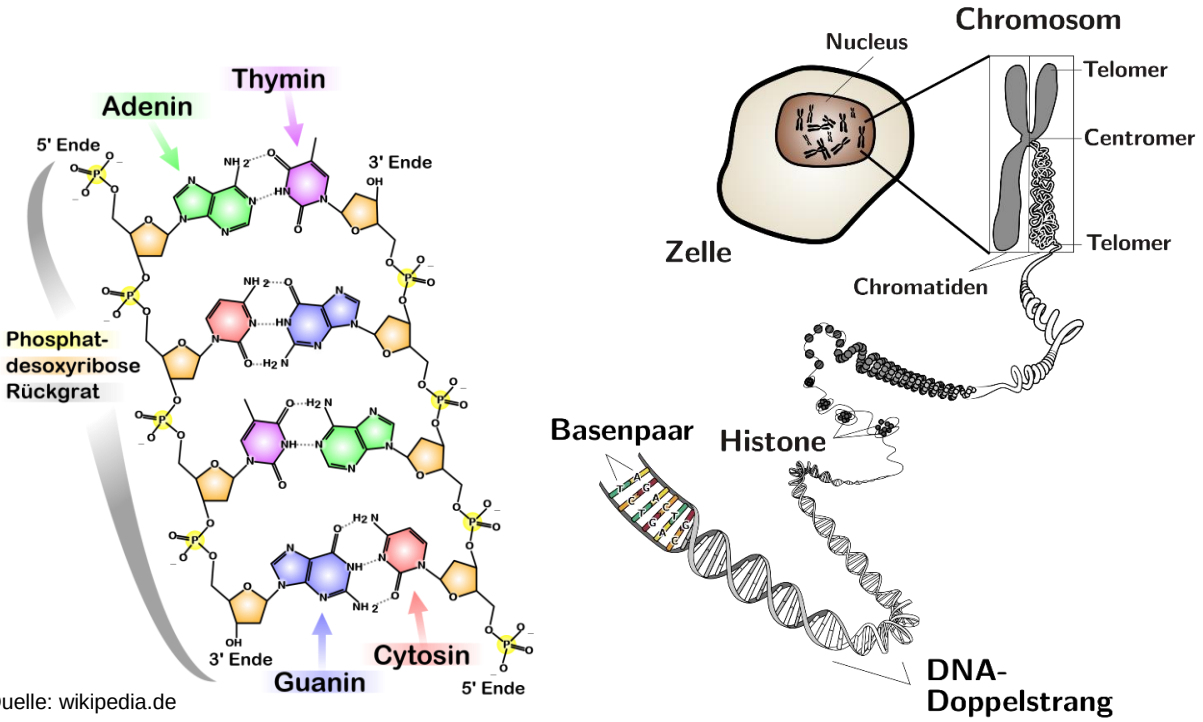
Detektion von DNA-Sequenzen in einem Genom

Prof. Dr. Dieter Koller

v. 18.04.2023



Quelle: wikipedia.de

Der Genetischer Code von Lebewesen besteht aus langen Sequenzen von DNA-Basen, zusammengesetzt aus den vier DNA-Basen Adenin (**A**), Guanin (**G**), Cytosin (**C**) oder Thymin (**T**). Je drei aufeinanderfolgende Basen werden zu einem Basentriplett (einem sog. Codon) zusammengefasst. Ein Codon kodiert entweder den Anfang, das Ende oder eine bestimmte Aminosäure eines Gens, z.B.: **ATG GCT CCA** … . Aufeinander folgende Ketten von meist 100 – 300 Aminosäuren bilden schließlich Proteine und damit die molekularen „*Werkzeuge*“ in einem Organismus.

In den Ribosomen, der „Proteinproduktionsmaschinerie“ der Zelle, wird DNA-Information verwendet, um aus einzelnen Aminosäuren eine Polypeptidkette zusammenzusetzen, wobei die je von einem Codon bestimmten Aminosäuren in der von DNA vorgegebenen Reihenfolge verknüpft werden. Erst mit der Faltung dieser Kette im wässrigen Zellmilieu entsteht dann die dreidimensionale Form eines bestimmten Proteinmoleküls.

Ein Virus wird nicht als Lebewesen betrachtet, weil es keinen eigenen Stoffwechsel und keine eigene Proteinproduktionsmaschinerie hat. Allerdings können Viren Zellen befallen und dort ihre DNA einschleusen, so dass diese dann Proteine für die Viren erzeugen und diese dann in großen Mengen freisetzen.

