Einführung in die Programmierung

Grundbegriffe

Prof. Dr. Peter Jüttner

Inhalt

- **Grundbegriffe**
  - Grundstruktur eines C Programms
  - Variable
  - Konstante
  - Typen
  - Ein-/Ausgabe
  - Operatoren
  - Kontrollstrukturen
  - Vektoren (Arrays, Felder)
  - Typen
  - C-Präprozessor

C-Präprozessor

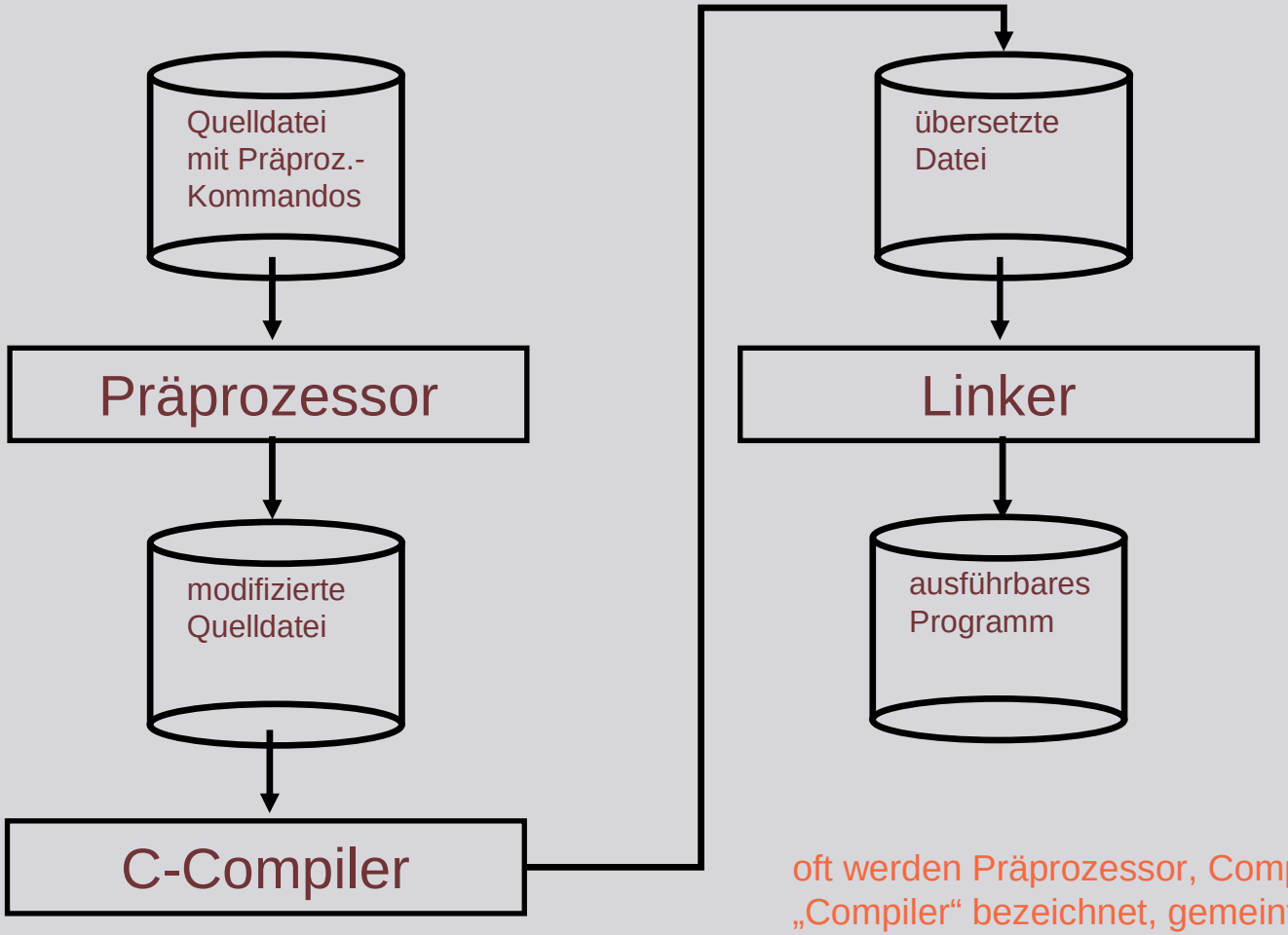
Hochschule Deggendorf

Der C-Präprozessor

- Programm
- kein Teil des C-Compilers
- Teil einer C-Programmierumgebung
- bearbeitet und modifiziert ggf. eine Quelldatei bevor der Compiler die modifizierte Quelldatei übersetzt
- modifizierte Quelldatei ist i. d. R. temporär
- hat standardisierte Kommandos, die aber nicht zur Sprache C gehören

