Hochschule **Rosenheim**
University of Applied Sciences

Prozedurale Programmierung

Präprozessor

Hochschule Rosenheim - University of Applied Sciences

WS 2018/19

Prof. Dr. F.J. Schmitt

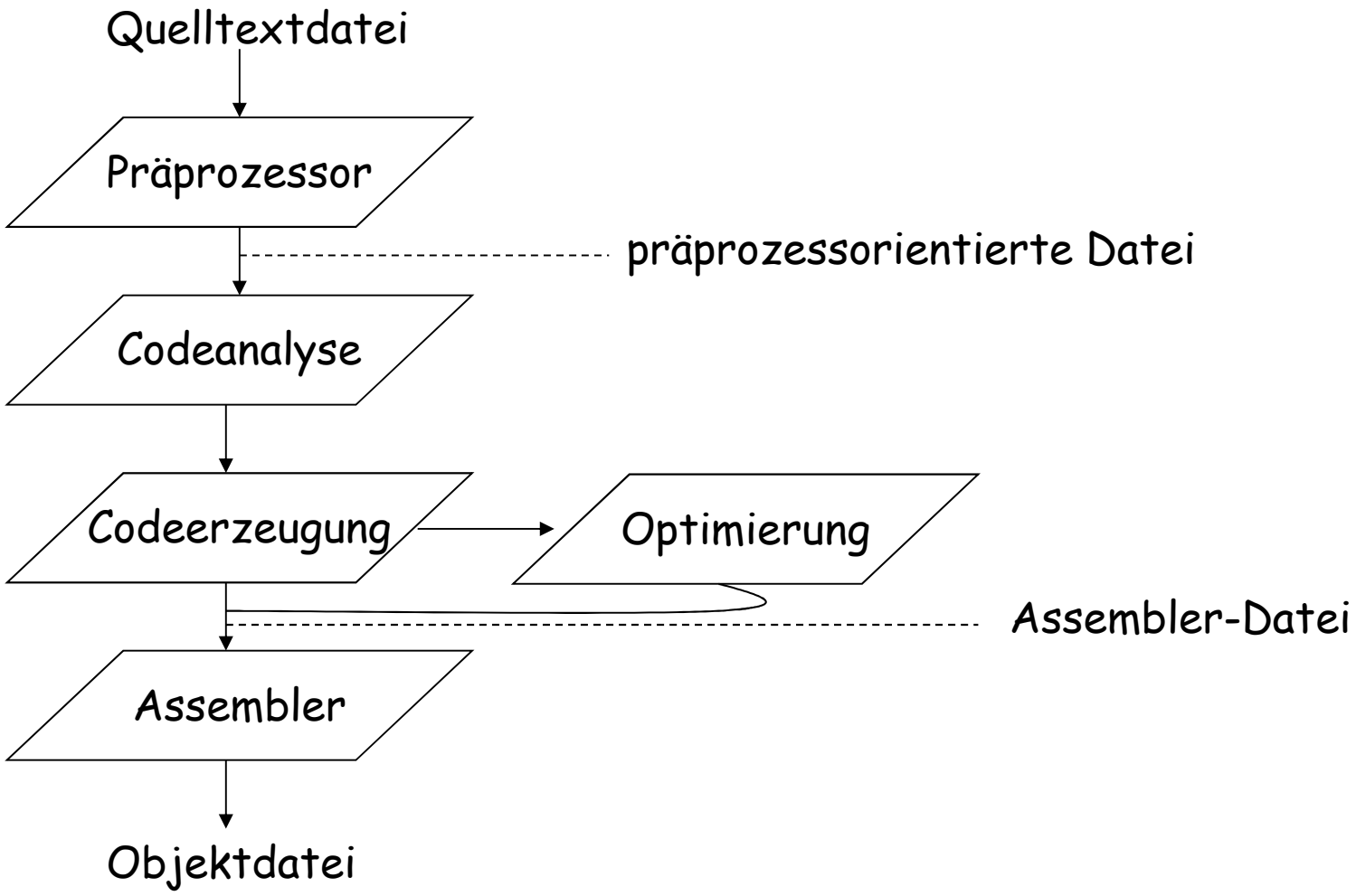
Kapitel 19 Der Präprozessor

- 19.1 Grundlegendes
- 19.2 Überblick über wichtige Präprozessor-Anweisungen

Was ist der Präprozessor? (1)

