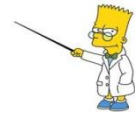


Восстановление ДНК



Цель работы – изучение, реализация и использование односвязных списков.

Сдаваемые файлы:

- linked_list.hpp (возможно вместе с linked_list.h)
- dna_element.cpp (возможно вместе с dna_element.h)
- dna_repairer.cpp

При необходимости сдать другие файлы, необходимо приложить описание данной необходимости в установленном порядке.

Постановка задачи:

Барт Симпсон выполнял важное поручение — нес в пробирке ДНК в хранилище. Поскользнувшись на мокром полу, он уронил пробирку. Она не разбилась, но ДНК от удара перемешались, и их хвосты соединились в неправильном порядке.

В школе есть специальный программно-управляемый аппарат по работе с ДНК. Барт должен написать программу, которая восстановит испорченные ДНК. Помогите Барту восстановить ДНК, чтобы опять не оставаться в школе после занятий.

ДНК заданы в виде строк в формате:

a1:G b2:C b3:C b4:A a4:T a5:G

b1:T b5:T b6:A a6:T b7:C a2:T a3:G

...

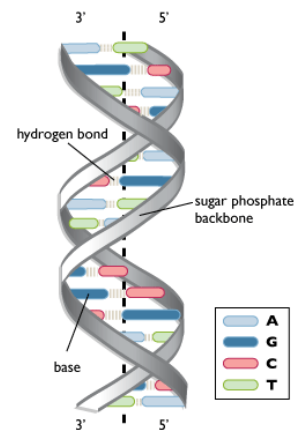
Разные ДНК обозначаются разными малыми латинскими буквами a, b, c...

Каждый элемент ДНК содержит одно из четырех оснований

Adenine,
Cytosine,
Thymine,
Guanine,

обозначенное первым символом.

Таким образом, a1:G означает «*первый элемент ДНК "a", который содержит Guanine*».



При восстановлении важно, чтобы последовательные кусочки, как, например, b2:C b3:C b4:A перемещались только целиком, без разделения на отдельные части, потому что лишние разрезания ДНК могут их испортить.

Необходимо:

1. Реализовать структуру данных список LinkedList, которая содержит операции
 - a. Обращения по индексу
 - b. Вставки, удаления, перемещения элементов в одном списке
 - c. Вставки, перемещения группы элементов в произвольное место списка из другого списка

- d. Определения размера списка
2. Реализовать структуру данных `DNAElement` для хранения одного элемента ДНК, который содержит
 - a. идентификатор ДНК, которой принадлежит элемент
 - b. номер элемента в ДНК
 - c. символ, обозначающий основание
 - d. заполнение полей из строки `[имя][номер]:[тип]`
3. Реализовать класс `DNARepairer`, который содержит
 - a. список `DNAStorage` для хранения всех ДНК. Каждый элемент этого списка отдельный ДНК. Т.е. `DNAStorage` это список списков.
(`LinkedList < LinkedList<DNAElement> >`);
 - b. метод `readfile` для чтения файла в `DNAStorage`;
 - c. метод `repairDNA` для починки всех ДНК;
 - d. метод `printDNA` для вывода содержимого `DNAStorage` в выходной поток.

Входные данные:

Строки, которые читаются из внешнего файла.

1. В N строках файла содержится описание испорченных кусков ДНК.
2. Элементы ДНК разделены одним пробелом;

Имя внешнего файла передается первым параметром в командной строке.

Выходные данные (консоль):

Строки, содержащие цепочки элементов испорченных ДНК

Строки, содержащие цепочки элементов восстановленных ДНК

Указания:

1. Реализовать класс «`LinkedList`» в соответствии с шаблоном.
2. Реализовать класс «`DNAElement`» в соответствии с шаблоном.
3. Реализовать класс «`DNARepairer`» в соответствии с шаблоном.
4. Проводить проверку входных данных.

Возбуждается исключение `std::invalid_argument`:

- если входных данных недостаточно (в строке не завершен элемент или не хватает строк);
- если входная строка содержит хотя бы один недопустимый символ, возбуждается исключение;

Примеры: вход и выход

Входная строка	Выход
a1:T c2:T a2:C a4:A b3:A b7:G b1:G a6:G b4:T c4:C b5:C a5:T a3:C c1:A b6:T c3:G b2:T	Broken DNAs: a1:T c2:T a2:C a4:A b3:A b7:G b1:G a6:G b4:T c4:C b5:C a5:T a3:C c1:A b6:T c3:G b2:T Good as it is: a1:T a2:C a3:C a4:A a5:T a6:G b1:G b2:T b3:A b4:T b5:C b6:T b7:G c1:A c2:T c3:G c4:C