C-Präprozessor

Präprozessor in der Entwicklungsumgebung



oft werden Präprozessor, Compiler und Linker als „Compiler“ bezeichnet, gemeint ist aber ein C-Entwicklungssystem

C-Präprozessor

Hochschule Deggendorf

Arbeitsweise

- reagiert auf Kommandos, die mit # beginnen
- Kommandos werden ausgeführt, danach entfernt
- Dateibearbeitung rein textuell
- von oben nach unten
- ohne „Rücksicht“ auf C-Syntax
- in allen Dateien, die im Präprozessorlauf bearbeitet werden

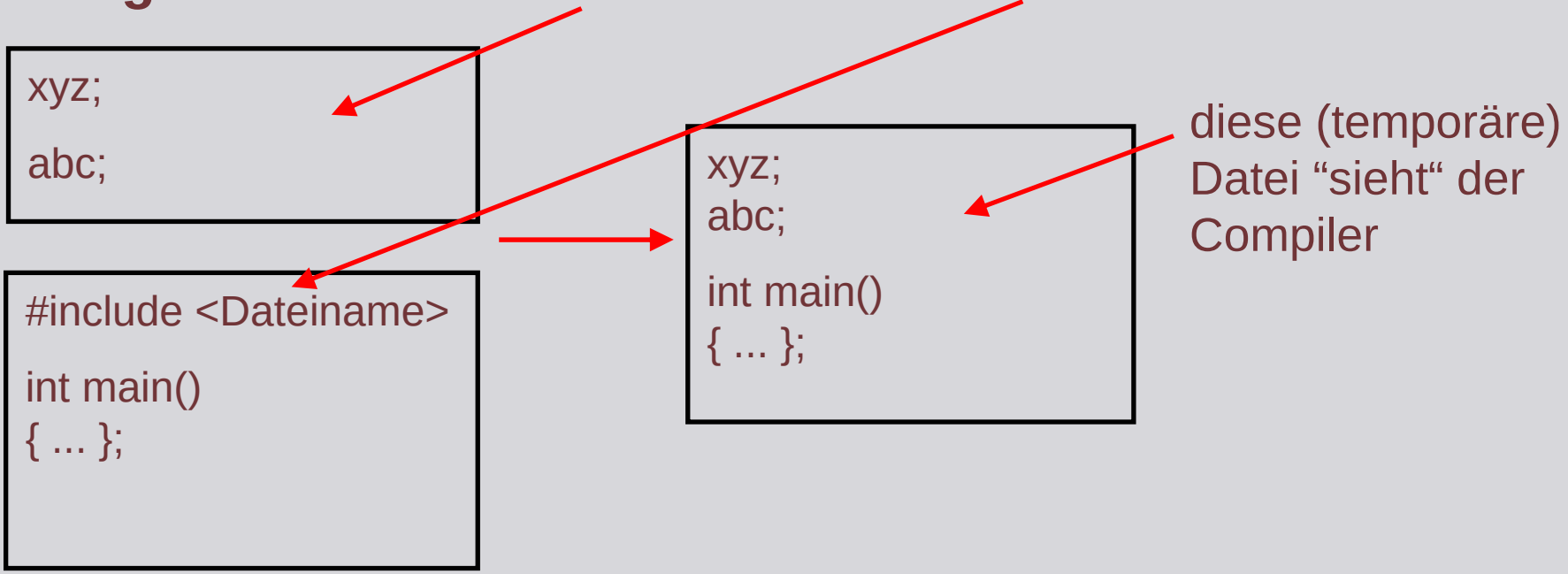
C-Präprozessor

Hochschule Deggendorf

Schon bekannte Präprozessor Funktion ...

