RELATÓRIO “JOGO DA VELHA VISUALG”

ALUNO: MATHEUS HENRIQUE FERNANDES MARTINS

CURSO: CIÊNCIAS DA COMPUTAÇÃO

1º PERÍODO

INTRODUÇÃO

Para a atividade do trabalho proposto no 2º Bimestre foi apresentado um problema em que se construísse um **jogo da velha** dentro da linguagem do **VISUALG,** com algumas regras descritas, a partir dai então comecei a pensar e ter algumas ideias de como iria conseguir realizar a atividade proposta do trabalho, era possível visualizar o código porém, haveria tamanha dificuldade para escreve-lo e quais funções e estruturas usar.

DESENVOLVIMENTO

1º FASE - PROTOTIPO INDEFERIDO -

A minha primeira ideia era criar um código que utilizasse apenas a função ***ESCOLHA***, para representar as coordenadas.

```
caso 11
   limpatela
   escreval("     1    2    3 ")
   escreval("    x  |    |    ")
   escreval("1 ____|____|____")
   escreval("      |    |    ")
   escreval("2 ____|____|____")
   escreval("    o  |    |    ")
   escreval("3     |    |    ")
```

```
caso 12
   limpatela
   escreval("     1    2    3 ")
   escreval("     o  | x  |    ")
   escreval("1 ____|____|____")
   escreval("      |    |    ")
   escreval("2 ____|____|____")
   escreval("      |    |    ")
   escreval("3     |    |    ")

caso 13
   limpatela
```

Porém logo no início já era notável que haveria erros e varias indagações a se fazer, primeiramente que o modo como foi realizado na imagem, na minha cabeça era para ser aleatório, e no inicio iria apenas fazer o "X" ser a principal peça a ser controlada, porém isso fugiria das principais regras do trabalho

2. **Jogadas Alternadas**
   - O sistema deve permitir que dois jogadores façam suas jogadas alternadamente, colocando 'X' ou 'O' em uma posição vazia do tabuleiro.

Neste protótipo usei as variáveis: ***jogador***, ***inimigo*** para descrever e dar função as peças, as variáveis: ***p, j, i ,q,*** no qual foi usada apenas a variável ***j,*** ler a opção de com qual peça você iria querer jogar.

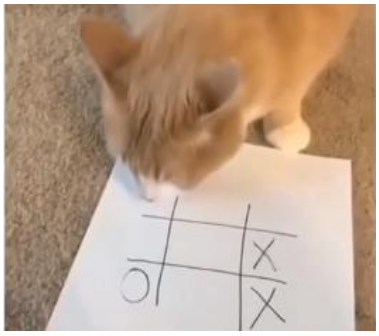
```
var
   jogador: inteiro
   inimigo: inteiro
   p : inteiro
   j: Inteiro
   i: Caracter
   q: Caracter
```

```
escreval ("Selecione com qual você quer jogar")
escreval ("[1] Jogar como X")
escreval ("[2] Jogar como O")
leia  (j)
escolha (j)

caso 1
   leia (jogador)

   limpatela
   escreval ("Você é o: X")
   escreval ("Derrote a: O")
```

E a cada momento que analisava este protótipo, via mais detalhes, como a falta do sistema de pontuação que também era parte das regras do trabalho, e as jogadas, do modo o qual eu estava fazendo, seria necessário varias possibilidades, tendo que repetir varias linhas e criando mais de 126 possibilidades diferentes, e isso sem contar a falta de liberdade onde a peça inimiga iria aparecer de forma aleatória após responder o texto com base na função escolha, era um projeto exorbitante porém impossível, logo foi necessário indeferir este protótipo e começar a estudar outras funções para fazer o trabalho.

2º FASE – ENTENDENDO O JOGO DA VELHA –

Depois de indeferir o protótipo, eu precisava pensar no que iria fazer, então precisava entender o conceito básico do jogo da velha:



- **Quadrados 3x3**
- **Escolha de símbolos: X ou O**
- **Jogo em turnos entre os 2 jogadores**
- **Vence marcando 3 posições retas, horizontal ou vertical**
- **Caso ninguém marque posições o jogo empata**

A maior dificuldade para mim seria fazer a alternação de turnos entre as peças “X” e “O” (oque envolveu uma pequena gambiarra no projeto final), porém primeiro era necessário compreender como funcionava o conjunto de engrenagens dentro do jogo da velha, outra dificuldade seria realizar a marcação das posições para contabilizar se houvesse as 3 posições necessárias para ganhar o jogo, ou para empatar.

