INSTITUTO FEDERAL MINAS GERAIS

CAMPUS SÃO JOÃO EVANGELISTA

BACHARELADO SISTEMAS DE INFORMAÇÃO

INTRODUÇÃO A PROGRAMAÇÃO

GABRIEL KÁICON BATISTA HILÁRIO – TURMA SI221

TRABALHO PRÁTICO: JOGO DA VELHA

SÃO JOÃO EVANGELISTA – MG
JULHO – 2022

INSTITUTO FEDERAL MINAS GERAIS

CAMPUS SÃO JOÃO EVANGELISTA

BACHARELADO SISTEMAS DE INFORMAÇÃO

INTRODUÇÃO A PROGRAMAÇÃO

GABRIEL KÁICON BATISTA HILÁRIO – TURMA SI 221

TRABALHO PRÁTICO: JOGO DA VELHA

Trabalho apresentado por exigência da disciplina de Introdução a Programação, ministrada pelo Professor Eduardo Augusto Costa Trindade, do 1° período do Curso de Graduação de Sistemas de Informação, do Instituto Federal Minas Gerais – Campus São João Evangelista.

SÃO JOÃO EVANGELISTA – MG
JULHO – 2022

SUMÁRIO

1. INTRODUÇÃO

Este documento tem como objetivo documentar as etapas de desenvolvimento de um trabalho prático da disciplina de Introdução a Programação, do curso de Bacharelado em Sistemas de Informação, do Instituto Federal de Minas Gerais, *campus* de São João Evangelista.

O trabalho foi desenvolvido com objetivo de desenvolver a lógica, e ainda sim poder se divertir com isso. Foram utilizadas as linguagens de C e C++, já que são linguagens de alto nível que se comunicam de forma direta com o hardware, podendo ser encontrada com facilidade em processadores, microcontroladores e drivers, o que aumenta bastante a eficiência do software, devida a rápida resposta.

A Linguagem C foi criada e implementada no início dos anos 70 por *Dennis Ritchie* em um microcomputador chamado DEC PDP-11, que usava o Sistema Operacional UNIX, com intuito de trabalhar com o acesso e controle dos espaços da memória das máquinas. Sendo hoje uma ótima opção para criar grandes jogos do mercado atual, editores de imagem e vídeo, robôs, sistemas de automação, e também muitos sistemas operacionais, onde estes são programados totalmente ou parcialmente em C, como por exemplo o Windows, o Linux e o Mac OS, isso porque podem precisar de comunicação rápida com o hardware. Ela é descrita como a linguagem mãe, porque funciona de forma binária, suportando qualquer arquitetura, e devido ao fato de diversas outras linguagens utilizadas hoje surgiram utilizando C como base. Exemplos disso: o PHP, Java, C# e o **C++**.

A linguagem C possui uma limitação, ela não permite que o arquivo ultrapassa 25.000 a 100.00 linhas de código. Para solucionar esse problema, em 1980 um estudioso chamado *Bjarne Stroustrup*, utilizou a linguagem C, para criar uma "nova" linguagem que se chamava inicialmente "*C with classes*", como o próprio nome diz, C com classes, pois usava dos conceitos de POO (Programação Orientada a Objetos) e 3 anos depois o nome foi mudado para linguagem C++.

Como foi utilizado os conhecimentos de ambas as linguagens, mesmo que uma seja fruto da outra, vale a pena deixar explícita a diferença entre ambas, sendo algumas delas, a extensão, a sintaxe, as bibliotecas, seus objetivos e seus paradigmas, sendo a última a principal diferença. Enquanto o paradigma do C é procedural, isto é, todo o código em 1 arquivo, o C++ é o multi-paradigma, isto é, o código pode ser dividido em mais arquivos que se comunicam entre si, e ainda sim, utilizar os mesmos recursos que o C, e a orientação a objetos do próprio C++.

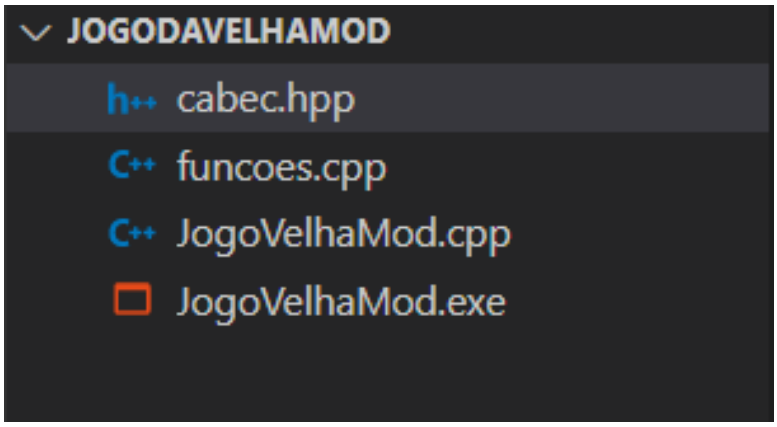
Foi dada a tarefa de criar um Jogo da Velha em C/C++, a princípio a lógica, do programa foi fácil de ser pensada, na hora de se executar, nem tanto. Foram utilizados conceitos a respeito de laços de repetição, sendo eles o *do-while, for* e *while*, conceitos a respeito de funções, conceitos de módulo, pois o programa teve uma divisão por modularização, ou seja, dividir o mesmo em arquivos, como por exemplo, um arquivo .cpp com as funções, um arquivo .hpp que seria o cabeçalho para chamada das funções, e um arquivo .cpp com a função *Main()*. Foram utilizados conceitos básicos de programação, como condicionais, sendo elas o *if* e o *else*, inclusão de bibliotecas e a declaração variáveis. Foi utilizado o VSCode como editor de código, e o MinGW instalado na máquina para compilação de códigos em C/C++.

