять текущее состояние игры во внешнем файле и по команде пользователя загружать его для продолжения.
- В меню программы должен быть пункт «О программе», по которому должно открываться описание программы (например, правила игры), а также инструкция пользователю по работе с этой программой;
- В меню программы должен быть пункт «Об авторе», по которому открываются данные о разработчике (фото или аватарка, фамилия и инициалы студента, группа, год разработки приложения, почта для связи);
- Установка состояния клетки должна осуществляться при помощи мыши;
- В программе должна быть реализована возможность игры пользователем и компьютером.

4.2. Требования к надежности:

- Программа не должна нарушать целостность компьютера пользователя.

4.3. Условия эксплуатации:

- Пользователь должен уметь обращаться с компьютером на базовом уровне.

4.4. Требования к составу и параметрам технических средств:

- Для использования программы требуется устройство с установленной на него системой Windows 10, манипулятор «мышь» и дисплей.

4.5. Требования к информационной и программной совместимости:

- Для разработки требуются следующие средства: система Windows 10, приложение Qt Creator.

- Для использования программы требуется предустановленная система Windows 10

5. Состав программной документации.
   - Пояснительная записка;
   - Текст программы.
6. Стадии и этапы разработки.

Номер этапа	Название этапа	срок
1	Разработка математической модели решения вычислительной задачи, создание функциональной схемы программы	с 03.11.2022 – по 01.12.2022
2	Формирование технического задания на разработку программы в соответствии с ГОСТ 19.201-78	с 01.12.2022 по 7.12.2022
3	Реализация кода программы на языке высокого уровня C++, тестирование и отладка программы	с 7.12.2022 по 25.12.2022
4	Реализация контрольного примера работы программы	с 25.12.2022 по 5.01.2023

7. Порядок контроля и приемки.

   Контроль и проверка работы осуществляется разработчиком, а также руководителем курсовой работы.

   Выводы по проведенному этапу работы: программа должна удовлетворять обширному списку требований.

3. КОД ПРОГРАММЫ НА ЯЗЫКЕ C++

Ниже представлен код программы на языке C++

main.cpp > ...

```cpp
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv); // Создание объекта приложения
    MainWindow w;               // Создание объекта главного окна
    w.show();                   // Отображение главного окна
    return a.exec();    // Выход из приложения по закрытию главного окна
}
```

Листинг 1 main.cpp

```cpp
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QPainter>

#include "aboutauthor.h"
#include "aboutwindow.h"
#include "gridwidget.h"

// Директива, которая говорит компилятору, что следующие объявления принадлежат пространству имен Qt
QT_BEGIN_NAMESPACE
namespace Ui {
class MainWindow;
}
QT_END_NAMESPACE
// Класс основного окна
// Класс MainWindow наследует от класса QMainWindow
class MainWindow : public QMainWindow {
  Q_OBJECT// Макрос, который позволяет использовать сигналы и слоты

```

Листинг 2 mainwindow.h часть 1

```cpp
public:
  MainWindow(QWidget *parent = nullptr); // Конструктор класса MainWindow
  ~MainWindow();  // Деструктор класса MainWindow

private slots:
  void setColumnCountInfo(const int &value); // Слот для установки значения количества столбцов
  void setRowCountInfo(const int &value); // Слот для установки значения количества строк
  void resetGrid(GridWidget::cellPopulationOption pattern); // Слот для сброса поля
  void editStartOrStopEvolvingButton(); // Слот для изменения текста кнопки
  void on_aboutButton_clicked(); // Слот для открытия окна "О программе"
  void on_authorButton_clicked(); // Слот для открытия окна "Об авторе"
  void saveGame(); // Слот для сохранения игры
  void loadGame(); // Слот для загрузки игры

```

Листинг 3 mainwindow.h часть 2

```
private:
    Ui::MainWindow *ui;  // Создание указателя типа MainWindow
    GridWidget *grid;  // Создание указателя типа GridWidget
    AboutWindow *aboutWindow;  // Создание указателя типа AboutWindow
    AboutAuthor *authorWindow;  // Создание указателя типа AboutAuthorWindow
    void setStartOrStopEvolvingButton();  // Метод для установки кнопки "Старт/Стоп"
    void editStartOrStopEvolvingButtonHelper(const QString &text);  // Метод для изменения текста кнопки
    void setLayout();  // Метод для установки расположения виджетов
    void setColumnCountSlider();  // Метод для установки слайдера для количества столбцов
    void setRowCountSlider();   // Метод для установки слайдера для количества строк
    void setRandomGridButton();  // Метод для установки кнопки "Случайное поле"
    void setEmptyGridButton();  // Метод для установки кнопки "Пустое поле"
    void setSaveLoadButtons();  // Метод для установки кнопок "Сохранить" и "Загрузить"
};
#endif
```

Листинг 4 mainwindow.h часть 3

mainwindow.cpp > ...

```
#include "mainwindow.h"

#include "aboutauthor.h"
#include "gridwidget.h"

#include <QFileDialog>
#include <QMessageBox>

#include "./ui_mainwindow.h"
// Конструктор класса MainWindow
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent), ui(new Ui::MainWindow), grid(new GridWidget(this)) {
    ui->setupUi(this);  // Инициализация интерфейса
    ui->gridLayout->addWidget(grid);  // Добавление виджета игрового поля в главный виджет
    // Настройка интерфейса
    setLayout();
    setSaveLoadButtons();
    setStartOrStopEvolvingButton();
    setColumnCountSlider();
    setRowCountSlider();
    setRandomGridButton();
    setEmptyGridButton();
}
```

Листинг 5 mainwindow.cpp часть 1

```
24  //·Деструктор·класса·MainWindow
25  MainWindow::~MainWindow()·{
26  ··delete·grid;
27  ··delete·ui;
28  }
29  //·Функция·установки·параметров·растяжения·для·компонентов
30  void·MainWindow::setLayout()·{
31  ··ui->mainLayout->setStretchFactor(ui->controlLayout,·4);
32  ··ui->mainLayout->setStretchFactor(ui->gridLayout,·6);
33  }
34  //·Функция·установки·кнопки·начала·игры
35  void·MainWindow::setStartOrStopEvolvingButton()·{
36  ··connect(ui->startOrStopEvolvingButton,·SIGNAL(clicked()),·this,
37  ··········SLOT(editStartOrStopEvolvingButton()));·//·Сигнал·для·изменения·текста·кнопки
38  ··connect(ui->startOrStopEvolvingButton,·SIGNAL(clicked()),·grid,
39  ··········SLOT(toggleEvolveDecision()));·//·Сигнал·для·начала·игры
40  }
41  //·Функция·изменения·текста·кнопки·начала·игры
42  void·MainWindow::editStartOrStopEvolvingButton()·{
43  ··if·(grid->getDoEvolve())·{
44  ····editStartOrStopEvolvingButtonHelper("Начать");
45  ··}·else·{
46  ····editStartOrStopEvolvingButtonHelper("Остановить");
47  ··}
48  }
```