3º FASE – ESTUDANDO NOVAS FUNÇÕES –

Para realizar o trabalho, observei que não seria possível colocando estruturas simples no código, como *enquanto, escolha,* então era necessário estudar funções mais “complexas” para conseguir construir um código mais estruturado, então me aprofundei em estruturas como : *vetor, procedimento, matriz,* o que me deu uma grande ajuda, apliquei elas no código final, inclusive a estrutura *procedimento*, que me ajudou a montar a interface do jogo da velha no código final, criei alguns algoritmos com esses códigos para praticar para ter uma noção de como cada um funcionava.

4º FASE – O CÓDIGO FINAL –

No Google Classroom em anexo junto com o relatório está a atividade proposta para o 2º trabalho, o código do jogo da velha no VisualG.

```
1 algoritmo "Jogo da Velha"
2 var
3    jog: vetor[1..3,1..3] de caractere
4    comp,cont, i, j, a, b, fim, ng: inteiro
5    vit, ep: inteiro
6    resp: caractere
```

As variáveis que utilizei, sendo *jog* com um vetor e matriz, que usei para fazer o tabuleiro 3x3, junto a uma função de procedimento. *Comp* é o compilador que alterna entre “X” e “O” mais para frente explica como, *cont* é o contador, para colocar a marcação de números que servem como “coordenadas” dentro da interface do código, na fase de testes irei explicar o pequeno problema que tive com essa variável, *i, j* para o vetor, *a, b* para orientação de números, *fim* para detectar quando ocorre vitória ou empate, *ng* para iniciar um novo jogo com placar atualizado.

```
procedimento interface()
inicio
   cont <- 0
   escreval ("================================")
   escreval ("-------- JOGO DA VELHA --------")
   escreval ("================================")
   escreval ("    PARTIDAS GANHAS", vit)
   escreval (" PARTIDAS EMPATADAS", ep)
   escreval ()
   escreval (" =============")
   para i de 1 ate 3 faca
      para j de 1 ate 3 faca
         escreva (" |":2)
         escreva  (jog[i,j]:2)
      fimpara
      escreva (" |")
      escreval ()
      escreval (" =============")
   fimpara
fimprocedimento
```

Procedimento para formação do tabuleiro 3x3

As variáveis *vit, ep* são para determinar se houve partidas ganhas ou partidas que houveram empate.

```
escolha (comp)
caso 1
   jog[a,b] <- " X"
caso 2
   jog[a,b] <- " O"
   comp <- 0
fimescolha
comp <- comp + 1
limpatela
```

Com a estrutura repita na linha 38, o compilador vai repetindo e alternado entre 1 e 2 na execução do código possibilitando a alternação de turnos entre os jogadores.

```
para i de 1 ate 3 faca
   se (jog[i,1] = jog[i,2]) e (jog[i,2] = jog[i,3]) então
      fim <- fim + 1
   fimse
fimpara
para i de 1 ate 3 faca
   se (jog[1,i] = jog[2,i]) e (jog[2,i] = jog[3,i]) então
      fim <- fim + 1
   fimse
fimpara
se (jog[1,1] = jog[2,2]) e (jog[2,2] = jog[3,3]) então
   fim <- fim + 1
fimse
se (jog[1,3] = jog[2,2]) e (jog[2,2] = jog[3,1]) então
   fim <- fim + 1
fimse
```

Para a condição de vitória, cada uma desses números em vermelho é uma posição dentro do quadrado do jogo da velha, se algum desses comandos for executado, o jogo termina em vitória.

```
ate fim = 1 entao
se fim = 1 entao
   escreval ()
   escreval ("VITORIA!")
   vit <- vit +1
fimse

se fim = 0 entao
   escreval ()
   escreval ("EMPATE!")
   ep <- ep + 1
fimse
```

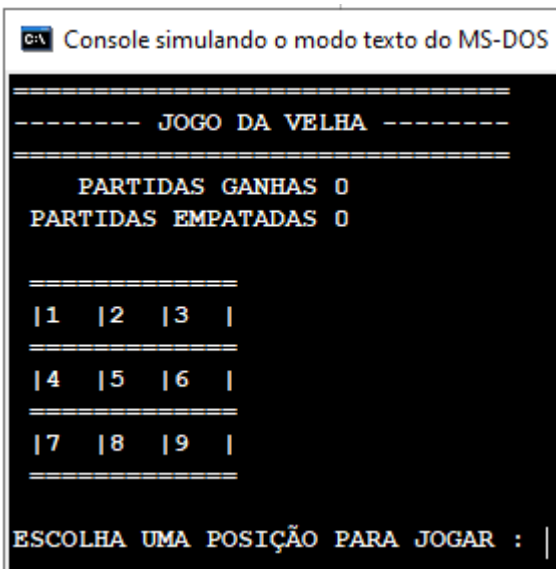
```
escreval ()
interface()
escreval ("DESEJA JOGAR NOVAMENTE?")
escreval ("[1] SIM [2] NÃO")
leia (ng)
escolha (ng)
   limpatela

caso 2
   interrompa
   fimalgoritmo
```

