Diese Notizen basieren auf der Youtube-Reihe Cpp von { bis } von Bytes'n'Objects.

1 Initialisierungslisten

In der Initialisierungsliste sollte man für die Initialisierung von Feldern einer Klasse keine anderen Felder der selben Klasse verwenden, sondern Konstruktorparameter oder Konstanten (Es sei denn man ist sich 100 prozentig sicher). Begründung ist, dass die Felder nicht entsprechend der Reihenfolge in der Liste initialisiert werden, sondern entsprechend der Deklaration in der Klasse.

2 Rule of three

Die drei sind: copy constructor, destructor und copy assignment operator. Und die Regel besagt, dass wenn ich einen von den drein in einer Klasse implementiere, sollte ich voraussichtlich auch die übrigen zwei mit implementieren.

3 Pointer

Man sollte statt nackten Pointern eigentlich nur shared-pointer verwenden.

4 Weak pointer

Wenn man auf einen Weak-pointer `w` der auf ein Objekt zeigt auf das kein shared-pointer mehr zeigt, bekommt man mit `w.lock()` wieder einen shared-pointer, der aber nicht gültig ist. Daher sollte man immer bevor man auf einen weak-pointer lock aufruft vorher mit `w.expired()` abchecken, ob das noch gültig ist. Oder man überprüft anschließend, ob der shared-pointer den man bekommt gültig ist.

5 struct

Felder und Methoden einer Struct sind immer alle defaultmäßig public.

6 aquisition is initialization

7 virtual und override

Virtual Funktionen werden zur Laufzeit gebunden, also nicht schon beim compilen. D.h. man hat zB eine Basisklasse Haustier und weiss, dass jedes Haustier einen Laut von sich geben soll, dann kann man in der Basisklasse diese Funktionalitaet schon bereit stellen, aber wenn man dann die Klasse Katze hat, die von der Basisklasse erbt, kann man darin dann die Funktion umschreiben so dass der Laut auch zur Katze passt. Das Program muss also so compiliert werden, dass auch zur Laufzeit noch festgestellt werden kann, was man genau vor sich hat (welches spezielle Haustier zB.).

7.1 override

Ist ein modifier, der angibt, dass man gerade explizit eine Funktion aus der Basisklasse überschreibt. Damit kann verhindert werden, dass man sich vertippt und aus versehen doch nicht überschreibt

8 int main() vs int main(void)

Beide sind in cpp gleichbedeutend. Der Unterschied ist nur in C, da main() bedeutet, dass die Fkt mit beliebeig vielen Parametern aufgerufen werden kann, wobei main(void) sagt, dass die Fkt nur ohne Parameter aufgerufen werden kann.

9 Inheritance/Vererbung

In Cpp kann man bei der Vererbung die modes public, protected und private mit angeben. Das bedeutet zB am Bsp von public, dass public members der Baseklasse in der abgeleiteten Klasse auch public bleiben und die protected members bleiben protected.

Private scheint der default zu sein und das bedeutet, dass die public und protected member der baseklasse private in der abgeleiteten Klasse werden.

Member die in der Baseklasse schon private waren werden gar nicht mit vererbt.

10 Fragen