# C++-Programmierung

## Modularer Programmaufbau

Lindemann

# Ausgangslage

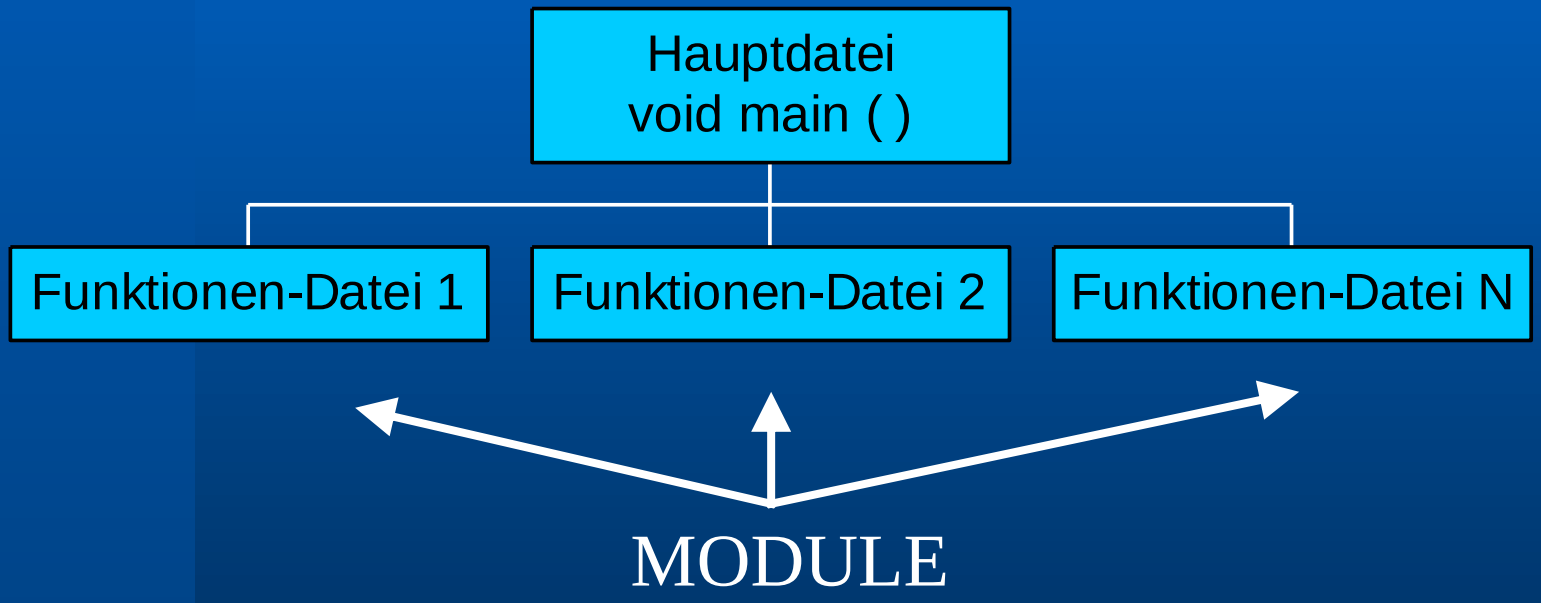
Größere Programmierprojekte werden schnell unübersichtlich, wenn der **gesamte Quellcode in einer Datei** steht. Vor allem, wenn der Quellcode **wenig strukturiert** ist.

# Strukturierung

Ein erster Schritt zur Strukturierung ist das Schreiben von Funktionen für alle wiederkehrenden Aufgaben.

Der zweite Schritt zur Strukturierung ist das Zusammenfassen ähnlicher Funktionen in separaten Dateien.

# Modularer Programmaufbau



# Schnittstellen

Damit das Hauptprogramm die Funktionen aus den Modulen nutzen kann, muss das Hauptprogramm die Deklarationen der Funktionen kennen.

# Schnittstellen - Header-Dateien

Dafür werden die Deklarationen der Funktionen aus einem Modul in eine *Header-Datei* eingetragen. Diese *Header-Datei* wird dann in der der Präprozessordirektiver der Haupt-Datei per **#include** eingebunden.

# Modularer Programmaufbau

Modul_1.cpp

```
int funk_1 ( int  i)
{
int x;
for ( x = i; x > 0 ; x--)
.....
}
float funk_2 ( char y)
{ ..... }
```

