programa

{

inclua biblioteca Arquivos --> A

inclua biblioteca Texto --> T

const inteiro quantidadeMutacao = 8

inteiro mutacao[quantidadeMutacao] , endereco

cadeia codons[quantidadeMutacao][2]

cadeia dna = "DNA1.txt"

cadeia d1 , d2 , linha

funcao logico verificaCodons (cadeia c1, cadeia c2)

{

inteiro indiceC1=0, indiceC2=0

cadeia matrizCodons[21][6]=

{

{"AUG"," "," "," "," "," "},//Metionina - INICIO

{"UGG"," "," "," "," "," "},//Triptofano

{"AAA","AAG"," "," "," "," "},//Lisina

{"AAU","AAC"," "," "," "," "},//Aspargina

{"CAA","CAG"," "," "," "," "},//Glutamina

{"CAU","CAC"," "," "," "," "},//Histidina

{"GAA","GAG"," "," "," "," "},//Acido Glutâmino

{"GAU","GAC"," "," "," "," "},//Acido Aspartico

{"UAU","UAC"," "," "," "," "},//Tirosina

{"UGU","UGC"," "," "," "," "},//Cisteina

{"UUU","UUC"," "," "," "," "},//FenilaLamina

{"AUU","AUC","AUA"," "," "," "},//Isoleucina

{"UAA","UAG","UGA"," "," "," "},//STOP

{"ACU","ACC","ACA","ACG"," "," "},//Treonina

{"CCU","CCC","CCA","CCG"," "," "},//Prolina

{"GCU","GCC","GCA","GCG"," "," "},//Alanina

{"GGU","GGC","GGA","GGG"," "," "},//Glicina

{"GUU","GUC","GUA","GUG"," "," "},//Valina

{"CGU","CGC","CGA","CGG","AGA","AGG"},//Arginina

{"UCU","UCC","UCA","UCG","AGU","AGC"},//Serina

{"UUA","UUG","CUU","CUC","CUA","CUG"} //Leucina

}

para(inteiro i=0; i<21; i++)

{

para(inteiro j=0; j<6; j++)

{

//Primeiro Codon

se(matrizCodons[i][j] == c1)

{

indiceC1 = i

}

//Segundo Codon

se(matrizCodons[i][j] == c2)

{

indiceC2 = i

}

}

}

se(indiceC1 == indiceC2)

{

retorne verdadeiro

}

senao

{

retorne falso

}

}//fim da funçâo

funcao cadeia transformarEmRna(cadeia baseN)

{

se(baseN == "A")

{

retorne "U"

}

senao se(baseN == "T")

{

retorne "A"

}

senao se(baseN == "C")

{

retorne "G"

}

senao se(baseN == "G")

{

retorne "C"

}

senao

{

retorne "A"

}

}

funcao acharMutacao()

{

endereco = A.abrir\_arquivo(dna, 0)

logico linhaEx = A.fim\_arquivo(endereco)

inteiro cont = 0 , contX = 0

inteiro nLinha = 0 , temFalha = 0

enquanto(linhaEx == falso)

{

temFalha = 0

para(inteiro i = 1 ; i <=3 ; i++)

{

linha = A.ler\_linha(endereco)

nLinha++

linhaEx = A.fim\_arquivo(endereco)

se(linhaEx == verdadeiro)

{

pare

}

d1 = T.extrair\_subtexto(linha, 0, 1)

d2 = T.extrair\_subtexto(linha, 2, 3)

se((d1 == "A" e d2 != "T") ou (d1 == "T" e d2 != "A") ou (d1 == "C" e d2 != "G") ou (d1 == "G" e d2 != "C"))

{

codons[contX][0] += transformarEmRna(transformarEmRna(d1))

codons[contX][1] += transformarEmRna(d2)

temFalha = 1

mutacao[cont] = nLinha

cont++

}

senao

{

codons[contX][0] += transformarEmRna(d2)

codons[contX][1] += transformarEmRna(d2)

}

linhaEx = A.fim\_arquivo(endereco)

}

se(temFalha == 0)

{

codons[contX][0] = ""

codons[contX][1] = ""

}

senao

{

se(verificaCodons(codons[contX][0] , codons[contX][1]))

{

escreva("\n\nSilenciosa")

}

senao

{

escreva("\n\nNão Silenciosa")

}

escreva("\nMutação na linha ", nLinha," ",d1,d2,"\n")

escreva("Codon experado :",codons[contX][0], "\nCodon :",codons[contX][1])

contX++

}

}

A.fechar\_arquivo(endereco)

}

funcao inicio()

{

acharMutacao()

}

}