23.12 Präprozessor

Montag, 23. Mai 2022 10:32

Präprozessoranweisungen / Makros

1. Nehmen Sie an, Sie arbeiten gerade an einem Projekt. Dieses Projekt verwendet die Library xyzlib. Zur Versionsverwaltung ist in der Header-Datei xyzlib.h der Library eine Makro XYZLIB_VER mit der Versionsnummer enthalten, welches beginnend ab 1 mit jeder Version hochgezählt wird. Auch sind dort von allen Funktionen, die extern im Zugriff stehen, entsprechende Prototypen enthalten.

`//main.c = Nutzer der Library`
`#include "xyzlib.h"`
`#define APP_MODE 2`
`int main(int argc, char **argv) {`
`  printf("starting\n");`
`  dosomething_xyz(42);`
`  printf("done\n");`
`  return 0;`
`}`
	//xyzlib.h = Library-Header
	#ifndef XYZLIB_H
	#define XYZLIB_H
	#define XYZLIB_VER 2
	
	void dosomething_xyz(int);
	
	#endif

   In der Version 3 der Library wurde ein neues Namensschema für die Funktionen eingeführt. Alle Funktionen innerhalb der Library beginnen nun mit dem Librarynamen xyz, so dass nun die Funktion dosomething_xyz() zu xyz_dosomesthing() umbenannt wurde.
   Damit der Nutzer dieser Library seinen Source-Code nicht sofort umstellen muss, sollen in der neuen Version die Funktionen sowohl mit dem neuen, als auch mit dem alten Namen zugänglich sein. Schreiben sie ein passendes Makro hierfür, welches den alten Namen auf den neuen Namen umsetzt.

2. Printf() Anweisungen werden häufig zu Debug-Zwecken in den Code eingefügt. Einige von diesen Debug-Ausgaben können nach Abschluss des Softwaretests sicherlich entfallen, andere sind jedoch dauerhaft während der Entwicklungsphase hilfreich. Im Release-Mode sollen diese Debug-Ausgaben jedoch nicht enthalten sein. Zur Unterscheidung des Übersetzungsmodus wird beim Compiler-Aufruf im Release-Mode das Makro NDEBUG gesetzt:
   Release-Mode: gcc -DNDEBUG ….
   Debug-Mode: gcc ….
   Schreiben sie ein DBGPRINT-Makro, welches im Debug-Mode den übergebenen String per printf() ausgibt und im Release-Mode keine Ausgabe tätigt. Der DBGPRINT Aufruf soll folglich im Source-Code dauerhaft enthalten sein. Über das Makro NDEBUG wird entweder die Ausgabe auf printf() 'umgeleitet' oder zu 'nichts' aufgelöst.

```
//Zum Testen Makro händisch im Code setzen
//Bzw. für den Release Mode dieses Makro in Kommentar setzen
#define NDEBUG

#define DBGPRINT(str)   ….
#else
#define DBGPRINT(str)   ….

int main(int argc, char *argv[])
{
   DBGPRINT("Main started\n");    //bleibt dauerhaft im Code
                                  //erzeugt im Relase-Mode jedoch keine ausführbaren
Code
}
```

3. Schreibe sie ein Function-like-Makro, welches einen String übergeben bekommt und diesen prüft, ob dort ein Großbuchstabe vorhanden ist. Wenn mindestens ein Element des Strings einen Großbuchstaben enthält, soll dieses Makro eine 1, andernfalls eine 0 zurückgeben.

```
#define STRISUPPER(str)
```

4. Asserts sind Funktionsaufrufe, die eine Zusicherung überprüfen und im Falle einer ungültigen Zusicherung das Programm kontrolliert beenden (Siehe auch MAN-Pages). Asserts werden oft zur Absicherung von Übergabeparameter eingesetzt, über die z.B. geprüft wird, ob ein Zeiger ungleich NULL übergeben wurde:

```
Void func(char *str,int para) {
  assert(str!=NULL);
  assert((para>=5) && (para<=10))
```

Realisieren sie die assert Anweisung über ein Function Like Makro, welche bei ungültiger Zusicherung die exit() Funktion aufruft.

```
#define assert( ….. )
```

5. Gegen sei folgender Code

```
#define LIST_OF_VARIABLES \
      X(value1) \
      X(value2) \
      X(value3)

#define X(name) int name;
   LIST_OF_VARIABLES
#undef X

void foo(void)
{
#define X(name) printf("%s = %d\n", #name, name);
LIST_OF_VARIABLES
#undef X
}
```

Was übergibt der Präprozessor an den C-Compiler, resp. Wie sieht der übersetzte C-Code aus