Análisis de Genes

Ejercicio N° 1

Objetivos	● Buenas prácticas en programación de Tipos de Datos Abstractos (TDAs) ● Modularización de sistemas ● Correcto uso de recursos (memoria dinámica y archivos) ● Encapsulación y manejo de Sockets
Instancias de Entrega	**Entrega 1:** clase 4 (04/04/2017). **Entrega 2:** clase 6 (18/04/2017).
Temas de Repaso	● Uso de structs y typedef ● Uso de macros y archivos de cabecera ● Funciones para el manejo de archivos en C ● Operadores bitwise ● Funciones para el manejo de Sockets
Criterios de Evaluación	● Criterios de ejercicios anteriores ● Cumplimiento de la totalidad del enunciado del ejercicio ● Ausencia de variables globales ● Ausencia de funciones globales salvo los puntos de entrada al sistema (*main*) ● Correcta encapsulación en TDAs y separación en archivos ● Uso de interfaces para acceder a datos contenidos en TDAs ● Empleo de memoria dinámica de forma ordenada y moderada ● Acceso a información de archivos de forma ordenada y moderada

Índice

Introducción

Descripción

Formato de Línea de Comandos
- Cliente
- Servidor

Códigos de Retorno

Entrada y Salida Estándar

Protocolo de Comunicación
- Cliente hacia Servidor
- Servidor hacia Cliente

Ejemplos de Ejecución
- Servidor
- Cliente
- Ejemplo 1
- Ejemplo 2

Restricciones

Referencias

Introducción

Como parte de un sistema distribuido para un laboratorio de biología genética, se nos pide desarrollar un programa en C que analice ácidos nucleicos diciéndonos a qué aminoácidos se van a traducir.

Descripción

Las cadenas de *ácido desoxirribonucleico* (*ADN*) presentes en el núcleo celular contienen la información necesaria para producir las proteínas funcionales para la vida.
Dentro del *ADN* de las células eucariotas hay secuencias de bases nitrogenadas que comienzan con una cadena promotora, siguen con los genes, y terminan con una serie de finalización.
Durante la **transcripción**, se genera una secuencia de *ácido ribonucleico* (*ARN*), cuyas bases nitrogenadas son complementarias a las del *ADN*. Estas cadenas de *ARN* tienen secciones llamadas *exones*, que se traducirán en un aminoácido, y otras llamadas *intrones*, que son eliminadas durante un proceso llamado *splicing*, que sucede durante la maduración del *ARN* (esta maduración es necesaria para que el ácido nucleico salga del núcleo).
Para los laboratoristas que nos encargan el software, es simple deducir esa cadena madura de *ARN* a partir del *ADN* original, y por ahora no necesitan automatizar esa deducción.
Nosotros partiremos de un archivo generado por los laboratoristas, que contiene caracteres que simbolizan

los cuatro tipos de base nitrogenada presentes en una cadena de *ARN* maduro (llamado *ARN mensajero*), y trabajaremos codificando esos caracteres en 2 bits cada uno:

Caracter	Base nitrogenada	Codificación en bits
A	Adenina	00
U	Uracilo	01
G	Guanina	10
C	Citocina	11

Así es que, por ejemplo, cada cadena 'AGC' presente en el archivo, la codificaremos como 001011 en binario.

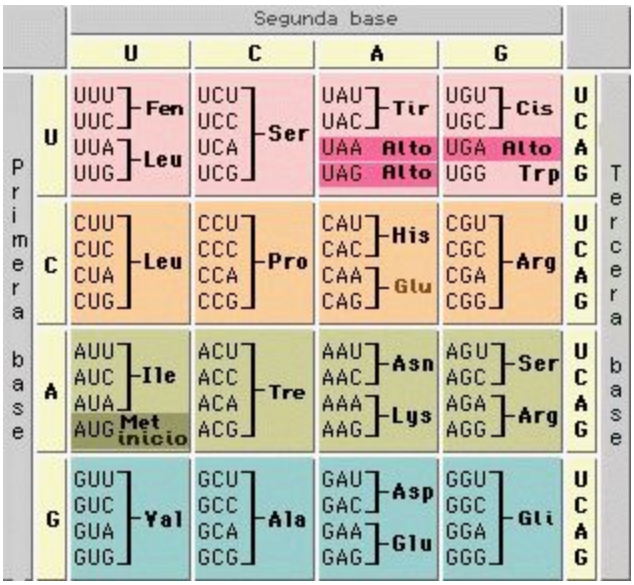
Cada secuencia de tres bases nitrogenadas (*codón*) presente en el *ARN* maduro, se va a traducir en un aminoácido. El mapeo de cada *codón* a su respectivo aminoácido es lo que se conoce como *código genético*.

En nuestro software nos interesará analizar cuáles son los aminoácidos más frecuentes generados a partir del archivo provisto por los laboratoristas.

