Dateien öffnen und schreiben

- Wie kann man ein Programm benutzen?
  - Interaktiv: Alle Daten direkt (z.B. per Tastatur) eingeben
  - Skriptgesteuert: Steuerungsdatei(en) mit Datensätzen und Anweisungen
- Erster Schritt: Das Programm muss Dateien lesen und schreiben können
- C++ (vgl. 1.VL): Umgang mit Dateien über Ein- und Ausgabestreams (`ifstream` und `ofstream`) → `cout_datei.cc`
- Eingabe des Dateinamens: Benutze ein Feld aus `char`-Variablen
- Wichtig: Zuerst immer prüfen, ob die Datei tatsächlich existiert
- Nicht vergessen: Alle geöffnete Dateien ordnungsgemäß schließen → `file-open.cpp`

- Datei bis zum Ende auslesen: So lange lesen, bis `fin >> entry` eben `false` zurückgibt → `file-open2.cpp`
- Bei strukturierten Dateien (z.B. Einträge von zwei Vektoren) könnten die ersten Einträge die Dateistruktur vorgeben (z.B. die Erste Zahl als Vektorenlänge) → `calc-sp.cpp`
- Was tun bei fehlerbehafteten Dateien?
- Spezielle Methoden geben Aufschluss über den Zustand des Streams
  - `fin.good()`: 1, wenn es keine Fehler gibt, sonst 0
  - `fin.eof()`: 1, sobald das Dateiende erreicht wurde, sonst 0
  - `fin.fail()`, `fin.bad()`: 1 bei Lesefehlern, sonst 0
- Wir können unser Programm zur Berechnung des Skalarprodukts nun entsprechend anpassen → `calc-sp2.cpp`

Matrizenoperationen

- Wie kann man eine Matrix als Zahlenfolge einlesen?
- Benutze Division ohne Rest und Modulo bei bekannter Größe
- Beispiel 3×3 Matrix

i	div 3	mod 3	Eintag
0	0	0	a_{11}
1	0	1	a_{12}
2	0	2	a_{13}
3	1	0	a_{21}
4	1	1	a_{22}
5	1	2	a_{23}
6	2	0	a_{31}
7	2	1	a_{32}
8	2	2	a_{32}

- Und so erhält man

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Matrizenoperationen

- Viele nützliche Operationen mit Matrizen lassen sich direkt implementieren
- Transponieren → `mat-transpose.cpp`

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \rightarrow \begin{pmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{pmatrix}$$

- Skalieren → `mat-scale.cpp`

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \rightarrow \lambda \begin{pmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{pmatrix}$$

- Spur berechnen → `mat-trace.cpp`

$$\mathrm{Tr}\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = a_{11} + a_{22} + a_{33}$$

- Multiplizieren

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

- Matrizenmultiplikation in der Indexschreibweise → `mat-mul.cpp`

$$c_{ij} = \sum_k a_{ik} b_{kj}$$

Nachbesprechung Übungsaufgaben

- Aufgabe 7: Sehr einfach, z.B. für a prüft man

$$b \leq a \leq c, \quad c \leq a \leq b$$

  Viele Lösungsmöglichkeiten $\rightarrow$ `a7-mittlere.cc`, `a7-mittlere-v2.cc`, ...
- Aufgabe 8: Es gilt

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$$

- Die Einträge in der n-ten Zeile (wenn man von 0 aus rechnet) des Pascalschen Dreiecks, ergeben die Koeffizienten der Produkte $a^k b^{n-k}$.
- Z.B. lautet die 5. Zeile: $1 \quad 4 \quad 6 \quad 4 \quad 1$, somit

$$(a+b)^4 = a^4 + 4 \times a^3 b + 6 \times a^2 b^2 + 4 \times a b^3 + b^4$$

- Implementierung mit Hilfsfeld unkompliziert $\rightarrow$ `a8-binoko.cc`

- Implementieriung ohne Hilfsfeld:
- Grundidee: Konstruiere die neue Zeile rückwärts, dann kann man die Werte der aktuellen Zeile gefahrlos überschreiben.
- Es wird kein Hilfsfeld benötigt
- Beispiel: 4. Zeile ist 1331, konstruiere daraus die 5. Zeile

1) `zeile[5]=1` $\Rightarrow$ `zeile`: 13311

2) `zeile[4]=zeile[4]+zeile[3]` $\Rightarrow$ `zeile`: 13341

3) `zeile[3]=zeile[3]+zeile[2]` $\Rightarrow$ `zeile`: 13641

4) `zeile[2]=zeile[2]+zeile[1]` $\Rightarrow$ `zeile`: 14641

- Fertig ist die 5. Zeile: $14641 \rightarrow$ `a8-binoko-v2.cc`

- Aufgabe 9: Programmiertechnisch einfach, sofern man die richtige Formel hat.
- Referenzdatum wählen, z.B. Montag, den 1.1.1601
- Sei Δa die Jahresdifferenz zum Referenzdatum, naiv: $\Delta t_1 = 365\Delta a$ Tage
- Schaltjahr: Man muss einen Zusatztag hinzufügen (29.02)
- Jedes 4. Jahr ist ein Schaltjahr: $\Delta t_1 + \Delta a/4$
- Es sei denn die Jahreszahl ist ohne Rest durch 100 teilbar: $\Delta t_1 + \Delta a/4 - \Delta/100$

- Immer Schaltjahr, wenn die Jahreszahl glatt durch 400 teilbar ist: $\Delta t_1 + \Delta a/4 - \Delta a/100 + \Delta a/400$
- Die Anzahl der Tage in jedem Monat als Feld implementieren
- Dann aus der Monatszahl t_2 berechnen.
- Extra Tag hinzufügen, falls es sich um einen Monat nach dem Februar handelt und das aktuelle Jahr ein Schaltjahr ist
- Alles aufaddieren, inkl. den aktuellen Monatstag
- Aus der Gesamtzahl der Tage mod 7 den Wochentag bestimmen: $\rightarrow$ `a9-wochentag.cc`

Vorbesprechung Übungsaufgaben

- Aufgabe 10: $A_{\text{Kreis}} = \pi r^2$, somit für $r = 1$: $A_{\text{Einheitskreis}} = \pi$
- Wir wollen $A_{\text{Einheitskreis}}$ bestimmen.
- Einheitskreis in ein Einheitsquadrat eingezeichnet: $A_{\text{Einheitskreis}}$ aus dem Verhältnis der beiden Flächen
- Schätzung: n_{Treffer}/n
- Zufallszahlen erzeugen $\rightarrow$ `random.cpp`
- Aufgabe 11: Datei geschickt einlesen (vgl. vorherige Beispiele mit Matrizen), dann die Tabelle durchlaufen und die Gesamtzahl der Tore pro Mannschaft ermitteln usw.
- Aufgabe 12: Mit den wertvollsten Münzen beginnen, d.h. die Schleife rückwärts laufen lassen. Dann Division ohne Rest und Modulo benutzen.