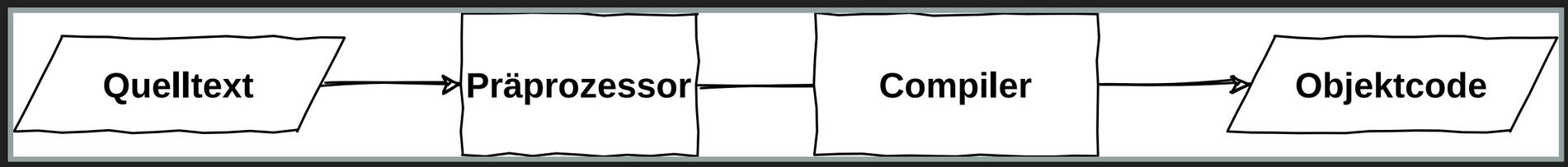
KONZEPTE SYSTEMNAHER PROGRAMMIERUNG

Technische Hochschule Mittelhessen

Andre Rein

– Präprozessor –

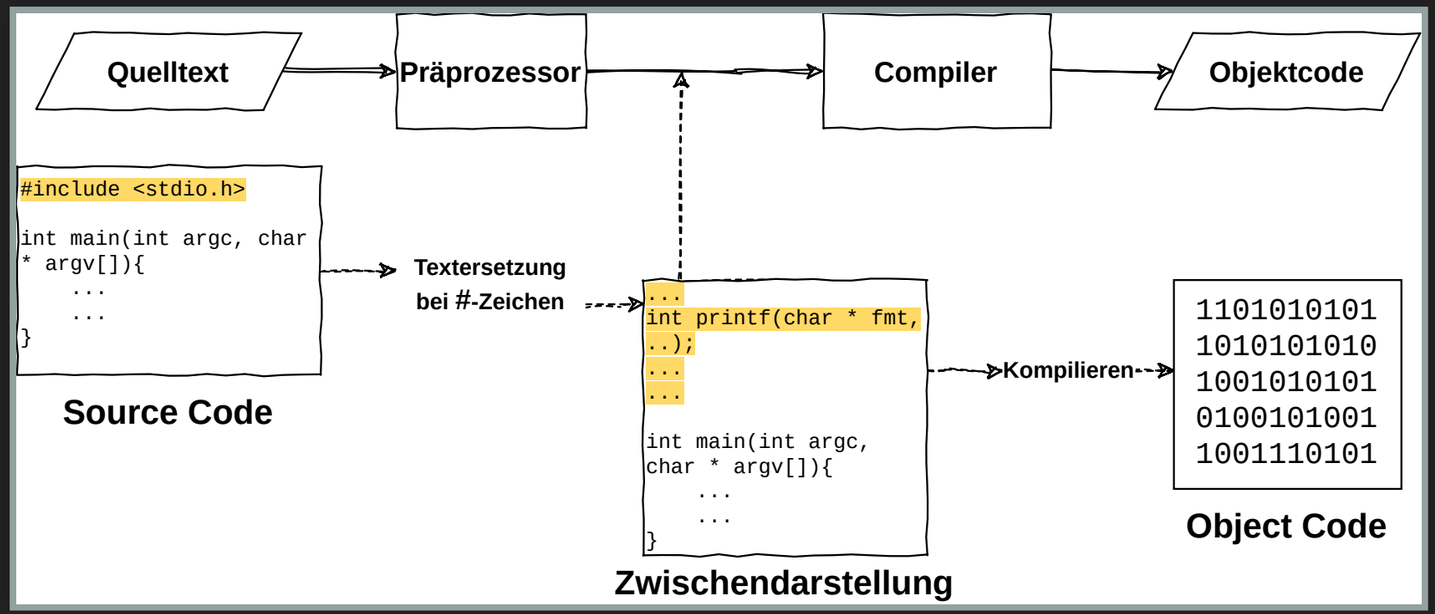
PRÄPROZESSOR



Der Präprozessor ist ein implizierter Teil des Kompiliervorgangs

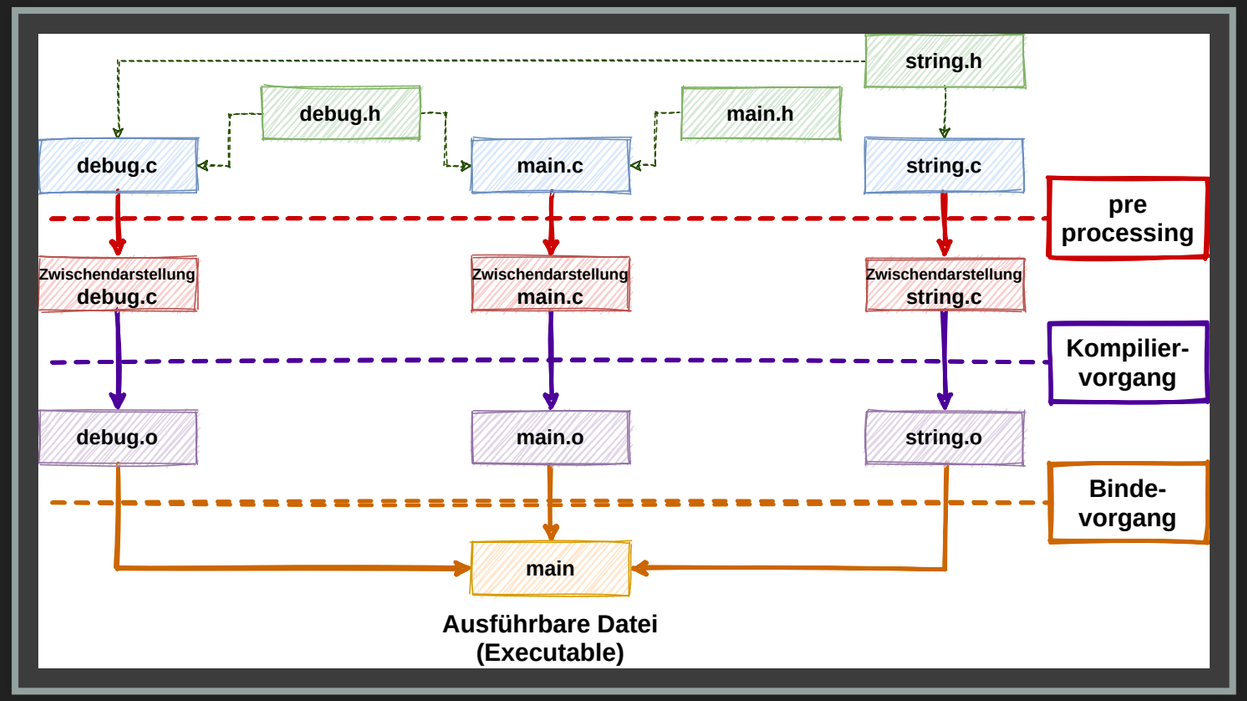
- Beim sog. *preprocessing* wird der Quelltext von einem Präprozessor analysiert und Textersetzungen vorgenommen.
- Hierbei gibt es verschiedene Arten der Analyse/Texterzsetzung. U.a.:
  1. Einschließen/Inkludieren von Header Dateien `#include`.
  2. Definition von Makros `#define` und spätere Ersetzung im Quelltext, wenn das definerte Makro (also der Makroname) im Quelltext wiedergefunden wird.
  3. Bedingt Kompilierung `#ifdef, #ifndef/ ...`.
- Diese mit einem Lattenkreuz `#` (engl. hash) beginnenden "*Präprozessor-Anweisungen*" nennt man **Direktiven**

PRÄPROZESSOR: EINSCHLUSS HEADER-DATEIEN



- Bei dem Inkludieren von Header Dateien, fügt der Präprozessor die Dateiinhalte der mittels `#include` angegebene Dateien, in die jeweilige Datei (die die `#include`-Direktive enthält) ein.
  - Dies ist eine einfache Textersetzung!
- Nach Abschluss aller Ersetzungen durch den Präprozessor, entsteht eine Zwischendarstellung, die dann als Eingabe zum eigentlichen Kompiliervorgang verwendet wird.
  - Mit dem Befehl `gcc -E <dateiname.c>` kann man sich diese Ausgabe erzeugen und ansehen.

