МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ»)

Факультет экономики, менеджмента и информационных технологий

Кафедра систем управления и информационных технологий в строительстве

КУРСОВОЙ ПРОЕКТ

по дисциплине: «Основы программирования и алгоритмизации»

Тема: «Разработка программы-головоломки Нумератор(«Numbrix») »

**Расчетно-пояснительная записка**

Разработал студент Е.Р. Ростошинский

Подпись, дата Инициалы, фамилия

Руководитель О.В. Минакова

Подпись, дата Инициалы, фамилия

Нормоконтролер О.В. Минакова

Подпись, дата Инициалы, фамилия

Защищена Оценка

дата

2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ФГБОУ ВО «ВГТУ»)

Кафедра систем управления и информационных технологий в строительстве

ЗАДАНИЕ

на курсовой проект

по дисциплине Основы программирования и алгоритмизации

Тема: «Разработка программы-головоломки Нумератор(«Numbrix»).»

Студент группы бИЦ-241 Ростошинский Егор Алексеевич

Фамилия, имя, отчество

Технические условия Windows 10, MicrosoftVisualStudio2019, язык программирования С

Содержание и объем проекта (графические работы, расчеты и прочее): 36 стр, 17 рисунков, 2 приложения

Сроки выполнения этапов анализ и постановка задачи (10.9-5.10.24); разработка пошаговой детализации программы (6.10-11.11.24); реализация программы (11.11-5.12.24); тестирование программы (6.12-11.12.24); оформление пояснительной записки (11.12-19.12.24).

Срок защиты курсового проекта

Руководитель О.В. Минакова

Подпись, дата Инициалы, фамилия

Задание принял студент Е.Р. Ростошинский

Подпись, дата Инициалы, фамилия

**Замечания руководителя**

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 5

1 Постановка задачи 6

2 Конструирование программы 10

3 Тестирование программы 17

ЗАКЛЮЧЕНИЕ 21

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 22

ПРИЛОЖЕНИЕ А 23

ПРИЛОЖЕНИЕ Б 36

# ВВЕДЕНИЕ

Головоломки являются популярным видом интеллектуальных развлечений, которые не только развивают логическое мышление, но и предоставляют возможность интересного времяпрепровождения. Одной из таких головоломок является «Numbrix» (Нумератор), цель которой заключается в заполнении игрового поля числами таким образом, чтобы они образовывали последовательную цепочку от минимального до максимального значения. Создание и генерация корректных игровых полей для этой головоломки требует тщательной проверки и отбора, что делает процесс разработки программы для их создания актуальной задачей.

Целью данного курсового проекта является разработка программы-игры «Numbrix» на языке программирования C, которая позволит генерировать игровые поля для головоломок, визуализировать их для проверки корректности и сохранять успешно сгенерированные поля в файл для дальнейшего использования. Программа должна обеспечивать удобный интерфейс для пользователя, позволяя ему выбирать необходимые функции, такие как генерация поля, его визуализация и сохранение.

Для достижения цели проекта необходимо выполнить следующие задачи:

1. Изучить особые черты игры-головоломки «Numbrix» (Нумератор), такие как её правила, механику и особенности генерации игровых полей.
2. Реализовать главное меню с возможностью выбора уровня, сложности и загрузки сохранений, а также интерфейс для ввода и редактирования значений на игровом поле.
3. Обеспечить пользователю возможность сохранения игрового поля в файл, а также взаимодействие с сохраненным полем.
4. Тестирование программы

Разработка программы осуществляется на языке программирования C в среде разработки Microsoft Visual Studio 2022.

# 1 Постановка задачи

«Numbrix» (Нумератор) — это логическая головоломка, в которой игроку необходимо заполнить пустые ячейки прямоугольной сетки числами так, чтобы они образовывали последовательную цепочку от минимального до максимального значения. Числа должны быть соединены по горизонтали или вертикали, перемещение по диагонали не допускается.

Игровое поле головоломки представляет собой квадратную сетку, где каждая ячейка может содержать число или быть пустой. В игровом поле не могут быть отрицательные или повторяющиеся значения. При этом игровое поле также имеет координаты для точного взаимодействия с каждой ячейкой.

Для реализации задачи следует использовать массив указателей на другие массивы типа int. Это предоставляет возможность динамической записи полей разных размеров. Поскольку головоломка представляет собой прямоугольную сетку, где каждая ячейка может содержать число или быть пустой (равна 0), двумерный массив идеально подходит для моделирования такой структуры. Он позволяет легко обращаться к ячейкам по их координатам (буква/цифра). Удобство двумерного массива также заключается в динамическом выделении памяти, что особенно важно, так как в игре предусмотрены уровни разных размеров. Кроме того, двумерный массив легко преобразуется в последовательность данных, что необходимо для сохранения в файл или загрузки из файла впоследствии.

Главное меню данной программы содержит поля:

1. Играть
2. Выбрать уровень
3. Загрузить сохранение
4. Сменить сложность игры
5. Выйти из игры

Необходимо реализовать функционал вывода в консоль главного меню, взаимодействия с ним пользователя, а также исполнение всех опций, представленных в этом меню.

При выборе поля “Играть” необходимо чтобы в консоль было выведено поле уровня, данные для которого находятся в файле уровня, выбранного пользователем со случайным удалением некоторого количества чисел. Количество удаляемых чисел напрямую зависит от выбранной пользователем сложности. Далее происходит сам процесс игры со сгенерированным полем.

Циклично пользователю необходимо ввести координаты (формата буква/цифра) выбираемой им ячейки. При этом проверяется корректность введенных координат (не занята ли ячейка по координатам и верен ли формат введенных координат). Далее пользователем вводится значение для выбранной ячейки. При этом проверяется нет ли уже такого значения на поле.

Введенное значение помещается в ячейку по координатам, после чего у пользователя необходимо спросить желает ли он закончить и сохранить поле сыгранный уровень в файл или нет.

При выборе пункта “Выбрать уровень” пользователю должна предоставляться возможность изменения игрового уровня (играемого поля) при выборе из трёх доступных.

При выборе пункта “Загрузить сохранение” пользователю, должно предоставляться ранее сыгранное и сохраненное в определенный файл сохранения поле. При этом обязательно чтобы ранее пользователем была сыграна некоторое поле и сохранено в файл.

