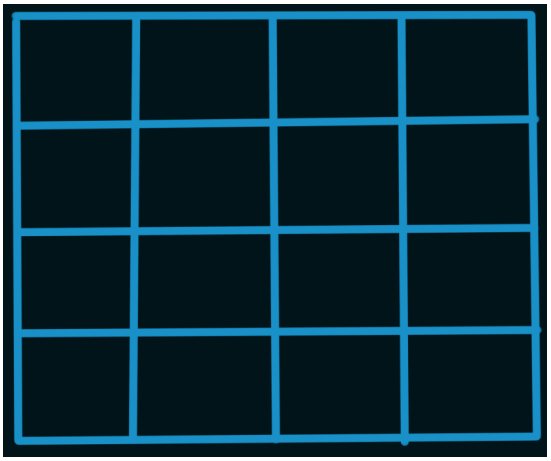
Exercício 4: Implementação da variação do Jogo da Velha

TicTacToe Mineiro

Alunos :

Daniel Alves Fonseca Neto dafn
Filipe Abner Soares Melo fasm

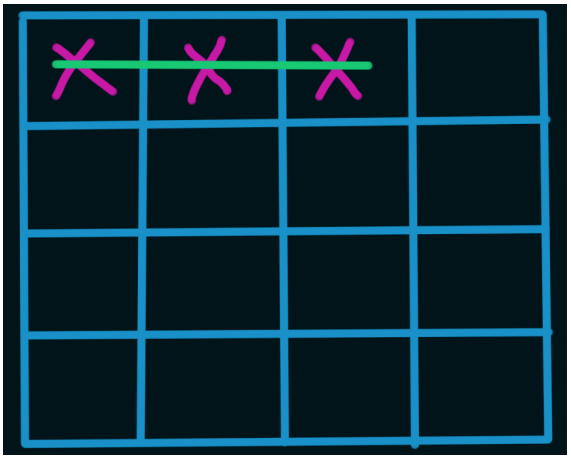
Esse é um jogo da velha 4x4

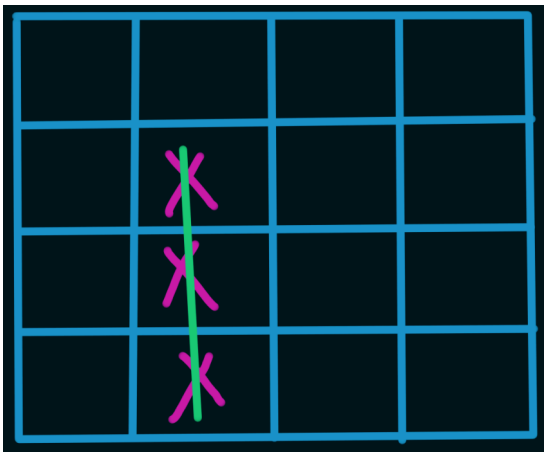


As regras são quase as mesmas do jogo da velha tradicional,

1º joga um símbolo e depois o outro joga, X e O, se o tabuleiro for todo preenchido e ninguém ganhar, então deu velha

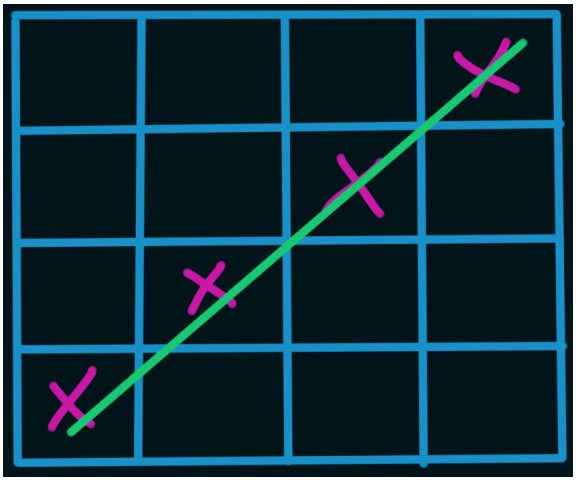
com a diferença de que é preciso marcar apenas uma sequência de 3 símbolos iguais na horizontal ou na vertical para ganhar…



