Praxis der Programmierung

Ausblick: Was es in C noch gibt.

Institut für Informatik und Computational Science
Universität Potsdam

Henning Bordihn

Unionen und Bitfelder

Unionen

- **Deklaration:** wie Strukturen, mit `union` statt `struct`
- Members (Komponenten) stellen Alternativen dar;
  zu jedem Zeitpunkt ist nur eine Komponente gespeichert
- **Zugriffe** mit Punkt- bzw. Pfeilschreibweise
- **Beispiel:** Repräsentation eines Punktes entweder
  mit karthesischen oder Radial-/Polarkoordinaten

Unionen: Beispiel

```
struct cartPoint {
    double x, y;
}

struct radPoint {
    double radius, phi
}

struct point {
    enum pointType ptype;
    union coordinates c;
}
```

```
union coordinates {
    struct cartPoint kp;
    struct radPoint rp;
}

enum pointType {
    CART, RAD
}
```

Bitfelder

- besteht aus vorgegebener Anzahl von Bits
- treten als Komponenten von Strukturen oder Unionen auf
- **Deklaration:** `ganzzahltyp name:bitanzahl;`
  Beispiel: `signed a:4; // Werte von -8 bis +7`
- Bitmuster werden als Werte des Ganzzahltyps interpretiert
- plattformabhängige Interpretation
- für hardwarenahe und Graphikprogrammierung eingesetzt

Präprozessoranweisungen

Präprozessoranweisungen (1)

- Ausführung vor der eigentlichen Kompilation
- `#Direktive Text`
- *bisher:*
  - Einfügen von Dateiinhalten mit `#include datei`

    `#include <stdio.h>`
  - Vereinbarunf symbolischer Konstanten und Makros mit

    `#define Bezeichner Ersatztext`

    `#define double_inc(a, b) a++; b++;`

Präprozessoranweisungen (2)

- bedingte Kompilierung
  - Compiler entscheidet anhand von Ausdrücken und Symbolen, welche Codeteile übersetzt werden sollen
  - Direktiven `#if #elif #else ...`
  - Einsatz z.B. um auf Compilerversionen reagieren zu können oder Programmversionen in einer Datei zu halten (*Write once!*)
- eigene Fehlermeldungen während des Kompilierens generieren
- ...

Die Standardbibliothek

Bibliotheksfunktionen (1)

- **Funktionen zur Ein- und Ausgabe (stdio.h)**
  - Schreiben von Zeichen oder Strings auf `stdout`
  - Lesen von Zeichen oder Strings von `stdin`
  - Lesen oder Schreiben von bzw. in Streams/Dateien, ...
- **String- und Speicherbearbeitung (string.h)**
  - Vergleich, Kopieren zweier Strings
  - Suchen von Zeichen oder Teilstrings in Strings
  - Bestimmen der Länge eines Strings, ...
  - analoge Funktionen für Datenblöcke im Speicher

    z.B. `strcpy()` *vs.* `memcpy()`, ...

Bibliotheksfunktionen (2)

- **Mathematische Funktionen (math.h)**

  `sin(), cos(), log(), log10(), floor(), pow(), ...`

- **Zahlenkonvertierungen, Speicherverwaltung, Zufallszahlen (stdlib.h)**

  - `abs(), atof(), atoi(), atol(), ...`
  - `malloc(), realloc(), free(), ...`
  - `rand(), ...`