При выборе пункта "Сменить сложность игры" пользователю предлагается выбрать уровень сложности из доступных вариантов:

1. Легко
2. Средне
3. Сложно

При вводе номера уровня сложности выбранное число увеличивается на единицу. Уровень сложности влияет на количество удаляемых чисел перед началом игры (max). Это количество высчитывается по формуле:

где:

* + - * round – округление значение в скобках по правилам математики
* size – размер поля,
* difficult – коэффициент сложности (2 – легко, 3 – средне, 4 – сложно).

При выборе поля “Выйти из игры” необходимо чтобы программа завершала свою работу.

Контрольные примеры

Первый контрольный пример:

Таблица 1 – Контрольный пример 1

|  |  |
| --- | --- |
| Уровень | 1 |
| Сложность | Легко |
| Количество генераций полей | 3 |
| Seed для запуска 1 | 1 |
| Seed для запуска 2 | 2 |
| Seed для запуска 3 | 3 |

Таким образом мы должны получить, при одинаковом исходном поле совершенно разные поля игры (отличия должны заключаться в разных пустых ячейках). При это количество пустых ячеек должно быть одинаковым для каждого поля. Согласно формуле, описанной выше, их должно быть 10 в каждом поле.

Второй контрольный пример:

Таблица 2 – контрольный пример 2

|  |  |
| --- | --- |
| Уровень | 2 |
| Сложность | Средне |
| Количество генераций полей | 1 |

Запустим игру с такими выбранными параметрами и попробуем произвести некорректные операции:

* Изменить значения в клетках, данных условием
* Ввести значение уже имеющееся в матрице
* Выбрать координаты несуществующих клеток на поле (больше по размеру)

Каждый раз при проведении некорректных операций при работе с полем мы должны получать корректную обработку этих действий.

Третий контрольный пример:

Таблица 3 – контрольный пример 3

|  |  |
| --- | --- |
| Уровень | 3 |
| Сложность | Сложно |
| Количество генераций полей | 1 |
| Количество изменяемых ячеек | 3 |

После изменения трех доступных ячеек сохраним полученное поле в файл, перезапустим программу и нажмем “Загрузить сохранение”. Мы должны получить корректно работающее ранее сохраненное поле.

# 2 Конструирование программы

Непосредственно работа программы-головоломки происходит за счёт выбора определенного пункта путем введения его номера. Для реализации стандартного ввода, вывода и дополнительных функций необходимо подключить библиотеки при помощи директивы #include.

Реализацию вывода главного меню и функционал взаимодействия с ним пропишем в функции main(). Она инициализирует переменные и настраивает корректность локализации, выводит описание игры. В бесконечном цикле while выводит главное меню и обрабатывает выбор пользователя: загружает уровень, удаляет случайные числа и запускает игру, выбирает уровень, загружает сохранение, изменяет сложность или завершает программу. Выполняет соответствующие действия в зависимости от выбора пользователя, также при вводе некорректного номера пункта главного меню указывает пользователю на это и просит ввести ещё раз. Функция возвращает 1 при корректной отработке. Блок-схема работы этой функции представлена на рисунке 5.

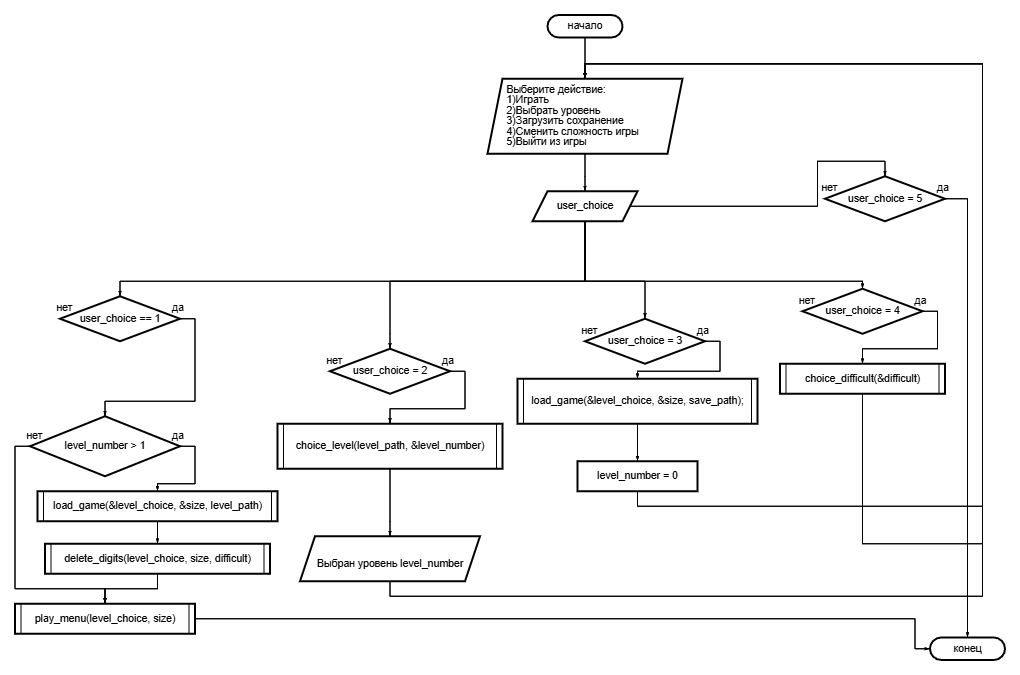


Рисунок 5 – блок-схема функции main

Функция int delete\_digits(int\*\* lvl\_array, unsigned int size, int difficult) вычисляет максимальное количество ячеек, которые нужно удалить, и в цикле max(вычисленное количество) раз генерирует случайные индексы строки и столбца, устанавливая значение ячейки в 0 (вместо 0 во время отрисовки матрицы будет записан пробел). Передаваемый в функцию параметр lvl\_array - двумерный массив, представляющий игровое поле, а size – размер этого двумерного массива (в одну плоскость). Параметр difficult же является коэффициентом сложности, получаемым при выборе сложности пользователем. Возвращает 1 при успешном удалении чисел.

Функция int random\_number(int max) генерирует случайное число с помощью rand() % max + 1 и возвращает его, при этом max – передаваемый параметр максимально возможного числа.

Функция int request\_choice() спрашивает у пользователя "да" или "нет" и возвращает соответствующее значение. Читает символ с клавиатуры, переводит его в нижний регистр с помощью lowercase(), и если символ 'y', возвращает 1; если символ 'n', возвращает 0; в случае других символов, повторяет запрос.

