Teil 3
Modularisierung

- Vorausdeklarationen
- Aufbau von komplexeren C++-Programmen
- Includes
- Compiling
- Dateiendungen
- Compiler

Vorausdeklarationen

```
int main()
{
  print("Ergebnis", 42 / 7);
  return 0;
}

void print(string text, int number)
{
  cout << text << ": " << number << endl;
}
```

Vorausdeklarationen

```
void print(string text, int number);

int main()
{
  print("Ergebnis", 42 / 7);
  return 0;
}

void print(string text, int number)
{
  cout << text << ": " << number << endl;
}
```

Vorausdeklarationen

```
void print(string, int);

int main()
{
  print("Ergebnis", 42 / 7);
  return 0;
}

void print(string text, int number)
{
  cout << text << ": " << number << endl;
}
```

Compilation Units

print.cpp

```
void print(string text, int number)
{
  cout << text << ": " << number << endl;
}
```

main.cpp

```
void print(string, int);

int main()
{
  print("Ergebnis", 42 / 7);
  return 0;
}
```

Header / Implementierung

print.cpp

```
#include <iostream>

void print(string text, int number)
{
  std::cout << text << ": " << number << std::endl;
}
```

print.h

```
void print(string, int);
```

main.cpp

```
#include "print.h"

int main()
{
  print("Ergebnis", 42 / 7);
  return 0;
}
```

Header / Implementierung

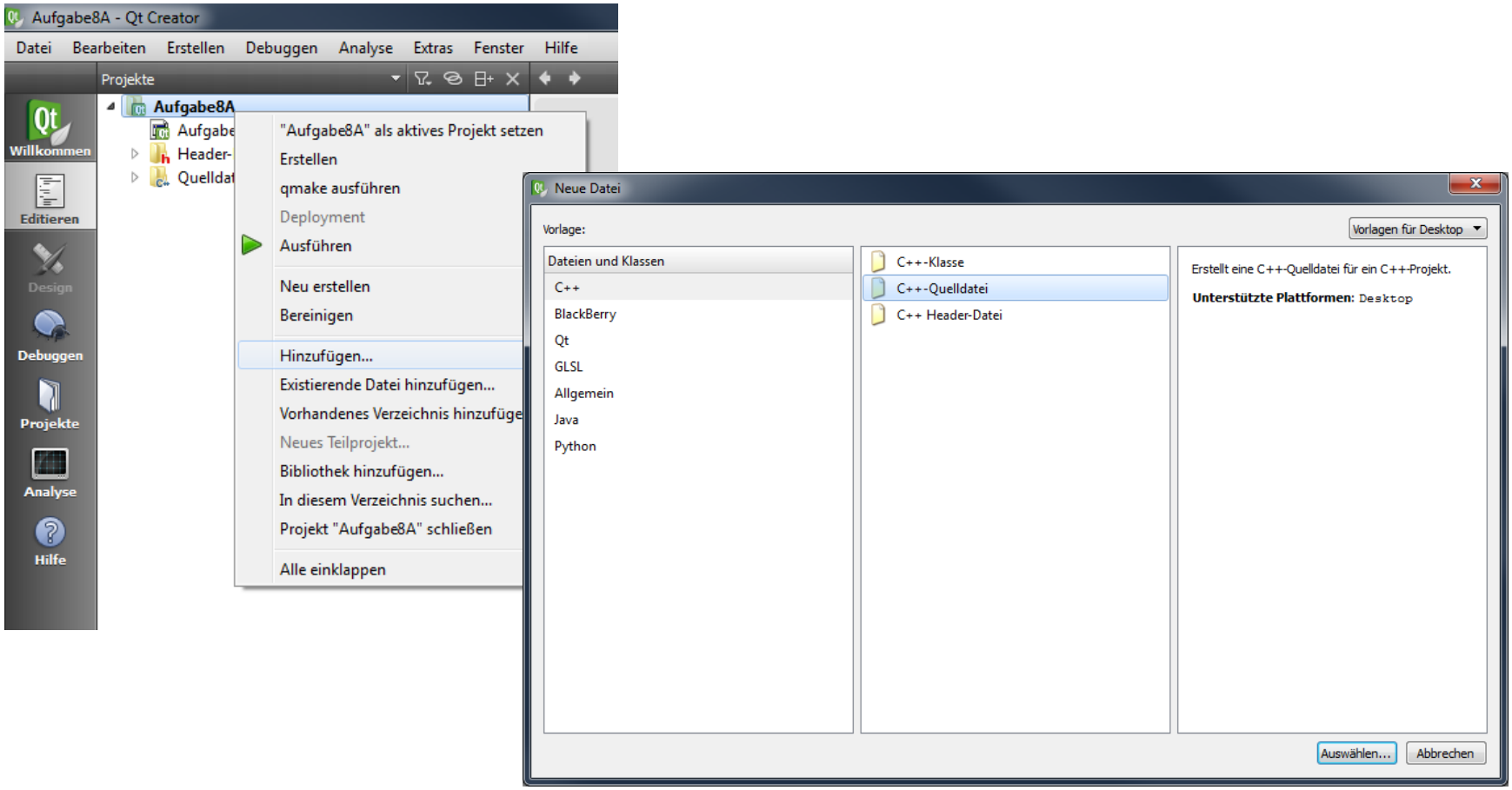
- Trennung von Interface und Implementierung bzw. von Deklaration und Definition
- Die meisten „Module“ haben je *ein* Header- und *ein* Implementierungs-File (Ausnahmen: main.cpp und Templates)
- Normalerweise inkludiert jedes Implementierungs-File seinen eigenen Header, und zwar als *erstes*

print.cpp

```
#include "print.h"
#include <iostream>

void print(string text, int number)
{
  std::cout << text << ": " << number << std::endl;
}
```

