9 Block 9: Von der Idee zum Programm: `Restrict.pl`

9.1 Lernziele

Ziel dieses Blocks ist es die bisher vorgestellten programmatischen Elemente und syntaktischen Regeln von `Perl` in dem zu schreibenden Programmbeispiel `Restrict.pl` zusammenzuführen.

Im Zentrum steht zu das *Planen* und *Entwerfen* eines Programms zu erlernen: Zuerst wird eine *Liste von Anforderungen* an das Programm erstellt, dann wird ein *natürlich-sprachliches* Programmkonzept, so genannter *Pseudocode*, entwickelt.

Auf der Basis des bisher Gelernten wird dann der Programmtext einer Musterlösung besprochen und analysiert. Danach soll das Programm `Restrict.pl` ausgeführt und der Programmtext studiert werden. Die Kommentare sollen ergänzt werden. Zusätzlich soll ein *Struktogramm* erstellt werden.

Anschliessend werden und *Erweiterungs- bzw. Verbesserungsvorschläge* gesammelt und Möglichkeiten diskutiert diese programmatisch umzusetzen.

9.2 Praxis

9.2.1 Generelle *Anforderungen* an `Restrict.pl`

„Das Programm soll auf einer bestimmten Nukleotid-Sequenz die Erkennungssequenz für ein Restriktionsenzym bestimmen.“

9.2.2 Detaillierte Spezifizierung von `Restrict.pl`

Das Programm `Restrict.pl` soll

- Restriktionsenzymdaten aus einer Datei lesen;
- Sequenzdaten aus einer Datei Lesen;
- Für ein vom Benutzer definiertes Enzym, die Erkennungssequenz auf der Sequenzdatei suchen;
- Gefundene Schnittstellen auf dem Bildschirm ausgeben;
- Die Suche in einer Schleife für weitere Enzyme wiederholen bis der Benutzer eingibt das Programm beenden zu wollen;

Welche Programmelemente müssen weiter im Detail spezifiziert werden?

9.2.2.1 Datenquellen

9.2.2.1.1 Eingabe

Sequenzdaten	Quelle	Datei im aktuellen Arbeitsverzeichnis (gleiches Verzeichnis wie `Restrict.pl`).
	Format	FASTA
	Auswahl	Beim Programmaufruf als 1.Kommandozeilen-Argument in der Art `perl Restrict.pl [Sequendatei-Name] [Argument 2].`
Enzymdaten	Quelle	Datei im aktuellen Arbeitsverzeichnis (gleiches Verzeichnis wie `Restrict.pl`).
	Format	Leerzeichen-separierte Tabelle des Typs: `Enzym_Muster.tab`
	Auswahl	Beim Programmaufruf als 2.Kommandozeilen-Argument in der Art `perl Restrict.pl [Sequendatei-Name] [Enzymdatei-Name].`
Auswahl eines Enzyms	Quelle	Kommandozeile.
	Format	Text des Typs: `HindIII`

9.2.2.1.2 Ausgabe

Auf dem Bildschirm sollen alle erkannten Restriktionsschnittstellen mit einer Positionsangabe (in bp vom Start der Eingabesequenz) ausgegeben werden:

```
Suche nach Erkennungssequenz CAGCTG fuer PvuII:
Schnittstellen auf +Strang (Startposition in bp) in pZERO2.fasta:
53 498 1879 2239
```

Die gesamte Ausgabe unter Laufzeit soll wie folgt aussehen:

```
Nach Erkennungssequenz fuer welches Enzym suchen (oder Ende)? HindIII
Die Erkennungssequenz fuer HindIII ist: AAGCTT
Suche nach Erkennungssequenz AAGCTT fuer HindIII:
Schnittstellen auf +Strang (Startposition in bp) in pZERO2.fasta:
276

Nach Erkennungssequenz fuer welches Enzym suchen (oder Ende)? PvuII
Die Erkennungssequenz fuer PvuII ist: CAGCTG
Suche nach Erkennungssequenz CAGCTG fuer PvuII:
Schnittstellen auf +Strang (Startposition in bp) in pZERO2.fasta:
53 498 1879 2239

Nach Erkennungssequenz fuer welches Enzym suchen (oder Ende)? KpnI
Die Erkennungssequenz fuer KpnI ist: GGTACC
Suche nach Erkennungssequenz GGTACC fuer KpnI:
Schnittstellen auf +Strang (Startposition in bp) in pZERO2.fasta:
282

Nach Erkennungssequenz fuer welches Enzym suchen (oder Ende)? Ende
```

