INSTITUTO FEDERAL CATARINENSE

CIÊNCIA DA COMPUTAÇÃO

***CAMPUS* VIDEIRA**

VITOR CARLET

ALGORITMO QUE IDENTIFICA SE O CÓDIGO GENÉTICO É DE UM SÍMIO OU DE UM HUMANO

Videira-SC

INTRODUÇÃO

A ideia base deste código é identificar uma sequência de código independente do tamanho da matriz, para isso utilizei uma função para ler a quantidade de letras, uma função para escrever as letras, uma função para corrigir as letras, e as funções para identificar os símios, no final do algoritmo, ele dá o resultado de se o código é pertencente a um símio ou humano.

2022

RELATÓRIO DE CÓDIGO - VITOR CARLET

```
int tTabela(){
    int tam, erro;
    int certo = 1;
    printf("Olá, digite a quantidade de caracteres que contem o seu DNA\n");
    scanf("%i", &tam);
        do{
            if(tam != 0 && tam >=16){
                for(int r = 0; r < tam; r++)
                {
                    if(r * r == tam)
                    {
                        certo = 0;
                    }
                }
            }
        }
    while (certo == 1 && tam > 0);
    return tam;
}
```

Inicialmente, criei uma função para o usuário não informar valores incorretos que podem acarretar erros no funcionamento do código .

```c
void guarda (int tam, char seq[tam]) // seq[36] cada letra em um quadrado
    {
      if (tam > 0)
        {
          printf("Esse algoritmo só consegue detectar diferenças entre A,G,T,C\n");
          printf ("\nPor favor insira a sequencia de DNA para verificacao!\n");
          scanf ("%s", seq);
        }
    }
```

Essa função inicia a apresentação do código ao usuário e pede a sequência em caracteres do código genético .

```c
int procuraRaiz (int tam)//pra construir a matriz necessitamos da raiz do numero
    {
      int raiz = sqrt (tam);//raiz de tam
      return raiz;
    }
```

Função simples para encontrar a raiz.

```c
void verificaLetras (int tam, char seq[tam]){ // verifica se a quantidade de letras e se as letras estao corretas
  int tTabela, erro;
  if (tam > 0)
    {
     do
    {
     tTabela = strlen (seq);
     erro = 0; // se o erro for 1 ele volta a função, se o erro for 0 ele passa dessa funcao
         if (tTabela != tam)
               {
                printf
               ("\nA sequencia ideal para ser verificada deve conter apenas %i letras",
                tam);
                printf ("\nPor favor digite ela novamente!\n\n");
                scanf ("%s", seq);
               }
         for (int i = 0; i < tam && tTabela == tam; i++)
               {
                if (seq[i] != 'a' && seq[i] != 't' && seq[i] != 'c'
                && seq[i] != 'g' && seq[i] != 'A' && seq[i] != 'T'
                && seq[i] != 'C' && seq[i] != 'G')
                      {
                       erro = erro + 1;
                       printf
                         ("\nHC! um erro na %i letra, (%c), ela nao faz parte do conjunto de DNA!\n",
                          i + 1, seq[i]);
                      }
               }
         if (erro > 0)
               {
                     printf("\nA sequencia ideal para ser verificada deve conter apenas as letras A, T, C e G!");
                     printf ("\nPor favor digite ela novamente!\n\n");
                     scanf ("%s", seq);
               }
    }
     while (tTabela > tam || erro > 0); //enquanto estiver errado vai voltar ao inicio dessa funcao
    }
}
```

Essa função chama o int tam e o vetor char seq para verificar se a quantidade de letras corresponde ao tam () e testar se as letras estão de acordo com o funcionamento do código.

Se não estiverem corretos, a variável erro não vai conseguir atribuir valor 1 e o código irá

se repetir até você acertar os valores .

```c
void escreve(int tam){
    printf("esse eh o tam:%i\n", tam);
}// verificar se o tam está correto
```

verificar se o tam está correto.

```c
void seqEmMaisculo (int tam, int raiz, char seq[tam], char dna[][raiz])//seq em maiusclo
{
    // transforma minusculos em maiusculos
  if (tam > 0)
        {
          printf ("\n\nA sequencia escrita foi:\n");//mostra a sequencia
          int d = 0;
              for (int i = 0; i < raiz; i++)
                {
                  for (int j = 0; j < raiz; j++, d++)
                        {
                          dna[i][j] = seq[d];
                          if (dna[i][j] == 'a' || dna[i][j] == 't' || dna[i][j] == 'c'
                          || dna[i][j] == 'g')
                                {
                                  dna[i][j] = toupper (dna[i][j]);
                                }
                          printf ("%c\t", dna[i][j]);
                        }
                  printf ("\n");
                }
        }
}               //mostra a seq
```