Функция int choice\_level(char\* level\_path) выводит меню выбора уровня, читает выбор пользователя, проверяет его корректность, форматирует level\_path в соответствии с выбранным уровнем и возвращает 1 при успешном выборе уровня. Блок-схема этой функции представлена на рисунке 6.

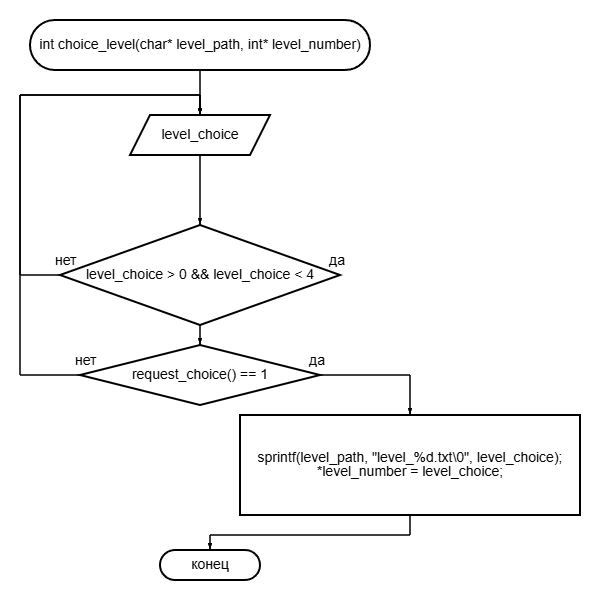


Рисунок 6 – блок-схема функции choice\_level

Функция int play\_menu(int\*\* lvl\_array, int size) в бесконечном цикле while выполняет отрисовку игрового поля с помощью функции pitch(), запрос выбора ячейки с помощью функции choice\_cell(), ввод значения для ячейки с помощью функции choice\_value(), изменение выбранной ячейки с помощью функции change\_cell() и запрос сохранения и выхода с помощью функции choice\_exit(). Передаваемый в функцию параметр lvl\_array - двумерный массив, представляющий игровое поле, а size – размер этого двумерного массива (в одну плоскость). Завершает игру и возвращает 1, если пользователь решает сохранить и выйти.

Функция int pitch(int\*\* lvl\_array, unsigned int size) выводит заголови столбцов с номерами. Затем отрисовывает горизонтальные линии с помощью вызова draw\_lines(). Далее итерируется по каждой строке и столбцу матрицы: если значение ячейки не равно нулю, выводит его форматированным представлением; если значение равно нулю, выводит пустое пространство. После каждой строки вызывает draw\_lines() для отрисовки разделительной линии. Передаваемый в функцию параметр lvl\_array - двумерный массив, представляющий игровое поле, а size – размер этого двумерного массива (в одну плоскость). Функция возвращает 1 при корректной отработке.

Функция int draw\_lines(int size\_lines) выводит символ пробела для выравнивания первой ячейки, затем в цикле выводит символ “-” заданное количество раз. После отрисовки линии переходит на новую строку. Передаваемый в функцию параметр size\_lines – высчитанное необходимое количество знаков “-” для корректного вывода их с полем. Функция возвращает 1 при корректной отработке.

Функция int choice\_cell(int\*\* lvl\_array, int size, int\* choice\_digit, int\* choice\_letter) просит пользователя ввести букву и число для выбора ячейки, проверяет их корректность, присваивает значения параметрам указателям choice\_digit и choice\_letter, запрашивает подтверждение выбора у пользователя с помощью функции request\_choice() и возвращает 1 при успешном выборе ячейки. Передаваемый в функцию параметр lvl\_array - двумерный массив, представляющий игровое поле, а size – размер этого двумерного массива (в одну плоскость). Блок-схема этой функции представлена на рисунке 7.

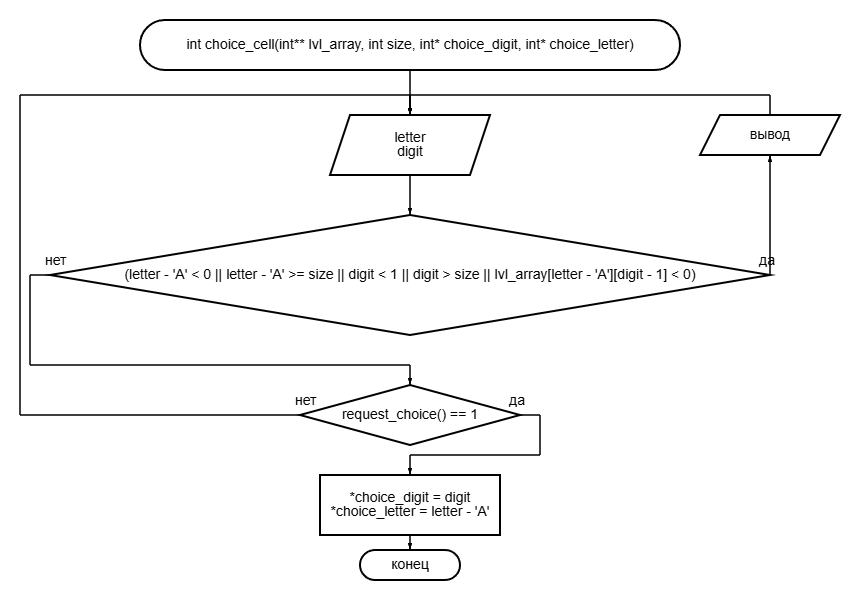
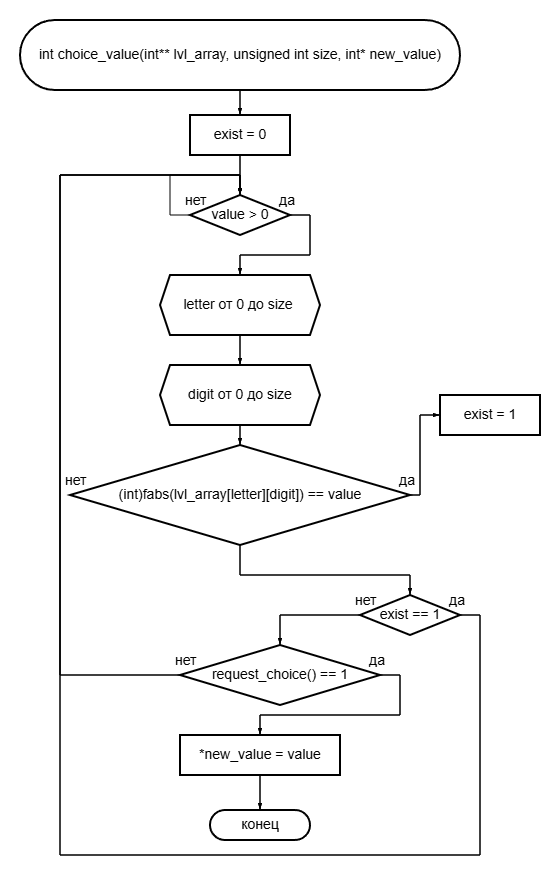


Рисунок 7 – блок-схема функции choice\_cell

Функция int choice\_value(int\*\* lvl\_array, unsigned int size, int\* new\_value) просит пользователя ввести значение, проверяет, что оно не присутствует в матрице, запрашивает подтверждение выбора у пользователя с помощью функции request\_choice(), присваивает значение параметру указателю new\_value и возвращает 1 при успешном выборе значения. Блок-схема этой функции представлена на рисунке 8.