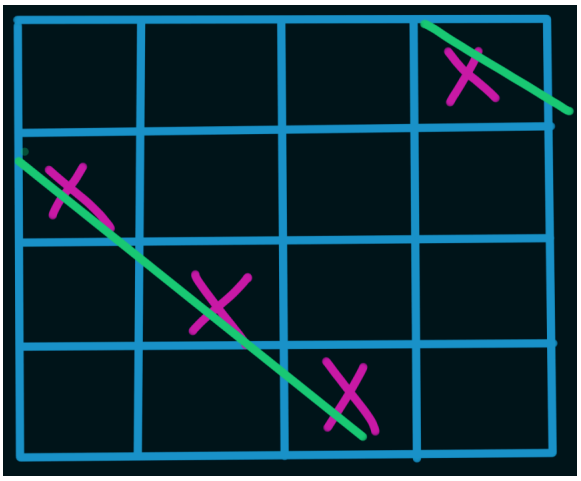


Mas para ganhar na diagonal a regra é que é preciso marcar uma sequência de 4 símbolos iguais, com 1 detalhe, marcar símbolos iguais em uma diagonal paralela para completar o 4 símbolos iguais também pode

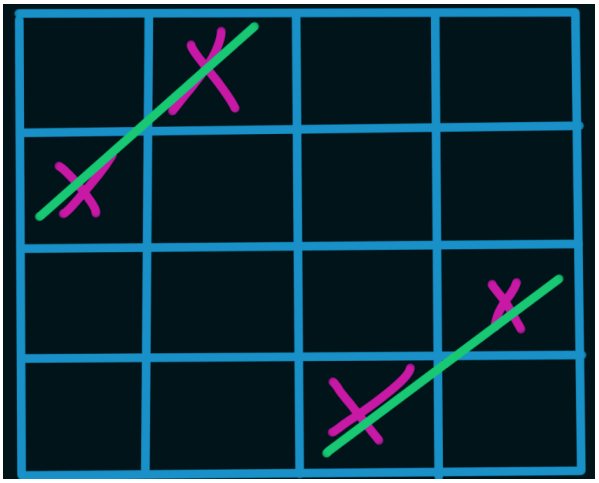
Caso 1 - 4 símbolos iguais na diagonal completa do tabuleiro



Caso 2 - 3 símbolos iguais em uma diagonal com mais 1 na diagonal paralela



Caso 3 - 2 símbolos iguais em uma diagonal e mais 2 símbolos na diagonal paralela



Sobre a construção do trabalho:

Existem 2 arquivos, **main.cpp** e **main2.cpp**

a main.cpp era um teste com um jogo da velha tradicional,
a main2.cpp é a implementação para a variação do jogo da velha 4x4 tictactoe mineiro.

para compilar é preciso ter o compilador de c++ instalado e usar o comando :

g++ main2.cpp -o JogoVelha

depois executar JogoVelha

Sobre o código :

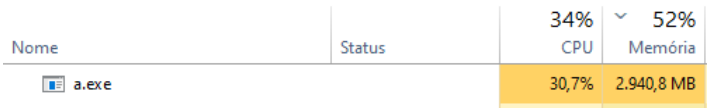
função main

```cpp
int main()
{
    std::vector<std::vector<char>> estadoInicial = {
        {' ', ' ', ' ', ' '},
        {' ', 'o', 'x', ' '},
        {' ', 'x', 'o', ' '},
        {' ', ' ', ' ', ' '}
    };

    No *raiz = new No();

    int posiInicial = 0;
    char simboloInicial = 'x';


    IAvsIA(raiz, estadoInicial, posiInicial, simboloInicial);


    deletaNo(raiz);

    return 0;
}
```

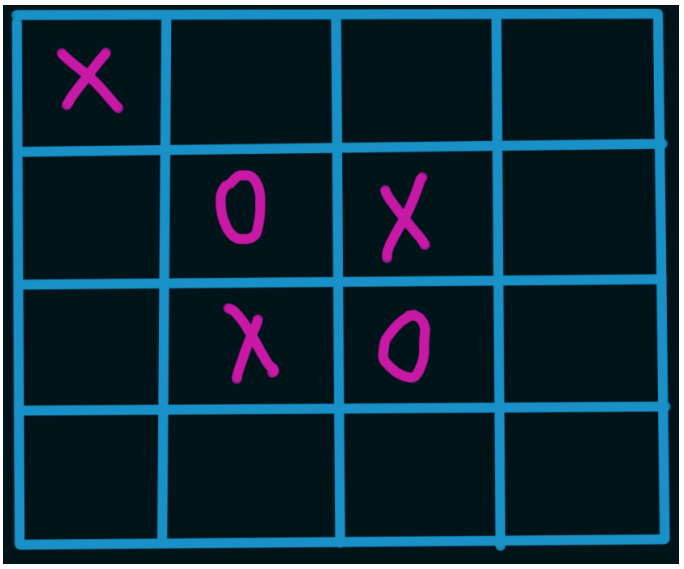
é preciso criar um estado inicial,
note que por ser um tabuleiro 4x4 é preciso já ter colocado algumas jogadas iniciais para não estourar a memória e ser mais rapido a geração da árvore de decisão !!!