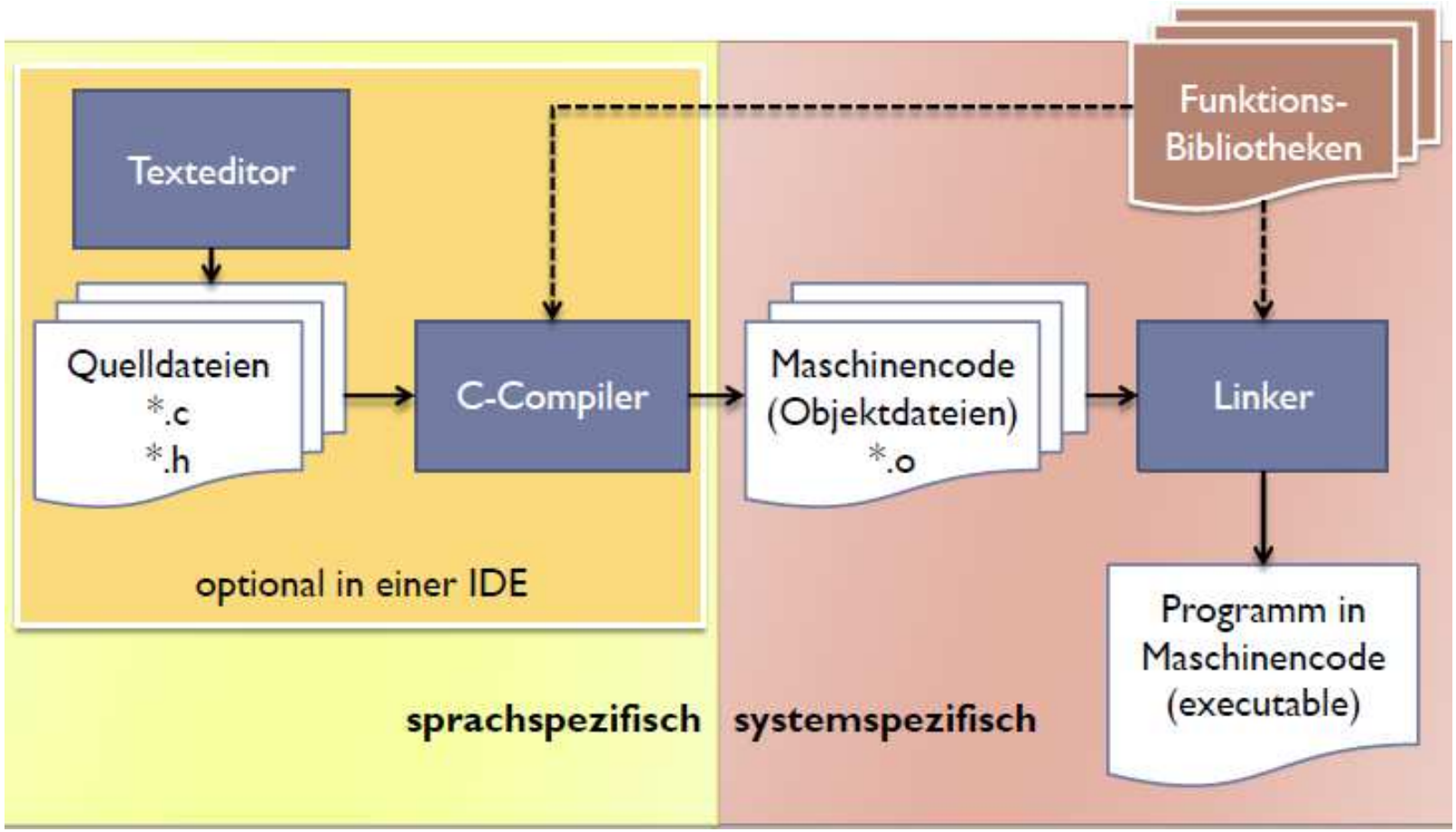
- **Klassifizierung und Konvertierung von Zeichen (ctype.h)**

  - Test ob ein Character alphanumerisch, ein Buchstabe, eine Zahl, ... ist
  - `tolower(), toupper()`

Bibliotheksfunktionen (3)

- **Festlegung länderspezifischer Zeichen und Darstellungen (locale.h)**
- **Datum und Uhrzeit (time.h)**
- einige mehr, versionsabhängig (s. Compiler-Handbuch)

Einbinden der Bibliotheksfunktionen (1)



Einbinden der Bibliotheksfunktionen (2)

- Header-Dateien enthalten nur Deklarationen (Signaturen) der Funktionen
- Prprozessor setzt diese an der Stelle der `#include`-Direktive ein
  $\rightsquigarrow$ Funktionen sind dem Compiler bekannt
- Implementierung erst für das Laufzeitsystem zum Zeitpunkt des Funktionsaufruf bedeutsam
- Linker fügt die Objektcode-Dateien zusammen
  - ggf. verschiedene C-Dateien und Bibliotheksdateien
  - Objekcode-Dateien verwenden relative/virtuelle Speicheradressen
  - Linker sorgt für einheitlichen Adressraum des ausführbaren Programms (Zuordnung eines virtuellen, überschneidungsfreien Adressraums)

Speicherklassen

Konzept der Speicherklassen

- Modifikation der Gültigkeit von Variablenvereinbarungen mit Hilfe von Schlüsselwörtern `extern, static, auto, register`
- erlaubt u.a. Verteilung eines C-Programms auf verschiedene Dateien
- Speicherklasse `extern` zeigt an, dass die Definition an anderer Stelle in derselben oder einer anderen Datei erfolgt

```
extern int value;   // Definition erfolgt an anderer Stelle
...
int value = 2014;   // muss vollstaendige Definition sein
```

- vorwärtsdeklarierte Funktionen sind implizit `extern`

```
void g(int n);              // steht fuer extern void g(int n);
void f(int n) { g(n-1); }
void g(int n) { f(n-1); }
```