#include <*Dateiname*>

Einfügen der Datei *Dateiname* an diese Stelle



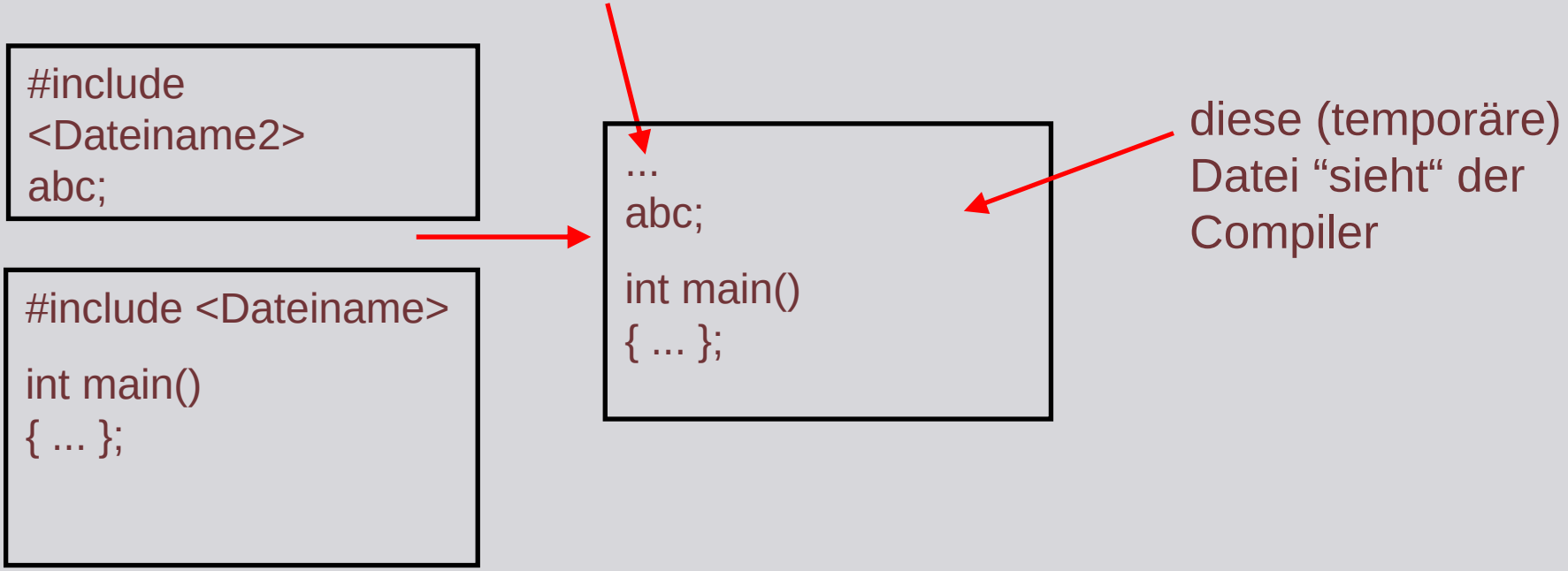
C-Präprozessor

Hochschule Deggendorf

Schon bekannte Präprozessor Funktion ...

#include in includierten Dateien möglich

Hier steht die Datei *Dateiname2*



C-Präprozessor

Hochschule Deggendorf

Präprozessorkommandos – Übersicht

- #include
- #define
- #ifdef
- #ifndef
- #else
- #endif

C-Präprozessor

Hochschule Deggendorf

#include-Kommando

- Kopiert die angegebene Datei an die Stelle der Direktive
- #include <datei> sucht in den Verzeichnissen der Entwicklungsumgebung nach datei*)
- #include “datei“ sucht in den Verzeichnissen in denen der Programmierer seine Quellen hat*)

*) diese Verzeichnisse sind per Option einstellbar in der Entwicklungsumgebung

C-Präprozessor

Hochschule Deggendorf

#define-Kommando

- definiert symbolische Konstante (Konstante, die keinen Speicherplatz belegen)

- Syntax: #define TEXT Ersatztext*)

  sucht “TEXT“ im folgenden Quellcode und ersetzt in durch „Ersatztext“

*) TEXT wird “traditionell“ in Großbuchstaben geschrieben

C-Präprozessor

#define-Kommando - Beispiel:

```
#include <stdio.h>
#include <stdlib.h>
#define PI 3.1415926

int main()
{
   float r;
   printf("Kreisbrechnung - Bitte Radius
eingeben:\n");
   scanf("%f",&r);

   printf("Kreisummfang: %.2f\n", 2*r*PI);
   printf("Kreisflaeche: %.2f\n", r*r*PI);

   system("PAUSE");
   return 0;

}
```

