

Entrega 1 Proyecto

Estructuras de datos

Grupo 2

Integrantes: Santiago Ibáñez Niño y David Santiago Calderón Idárraga

1. Definición de TADs del sistema de archivos FASTA

a. TAD Secuencia

Datos mínimos:

- NombreSecuencia, Cadena de caracteres, Identifica la secuencia por ejemplo ">Full_SEQUENCE".
- Lineas, Lista de caracteres, las bases que están en el archivo Fasta como: "CTCCGGTGAGAAATTTTGGGATGTATCAAATCACGGTCCTACTAC".
- anchoLinea, número entero, la cantidad de columnas por línea usadas para justificar las líneas.
- Completo, Booleano, Determina si la secuencia que se está completa o es una parte.

Operaciones:

- ObtenerNombre(): retorna el nombre/descripción de la secuencia.
- ObtenerLongitud(): retorna la cantidad total de bases (sin contar "-").
- EsCompleta(): indica si la secuencia está completa o no.
- ContarBase(b): retorna cuántas veces aparece una base b en la secuencia.

b. TAD Histograma

Datos mínimos:

- codigo, arreglo de caracteres, representan en letras las bases nitrogenadas del ADN.
- Resultado, arreglo de enteros, representa la cantidad de caracteres o bases nitrogenadas que se pueden encontrar en una secuencia.

Operaciones:

- OrdenSegunTabla(): verifica el orden de las bases en una tabla de prioridad de bases dada por el enunciado.

c. TAD Genoma

Datos Mínimos:

- Conjunto, Lista de secuencias, agrupa las secuencias de un genoma del archivo Fasta.
- CantSecuencias, entero, contador de cuantas secuencias hay en el genoma.

Operaciones:

- ListarSecuencias(): Entra al conjunto de secuencias del genoma, y muestra por pantalla la cantidad de secuencias y cuantas bases contiene cada una de ellas.

- `ObtenerSecuencias(nombre)`: Se encarga de hallar el histograma de la secuencia que recibe, teniendo en cuenta la tabla de ordenamiento.
- `ExisteSubSecuencia(sub)`: Busca en el genoma, si una serie de bases ingresadas por el usuario existen en alguna de las secuencias, y si existen cuantas veces se repiten.
- `EnmascararSubsecuencia(sub)`: Reemplaza la subsecuencia proporcionada por el usuario por el carácter 'X'.

d. TAD Archivo

Datos Mínimos:

- Ruta, cadena de caracteres, indica la ruta y el nombre del archivo que se está trabajando.

Operaciones:

- `leer(ruta)`: muestra nombre y número de x bases de cada secuencia cargada.
- `guardarArchivo(genoma, nombreArchivoGuardado)`: Guarda en el archivo `nombre_archivo` las secuencias cargadas en memoria. Se debe teniendo en cuenta el ancho de línea(justificación).

e. TAD Menu

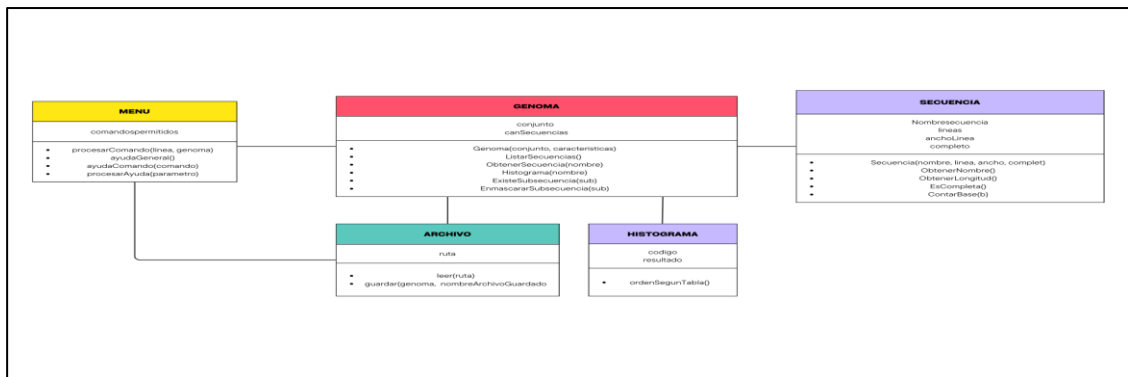
Datos Mínimos:

- `Comandospermitidos`, arreglo de cadena de caracteres, contiene todos los comandos posibles y ejecutables por el programa para manipular los archivos Fasta.

Operaciones:

- `ProcesarComando(linea, genoma)`: Determina que acción va a tomar el programa según lo que haya solicitado el usuario y sea válido.
- `AyudaGeneral()`: Muestra todos los comandos disponibles con sus respectivos parametros.
- `AyudaComando(comando)`: Da información más específica de cada comando posible del sistema.
- `ProcesarAyuda(parámetro)`: Determina si la ayuda solicitada fue general o específica por un comando.