Eine Möglichkeit der Immunität gegenüber gewissen Viren ist, bestimme DNA-Sequenzen zur Kodierung von Virenproteinen vor dem Zusammensetzen zu erkennen und die Bildung dieser Proteine zu verhindern.

Im Folgenden werde Sie ein paar C-Funktionen und ein C-Programm schreiben, das Ihnen die Möglichkeit gibt, „*schädliche*“ DNA-Sequenzen einzugeben und diese dann in einem vorhandenen Genetischen Code zu finden.

Aufgabe 1

Schreiben Sie eine **C-Funktion**, die nur die Eingabe der Taste **A, C, G** oder **T** erlaubt, d.h. wenn der entsprechende **Groß- oder Kleinbuchstabe** von dem Zeichen gedrückt wird, dann wird der entsprechenden Großbuchstaben dargestellt und auch zurückgeliefert (Kleinbuchstaben müssen dabei in Großbuchstaben gewandelt werden). Ebenfalls erlaubt sein soll die Eingabetaste (`'\n'` **= LF = ASCII 10**) deren Wert allerdings nicht ausgegeben, sondern nur zurückgeliefert werden soll (Vorsicht: in Windows ist ein Zeilenumbruch CR LF = 13 10, so dass sowohl beim ASCII Wert 10 als auch beim ASCII Wert 13 ein `'\n'` zurückgegeben werden soll). Beim Drücken einer beliebigen anderen Taste soll nur mit Ausgabe des ASCII Zeichens 7 durch putchar(7) ein Ton ausgegeben werden und die Eingabe wiederholt werden. Ein NICHT erlaubtes Zeichen soll gar nicht auf dem Bildschirm erscheinen (Verwenden Sie die getch()-Funktion auf Ihrer Plattform).

Funktionsprototyp: `char getDNABase();`

Aufgabe 2

Schreiben Sie eine **C-Funktion**, die eine Folge, aus den Zeichen 'A', 'C', 'G' und 'T' mit Hilfe der Funktion von **Aufgabe 1** von der Tastatur einliest. Die Eingabe wird mit der Eingabetaste abgeschlossen, wobei die Gesamtanzahl der Zeichen in der Folge ein **Vielfaches von 3** sein muss. D.h. die Funktion darf erst beendet werden, wenn die Eingabetaste gedrückt worden ist **und** die Anzahl der bisher eingegebenen Zeichen ganzzahlig durch 3 teilbar ist.

Hinweis: Übergeben Sie der Funktion eine Zeichenkette, die die Folge der eingegebenen Zeichen aufnehmen kann, und geben Sie die Anzahl der eingegebenen Zeichen zurück. Die Eingabetaste `'\n'` soll dabei nicht in der Zeichenkette abgespeichert werden.

Funktionsprototyp: `int getDNASequence(char seq[]);`

Aufgabe 3

Nutzen Sie eine **speicheroptimierte Kodierung**, basierend auf einer C-Enumeration (**`enum`**), um ein Basentriplett aus den drei Basen **A**, **G**, **C** und **T** in einem einzigen Byte zusammenzufassen und schreiben Sie eine **C-Funktion**, die aus einer Zeichenkette bestehend aus **drei Zeichen** aus dem Zeichenvorrat 'A', 'C', 'G' und 'T' (einem sog. Codon) mit Hilfe dieser Kodierung einen einzigen Byte-Wert berechnet und diesen zurückliefert. Verwenden Sie den unten genannten Namen für die Enumeration.

Hinweis: Da jede Base einen von den vier obigen Werten annehmen kann, gibt es für drei Basen hintereinander genau 4*4*4 = 64 Kombinationen. Vergleichen Sie zum Beispiel die 4 verschiedenen Basen mit 4 Ziffern, so wie man 2 Ziffern in einem Bit repräsentieren kann.

```
enum DNABase {A, C, G, T};
```

Funktionsprototyp: `unsigned char encode(char seq[]);`

Aufgabe 4

Das menschliche Genom besteht aus langen Ketten von DNA-Basenpaaren, die auf insgesamt 2*23 = 46 Chromosomen verteilt sind. Die gesamte Erbinformation ist damit in einer Folge von DNA-Basen der Gesamtlänge von 3.2 Milliarden Basenpaare kodiert.

a) Überlegen Sie sich ein **geeignetes Dateiformat**, um mit Hilfe der speicheroptimierten Kodierung von **Aufgabe 3** ein gesamtes menschliches Genom zu speichern. Geben Sie die dazu erforderliche **Dateigröße in Megabyte** an.

