МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра автоматики и процессов управления

КУРСОВОЙ ПРОЕКТ

по дисциплине «Алгоритмы и структуры данных»

Тема: разработка программы судоку «Advanced»

Студент гр. 1326 Раскин А.Р.

Преподаватель Спиридонов Р.Е.

Санкт-Петербург

2023г

Содержание

ВВЕДЕНИЕ

В данном курсовом проекте представлена актуальная на сегодняшний день игра под названием судоку. Что же это за игра и как в неё играть?

Судоку - это квадрат размером 9х9. Он разделен на 9 небольших квадратиков 3х3. В некоторых клетках в начале игры стоят числа от 1 до 9. Цель игры - заполнить пустые клетки по специальному правилу. Это задача может быть и простой и невозможно сложной, в зависимости от предложенного варианта. В судоку хорошо играть в транспорте по дороге на работу или домой.

Правила игры в судоку.

Правила игры судоку совершенно просты. Игровое поле судоку состоит из 81 клетки, находящихся в 9 столбцах, 9 строках и 9 малых квадратах. Это собственно и есть магический квадрат судоку. В зависимости от уровня сложности некоторые клетки уже содержат числа. Ваша задача - заполнить остальные клетки, используя Ваши серые клеточки. Необходимо обратить внимание на следующее правило: В игре участвую только числа с 1 до 9. Игровое поле (квадрат 9х9) должен быть заполнен таким образом, что, каждое число (с 1 до 9) встречается в каждой строке, в каждом столбце и в каждом малом квадрате (3х3) только один-единственный раз. Правильная головоломка имеет одно решение.

5	3	**9**	8	7	6	**4**	1	2
7	2	**8**	**3**	1	**4**	**9**	6	5
6	4	**1**	2	9	5	**7**	**3**	8
4	**6**	**2**	**5**	3	9	8	7	1
3	**8**	**5**	**7**	**2**	1	**6**	**4**	**9**
1	9	**7**	**4**	6	**8**	**2**	5	3
2	5	6	**1**	8	7	3	9	**4**
9	1	**3**	6	**4**	2	5	**8**	**7**
8	**7**	4	9	**5**	**3**	1	2	**6**

Рис. 1 Пример решённого судоку

Что представлено в курсовом проекте?

В курсовом проекте представлена игровая программа, способная составлять судоку и решать его на любом этапе игрового процесса.

Игра должна иметь:

1. Алгоритм решения произвольного игрового поля и уже заданной игры судоку.
2. Алгоритм генерации случайного игрового поля судоку.
3. Выбор размерности поля.
4. Выбор алфавита игры.

5. Выбор алфавита игры.
6. Функция подсказки.
7. Функция проверки.
8. Функция ведения статистики игр.

Результатом данного курсового проекта является игра судоку с интерфейсом, которая сохраняет статистику игр между запусками и быстро обрабатывает проверки корректности решения после каждого хода игрока.

В первой главе «Теоретическая часть» рассматриваются возможности Microsoft Visual Studio, описывается алгоритм работы программы, разбирается теория и основные задачи курсовой работы.

Во второй главе «Проектирование и разработка системы» описывается написанная программа, тестирование и результаты работы.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Описание предметной области

Судоку различаются по уровню сложности в зависимости от размера квадрата - для профессионалов существуют судоку 15x15 и 16x16 клеток. У нас же в проекте мы будем рассматривать более простые версии: 2x2, 3x3, 4x4 и 5x5.

После запуска программы вам будет дан выбор какой размер поля выбрать. Потом будет предложена символика данных на поле (цифры или буквы). Далее вы сможете ввести своё собственное поле или довериться компьютеру, который составит поле за вас. После вас ждет последний выбор перед началом игры, это выбор сложности, от него зависит сколько пустых клеток будет у вас на поле.

Это моя собственная интерпретация классической игры судоку с некоторыми добавлениями для более веселой и долгой игры.

1.2 Требования к программе

Задачей курсового проекта является разработка и создание игры судоку, позволяющей любому желающему насладиться этой увлекательной головоломкой.

Общие требования к программе:

1) Программа должна поддерживать систему меню, пункты которых соответствуют выполнению функций, предусмотренных общей частью задания.
2) Должны быть реализованы функции подсказки и проверки введенного значения с дальнейшей цветовой маркировкой.
3) Интерфейс программы должен быть понятен и содержать проверки на неправильно введенные данные.

1.3. Архитектура системы и выбор средств разработки

Разработка программы производиться на языке программирования C++, на базе операционной системе Windows.

Данный язык программирования за счёт того, что он является высокоуровневым компилируемым языком программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день C++ является одним из самых популярных и распространенных языков.

В отличие от Си язык C++ позволяет писать приложения в объектно-ориентированном стиле, представляя программу как совокупность взаимодействующих между собой классов и объектов. Что упрощает создание крупных приложений.

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов.

Эта интегрированная среда разработки была выбрана за счёт: IntelliCode — это мощный набор средств автоматического завершения кода, подробной аналитике о коде, простоте интерфейса и интегрируемой отладке. Все эти вещи упростили и ускорили выполнение поставленной задачи.