  
Рисунок 8 – блок-схема функции choice\_value

Функция int change\_cell(int\*\* lvl\_array, char letter, int digit, int value) преобразует букву передаваемого параметра letter (букву представляющую строку) в индекс строки, вычитая 'A', и уменьшает значение передаваемого парамаетра digit (числа номера столбца, который ранее выбрал пользователь) на 1 для получения индекса столбца. Затем присваивает параметр value, являющемся новым значением для ячейки, ячейке lvl\_array[letter][digit - 1] через передаваемый параметр указатель на матрицу lvl\_array. Блок-схема этой функции представлена на рисунке 15.

Функция int choice\_exit(int\*\* lvl\_array, unsigned int size) запрашивает у пользователя, хочет ли он сохранить и выйти, и если да, сохраняет состояние игры в файл с помощью функции save\_game(). Передаваемый в функцию параметр lvl\_array - двумерный массив, представляющий игровое поле, а size – размер этого двумерного массива (в одну плоскость). Возвращает 1, если пользователь решает продолжить игру.

Функция int choice\_difficult(int\* difficult) выводит меню выбора сложности, читает выбор пользователя, проверяет его корректность, обновляет значение через передаваемый параметр указатель difficult в соответствии с выбором и возвращает 1 при успешном выборе сложности. Блок-схема этой функции представлена на рисунке 9.

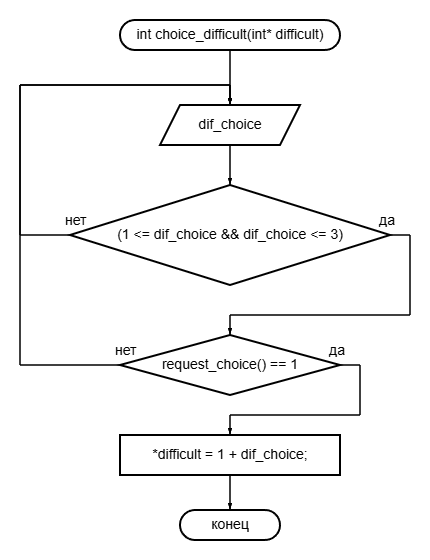


Рисунок 9 – блок-схема функции choice\_difficult

Функция int save\_game(int\*\* lvl\_array, unsigned int size, char path[]) открывает файл по указанному пути в передаваемом параметре path для записи, записывает размер матрицы, а то есть передаваемый параметр size, записывает значения матрицы в файл, а точнее записывает значения из передаваемого параметра lvl\_array, закрывает файл, выводит сообщение о сохранении и возвращает 1 при успешном сохранении.

Функция int load\_game(int\*\*\* lvl\_array, unsigned int\* size, char path[]) открывает файл по указанному пути в переданном параметре path для чтения, читает размер матрицы и присваивает его параметру указателю size, выделяет память для матрицы, которая передается параметром указателем lvl\_array, размером size x size, читает значения матрицы из файла и записывает их в матрицу lvl\_array, закрывает файл и возвращает 1 при успешной загрузке.

# 3 Тестирование программы

При запуске программы нам выводится главное меню. Согласно первому контрольному примеру, запустим первый уровень (стандартные значения запуска игры без умышленных изменения номера уровня и сложности есть уровень 1 и сложность “Легко”). Таким образом у нас запустится поле с некоторыми удаленными значениями. Далее перезапустим нашу программу с аналогичными значениями уровня и сложности ещё два раза и получим совершенно разные поля. Все действия и результат представлены на рисунке 10.