2. DESENVOLVIMENTO

Será apresentado como o trabalho foi feito, com uma explicação detalhada, logo abaixo as partes comentadas do programa.

2.1. Arquivos

O arquivo está em uma pasta nomeada de JogoVelhaMod que contém 4 arquivos. No arquivo funcoes.cpp possui as funções utilizadas, no arquivo cabec.hpp contém a chamada das funções que estão em funcoes.cpp, um arquivo JogoVelhaMod.cpp, que contém a função Main, para compilar o programa, e o JogoVelhaMod.exe que é executável do código.



2.2. Funções

Um detalhe importante, todas as minhas funções, tem a matriz como parâmetro, para que ela possa ser manipulada e analisada lá dentro da função.

Veja abaixo uma imagem de todo o código, contido no arquivo de funções:

```cpp
#include <iostream>
#include <windows.h>
#include <time.h>

using namespace std;

// todas as funções, passam a matriz como parametro, pois ela será verificada dentro deles

void ImprimeJogo(char jogo[3][3]){
    // utilizado para imprimir o jogo no ínicio, preenchendo com espaços
    int linha, coluna;
    for (linha = 0; linha < 3; linha++){
        for (coluna = 0; coluna < 3; coluna++){
            jogo[linha][coluna] = ' ';
        }
    }
}

void ImprimeJogoCoo(char jogo[3][3]){
    int l, c;
    //imprimir o jogo ao longo das inserções, deixando as coordenadas visíveis para facilitar o jogo
    cout << "\n\n\t 0   1   2\n\n"; //imprimir as coordenadas das colunas
    for(l = 0; l < 3; l++){
        for(c = 0; c < 3; c++){
            if(c == 0){
                printf("\t"); //quando estiver na coluna 0 sempre vai dar uma tabulação, para sempre imprimir o | na posição certa
            }
            printf(" %c ", jogo[l][c]); // imprimir o jogo como ele está, depois de ser feita a alteração
            if(c < 2){ //sempre que a coluna for menor que 2, ele vai colocar o |
                printf("|");
            }
            if(c == 2){ //sempre que for igual a dois, vai ser imprimida a coordenada das linhas na frente
                printf("  %d", l);
            }
        }
        if(l < 2){ //Após percorrer o laço de colunas, se a linha for abaixo de 2, ele vai imprimir a linha que divide as linhas
            printf("\n\t-----------");
        }
        printf("\n");
    }
}

boolean vitoriaLinha(char jogo[3][3]){
    if(jogo[0][0] == '0' && jogo[0][1] == '0' && jogo[0][2] == '0' ||   //1ª linha
       jogo[1][0] == '0' && jogo[1][1] == '0' && jogo[1][2] == '0' ||   //2ª linha
       jogo[2][0] == '0' && jogo[2][1] == '0' && jogo[2][2] == '0'){    //3ª linha
        return true; //o true é para a vitória do jogador 1
    }

    if(jogo[0][0] == 'X' && jogo[0][1] == 'X' && jogo[0][2] == 'X' ||   //1ª linha
       jogo[1][0] == 'X' && jogo[1][1] == 'X' && jogo[1][2] == 'X' ||   //2ª linha
       jogo[2][0] == 'X' && jogo[2][1] == 'X' && jogo[2][2] == 'X'){    //3ª linha
        return false; //o false é para a vitória do jogador 2
    }
}

boolean vitoriaColuna(char jogo[3][3]){
    if(jogo[0][0] == '0' && jogo[1][0] == '0' && jogo[2][0] == '0' ||   //1ª linha
       jogo[0][1] == '0' && jogo[1][1] == '0' && jogo[2][1] == '0' ||   //2ª linha
       jogo[0][2] == '0' && jogo[1][2] == '0' && jogo[2][2] == '0'){    //3ª linha
       return true; //o true é para a vitória do jogador 1
    }

    if(jogo[0][0] == 'X' && jogo[1][0] == 'X' && jogo[2][0] == 'X' ||   //1ª linha
       jogo[0][1] == 'X' && jogo[1][1] == 'X' && jogo[2][1] == 'X' ||   //2ª linha
       jogo[0][2] == 'X' && jogo[1][2] == 'X' && jogo[2][2] == 'X'){    //3ª linha
       return false; //o false é para a vitória do jogador 2
    }
}

boolean diagonalPrincipal(char jogo[3][3]){
    if(jogo[0][0] == '0' && jogo[1][1] == '0' && jogo[2][2] == '0'){    //diagonal Principal
        return true; //o true é para a vitória do jogador 1
    }

    if(jogo[0][0] == 'X' && jogo[1][1] == 'X' && jogo[2][2] == 'X'){    //diagonal Principal
        return false; //o false é para a vitória do jogador 2
    }
}
```