Листинг 6 mainwindow.cpp часть 2

```
49  //·Изменение·текста·кнопки
50  void·MainWindow::editStartOrStopEvolvingButtonHelper(const·QString·&text)·{
51  ··ui->startOrStopEvolvingButton->setText(text);
52  }
53
54  //·Функция·установки·сладера·для·количества·столбцов
55  void·MainWindow::setColumnCountSlider()·{
56  ··ui->columnCountSlider->setMaximum(100);
57  ··ui->columnCountSlider->setMinimum(1);
58  ··ui->columnCountSlider->setValue(20);
59  ··ui->columnCountSlider->setTickPosition(QSlider::TicksBelow);
60
61  ··connect(ui->columnCountSlider,·SIGNAL(valueChanged(int)),·grid,
62  ··········SLOT(setColumnCount(const·int·&)));·//·Сигнал·для·изменения·количества·столбцов
63  ··connect(ui->columnCountSlider,·SIGNAL(valueChanged(int)),·this,
64  ··········SLOT(setColumnCountInfo(const·int·&)));·//·Сигнал·для·изменения·информации·о·количестве·столбцов
65  ··connect(grid,·SIGNAL(universeSizeAdjustable(const·bool·&)),
66  ··········ui->columnCountSlider,·SLOT(setEnabled(bool)));·//·Сигнал·для·блокировки·слайдера
67  }
```

Листинг 7 mainwindow.cpp часть 3

```
68  //·Функция·установки·сладера·для·количества·строк
69  void·MainWindow::setRowCountSlider()·{
70  ··ui->rowCountSlider->setMaximum(100);
71  ··ui->rowCountSlider->setMinimum(1);
72  ··ui->rowCountSlider->setValue(20);
73  ··ui->rowCountSlider->setTickPosition(QSlider::TicksBelow);
74
75  ··connect(ui->rowCountSlider,·SIGNAL(valueChanged(int)),·grid,
76  ··--------SLOT(setRowCount(const·int·&)));·//·Сигнал·для·изменения·количества·строк
77  ··connect(ui->rowCountSlider,·SIGNAL(valueChanged(int)),·this,
78  ··--------SLOT(setRowCountInfo(const·int·&)));·//·Сигнал·для·изменения·информации·о·количестве·строк
79  ··connect(grid,·SIGNAL(universeSizeAdjustable(const·bool·&)),
80  ··--------ui->rowCountSlider,·SLOT(setEnabled(bool)));·//·Сигнал·для·блокировки·слайдера
81  }
82  //·Функция·установки·кнопки·случайной·расстановки
83  void·MainWindow::setRandomGridButton()·{
84  ··connect(ui->randomGridButton,·&QPushButton::clicked,·this,
85  ··--------[this]·{·resetGrid(grid->Random);·});·//·Сигнал·для·сброса·расстановки
86  }
87  //·Функция·установки·кнопки·пустой·расстановки
88  void·MainWindow::setEmptyGridButton()·{
89  ··connect(ui->emptyGridButton,·&QPushButton::clicked,·this,
90  ··--------[this]·{·resetGrid(grid->Empty);·});·//·Сигнал·для·сброса·расстановки
91  }
92
```

Листинг 8 mainwindow.cpp часть 4

```
93  //·Функция·сброса·расстановки·в·соответствии·с·выбранным·типом·расстановки
94  void·MainWindow::resetGrid(GridWidget::cellPopulationOption·pattern)·{
95  ··grid->stopEvolve();
96  ··editStartOrStopEvolvingButtonHelper("Начать");
97  ··grid->setDoEvolve(false);
98  ··grid->deleteGrid();
99  ··grid->createGrid(pattern);
100 ··grid->update();
101 }
102
103 //·Функция·нажатия·кнопки·"О·программе"
104 void·MainWindow::on_aboutButton_clicked()·{
105 ··aboutWindow·=·new·AboutWindow(this);·//·Создание·окна·"О·программе"
106 ··aboutWindow->show();·//·Отображение·окна·"О·программе"
107 }
108 //·Функция·нажатия·кнопки·"Об·авторе"
109 void·MainWindow::on_authorButton_clicked()·{
110 ··authorWindow·=·new·AboutAuthor(this);·//·Создание·окна·"Об·авторе"
111 ··authorWindow->show();·//·Отображение·окна·"Об·авторе"
112 }
113 //·Функция·установки·кнопок·сохранения·и·загрузки
114 void·MainWindow::setSaveLoadButtons()·{
115 ··connect(ui->saveButton,·SIGNAL(clicked()),·this,·SLOT(saveGame()));·//·Сигнал·для·сохранения·игры
116 ··connect(ui->loadButton,·SIGNAL(clicked()),·this,·SLOT(loadGame()));·//·Сигнал·для·загрузки·игры
117 }
```

Листинг 9 mainwindow.cpp часть 5

```
118  //·Функция·сохранения·игры
119  void·MainWindow::saveGame()·{
120  ··grid->stopEvolve();
121  ··editStartOrStopEvolvingButtonHelper("Начать");
122  ··grid->setDoEvolve(false);
123  ··QString·fileName·=·QFileDialog::getSaveFileName(·
124  ·····this,·tr("Сохранить·игру"),·QDir::currentPath(),
125  ·····tr("Файлы·игры·(*.game)"));··//·Открытие·диалогового·окна·с·выбором
126  ·····························//·директории·сохрания·и·именем·файла
127  ··if·(fileName.isEmpty())·{
128  ····return;
129  ··}·else·{
130  ····QFile·file(fileName);
131  ····if·(!file.open(QIODevice::WriteOnly))·{
132  ·····QMessageBox::information(this,·tr("Невозможно·открыть·файл"),
133  ·····························file.errorString());
134  ·····return;
135  ····}
136  ····QDataStream·out(&file);
137  ····out.setVersion(QDataStream::Qt_5_9);
138  ····out·<<·grid->getColumnCount()·<<·grid->getRowCount();·//·Сохранение·размеров·поля
139  ····for·(int·i·=·0;·i·<·grid->getRowCount();·i++)·{
140  ·····for·(int·j·=·0;·j·<·grid->getColumnCount();·j++)·{
141  ·····out·<<·grid->getCellState(i,·j);·//·Сохранение·состояния·клетки
142  ·····}
143  ····}
144  ··}
145  }
```

Листинг 10 mainwindow.cpp часть 6

```
146  //·Функция·загрузки·игры
147  void·MainWindow::loadGame()·{
148  ··grid->stopEvolve();
149  ··editStartOrStopEvolvingButtonHelper("Начать");
150  ··grid->setDoEvolve(false);
151  
152  ··QString·fileName·=·QFileDialog::getOpenFileName(
153  ·····this,·tr("Загрузить·игру"),·QDir::currentPath(),
154  ·····tr("Файлы·игры·(*.game)"));··//·Открытие·диалогового·окна·с·выбором·файла
155  ··if·(fileName.isEmpty())·{
156  ····return;
157  ··}·else·{
158  ····QFile·file(fileName);
159  ····if·(!file.open(QIODevice::ReadOnly))·{
160  ·····QMessageBox::information(this,·tr("Невозможно·открыть·файл"),
161  ·····························file.errorString());
162  ·····return;
163  ····}
164  ····QDataStream·in(&file);
165  ····in.setVersion(QDataStream::Qt_5_9);
166  ····int·columnCount,·rowCount;
167  ····in·>>·columnCount·>>·rowCount;
168  ····ui->rowCountSlider->setValue(rowCount);·//·Установка·количества·строк
169  ····ui->columnCountSlider->setValue(columnCount);·//·Установка·количества·столбцов
170  ····grid->createGrid(grid->Empty);
171  ····for·(int·i·=·0;·i·<·grid->getRowCount();·i++)·{
172  ·····for·(int·j·=·0;·j·<·grid->getColumnCount();·j++)·{
173  ·····int·state;
174  ·····in·>>·state;
175  ·····grid->setCellState(i,·j,·state);·//·Загрузка·состояния·клетки
176  ·····}
177  ····}
178  ··}
179  }
180
```