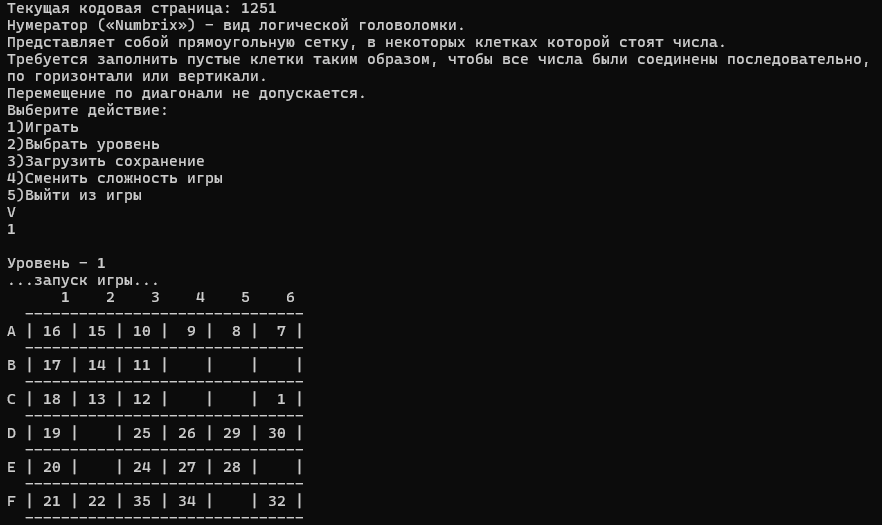
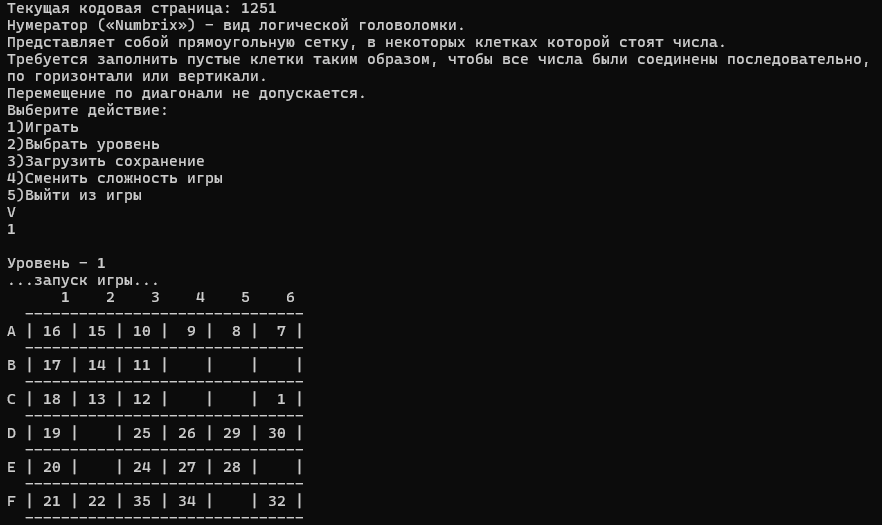
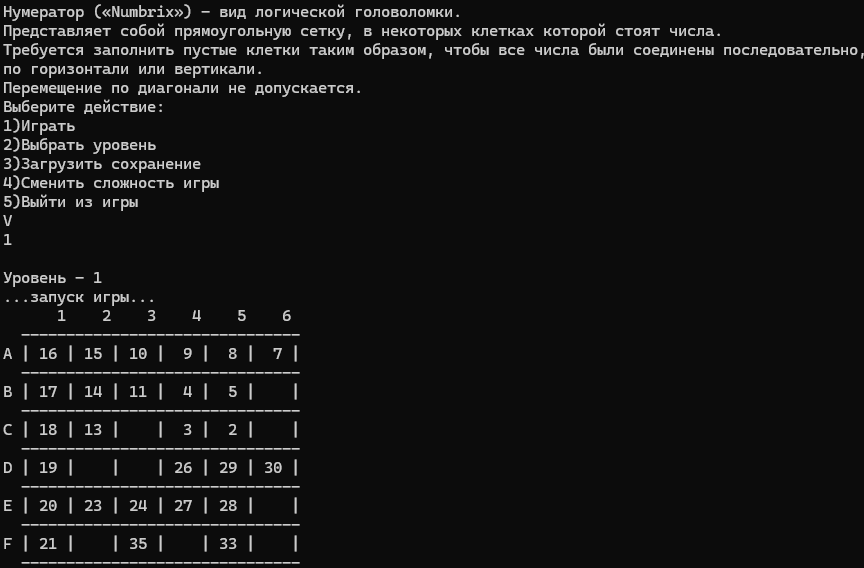
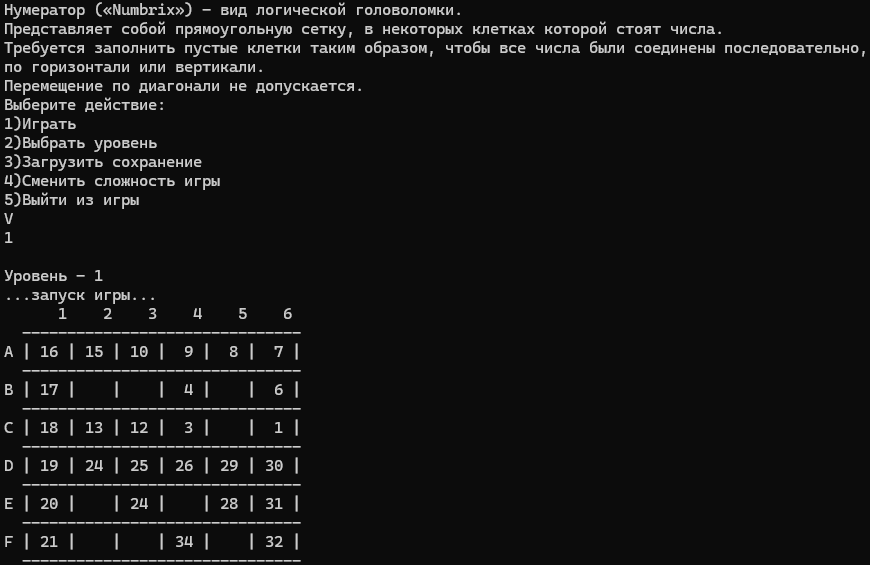
  

Рисунок 10 - главное меню и выполнение контрольного примера 1

Далее в главном меню нашей игры введем 2 для выбора уровня, после чего у нас откроется меню смены уровня, представленное на рисунке 20. Выберем необходимый нам уровень 2 и подтвердим выбор. Затем следует сменить сложность игры, для этого введем 4 для открытия меню сложности игры, в котором выберем сложность “Средне” введя 2. Далее запустим игру с такими параметрами и получим ожидаемое поле. Затем попробуем ввести некорректные данные: координаты уже занятой (именно по игровому условию) ячейки, значение, уже существующее на поле и координаты, большие размерности поля. Получим корректную обработку наших действий. Все проделанные действия представлены на рисунках 11, 12, 13.

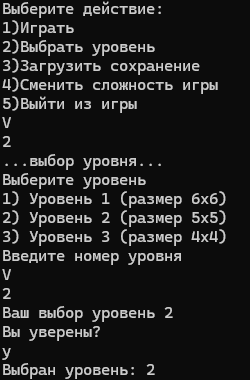
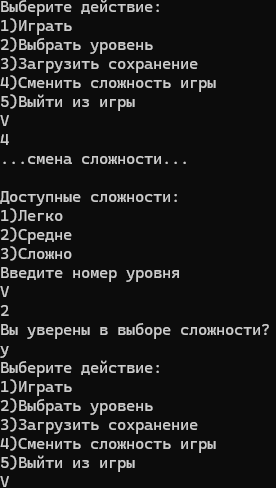
 

Рисунок 11 - смена уровня и сложности

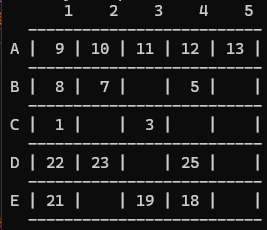


Рисунок 12 - полученное поле для контрольного примера 2

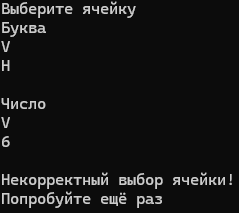
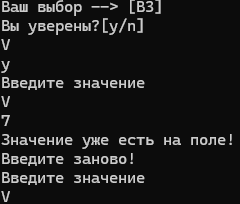
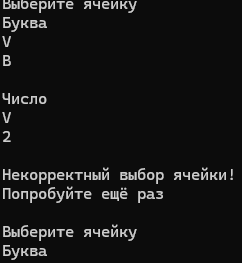
 ’

Рисунок 13 - проверка правильности обработки ввода некорректных данных

Для исполнения контрольного примера 3, выполним действия аналогично прошлому пункту, лишь в меню выбора уровня выберем уровень 3 и в меню выбора сложности выберем “Сложно”. Далее запустим игру и добавим значение в 3 пустых клетки. После 3 добавления сохраним и выйдем из игры. В дальнейшем запустим игру ещё раз и в главном меню введя 3 для загрузки сохранения загрузим сыгранное ранее поле. Тем самым нам выведется в консоль поле из прошлой игры, и мы сможем продолжить игру. Все действия, описанные в этом пункте, представлены на рисунках 14, 15, 16, 17.