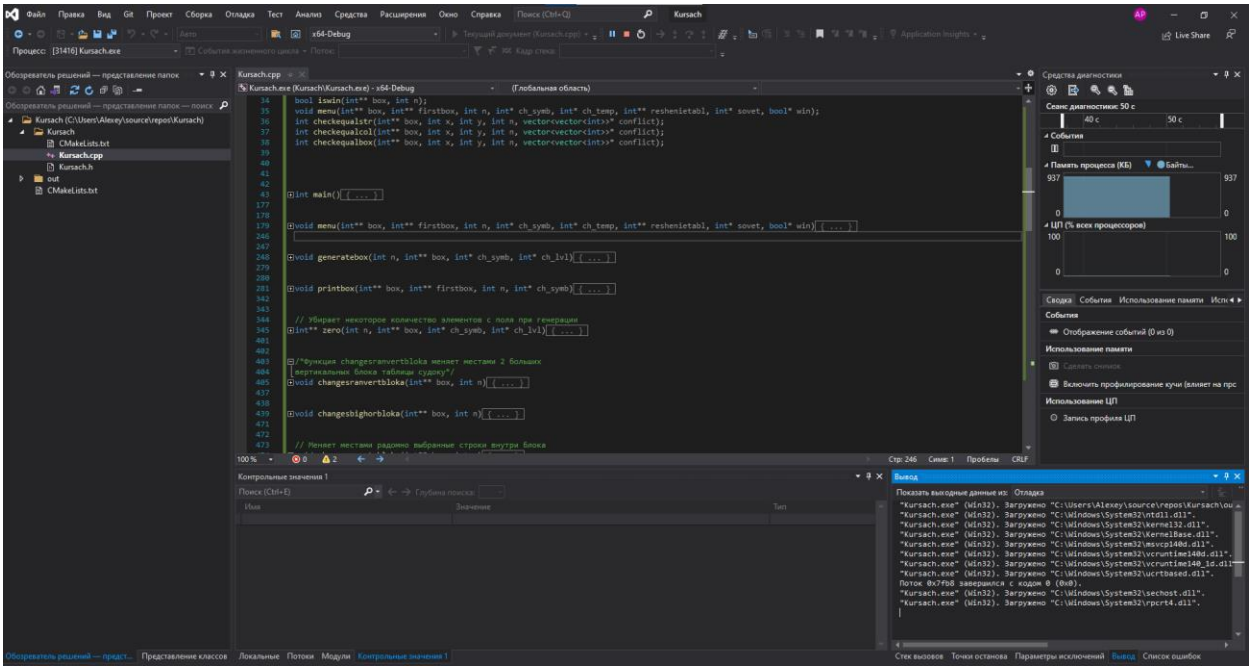

Рис. 2 Интерфейс Visual Studio 2019

Также, Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода, добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения.

2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ

2.1. Описание функциональности программы и использованных библиотек

Код программы должен выполнять стандартные задачи для игры судоку. Такие как, ввод и удаление элемента в поле, ввод начальных данных из файла и сохранение статистики игр в файл, вызов подсказки и проверка на наличие решения в данной игре.

В данном проекте для создания приложения необходимо подключить следующие библиотеки:

1) Iostream – объявляет объекты, управляющие чтением из стандартных потоков и записью в них.
2) String – класс с методами и переменными для организации работы со строками в языке программирования C++.
3) Fstream – определяет несколько классов, поддерживающих операции iostreams для последовательностей, хранящихся во внешних файлах.
4) Windows.h – является специфичным для Windows заголовочным файлом для языков программирования C и C ++, который содержит объявления для всех функций в API Windows, всех распространенных макросов, используемых программистами Windows, и всех типов данных, используемых различными функциями и подсистемы.
5) Ctime – включает время заголовка стандартной библиотеки C и добавляет связанные имена в std пространство имен.
6) Vector - это шаблон класса для контейнеров последовательности. Вектор хранит элементы заданного типа в линейном расположении и обеспечивает быстрый случайный доступ к любому элементу.

2.2. Функциональность программы

Главная часть программы состоит из функций, в которых реализованы определённые команды. В таблице 1 представлены основные функции и их краткие описания.

Таблица 1. Описание основных функций программы

Название функции	Описание
Advice	Функция advice подсказывает пользователю следующих ход и заполняет одну клетку на поле, приближая пользователя к победе
Statistica	Записывает статистику в файл stat.txt после каждой партии
Printbox	Печатает матрицу поля в виде судоку, раскрашивая при этом конфликтующие элементы и элементы поставленные изначально
Checkequal	Объединяет все функции проверки совпадения, проверяет корректность постановки элементов на поле
CheckCin	Проверяет корректность постановки на поле символов, букв и номеров ячеек, в которые производится постановка
Isemptyelment	Функция isemptyel проверяет, остались ли в т ице судоку незаполненные ячейки

Findemptyelement	Функция findemptycnt считает количество пустых ячеек в таблице судоку
Fillsolvet	Функция fillsolvet запоняет первые 2 строки массива решений solvetable, содержащие координаты незаполненных ячеек (в строке с индексом 0 хранятся номера строк пустых ячеек, а в строке с индексом 1 - соответствующие номера столбцов)
Countsolves	Функция countsolves заполняет массив решений solvetable и возвращает количество решений судоку
Entry	Выводит начальное меню игры, считывает характеристики поля
Zero	Убирает некоторое количество элементов с поля при генерации
Generatebox	Формирование базовой таблицы судоку
Put	Дает пользователю возможность поставить символ на поле
Remove	Дает пользователю возможность удалить символ на поле
Iswin	Возвращает true если пользователь заполнил поле правильно, иначе - false

Menu	С помощью функции menu осуществляется навигация по действиям, которые можно выполнять в программе
Changesranvertbloka	Функция changesranvertbloka меняет местами 2 больших вертикальных блока таблицы судоку
Changesrancolumnsbloka	Функция changesrancolumnsbloka меняет местами 2 больших вертикальных блока таблицы судоку
Resheniesudoku19	Рекурсивная функция solvesudoku19 решает судоку, пытаясь подставить в незаполненные ячейки значения от 1 до n*n, если это возможно
Resheniesudoku91	Рекурсивная функция solvesudoku91 решает судоку, пытаясь подставить в незаполненные ячейки значения от n*n до 1, если это возможно

Реализацию данных функций можно увидеть в Приложении П1.