E logo abaixo cada uma das partes da imagem anterior, com a explicação detalhada, além dos comentários feitos no próprio código:

a. A função ImprimeJogo, de tipo void, serve apenas para preencher a matriz, com espaços, por meio de 1 laço de repetição, o *for*. Ela é chamada apenas no início de cada partida no arquivo principal

```
 9   void ImprimeJogo(char jogo[3][3]){
10       // utilizado para imprimir o jogo no ínicio, preenchendo com espaços
11       int linha, coluna;
12       for (linha = 0; linha < 3; linha++){
13           for (coluna = 0; coluna < 3; coluna++){
14               jogo[linha][coluna] = ' ';
15           }
16       }
17   }
```

b. A função ImprimeJogoCoo, de tipo void, serve para imprimir o jogo, com os valores já inseridos, com “interface gráfica”, como está descrito nos comentários.

```
19  void ImprimeJogoCoo(char jogo[3][3]){
20      int l, c;
21      //imprimir o jogo ao longo das inserções, deixando as coordenadas visíveis para facilitar o jogo
22      cout << "\n\n\t 0   1   2\n\n"; //imprimir as coordenadas das colunas
23      for(l = 0; l < 3; l++){
24          for(c = 0; c < 3; c++){
25              if(c == 0){
26                  printf("\t"); //quando estiver na coluna 0 sempre vai dar uma tabulação, para sempre imprimir o | na posição certa
27              }
28              printf(" %c ", jogo[l][c]); // imprimir o jogo como ele está, depois de ser feita a alteração
29              if(c < 2){ //sempre que a coluna for menor que 2, ele vai colocar o |
30                  printf("|");
31              }
32              if(c == 2){ //sempre que for igual a dois, vai ser imprimida a coordenada das linhas na frente
33                  printf("  %d", l);
34              }
35          }
36          if(l < 2){ //Após percorrer o laço de colunas, se a linha for abaixo de 2, ele vai imprimir a linha que divide as linhas
37              printf("\n\t-----------");
38          }
39          printf("\n");
40      }
41  }
```

c. A função vitoriaLinha de tipo boolean, a princípio a coluna é alterada e a linha se mantém a mesma, pois estamos verificando a vitória por linha. A posição da matriz é definida por jogo[linha][coluna]. Se a coluna na posição 0, linha na posição 0, **E** a coluna na posição 1 e linha na posição 0, **E** a coluna na posição 2 e linha na posição 0, forem todas verdadeiras temos uma vitória por linha. Há um **OUs** para verificar a linha seguinte, caso a anterior seja falsa. Por se tratar de uma função booleana, deve ter um retorno, se atender ao primeiro *if* retorna *true*, e significa que o jogador 1, ganhou, se atender ao segundo *if* retorna *false* e significa que o jogador 2 ganhou.

```
43  boolean vitoriaLinha(char jogo[3][3]){
44      if(jogo[0][0] == '0' && jogo[0][1] == '0' && jogo[0][2] == '0' ||   //1ª linha
45         jogo[1][0] == '0' && jogo[1][1] == '0' && jogo[1][2] == '0' ||   //2ª linha
46         jogo[2][0] == '0' && jogo[2][1] == '0' && jogo[2][2] == '0'){    //3ª linha
47          return true; //o true é para a vitória do jogador 1
48      }
49
50      if(jogo[0][0] == 'X' && jogo[0][1] == 'X' && jogo[0][2] == 'X' ||   //1ª linha
51         jogo[1][0] == 'X' && jogo[1][1] == 'X' && jogo[1][2] == 'X' ||   //2ª linha
52         jogo[2][0] == 'X' && jogo[2][1] == 'X' && jogo[2][2] == 'X'){    //3ª linha
53          return false; //o false é para a vitória do jogador 2
54      }
55  }
```