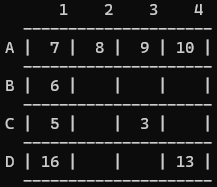


Рисунок 14 - поле для контрольного примера 3

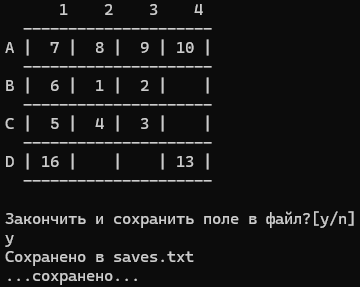


Рисунок 15 - сохранение поля со вставленными числами

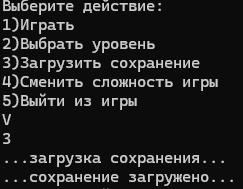


Рисунок 16 - перезапуск и загрузка ранее сохраненного поля

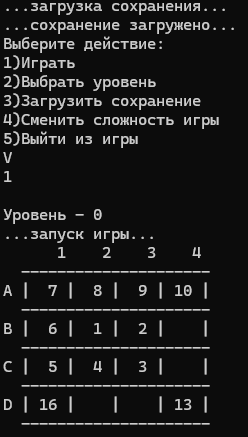


Рисунок 17 - запущено сохраненное поле

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была разработана программа-головоломка "Numbrix" на языке программирования C. Программа обеспечивает генерацию игровых полей различного уровня сложности, их визуализацию в консоли и сохранение в файл для дальнейшего использования.

В ходе работы были реализованы генерация игрового поля с случайным удалением чисел в зависимости от выбранного уровня сложности, взаимодействие с пользователем через консольное меню для выбора уровня, сложности и загрузки сохранений, возможность изменения значений ячеек и сохранения состояния игры в файл.

Код на GitHub:

https://github.com/wellwhyn0t/KP\_WORK

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. О. В. Минакова, О. В. Курипта: «Практикум по си»: [Интернет-ресурс]: https://sites.google.com/view/course-of-study1-c/
2. Сайт о программировании: [Интернет-ресурс]: <https://metanit.com/>
3. Онлайн-редактор блок-схем: [Сайт] – URL: <https://programforyou.ru/block-diagram-redactor>
4. Онлайн справочник по языку СИ: [Интернет-ресурс]: http://www.c-cpp.ru/books
5. Онлайн конспект лекций по языку СИ: [Интернет-ресурс]: http://natalia.appmat.ru/c&c++/index.php
6. Форум программистов CyberForum: [Интернет-ресурс]: <https://www.cyberforum.ru/c/>
7. Сайт Microsoft о Visual Studio: [Интернет-ресурс]: <https://learn.microsoft.com/en-us/previous-versions/60k1461a(v=vs.140)?redirectedfrom=MSDN>
8. Пособие по языку си от Microsoft: [Интернет-ресурс]: https://learn.microsoft.com/en-us/cpp/c-language/c-language-reference?view=msvc-170&redirectedfrom=MSDN
9. Иллюстрированный самоучитель по практике программирования: [Интернет-ресурс]: https://paulfertser.info/k\_and\_p/index.html
10. Онлайн пособие по языкам программирования: [Интернет-ресурс]: https://code.mu/ru/
11. Язык Си матрицы: [Интернет-ресурс]: <https://istarik.ru/blog/programmirovanie/13.html>
12. Краткое руководство по языку СИ: [Интернет-ресурс]: https://labs-org.ru/c-4/

# ПРИЛОЖЕНИЕ А

#include <stdio.h>

#include <locale.h>

#include <math.h>

#include <time.h>

#include <stdlib.h>

/\*

выводит в консоль главное меню с описанием игры, предоставляет функционал выбора действий

возвращает 1 при корректной отработке программы

\*/

int main();

/\*

отрисовывает поле в консоли

lvl\_array - матрица, size - размер матрицы

возвращает 1 при успешной отрисовки поля

\*/

int pitch(int\*\*, unsigned int);

/\*

отрисовывает линии определенного размера в консоли

size\_lines - размер горизонтальной линии

возвращает 1 если отрисовка линии прошла успешно

\*/

int draw\_lines(int);

/\*

меняет в матрице значение по координатам формата [БУКВА\ЦИФРА] на новое значение

lvl\_array - матрица, letter - буква координаты, digit - число координаты, value - новое значение

возвращает 1 при успешной смене ячейки

\*/

int change\_cell(int\*\*, char, int, int);

/\*

выводит в консоль меню выбора уровня, дает пользователю сделать выбор уровня по его номеру

возвращает указатель на строку, являющуюся адресом файла с текстом уровня

\*/

int choice\_level(char\*, int\*);

/\*

игровое меню, вызванное из главного меню, проводящее действия для изменения матрицы

lvl\_array - матрица, size - размер матрицы

возвращает 1 при окончании игры пользователем

\*/

int play\_menu(int\*\*, int);

/\*

загружает матрицу, её размер с файла по пути (будь то матрица уровня или сохранения)

lvl\_array - матрица в которую записываются значения из файла, size - размер матрицы, path - путь к файлу

возвращает 1 при успешной загрузки с файла

\*/

int load\_game(int\*\*\*, unsigned int\*, char[]);

/\*

удаляет случайные числа из матрицы

lvl\_array - матрица, size - размер матрицы, difficult - коэффицент для расчёта сложности

возвращает 1 при успешном удалении чисел

\*/

int delete\_digits(int\*\*, unsigned int, int);

/\*

генерирует случайное число от 0 до max

max - максимальное генерируемое число

возвращает случайное число

\*/

int random\_number(int);

