EIE

Escuela de
Ingeniería Eléctrica

UNIVERSIDAD DE COSTA RICA

UNIVERSIDAD DE COSTA RICA

FACULTAD DE INGENIERIA

ESCUELA DE INGENIERIA ELÉCTRICA

ESTRUCTURAS ABSTRACTAS DE DATOS Y ALGORITMOS PARA INGENIERÍA

LABORATORIO 0:
INTRODUCCIÓN C++ Y PYTHON

ESTUDIANTES:
JESÚS ZÚÑIGA MÉNDEZ
DENNIS CHAVARRÍA SOTO

PROFESOR:
RICARDO ROMÁN BRENES; M. SC.

II CICLO 2019

Índice

Índice de figuras

1. Introduccion.

El objetivo primordial del primer laboratorio del curso de Estructuras Abstractas y Algoritmos para Ingeniería es el de reintegrar y poner en práctica los conocimientos de los lenguajes de programación Python y C++ de los estudiantes, como una práctica inicial para recordar la sintaxis, lógica y funciones que pueden aplicarse para cumplir diferentes propósitos. La meta es crear un programa en cada uno de los lenguajes antes mencionados, el cuál debe cumplir con la misma función, solicitar una cadena de ARN y compararla para determinar cuáles son las proteínas constituyentes.

Para desarrollar ambas versiones, se tomó como punto de partida la idea de utilizar una base de datos que contiene toda la información necesaria, de este modo, el programa podría ser más compacto y eficiente, pues, no tendría que desarrollarse una gran cantidad de condicionales ni analizar múltiples patrones, sino, simplemente, dejar que el proceso de automatización se encargue de analizar la información proporcionada.

2. Funcionamiento del programa.

2.1. Python

Para este laboratorio, se solicitó crear un programa que solicite una cadena de ARN y este se capaz de dividir la cadena en trios de letras, las cuáles, según una determinada tabla de codones, pueden ser ordenadas de formas específicas que corresponden a una proteína. Usando esta premisa, se construye una ”base de datos”llamada ARN Table, que contiene 3 columnas, la primera corresponde a un codón, la segunda a la abreviatura del nombre que recibe tal aminoácido; y la tercera, al nombre en cuestión. Tronco del programa o Main Lo primero que se establece es que se creará un arreglo que pueda contener el codón, para ello, se pensó en que se debe solicitar la cadena de ARN y tomar cada una de las letras de esta y agregarlas al arreglo, no obstante, como solo se pueden comparar tres letras, lo que se crea es un ciclo que repita de 0 a 3, iterando entre las letras de la cadena, pero considerando que se tiene una variable ordenador que se encarga de pasar de codon en lo que introdujo el usuario, así, al haber copiado las letras para uno de estos, esta variable itera y se pueden copiar lo siguiente según corresponde por medio de .ªrreglo.apend()”.

Anteriormente se explico que se utiliza un ciclo que tiene un rango y permite iterar entre letras, efectivamente, sin embargo, para que pueda recorrer completamente lo que introdujo el usuario, es necesario otro ciclo. Primeramente se obtiene el módulo de la cantidad de letras que tiene la cadena de ARN introducida, de este modo, si es diferente de 0 se lanza un mensaje de error y se cierra el programa, por otro lado, si la condición anterior se cumple, se procede a dividir la longitud entre 3, este corresponde a la cantidad de veces que el çiclo exterior”deberá repetir aquel que obtiene las letras de la entrada para añadirlas al codón.

Ya que se comprende el funcionamiento de los ciclos en el programa, resalta cuando se invoca la función Çomparator”. Esta, junto con el ”borrado”del contenido de codon ocurren justo antes de repetir el çiclo exterior”. Funcion comparator Cuando se invoca la función comparator, también se pasa un parámetro que tiene las tres letras que deben ser comparadas. Lo primero es abrir el archivo .ªRN TABLE.TXT”que contiene todos los codones que deberán ser comparados. A continuación, se recurre nuevamente a dos ciclos, uno exterior para iterar por cada una de las lineas de la base de datos y otro, que será un ciclo interior que compara las letras del parámetro con las de la primera columna de la línea obtenida de la base de datos. La lógica tras esta función es que al salir de estas 3 repeticiones, si la variable llamada counter (la cual suma un punto por cada letra coincidente) tiene un valor de 3, entonces se procede a un condicional, en el caso anterior se imprime la tercera columna obtenida de la base de datos, lo cual corresponde, como se mencionó al inicio, al nombre del aminoácido. En la situación en la cual se termina el ciclo exterior, pues, no hubo coincidencias, se lanza el mensaje de que el codón no corresponde a ninguno de la lista.

2.2. C++

Para resolver el problema en c++ se utilizo una función la cual recorre un archivo de texto linea por linea donde esta guardada la información de cada codón, una vez comparada y encontrado el resultado esta devuelve la letra correspondiente, por ultimo se imprime la cadena de caracteres correspondiente a toda la cadena de codónes que se envió.

3. Conclusiones.

Después de desarrollar la primera version de este programa se pueden obtener las siguientes conclusiones.

- Fue posible implementar un programa en los lenguajes de programación, Python y C++,para determinar el nombre de los codones constituyentes de una cadena de ARN.
- Se logró, de manera satisfactoria, utilizar los conocimientos fundamentales de los cursos anteriores de programación, para desarrollar los algoritmos solicitados para la práctica de laboratorio.
- Los programas fueron desarrollados de la forma propuesta en grupo, mediante el uso de una base de datos, permitiendo, así, reducir la extensión del código.