Листинг 11 mainwindow.cpp часть 7

```
181  //·Функция·изменения·значения·количества·строк·в·интерфейсе
182  void·MainWindow::setRowCountInfo(const·int·&nRows)·{
183  ··int·newStrLen·=·snprintf(NULL,·0,·"Количество·строк:·%d",·nRows)·+·1;
184  ··char·buffer[newStrLen];
185  ··snprintf(buffer,·newStrLen,·"Количество·строк:·%d",·nRows);
186  ··QString·newStr·=·buffer;
187  ··ui->rowCountSliderInfo->setText(newStr);
188  }
189  //·Функция·изменения·значения·количества·столбцов·в·интерфейсеы
190  void·MainWindow::setColumnCountInfo(const·int·&nColumns)·{
191  ··int·newStrLen·=·snprintf(NULL,·0,·"Количество·столбцов:·%d",·nColumns)·+·1;
192  ··char·buffer[newStrLen];
193  ··snprintf(buffer,·newStrLen,·"Количество·столбцов:·%d",·nColumns);
194  ··QString·newStr·=·buffer;
195  ··ui->columnCountSliderInfo->setText(newStr);
196  }
197
```

Листинг 12 mainwindow.cpp часть 8

h gridwidget.h > ...

```
#ifndef GRIDWINDOW_H
#define GRIDWINDOW_H

#include <QPainter>
#include <QWidget>

namespace Ui { // Используется для доступа к элементам интерфейса
class GridWidget;
}
// Класс игрового поля
// Наследуется от QWidget
class GridWidget : public QWidget {
  Q_OBJECT //  Макрос для поддержки сигналов и слотов
public:
  GridWidget(QWidget* parent = nullptr);  // Конструктор
  ~GridWidget(); // Деструктор
  int getCellState(const int& columnIndex, const int& rowIndex); // Получение
                                                                 // состояния
                                                                 // клетки
  void setCellState(const int& columnIndex, const int& rowIndex,
                    const int& state); // Установка состояния клетки
  void deleteGrid(); // Удаление поля
  int getRowCount() const; // Получение количества строк
  int getColumnCount() const; // Получение количества столбцов
  bool getDoEvolve(); // Получение состояния автоматической эволюции
  void setDoEvolve(bool value); // Установка состояния автоматической эволюции
  enum cellPopulationOption {
    Empty,
    Random,
  }; // Типы заполнения поля

```

Листинг 13 gridwidget.h часть 1

```
protected:
    void paintEvent(
        QPaintEvent* event) override;  // Переопределение метода отрисовки
    void mousePressEvent(
        QMouseEvent* event) override;  // Переопределние метода обработки нажатий

public slots:
    void createGrid(cellPopulationOption pattern); // Создание поля по заданному
                                                   // шаблону
    void stopEvolve(); // Остановка автоматической эволюции
    void setRowCount(const int& nRows); // Установка количества строк
    void setColumnCount(const int& nColumns); // Установка количества столбцов

signals:
    void universeSizeAdjustable(
        const bool& cmd);  // Сигнал о возможности изменения размера поля

private:
    int rowCount = 20;     // Количество строк
    int columnCount = 20;  // Количество столбцов
    int** grid;  // Массив для хранения состояния клеток
    QTimer* timer;  // Таймер для автоматической эволюции
    bool doEvolve =
        false;  // Переменная для хранения состояния автоматической эволюции

    void setTimer(); // Установка таймера

    void evolveContinuous(); // Автоматическая эволюция

    QColor universeBorderColour = "#202020";  // Цвет границы поля
    qreal universeBorderThickness = 5.0;  // Толщина границы поля
    QColor universeFieldColour = Qt::white;  // Цвет поля
    QColor cellFieldColour = Qt::black;      // Цвет клетки
    qreal cellGridMargin = 1.0;  // Отступ сетки клеток

```

Листинг 14 gridwidget.h часть 2

```
    qreal calcUniverseWidth(); // Расчет ширины поля  в пикселях
    qreal calcUniverseHeight(); // Расчет высоты поля в пикселях
    qreal calcCellWidth(); // Расчет ширины клетки в пикселях
    qreal calcCellHeight(); // Расчет высоты клетки в пикселях

    void paintUniverseBorder(QPainter& painter); // Отрисовка границы поля
    void paintCellGrid(QPainter& painter); // Отрисовка сетки клеток

    void createEmptyGrid(); // Создание пустого поля
    void createRandomGrid(); // Создание поля с рандомным заполнением
private slots:
    void toggleEvolveDecision(); // Переключение состояния автоматической
                                 // эволюции
    void evolveOnce(); // Однократная эволюция
};
#endif
```

Листинг 15 gridwidget.h часть 3

gridwidget.cpp > ...

```
#include "gridwidget.h"

#include <qpainter.h>
#include <qwidget.h>

#include <QMouseEvent>
#include <QTimer>

#include "gamelogic.h"

// Конструктор класса GridWidget
GridWidget::GridWidget(QWidget* parent) : QWidget(parent) {
    setTimer(); // Установка таймера
    createGrid(Empty);
}
// Деструктор класса GridWidget
GridWidget::~GridWidget() {}

//Установки таймера
void GridWidget::setTimer() {
    timer = new QTimer(this);  // Создание таймера
    timer->setInterval(200);  // Установка интервала таймера
    connect(timer, SIGNAL(timeout()), this,
            SLOT(evolveOnce()));  // Подключение таймера к функции обновления поля
}

// Продолжить игру
void GridWidget::evolveContinuous() {
    timer->start(); // Запуск таймера
    emit universeSizeAdjustable(false); // Отключение возможности изменения размера поля
    this->setAttribute(Qt::WA_TransparentForMouseEvents, true); // Отключение возможности
                                                                //взаимодействия с полем
}

```

Листинг 16 gridwidget.cpp часть 1

```cpp
// Остановка игры
void GridWidget::stopEvolve() {
  timer->stop(); // Остановка таймера
  emit universeSizeAdjustable(true); // Включение возможности изменения размера поля
  this->setAttribute(Qt::WA_TransparentForMouseEvents, false); // Включение возможности
                                                               //взаимодействия с полем
}

// 1-кратное обновление поля
void GridWidget::evolveOnce() {
  evolveNextGeneration(grid, rowCount, columnCount); // Вызов функции обновления поля
  update(); // Обновление поля
}
// Получить состояние игры
bool GridWidget::getDoEvolve() { return doEvolve; }
// Установить состояние игры
void GridWidget::setDoEvolve(bool value) { doEvolve = value; }
// Получения состояния клетки игрового поля
int GridWidget::getCellState(const int& rowIndex, const int& columnIndex) {
  return grid[rowIndex][columnIndex];
}

