Kapitel 4: Der C-Präprozesor

C-Präprozessoranweisungen

- Präprozessor-Anweisungen beginnen mit `#`
- Präprozessor-Anweisungen werden vor dem Kompiliervorgang ausgeführt
- Präprozessor-Features:
  - Zusammenführen von Zeilen die durch `\` aufgeteilt wurden
  - Definition und Ersetzung von Makros (`#define`)
  - Einschleusung von Dateiinhalten (`#include`)
  - Beeinflussung des Compilers (`#pragma`)
  - Bedingte Kompilierung (`#ifdef`)

`https://de.wikibooks.org/wiki/C-Programmierung:_Präprozessor#.23pragma`

Definition und Ersetzung von Makros

In C sind Makros mit und ohne Parameter zulässig

```
#define MAKRO $Ersatztext
#define MAKRO(<Parameterliste>) $Ersatztext
```

Beispiele:

```
#define MONTHS_PER_YEAR 12
#define MAXIMUM(a,b) ((a) > (b) ? (a) : (b))
```

Nutzung:

```
int foo=7, bar=10;
int monthly_costs = MAXIMUM(foo, bar); // (foo) > (
    bar) ? (foo): (bar)
int annual_costs = monthly_cost * MONTHS_PER_YEAR;
    // monthly_cost * 12
```

ANSI-C Macros

Fünf Makros die jeder ANSI-C-Compiler unterstützen muss.

`__LINE__`	Zeilennummer der momentanen Quelldatei
`__FILE__`	Name der momentanen Quelldatei
`__DATE__`	Übersetzungsdatum der Quelldatei
`__TIME__`	Übersetzungszeit der Quelldatei
`__STDC__`	Erkennungsmerkmal für ANSI-C Compiler

```
1  #include <stdio.h>
2
3  int main() {
4    if(__STDC__) {
5      printf("%s␣%d:␣",__FILE__, __LINE__);
6      printf("%s␣(%s)\n", __DATE__, __TIME__);
7    }
8    return 0;
9  }
```

Rekursive Makrodefinition

Falls der Name eines Makros innerhalb seiner eigenen Definition auftaucht, wird nicht ersetzt, sondern übernommen.

Beispiele

```
#define sqrt(x) printf("sqrt(%f) = %f\n",x, sqrt(x))
...
sqrt(9.61); // printf("sqrt(%f) = %f\n",9.61, sqrt(9.61));
```

Frage: Warum findet diese Ersetzung nicht statt?

Einschleusung von Dateiinhalten

- Üblicherweise werden mit der `#include` Anweisung `Headerdateien` (Endung `.h`) eingebunden
- Eigene Headerdateiein werden in Anführungszeichen angegeben (z.B. `#include "foobar.h"`)
- Die folgenden im ANSI-C Standard festgelegten Dateien werden in spitzen Klammern angegeben (z. B. `#include<stdio.h>`)

```
<assert.h>  <ctype.h>        <complex.h>  <errno.h>
<fenv.h>    <float.h>        <iso646.h>   <inttypes.h>
<limits.h>  <locale.h>       <math.h>     <setjmp.h>
<signal.h>  <stdalign.h>     <stdarg.h>   <stdatomic.h>
<stdbool.h> <stddef.h>       <stdio.h>    <stdint.h>
<stdlib.h>  <stdnoreturn.h>  <string.h>   <tgmath.h>
<threads.h> <time.h> <uchar.h> <wchar.h> <wctype.h>
```

Siehe `https://en.wikipedia.org/wiki/C_standard_library`

Bedingte Kompilierung

Mit Hilfe von `#ifdef`-Anweisungen kann festgelegt werden ob bestimmte Programmteile kompiliert werden oder nicht.

Schlüsselwort	Bedeutung
`#if $bedingung`	Test ob eine Bedingung erfüllt ist
`#ifdef $name`	Test ob ein Makro definiert wurde
`#ifndef $name`	Test ob ein Makro **noch nicht** definiert wurde
`#else`	Leitet den `else`-Programmteil ein
`#endif`	Beendet eine bedingte Kompilierungskonstruktion

Bedingte Kompilierung: Beispiel

```
1  #include<stdlib.h>
2  #include<stdio.h>
3
4  #define BAR
5
6  int main(){
7  #ifdef FOO
8    puts("Macro FOO is defined.");
9
10 #elif defined BAR
11   puts("Macro BAR is defined.");
12
13 #else
14   puts("Neither FOO nor BAR is defined.");
15 #endif
16
17   return EXIT_SUCCESS;
18 }
```

Endianness (byte order)

Je nach Rechner-Architektur werden Ganzzahlen, die aus mehreren Bytes bestehen, unterschiedlich im Speicher abgelegt.

- Big-Endian: Das höchstwertigste Byte wird zuerste gespeichert.

int x = 0x0A1B2C3D;			
0A	1B	2C	3D

- Little-Endian: Das niederwertigste Byte wird zuerste gespeichert.

int x = 0x0A1B2C3D;			
3D	2C	1B	0A

Bedingte Kompilierung: Real-World-Beispiel

```
1  #if __BYTE_ORDER == __LITTLE_ENDIAN
2    #define TO_LITTLE_ENDIAN_64(n) (n)
3    #define TO_LITTLE_ENDIAN_32(n) (n)
4
5  #elif __BYTE_ORDER == __BIG_ENDIAN
6    #define TO_LITTLE_ENDIAN_64(n) bswap_64(n)
7    #define TO_LITTLE_ENDIAN_32(n) bswap_32(n)
8
9  #else
10   #warning "byte␣order␣couldn't␣be␣detected".
11   #define TO_LITTLE_ENDIAN_64(n) (n)
12   #define TO_LITTLE_ENDIAN_32(n) (n)
13
14 #endif
```

Kompilierung abbrechen

```
1 #error "error message"
```

Mit der Präprozessoranweisung `#error` lässt sich der Compilierungsvorgang mit einer Fehlermeldung abbrechen.

Beispiel

```
1 #if __BYTE_ORDER == __BIG_ENDIAN
2 #error "BIG ENDIAN systems are not supported"
3 #endif
```

Include-Guard-Makro

Wird eine Headerdatei von mehreren C-Dateien, welche Teile eines Programmes sind, genutzt, kann es zu unerlaubten doppelten Definitionen kommen.

Daher verwendet man sogenannte *Include-Guards*.

Beispiel:

```
// foo.h
#ifndef FOO_H
#define FOO_H

#define FNORD 23

#endif /* FOO_H */
```

Alternative: Anweisung `#pragma once` zu Beginn einer Headerdatei.

Zusammenfassung

Sie sollten . . .

- . . . C-Präprozessor-Makros schreiben können.
- . . . die C-Präprozessor-Anweisungen zur bedingten Kompilierung beherschen.
- . . . wissen was es mit der Anweisung `#error` auf sich hat.
- . . . verstanden haben, warum es in C `Include-Guards` gibt.