4x4 = 16 → 16! = +- 200 gigas de memória

se ja preenchermos 4 espaços então as possibilidades se limitam a menos que 12! o que da +- no maximo 3 - 4 gigas de memória.



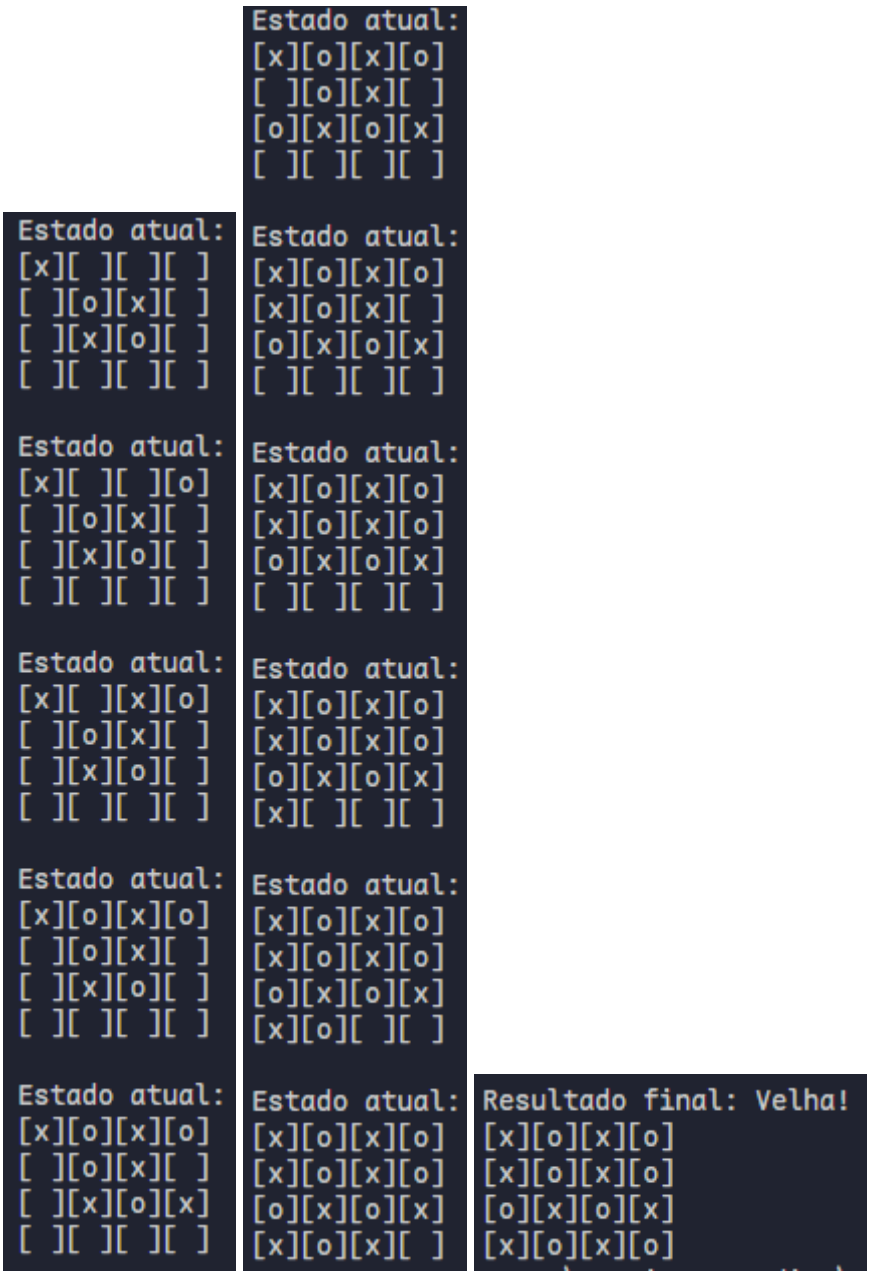
Definimos a posição inicial do primeiro simbolo a ser jogado após o estado inicial do tabuleiro, nesse caso é X e na posição 0.