PRÄPROZESSOR: EINSCHLUSS HEADER-DATEIEN



Beispielhafter Auszug main.c

```
#include <string.h> 1
#include "debug.h"  2
#include "main.h"   2
...
```

1. Datei `string.h` wird in Standard-Inklusion-Pfaden gesucht und verwendet.
   Z.B. Linux: `/usr/include`, `/usr/local/include`
2. Dateien `debug.h` und `main.h` werden zuerst im aktuellen Verzeichnis, danach in Standard-Inklusion-Pfaden gesucht und verwendet.

- Erzeugung von Objektcode `.o` mit `gcc -c <dateiname.c>` also z.B.
  - `gcc -c main.c` → `main.o`

PRÄPROZESSOR: TEXTERSETZUNG DURCH MAKROS

- Symbolische Namen für Konstante Werte mit `#define`-Direktive
  - `#define ANTWORT_AUF_ALLE_FRAGEN 42`

```c
#define ANTWORT_AUF_ALLE_FRAGEN 42 (1)

int main (int argc, char *argv[]) {
    int x = 5;
    if (x == ANTWORT_AUF_ALLE_FRAGEN) { (2)
        printf("Das ist korrekt!\n")
    } else {
        printf("Leider falsch!\n")
    }
    return 0;
}
```

1. Definition des Makros. *Laut Konvention alles in Großbuchstaben!*
2. Textersetzung → `if(x == 42){`

Durch die Verwendung von symbolischen Konstanten kann bestimmten Werten "Bedeutung" verliehen werden. Dem Compiler ist es natürlich vollkommen egal ob so etwas benutzt wird, es hilft jedoch dem Programmierer eine bessere Übersicht bzw. ein besseres Verständnis des Codes zu erzeugen!

PRÄPROZESSOR: INLINING

Inlining → Definition von einfachen und wiederkehrenden Rechnungen oder Umformungen mittels `#define`-Direktive

```c
#define FUNC(x) (3*(x)+1) 1

int main (int argc, char *argv[]) {
    int a,b;
    int c = FUNC(a)+FUNC(b); 2
    //Ergibt: int c = (3*(a)+1) + (3*(b)+1) 3
    return 0;
}
```

1. Definition des Makros `FUNC(x)`
2. 2 Fache Verwendung
3. Expansion: Ergebnis der Textersetzung!

Der Präprozessor ist eine reine Textersetzung, es muss vollständig geklammert werden. Der Präprozessor kennt keine Prioritäten, wie z.B. Punkt vor Strich-Rechnung.

- Beispiel 1:
  - `#define FUNC(x) (3*(x)+1)` → `FUNC(4+1)` → `(3*(4+1)+1)` → 16 → OK
  - `#define FUNC(x) (3* x +1)` → `FUNC(4+1)` → `(3*4+1+1)` → 14 → Nicht OK!
- Beispiel 2:
  - `#define ADD1(x) ((x)+1)` → `2*ADD1(3)` → `2*((3)+1)` → 8 → OK!
  - `#define ADD1(x) x+1` → `2*ADD1(3)` → `2*3+1` → 7 → Nicht OK!

Generell gilt: **Alles Klammern**!

PRÄPROZESSOR: BEDINGTE KOMPILIERUNG

Datei main.c

```c
#include <stdio.h>
#define DEBUG

int main (int argc, char *argv[]) {
#ifdef DEBUG
        printf("Debugausgabe ... \n");
#endif
        printf("Normale Ausgabe \n");
    return 0;
}
```

```
$ gcc -o main main.c
$ ./main
Debugausgabe ...
Normale Ausgabe ...
$
```

Datei main.c

```c
#include <stdio.h>
//#define DEBUG

int main (int argc, char *argv[]) {
#ifdef DEBUG
        printf("Debugausgabe ... \n");
#endif
        printf("Normale Ausgabe \n");
    return 0;
}
```

```
$ gcc -o main main.c
$ ./main
Normale Ausgabe ...
$
```

PRÄPROZESSOR: BEDINGTE KOMPILIERUNG

```c
#include <stdio.h>
#define LINUX

int main (int argc, char *argv[]) {
#ifdef LINUX
        ...
        ...
        ...
#else
        ...
        ...
        ...
#endif
    return 0;
}
```

- Ein weiteres Beispiel ist hierbei die bedingte Kompilierung für bestimmte Betriebssysteme.

