Struct, union, enum, Funktionen

- Unterschied im Speicherplatzbedarf zwischen `struct` und `union` → `unionAndStruct.cpp`
- Man kann mit `union` etwas Speicherplatz sparen → `unionAndStruct2.cpp`
- Bei `enum` muss die Nummerierung nicht unbedingt bei 0 anfangen → `enum.cpp`
- Funktionen dürfen sich lediglich durch ihre Argumente unterscheiden → `fun.cpp`

Nachbesprechung Übungsaufgaben

- Aufgabe 13: $\rightarrow$ `a13-lgs-gauss.cc`, Schritt-für-Schritt $\rightarrow$ `gauss-demo.cc`
- Falsches Ergebnis bei `a13-lgs2.dat` ohne die Spaltenpivotisierung
- Aufgabe 14: $\rightarrow$ `a14-vollkZ.cc`

Vorbesprechung Übungsaufgaben

- Aufgabe 15: Datei öffnen und die ersten beiden Einträge einlesen $\Rightarrow$ Bildgröße $X \times Y$
- Äußere Schleife, die über die Anzahl der Spalten (Y) läuft
- y als aktuelle Spalte
- In jedem Durchlauf zwei Einträge einlesen (Zeichencode, # d. Wiederholungen)
- Innere Schleife für die Bildschirmausgabe
- Was passiert, wenn der Zeichencode nicht existiert?
- Achtung: # d. Wiederholungen kann über die Zeilengrenzen hinausgehen!
- Daher die Position in der aktuellen Zeile (x) beachten und wenn nötig (`x mod X==0`) zu der nächsten Spalte übergehen (`y++`)

- Aufgabe 16: Solange Wertepaare einlesen, bis die Datei zu Ende ist
- Anzahl der Stützstellen mitzählen!
- Ergebnisdatei erstellen und öffnen
- Endlose Schleife (`while(true){...}`) für die Benutzereingaben
- Berechnung des Interpolationspolynoms für einen neuen Wert: Doppelschleife (Summe += und Produkt *=), $k \neq i$ beachten!
- Den interpolierten Wert auf den Bildschirm ausgeben und in die Datei schreiben

- Gnuplot: Opensource Tool zum Plotten `http://www.gnuplot.info/`

```
plot 'a16-interpol.dat' using 1:2
```

oder

```
plot 'a16-interpol.dat' using 1:2 smooth unique
```

- Skripten möglich `gnuplot-command.txt`
- Siehe auch `Gnuplot.pdf`
- Alternative: Python mit matplotlib $\Rightarrow$ `example-matplotlib-call.py`

- Nim-Spiel: Mensch gegen Mensch als unendliche Schleife implementieren
- Davor werden die Holzstapeln erzeugt (Zufallsgenerator)
- In jedem Zug die Anzahl der Hölzer in beiden Stapeln ausgeben
- Auf Regelverstoße prüfen!

- Mensch gegen Maschine: Jeder Spielzug als (m, n) parametrisierbar: Anzahl der verbleibenden Hölzer in beiden Stapeln
- Es gibt Stellen, die für den Spieler günstig oder ungünstig sind.
- Günstige Stelle: Wenn man geschickt spielt, kann man den Gegenspieler auf eine ungünstige Stelle versetzen
- Ungünstige Stelle: Keine Chance, den Gegenspieler auszutricksen.
- Bei $(1, 2)$, $(3, 5)$, $(4, 7)$ usw. handelt es sich um ungünstige Stellen.
- Egal wie man die Hölzer abzieht, der Gegenspieler kann den Spieler immer auf eine andere ungünstige Stelle versetzen
- Bsp $(3, 5)$: Egal was man macht, man landet entweder bei $(1, 2)$ oder man verliert gleich.
- Strategie: Computer kennt alle ungünstigen Stellen, muss den Spieler auf eine solche Stelle versetzen