d. A função vitoriaColuna de tipo boolean, verifica cada uma das possibilidades por coluna. Segue a mesma lógica da vitoriaLinha, porém a linha é alterada e a coluna se mantém a mesma, pois estamos verificando a vitória por coluna. A posição da matriz é definida por jogo[linha][coluna]. Se a coluna na posição 0, linha na posição 0, **E** a coluna na posição 0 e linha na posição 1, **E** a coluna na posição 0 e linha na posição 2, forem todas verdadeiras temos uma vitória por coluna. Há um **OUs** para verificar a coluna seguinte, caso a anterior seja falsa. Mesmo esquema da função anterior, por se tratar de uma função booleana, deve ter um retorno, se atender ao primeiro *if* retorna *true*, e significa que o jogador 1 ganhou, se atender ao segundo *if* retorna *false* e significa que o jogador 2 ganhou.

```
57  boolean vitoriaColuna(char jogo[3][3]){
58      if(jogo[0][0] == '0' && jogo[1][0] == '0' && jogo[2][0] == '0' ||   //1ª linha
59         jogo[0][1] == '0' && jogo[1][1] == '0' && jogo[2][1] == '0' ||   //2ª linha
60         jogo[0][2] == '0' && jogo[1][2] == '0' && jogo[2][2] == '0'){    //3ª linha
61         return true; //o true é para a vitória do jogador 1
62      }
63
64      if(jogo[0][0] == 'X' && jogo[1][0] == 'X' && jogo[2][0] == 'X' ||   //1ª linha
65         jogo[0][1] == 'X' && jogo[1][1] == 'X' && jogo[2][1] == 'X' ||   //2ª linha
66         jogo[0][2] == 'X' && jogo[1][2] == 'X' && jogo[2][2] == 'X'){    //3ª linha
67         return false; //o false é para a vitória do jogador 2
68      }
69  }
```

e. A função diagonalPrincipal de tipo boolean, onde ele verifica se a diagonal principal está preenchida. Lembrando que a diagonal principal é a mesma diagonal principal de matriz na matemática, ela parte do primeiro termo da primeira linha, passando pelo termo central em relação às colunas e às linhas, e indo até o último termo da última linha. Para as condicionais, é o mesmo esquema das anteriores, por se tratar de uma função booleana, deve ter um retorno, se atender ao primeiro *if* retorna *true*, e significa que o jogador 1 ganhou, se atender ao segundo *if* retorna *false* e significa que o jogador 2 ganhou.

```
71  boolean diagonalPrincipal(char jogo[3][3]){
72      if(jogo[0][0] == '0' && jogo[1][1] == '0' && jogo[2][2] == '0'){    //diagonal Principal
73          return true; //o true é para a vitória do jogador 1
74      }
75
76      if(jogo[0][0] == 'X' && jogo[1][1] == 'X' && jogo[2][2] == 'X'){    //diagonal Principal
77          return false; //o false é para a vitória do jogador 2
78      }
79  }
```

f. A função diagonalSecundaria de tipo boolean, onde ele verifica se a diagonal secundária está preenchida. A diagonal secundária, seria uma espécie de espelho da diagonal principal, logo, pega o último termo da primeira linha, o termo central da matriz em relação às linhas e às colunas, e vai até o primeiro termo da última linha. Mesmo esquema da anterior, o primeiro *if* é para verificar as jogadas do jogador 1, e o segundo *if* do jogador dois. Por se tratar de uma função booleana, deve ter um retorno, se atender ao primeiro *if* retorna *true*, e significa que o jogador 1 ganhou, se atender ao segundo *if* retorna *false* e significa que o jogador 2 ganhou.

```
81  boolean diagonalSecundaria(char jogo[3][3]){
82      if(jogo[0][2] == '0' && jogo[1][1] == '0' && jogo[2][0] == '0'){    //diagonal secundária
83          return true; //o true é para a vitória do jogador 1
84      }
85
86      if(jogo[0][2] == 'X' && jogo[1][1] == 'X' && jogo[2][0] == 'X'){    //diagonal secundária
87          return false; //o false é para a vitória do jogador 2
88      }
89  }
```

2.3. Cabeçalho

Um arquivo de extensão *.hpp, que inclui o arquivo de funções, para fazer a chamada das funções do arquivo funcoes.cpp. Esse arquivo funciona como ponte de comunicação entre os arquivos funcoes.cpp e JogoVelhaMod.cpp. Ele será chamado no arquivo principal, para que seja possível chamar as funções que estão no arquivo de funcoes.cpp.

Veja abaixo uma imagem de todo o código:

```
1   #ifndef FUNCOES_H
2   #define FUNCOES_H
3   #include "funcoes.cpp"
4
5   void ImprimeJogo(char* jogo[3][3]);
6   void ImprimeJogoCoo(char* jogo[3][3]);
7   boolean vitoriaLinha(char* jogo[3][3]);
8   boolean vitoriaColuna(char*  jogo[3][3]);
9   boolean diagonalPrincipal(char* jogo[3][3]);
10  boolean diagonalSecundaria(char* jogo[3][3]);
11
12  #endif
```

2.4. Arquivo Principal (*Main*)

É o arquivo onde todo o código será compilado, fazendo a chamada de funções, e fazendo uso da lógica, e laços de repetição, condicionais, inserção e saída de dados, declaração de variáveis, e manipulação de variáveis.