2.3. Тестирование системы и описание дополнительных функций

При тестировании программы проблем и ошибок не было обнаружено. Программа работает правильно, отвечает всем пунктам технического задания и не нуждается в устранении неполадок. Программу можно и дальше дорабатывать, таким образом любой пользователь сможет воплотить свои задумки и идеи, которые он бы хотел увидеть в данной игре. Можно внедрить данную программу на сайт, сделать более принятый интерфейс, сделать более большие размеры поля или игру с ограничение по времени.

3. ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были полностью реализованы поставленные цели и задачи, а именно:

1. Анализ предметной области задачи
2. Изучение различных алгоритмов и выбор самого практичного для решения поставленной задачи.
3. Реализация программы в объектно-ориентированном стиле по принципам модульности и иерархичности.

Игра имеет:

1. Алгоритм решения произвольного игрового поля и уже заданной игры судоку.
2. Выбор алфавита игры и размера поля.
3. Алгоритм генерации случайного игрового поля судоку.
4. Алгоритм проверки, показывающий сколько решений, существует на данный момент.
5. Возможность ведения статистики игр в отдельный файл.

При разработке программе использовались функции, массивы, циклы, указатели и алгоритмы, изученные в течении курса. Разработанная программа позволяет играть в классическую игру судоку на разных уровнях сложности. Данная курсовая работа позволила укрепить знания, которые были получены в процессе учебы, и реализовать их в виде данной программы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Изучаем C++ через программирование игр / Доусон Майкл. – Изд. СПб, 2021. – 109 с.
2. Павловская Т.А. С++. Программирование на языке высокого уровня. Учебник для вузов. - СПб.: Питер, 2007. - 432 с.
3. Электронная энциклопедия. [Электронный ресурс], URL: http://ru.wikipedia.org/.
4. Области исследований, использование. [Электронный ресурс], URL: http://office.microsoft.com/.
5. Объектно-ориентированное программирование в С++/ Лафоре Роберт, 2018. – 59с.
6. Язык программирования С++/ Стивен Прата, 2001. – 147с.
7. Интернет-ресурс [Электронный ресурс], StuDocu - URL: https://www.studocu.com/ru/