/\*

проверяет уверен ли пользователь в выборе чего-либо (да\нет)

возвращает 1 - если да, 0 - если нет

\*/

int request\_choice();

/\*

спрашивает у пользователя координаты ячейки(формата[БУКВА / ЦИФРА]) и, проверив корректность выбранной ячейки,

присваивает его переменным координат из меню

lvl\_array - матрица, size - размер матрицы, choice\_digit - выбранная цифра координаты, choice\_letter - выбранная буква координаты

возвращает 1 при успешном выборе координат пользователем

\*/

int choice\_cell(int\*\*, int, int\*, int\*);

/\*

спрашивает у пользователя число для заполнения ячейки и, проверив корректность выбранного числа,

присваивает его переменной числа из меню

lvl\_array - матрица, size - размер матрицы, new\_value - новое значение, вводимое пользователем)

возвращает 1 при успешном выборе нового значения

\*/

int choice\_value(int\*\*, unsigned int, int\*);

/\*

спрашивает у пользователя закончить ли игру и, при утвердительном ответе, сохраняет в файл сыгранную матрицу и её размер

lvl\_array - матрица, size - размер матрицы

возвращает 1 при желании пользователя продолжить игру

\*/

int choice\_exit(int\*\*, unsigned int);

/\*

меню выбора сложности (Легко, Средне, Сложно)

difficult - коэффицент сложности

возвращает 1 при успешной смене сложности

\*/

int choice\_difficult(int\*);

/\*

сохраняет матрицу и её размер по указанному пути

lvl\_array - матрица, size - размер матрицы, path - путь к файлу для сохранения

возвращает 1 при успешном сохранении в файл

\*/

int save\_game(int\*\*, unsigned int, char[]);

int main()

{

system("chcp 1251");

setlocale(LC\_ALL, "ru");

puts("Нумератор («Numbrix») - вид логической головоломки.\nПредставляет собой прямоугольную сетку, в некоторых клетках которой стоят числа. \nТребуется заполнить пустые клетки таким образом, чтобы все числа были соединены последовательно,\nпо горизонтали или вертикали.\nПеремещение по диагонали не допускается.");

int level\_number = 1; /\* номер запускаемого уровня (исходно - 1) \*/

int user\_choice = -1; /\* номер пункта, выбираемый пользователем \*/

int difficult = 2; /\* коэффициент сложности \*/

int\*\* level\_choice = NULL; /\* матрица чисел для игры \*/

unsigned int size; /\* размер матрицы \*/

char level\_path[20] = "level\_1.txt\0"; /\* путь к уровню (исходно к уровню 1) \*/

static char save\_path[] = "saves.txt\0"; /\* путь к файлу с сохранением \*/

srand(time(NULL)); /\* инициализирует генератор случайных чисел \*/

/\* меню выбора \*/

while (1)

{

puts("Выберите действие: ");

puts("1)Играть\n2)Выбрать уровень\n3)Загрузить сохранение\n4)Сменить сложность игры\n5)Выйти из игры\nV");

scanf("%d", &user\_choice);

switch (user\_choice)

{

case 1:

/\* выбрано "Играть" \*/

printf("\nУровень - %d", level\_number);

if (level\_number > 0)

{

load\_game(&level\_choice, &size, level\_path);

delete\_digits(level\_choice, size, difficult);

}

play\_menu(level\_choice, size);

return 1;

case 2:

/\* выбрано "Выбрать уровень" \*/

puts("...выбор уровня...");

choice\_level(level\_path, &level\_number);

printf("Выбран уровень: %d\n", level\_number);

break;

case 3:

/\* выбрано "Загрузить сохранение" \*/

puts("...загрузка сохранения...");

load\_game(&level\_choice, &size, save\_path);

level\_number = 0;

puts("...сохранение загружено...");

break;

case 4:

/\* выбрано "Сменить сложность игры" \*/

choice\_difficult(&difficult);

break;

case 5:

/\* выбрано "Выйти из игры" \*/

puts("...выключение игры...");

return 1;

default:

/\* ошибка выбора пункта \*/

puts("Выбран неверный пункт, попробуйте ещё раз");

break;

}

}

}

// Функции для работы с матрицей

int delete\_digits(int\*\* lvl\_array, unsigned int size, int difficult)

{

double difficult\_parametr = (double)(size \* size - 1) / 8. + 1;

int max = difficult\_parametr \* difficult; // максимальное число пустых полей в матрице

for (int counter = 0; counter < max; counter++)

lvl\_array[random\_number(size - 1)][random\_number(size - 1)] = 0;

return 1;

}

int random\_number(int max)

{

return (rand() % max) + 1;

}

int change\_cell(int\*\* lvl\_array, int letter, int digit, int value)

{

digit -= 1; // число понижается на 1 для получения именно индекса матрицы

lvl\_array[letter][digit] = value;

return 1;

}

int play\_menu(int\*\* lvl\_array, int size)

{

while (1)

{

int choice\_digit, choice\_letter, new\_value;

puts("\n...запуск игры...");

/\* отрисовка поля \*/

pitch(lvl\_array, size);

/\* выбор ячейки \*/

choice\_cell(lvl\_array, size, &choice\_digit, &choice\_letter);

/\* ввод значения \*/

choice\_value(lvl\_array, size, &new\_value);

/\* изменения значения в ячейке \*/

change\_cell(lvl\_array, choice\_letter, choice\_digit, new\_value);

/\* отрисовка изменённого поля \*/

pitch(lvl\_array, size);

/\* сохранение и выход при желании \*/

if (choice\_exit(lvl\_array, size) == 1)

return 1;

}

}

// Функции для взаимодействия с пользователем

int choice\_level(char\* level\_path, int\* level\_number)

{

int level\_choice = -1;

while (1) {

puts("Выберите уровень\n1) Уровень 1 (размер 6х6)\n2) Уровень 2 (размер 5х5)\n3) Уровень 3 (размер 4х4)\nВведите номер уровня\nV");

scanf("%d", &level\_choice);

/\* проверяем корректность введенного номера уровня \*/

if (level\_choice > 0 && level\_choice < 4) {

printf("Ваш выбор уровень %d\nВы уверены?\n", level\_choice);

if (request\_choice() == 1) {

sprintf(level\_path, "level\_%d.txt\0", level\_choice); // Формируем путь к файлу уровня

\*level\_number = level\_choice; // Обновляем номер уровня

return 1;

}

else puts("\nВыберите ещё раз!");