Dateigröße für gesamtes menschliches Genom: _____ MB

b) Schreiben Sie eine **C-Funktion**, die das nächste DNA-Basentriplet (Codon) von einer in Ihrem gewählten Format bereits geöffneten Datei liest und zurückliefert. Dazu wird dieser Funktion nur der **Dateizeiger** auf die geöffnete Datei übergeben.

Funktionsprototyp: `unsigned char readCodon(FILE *fp);`

Aufgabe 5

Ein einzelnes Gen besteht aus einer Folge von vielen Tausend dieser Basentriplets und repräsentiert in der Regel eine bestimmte Erbinformation. Um die Präsenz eines bestimmten Gens im Genom nachzuweisen, muss eine bestimmte DNA-Sequenz gesucht werden.

Schreiben Sie eine **Hauptfunktion**, die zunächst mit Hilfe von **Aufgabe 2** die DNA-Sequenz eines zu suchenden Gens einließt. Geben Sie dazu vorher den Text "Geben Sie die DNA-Sequenz des Gens ein: " aus. Beschränken Sie die Länge der DNA-Sequenz aus praktischen Gründen auf maximal 3000 DNA Basen. Die Basentriplets dieser DNA-Sequenz sollen dann unter Verwendung der Funktion in **Aufgabe 3** in eine Byte-Folge kodiert werden, d.h. jedes Basentriplet belegt ein Byte. Verwenden Sie dazu ein geeignetes Array.

Öffnen Sie dann eine Datei mit Namen "genom.dat" im **Binär-Format**. Achten Sie dabei auf eine korrekte Dateibehandlung mit Fehlermeldungen. Diese Datei enthält im Test bereits eine DNA -Sequenz nach obigem Format, nachdem jeweils drei aufeinander folgende Basen binär kodiert in einem Byte abgelegt sind. **Suchen** Sie sodann in dieser Datei Ihre eingegebene DNA-Sequenz und geben Sie beim Auffinden bzw. Nicht-Auffinden eine entsprechende Meldung aus: "Gen gefunden" oder "Gen nicht gefunden". Z.B.:

```
C:\src\Info2\bmc> A1-5
Geben Sie die DNA-Sequenz des Gens ein: AGT-Gen gefunden!
C:\src\Info2\bmc> A1-5
Geben Sie die DNA-Sequenz des Gens ein: GGA-Gen nicht gefunden!
```

Dabei ist Codon AGT in der Genom-Datei enthalten und GGA nicht.

Hinweise:

- Schreiben sie **nur** die **`main()`** Funktion und gehen Sie davon aus, dass die anderen Funktionen bereits implementiert sind.
- Verwenden Sie **nicht** **`#include <conio.h>`**. Diese wird nicht von Moodle unterstützt. Die Funktion `getch()`, die in `getDNABase()` benutzt wird, wird anders zur Verfügung gestellt.
- Beim Test werden verschieden DNA-Sequenz Eingaben überprüft, von denen manche in der Genom Datei enthalten sind, andere nicht. Diese DNA-Sequenzen können irgendwo in der Genom Datei stehen. Es ist nicht erforderlich, ein mehrfaches Vorkommen der DNA-Sequenz in der Genom Datei zu erkennen.

Ergänzung

In der Abbildung unten finden zur Information den erst kürzlich auf GitHub veröffentlichen mRNA-Code des BioNTech Impfstoffes gegen Corona Infektionen:

Figure 1: Spike-encoding contig assembled from BioNTech/Pfizer BNT-162b2 vaccine.

Cyan: Putative 5' UTR
Green: Start Codon
Yellow: Signal Peptide
Orange: Spike encoding region
Red: Stop codon(s)
Purple: 3' UTR
Blue: Start of polyA region (incomplete)

Quelle: https://github.com/NAalytics/Assemblies-of-putative-SARS-CoV2-spike-encoding-mRNA-sequences-for-vaccines-BNT-162b2-and-mRNA-1273/blob/main/Assemblies%20of%20putative%20SARS-CoV2-spike-encoding%20mRNA%20sequences%20for%20vaccines%20BNT-162b2%20and%20mRNA-1273.docx.pdf