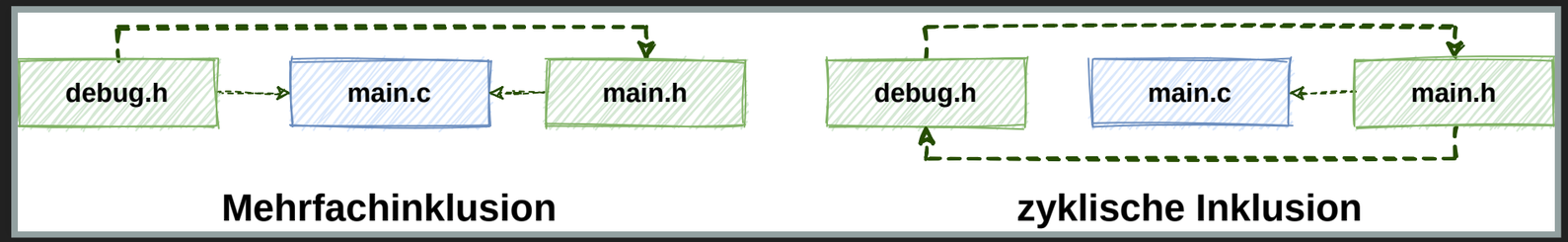
```c
#include <stdio.h>

int main (int argc, char *argv[]) {
#if 0
        ...
        ...
        ...
#endif
    return 0;
}
```

- Eine weitere Variante ist ein `#if`
  - Kann man z.B. verwenden man bestimmte Abschnitte einfach auskommentieren möchte!
  - Alle Zeilen zwischen #if 0 und #endif werden somit nicht kompiliert.

PRÄPROZESSOR: AUSSCHLUSS MEHRFACHINKLUSION

Bei der Inklusion mittels `#include` kann es u.U. zyklische oder mehrfache Inklusionen geben. Dies ist ein Problem, da z.B. Definitionen von Typen und Funktionen nur einmalig pro Datei, und damit im späteren kompiliertem Objekt Code, vorhanden sein.



- Oftmals inkludieren Header-Dateien auch andere Header Dateien
  - Wenn man 2 Header inkludiert und einer der Header den anderen Header ebenfalls inkludiert kommt es zur **Mehrfachinklusion**
  - Wenn man nun 2 Header inkludiert, die sich beide wiederum selbst inkludieren, kommt es zu einer **zyklischen Inklusion**.

PRÄPROZESSOR: AUSSCHLUSS MEHRFACHINKLUSION

Beispiel

```
#ifndef SYMBOL_NAME (1)
#define SYMBOL_NAME (2)
... // Funktionsprototypen
... // Makros
... // Typdefinitionen
...
#endif (3)
```

Datei debug.h

```
#ifndef _DEBUG_H (1)
#define _DEBUG_H (2)
... // Funktionsprototypen
... // Makros
... // Typdefinitionen
...
#endif (3)
```

(1) Erste Zeile der Header Datei

(2) Zweite Zeile der Header Datei

(3) Letzte Zeile der Header Datei

- **Konvention**:
  - Dateiname: `test.h` → `SYMBOL_NAME`: `_TEST_H`
  - Dateiname: `debug.h` → `SYMBOL_NAME`: `_DEBUG_H`
- Alternative Symbolnamen sind `_TEST_H`, `TEST_H` oder `_DEBUG_H`, `DEBUG_H`

PRÄPROZESSOR

Weitere detaillierte Informationen zum GNU C Präprozessor `cpp`:

- https://gcc.gnu.org/onlinedocs/gcc-10.2.0/cpp/index.html#SEC_Contents