Como nos interesa en un futuro dejar un análisis estadístico centralizado en un servidor, deberemos desarrollar un sistema distribuido, que en la versión requerida actualmente tendrá sólo dos procesos secuenciales:

1) El proceso 'cliente', que se encargará de codificar el archivo provisto y enviar la información codificada al otro proceso;
2) Y el proceso 'servidor', cuya responsabilidad será recibir la información codificada, contar cuáles son los *codones* más frecuentes, y responderle al proceso cliente una lista de aminoácidos correspondientes, según el código genético.

El código genético es el siguiente:



Código genético

Cada codón está compuesto por tres bases. Para interpretar la tabla previa se toma la primer base para seleccionar en qué fila de dicha tabla hay que buscar. Luego se toma la segunda base para buscar la columna. Y finalmente, dentro del cuadrante seleccionado se busca por la tercer base y así se obtiene el aminoácido que dicho codón genera.

Donde los nombres de los aminoácidos están abreviados:

Abreviación	**Nombre completo**
Fen	Fenilalanina
Leu	Leucina
Ser	Serina
Tir	Tirosina
Cis	Cisteína
Trp	Triptófano
Pro	Prolina
His	Histidina
Glu	Ácido glutámico
Ile	Isoleucina

Abreviación	**Nombre completo**
Met	Metionina
Tre	Treonina
Asn	Asparagina
Lys	Lisina
Arg	Arginina
Val	Valina
Ala	Alanina
Asp	Ácido aspártico
Gli	Glicina

Notar que hay codones de **Alto** (también conocidos como de **stop**). Estos codones no generan ningún aminoácido sino que marcan el fin de una cadena de codones. Esta cadena de codones genera una cadena de aminoácidos llamada proteína.
Por lo que es razonable pensar que por cada codón de tipo Alto se genera una proteína, de ahí que contar dichos codones es equivalente a contar la cantidad de proteínas que se generan.

Tomemos como ejemplo el codón UAC. La primer base, U, nos indica que hay que buscar el aminoácido en la primer fila de la tabla mientras que la segunda base, A, nos señala la tercer columna.
Dentro del cuadrante seleccionado vemos que hay cuatro codones: UAU, UAC, CAA y CAG. La tercer base, C, nos indica cual de esos cuatro codones es el correcto y la tabla indica que el codón UAC genera el aminoácido Tir, que es la abreviación de Tirosina.

Formato de Línea de Comandos

Cliente

Al proceso cliente lo invocaremos con la siguiente línea:

```
./tp client <ip-del-servidor> <puerto-del-servidor> <nombre-de-archivo>
```

Donde el nombre del archivo será el del provisto por los laboratoristas.

Servidor

El proceso servidor será invocado de la siguiente manera:

```
./tp server <puerto-del-servidor>
```

Códigos de Retorno

En cualquier caso, tanto el servidor como el cliente retornarán 0.

Entrada y Salida Estándar

- Ninguno de los procesos recibirán nada por entrada estándar.
- Sólo el cliente imprimirá por salida estándar la respuesta del servidor, mientras que el servidor no imprime nada.

Protocolo de Comunicación

Cliente hacia Servidor

Cada base nitrogenada A, U, G y C (cada caracter leído del archivo de entrada) se traducirá en 2 bits, por lo que cada codón (compuesto de 3 bases nitrogenadas) lo podremos codificar en 6 bits, y por ende “cabrá” en 1 byte. Dado que estamos trabajando con bytes de 8 bits, tomaremos como convención poner los 2 primeros bits en cero para luego poner los siguientes 6 bits la codificación del codón.

Por ejemplo, dado el codón ‘ACG’ en el archivo quedará codificada en un solo byte ‘00001011’, mientras que el codón ‘UAG’ se codificará como ‘00010011’.

Los bytes resultantes de la codificación de todos los codones serán los que se enviarán desde el proceso cliente al proceso servidor usando el protocolo TCP/IP. (Ver ejemplos de ejecución para mayor claridad).
Luego de enviar todos los codones presentes en el archivo de entrada, el cliente debe cerrar la conexión TCP en el sentido hacia el servidor tal que el cliente no podrá enviar más bytes hacia el servidor pero nada le impedirá recibir bytes desde este.
De hecho, el cliente deberá quedar a la espera de una respuesta proveniente del servidor.

Servidor hacia Cliente

El servidor, luego de recibir los codones codificados, le responderá al cliente un texto con los tres aminoácidos más frecuentes, termina la conexión TCP y finaliza el programa.

No se requiere que el servidor sea capaz de aceptar ni procesar a múltiples clientes a la vez. El servidor solo aceptara y procesara a un único cliente.

