Hintergrundinformationen zur Vorlesung
GRUNDLAGEN DER INFORMATIK I
Studiengang Elektrotechnik
WS 02/03

AG Betriebssysteme FB3
Kirsten Berkenkötter

Erstellen von Bibliotheken

1 Hintergrund: Gültigkeitsbereich von Variablen

1. modulglobal: innerhalb einer gesamten Programmdatei
   Vorsicht bei der Verwendung, wenn mehrere Funktionen darauf zugreifen können
2. programmglobal: innerhalb eines gesamten Programms, Schlüsselwort *extern* (s. Header)
3. lokal: innerhalb eines Blocks, d.h. geschweifte Klammern
   Variablendeklaration nur direkt nach Blockanfang
   wenn innerhalb und außerhalb eines Blocks dieselben Variablennamen: innere Variable wird genommen (auch bei verschiedenen Typen)

Beispiel: Modulglobale und lokale Variablen

```
#include <stdio.h>

    int a = 5;      //modulglobale Variable
    int main()
    {
       int x = 10;     //lokale Variable in main
       {
          int x;     //lokale Variable im inneren Block
          int sum;     //lokale Variable im inneren Block
          for(x = 1; x <= 100; x = x+2)
             sum+=x;
          printf("Die Summe der ungeraden Zahlen von 1 bis 100 ist %d\n", sum);    //2500
          printf("x = %d\n", x);     //x ist 101, da lokale Variable aus innerem Block
       }
       printf("x = %d\n", x);     //x ist 10, da lokale Variable aus main
    }
```

2 Hintergrund: Lebensdauer von Variablen

1. statisch, d.h. während der gesammten Programmablaufzeit vorhanden

   (a) Schlüsselwort *extern*: statisch, programmglobal, Speicherung im Datensegment
   Besonderheit: keine Initialisierung bei der Deklaration, da nur Speicher zugewiesen wird
   Besonderheit: extern, wo definiert, Variable ohne extern in anderer Datei, z.B. *extern int x;* in Bibliothek und *int x* in der Anwendung

   (b) Schlüsselwort *static* in Funktion: statisch, lokal, Speicherung im Datensegment
   (c) Schlüsselwort *static* außerhalb von Funktion: statisch, modulglobal, im Datensegment
   (d) modulglobale Variablen sind genau wie programmglobale Variablen immer statisch

2. kurzzeitig (automatic) nur bei Bedarf, d.h. bei Funktion oder Block

   (a) Schlüsselwort *auto* (überflüssig): kurzzeitig, lokal, Speicherung im Stack
   (b) Schlüsselwort *register*: kurzzeitig, lokal, im Register (aus Zeitgründen, schnellere Verarbeitung, aber keine Möglichkeit, den Adressregister *&* zu benutzen, da die Variable im Register keine Adresse hat!)

3 Hintergrund: Forward-Deklaration von Funktionen

1. erste Möglichkeit: Funktionen vor ihrem Gebrauch deklarieren, dann sind sie bekannt und können benutzt werden
2. zweite Möglichkeit: Funktionskopf deklarieren (Name, Rückgabetyp, Parametertypen), dann kann die Funktion benutzt werden, obwohl sie erst später ausdeklariert wird

Beispiel: Forward-Deklaration

```
#include <stdio.h>

int main()
{
   unsigned long fakultaet(int);
   int zahl = 5;

   printf("Fakultaet von %d ist %u\n", zahl, fakultaet(zahl));
}

unsigned long fakultaet(int n)
{
   unsigned long fak = 1;
   int i;

   for(i = 1; i<=n; i++)
      fak *= i;
   return fak;
}
```

4 Erstellen einer Bibliothek

4.1 Code für die Bibliothek

- zunaechst wird der Quellcode benötigt, d.h. eine **.c*-Datei mit den Funktionen, aber keine *main*-Funktion
- kompiliert wird dann mit *gcc -c biblio.c*
- Typdefinitionen und programmglobale Variablen sind am Besten in der Header-Datei aufgehoben (s. 4.2), dann muss die Header-Datei auch in der Code-Datei eingebunden werden
- wenn Funktionen in der Code-Datei Zeiger zurückgeben, dann müssen dafür modulglobale Variablen (Speicherung im Datensegment des Speichers) verwendet werden (lokale Variablen werden auf dem Stack gespeichert und gehen nach Ablauf der Funktion verloren, Zeiger auf diese Variablen sind dann ungueltig!)

4.2 Header-Datei für die Bibliothek

- weiterhin wird eine Header-Datei benötigt, welche die Funktions-Prototypen (d.h. Forward-Deklaration der entsprechenden Funktionen) enthält und eventuelle programmglobale Variablen mit dem Schlüsselwort *extern* deklariert, sowie Typdefinitionen
- Benennung der Datei mit Endung .h, *z.*B. *biblio.h*

4.3 Anwendung einer Bibliothek

- die Bibliothek kann dann mit anderen Programmen benutzt werden, indem man die neue Headerdatei einbindet
  #include "biblio.h"
- in diesem Fall müssen allerdings die C-Dateien gelinkt werden, d.h. beim Kompilieren muss angegeben werden, welche weiteren Dateien benötigt werden
- ist die Bibliothek bereits kompiliert, geschieht das mit
  gcc -o programm biblio.o programm.c
- ist die Bibliothek noch nicht kompiliert, geschieht das mit
  gcc -o programm biblio.c programm.c
- wenn externe Variablen der Bibliothek aus dem Header verwendet werden, müssen diese ohne das Schlüsselwort *extern* nochmals deklariert werden, also z.B. *extern int x;* in der Datei *biblio.c* und *int x;* in der Datei *programm.c*

4.4 Beispiel Listenpaket

- Code: listenpaket.c
- Header-Datei: listenpaket.h
- Anwendung: listenprogramm.c