2. Diagrama de TADS



El diagrama de TADS muestra un sistema bien estructurado para la gestión y análisis de genomas, donde cada clase cumple un papel específico y se relaciona de forma coherente con las demás. El menú funciona como interfaz de interacción, el genoma concentra la lógica principal de manejo de secuencias, mientras que la clase secuencia abstrae la información genética en unidades manejables. Además, el archivo asegura la persistencia de datos y el histograma organiza los resultados de los análisis. En conjunto, el diseño refleja un enfoque modular y claro que facilita la reutilización, el mantenimiento y la comprensión del sistema.

3. Plan de Pruebas – Comando enmascarar subsecuencia

Objetivo

Verificar que el comando enmascarar identifique correctamente una subsecuencia dentro de las secuencias cargadas en memoria y la modifique reemplazando sus bases por el código 'X', reflejando la cantidad de enmascaramientos realizados.

Casos de Prueba

- I. Caso 1: No hay secuencias cargadas
 - a. Entrada: enmascarar ATCG
 - b. Condición inicial: No se ha ejecutado cargar.
 - c. Salida esperada: “No hay secuencias cargadas en memoria.”.

Criterio de Éxito: El sistema no intenta modificar nada y muestra el mensaje de error correcto.

- II. Caso 2: Subsecuencia no encontrada
 - a. Entrada: enmascarar GGGG
 - b. Condición inicial: Se cargó un archivo válido con secuencias, pero ninguna contiene "GGGG".
 - c. Salida esperada: La subsecuencia dada no existe dentro de las secuencias cargadas en memoria, por tanto, no se enmascara nada.

Criterio de Éxito: No se modifica ninguna secuencia y se informa correctamente.

- III. Caso 3: Una subsecuencia encontrada
 - a. Entrada: enmascarar ATCG
 - b. Condición inicial: Una de las secuencias cargadas contiene "ATCG" una sola vez.
 - c. Salida esperada: 1 subsecuencia ha sido enmascarada dentro de las secuencias cargadas en memoria.

Criterio de éxito: El sistema reemplaza "ATCG" por "XXXX" y confirma con el mensaje esperado.

IV. Caso 4: Varias subsecuencias encontradas

- a. Entrada: enmascarar TTA
- b. Condición inicial: Las secuencias cargadas contienen "TTA" repetida 3 veces en total.
- c. Salida esperada: 3 subsecuencias han sido enmascaradas dentro de las secuencias cargadas en memoria.

Criterio de éxito: Todas las ocurrencias de "TTA" son reemplazadas por "XXX" en memoria y se informa el número correcto.

Criterios de Éxito Globales

- i. El sistema siempre valida si existen secuencias cargadas antes de intentar enmascarar.
- ii. El número de subsecuencias enmascaradas reportado en pantalla coincide con las modificaciones realizadas.
- iii. No se producen cambios si la subsecuencia no existe.
- iv. El enmascarado es persistente: si luego se ejecuta guardar, las secuencias modificadas deben reflejar los 'X'.

4. Diagrama de flujo comandos

INICIO



Mostrar menú de comandos disponibles



Usuario ingresa comando



| **EVALUAR COMANDO** |



[\$cargar archivo]

→ **menu.cxx recibe el comando**

→ **Llama a Archivo::leerArchivo(nombreArchivo)**

→ **Archivo.cxx:**

- **Abre archivo FASTA**

- **Crea objetos Secuencia (id + cadena ADN)**

- Envía cada Secuencia a Genoma::agregarSecuencia()

→ Genoma guarda todas las secuencias en memoria

[\$listar_secuencias]

→ menu.cxx recibe el comando

→ Llama a Genoma::mostrarSecuencias()

→ Genoma:

- Recorre vector de secuencias

- Imprime el nombre y la cantidad de bases

[\$histograma]

→ menu.cxx recibe el comando

→ Llama a Histograma::generar(genoma)

→ Histograma.cxx:

- Recorre todas las secuencias en Genoma

- Cuenta ocurrencias de A, C, G, T

- Calcula frecuencias

- Muestra histograma textual en consola

[\$es_subsecuencia]

→ menu.cxx recibe el comando con parámetros

→ Llama a Secuencia::esSubsecuencia(otra)

→ Secuencia.cxx:

- Verifica si la cadena de “otra” está contenida en la secuencia

→ Muestra en consola la cantidad de veces que está contenida

[\$enmascarar]

- **menu.cxx** recibe el comando con parámetros
- **Llama a Secuencia::enmascarar**
- **Busca la subsecuencia en las secuencias de Genoma**
- **Secuencia.cxx:**
 - **Sustituye caracteres de la región seleccionada por 'N'**
 - **Actualiza la secuencia en Genoma**
- **Confirmación en consola**

[\$guardar archivo]

- **menu.cxx** recibe el comando
- **Llama a Archivo::guardarArchivo(nombreArchivo, genoma)**
- **Archivo.cxx:**
 - **Recorre todas las secuencias del Genoma**
 - **Escribe en formato FASTA:**
 - >id**
 - secuencia**
 - **Cierra archivo**
- **Mensaje de confirmación en consola**

[\$ayuda]

- **menu.cxx** recibe el comando
- **Recorre la lista de comandos permitidos**

→ **Imprime en consola la lista y descripción breve**

[\$salir]

→ **menu.cxx recibe el comando**

→ **Finaliza el bucle principal**

→ **Libera memoria (si aplica)**

→ **Termina ejecución**

→ **FIN**