nach Präprozessorlauf

```
… hier stehen die includierten Dateien …

int main()
{
   float r;
   printf("Kreisbrechnung - Bitte Radius
eingeben:\n");
   scanf("%f",&r);

   printf("Kreisummfang: %.2f\n", 2*r* 3.1415926);
   printf("Kreisflaeche: %.2f\n", r*r* 3.1415926);

   system("PAUSE");
   return 0;

}
```

symbolische Konstante “PI“ ersetzt durch “3.1415926“

C-Präprozessor

Hochschule Deggendorf

#define-Kommando – Zweck

- konsistente Änderungen
- übersichtlicher Code
- bedingte Compilierung

```
#include < ... >
#define ANZAHL 10

int main()
{
   float z = ANZAHL;
   ...
   for (int i=0; i< ANZAHL; i++)
   { ... z = ... ; };
    ...
    if (z = ANZAHL
    ...
}
```

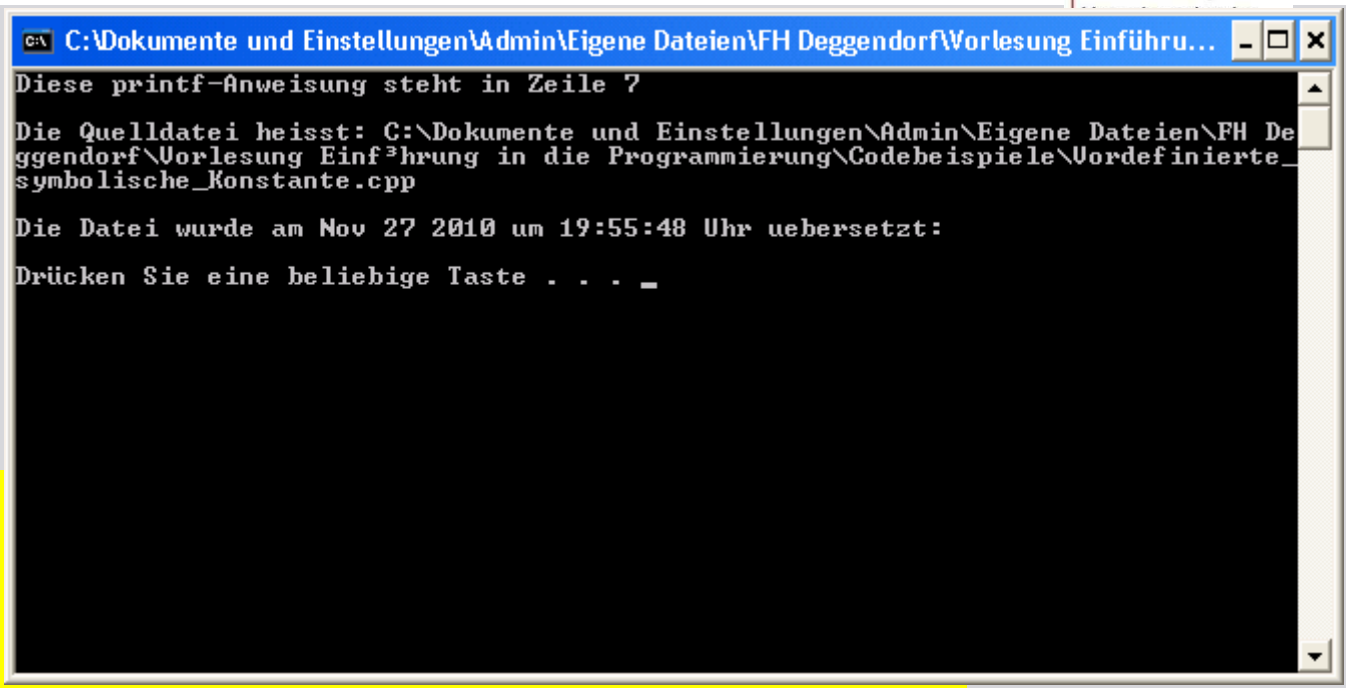
C-Präprozessor

Hochschule Deggendorf

#define-Kommando – vordefinierte Symb. Konstante

- __LINE__ hier wird die Zeilennummer eingesetzt, in der das Symbol steht
- __FILE__ hier wird der Dateinamen der Programmquelle eingesetzt
- __DATE__ hier wird das Datum der Übersetzung (als Zeichenkette) eingesetzt
- __TIME__ hier wird durch die Zeit der Übersetzung (als Zeichenkette) eingesetzt