- Präprozessor ist ein Textersetzer, der vor dem eigentlichen Übersetzungsvorgang aufgerufen wird
  - Jeder C-Compiler ist damit ausgestattet
- Ablauf:
  - Lesen der Quelltext-Datei
  - Ausführen aller Präprozessoranweisungen
  - Modifikationen im Programmtext
    - Texte können eingefügt, ersetzt, verändert oder ausgeschnitten werden
    - Entfernung aller Kommentare
  - Modifizierter Code wird an Compiler weitergereicht

Was ist der Präprozessor? (2)



Was ist der Präprozessor? (3)

- Sämtliche Präprozessor-Anweisungen beginnen mit dem Symbol #
- Präprozessor durchsucht den Quelltext nach diesen Anweisungen und führt sie aus
- Präprozessor-Anweisungen werden aus dem Text entfernt
- Präprozessor berücksichtigt weder Syntax noch Semantik von C
  - Gefahr der Verstümmelung des Programmtexts
  - Vorsichtige Anwendung notwendig!

Quellcode → Präprozessor

Originalcode (komplett)

```
#include <stdio.h>  // Standard Input/ Output  z.B. scanf, printf
#define TEXT "Dies ist ein Text\n"

int main(void)
{
  const float zahl = 5.2f;  // definiert eine Konstante

  printf ("Zahl: %10.2f\n", zahl);  /* Ausgabe */
  printf(TEXT);

  return(0);
}
```

nach Präprozessor
(kleiner Ausschnitt)

```
…
#line 283 "c:\\program files (x86)\\microsoft visual studio 10.0\\vc\\include\\stdio.h"
 __declspec(dllimport) int __cdecl _pclose(   FILE * _File);
 __declspec(dllimport) FILE * __cdecl _popen(   const char * _Command,    const char * _Mode);
 __declspec(dllimport) int __cdecl printf(    const char * _Format, ...);
…

int main(void)
{
  const float zahl = 5.2f;
  printf ("Zahl: %10.2f\n", zahl);
  printf("Dies ist ein Text\n");

  return(0);
}
```

Überblick Anweisungen

- Wichtige Präprozessor-Anweisungen sind:
  - `#include`-Anweisung
  - `#define`-Anweisung
  - `#if`-Anweisung
  - Anweisungen `#ifdef` und `#ifndef`

`#include`-Anweisung (1)

- Einfügen von Header-Dateien in den Programmtext
  - Enthalten Informationen, die in mehreren C-Dateien benötigt werden
    - Definitionen von Präprozessorkonstanten
    - Typ- und Strukturdefinitionen
    - Funktionsdeklarationen
    - Dürfen keine ausführbaren Programmtexte enthalten

#include-Anweisung (2)

- Zwei unterschiedliche Typen von Header-Dateien
  - Standard-Header-Dateien
    - `#include <stdio.h>`
    - wird im Standard-Installationsverzeichis des C-Compilers gesucht
  - Selbst definierte Header-Dateien
    - `#include "header-datei.h"`
    - Dateiname muss relativ zum Projektverzeichnis angegeben werden
    - Pfadangabe in Anführungszeichen

#define-Anweisung (1)

- Definition von Präprozessorkonstanten
  - `#define HALLO "Hallo Welt!"`
  - `#define NUM  100`

Kein Strichpunkt!