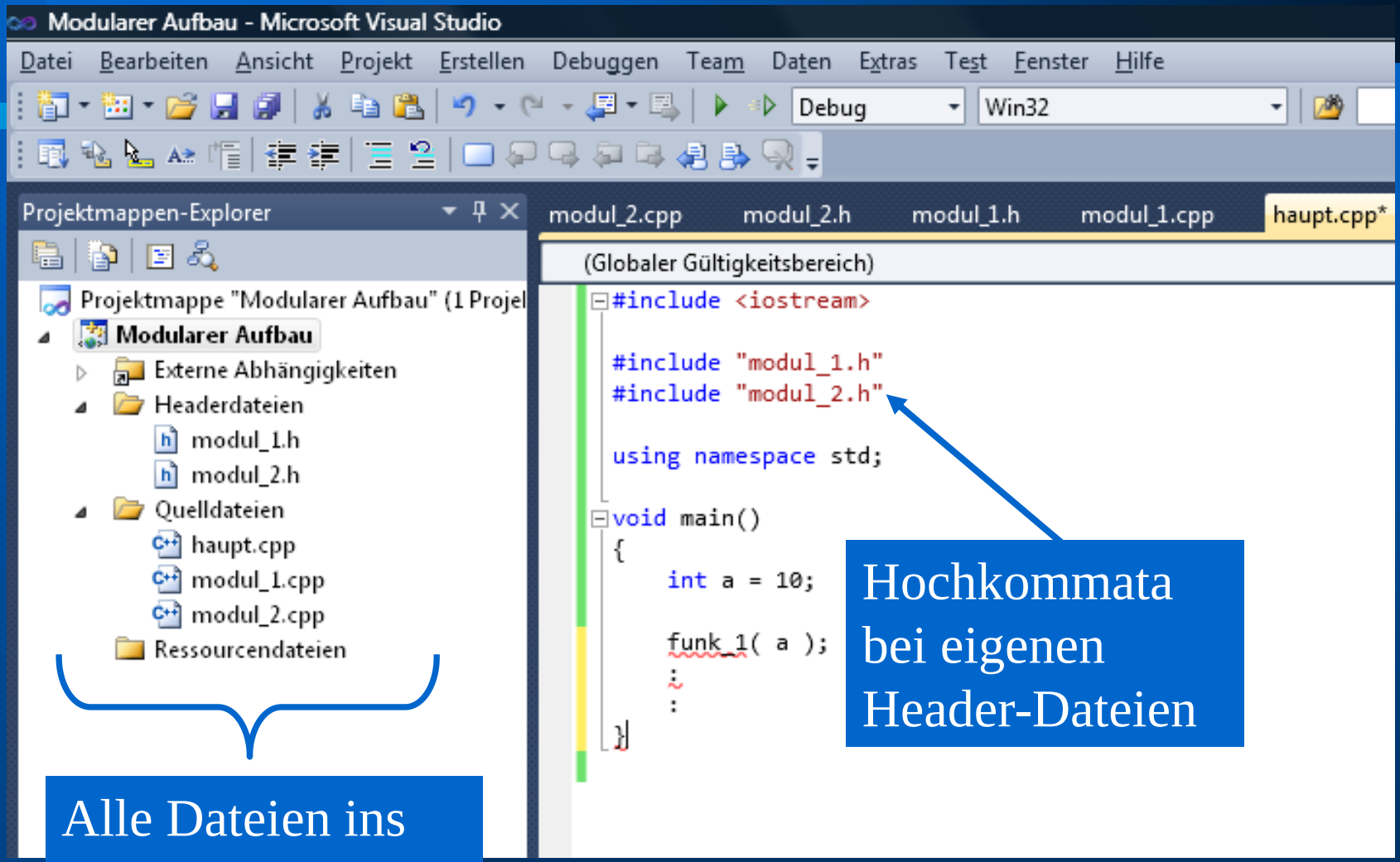
Modul_1.h

```
int funk_1 ( int );
float funk_2 ( char );
.............
```

Haupt.cpp

```
#include "Modul_1.h"
void main ( )
{ ..... }
```

# Umsetzung im Visual Studio




Alle Dateien ins Projekt einbinden

Hochkommata bei eigenen Header-Dateien

# Zusammenfassung

**Modularer Programmaufbau hat folgende Vorteile:**

1. **Der Quellcode ist übersichtlicher und besser organisiert.**
2. **Der Quellcode kann besser wiederverwendet werden. ( Zeiteinsparung, Kosteneinsparung)**
3. **Die Implementierung kann auf mehrere Programmierer verteilt werden. Es müssen nur vorher die Schnittstellen definiert werden.**

# Präprozessordirektive

Header-Dateien sollten nicht mehrfach eingebunden werden.
Sie sollten generell diesen Aufbau haben:

Modul.h

Falls Modul_h noch nicht definiert wurde, führe die nachfolgenden Anweisungen bis zum #endif aus!

```
#ifndef Modul_h       // if not defined
#define Modul_h       // define

   int funk ( int );
   float funk2 ( char );
   :
#endif
```

Alternativ:
#pragma once

# Aufgabe

Ausgangsbasis ist das bereits vorhandene Programm, welches aus drei eingegebenen Zahlen das Minimum und Maximum bestimmt.

Aufgabe 5.1 (S. 227)

Versuchen Sie alle wiederkehrenden Programmabschnitte in Funktionen auszulagern.

Bilden Sie Module für ähnliche Funktionen.