```

Листинг 17 gridwidget.cpp часть 2

```
57  //·Установка·состояния·клетки·игрового·поля
58  void·GridWidget::setCellState(const·int&·rowIndex,·const·int&·columnIndex,
59  ····const·int&·state)·{
60  ·grid[rowIndex][columnIndex]·=·state;
61  }
62  //·Начать/остановить·игру
63  void·GridWidget::toggleEvolveDecision()·{
64  ··doEvolve·=·!doEvolve;
65  ··if·(doEvolve)·{
66  ····evolveContinuous();
67  ··}·else·{
68  ····stopEvolve();
69  ··}
70  }
71
72  //Заполнения·игрового·поля·по·паттерну
73  void·GridWidget::createGrid(cellPopulationOption·pattern)·{
74  ··if·(pattern·==·Empty)·{
75  ····createEmptyGrid();
76  ··}·else·if·(pattern·==·Random)·{
77  ····createRandomGrid();
78  ··}
79  }
80
81  //Создание·пустого·поля
82  void·GridWidget::createEmptyGrid()·{
83  ··grid·=·new·int*[rowCount];
84  ··for·(int·rowIdx·=·0;·rowIdx·<·rowCount;·++rowIdx)·{
85  ····grid[rowIdx]·=·new·int[columnCount];
86  ····std::fill(grid[rowIdx],·grid[rowIdx]·+·columnCount,·0);
87  ··}
88  }
89
```

Листинг 18 gridwidget.cpp часть 3

```
90  //Создание поля с рандомными заполнением
91  void GridWidget::createRandomGrid() {
92      grid = new int*[rowCount];
93      for (int rowIdx = 0; rowIdx < rowCount; rowIdx++) {
94          grid[rowIdx] = new int[columnCount];
95          for (int columnIdx = 0; columnIdx < columnCount; columnIdx++) {
96              grid[rowIdx][columnIdx] = 0 + (rand() % (1 - 0 + 1)) == 1;
97              ;
98          }
99      }
100 }
101
102 // Удаление массива с клетками из памяти
103 void GridWidget::deleteGrid() {
104     for (int rowIdx = 0; rowIdx < rowCount; rowIdx++) {
105         delete[] grid[rowIdx];
106     }
107     delete[] grid;
108 }
109 // Получение количества строк
110 int GridWidget::getRowCount() const { return rowCount; }
111
112 // Установка количества строк
113 void GridWidget::setRowCount(const int& nRows) {
114     deleteGrid();
115     rowCount = nRows;
116     createGrid(Empty);
117     update();
118 }
```

Листинг 19 gridwidget.cpp часть 4

```
119  //·Получение·количества·столбцов
120  int·GridWidget::getColumnCount()·const·{·return·columnCount;·}
121  //·Установка·количества·столбцов
122  void·GridWidget::setColumnCount(const·int&·nColumns)·{
123  ··deleteGrid();
124  ··columnCount·=·nColumns;
125  ··createGrid(Empty);
126  ··update();
127  }
128  //·Обработка·нажатий·на·игровое·поле·пользователем
129  void·GridWidget::mousePressEvent(QMouseEvent*·event)·{
130  ··int·rowIdx·=·static_cast<int>(std::floor(
131  ··········(event->y()·-·0.75·*·universeBorderThickness)·/·calcCellHeight()));
132  ··········//·Вычисление·индекса·строки
133  ··int·columnIdx·=·static_cast<int>(std::floor(
134  ··········(event->x()·-·0.75·*·universeBorderThickness)·/·calcCellWidth()));·
135  ··········//·Вычисление·индекса·столбца
136  ··grid[rowIdx][columnIdx]·^=·1;·//·Инвертирование·состояния·клетки
137  ··update();
138  }
139
140  //Отрисовкв·игрового·поля
141  void·GridWidget::paintEvent(QPaintEvent*·event)·{
142  ··QPainter·painter(this);·//·Создание·объекта·отрисовщика
143  ··paintUniverseBorder(painter);·//·Отрисовка·рамки·игрового·поля
144  ··paintCellGrid(painter);·//·Отрисовка·клеток
145  }
146
```

Листинг 20 gridwidget.cpp часть 5

```
146
147 //Отрисовка рамки игрового поля
148 void GridWidget::paintUniverseBorder(QPainter& painter) {
149     QRect universeBorder(0, 0, width(), height()); // Рамка игрового поля
150     painter.setBrush(QBrush(universeFieldColour)); // Цвет заливки
151     painter.fillRect(universeBorder, painter.brush()); // Заливка игрового поля
152     painter.setPen(QPen(universeBorderColour, universeBorderThickness));
153     painter.drawRect(universeBorder); // Рисование рамки
154 }
155
156 //Отрисовка клеток
157 void GridWidget::paintCellGrid(QPainter& painter) {
158     for (int rowIdx = 0; rowIdx < rowCount; ++rowIdx) {
159         for (int columnIdx = 0; columnIdx < columnCount; ++columnIdx) {
160             if (grid[rowIdx][columnIdx] == 1) {
161                 qreal cellLeftIdx = 0.75 * universeBorderThickness +
162                                     calcCellWidth() * columnIdx + cellGridMargin;
163                 qreal cellTopIdx = 0.75 * universeBorderThickness +
164                                    calcCellHeight() * rowIdx + cellGridMargin;
165                 QRect cellField(cellLeftIdx, cellTopIdx,
166                                 calcCellWidth() - cellGridMargin,
167                                 calcCellHeight() - cellGridMargin);
168                 painter.setBrush(QBrush(cellFieldColour));
169                 painter.fillRect(cellField, painter.brush());
170             }
171         }
172     }
173 }
174
175 // Набор функций для расчета размера игрового поля и клеток
176 qreal GridWidget::calcUniverseWidth() {
177     return width() - 1.5 * universeBorderThickness;}
178 qreal GridWidget::calcUniverseHeight() {
179     return height() - 1.5 * universeBorderThickness;}
180 qreal GridWidget::calcCellWidth() { return calcUniverseWidth() / columnCount; }
181 qreal GridWidget::calcCellHeight() { return calcUniverseHeight() / rowCount; }
```

Листинг 21 gridwidget.cpp часть 6

```
h gamelogic.h > ...
1 #ifndef GAMELOGIC_H
2 #define GAMELOGIC_H
3 // Функция для подсчета суммы соседей
4 int neighbourSum(int** grid, const int& rowCount, const int& columnCount, const int& rowIdx, const int& columnIdx);
5 // Функция для эволюции поля
6 void evolveNextGeneration(int** grid, const int& rowCount, const int& columnCount);
7 #endif
```

Листинг 22 gamelogic.h

gamelogic.cpp > ...

```cpp
#include <algorithm>
// Функция для подсчета количства клеток-соседей
int neighbourSum(int** grid, const int& rowCount, const int& columnCount,
                 const int& rowIdx, const int& columnIdx) {
    int sum = 0;
    for (int y = std::max(rowIdx - 1, 0); y < std::min(rowIdx + 2, rowCount);
         y++) {
        for (int x = std::max(columnIdx - 1, 0);
             x < std::min(columnIdx + 2, columnCount); x++) {
            sum += grid[y][x] & 1;
        }
    }
    return sum -= grid[rowIdx][columnIdx] & 1; // Вычитаем из суммы соседей
                                               // клетку, для которой считаем
                                               // соседей
}
// Фукнция для вычисления следующего поколения
void evolveNextGeneration(int** grid, const int& rowCount,
                          const int& columnCount) {
    for (int rowIdx = 0; rowIdx < rowCount; rowIdx++) {
        for (int columnIdx = 0; columnIdx < columnCount; columnIdx++) {
            int ns = neighbourSum(grid, rowCount, columnCount, rowIdx, columnIdx);
            if ((ns | grid[rowIdx][columnIdx]) == 3) {
                grid[rowIdx][columnIdx] |= 2; // Если соседей 2 или 3, то клетка
                                              // остается живой
            }
        }
    }
// Переносим состояние клетки в следующее поколение
    for (int rowIdx = 0; rowIdx < rowCount; rowIdx++) {
        for (int columnIdx = 0; columnIdx < columnCount; columnIdx++) {
            grid[rowIdx][columnIdx] >>= 1;
        }
    }
}
```