Veja abaixo uma imagem de todo o código desse arquivo principal:

```cpp
#include <stdio.h>
#include <stdlib.h>
#include "cabec.hpp"

int main(){
    UINT CPAGE_UTF8 = 65001;
    UINT CPAGE_DEFAULT = GetConsoleOutputCP();
    SetConsoleOutputCP(CPAGE_UTF8);
    system("cls");

    int linha, coluna, jogador = 1, jogadas = 0, op, j1 = 0, j2 = 0;
    char jogo[3][3];
    boolean ganhou = false;

    do{
        // laço para jogar novamente
        jogadas = 0; //jogadas inicializadas em 0, para que o laço de jogadas possa ser finalizado, caso não haja vencedor
        ImprimeJogo(jogo);
        do{
            // repete até alguém ganhar ou atingir 9 jogadas e dar empate

            do{
            // ler coordenadas
                system("cls");
                cout << "___Jogo da Velha___\n\n\t\tPlacar\n\nJogador 1: " << j1 << "\t\tJogador 2: " << j2;
                ImprimeJogoCoo(jogo);
                cout << "\n JOGADOR 1 = 0\n JOGADOR 2 = X \n";
                cout << "\n JOGADOR " << jogador << " \nDigite a linha dê um espaço digite a coluna que deseja jogar: ";
                scanf("%d%d", &linha, &coluna);
            //Laço que impede de digitar números maiores que as coordenadas, com uma mensagem diferente
                while(linha >= 3 || coluna >= 3){
                    system("cls");
                    cout << "___Jogo da Velha___\n\n\t\tPlacar\n\nJogador 1: " << j1 << "\t\tJogador 2: " << j2;
                    ImprimeJogoCoo(jogo);
                    cout << "\n JOGADOR 1 = 0\n JOGADOR 2 = X \n";
                    cout << "\n JOGADOR " << jogador << " \nDigite a linha dê um espaço digite a coluna que deseja jogar(valores entre 0 e 2): ";
                    scanf("%d%d", &linha, &coluna);
                }
            }while((linha < 0 || linha > 2) || (coluna < 0 || coluna > 2) || (jogo[linha][coluna] != ' '));

            // inserção de coordenadas
            if(jogador == 1){
                jogo[linha][coluna] = '0';
                jogador++; //incremento para ser o jogador 2
            }
            else{
                jogo[linha][coluna] = 'X';
                jogador = 1; //retornar para 1, para ser o jogador 1
            }
            jogadas++;  //incremento de jogadas, pois com 9 jogadas acaba
            system("cls");
    //Verificação de vitórias
        // alguém ganhou por linha
            if(vitoriaLinha(jogo) == true){
                cout << "\nO jogador 1 venceu por linha!\n";
                j1++;
                ganhou = true;
            }
            if(vitoriaLinha(jogo) == false){
                cout << "\nO jogador 2 venceu por linha!\n";
                j2++;
                ganhou = true;
            }
        // alguém ganhou por coluna
            if(vitoriaColuna(jogo) == true){
                cout << "\nO jogador 1 venceu por coluna!\n";
                j1++;
                ganhou = true;
            }
            if(vitoriaColuna(jogo) == false){
                cout << "\nO jogador 2 venceu por coluna!\n";
                j2++;
                ganhou = true;
            }
        // alguém ganhou na diagonal principal
            if(diagonalPrincipal(jogo) == true){
                cout << "\nO jogador 1 venceu na diagonal. principal!\n";
                j1++;
                ganhou = true;
            }
            if(diagonalPrincipal(jogo) == false){
                cout << "\nO jogador 2 venceu na diagonal. principal!\n";
                j2++;
                ganhou = true;
            }
        // alguém ganhou na diagonal secundária
            if(diagonalSecundaria(jogo) == true){
                cout << "\nO jogador 1 venceu na diagonal Secundária!\n";
                j1++;
                ganhou = true;
            }
            if(diagonalSecundaria(jogo) == false){
                cout << "\nO jogador 2 venceu na diagonal Secundária!\n";
                j2++;
                ganhou = true;
            }
        }while(ganhou == false && jogadas < 9);

    // imprimir jogo final
        cout << "___Jogo da Velha___\n\n\t\tPlacar\n\nJogador 1: " << j1 << "\t\tJogador 2: " << j2;
        ImprimeJogoCoo(jogo);
    //caso de empate
        if(ganhou == false){
            cout << "\nO jogo finalizou sem gamhador!\n";
        }

        cout << "\nDigite 1 para jogar novamente: \n";
        cin >> op;
    }while(op == 1); // fim do laço para jogar novamente
    return 0;
}
```

A seguir, serão mostradas as partes do programa com explicações detalhadas de cada parte do programa, com a imagem logo abaixo, além dos comentários feitos no mesmo:

a. Inclusão de cabec.hpp, para chamar as funções de funcoes.cpp. Algumas configurações passadas em sala de aula, para que o compilador aceite os caracteres especiais na impressão. Logo abaixo tem a declaração de variáveis, cada uma com objetivo específico. Abaixo serão listadas todas elas e seu respectivo uso:

