Bedeutung und Korrektheit von C Programmen

Vorlesung vom 29.04.2008: Typen in C: Konversionen, Zeiger und Felder

Christoph Lüth & Lutz Schröder

SS 08

Fahrplan Teil I

- Teil I: C
  - Einführung in den Standard
  - Typen in C: Deklarationen
  - Typen in C: Konversionen, Felder vs. Zeiger
- Teil II: Bedeutung
- Teil III: Eigenschaften

Fahrplan heute

- Typkonversionen (Typwechsel ohne Anmeldung)
- Felder vs. Zeiger — Das C-Speichermodell

Verschiedene Typen im Standard

- Object types, incomplete types, function types (§6.2.5)
  - Basic types: standard/extended signed/unsigned integer types, floating types (§6.2.5)
  - Derived types: structure, union, array, function types
- Qualified Type (§6.2.5)
  - `float const *` qualified (qualified pointer to type)
  - `const float *` not qualified (pointer to qualified type)
- Compatible Types (§6.2.7)
- Assignable Types (§6.5.16, §6.5.2.2)

Typkonversionen in C

- Integer promotion (§6.3.1.1):
  - `char`, `enum`, `unsigned char`, `short`, `unsigned short`, *bitfield*
- `float` promoted to `double`
- `T []` promoted to `T *`
- Usual arithmetic conversions (§6.3.1.8)
- Konversion von Funktionsargumenten:
  - Nur bei K&R-artigen Deklarationen, nicht bei Prototypen!
  - Moral: Stil bei Prototyp und Definition nicht mischen!

Zeiger und Felder

- Zeiger: kann auf ein Objekt im Speicher zeigen:

  `int *x;`
- Feld: reserviert Speicherbereich für `n` Objekte:

  `int y[100];`
  - Index beginnt immer mit 0
  - `y[i]` wird konvertiert zu `*(y+i)`
  - Mehrdimensionale Felder ... möglich
- Aber: `x` $\not\sim$ `y`

Wann sind Felder Zeiger?

- Wann kann ein Array in ein Zeiger konvertiert werden?

Externe Deklaration: `extern char a[]`	Keine Konversion
Definition: `char a [10]`	Keine Konversion
Funktionsparameter: `f(char a[])`	Konversion möglich
In einem Ausdruck: `x= a[3]`	Konversion möglich

- Wenn Konversion möglich, dann durch Semantik erzwungen
- Tückisch: Externe Deklaration vs. Definition

Wann sind Felder Zeiger?

1. Ein Arrayname in einem Ausdruck (im Gegensatz zu einer Deklaration) wird vom Compiler als Zeiger auf das erste Element der des Feldes behandelt. (§6.3.2.1)
2. Ein Subskript ist immer äquivalent zu einem Offset eines Zeigers (§6.5.2.1)
3. Ein Arrayname in einer Parameterdeklaration einer Funktion wird vom Compiler als Zeiger auf das erste Element des Feldes behandelt. (§6.7.5.3)

Mehrdimensionale Felder

- Deklaration: `int foo[2][3][5];`
- Benutzung: `foo[i][j][k];`
- Stored in Row major order (Letzter Index variiert am schnellsten) (§6.5.2.1)
- Kompatible Typen:
  - `int (* p)[3][5] = foo;`
  - `int (*r)[5]= foo[1];`
  - `int *t = foo[1][2];`
  - `int u = foo[1][2][3];`

Mehrdimensionale Felder als Funktionsparameter

- Regel 3 gilt nicht rekursiv:
  - Array of Arrays ist Array of Pointers, nicht Pointer to Pointer
- Mögliches Matching:

Parameter	Argument
`char (*c)[10];`	`char c[8][10]; char (*c)[10];`
`char **c;`	`char *c[10]; char **c;`
`char c[][10];`	`char c[8][10]; char (*c)[10];`

- `f(int x[][]);` nicht erlaubt (§6.7.5.2)
  - NB. Warum ist `int main(int argc, char *argv[])` erlaubt?

Zusammenfassung

- Typkonversionen in C: meist klar, manchmal überraschend
- Pointer und Felder:
  - Bei Deklaration unterschiedlich (Ausnahme Funktionsparameter)
  - Bei Gebrauch Konversion