



# JOGO DA VELHA

CONCORRÊNCIA EM THREADS

ALUNOS: VICTOR HUGO COSTA, GIOVANA OLIVEIRA E RYAN KAYKY



# SOBRE O PROJETO

O projeto a ser desenvolvido, consiste na criação do jogo da velha. Neste caso, o jogo será feito por dois usuários artificiais (ou seja duas threads) no mesmo jogo da velha. Cada usuário deve operar o jogo no mesmo tabuleiro do jogo da velha 3 por 3. Assim, o recurso compartilhado será tabuleiro do jogo da velha, logo os jogadores devem esperar a sua vez de jogar e obdecer a regras do jogo.



```
● ● ●  
1  /*  
2   [0][0] [1][0] [2][0]  
3   [0][1] [1][1] [2][1]  
4   [0][2] [1][2] [2][2]  
5 */  
6  char tabuleiro[3][3] = {{' ', ' ', ' '},  
7   {' ', ' ', ' '},  
8   {' ', ' ', ' '} };  
9  
10 char ganhou = 'f';  
11 char verifica(int id)  
12 {  
13     simb = id == 1 ? 'x' : 'o';  
14  
15     // Verifica linhas e colunas  
16     for (int i = 0; i < 3; i++)  
17     {  
18         if (tabuleiro[i][0] == simb && tabuleiro[i][1] == simb && tabuleiro[i][2] == simb ||  
19             tabuleiro[0][i] == simb && tabuleiro[1][i] == simb && tabuleiro[2][i] == simb)  
20         {  
21             return 'v';  
22         }  
23     }  
24  
25     // Verifica diagonais  
26     if (tabuleiro[0][0] == simb && tabuleiro[1][1] == simb && tabuleiro[2][2] == simb ||  
27         tabuleiro[0][2] == simb && tabuleiro[1][1] == simb && tabuleiro[2][0] == simb)  
28     {  
29         return 'v';  
30     }  
31     return 'f';  
32 }  
33 }
```

# TABULEIRO

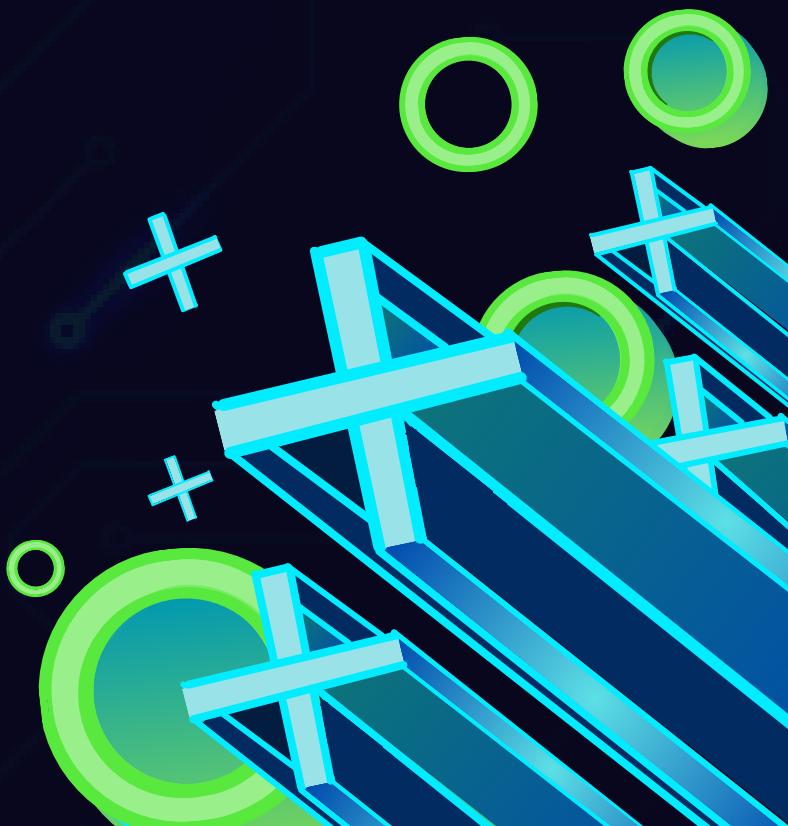
- Matriz 3x3
- Forma de inserção
- Verificação de vitória



```
1 // Regiao critica
2 char ganhou = 'f';
3 int should_sleep = 1; // variável de condição
4 int quantidade_jogada = 0;
5 int max_jogada = 9;
6 char tabuleiro[3][3] = {{' ', ' ', ' '}, 
7                           {' ', ' ', ' '}, 
8                           {' ', ' ', ' '} };
```

# REGIAO CRITICA

- Tabuleiro
- Should\_sleep
- “Verificação de vitória”



# JOGADA THREAD

```

1 void *jogada_tread(void *arg)
2 {
3     int *id = (int *)arg;
4     while (1)
5     {
6         // Sistema para acordar a thread
7         pthread_mutex_lock(&mutex);
8         while (*id == 1 ? !should_sleep : should_sleep)
9         {
10             pthread_cond_wait(&cond, &mutex);
11         }
12         pthread_mutex_unlock(&mutex);
13         // acaba aqui
14
15         if (quantidade_jogada != max_jogada)
16         {
17             while (1)
18             {
19                 int r_l = 0 + rand() % (2 - 0 + 1);
20                 int r_c = 0 + rand() % (2 - 0 + 1);
21                 if (tabuleiro[r_l][r_c] == ' ')
22                 {
23                     tabuleiro[r_l][r_c] = *id == 1 ? 'x' : 'o';
24                     break;
25                 }
26             }
27             for (int i = 0; i < 3; i++)
28             {
29                 for (int j = 0; j < 3; j++)
30                 {
31                     if (j != 2)
32                     {
33                         printf("| %c ", tabuleiro[i][j]);
34                     }
35                     else
36                     {
37                         printf("| %c |", tabuleiro[i][j]);
38                     }
39                 }
39                 printf("\n");
40             }
41         }
42     }
}

```

```

1 printf("Jogador %d Joga\n", *id);
2         quantidade_jogada++;
3
4 ganhou = verifica(*id);
5 if (ganhou == 'v')
6 {
7     printf("\nJogador %d ganhou\n", *id);
8     quantidade_jogada = max_jogada; // Encerra o jogo
9 }
10 else if (ganhou == 'f' && quantidade_jogada >= max_jogada)
11 {
12     printf("Empatou\n");
13     quantidade_jogada = max_jogada; // Encerra o jogo
14 }
15
16 sleep(1);
17 pthread_mutex_lock(&mutex);
18 should_sleep = !should_sleep; // thread 1 ou 2 dorme
19 pthread_mutex_unlock(&mutex);
20 pthread_cond_signal(&cond);
21
22 }
23 }
24

```

- Acorda Thread
- Jogar
- Random
- Pós verificação
- Dorme Thread

# IMPLEMENTAÇÃO