- linha: utilizado para receber o valor que será parte da coordenada da linha da matriz.
- coluna: utilizado para receber o valor que será parte da coordenada da coluna da matriz.
- jogador: serve para alternar quem irá jogar, obvio que se inicia no jogador 1.
- jogadas: é o contador de jogadas, que sofre incrementos ao longo das jogadas e serve como condição de parada do laço de repetição da partida.
- op: ao fim da partida, ela recebe um valor do usuário para jogar novamente, ou encerrar as partidas.
- j1: pontuação do jogador 1, que sofre incremento conforme ele ganha as partidas.
- j2: pontuação do jogador 2, que sofre incremento conforme ele ganha as partidas.
- jogo[3][3]: matriz que será o “tabuleiro” do jogo da velha.
- ganhou: é uma variável booleana, que sofre alteração quando alguém ganha e serve como condição alternativa de parada do laço de repetição da partida.

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include "cabec.hpp"
4
5   int main(){
6       UINT CPAGE_UTF8 = 65001;
7       UINT CPAGE_DEFAULT = GetConsoleOutputCP();
8       SetConsoleOutputCP(CPAGE_UTF8);
9       system("cls");
10
11      int linha, coluna, jogador = 1, jogadas = 0, op, j1 = 0, j2 = 0;
12      char jogo[3][3];
13      boolean ganhou = false;
```

b. Há 3 *do-while's*, o primeiro para iniciar a partida e possibilitar o início de uma nova partida ao fim da anterior, o segundo para verificar as posições, executar a partida e determinar um vencedor, e o terceiro para inserir os caracteres nas devidas coordenadas. As jogadas são iniciadas em 0, dentro do primeiro *do-while*, pois serão incrementadas dentro dele, e se ao fim do laço, forem jogar novamente, ela começará em 0 de novo. A função ImprimeJogo(jogo) serve para preencher a matriz com espaços, como já foi dito.

```
15  do{
16      // laço para jogar novamente
17      jogadas = 0; //jogadas inicializadas em 0, para que o laço de jogadas possa ser finalizado, caso não haja vencedor
18      ImprimeJogo(jogo);
19      do{
20          // repete até alguém ganhar ou atingir 9 jogadas e dar empate
21
22          do{
23          // ler coordenadas
24              system("cls");
25              cout << "___Jogo da Velha___\n\n\t\tPlacar\n\nJogador 1: " << j1 << "\t\tJogador 2: " << j2;
26              ImprimeJogoCoo(jogo);
27              cout << "\n JOGADOR 1 = 0\n JOGADOR 2 = X \n";
28              cout << "\n JOGADOR " << jogador << " \nDigite a linha dê um espaço digite a coluna que deseja jogar: ";
29              scanf("%d%d", &linha, &coluna);
30          //Laço que impede de digitar números maiores que as coordenadas, com uma mensagem diferente
31              while(linha >= 3 || coluna >= 3){
32                  system("cls");
33                  cout << "___Jogo da Velha___\n\n\t\tPlacar\n\nJogador 1: " << j1 << "\t\tJogador 2: " << j2;
34                  ImprimeJogoCoo(jogo);
35                  cout << "\n JOGADOR 1 = 0\n JOGADOR 2 = X \n";
36                  cout << "\n JOGADOR " << jogador << " \nDigite a linha dê um espaço digite a coluna que deseja jogar(valores entre 0 e 2): ";
37                  scanf("%d%d", &linha, &coluna);
38              }
39          }while((linha < 0 || linha > 2) || (coluna < 0 || coluna > 2) || (jogo[linha][coluna] != ' '));
```

c. Aprofundando mais um pouco no terceiro *do-while*. Na figura 1, ele começa com uma limpeza de tela, imprime o placar, logo em seguida imprime o jogo da velha, como na figura 2. Logo em seguida mostra qual é o carácter de cada jogador, e em seguida a vez de qual jogador é. Utilizei o *scanf*, para ler dois valores sem precisar dar *Enter*, porém o *cin* também me atenderia. Abaixo tem o início de outro *while* para impedir que o usuário digite valores acima 2, e exibir uma mensagem diferente da anterior para que o usuário digite um valor dentro das possibilidades, e repete tudo acima, até que o jogador da vez, digite um valor para sair de fato do *while*, e assim sair do terceiro *do-while*.