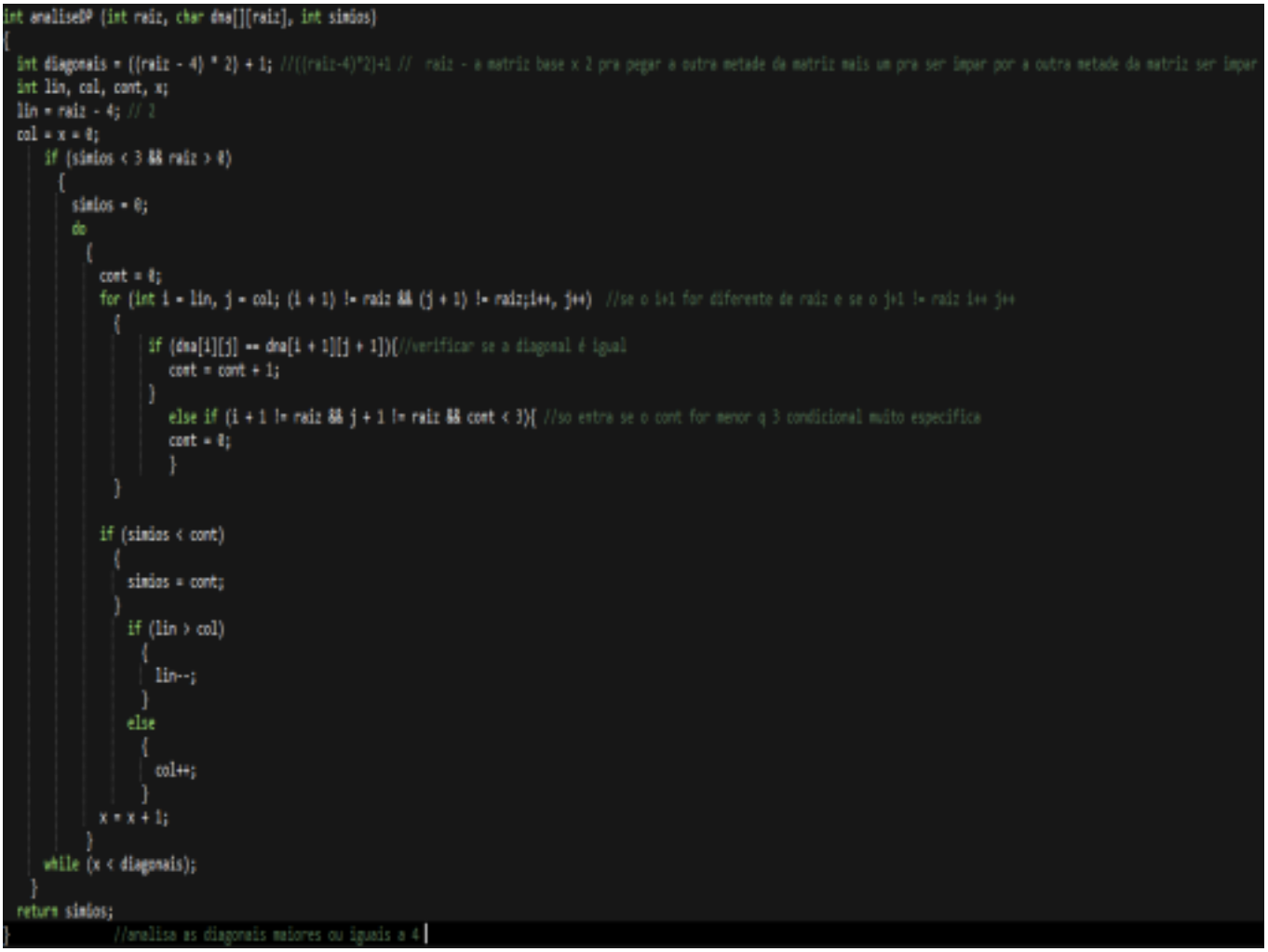
Essa função transforma os caracteres minúsculos em maiúsculos e depois de corrigidos, printa a matriz na tela.

```c
int analiseLinhas (int raiz, char dna[][raiz])
{
  int simios = 0;
  for (int i = 0; i < raiz && raiz > 0 && simios < 3; i++)
        {
          for (int j = 0; j < raiz && simios < 3; j++)
            {
              if (dna[i][j] == dna[i][j + 1])
                {
                  simios = simios + 1;
                  printf("%c",dna[i][j], dna[i][j + 1]);
                }
              else if (j + 1 != raiz && simios < 3)
                {
                  simios = 0;
                }
            }
        }
  return simios;
}               //analisa as linhas
```