Projekt erweitern



Aufgaben

Aufgabe 8A

- die Funktionen für Potenz, Fakultät und Binomialkoeffizient sollen in eine separate Datei verschoben werden

Aufgabe 8B

- die Funktionen für Teiler und Primzahlen sollen in eine separate Datei verschoben werden

Verschachtelte Includes

one.h

```
#include "two.h"
void one();
```

two.h

```
void two();
```

one.cpp

```
void one()
{
  two(); ...
}
```

two.cpp

```
void two()
{
  ...
}
```

main.cpp

```
#include "one.h"
#include "two.h"

int main()
{
  one(); two(); ...
}
```

Verschachtelte Includes

one.h

```
void two();
void one();
```

two.h

```
void two();
```

one.cpp

```
void one()
{
   two(); ...
}
```

two.cpp

```
void two()
{
   ...
}
```

main.cpp

```
#include "one.h"
#include "two.h"

int main()
{
   one(); two(); ...
}
```

Verschachtelte Includes

one.h

```
void two();
void one();
```

two.h

```
void two();
```

one.cpp

```
void one()
{
  two(); ...
}
```

two.cpp

```
void two()
{
  ...
}
```

main.cpp

```
void two();
void one();
void two();
int main()
{
  one(); two(); ...
}
```

Include Guards

one.h

```
#ifndef ONE_H
#define ONE_H

#include "two.h"

void one();

#endif
```

two.h

```
#ifndef TWO_H
#define TWO_H

void two();

#endif
```

main.cpp

```
#include "one.h"
#include "two.h"

int main()
{
  one(); two(); ...
}
```

Aufgaben

Aufgabe 9A

- die Header-Dateien sollen mit Include-Guards versehen werden (falls noch nicht geschehen)

Aufgabe 9B

- die Header-Dateien sollen mit Include-Guards versehen werden (falls noch nicht geschehen)

Compiling / Linking

```cpp
#include <iostream>

void print(string text, int number)
{
  cout << text << ": " << number << endl;
}
```

print.cpp

```cpp
void print(string, int);
```

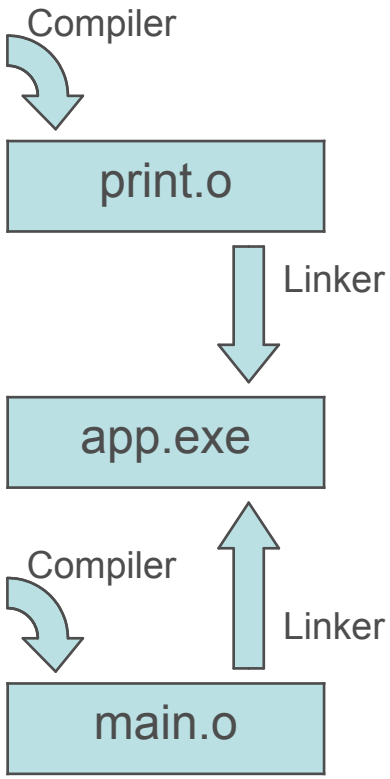
print.h

Präprozessor

```cpp
#include "print.h"

int main()
{
  print("Ergebnis", 42 / 7);
  return 0;
}
```

main.cpp



Makefiles

```
COMPILER = g++
LINKER   = g++

app.exe: main.o second.o
  $(LINKER) -o app.exe main.o second.o

main.o: main.cpp
  $(COMPILER) -c main.cpp

second.o: second.cpp
  $(COMPILER) -c second.cpp
```

erstes „Target“

```
C:\usr\x\project> make
```

Makefiles

```
COMPILER = g++
LINKER   = g++

app: main.o second.o
  $(LINKER) -o app main.o second.o

main.o: main.cpp
  $(COMPILER) -c main.cpp

second.o: second.cpp
  $(COMPILER) -c second.cpp

clean: second.cpp
  del main.o second.o
```

```
C:\usr\x\project> make clean
```

Dateiendungen

- Header
  - **.h**
  - .hpp
  - .h++ (problematisch)
  - .hxx
  - .hh
- Implementierung
  - **.cpp**
  - .c++ (problematisch)
  - .cxx
  - .cc
  - .c (für C, nicht C++)
- Projekte
  - .pro (Qt Creator)
  - .vcxproj (Visual Studio)
  - .sln (Visual Studio)
- Einstellungen
  - .pro.user (Qt Creator)
  - .suo (Visual Studio)
- Kompilat
  - .o
  - .obj
- Sonstige
  - .pdb (Debug-Info Visual Studio)

Compiler

- Microsoft Visual C++
  - proprietär
  - x86/x64
  - Windows
- C++Builder (Borland)
  - proprietär
  - x86
  - Windows, OS X
- Intel C++ Compiler
  - proprietär
  - x86/x64 (Intel / AMD)
  - Windows, OS X, Linux, Android (Intel)
- IBM XL C++
  - proprietär
  - POWER, z etc.
  - Linux, AIX, z/OS etc.
- GCC (GNU Compiler)
  - GPL
  - x86/x64, ARM, PowerPC, MIPS, Atmel AVR, Motorola 68000, Alpha, VAX, PDP-11 etc.
  - Linux, BSD, Android, iOS etc.
- MinGW
  - „Minimalist GNU for Windows“
- Clang (LLVM)
  - NCSA
  - x86/x64, ARM, MIPS, PowerPC, Nvidia, Qualcomm, SPARC, z etc.
  - OS X, FreeBSD, Linux, MINIX etc.
- Comeau, Digital Mars usw.