Листинг 23 gamelogic.cpp

h aboutauthor.h > ...

```h
#ifndef ABOUTAUTHOR_H
#define ABOUTAUTHOR_H

#include <QDialog>

namespace Ui { // Используется для доступа к элементам интерфейса
class AboutAuthor;
}
// Класс для отображения информации об авторе
// Класс наследуется от QDialog
class AboutAuthor : public QDialog {
  Q_OBJECT // Макрос для поддержки сигналов и слотов

public:
  explicit AboutAuthor(QWidget *parent = nullptr); // Конструктор
  ~AboutAuthor(); // Деструктор

private:
  Ui::AboutAuthor *ui; // Указатель на элементы интерфейса
};
#endif
```

Листинг 24 aboutauthor.h

C++ aboutauthor.cpp > ...

```cpp
#include "aboutauthor.h"

#include "ui_aboutauthor.h"
// Конструктор класса AboutAuthor
AboutAuthor::AboutAuthor(QWidget *parent)
    : QDialog(parent), ui(new Ui::AboutAuthor) {
  ui->setupUi(this); // Инициализация интерфейса
}
// Деструктор класса AboutAuthor
AboutAuthor::~AboutAuthor() { delete ui; }
```

Листинг 26 aboutauthor.cpp

```h
#ifndef ABOUTWINDOW_H
#define ABOUTWINDOW_H

#include <QDialog>

namespace Ui { //Используется для доступа к элементам интерфейса
class AboutWindow;
}
//Класс для отображения информации о программе
//Класс наследуется от QDialog
class AboutWindow : public QDialog
{
    Q_OBJECT //Макрос для поддержки сигналов и слотов

public:
    explicit AboutWindow(QWidget *parent = nullptr); //Конструктор
    ~AboutWindow(); //Деструктор

private:
    Ui::AboutWindow *ui; //Указатель на элементы интерфейса
};
#endif
```

Листинг 27 aboutwindow.h

```cpp
#include "aboutwindow.h"

#include "ui_aboutwindow.h"
//Конструктор класса AboutWindow
AboutWindow::AboutWindow(QWidget *parent)
    : QDialog(parent), ui(new Ui::AboutWindow) {
    ui->setupUi(this); //Инициализация интерфейса
}
//Деструктор класса AboutWindow
AboutWindow::~AboutWindow() { delete ui; }
```

Листинг 28 aboutwindow.cpp

4. КОНТРОЛЬНЫЙ ПРИМЕР РАБОТЫ ПРОГРАММЫ

Ниже представлены скриншоты контрольного примера работы программы игра "Жизнь", которая была разработана в соответствии с функциональной схемой.

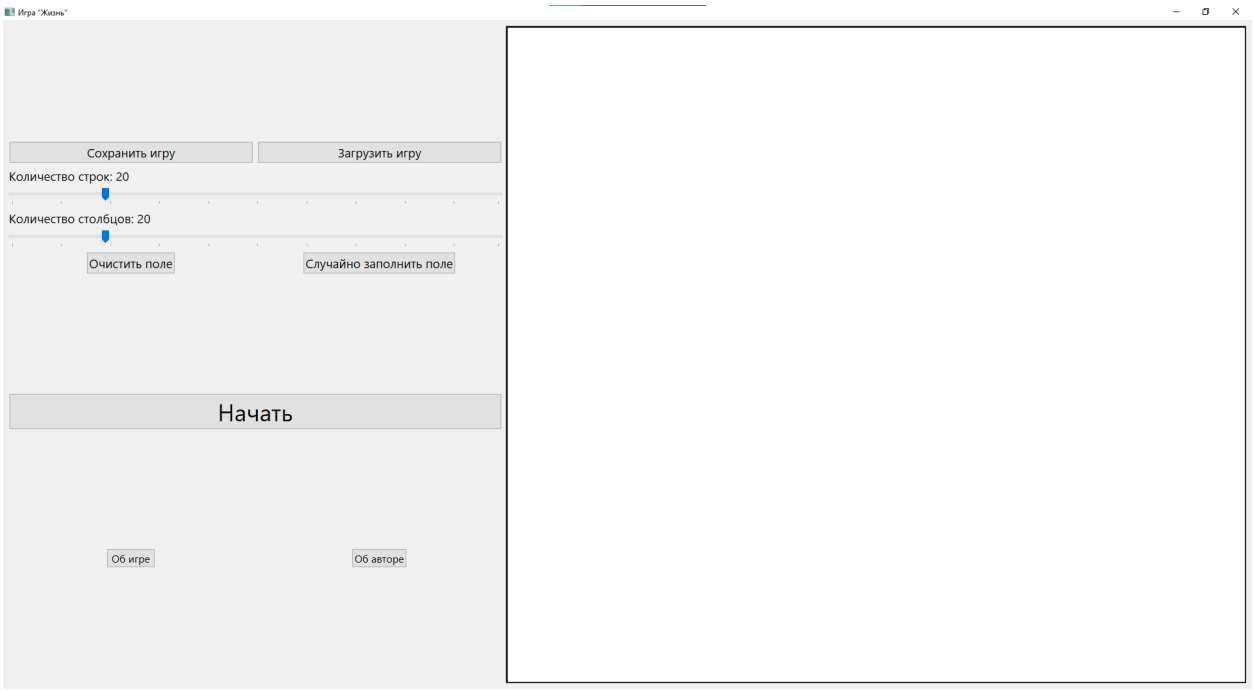

Рисунок 5. - Стартовое состояние программы. Основное окно программы.