- Zielsetzung:
  - Zusammenfassung von Werten, die häufig im Programmtext verkommen, an einer Stelle
  - Erhöhung der Übersichtlichkeit und Wartbarkeit des Quelltextes

#define-Anweisung (2)

- Präprozessor führt keine syntaktischen Prüfungen durch
  - `#define NUM 5x&;*`

Fehler erst beim Compilieren in der Zeile und Datei, in der NUM verwendet wird!

`#define`-Anweisung (3)

- Definition von sog. Makros
  - Automatisches Einsetzen häufig verwendeter Befehle
  - Durchführung von Textersetzungen

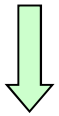
- Beipsiel:
  - Definition des Makros `ERRLOG` mit dem Parameter `text`

```
#define ERRLOG(text)  fprintf(stderr, text)
```

#define-Anweisung (4)

- Fortsetzung Beipsiel:
  - Aufruf des Makros `ERRLOG`

```
ERRLOG("Eingabefehler!");
```




```
fprintf(stderr, "Eingabefehler!");
```

Präprozessormakros sind keine Funktionen. Es werden keine Typüberprüfungen durchgeführt!

#define-Anweisung (5)

- Beachte:
  - Bei der Definition des Makros darf zwischen dem Namen des Makros und der öffnenden Klammer kein Abstand eingefügt werden

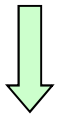
```
#define ERRLOG (text)  fprintf(stderr, text)
```

Präprozessor ersetzt jedes Vorkommen von ERRLOG durch folgenden Text

```
(text)  fprintf(stderr, text)
```

#define-Anweisung (6)

- Beachte:
  - `#define ERRLOG (text) fprintf(stderr, text)`
  - Aufruf: `ERRLOG("Eingabefehler!");`



Präprozessor

```
(text) fprintf(stderr, text)("Eingabefehler!");
```

  - Abstand zwischen den Namen und öffnenden Klammer führt dazu, dass Makro nicht erkannt wird!

#define-Anweisung (7)

- Weitere Beispiele Makros:

```
#define MIN(a,b) ((a) < (b) ? (a) : (b))

#define MAX(a,b) ((a) > (b) ? (a) : (b))

#define ABS(x) ((x) < 0 ? (-(x)) : (x))
```

#if-Anweisung (1)

- Dient zum Ein- und Ausblenden von Textpassagen
- wird mit dem Befehl `#endif` abgeschlossen

```
#if 0
//Programmtext
#endif
```

Ausblenden des Programmteils

```
#if 1
//Programmtext
#endif
```

Einblenden des Programmteils

#if-Anweisung (2)

- Ein- und Ausblenden von alternativen Textpassagen

```
#if 1
//Programmtext1
#else
//Programmtext 2
#endif
```