Depois a função IAvsIA gera a arvore de possibilidade de jogadas e pontua as ramificações contando em quantas jogadas levam a vencedor como X ou O , e quantas deu velha.

```
contador x raiz : 7768728
contador o raiz : 4658580
contador velha raiz : 5702400
```

Aqui esta o passo a passo das jogadas



esse jogo tendeu a isso por conta do balanceamento dos multiplicadores de X,O e velha que estavam assim :

```
int multiplicador_x = 10;
int multiplicador_o = 10;
int multiplicador_velha = 10;
```

para essa função de decisão :

para IA que joga pro X nesse exemplo:

```
valor = ((filho->pontuacao_X * multiplicador_x) + (filho->pontuacao_V * multiplicador_velha)) - (filho->pontuacao_O * multiplicador_o);
```

é possivel ver que nas jogadas finais as decisões foram diferente para o novo balanceamento

```
int multiplicador_x = 10;
int multiplicador_o = 10;
int multiplicador_velha = 7;
```

```
Estado atual:
[x][o][x][o]
[ ][o][x][ ]
[o][x][o][x]
[ ][ ][ ][ ]

Estado atual:
[x][o][x][o]
[ ][o][x][ ]
[o][x][o][x]
[x][ ][ ][ ]

Estado atual:
[x][o][x][o]
[ ][o][x][ ]
[o][x][o][x]
[x][o][ ][ ]

Estado atual:
[x][o][x][o]
[ ][o][x][ ]
[o][x][o][x]
[x][o][ ][x]

Estado atual:
[x][o][x][o]
[ ][o][x][o]
[o][x][o][x]
[x][o][ ][x]

Estado atual:
[x][o][x][o]
[x][o][x][o]
[o][x][o][x]
[x][o][ ][x]

Resultado final: Velha!
[x][o][x][o]
[x][o][x][o]
[o][x][o][x]
[x][o][o][x]
```

ainda deu velha

Com os novos multiplicadores para velha = 0

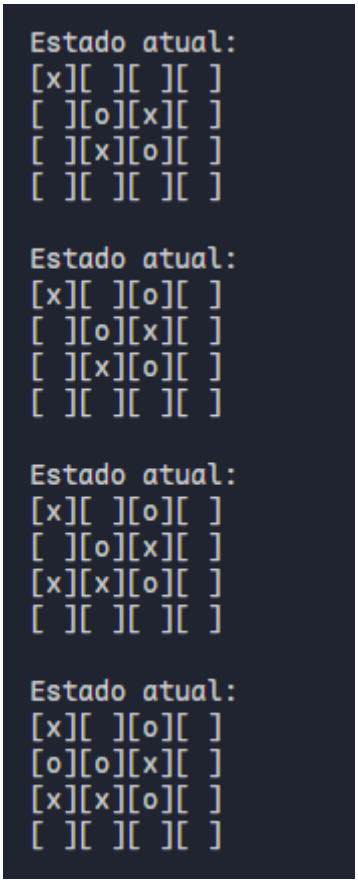
```
int multiplicador_x = 10;
int multiplicador_o = 10;
int multiplicador_velha = 0;
```

```
Estado atual:
[x][ ][ ][ ]
[ ][o][x][ ]
[ ][x][o][ ]
[ ][ ][ ][ ]

Estado atual:
[x][ ][o][ ]
[ ][o][x][ ]
[ ][x][o][ ]
[ ][ ][ ][ ]

Estado atual:
[x][ ][o][ ]
[ ][o][x][ ]
[x][x][o][ ]
[ ][ ][ ][ ]

Estado atual:
[x][ ][o][o]
[ ][o][x][ ]
[x][x][o][ ]
[ ][ ][ ][ ]

Resultado final: Vencedor: x
[x][ ][o][o]
[x][o][x][ ]
[x][x][o][ ]
[ ][ ][ ][ ]
```

o ganhador foi o primeiro a jogar, nesse caso o X

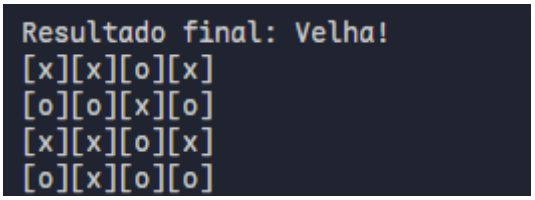
Achei o multiplicador pra velha = 3 o mais balanceado para este estado inicial de jogo

```
int multiplicador_x = 10;
int multiplicador_o = 10;
int multiplicador_velha = 3;
```



tirei esse print pois é possível ver uma decisão da IA bolinha impedir o X de ganhar na jogada 4

e no final o jogo foi velha



https://github.com/DanielAlvesFonsecaNeto/TicTacToeMineiro