ПРИЛОЖЕНИЕ. ЛИСТИНГИ

Листинг П1. Код осинового файла cpp.

```
#include <iostream>
#include <windows.h>
#include <string>
#include <ctime>
#include <vector>
#include <fstream>

using namespace std;

void advice(int** box, int n, int** reshenietabl);
void statistica(int sovet, bool finality, int n, int lvl);
void printbox(int** box, int** firstbox, int n, int* ch_symb);
bool checkequal(int** box, int x, int y, int n, int k);
int OneSymbTranslat(char symb);
int SimpleCheckSIn();
int CheckCin(int* ch_symb, int N);
void translattoint(int N, char** boxchar, int** box);
bool isemptyelment(int** box, int n, int& i, int& j);
int findemptyelement(int** box, int n);
void fillsolvet(int** box, int n, int** reshenietabl);
bool resheniesudoku19(int** box, int n);
bool resheniesudoku91(int** box, int n);
int countsolves(int** box, int n, int** reshenietabl);
int proverkagetchoice(int max);
int entry(int* ch_size, int* ch_lvl, int* ch_symb, int* ch_file);
int** zero(int n, int** box, int* ch_symb, int* ch_lvl);
void changesranvertbloka(int** box, int n);
void changesbighorbloka(int** box, int n);
void changesranstrbloka(int** box, int n);
void changesrancolumnsbloka(int** box, int n);
void generatebox(int n, int** box, int* ch_symb, int* ch_lvl);
void put(int** box, int** firstbox, int n, int* ch_symb, int* ch_temp, int**
reshenietabl);
void remove(int** box, int** firstbox, int n, int* ch_symb, int* ch_temp,
int** reshenietabl);
bool iswin(int** box, int n);
void menu(int** box, int** firstbox, int n, int* ch_symb, int* ch_temp, int**
reshenietabl, int* sovet, bool* win);
int checkequalstr(int** box, int x, int y, int n, vector<vector<int>>*
conflict);
int checkequalcol(int** box, int x, int y, int n, vector<vector<int>>*
conflict);
int checkequalbox(int** box, int x, int y, int n, vector<vector<int>>*
conflict);



int main()
{
    setlocale(LC_ALL, "rus");
    int ch_size, ch_lvl, ch_symb, ch_file, sovet = 0, ch_temp = 1;
    int work = 2;
    bool win;
```

```
    while (work == 2)
    {

        entry(&ch_size, &ch_lvl, &ch_symb, &ch_file);
        int n = ch_size + 1, N = n * n;
        int** box = new int* [N];
        for (int i = 0; i < N; i++)
        {
            box[i] = new int[N];
        }
        if (ch_file == 1)
        {
            system("cls");
            generatebox(n, box, &ch_symb, &ch_lvl);

        }

        else
        {

            if (ch_symb == 1)
            {
                ifstream file("matrix.txt");
                if (!file.is_open())
                {
                    cout << "\tФайл не открыт!" << endl;
                    return -1;
                }

                for (int i = 0; i < N; i++)
                {
                    for (int j = 0; j < N; j++)
                    {

                        file >> box[i][j];
                        if (file.fail() || box[i][j] < 0)
                        {
                            cout << "\tВ файле присутствуют некорректные
данные, исправьте их и запустите программу ещё раз." << endl;
                            return 0;
                        }
                    }
                }

                file.close();
            }

            else
            {

                ifstream filech("matrix.txt");
                if (!filech.is_open())
                {
                    cout << "\tФайл не открыт!" << endl;
                    return -1;
                }
                char** boxchar = new char* [N];
                for (int i = 0; i < N; i++)
                {
                    boxchar[i] = new char[N];
                }
```

```
                for (int i = 0; i < N; i++)
                {
                    for (int j = 0; j < N; j++)
                    {
                        filech >> boxchar[i][j];
                        if (filech.fail() || boxchar[i][j] < 0 ||
(OneSymbTranslat(boxchar[i][j]) < 0 || OneSymbTranslat(boxchar[i][j]) > N))
                        {
                            cout << "\tВ файле присутствуют некорректные
данные, исправьте их и запустите программу ещё раз." << endl;
                            return 0;
                        }
                    }
                }
                filech.close();
                translattoint(N, boxchar, box);
                delete[] boxchar;
            }
        }
        int** firstbox = new int* [N];
        for (int i = 0; i < N; i++)
        {
            firstbox[i] = new int[N];
        }
        for (int i = 0; i < N; i++)
        {
            for (int j = 0; j < N; j++)
            {
                firstbox[i][j] = box[i][j];
            }
        }

        int emptyel = findemptyelement(box, n);
        int** reshenietabl = new int* [4];
        for (int i = 0; i < 4; i++) {
            reshenietabl[i] = new int[emptyel];
        }

        do
        {
            menu(box, firstbox, n, &ch_symb, &ch_temp, reshenietabl, &sovet,
&win);
            system("cls");
            if (win)
            {
                cout << endl;
                printbox(box, firstbox, n, &ch_symb);
                HANDLE handle = GetStdHandle(STD_OUTPUT_HANDLE);
                SetConsoleTextAttribute(handle, 10);
                cout << "\n\tПоздравляем! Вы решили судоку" << endl;
                SetConsoleTextAttribute(handle, 15);
                cout << "\n\tХотите выйти?" << endl;
            }
            else cout << "\n\n\n\tУверены, что хотите выйти?" << endl;
            cout << "\n\t1 - Да" << endl;
            cout << "\n\t2 - Нет, начать заново" << endl;
            if (!win) cout << "\n\t3 - Вернуться обратно" << endl;
            work = SimpleCheckSIn();
            while (work < 1 || work > 3)
            {
                cout << "\n\tВы ввели неверное значение.";
                work = SimpleCheckSIn();
```

```
            }
        } while (work == 3);
        statistica(sovet, win, ch_size + 1, ch_lvl);
        cout << "\n\n\n\tСпасибо за игру! Статистика находится в файле
stat.txt.\n";
        Sleep(3000);
    }
    return 0;
}


void menu(int** box, int** firstbox, int n, int* ch_symb, int* ch_temp, int**
reshenietabl, int* sovet, bool* win)
{
    int choice, res, tip = 0;
    fillsolvet(box, n, reshenietabl);
    do
    {
        system("cls");
        cout << endl;
        printbox(box, firstbox, n, ch_symb);