```
22          do{
23          // ler coordenadas
24              system("cls");
25              cout << "___Jogo da Velha___\n\n\t\tPlacar\n\nJogador 1: " << j1 << "\t\tJogador 2: " << j2;
26              ImprimeJogoCoo(jogo);
27              cout << "\n JOGADOR 1 = 0\n JOGADOR 2 = X \n";
28              cout << "\n JOGADOR " << jogador << " \nDigite a linha dê um espaço digite a coluna que deseja jogar: ";
29              scanf("%d%d", &linha, &coluna);
30          //Laço que impede de digitar números maiores que as coordenadas, com uma mensagem diferente
31              while(linha >= 3 || coluna >= 3){
32                  system("cls");
33                  cout << "___Jogo da Velha___\n\n\t\tPlacar\n\nJogador 1: " << j1 << "\t\tJogador 2: " << j2;
34                  ImprimeJogoCoo(jogo);
35                  cout << "\n JOGADOR 1 = 0\n JOGADOR 2 = X \n";
36                  cout << "\n JOGADOR " << jogador << " \nDigite a linha dê um espaço digite a coluna que deseja jogar(valores entre 0 e 2): ";
37                  scanf("%d%d", &linha, &coluna);
38              }
39          }while((linha < 0 || linha > 2) || (coluna < 0 || coluna > 2) || (jogo[linha][coluna] != ' '));
```

Figura 1

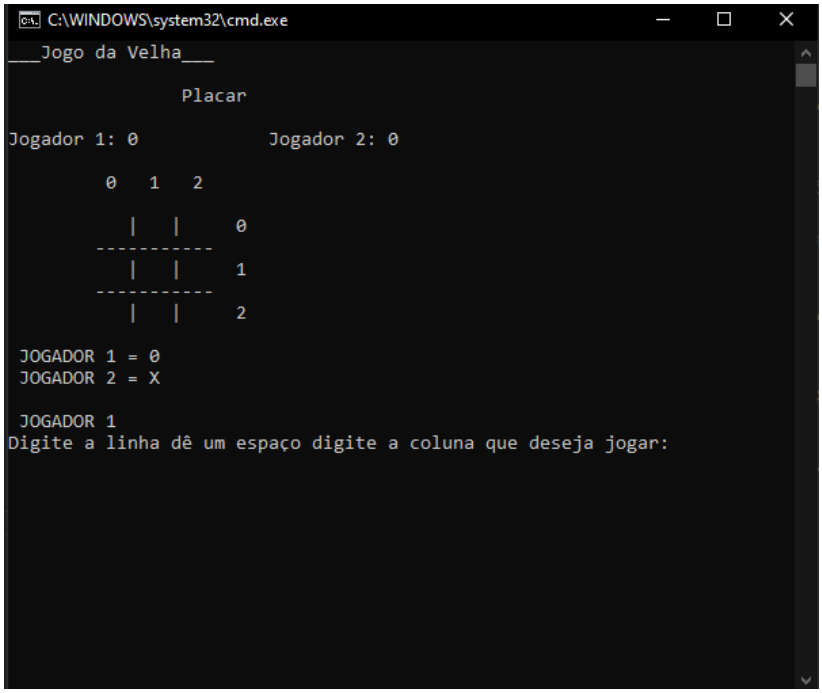


Figura 2

d. Abaixo do terceiro *do-while*, tem *if-else* para saber o que inserir de acordo com o jogador. Se for o jogador 1, ele vai cair no *if* e colocar 0 na coordenada digitada, vai fazer um incremento no valor de jogador, e o valor será 2, e vai contar mais 1 jogada, e ele irá percorrer o resto do segundo *do-while*, e fazer a verificação de posições para ver se há vitórias, não havendo, ele retorna para o terceiro *do-while*, porém com o jogador = 2, logo é a vez do jogador 2, que após inserir a coordenada, vem pras condicionais, onde o jogador não é igual a 1, logo ele cai no *else*, e vai colocar o X na coordenada digitada, e vai voltar o jogador para 1, incrementar a jogada mais uma vez, e ir para o terceiro *do-while*, e vai ser a vez do jogador 1 novamente, e tudo se repetirá, até jogadas ser igual a 9 ou ter um vencedor.

```
41          // inserção de coordenadas
42          if(jogador == 1){
43              jogo[linha][coluna] = '0';
44              jogador++; //incremento para ser o jogador 2
45          }
46          else{
47              jogo[linha][coluna] = 'X';
48              jogador = 1; //retornar para 1, para ser o jogador 1
49          }
50          jogadas++;  //incremento de jogadas, pois com 9 jogadas acaba
```

e. Aqui são feitas as verificações de vitórias de acordo com as funções booleanas de verificação de vitória, onde o *true* significa que o jogador 1 ganhou, e *false* que o jogador 2 ganhou. Se já tiverem jogadas para ganhar de alguma forma, seja por linha, coluna ou diagonal, ele conta mais 1 para o vencedor no placar, e muda o ganhou para *true*, assim acabando o segundo *do-while*.