Analisa as linhas aumentando apenas as colunas e somando se for o mesmo caracter,
não soma na mesma coluna porque se j+1 for diferente de raiz, mas os símios ainda forem
menores que 3, ele zera novamente.

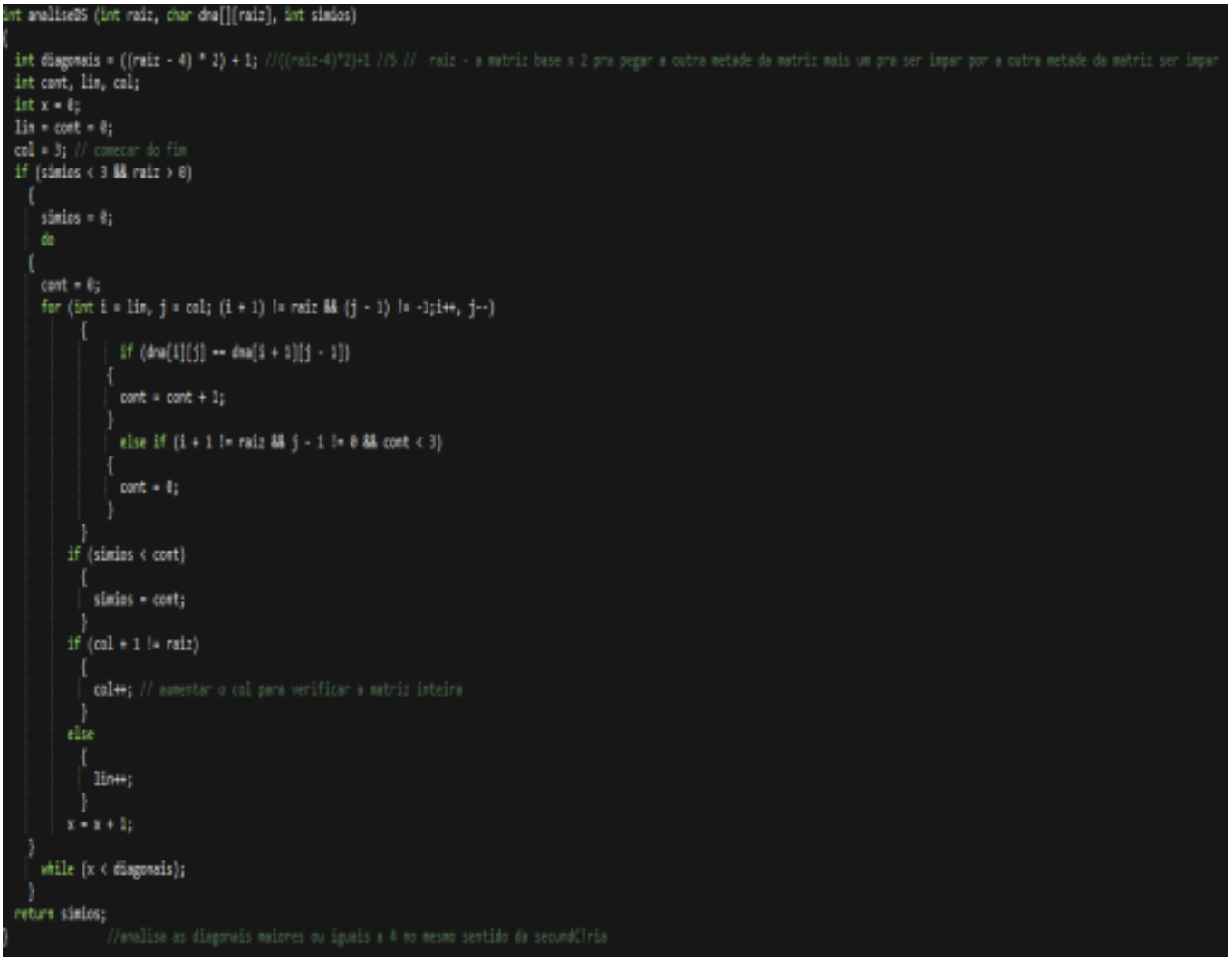
```
int analiseCol (int raiz, char dna[][raiz], int simios)
    {
      for (int i = 0; i < raiz && raiz > 0; i++)
            {
              for (int j = 0; j < raiz && simios < 3; j++)
                {
                  if (dna[j][i] == dna[j + 1][i])
                    {
                      simios = simios + 1;
                      printf("%c",dna[i][j], dna[j + 1][i]);
                    }
                  else if (j + 1 != raiz && simios < 3)
                    {
                      simios = 0;
                    }
                }
            }
      return simios;
    }               //analisa as colunas
```

