TRABALHO FINAL DE ALGORITMOS

INSTITUTO FEDERAL CATARINENSE - CAMPUS VIDEIRA

CIÊNCIAS DA COMPUTAÇÃO

LUCAS SONEGO GOMES DE MAGALHÃES

Desenvolvimento do algoritmo

Primeiramente pedimos ao usuário para digitar o tamanho que ele quer que seja analisado, sendo esse tamanho maior que 16 e que possua raiz.

```
int
tamanho ()
{
 int tam, erro;
 int certo = 1;
 printf ("\ninforme um numero maior ou igual a  16 e que tenha raiz:\n");
 scanf ("%i", &tam);
```

Em seguida, o programa analisa o tamanho e verifica se é diferente de 0 e que seja maior ou igual a 16.

```
do
  {
    if (tam != 0 && tam >= 16)    // confere o tamnho do dna se e maior que 16
  {
    for (int r = 0; r < tam; r++)
      {
        if (r * r == tam)
      {
        certo = 0;
      }
      }
  }
  }
while (certo == 1 && tam > 0);
return tam;
```

Na próxima função é verificado a raiz do número escolhido pelo usuário e armazena ela para poder usar na verificação das diagonais e linha, coluna.

```
int
Raiz (int tam)
{
  int raiz = sqrt (tam);
  return raiz;
}
```

Temos uma função que faz uma solicitação para o usuário das letras A, T, C ou G sendo armazenadas em vetor.

```
void
armazena (int tam, char seq[tam])   // armazena os dados do dna A T C G
{
  if (tam > 0)
    {
      printf ("digite somente  A,G,T,C");
      printf ("\nA sequencia de dna para ser analisada:\n");
      scanf ("%s", seq);
    }
}
```

Logo após armazenar as letras digitadas pelo usuário fiz uma função que percorre todo o vetor e verifica a quantidade de letras digitadas pelo usuário para verificar se confere com a escolhida lá no início e caso não seja pede para o usuário digitar novamente, nessa parte é utilizado o comando strlen que percorre e informa a quantidade de carácter. Também na mesma função foi feito uma verificação do tamanho das letras digitadas pelo usuário se são maiúsculas ou minúsculas e se contém alguma outra que esteja fora do requerimento A, T, C ou G; Caso haja algum erro é informado para o usuário onde e qual é esse erro e solicita uma nova digitação.

```
void
verificaTamLet (int tam, char seq[tam]) // verificaC'C#o do tar
{
  int tamanho, erro;
  if (tam > 0)
    {
      do
    {
      tamanho = strlen (seq);    // utiliza a funC'C#o strlen pa
      erro = 0;
      if (tamanho != tam)    // verifica se o numero de letras d
        {
          printf ("\n a sequencia deve possuir somente  %i letr
          printf ("\ndigite novamente\n\n");
          scanf ("%s", seq);
        }
      for (int i = 0; i < tam && tamanho == tam; i++)    // perc
        {
          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 ("\n contem um erro na %iB* letra, (%c),\n",
              i + 1, seq[i]);
```

Essa próxima sequência é uma continuação da parte anterior onde utiliza-se toupper que vai receber os argumentos(caracteres) que será convertido para inteiro. Mesmo que **toupper**() receba um inteiro como argumento, o caractere é passado para a função. Internamente, assim esse caractere é convertido em seu valor ASCII para a verificação; assim transformando qualquer letra que o usuário tenha digitado minúscula em maiúscula.

```
void
mostraSeqemMaius (int tam, int raiz, char seq[tam], char dna[][raiz])
{

  if (tam > 0)
    {
      printf ("\n\n a sequencia do dna foi:\n");
      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");
```

A análise da linha é feita de forma simples sendo a raiz do número maior que zero e simions menor que 3, pois tem início na própria letra sendo por conta disso 3 para não ultrapassar o tamanho de 4 para confirmar um símio onde é feito um loop para tentar encontrar uma sequência caso seja encontrada simions recebe 1 e retorna positivo. Esse mesmo método de verificação se dá para as colunas.

```
int
analiseLin (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;
```

Já na análise das diagonais que possuem um esquema bem diferente, através de vídeos na internet e algumas pesquisas consegui fazer a verificação das matrizes tanto principal quanto secundária, através de muitos erro e ainda ficando difícil de entender o cálculo da raiz - 4 vezes 2 + 1 para poder analisar a matriz completa e não verificar a mesma coisa várias vezes com + 1 para não ficar par e aplicando o loop entra somente no else if o contador for menor que 3;tirando 1 para que se movimente a matiz, para não dar erro e pegar informações fora da área que precisa ser analisada. No fim retorna se é símio ou não.

```
int
analiseDP (int raiz, char dna[][raiz], int simios)  //analisa as diag
{
  int diagonais = ((raiz - 4) * 2) + 1;
  int lin, col, cont, x;
  lin = raiz - 4;
  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++)
        {
          if (dna[i][j] == dna[i + 1][j + 1])
        {
          cont = cont + 1;
        }
```

Por final temos uma função que vai verificar se algum símio das análises é é maior que 3 casos seja positiva dna simon, caso contrário humano.
Através desse resultado é possível chegar ao resultado solicitado pelo usuário de maneira mais simples possível.

```
void
isSimian (int simios, int tam)  // faz
{
  if (tam > 0)
    {
      if (simios >= 3)
    {
      printf (" e um simion");
    }
      else
    {
      printf (" e um humano");
    }
    }
}
```