9.2.3 Pseudocode für `Restrict.pl`

- **Eingabe 1: Einlesen der FASTA-Sequenzdatei über Kommandozeilenargument 1:**
  - Lesen des Kommandozeilenarguments 1;
  - Öffnen der Datei zum Lesen;
  - Zuweisung des Datei-Inhalts an eine Variable (Datentyp X);
  - Datei schliessen;
  - Entfernen der FASTA-Kopfzeile aus dem zu untersuchenden Sequenztext und Zuweisung der *Rohsequenz* an eine Variable (Datentyp X);
- **Eingabe 2: Einlesen der Datei mit den Enzymdaten über Kommandozeilenargument 2:**
  - Lesen des Kommandozeilenarguments 2;
  - Öffnen der Datei zum Lesen;
  - Zuweisung des Datei-Inhalts an eine Variable (Datentyp X);
  - Datei schliessen;
  - Entfernen der Kommentarzeilen aus dem zu untersuchenden Dateitext und Zuweisung der *Enzymdaten* an eine Variable (Datentyp X);
- **Datenanalyse: Finden der Schnittstellen:**
  - Solange Benutzer Programm nicht beenden will...
    - Eingabeaufforderung: Welches Enzym suchen?
    - Einlesen der Auswahl in Variable
    - Wenn Variable ein Enzym-Name der in Eingabe 2 vorhanden ist...
      - Bereitstellen der Erkennungssequenz aus den Daten von Eingabe 2
      - Schnittstellen für Auswahl auf Eingabe 1 suchen
        - Mustererkennungs-Schleife über die Zeichen der Rohsequenz
          - Rückgabe der Positionen aller Erkennungssequenzen
      - Schnittstellen auf dem Bildschirm ausgeben
    - ...sonst Ausgabe, dass keine entsprechenden Enzymdaten vorhanden; Programmende
  - ...sonst Programmende