Analisa as colunas aumentando apenas as linhas e somando se for o mesmo caractere,
não soma na mesma coluna porque se j+1 for diferente de raiz, mas os símios ainda forem
menores que 3, ele zera novamente.



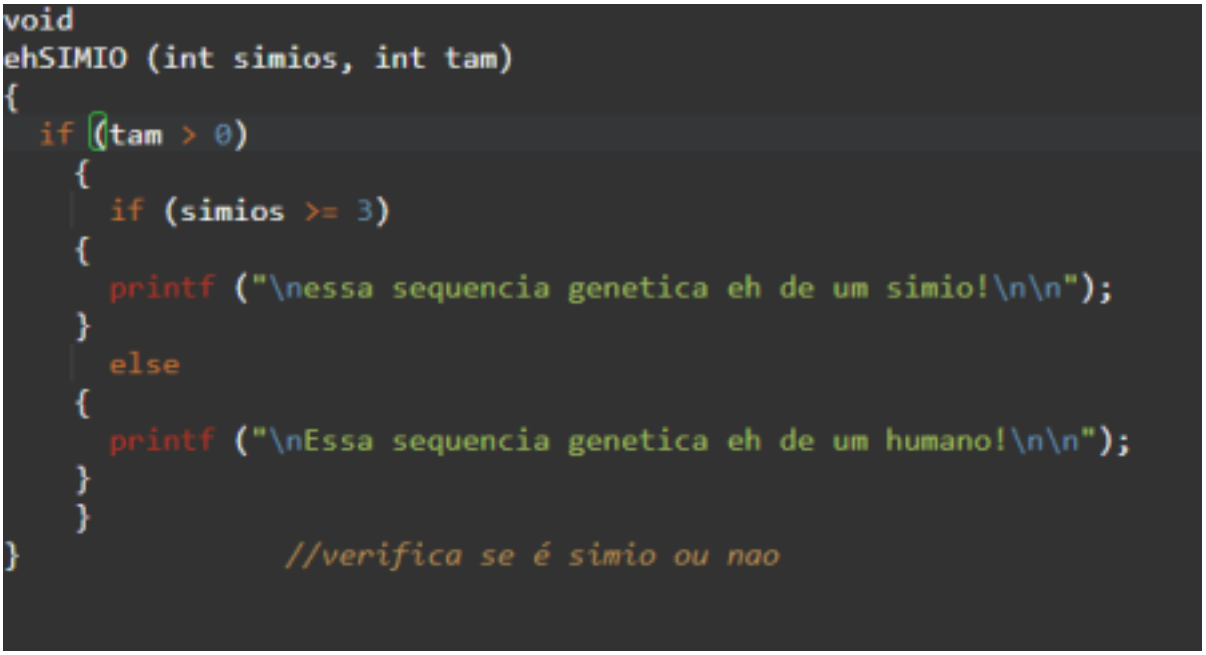
```
int analiseDP (int raiz, char dna[][raiz], int simios)
{
  int diagonais = ((raiz - 4) * 2) + 1; ///((raiz-4)*2)+1 //  raiz - a matriz base x 2 pra pegar a outra metade da matriz mais um pra ser impar por a outra metade da matriz ser impar
  int lin, col, cont, x;
  lin = raiz - 4; // 2
  col = x = 0;
     if (simios < 3 && raiz > 0)
       {
         simios = 0;
         do
           {
             cont = 0;
             for (int i = lin, j = col; (i + 1) != raiz && (j + 1) != raiz;i++, j++)  //se o i+1 for diferente de raiz e se o j+1 != raiz i++ j++
               {
                   if (dna[i][j] == dna[i + 1][j + 1]){//verificar se a diagonal é igual
                      cont = cont + 1;
                   }
                      else if (i + 1 != raiz && j + 1 != raiz && cont < 3){ //so entra se o cont for menor q 3 condicional muito especifica
                      cont = 0;
                      }
               }

             if (simios < cont)
               {
                 simios = cont;
               }
                 if (lin > col)
                   {
                     lin--;
                   }
                 else
                   {
                     col++;
                   }
             x = x + 1;
           }
       while (x < diagonais);
    }
  return simios;
}            //analisa as diagonais maiores ou iguais a 4
```