Quando um jogo é finalizado aparece na tela do VisualG se deseja jogar o jogo novamente, caso aperte o número 2 que significa “não”, o algoritmo é interrompido, caso 1, o jogo se repete até conseguir 3 vitorias.

FASE DE TESTES

1º INTERFACE DO JOGO DA VELHA

```
inicio
   cont <- 0
   escreval ("================================")
   escreval ("-------- JOGO DA VELHA --------")
   escreval ("================================")
   escreval ("    PARTIDAS GANHAS", vit)
   escreval (" PARTIDAS EMPATADAS", ep)
   escreval ()
   escreval (" =============")
   para i de 1 ate 3 faca
      para j de 1 ate 3 faca
         escreva (" |":2)
         escreva  (jog[i,j]:2)
      fimpara
      escreva (" |")
      escreval ()
      escreval (" =============")
   fimpara
fimprocedimento
```



A interface do jogo é formada por uma entrada de *procedimento*, as informações das partidas, o corpo da estrutura do tabuleiro do jogo, os números como “coordenadas” onde as peças irão ser colocadas, e o texto orientando o jogador onde a peça deve ser colocada, os números são orientados pela variável *cont*, que cria uma contagem de 1 até 9 para o tabuleiro, inicialmente eu tinha um problema com essa variável, pois ela não estava voltando para o número 1, e sim dando sequencia depois do 9.

2º CONDIÇÃO DE VITORIA

```
para i de 1 ate 3 faca
   se (jog[i,1] = jog[i,2]) e (jog[i,2] = jog[i,3]) então
      fim <- fim + 1
   fimse
fimpara
para i de 1 ate 3 faca
   se (jog[1,i] = jog[2,i]) e (jog[2,i] = jog[3,i]) então
      fim <- fim + 1
   fimse
fimpara
se (jog[1,1] = jog[2,2]) e (jog[2,2] = jog[3,3]) então
   fim <- fim + 1
fimse
se (jog[1,3] = jog[2,2]) e (jog[2,2] = jog[3,1]) então
   fim <- fim + 1
```

Para a condição de vitória ser executada, deve ocorrer o alinhamento de 3 peças iguais na vertical ou horizontal, o fragmento do código acima mostra como isso ocorre, tendo uma posição da interface do jogo em cada linha, e quando ocorre o alinhamento a variável *fim* recebe +1, dando o ponto de vitória, que é caso fim chegue a 1, ao contrario do empate, que o contador do *fim* fica em 0, logo levando o empate na partida

3º POSICIONAMENTO DAS PEÇAS NO TABULEIRO

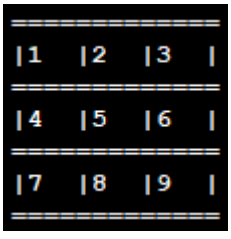
GLOBAL	JOG[1,1]	C	"1"
GLOBAL	JOG[1,2]	C	"2"
GLOBAL	JOG[1,3]	C	"3"
GLOBAL	JOG[2,1]	C	"4"
GLOBAL	JOG[2,2]	C	"5"
GLOBAL	JOG[2,3]	C	"6"
GLOBAL	JOG[3,1]	C	"7"
GLOBAL	JOG[3,2]	C	"8"
GLOBAL	JOG[3,3]	C	"9"

Para marcar o posicionamento das peças cada numero dentro do tabuleiro, é representado nos vetores ao lado, como o exemplo abaixo:

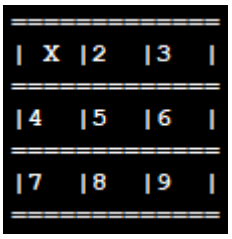
Escopo	Nome	Tipo	Valor
GLOBAL	JOG[1,1]	C	" X"
GLOBAL	JOG[1,2]	C	" O"
GLOBAL	JOG[1,3]	C	" X"
GLOBAL	JOG[2,1]	C	"4"
GLOBAL	JOG[2,2]	C	"5"
GLOBAL	JOG[2,3]	C	"6"
GLOBAL	JOG[3,1]	C	"7"
GLOBAL	JOG[3,2]	C	"8"
GLOBAL	JOG[3,3]	C	"9"

As posições são determinadas de 1 a 3, o painel ao lado mostra as posições em que foram colocadas as peças X e O

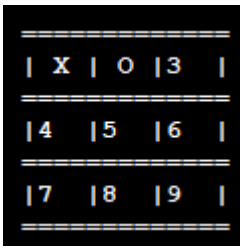
4º ALTERNAÇÃO ENTRE X e O!



