Betriebssysteme

Kapitel 1.2: Die Programmiersprache C
Typen und Typ-Deklarationen

Stand: WS 08/09

Prof. Dr. Wolfgang Küchlin

Dipl.-Inform., Dr. sc. techn. (ETH)

Arbeitsbereich Symbolisches Rechnen
Wilhelm-Schickard-Institut für Informatik
Fakultät für Informations- und Kognitionswissenschaften

Universität Tübingen

Steinbeis Transferzentrum
Objekt- und Internet-Technologien (OIT)

Wolfgang.Kuechlin@uni-tuebingen.de
http://www-sr.informatik.uni-tuebingen.de

SR 

Typen und Typ-Deklarationen

- Abgeleitete Typen (derived types)
  - aus Basistypen können komplexere Typen abgeleitet werden
  - Zeiger (pointer), Reihungen (arrays), Verbunde (structures, records), Funktionstypen (function types)
- Sei T ein Typ.
  - mittels der **Deklarations-Operatoren** *, &, [ ], ( ) erhält man abgeleitete Typen
  - T* pointer to (object of type) T
  - T& reference of (object of type) T
  - T[ ] array of (objects of type) T
  - T( ) function returning (object of type) T

SR 

Typ-Deklarationen und Ausdrucks-Operatoren

- Den Deklarations-Operatoren *, &, [ ], ( ) für Typen entsprechen die **Ausdrucks-Operatoren**
  - * Dereferenzieren, Pointer-Inhalt
  - & Adresse (L-value) von
  - [ ] Array-Zugriff
  - ( ) Funktions-Aufruf
- Sei v eine Variable
  - Ist v vom Typ T*, so ist *v vom Typ T
  - Ist v vom Typ T[ ], so ist v[0] vom Typ T
  - Ist v vom Typ T( ), so ist v() vom Typ T
  - Ist pf ein Zeiger auf eine Funktion, so ist *pf die Funktion. (*pf)(x) ruft eine Funktion mit Argument x auf

SR 

Deklaration von Variablen von abgeleiteten Typen

- Prinzip: Die Deklaration einer Variable vom Typ T (T x;) spiegelt die spätere Benutzung wider
- Vorgehensweise zur Typdeklaration
  - Sei B ein Basistyp. Eine Variable x erhält einen von B abgeleiteten Typ durch das **Typ-Deklarationsverfahren**
    1. Schreibe zunächst die Deklaration **B x;**
    2. Denke x als Variable vom gewünschten Typ. Dekoriere nun x solange mit den (Ausdrucks-)Operatoren *, &, [ ], ( ) bis der Basistyp herauskommt. (Der Basistyp einer Funktion ist der des Funktions-Ergebnisses).
    3. Das Resultat ist die richtige Deklaration von x, wenn man die Ausdrucks-Operatoren als Deklarations-Operatoren interpretiert.
    4. Lässt man den Variablennamen weg, hat man den Typ-Ausdruck
    5. Stellt man typedef davor, hat man statt der Variable einen Namen für den abgeleiteten Typ.

SR 

Deklaration von Variablen von abgeleiteten Typen

- Bemerkung:
  - * und & sind Präfix-Operatoren, [ ] und () sind Postfix-Operatoren. [ ] und () binden stärker als * und &. Im Zweifelsfall unbedingt Klammern schreiben!!
- Beispiele

Gwünschter Typ	Nach Verfahren	Empfohlene Schreibweise	Typ-Ausdruck
Pointer to int	int *pi	int* pi	int*
Pointer to array of 10 int	int (*pai)[10]	int (*pai)[10]	int (*)[10]
Array of 10 pointers to int	int *(api[10])	int* api[10]	int *[10]
Pointer to pointer to char	char *(*ppc)	char** ppc	char**
Function: int → int pointer	int *(ipfi(int))	int* ipfi(int)	int* (int)
Pointer to function: int → int	int (*pifi)(int)	int (*pifi)(int)	int (*)(int)

Reihungen und Zeiger

- Reihungen sind gegeben durch einen Zeiger auf den Anfang (also auf das Element mit Index 0).
  - der Typ int[ ] ist ein Synonym für den Typ int*
- Beispiel: Nach einer Deklaration int x[10] ist x vom Typ int*.
  - x[0] ist synonym für (wird umgewandelt in) *x.
  - x[1] ist synonym für *(x+1)
- Reihungsgrenzen können beim Zugriff nicht überwacht werden
  - ein direkter Zugriff über den Zeiger mit explizit angegebener Zeiger-Arithmetik ist immer möglich.
  - Beliebt: p=i; x = *p++ statt x=p[i]; i=i+1;