        cout << "\tВыберите операцию:" << endl;
        cout << "\n\t1 - Поставить символ  2 - Удалить символ  3 - Подсказка
4 - Проверка наличия решения  0 - Выход"
            << endl;
        choice = SimpleCheckSIn();

        while (choice < 0 || choice > 4)
        {
            cout << "\tВы ввели неверное значение. Попробуйте ещё раз" <<
endl;
            choice = SimpleCheckSIn();
        }
        system("cls");

        if (choice == 1)
        {
            cout << endl;
            printbox(box, firstbox, n, ch_symb);
            put(box, firstbox, n, ch_symb, ch_temp, reshenietabl);
        }

        else if (choice == 2)
        {
            cout << endl;
            printbox(box, firstbox, n, ch_symb);
            remove(box, firstbox, n, ch_symb, ch_temp, reshenietabl);
        }

        else if (choice == 3) {
            tip++;
            advice(box, n, reshenietabl);
            Sleep(3000);
        }

        else if (choice == 4)
        {
            cout << endl;
            printbox(box, firstbox, n, ch_symb);
            res = countsolves(box, n, reshenietabl);
            if (res == 1)
            {
```

```
                cout << "\n\tСуществует 1 способ решить судоку на данном
этапе" << endl;
                Sleep(3000);
            }
            else if (res == 2)
            {
                cout << "\n\tСуществует несколько способов решить судоку на
данном этапе" << endl;
                Sleep(3000);
            }
            else if (res == 0)
            {
                cout << "\n\tНе существует способов решить судоку на данном
этапе" << endl;
                Sleep(3000);
            }
        }

    } while (choice != 0 && !iswin(box, n));
    *win = iswin(box, n);
    *sovet = tip;
}


void generatebox(int n, int** box, int* ch_symb, int* ch_lvl)
{

    srand(time(NULL));
    int i, j;
    for (i = 0; i < n * n; i++)
    {
        for (j = 0; j < n * n; j++)
        {
            box[i][j] = (i * n + i / n + j) % (n * n) + 1;
        }
    }
    int random2 = 3 + rand() % 5;

    /*"Перемешивание" изначальной таблицы судоку различными методами,
    не влияющими на удовлетворение этой таблицы правилам игры*/
    while (random2 > 0)
    {
        changesranstrbloka(box, n);
        changesrancolumnsbloka(box, n);
        changesbighorbloka(box, n);
        changesranvertbloka(box, n);
        changesrancolumnsbloka(box, n);
        changesranvertbloka(box, n);
        changesbighorbloka(box, n);
        changesranstrbloka(box, n);
        changesranvertbloka(box, n);
        random2--;
    }
    zero(n, box, ch_symb, ch_lvl);
}


void printbox(int** box, int** firstbox, int n, int* ch_symb)
{
    int N = n * n;
    vector<vector<int>> conflict;
    for (int y = 0; y < N; y++)
```

```
    {
        for (int x = 0; x < N; x++)
        {
            int str = checkequalstr(box, x, y, n, &conflict);
            int colmn = checkequalcol(box, x, y, n, &conflict);
            int block = checkequalbox(box, x, y, n, &conflict);
        }
    }
    HANDLE handle = GetStdHandle(STD_OUTPUT_HANDLE);
    for (int row = 0; row < N; row++)
    {
        cout << "\t";
        for (int col = 0; col < N; col++)
        {
            if (col % n == 0 && col != 0)
                cout << "| ";
            if (firstbox[row][col] != 0) SetConsoleTextAttribute(handle, 7);
            for (int c = 0; c < conflict.size(); c++)
            {
                if (row == conflict[c][0] && col == conflict[c][1])
                {
                    // делаем символ красным, если он конфликтующий и был
поставлен изначально
                    if (firstbox[row][col] != 0)
SetConsoleTextAttribute(handle, 4);
                    // делаем символ светло-красным, если он конфликтующий и
был поставлен пользователем
                    else SetConsoleTextAttribute(handle, 12);
                    break;
                }
            }
            if (*ch_symb == 2 && box[row][col] != 0)
            {
                cout.setf(ios::left);
                cout.width(2);
                cout << static_cast<char>(box[row][col] + 64);
            }
            else {
                cout.setf(ios::left);
                cout.width(2);
                cout << box[row][col];
            }
            cout << " ";
            SetConsoleTextAttribute(handle, 15);
        }
        if ((row + 1) % n == 0 && row != N - 1)
        {
            cout << endl << "\t";
            for (int i = 0; i < N + n - 1; i++)
            {
                cout.setf(ios::left);
                cout.width(2);
                cout << "---";
            }
        }
        cout << endl;
    }
    cout << endl;
}


// Убирает некоторое количество элементов с поля при генерации
```

```
int** zero(int n, int** box, int* ch_symb, int* ch_lvl)
{
    srand(time(NULL));
    int N = n * n * n * n;
    int del;
    /* В зависимости от размера поля и уровня сложности определяем количество
элементов,
       которые стоит удалить*/
    if (*ch_lvl == 1)
    {
        if (n == 4) del = N - (N * 82 / 100);
        else if (n == 5) del = N - (N * 88 / 100);
        else del = N - (N * 48 / 100);
    }
    else if (*ch_lvl == 2)
    {
        if (n == 4) del = N - (N * 78 / 100);
        else if (n == 5) del = N - (N * 86 / 100);
        else del = N - (N * 43 / 100);
    }
    else if (*ch_lvl == 3)
    {
        if (n == 4) del = N - (N * 74 / 100);
        else if (n == 5) del = N - (N * 84 / 100);
        else del = N - (N * 38 / 100);
    }
    else if (*ch_lvl == 4)
    {
        if (n == 4) del = N - (N * 68 / 100);
        else if (n == 5) del = N - (N * 82 / 100);
        else del = N - (N * 33 / 100);
    }
    else
    {
        if (n == 4) del = N - (N * 64 / 100);
        else if (n == 5) del = N - (N * 80 / 100);
        else del = N - (N * 28 / 100);
    }

    /*Рандомно выбираем координаты ячеек, которые хотим обнулить*/
    int x, y;
    while (del > 0)
    {
        x = 0 + rand() % (n * n);
        y = 0 + rand() % (n * n);
        /*Если в выбранной ячейке стоит 0, перевыбираем ячейку до тех пор,
пока не найдём ещё заполненную*/
        while (box[y][x] == 0)
        {
            x = 0 + rand() % (n * n);
            y = 0 + rand() % (n * n);
        }
        box[y][x] = 0;