9.2.4 Programmtext von `Restrict.pl`

```
#!/usr/bin/perl -w

# Programm-Name: Restrict.pl
# Autor: picker
# Datum: 11/03
# Aufruf: perl Restrict.pl [Dateiname Sequenz] [Dateiname Enzyme]

######################################################
# Block 1: Oeffnen und Lesen der FASTA Sequenzdatei #
######################################################

# Zugriff auf das erste Kommandozeilen-Argument [Dateiname Sequenz].
$Seq_Datei_Name = $ARGV[0];
chomp $Seq_Datei_Name;
# Oeffnen der FASTA Datei zum Lesen:
open (FILE, "<$Seq_Datei_Name");
# Zuweisung des Dateiinhalts an einen Array:
@Seq_Datei = <FILE>;
close FILE;
# Aufruf der Subroutine "lesen_sequenz_daten" zum Lesen der Sequenzinformation
# aus @Seq_Datei, mit dem gesamten Inhalt (inkl. ">..." Kopfzeile).
$Sequenz = lesen_sequenz_daten(@Seq_Datei);
# $Sequenz enthaelt nun die ganze Sequenz als eine Zeichnkette.

#############################################################################
# Block 2: Oeffnen und Lesen der Datei mit der Restriktionsenzym-Information #
#############################################################################

# Zugriff auf das zweite Kommandozeilen-Argument [Dateiname Enzyme].
$Enzym_Datei_Name = $ARGV[1];
chomp $Enzym_Datei_Name;
# Oeffnen der Enzym Datei zum Lesen:
open (FILE, "<$Enzym_Datei_Name");
# Zuweisung des Dateiinhalts an einen Array:
@Enzym_Datei = <FILE>;
close FILE;
# Aufruf der Subroutine "lesen_enzym_daten" zum Lesen der Information
# aus @Enzym_Datei, mit dem gesamten Inhalt (inkl. ">..." Kopfzeile).
%Enzyme = lesen_enzym_daten(@Enzym_Datei);
# %Enzyme enthaelt nun die ganze Enzym-Information.

######################################################
# Block 3: Finden von Schnittstellen auf der Sequenz #
######################################################

do {
    print "\nNach Erkennungssequenz fuer welches Enzym suchen (oder Ende)? ";
    $Auswahl = <STDIN>;
    chomp $Auswahl;
    if (exists $Enzyme{$Auswahl}) {
        $Enzym_Sequenz = $Enzyme{$Auswahl};
        print "Die Erkennungssequenz fuer $Auswahl ist: $Enzym_Sequenz\n";
        @schnittstellen = schnittstellen_finden($Enzym_Sequenz, $Sequenz);
        if (@schnittstellen) {
            print "Suche nach Erkennungssequenz $Enzym_Sequenz fuer $Auswahl:\n";
            print "Schnittstellen auf +Strang (Startposition in bp) in $Seq_Datei_Name:\n";
            print join(" ", @schnittstellen), "\n";
        }
        else {
            print "Keine Schnittstellen fuer $Auswahl auf +Strang von $Seq_Datei_Name.\n";
        }
    }
    else {
        unless ($Auswahl =~ /Ende/) {
            print "Keine Daten fuer Enzym $Auswahl in der Quelldatei $Enzym_Datei_Name";
        }
    }
} until ($Auswahl =~ /Ende/);
exit;


#############################################################################
```

```
# Subroutine (Unterprogramm), das einen Array mit Dateininhalten
# in eine String-Variable $Sequenz speichert.
sub lesen_sequenz_daten {
    my (@fasta_datei) = @_;
    my $sequenz ='';
    foreach my $zeile (@fasta_datei) {
        if ($zeile =~ /^>/) {
            next;
        }
        else {
            substr($zeile, -1) = "";
            $sequenz .= $zeile;
        }
    }
    return $sequenz;
}

################################################################################
# Subroutine (Unterprogramm), das einen Array mit Dateininhalten
# in einem Hash (assoziative Liste) speichert.
sub lesen_enzym_daten {
    my (@enzym_datei) = @_;
    my %enzym_hash = ();
    my $enzym_name;
    my $enzym_seq;
    foreach my $zeile (@enzym_datei) {
        if ($zeile =~ /^#/) {
            next;
        }
        else {
        my @daten_ein_enzym = split (" ", $zeile);
        $enzym_name = shift @daten_ein_enzym;
        $enzym_seq = pop @daten_ein_enzym;
        $enzym_hash{$enzym_name} = "$enzym_seq";
        }
    }
    return %enzym_hash;
}

################################################################################
# Subroutine (Unterprogramm), das auf einer Zielsequenz Erkennungssequenzen
# fuer Restriktionsenzyme sucht und das Ergebnis in einem Array speichert und
# zurueckgibt.
sub schnittstellen_finden {
    my ($erkennungs_sequenz, $sequenz) = @_;
    my @positionen = ();
     while ($sequenz =~ /$erkennungs_sequenz/ig) {
        push (@positionen, pos($sequenz) - length($&)+1);
    }
    return @positionen;
}
```

9.2.4.1 Übung

9.2.4.1.1 Aufgabe

- Führen sie das Programm `Restrict.pl` auf ihrem Computer aus. Beobachten sie die Ausgabe bei verschiedenen Eingaben und studieren sie den Programmtext;
- Fertigen sie ein detailliertes Struktogramm an;
- Machen sie Vorschläge zur Erweiterung, Verbesserung.