Einblenden von Programmtext 1

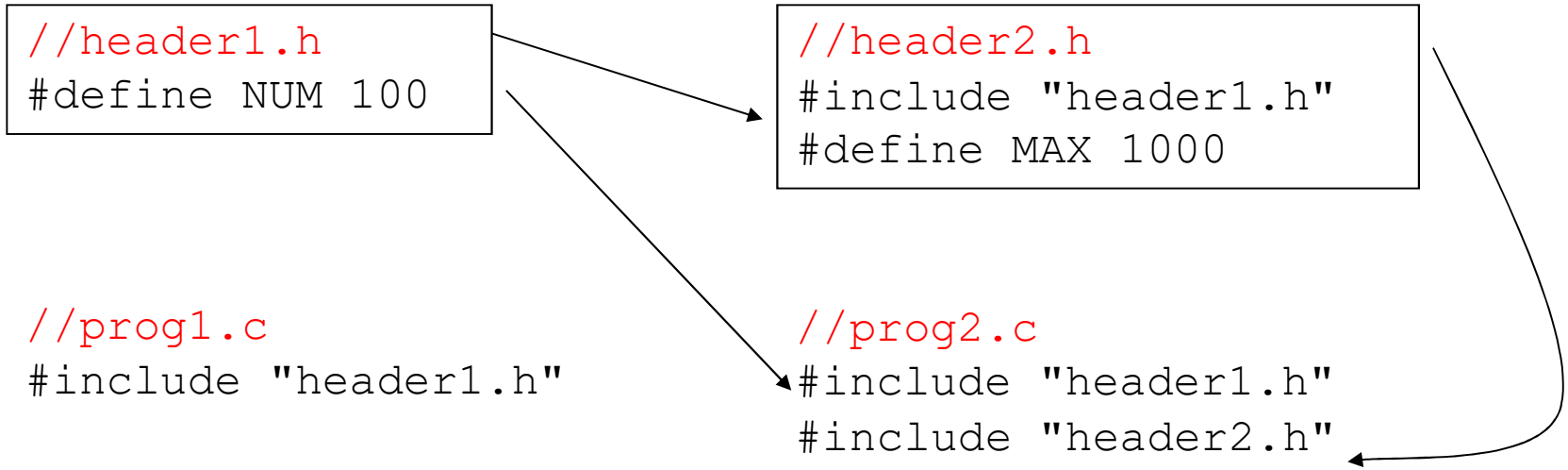
Ausblenden von Programmtext 2

#if-Anweisung (3)

- Anwendungsbeispiel 1:
  - Ein- bzw. Ausblenden vom Programmierer zusätzlich eingefügte Ausgaben zur Fehlersuche
  - ```
    #define DEBUG 0
    //...
    //...
    #if DEBUG
    printf("wert = %ld\n",wert);
    printf("Schleifenzaehler i = %ld\n",i);
    #endif
    //...
    ```
  - Wird Präprozessorkonstante auf 1 gesetzt => zusätzliche Informationen über innere Zustände – hier Variablenwerte – werden ausgegeben
    ```

Anweisungen `#ifdef` und `#ifndef` (1)

- Problem: Header-Dateien dürfen nicht mehrmals eingefügt werden



Präprozessorkonstante NUM darf nur einmal definiert werden!

genauso: alles andere! (Typdefinitionen, structs, Funktionsdeklarationen, …)

Anweisungen `#ifdef` und `#ifndef` (2)

- Ein- und Ausblenden von Textpassagen in Abhängigkeit davon ob eine Präprozessorkonstante definiert ist

```
//header1.h
#ifndef HEADER1_H
#define HEADER1_H

#define NUM 100

#endif
```

```
//header2.h
#ifndef HEADER2_H
#define HEADER2_H
#include "header1.h"
#define MAX 1000

#endif
```

```
//prog2.c
#include "header1.h"
#include "header2.h"
```

Nach erstem Laden der Header-Datei wird zugehörige Präprozessorkonstante definiert => verhindert erneutes Laden

Weitere Anweisungen

- #undef
  - vorheriges #define wird aufgehoben
- #error
  - gibt eine Fehlermeldung beim Übersetzen aus
- #pragma
  - Compiler-Anweisungen
  - Beispiele:
    - Abschalten von „scanf()-Warnungen“:
      #pragma warning(disable : 4996)
    - OpenMP (Open Multi-Processing Bibliothek)
      parallele for-Anweisung:
      #pragma omp parallel for private(i)

Zusammenfassung

- C-Präprozessor
  - läuft vor dem eigentlichen Compiler
  - Anweisungen werden mit # eingeleitet
  - kein Semikolon am Zeilenende
- wichtige Anweisungen
  - #include Einfügen von Header Dateien
  - #define Definition von Konstanten oder Makros
  - #if, #else, #endif (bedingtes) Auskommentieren von Quellcode
  - #ifndef, #ifdef Prüfen, ob NAME definiert wurde
- Header Dateien
  - immer mit #ifndef, #define, #endif Klammern