        del--;// уменьшаем количество элементов, котрые надо удалить
    }

    return box;
}


/*Функция changesranvertbloka меняет местами 2 больших
вертикальных блока таблицы судоку*/
```

```
void changesranvertbloka(int** box, int n)
{
    srand(time(NULL));
    int randres1, randres2;
    int b1, b2;
    int i, j;

    if (n == 2)
    {
        randres1 = 1;
        randres2 = 2;
    }
    else
    {
        do
        {
            randres1 = 1 + rand() % n;
            randres2 = 1 + rand() % n;
        } while (randres1 == randres2);
    }
    b1 = (randres1 - 1) * n;
    b2 = (randres2 - 1) * n;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n * n; j++)
        {
            swap(box[j][b1], box[j][b2]);
        }
        b1++;
        b2++;
    }
}


void changesbighorbloka(int** box, int n)
{
    srand(time(NULL));
    int randres1, randres2;
    int b1, b2;
    int i, j;

    if (n == 2)
    {
        randres1 = 1;
        randres2 = 2;
    }
    else
    {
        do
        {
            randres1 = 1 + rand() % n;
            randres2 = 1 + rand() % n;
        } while (randres1 == randres2);
    }
    b1 = (randres1 - 1) * n;
    b2 = (randres2 - 1) * n;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n * n; j++)
        {
            swap(box[b1][j], box[b2][j]);
        }
```

```
            b1++;
            b2++;
        }
    }


    // Меняет местами радомно выбранные строки внутри блока
    void changesranstrbloka(int** box, int n)
    {
        srand(time(NULL));
        vector <int> range;
        vector <int> result;
        for (int i = 0; i < n * n; i++)
        {
            range.push_back(i);
        }
        int chosestr1 = 0 + rand() % (n * n - 1);
        int x1 = chosestr1 / n;
        int X = x1 * n;
        int x2 = X + n - 1;

        for (int i = 0; i < n * n; i++)
        {
            if ((range[i] >= X && range[i] <= x2) && i != chosestr1)
            {
                result.push_back(range[i]);
            }
        }
        range.clear();
        int chose;
        int chosestr2;
        if (n == 2)
        {
            chosestr2 = result[0];
        }
        else
        {
            chose = 0 + rand() % (result.size() - 1);
            chosestr2 = result[chose];
        }
        for (int i = 0; i < n * n; i++)
        {
            swap(box[chosestr1][i], box[chosestr2][i]);
        }
    }


    // Меняет местами радомно выбранные столбцы внутри блока
    void changesrancolumnsbloka(int** box, int n)
    {
        srand(time(NULL));
        vector <int> range;
        vector <int> result;
        for (int i = 0; i < n * n; i++)
        {
            range.push_back(i);
        }
        int chosecol1 = 0 + rand() % (n * n - 1);
        int x1 = chosecol1 / n;
        int X = x1 * n;
        int x2 = X + n - 1;
```

```
    for (int i = 0; i < n * n; i++)
    {
        if ((range[i] >= X && range[i] <= x2) && i != chosecol1)
        {
            result.push_back(range[i]);
        }
    }
    range.clear();
    int chose;
    int chosecol2;
    if (n == 2)
    {
        chosecol2 = result[0];
    }
    else
    {
        chose = 0 + rand() % (result.size() - 1);
        chosecol2 = result[chose];
    }
    for (int i = 0; i < n * n; i++)
    {
        swap(box[chosecol1][i], box[chosecol2][i]);
    }
}


int entry(int* ch_size, int* ch_lvl, int* ch_symb, int* ch_file)
{
    system("cls");
    cout << "\n\tДобро пожаловать в судоку!\n" << endl;
    cout << "\tВыберите размер одного блока поля для игры:\n" << endl;
    cout << "\t\t1 - 2x2" << endl;
    cout << "\t\t2 - 3x3" << endl;
    cout << "\t\t3 - 4x4" << endl;
    cout << "\t\t4 - 5x5" << endl;
    *ch_size = proverkagetchoice(4);
    while (*ch_size == EOF)
    {
        cout << "\n\n\n";
        cout << "\tНеверный ввод! Выберите размер одного блока поля для
игры:" << endl;
        *ch_size = proverkagetchoice(4);
    }

    system("cls");
    cout << "\n\n\n";
    cout << "\tКакими символами хотите играть на поле?\n" << endl;
    cout << "\t\t1 - Цифрами" << endl;
    cout << "\t\t2 - Буквами" << endl;
    *ch_symb = proverkagetchoice(2);
    while (*ch_symb == EOF)
    {
        cout << "\n\n\n";
        cout << "\tНеверный ввод! Какими символами хотите играть на поле?" <<
endl;
        *ch_symb = proverkagetchoice(2);
    }

    system("cls");
    cout << "\n\n\n";
    cout << "\tХотите, чтобы программа сгенерировала начальное поле, или
введёте поле сами из файла?\n" << endl;
```

```
    cout << "\t\t1 - Программа" << endl;
    cout << "\t\t2 - Самостоятельный ввод" << endl;
    *ch_file = proverkagetchoice(2);
    while (*ch_file == EOF)
    {
        cout << "\n\n\n";
        cout << "\t\nНеверный ввод! Введите правильные данные:" << endl;
        *ch_file = proverkagetchoice(2);
    }
    if (*ch_file == 2)
        return 0;

    system("cls");
    cout << "\n\n\n";
    cout << "\tВыберите уровень сложности:\n" << endl;
    cout << "\t\t1 - Легкий" << endl;
    cout << "\t\t2 - Средний" << endl;
    cout << "\t\t3 - Сложный" << endl;
    cout << "\t\t4 - Экспертный" << endl;
    cout << "\t\t5 - Безумный" << endl;
    *ch_lvl = proverkagetchoice(5);
    while (*ch_lvl == EOF)
    {
        cout << "\n\n\n";
        cout << "\t\nНеверный ввод! Выберите уровень сложности:" << endl;
        *ch_lvl = proverkagetchoice(5);
    }
    return 0;
}


void put(int** box, int** firstbox, int n, int* ch_symb, int* ch_temp, int**
reshenietabl)
{
    int row, col, value, N = n * n;
    cout << "\n\tВведите строку ячейки, на которую хотите поставить элемент:
";
    row = CheckCin(ch_temp, N);
    row--;

    cout << "\n\tВведите столбец ячейки, на которую хотите поставить элемент:
";
    col = CheckCin(ch_temp, N);
    col--;

    cout << "\n\tВведите значение элемента, который хотите поставить: ";
    value = CheckCin(ch_symb, N);

    while (firstbox[row][col] != 0)
    {
        cout << "\n\tНа выбранную ячейку нельзя поставить значение. Введите
другую.";
        Sleep(3000);
        system("cls");
        cout << endl;
        printbox(box, firstbox, n, ch_symb);
        cout << "\n\tВведите строку ячейки, на которую хотите поставить
элемент: ";
        row = CheckCin(ch_temp, N);
        row--;

        cout << "\n\tВведите столбец ячейки, на которую хотите поставить
```

```
элемент: ";