COMP	I	1



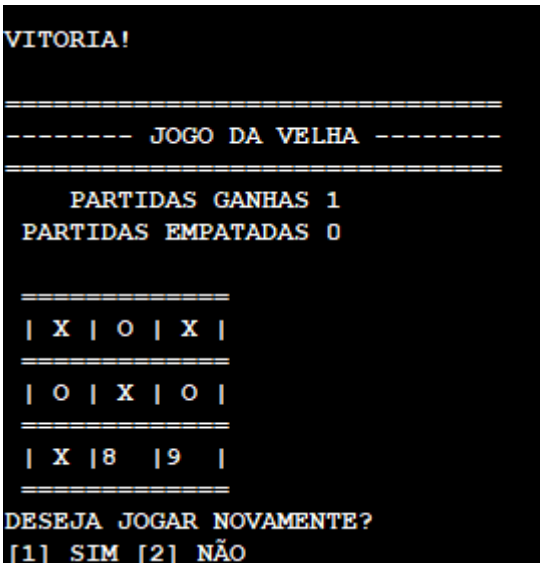
COMP	I	2



COMP	I	1

Para realizar a alternação de turnos a variável *comp* fica alternando entre 1 e 2, quando 1 X, quando 2 O, sendo a variável atribuída a 0 e somando e resetando seu valor de soma.

5º FIM DE JOGO!




```
ate fim = 1 entao
se fim = 1 entao
   escreval ()
   escreval ("VITORIA!")
   vit <- vit +1
fimse

se fim = 0 entao
   escreval ()
   escreval ("EMPATE!")
   ep <- ep + 1
fimse
```

Para concluir o jogo deve-se alinhar 3 peças na vertical ou horizontal, após isso o jogo termina, e o contador de vitorias é aumentado em 1, caso empate o contador de empates é aumentado em 1 e pergunta se quer jogar novamente, caso 1 o jogo repete, onde ocorre o mesmo processo mais 2 vezes, caso 2 o programa do jogo é interrompido dando fim no algoritmo, essa estrutura de repetição se cria até completar 3 partidas ganhas, e em cada estrutura de repetição de partida o contador de vitórias mantem os números, algo que foi bem difícil de configurar no inicio.

```
escreval ()
interface()
escreval ("DESEJA JOGAR NOVAMENTE?")
escreval ("[1] SIM [2] NÃO")
leia (ng)
escolha (ng)
   limpatela

caso 2
   interrompa
   fimalgoritmo

caso 1
limpatela
comp <- 1
para i de 1 ate 3 faca
   para j de 1 ate 3 faca
      cont <- cont+1
      jog[i,j] <- NumpCarac(cont)
   fimpara
fimpara
```

```
273
274    ate fim = 1 entao
275    se fim = 1 entao
276       escreval ()
277       escreval ("VITORIA!")
278       vit <- vit +1
279    fimse
280
281    se fim = 0 entao
282       escreval ()
283       escreval ("EMPATE!")
284       ep <- ep + 1
285    fimse
286
287    limpatela
288    interface()
289    escreval ()
290    escreval (" PARABENS VOCÊ VENCEU!")
291 fimescolha
292 fimescolha
293 fimalgoritmo
```

CONCLUSÃO

Para concluir o trabalho aqui deixo as considerações finais, dificuldades e possíveis melhorias que poderiam ser feitas.

As dificuldades algumas já descrevi acima, como escolha no uso de funções e variáveis na estrutura do código, entendimento de algumas funções como procedimento e matriz, dificuldade para encontrar o erro que levava o *cont* a não resetar os números (por incrível que pareça), outra dificuldade foi fazer a numeração para a contagem de vitorias.

Para possíveis melhorias talvez achar alguma maneira que eu não precisasse estender o código para quase mais de 290 linhas, pois eu usei 2 estruturas de *escolha* para dar reset no jogo, e diminuir a complexidade para posicionar as peças, e configurar os resultados de vitória no jogo da velha.

FIM