}

else puts("\nНекорректный номер уровня, попробуйте ещё раз!");

}

}

int choice\_cell(int\*\* lvl\_array, int size, int\* choice\_digit, int\* choice\_letter)

{

char letter; // буква координаты поля

int digit; // число координаты поля

while (1)

{

puts("\nВыберите ячейку\nБуква\nV");

getchar();

scanf("%c", &letter);

puts("\nЧисло\nV");

scanf("%d", &digit);

// проверка корректности выбранных координат ячейки

if (letter - 'A' < 0 || letter - 'A' >= size || digit < 1 || digit > size || lvl\_array[letter - 'A'][digit - 1] < 0)

puts("\nНекорректный выбор ячейки!\nПопробуйте ещё раз");

else

{

printf("Ваш выбор --> [%c%d]\nВы уверены?[y/n]\nV\n", letter, digit);

if (request\_choice() == 1)

{

\*choice\_digit = digit;

\*choice\_letter = letter - 'A';

return 1;

}

}

}

}

int choice\_value(int\*\* lvl\_array, unsigned int size, int\* new\_value)

{

int value;

int exist; /\* переменная указывающая есть ли значение уже на поле(1 - да; 0 - нет) \*/

while (1)

{

puts("Введите значение\nV");

scanf("%d", &value);

if (value > 0)

{

/\* проверка есть ли уже значение в матрице \*/

exist = 0;

for (int letter = 0; letter < size; letter++)

{

for (int digit = 0; digit < size; digit++)

{

if ((int)fabs(lvl\_array[letter][digit]) == value)

{

printf("Значение уже есть на поле!\n");

exist = 1;

break;

}

if (exist == 1) break;

}

if (exist == 1) break;

}

if (exist == 0)

{

printf("Ваш выбор значения--> [%d]\nВы уверены?[y/n]\nV\n", value);

if (request\_choice() == 1)

\*new\_value = value;

return 1;

}

}

printf("Введите заново!\n");

}

}

int choice\_exit(int\*\* lvl\_array, unsigned int size)

{

puts("\nЗакончить и сохранить поле в файл?[y/n]");

char save\_path[] = "saves.txt\0";

if (request\_choice() == 1)

{

save\_game(lvl\_array, size, save\_path);

return 1;

}

}

int choice\_difficult(int\* difficult)

{

puts("...смена сложности...");

int dif\_choice; // вводимое значение сложности, используемое для проверки условия

puts("\nДоступные сложности:\n1)Легко\n2)Средне\n3)Сложно");

while (1)

{

puts("Введите номер уровня\nV");

scanf("%d", &dif\_choice);

if (1 <= dif\_choice && dif\_choice <= 3)

{

// проверка корректности выбора уровня

printf("Вы уверены в выборе сложности?\n");

if (request\_choice() == 1)

{

\*difficult = 1 + dif\_choice;

return 1;

}

}

puts("\nВведён некорректный номер уровня, попробуйте ещё раз");

}

}

int request\_choice()

{

char request;

while (1)

{

getchar();

scanf("%c", &request);

request = tolower(request);

if (request == 'y')

return 1;

if (request == 'n')

return 0;

puts("Нет, выбери [y/n]");

}

}

// Функции для отрисовки поля

int draw\_lines(int size\_lines)

{

for (int i = 0; i < size\_lines; i++)

{

if (i == 0) printf("%2c", ' ');

printf("%c", '-');

}

printf("\n");

return 1;

}

int pitch(int\*\* lvl\_array, unsigned int size)

{

int size\_lines = size \* 5 + 1; // количество горизонтальных линий для отделения строк ("---")

printf("%4c", ' ');

for (int cell = 0; cell < size; cell++)

printf("%3d ", cell + 1);

printf("\n");

draw\_lines(size\_lines);

for (int letter = 0; letter < size; letter++)

{

printf("%c ", 'A' + letter);

for (int digit = 0; digit < size; digit++)

{

if (lvl\_array[letter][digit] != 0)

printf("|%3d ", (int)fabs(lvl\_array[letter][digit]));

else

printf("|%4c", ' ');

}

printf("|\n");

draw\_lines(size\_lines);

}

return 1;

}

// Функции для работы с файлами

int save\_game(int\*\* lvl\_array, unsigned int size, char path[])

{

FILE\* file;

file = fopen(path, "w");

if (file == NULL)

{

perror("Ошибка открытия файла для записи");

return 0;

}

fprintf(file, "%u\n", size);

for (int letter = 0; letter < size; letter++)

{

for (int digit = 0; digit < size; digit++)

{

fprintf(file, "%d ", lvl\_array[letter][digit]);

}

fprintf(file, "\n");

}

fclose(file);

printf("Сохранено в %s\n", path);

puts("...сохранено...");

return 1;

}

int load\_game(int\*\*\* lvl\_array, unsigned int\* size, char path[])

{

FILE\* file;

file = fopen(path, "r");

if (file == NULL)

{

perror("Ошибка открытия файла для чтения");

return 0;

}

fscanf(file, "%u", size);

\*lvl\_array = (int\*\*)malloc(\*size \* sizeof(int\*));

if (\*lvl\_array == NULL)

{

perror("Ошибка выделения памяти для матрицы");

fclose(file);

return 0;

}

for (int letter = 0; letter < \*size; letter++)

{

(\*lvl\_array)[letter] = (int\*)malloc(\*size \* sizeof(int));

if ((\*lvl\_array)[letter] == NULL) {

perror("Ошибка выделения памяти для списка");

fclose(file);

return 0;

}

for (int digit = 0; digit < \*size; digit++)

{

fscanf(file, "%d", &(\*lvl\_array)[letter][digit]);

}

}

fclose(file);

return 1;

}

# ПРИЛОЖЕНИЕ Б

Уровень 1 (level\_1.txt)

6

-16 -15 -10 -9 -8 -7

-17 -14 -11 -4 -5 -6

-18 -13 -12 -3 -2 -1

-19 -24 -25 -26 -29 -30

-20 -23 -24 -27 -28 -31

-21 -22 -35 -34 -33 -32

Уровень 2 (level\_2.txt)

5

-9 -10 -11 -12 -13

-8 -7 -6 -5 -14

-1 -2 -3 -4 -15

-22 -23 -24 -25 -16

-21 -20 -19 -18 -17

Уровень 3 (level\_3.txt)

4

-7 -8 -9 -10

-6 -1 -2 -11

-5 -4 -3 -12

-16 -15 -14 -13