El formato de la respuesta es el siguiente (el servidor responde en modo texto, cada byte es un caracter):

```
Cantidad de proteínas encontradas: <número_de_codones_de_stop>

Aminoácidos más frecuentes:
1) <nombre_aminoácido_1>: <cantidad_aminoácido_1>
2) <nombre_aminoácido_2>: <cantidad_aminoácido_2>
3) <nombre_aminoácido_3>: <cantidad_aminoácido_3>
```

Casos especiales:

- Si existen menos de tres aminoácidos traducidos desde la cadena dada, devolver tantos como haya presentes.
- Los empates en cantidad se resuelven alfabéticamente por nombre de aminoácido.
- Tener en cuenta que hay varios codones que se traducen en los mismos aminoácidos, y que lo que se debe contar es la cantidad de aminoácidos.
- Para unificar criterios, tomaremos el número de codones de **Alto** como la cantidad de proteínas.

Ejemplos de Ejecución

Servidor

Se nos ha pedido una versión preliminar en la que el servidor:

1) Recibe por línea de comandos el puerto en el que debe escuchar una única conexión.
2) Acepta un cliente.
3) Lee todos los datos que éste le envíe (hasta que el cliente cierre la conexión en ese sentido) procesándolos.
4) Contesta por la misma conexión con la cantidad de proteinas (cantidad de codones **Alto**) y el ranking de los tres aminoácidos que más se van a generar a partir de los codones recibidos.
5) Termina su ejecución.

Cliente

Durante su ejecución, el cliente:

1) Recibe por línea de comandos el par ip:puerto al que se debe conectar.
2) Lee de un archivo de texto caracteres ‘A’, ‘G’, ‘C’ y ‘U’, que representan las cuatro bases nitrogenadas posibles en una cadena de ARN.
3) Envía los distintos codones codificados al servidor, según el protocolo de comunicación.
4) Cierra el modo escritura de la conexión.
5) Recibe la respuesta del servidor y la imprime por salida estándar.
6) Finaliza la ejecución.

Ejemplo 1

Tenemos el archivo **arn.in**, cuyo contenido es la cadena: 'AUGAUUACCAUUACCAUUACCGCUUGA'.
Ejecutamos el servidor con:
./tp server 8080
Luego ejecutamos el cliente con:
./tp client 127.0.0.1 8080 arn.in

El cliente deberá leer el archivo y enviar a medida que lo lee los distintos codones (ternas de bases nitrogenadas) codificados al servidor.

En este caso deberá enviar los siguientes **9 bytes**, cuya codificación se muestra a continuación en binario y debajo las correspondientes bases para mayor claridad:

```
00000110 00000101 00001111 00000101 00001111 00000101 00001111 00101101 00011000
   A U G    A U U    A C C    A U U    A C C    A U U    A C C    G C U    U G A
```

El servidor lee estos bytes.

Luego, los aminoácidos se traducen a partir de esos codones:

```
   A U G    A U U    A C C    A U U    A C C    A U U    A C C    G C U    U G A
     Met      Ile      Tre      Ile      Tre      Ile      Tre      Ala     Alto
```

Notemos como el existe un único codón de tipo **Alto**, el último codón UGA por lo tanto la secuencia consta de una única proteína.

Notemos además que las respectivas frecuencias son:

```
Met: 1
Ile: 3
Tre: 3
Ala: 1
Fen: 0
....
Gli: 0
```

Y los aminoácidos más frecuentes son `Met` y `Ala` con 1 y `Ile` y `Tre` con 3 (nótese como el codón de **Alto** no es un aminoácido). Al haber empate entre `Ile` y `Tre` se los ordena por orden alfabéticos: primero `Ile` y luego `Tre`. Sucede lo mismo con `Met` y `Ala` y el orden resultante es `Ala` y `Met`.

Dado que se debe retornar los tres aminoácidos más frecuentes la solución es `Ile`, `Tre` y `Ala`.
La respuesta del servidor hacia el cliente es:

```
Cantidad de proteínas encontradas: 1

Aminoácidos más frecuentes:
1) Isoleucina: 3
2) Treonina: 3
3) Alanina: 1
```

Ejemplo 2

Si partimos del archivo **arn_con_colisiones.in**, cuyo contenido es: 'AUGUUUUCUUUCUCCUACUUUUAA', los datos a enviar serán:

```
00000110 00010101 00011101 00010111 00011111 00010011 00010101 00010000
   A U G    U U U    U C U    U U C    U C C    U A C    U U U    U A A
```

En este caso, el servidor deberá responder:

```
Cantidad de proteínas encontradas: 1

Aminoácidos más frecuentes:
1) Fenilalanina: 3
2) Serina: 2
3) Metionina: 1
```

Restricciones

La siguiente es una lista de restricciones técnicas exigidas por el cliente:

1. El sistema debe desarrollarse en ISO C (C99).
2. Está prohibido el uso de variables globales.

Referencias

[1] Operadores Bitwise en C: https://en.wikipedia.org/wiki/Bitwise_operations_in_C
[2] Más información sobre el código genético: https://en.wikipedia.org/wiki/Genetic_code
[3] Frecuencias de codones en el ARN de distintos organismos: http://www.kazusa.or.jp/codon/