Sudoku solver 3000

Поиском в глубину с помощью рекурсивной функции (наверное)

by Скунов Тимур

Цели проекта:

1. Разработать и написать **универсальный** алгоритм решения судоку
2. Оптимизировать его работу
3. Ещё оптимизировать
4. И ещё…

Основной принцип:

```
bool solve_field()
{
    int x, y;
    if (!find_empty_cell(x, y))
        return 1;
    for (int num = 1; num <= 9; num++)
    {
        if (check_cell(x, y, num))
        {
            field[x][y] = num;
            if (solve_field()) return 1;
            field[x][y] = 0;
        }
    }
    return 0;
}
```

Найдя пустую ячейку в матрице, функция заполняет её первой цифрой, не противоречащей базовым правилам, и переходит к следующей ячейке. Если её невозможно заполнить с учётом предыдущих предположений, предыдущая ячейка заполняется следующей возможной цифрой, а при невозможности этого функция переходит к пред-предыдущей и так далее. Когда все ячейки заполнены правильно или все потенциальные варианты решений не подходят, функция возвращает 1 или 0 соответственно.

Поиск пустых ячеек:

```
bool find_empty_cell(int& x, int& y) {
    for (x = 0; x < 9; x++)
    {
        for (y = 0; y < 9; y++)
        {
            if (field[x][y] == 0) return 1;
        }
    }
    return 0;
}
```

В функцию передаются адреса переменных-координат. При нахождении пустой ячейки, функция сохраняет её координаты и возвращает 1, а если все ячейки заполнены (поле решено), возвращает 0.

Выводы:

- Искать готовые решения и использовать их доработанные идеи - полезный метод
- Разбиение кода на функции сильно помогает адаптировать новые “детали”