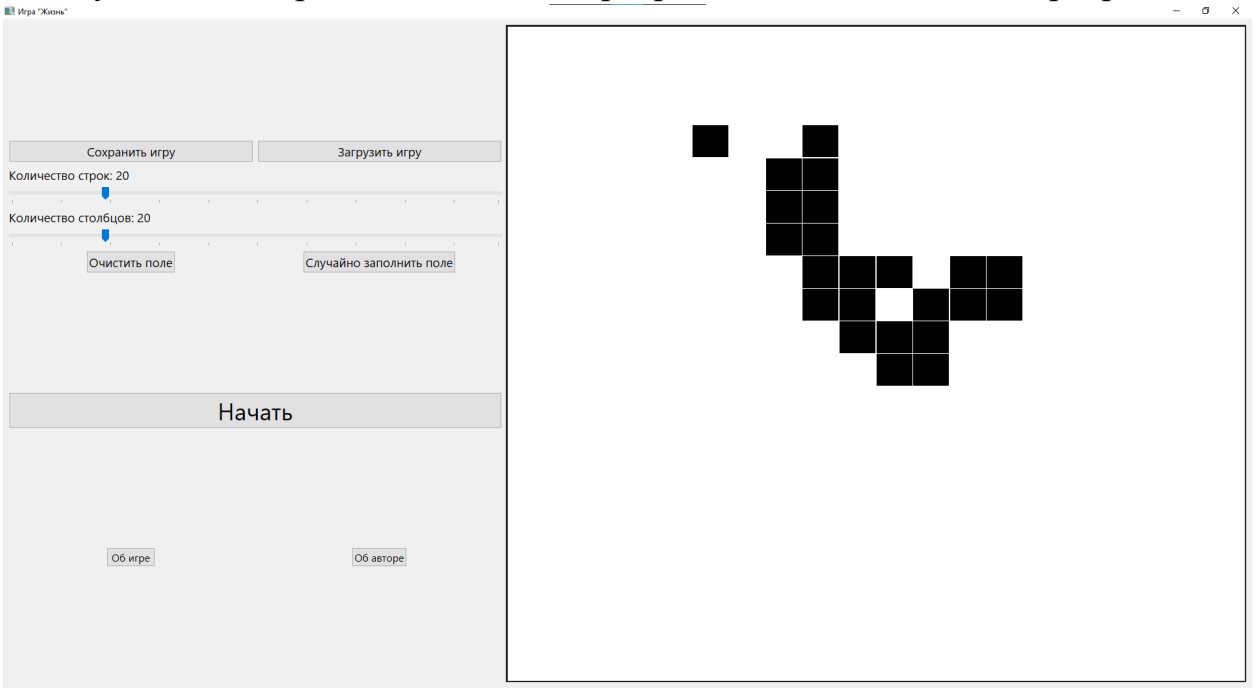

Рисунок 6. - Основное окно программы. Пользователь изменил состояние клеток на игровом поле.

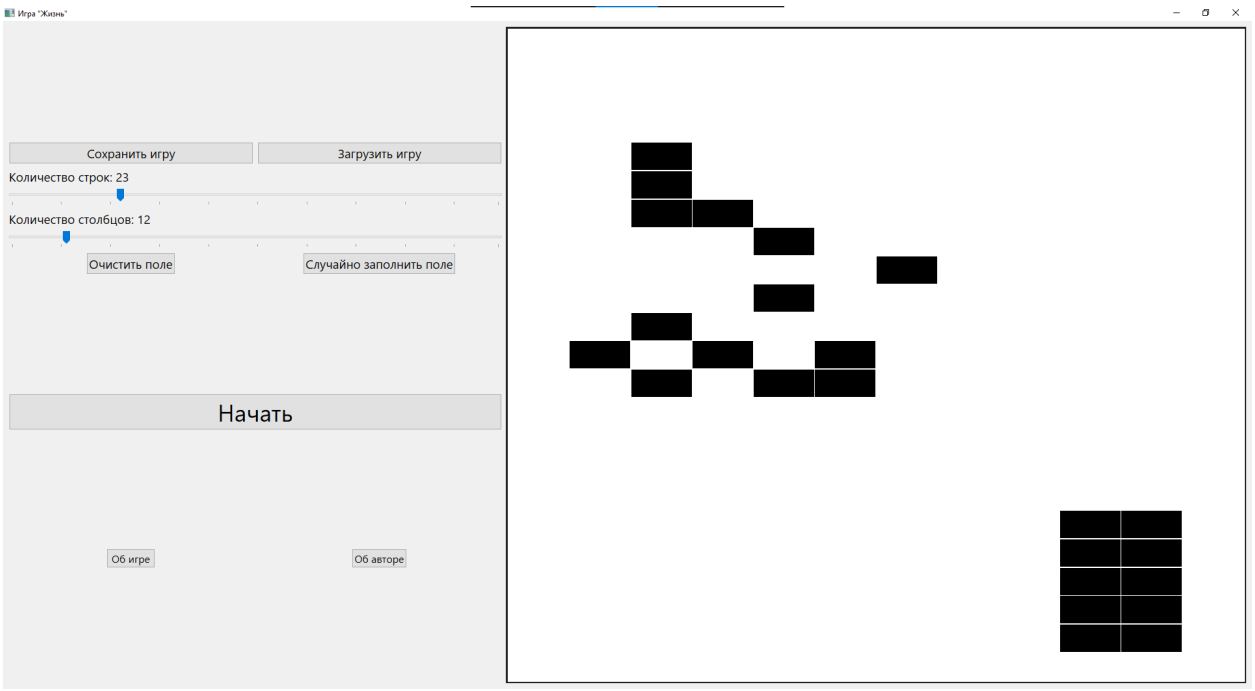

Рисунок 7. - Основное окно программы. Пользователь изменил размер игрового поля и состояние клеток на нем.

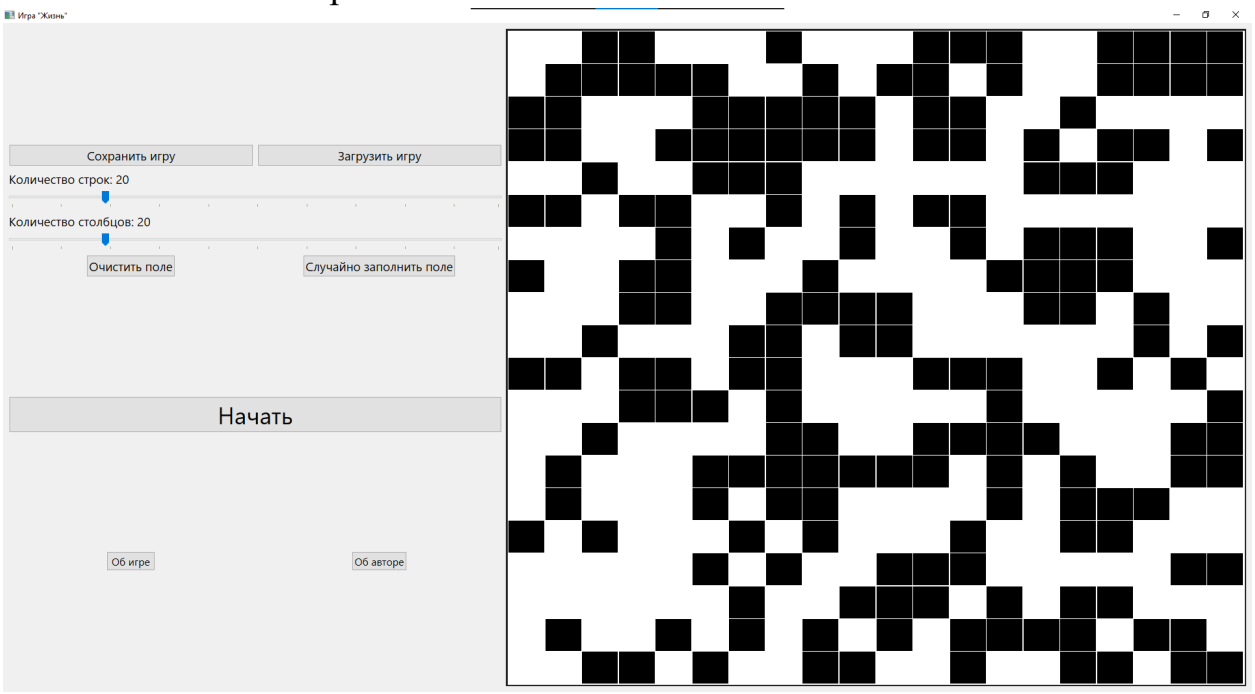

Рисунок 8. - Основное окно программы. Пользователь нажал кнопку «Случайно заполнить поле».