C-Präprozessor



#define

- vordefinierte Symb. Konstante

Beispiel:

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{

  printf("Diese printf-Anweisung steht in Zeile %d\n\n", __LINE__);

  printf("Die Quelldatei heisst: %s\n\n", __FILE__);

  printf("Die Datei wurde am %s um %s Uhr uebersetzt:\n\n", __DATE__ , __TIME__ );

  system("PAUSE");
  return 0;
}
```

C-Präprozessor

Hochschule Deggendorf

#define-Kommando – Makros

- definiert parametrierte kurze Codesequenzen
- Syntax: #define MAKRONAME(Parameter$_1$, Parameter$_2$,…, Parameter$_n$) Ersatztext*)

  sucht “MAKRONAME“ im folgenden Quellcode und ersetzt in durch „Ersatztext“

*) MAKRONAME wird “traditionell“ in Großbuchstaben geschrieben

C-Präprozessor

#define-Kommando – Makros Beispiel

```
#include < ...>
#define ABS(x) ((x)>0) ? (x) : (-x)
#define MAX(x,y) ((x)>(y)) ? (x) : (y)

int main()
{
   float f, abs_f;
   float a,b,max;
   printf("Absolutbetrag - Bitte Zahl eingeben:\n");
   scanf("%f",&f);
   abs_f = ABS(f);

   printf("Absolutbetrag von %.2f ist %.2f\n", f, abs_f);

   printf("Ermittlung Maximum\n");
   printf("Bitte 1. Zahl eingeben:\n");
   scanf("%f",&a);
   printf("Bitte 1. Zahl eingeben:\n");
   scanf("%f",&b);
   max = MAX(a,b);

   printf("Maximum von %.2f und %.2f ist %.2f\n", a, b,
max);