        col = CheckCin(ch_temp, N);
        col--;

        cout << "\n\tВведите значение элемента, который хотите поставить: ";
        value = CheckCin(ch_symb, N);
    }

    box[row][col] = value;
}


void remove(int** box, int** firstbox, int n, int* ch_symb, int* ch_temp,
int** reshenietabl)
{
    int row, col, temp, N = n * n, c = 0;
    int i, j;
    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            if (box[i][j] == firstbox[i][j]) c++;
        }
    }
    if (c == N * N) {
        cout << "\n\tУдаление элементов в исходной матрице невозможно" <<
endl;
        Sleep(3000);
        return;
    }
    cout << "\n\tВведите строку элемента, который хотите удалить: ";
    row = CheckCin(ch_temp, N);
    row--;

    cout << "\n\tВведите столбец элемента, который хотите удалить: ";
    col = CheckCin(ch_temp, N);
    col--;
    temp = box[row][col];
    box[row][col] = 0;
    while (firstbox[row][col] != 0)
    {
        box[row][col] = temp;
        cout << "\n\tВведенный элемент нельзя удалить. Введите другой.";
        Sleep(3000);
        system("cls");
        cout << endl;
        printbox(box, firstbox, n, ch_symb);
        cout << "\n\tВведите строку элемента, который хотите удалить: ";
        row = CheckCin(ch_temp, N);
        row--;

        cout << "\n\tВведите столбец элемента, который хотите удалить: ";
        col = CheckCin(ch_temp, N);
        col--;

        temp = box[row][col];
        box[row][col] = 0;
    }
}


void translattoint(int N, char** boxchar, int** box)
{
    for (int i = 0; i < N; i++)
```

```
    {
        for (int j = 0; j < N; j++)
        {
            if (boxchar[i][j] == '0') box[i][j] = 0;
            else box[i][j] = static_cast<int>(boxchar[i][j]) - 64;
        }
    }
}


int OneSymbTranslat(char symb)
{
    int sym;
    if (symb == '0') sym = 0;
    else sym = static_cast<int>(symb) - 64;
    return sym;
}


int SimpleCheckSIn()
{
    while (true)
    {
        int n;
        cin >> n;
        if (cin.fail())
        {
            cin.clear();
            cin.ignore(32767, '\n');
        }
        else
        {
            cin.ignore(32767, '\n');
            return n;
        }
    }
}


int CheckCin(int* ch_symb, int N)
{
    if (*ch_symb == 1)
    {
        while (true)
        {
            int n;
            cin >> n;
            if (cin.fail() || n < 1 || n > N)
            {
                cin.clear();
                cin.ignore(32767, '\n');
                cout << "\tНеверно введены данные. Попробуйте ещё раз." <<
endl;
            }
            else
            {
                cin.ignore(32767, '\n');
                return n;
            }
        }
    }
    else
```

```
    {
        while (true)
        {
            int sym;
            char symb;
            cin >> symb;
            if (symb == '0')
            {
                sym = 0;
            }
            else sym = static_cast<int>(symb) - 64;

            if (cin.fail() || sym < 1 || sym > N)
            {
                cin.clear();
                cin.ignore(32767, '\n');
                cout << "\tНеверно введены данные. Попробуйте ещё раз." <<
endl;
            }
            else
            {
                cin.ignore(32767, '\n');
                return sym;
            }
        }
    }
}


// Проверяет на корректность введенное значение и сравнивает его с
максимальным возможным
int proverkagetchoice(int max)
{
    string temp;
    getline(cin, temp);
    unsigned len = temp.length();
    if (len == 0) return EOF;
    const char* num = "123456789";
    for (int i = 0; i < len; i++) {
        if (!strchr(num, temp[i]) || temp[i] == ' ') return EOF;
    }
    if (stoi(temp) > max) return EOF;
    else return stoi(temp);
}


/* Функция advice подсказывает пользователю следующих ход
и заполняет одну клетку на поле, приближая пользователя к победе */
void advice(int** box, int n, int** reshenietabl) {
    int res;
    res = countsolves(box, n, reshenietabl);  //В res сохраняется количество
решений на данном этапе игры
    /*Если существует одно или несколько решений, программа подскажет
пользователю
    следующий ход из массива, в котором хранится решение(я) судоку на данном
этапе*/
    if ((res == 1) || (res == 2)) {
        box[reshenietabl[0][0]][reshenietabl[1][0]] = reshenietabl[2][0];
        cout << "\n\tИщите подсказку в ячейке с координатами: " <<
reshenietabl[0][0] + 1 << " " << reshenietabl[1][0] + 1 << endl;
    }
    else if (res == 0) {
```

```
        cout << "\n\tПри данной комбинации на поле дальнейшее решение
невозможно" << endl;
    }
}


int checkequalstr(int** box, int x, int y, int n, vector<vector<int>>*
conflict)
{
    vector<int> chelp;
    if (box[y][x] != 0)
    {
        for (int j = 0; j < n * n; j++)
        {
            if (box[y][x] == box[y][j] && j != x)
            {
                chelp.push_back(y);
                chelp.push_back(x);
                conflict->push_back(chelp);
                chelp.clear();
                chelp.push_back(y);
                chelp.push_back(j);
                conflict->push_back(chelp);
                chelp.clear();
                return 1;
            }
        }
    }
    return 0;
}


// Проверка на совпадение цифр в столбце
int checkequalcol(int** box, int x, int y, int n, vector<vector<int>>*
conflict)
{
    vector<int> chelp;
    if (box[y][x] != 0)
    {
        for (int i = 0; i < n * n; i++)
        {
            if (box[y][x] == box[i][x] && i != y)
            {
                chelp.push_back(y);
                chelp.push_back(x);
                conflict->push_back(chelp);
                chelp.clear();
                chelp.push_back(i);
                chelp.push_back(x);
                conflict->push_back(chelp);
                chelp.clear();
                return 1;
            }
        }
    }
    return 0;
}


// Проверка на совпадение элементов внутри одного блока n*n
int checkequalbox(int** box, int x, int y, int n, vector<vector<int>>*
conflict)
```

```
{
    vector<int> chelp;
    int k = x / n;
    int p = y / n;
    int X = k * n;
    int Y = p * n;
    if (box[y][x] != 0)
    {
        for (int i = Y; i < n + Y; i++)
        {
            for (int j = X; j < n + X; j++)
            {
                if (box[i][j] == box[y][x] && (i != y || j != x))