```
52      //Verificação de vitórias
53          // alguém ganhou por linha
54              if(vitoriaLinha(jogo) == true){
55                  cout << "\nO jogador 1 venceu por linha!\n";
56                  j1++;
57                  ganhou = true;
58              }
59              if(vitoriaLinha(jogo) == false){
60                  cout << "\nO jogador 2 venceu por linha!\n";
61                  j2++;
62                  ganhou = true;
63              }
64          // alguém ganhou por coluna
65              if(vitoriaColuna(jogo) == true){
66                  cout << "\nO jogador 1 venceu por coluna!\n";
67                  j1++;
68                  ganhou = true;
69              }
70              if(vitoriaColuna(jogo) == false){
71                  cout << "\nO jogador 2 venceu por coluna!\n";
72                  j2++;
73                  ganhou = true;
74              }
```

```
75          // alguém ganhou na diagonal principal
76              if(diagonalPrincipal(jogo) == true){
77                  cout << "\nO jogador 1 venceu na diagonal. principal!\n";
78                  j1++;
79                  ganhou = true;
80              }
81              if(diagonalPrincipal(jogo) == false){
82                  cout << "\nO jogador 2 venceu na diagonal. principal!\n";
83                  j2++;
84                  ganhou = true;
85              }
86          // alguém ganhou na diagonal secundária
87              if(diagonalSecundaria(jogo) == true){
88                  cout << "\nO jogador 1 venceu na diagonal Secundária!\n";
89                  j1++;
90                  ganhou = true;
91              }
92              if(diagonalSecundaria(jogo) == false){
93                  cout << "\nO jogador 2 venceu na diagonal Secundária!\n";
94                  j2++;
95                  ganhou = true;
96              }
97          }while(ganhou == false && jogadas < 9);
```

f. Caso alguém ganhe, o valor de ganhou será *true* e consequentemente o número de jogadas será menor que 9, e utilizando da tabela-verdade, o resultado será falso e ele sairá do segundo *do-while*, e irá imprimir o placar atualizado e o jogo final. Caso ninguém ganhe, o valor de ganhou será *false* e consequentemente o número de jogadas será igual a 9, e utilizando da tabela verdade, o resultado será falso também, e ele vai sair do segundo *do-while*, e cair no *if*, e falar que empatou.

```
97          }while(ganhou == false && jogadas < 9);
98
99      // imprimir jogo final
100         cout << "___Jogo da Velha___\n\n\t\tPlacar\n\nJogador 1: " << j1 << "\t\tJogador 2: " << j2;
101         ImprimeJogoCoo(jogo);
102     //caso de empate
103         if(ganhou == false){
104             cout << "\nO jogo finalizou sem gamhador!\n";
105         }
106
```

g. Após isso, será exibida uma mensagem, perguntando se desejam jogar novamente, caso digite 1, ele reiniciará o primeiro *do-while*, passando por todas as partes citadas acima, mas caso seja um número diferente, ele encerra primeiro *do-while* e o programa termina.

```
106
107             cout << "\nDigite 1 para jogar novamente: \n";
108             cin >> op;
109         }while(op == 1); // fim do laço para jogar novamente
110         return 0;
111     }
```

3. CONCLUSÃO

Ao longo do trabalho, encontrei dificuldades ao montar a “interface gráfica”, porém utilizei o C, para poder resolver depois de ler sites e fóruns, tentei laços de repetição para verificar as vitórias, mas estava dando erro, por se tratar de uma matriz pequena, fiz caso por caso. Consegui desenvolver a lógica, e o que eu via como um bicho de 7 cabeças, era na verdade algo simples de se resolver, era só me esforçar um pouco e exercitar a lógica. A princípio tinha feito um arquivo procedural e sem funções, porém com conhecimentos de final de semestre, organizei ele de forma multi-paradigma e com funções. Com certeza melhorou minha visão e entendimento a respeito de álgebra booleana, e estou contente com meu desenvolvimento e resultado final. Ainda não é um programa com visões empresariais, pois não visa lucro, não tem segurança, e nem é necessário, não tem fins para gestão, e nem é um software que gera lucro, porém a demanda de tempo, lógica a ser pensada e desenvolvida, são similares a de uma visão empresarial.

4. REFERÊNCIAS

Alisson. História do C / C++. **DEVMEDIA**, 2012. Disponível em: https://www.devmedia.com.br/historia-do-c-c/24029#:~:text=A%20Linguagem%20C%20foi%20inventada,BCPL%2C%20desenvolvida%20por%20Martin%20Richards. Acesso em: 2 de julho de 2022.

Larissa Gabriela. As linguagens C e C++: qual a diferença entre elas?. **Alura**. 2021. Disponível em: https://www.alura.com.br/artigos/linguagens-c-c-qual-diferenca-entre-elas?gclid=CjwKCAjw2f-VBhAsEiwAO4lNeOf1MGIwXhjJ6qsEyv3yyyBADdauZUBn8SdioVSxViB0BsScC6FgyRoCQQkQAvD_BwE. Acesso em: 2 de julho de 2022.

Linguagem C – Exemplos e aplicações da programação nessa linguagem. **I DO CODE**, 12/06/2020.Disponível em: https://idocode.com.br/blog/programacao/exemplos-e-aplicacoes-da-linguagem-c/#:~:text=A%20linguagem%20C%20pode%20ser,Linux%20e%20o%20Mac%20OS. Acesso em: 2 de julho de 2022