}
```

C-Präprozessor

Hochschule Deggendorf

#define-Kommando – Makros - Beispiel

- **Makrodefinition**

  #define MAX(x,y) ((x)>(y))?(x):(y)

- **Makroaufruf**

  … = MAX(a, b);

- **expandiert**

  … = (a>b)?(a):(b);

Textuelle Ersetzung der formalen Parameter durch die Aufrufparameter

C-Präprozessor

Hochschule Deggendorf

#define-, #ifdef, #ifndef, #else-, #endif- Kommandos – bedingte Compilierung

- **Codeteile (bedingt) von Compilierung ausschließen**

  ➔ Variantenbildung

  ➔ Verhindern von mehrfachem Includieren einer Headerdatei

C-Präprozessor

Hochschule Deggendorf

#define-, #ifdef, #ifndef, #else-, #endif- Kommandos – bedingte Compilierung

➔**Variantenbildung**

- ➔Code für unterschiedliche Microcontroller
- ➔Unterschiedliche Programmvarianten

C-Präprozessor

Hochschule Deggendorf

#define-, #ifdef, #ifndef, #else-, #endif- Kommandos – bedingte Compilierung

- #ifdef KONSTANTE - folgender Code wird nur compiliert, wenn KONSTANTE per #define definiert
- #ifndef KONSTANTE - folgender Code wird nur compiliert, wenn KONSTANTE ***nicht*** per #define definiert
- #endif - schließt vorhergehendes #ifdef oder #ifndef ab
- #else – Alternative zu vorhergehenden #ifdef oder #ifndef
- diese Präprozessoranweisungen dürfen auch geschachtelt werden

C-Präprozessor

#define-, #ifdef, #ifndef, #else-, #endif- Kommandos – bedingte Compilierung

- **Variantenbildung**

Symbolische Konstante VARIANTE1 kann ggf. extern (Compileroption*)) definiert werden

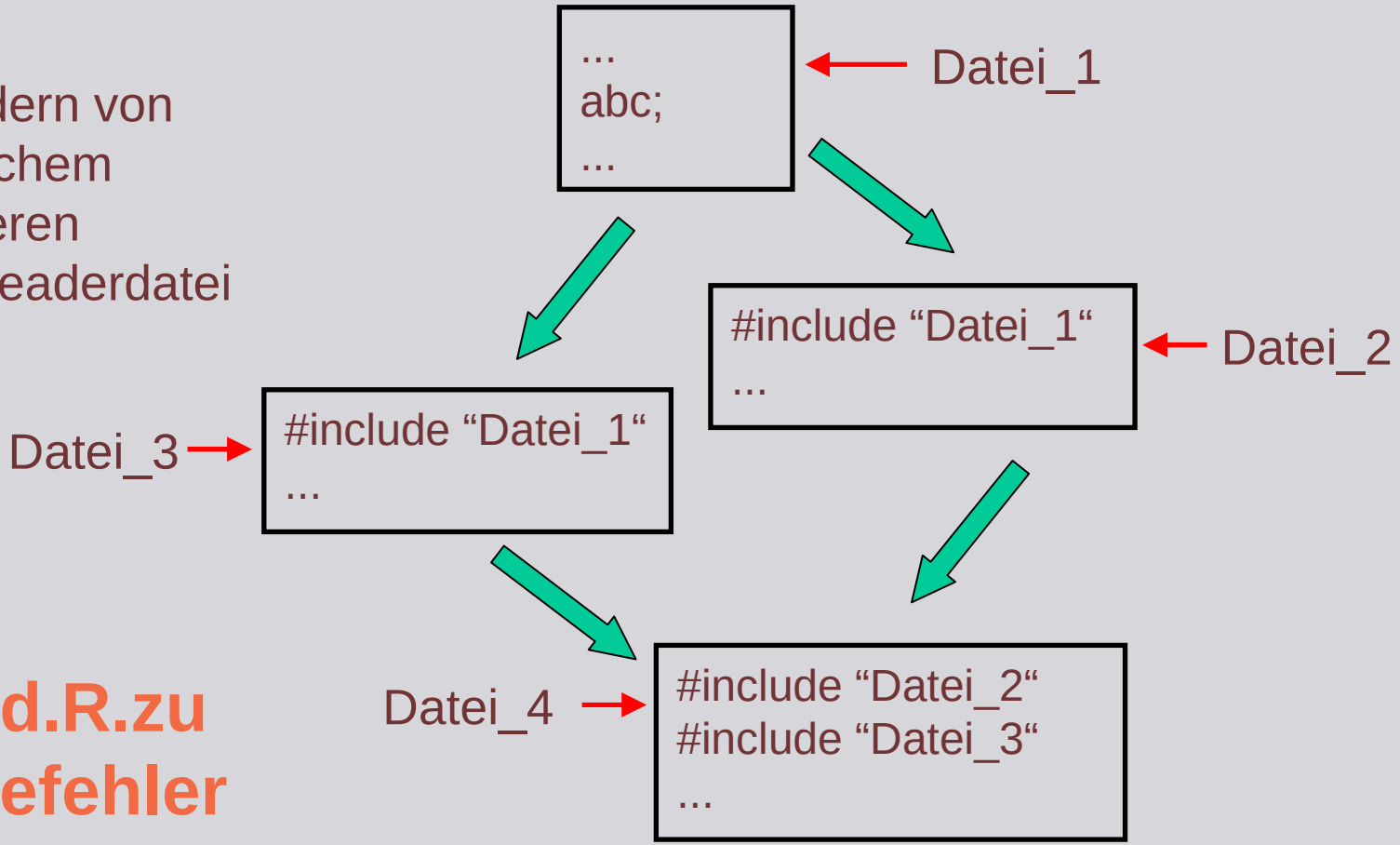
```
#include < ... >

int main()
{
   #ifdef VARIANTE1

   ... hier steht der Code für Variante 1 ...

   #else

   ... hier steht der Code für Variante 2 ...

   #endif

   ... hier steht der Code für beide Varianten
...

}
```

*) –D VARIANTE1 bei Bloodshed

C-Präprozessor

#define-, #ifdef, #ifndef, #else-, #endif- Kommandos – bedingte Compilierung

- Verhindern von mehrfachem Includieren einer Headerdatei



Führt i.d.R.zu Compilefehler

C-Präprozessor

#define-, #ifdef, #ifndef, #else-, #endif- Kommandos – bedingte Compilierung

- Verhindern von mehrfachem Includieren einer Headerdatei

➔ **Lösung**

zu includierende Datei datei.h →

```
#ifndef DATEI_H
#define DATEI_H
  ...
  hier steht der Code der Datei
  ...
#endif
```

Inhalt wird nur einmal compiliert, auch wenn Datei mehrfach includiert

Präprozessor

Hochschule Deggendorf

Zum Schluss dieses Abschnitts ...

Noch Fragen ??