4 EM LINHA

Estruturas de Dados e Algoritmos
Departamento de Engenharia Eletrotecnica e de Computadores

Inês Jorge da Silva e Ferreira 2019234524
André Guilherme dos Santos Neto 2019237495

cout << "Objetivos" << endl;

- Ter um código, simples, inteligente e encapsulado.
- Manter uma interface de Jogo limpa e objetiva.
- Completar o Nivel III + extra.

cout << "Principais Dificuldades" << endl;

- Manter a simplicidade e a independência dos atributos e métodos.
- Solucionar uma estrutura de dados que satisfizesse as necessidades do projeto.

cout << "Estrutura de Dados" << endl;

```
class CGame{

  private:
    CBoard *board;
    string player_name;
    char current_player;
    int difficulty;
```

Chegámos à conclusão que os membros relacionados com o tabuleiro podiam ter a sua própria classe.

Todos os métodos da classe CBoard são independentes à classe CGame à exceção da jogada (recebe quem joga e onde).

```
class CBoard{
    friend class CGame;

  private:
    constexpr static const int MAX_C = 7;
    constexpr static const int MAX_R = 6;
    constexpr static const int SIZE = MAX_R*MAX_C + 1;
    char ***boards;
    int turn;
```

O maior desafio foi sem dúvida o extra (quase que ficou operacional) mas a necessidade de criar uma tabela de tabuleiros (char ***boards) resolveu-nos o problema do undo/re-faz jogada (que agora pode ser chamado sem limites!).

Return (0);