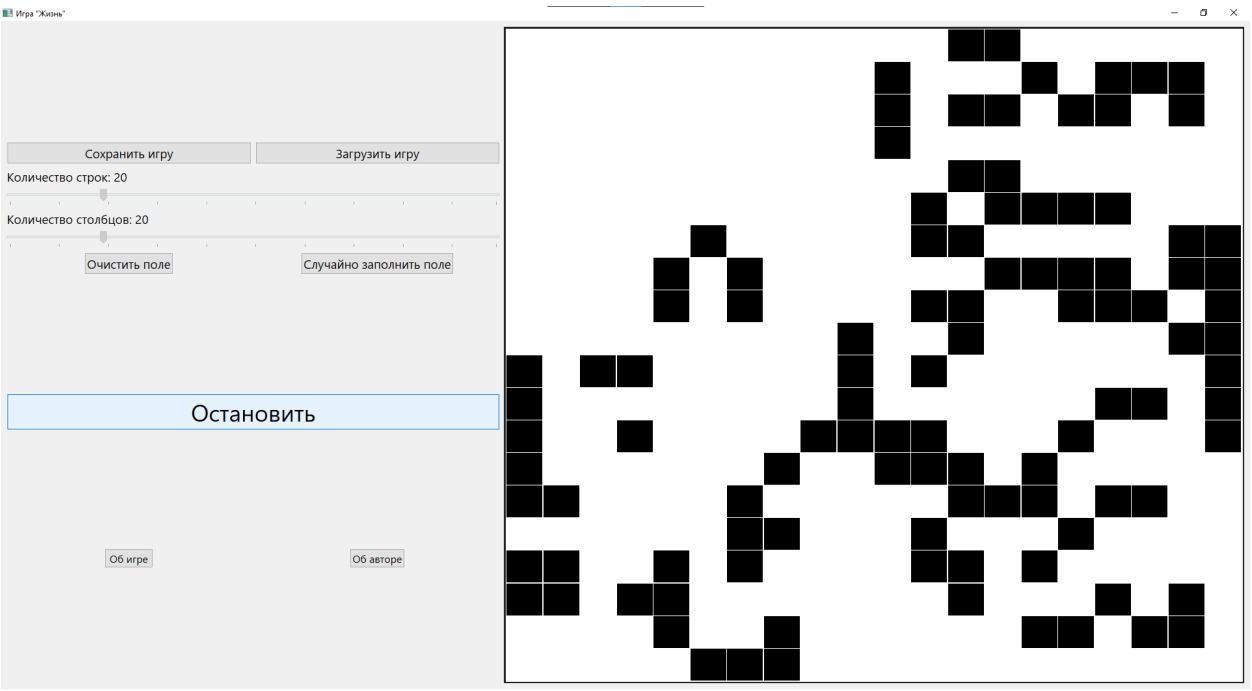

Рисунок 9. - Основное окно программы. Пользователь нажал кнопку «Начать».

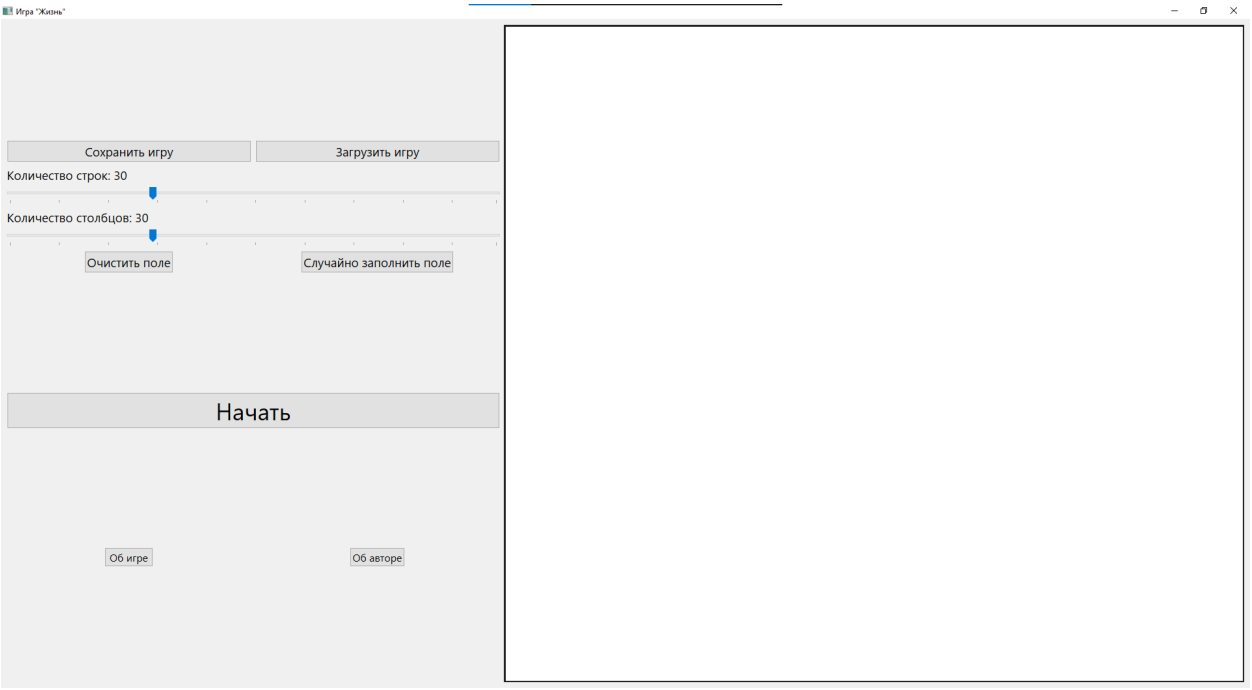

Рисунок 10. - Основное окно программы. Пользователь нажал кнопку «Очистить поле»

Рисунок 11. - Основное окно программы. Пользователь нажал кнопку «Сохранить игру».