                {
                    chelp.push_back(i);
                    chelp.push_back(j);
                    conflict->push_back(chelp);
                    chelp.clear();
                    chelp.push_back(y);
                    chelp.push_back(x);
                    conflict->push_back(chelp);
                    chelp.clear();
                    return 1;
                }
            }
        }
    }
    return 0;
}


// Объединяет все функции проверки совпадения, проверяет корректность
постановки элементов на поле
bool checkequal(int** box, int x, int y, int n, int k)
{
    vector<vector<int>> conflict;
    box[y][x] = k;
    int str = checkequalstr(box, x, y, n, &conflict);
    int col = checkequalcol(box, x, y, n, &conflict);
    int block = checkequalbox(box, x, y, n, &conflict);
    box[y][x] = 0;
    return str + col + block != 0;
}


void fillsolvet(int** box, int n, int** reshenietabl)
{
    int i, j, k = 0;
    for (i = 0; i < n * n; i++)
    {
        for (j = 0; j < n * n; j++)
        {
            if (box[i][j] == 0)
            {
                reshenietabl[0][k] = i;
                reshenietabl[1][k] = j;
                k++;
            }
        }
    }
}
```

```
/*Рекурсивная функция resheniesudoku19 решает судоку, пытаясь подставить
в незаполненные ячейки значения от 1 до n*n, если это возможно*/
bool resheniesudoku19(int** box, int n)
{
    int i, j;
    if (!isemptyelment(box, n, i, j)) return true;
    for (int k = 1; k <= n * n; k++)
    {

        if (!checkequal(box, j, i, n, k))
        {
            box[i][j] = k;
            if (resheniesudoku19(box, n)) return true;
            box[i][j] = 0;
        }
    }
    return false;
}


/*Рекурсивная функция resheniesudoku91 решает судоку, пытаясь подставить
в незаполненные ячейки значения от n*n до 1, если это возможно*/
bool resheniesudoku91(int** box, int n)
{
    int i, j;
    if (!isemptyelment(box, n, i, j)) return true;
    for (int k = n * n; k >= 1; k--)
    {

        if (!checkequal(box, j, i, n, k)) {
            box[i][j] = k;
            if (resheniesudoku91(box, n)) return true;
            box[i][j] = 0;
        }
    }
    return false;
}


bool isemptyelment(int** box, int n, int& i, int& j)
{
    for (i = 0; i < n * n; i++)
        for (j = 0; j < n * n; j++)
            if (box[i][j] == 0) return true;
    return false;
}


int findemptyelement(int** box, int n)
{
    int i, j;
    int empt = 0;
    for (i = 0; i < n * n; i++)
    {
        for (j = 0; j < n * n; j++)
        {
            if (box[i][j] == 0)
            {
                empt++;
            }
        }
```

```
    }
    return empt;
}


int countsolves(int** box, int n, int** reshenietabl)
{
    int i, j, k;
    int cntsame = 0;
    int** firstbox = new int* [n * n];
    for (i = 0; i < n * n; i++)
    {
        firstbox[i] = new int[n * n];
    }
    for (i = 0; i < n * n; i++)
    {
        for (j = 0; j < n * n; j++)
        {
            firstbox[i][j] = box[i][j];
        }
    }

    int emptyel = findemptyelement(box, n);
    fillsolvet(box, n, reshenietabl);
    for (i = 0; i < emptyel; i++) {
        reshenietabl[2][i] = 0;
        reshenietabl[3][i] = 0;
    }


    if (resheniesudoku19(box, n))
    {
        for (i = 0; i < n * n; i++)
        {
            for (j = 0; j < n * n; j++)
            {
                for (k = 0; k < emptyel; k++)
                {
                    if ((reshenietabl[0][k] == i) && (reshenietabl[1][k] ==
j))
                    {
                        reshenietabl[2][k] = box[i][j];
                    }
                }
            }
        }
    }
    else { return 0; }

    for (i = 0; i < n * n; i++)
    {
        for (j = 0; j < n * n; j++)
        {
            box[i][j] = firstbox[i][j];
        }
    }

    if (resheniesudoku91(box, n))
    {
        for (i = 0; i < n * n; i++)
        {
            for (j = 0; j < n * n; j++)
```

```
            {
                for (k = 0; k < emptyel; k++)
                {
                    if ((reshenietabl[0][k] == i) && (reshenietabl[1][k] ==
j))
                    {
                        reshenietabl[3][k] = box[i][j];
                    }
                }
            }
        }
    }
    else { return 0; }

    for (i = 0; i < n * n; i++)
    {
        for (j = 0; j < n * n; j++)
        {
            box[i][j] = firstbox[i][j];
        }
    }

    for (j = 0; j < emptyel; j++)
    {
        if ((reshenietabl[2][j] == reshenietabl[3][j]) && (reshenietabl[2][j]
!= 0))  cntsame++;
    }
    if (cntsame == emptyel) return 1;
    else return 2;
}


// Возвращает true если пользователь заполнил поле правильно, иначе - false
bool iswin(int** box, int n) {
    vector<vector<int>> conflict;
    for (int y = 0; y < n * n; y++) {
        for (int x = 0; x < n * n; x++) {
            // проверяем конфликты в строке с элементом
            int str = checkequalstr(box, x, y, n, &conflict);
            // проверяем конфликты в столбце с элементом
            int col = checkequalcol(box, x, y, n, &conflict);
            // проверяем конфликты в блоке с элементом
            int block = checkequalbox(box, x, y, n, &conflict);
            // если есть конфликт, возвращаем false
            if (str + col + block != 0) return false;
        }
    }
    return findemptyelement(box, n) == 0;
}


void statistica(int sovet, bool finality, int n, int lvl)
{
    ofstream file("stat.txt", ios_base::out | ios_base::app);
    if (!file.is_open())
    {
        cout << "Файл не открыт!" << endl;
        Sleep(2000);
    }

    char buffer[80];
    time_t seconds = time(NULL);
```

```
    tm timeinfo;
    localtime_s(&timeinfo, &seconds);
    const char* format = "%B %d, %Y %H:%M:%S";
    strftime(buffer, 80, format, &timeinfo);

    string fin;
    if (finality) fin = "Да"; else fin = "Нет";

    file << "\n-----------------------------------------------------\n";
    file << "\tДата партии: " << buffer << endl;
    file << "\tРазмер поля: " << n << "x" << n << endl;
    file << "\tСложность: " << lvl << endl;
    file << "\tКоличество использованных подсказок: " << sovet << endl;
    file << "\tБыла ли партия завершена: " << fin;
    file << "\n-----------------------------------------------------";

    file.close();
}
```