Notizen AIK ProgTech 3: Programme mit mehreren Files

Klaus Kusche

Worum geht es?

- Größere Projekte werden meist *auf mehrere Files und mehrere Entwickler verteilt.*
- Übliche Faustregel für die *Aufteilung*:
  - **Je ein `.h`-File und ein `.cpp`-File pro Klasse.**
    (Außer alles, was zur Klasse gehört, steht direkt im **`class`**:
    Dann gibt es nur einen **`.h`**-File, keinen **`.cpp`**-File.)
  - **Ein separater `.cpp`-File für `main`.**

Was gehört in den `.h`-File?

In den **`.h`**-File gehört alles, was ein anderer **`.cpp`**-File *von der Klasse wissen muss, um sie verwenden zu können* (die *Deklaration* der Klasse):

- Das **`class { ... };`** samt Inline-Methoden-Code.
- **`#include`**'s für alle im **`class`** verwendeten Dinge.
- **`typedef`**'s, **`#define`**-Konstanten, Funktions-Prototypen ... für im **`class`** verwendete Dinge und Dinge, die man zur Verwendung der Klasse braucht.
- Eventuelle separate **`inline`**-Methoden-Definitionen.
- Später: Alle Template-Definitionen.

Ein **`.h`**-File wird nie für sich allein kompiliert, sondern immer nur beim Kompilieren anderer Files *inkludiert* (typischerweise in *mehreren* **`.cpp`**-Files).

Der **`.h`**-File darf *keinen Code erzeugen* und *keinen Speicherplatz anlegen*!!!
(Der Code / Speicherplatz würde sonst *mehrfach im kompilerten Code stehen*, wenn der **`.h`**-File von mehreren **`.cpp`**-Files inkludiert wird).

Andere **`.cpp`**-Files und **`main`** sehen von der Klasse nur den **`.h`**-File, *nicht* den **`.cpp`**-File.

Was gehört in den `.cpp`-File?

In den **`.cpp`**-File gehört die *Implementierung* (das *"Innenleben"*) der Klasse:

- Der Code für alle Methoden, bei denen er nicht gleich im **`class`** steht.
- Ein **`#include`** *für den eigenen Header-File (wichtig!).*
- **`#include`**'s, **`typedef`**'s, Konstanten, ..., die im Code gebraucht werden und noch nicht im **`.h`**-File gemacht wurden (nur die Klasse intern etwas angehen).
- Die Definitionen und Initialisierungen der **`static`**-Membervariablen.
- Eventuell die Definitionen der **`friend`**-Funktionen.

Jeder **`.cpp`**-File wird für sich allein kompiliert, nie inkludiert.

Wie ist das mit dem `#ifndef` *?*

Ein Header-File könnte beim Kompilieren *eines* **.cpp**-Files *mehrfach inkludiert werden* (direkt und indirekt).
Dann würde sich der Compiler über doppelte Deklarationen beschweren.

Man muss daher dafür sorgen, dass *ab dem 2. Mal ein leerer File inkludiert* wird.
Man macht das so:

- Ganz oben im **.h**-File:

  ```
  #ifndef _MYCLASS_H          (oder wie der Header eben heißt)
  #define _MYCLASS_H 1
  ```

- Ganz unten im File:

  ```
  #endif
  ```

Der *Name* des Defines ist *frei wählbar* (***_xxx_H*** ist üblich)
und muss in jedem Header ein *anderer* sein!

Effekt:

- Beim *ersten* Include ist **_MYCLASS_H** noch nicht definiert,
  das **#ifndef** daher **true**, und der Fileinhalt sichtbar.
- Ab dem *zweiten* Include ist **_MYCLASS_H** definiert, das **#ifndef** daher **false**,
  und alles bis zum **#endif** wird übersprungen.

Hinweise DevCpp

- Immer wenn man mehr als einen **.cpp**-File hat, muss man ein *Projekt anlegen*
  und die einzelnen **.cpp**-*Files zu dem Projekt hinzufügen.*
- DevCpp (so, wie es bei uns eingestellt ist) compiliert die **.cpp**-Files nicht neu,
  wenn man einen *Header-File ändert* ==› "Alles neu bauen" statt "bauen" aufrufen!
- Alle Verzeichnis-Angaben sind relativ zu dem *Verzeichnis, in dem der Projektfile* liegt
  (nicht relativ zu dem Verzeichnis, in dem der Source-File liegt!):
  - Suche von **#include "..."** -Files.
  - Suche von Libraries und DLL's.
  - Ablegen des erzeugten **.exe** .

  ==› Bei uns bei Verwendung der SDL:
  Der SDL-Ordner mit den SDL-Header-Files, der **.a**-File mit den SDL-Libraries
  und die **SDL.dll** müssen in den *Projektordner*!