Analisa a diagonal principal basicamente aumentado os 2 inteiros (i,j) sequencialmente no mesmo for, tem a condicional para não acumular entre as diagonais, têm a condicional aumentar o lin com matrizes maiores que 4 e utiliza o inteiro diagonal que vai fazer repetir o do enquanto
ele não for atingido ao máximo.



```
int analise2S (int raiz, char dna[][raiz], int simios)
{
 int diagonais = ((raiz - 4) * 2) + 1; //((raiz-4)*2)+1 //5 //  raiz - a matriz base x 2 pra pegar a outra metade da matriz mais um pra ser impar por a outra metade da matriz ser impar
 int cont, lin, col;
 int x = 0;
 lin = cont = 0;
 col = 3; // comecar do fim
 if (simios < 3 && raiz > 0)
   {
    simios = 0;
    do
   {
    cont = 0;
    for (int i = lin, j = col; (i + 1) != raiz && (j - 1) != -1;i++, j--)
          {
              if (dna[i][j] == dna[i + 1][j - 1])
             {
              cont = cont + 1;
             }
              else if (i + 1 != raiz && j - 1 != 0 && cont < 3)
             {
              cont = 0;
             }
          }
        if (simios < cont)
          {
           simios = cont;
          }
        if (col + 1 != raiz)
          {
           col++; // aumentar o col para verificar a matriz inteira
          }
        else
          {
           lin++;
          }
        x = x + 1;
   }
    while (x < diagonais);
   }
 return simios;
}            //analise as diagonais maiores ou iguais a 4 no mesmo sentido da secundÁria
```

analisa a diagonal secundária decrementando a última coluna até chegar na primeira e vai aumentando a linha para acompanhar esse processo, utiliza as mesmas ideias da primeira coluna a única diferença é que se a matriz for maior que 4, ele vai incrementando o col até a raiz.



```
void
ehSIMIO (int simios, int tam)
{
  if (tam > 0)
    {
      if (simios >= 3)
    {
      printf ("\nessa sequencia genetica eh de um simio!\n\n");
    }
      else
    {
      printf ("\nEssa sequencia genetica eh de um humano!\n\n");
    }
    }
}               //verifica se é simio ou nao
```

função para apresentar ao usuário o resultado de sua pesquisa através do algoritmo.

```c
int main(void)
{
    int tam, raiz, simio, escrever, testes = 0;
      do
        {
          tam = tTabela ();
          raiz = procuraRaiz(tam);
          char seq[tam], dna[raiz][raiz];
          guarda (tam, seq);
          verificaLetras (tam, seq);
          seqEmMaisculo (tam, raiz, seq, dna);
          simio = analiseLinhas(raiz, dna);
          simio = analiseCol(raiz, dna, simio);
          simio = analiseDP(raiz, dna, simio);
          simio = analiseDS (raiz, dna, simio);
          ehSIMIO(simio,tam);



          escreve(tam);

        }
  while (tam != 0);

  printf ("voce apertou 0 então acabamos por aqui!");

  return 0;
}
```

Int main modularizado com todas as funções apresentadas.

• Então esse código simples pede para o usuário informar os valores; • Apresenta aviso de erro;
• Encontra a raiz;
• Função para verificar se as letras estão corretas e formata-las para maiúsculo;
• Função para verificar se o tam está correto;
• Função para análise de linhas utilizando apenas condicionais for; • Função para análise de colunas utilizando apenas condicionais for; • Analisa raízes independente do tamanho;
• Analisa diagonais tanto da direita como da esquerda; • Apresenta se o código genético do animal é Simio ou Humano.