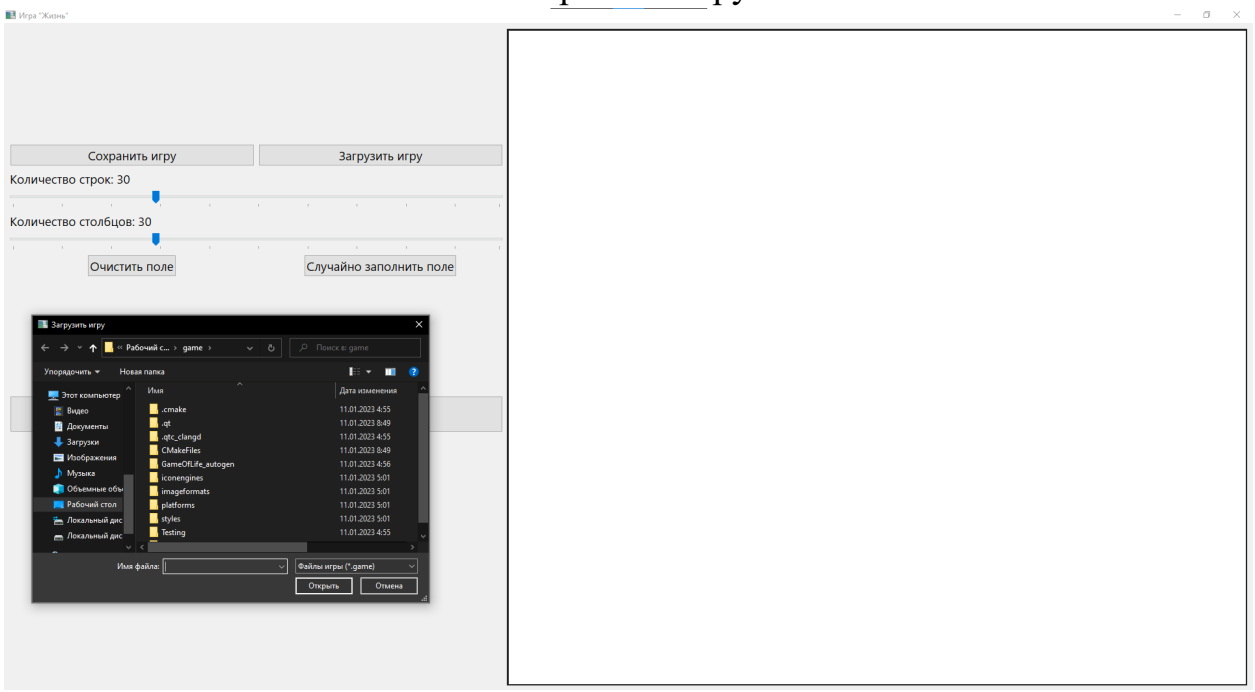

Рисунок 12. - Основное окно программы. Пользователь нажал кнопку «Загрузить игру».

Рисунок 13. - Основное окно программы. Пользователь загрузил предыдущее сохранение.

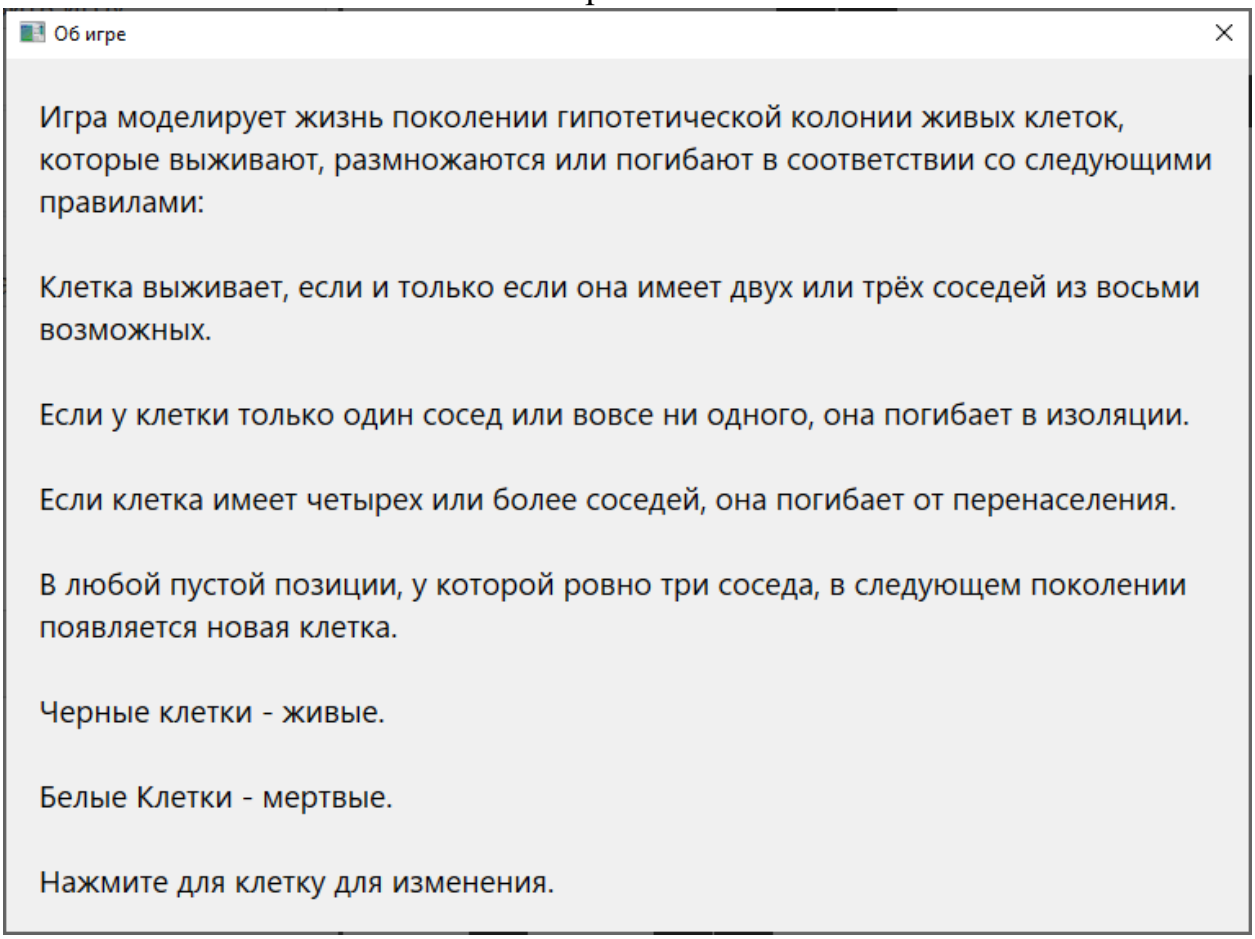

Рисунок 14. - Окно «Об игре».

Рисунок 15. - Окно «Об авторе».

Вывод по проделанному этапу: программа выполнена и отвечает всем требованиям.

ЗАКЛЮЧЕНИЕ

Выполнены следующие задачи курсовой работы:

1. Разработать математическую модель решения вычислительной задачи, привести функциональную схему программы – модель разработана и представлена в пояснительной записке.
2. Сформировать техническое задание на разработку программы в соответствии с ГОСТ 19.201-78 – техническое задание сформировано и представлено в пояснительной записке.
3. Реализовать код программы на языке высокого уровня C++, протестировать его и отладить – код программы реализован и представлен в пояснительной записке.
4. Реализовать контрольный пример работы программы, начиная с открытия, показать все этапы работы приложения – программа выполнена, выполняет свои функции, ее функции описаны и представлены в пояснительной записке.

В заключении следует отметить, что все задачи курсовой работы решены успешно. Благодаря использованию функциональной схемы, мы смогли реализовать программу, которая способна корректно моделировать игру "Жизнь" и демонстрировать развитие различных паттернов. Скриншоты контрольного примера работы программы являются доказательством этого. Таким образом, цель курсовой работы была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19.201-78. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению.
2. Абрамов С.А. и др. Задачи по программированию – М.:Наука, 1988 – 224 с.
3. Рысин М.Л. и др. Основы программирования на языке С++: Учеб. Пособие. – М.: МИРЭА – Российский технологический университет, 2022. – 118 с.
4. Рудаков А.В. Технология разработки программных продуктов. Практикум: учеб.пособие для студ. – 4 -е изд. – М.: Издательский центр «Академия», 2014. – 192 с.
5. The Qt Company, Qt Documentation [Электронный ресурс]: документация. – https://doc.qt.io/ (дата обращения: 25.12.2022) .