4. Bibliografía

Referencias

5. Anexos

5.1. C++

5.1.1. main.cpp

"main.cpp"

```cpp
/**
 * @file main.c
 * @author Jesus Zuñiga Mendez
 * @brief Archivo pricipal, laboratorio que permite refrescar conceptos basicos
 * @version 1.0
 * @date 26 de agosto de 2019
 * @copyright Copyleft (l) 2019
 */
#include "./include/includes.h"

using namespace std;
/**
 * @brief Función que imprime una bienvenida para el usuario
 */
void Bienvenida(){
    printf("  _____    _                                          _       _           _  _  _
    \n |  _ \\ (_)                                        (_)     | |         | || || |
    \n | |_) | _   ___   _ __ __    __ ___   _ __    _   __| |  ___    | || || |
    \n |  _ < | | / _ \\| '_ \\\\ \\ / // _ \\| '_ \\  | | / _` | / _ \\   | || || |
    \n | |_) || || ___/| | | |\\ V /|  __/| | | || || (_| || (_) |  |_||_||_|
    \n |____/ |_| \\___||_| |_| \\_/  \\___||_| |_||_| \\__,_| \\___/   (_)(_)(_)\n");
    printf("\n");
    printf("\n");
}

/**
 * @brief Funcion Principal
 */
int main(int argc, char** argv){
    Bienvenida();
    string codon;
    string analizar = "";
    string resultado = "";
    if (argc >1){
        for (int i =1;i < argc; i++){
            analizar =  argv [i];
            if ((analizar.length() % 3) == 0){
                for (unsigned int i =0; i< analizar.length(); i++){
                    codon = codon + analizar[i];
                    if (((i+1) % 3) == 0){
                        resultado = resultado + CompararProteina(codon);
                        codon = "";
                    }
                }
                cout << resultado << endl;
                resultado = "";
            }else{
```

```
48                  cout << "la lista " << analizar <<
49                  " no tiene una extension correcta" << endl;
50              }
51          }
52      }
53      return 0;
54  }
```

5.1.2. tools.cpp

"tools.cpp"

```cpp
#include "../include/includes.h"

using namespace std;
/*@brief Funcion que lee un archivo linea por linea y compara la
  proteina recibida contra la del archivo
 *@param proteina es la proteina que se debe comparar
 *@return devuelve la letra correspondiente a la proteina
 */
string CompararProteina(string proteina){
    ifstream archivo;
    string ruta= "./DB/listaAminoacidos";
    string linea;
    string resultado = "";
    archivo.open(ruta.c_str(),ios::in);
    if (archivo.fail()){
        cout << "No se pudo abrir el archivo " < endl;
        exit(1);
    }

    while (!archivo.eof()){
        getline(archivo,linea);
        char coma = ',';
        string aminoacido = "";
        string letra = "";
        string nombre = "";
        int contadorComas = 0;
        for (unsigned int i = 0; i < linea.length() ; i++){
            if (linea[i] == coma){
                contadorComas++;
            }else{
                if (contadorComas == 0){
                    aminoacido = aminoacido + linea[i];
                }else if (contadorComas == 1){
                    letra = letra + linea[i];
                }else if (contadorComas == 2){
                    nombre = nombre + linea[i];
                }
            }
        }
        if (strcmp(aminoacido.c_str(), proteina.c_str()) == 0){
            resultado = letra;
        }
    }

    archivo.close();
    return resultado;
}
```

5.1.3. Includes.h

Ïncludes.h"

```
#ifndef INCLUDES_H
#define INCLUDES_H
    #include <iostream>
    #include <cstring>
    #include <fstream>

    std::string CompararProteina(std::string);
#endif
```

5.2. Python

5.2.1. pricipal.py

"Principal.py"

```python
#@file ARN_Comparator.c
#@Author Dennis Chavarria
#@date 22/8/2019
#@brief Programa con capacidad de comparar una cadena de ARN introducida,
# con una base de datos, para asi, devolver el nombre de las tripletas que la componen


from numpy import array
#Print("Dasio Software Solutions (C)\n")

#@brief Comparator: Esta funcion debe comparar el codon con el elemento de la primera
#  columna de el archivo ARN_TABLE y devolver un resultado sobre si corresponde a una
# proteina conocida o no
#@param Codon corresponde a tres letras, de la linea introducida al inicio.
#  Este parametro es el que se compara con los datos de la tabla
#@return 0 De este modo sale de la funcion sin tener que anadir mas logica al programa

def comparator(codon):
    file=open("ARN_TABLE.txt", "r")
    counter=0
    print(codon)
    for j in range (0,64):
        file_row=file.readline()
        line_elements=file_row.split(';')
        for i in range (0,3):
            if codon[i]==line_elements[0][i]:
                counter+=1
        if counter==3:
            print("El codon anterior corresponde a: %s" %line_elements[2])
            return(line_elements[1])  #This return is required because, by this, we dont
        need to add extra lines if the codon matches with any of the list.
        else:
            counter=0
    print("El codon %s no corresponde a ningun aminoacido" %codon)
    file.close()
#@brief Main: Esta parte de el codigo constituye el eje central y se encarga de solicitar
#  la cadena de ARN e invocar la foo Comparator. Ademas, determina si la cadena es analizal


arn=input("Introduzca la cadena de ARN\n")
arn_row=arn.upper()
print("\n")
arn_verifier=len(arn_row)%3
codon=[]
codon_translate=[]

if (arn_verifier == 0):
    ordenador=0
    repeater=int(len(arn_row)/3) #remember, it starts at zero
```

```
    for i in range (0, repeater): #Ciclo para estudiar toda la fila
        for i in range (0, 3): #Ciclo para crear el codon por analizar
            codon.append(arn_row[i+(ordenador*3)])
        ordenador+=1
        codon_translate.append(comparator(codon))
        codon=[]
    print("\nEntonces, la cadena es %s" %codon_translate)
else:
    print("Debe introducir una cadena con una cantidad de letras que sea multiplo de 3")
```