universitätbonn

Zeigervariablen und Arrays

Assignment is undoubtedly the most characteristic feature of programming a digital computer, and one that most clearly distinguishes it from other branches of mathematics.

C. A. R. Hoare (1969)

Variablen

Referenzen

Zuweisungen

Variablen

- Variablen dienen der Speicherung von Werten
  - **Wert** (value) ist ein Element eines Datentyps der zugehörigen Programmiersprache
    - ⇨ Jede Variablen hat einen Typ
  - Jede Variable hat einen Namen (name)
- Variable ist (zunächst) Paar (Name, Wert)
- Abstraktion des Konzepts der Speicherstelle
  - **(Adresse, Inhalt)**

Variablen

- Übersetzer ordnet Variablen Speicherstellen zu
  - Abbildung von
    - ⇨ Name auf Addresse
    - ⇨ Wert auf Inhalt der Adresse
      - – Aus Wert kann gemäß des Typs der Variablen auf den Inhalt geschlossen werden
- Im laufenden Programm Variable vollständiger durch Tupel **((Name, Wert), (Adresse, Inhalt))** charakterisiert

Variablen

- Die genaue Auflistung der Speicherstelle ist aber oft nicht realistisch und auch nicht nützlich
  - Übersetzer legen die Adresse einer Variablen i. a. nicht auf Hardware-Ebene fest, sondern abstrakter
    - ⇨ Etwa relativ zum Anfang eines Speicherbereichs
  - Die endgültige Zuordnung wird dann vom Betriebssystem in Verbindung mit der memory management unit (MMU) der Hardware erst zur Laufzeit vorgenommen
- Wir sprechen lieber abstrakter von **Verweis** (auf einen Speicherplatz) oder von **Referenz**
  - Statt von einer Adresse
  - In Schaubildern verwendet man oft Zeiger (pointer), um Referenzen darzustellen
    - ⇨ Symbolisiert durch Pfeile
    - ⇨ Insbesondere wenn die konkreten Adressen gar keine Rolle spielen

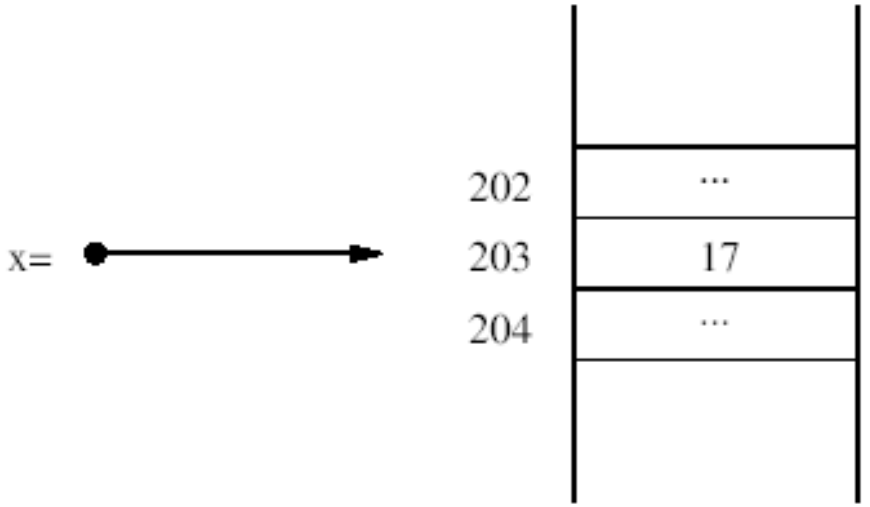
Variablen

- **Beispiel:**
  - Der Wert der Variablen mit dem Namen x ist 17
  - Die Referenz ist die Adresse 203, an der der Wert gespeichert ist
  - Der dort tatsächlich gespeicherte Inhalt ist das Bitmuster 10001

    Wenn wir 17 als Literal für einen int Wert ansehen dürfen

    Und die führenden 0 nicht darstellen

- Eine noch abstraktere Form der Darstellung der Referenz ist der Pfeil auf die entsprechende Speicherstelle

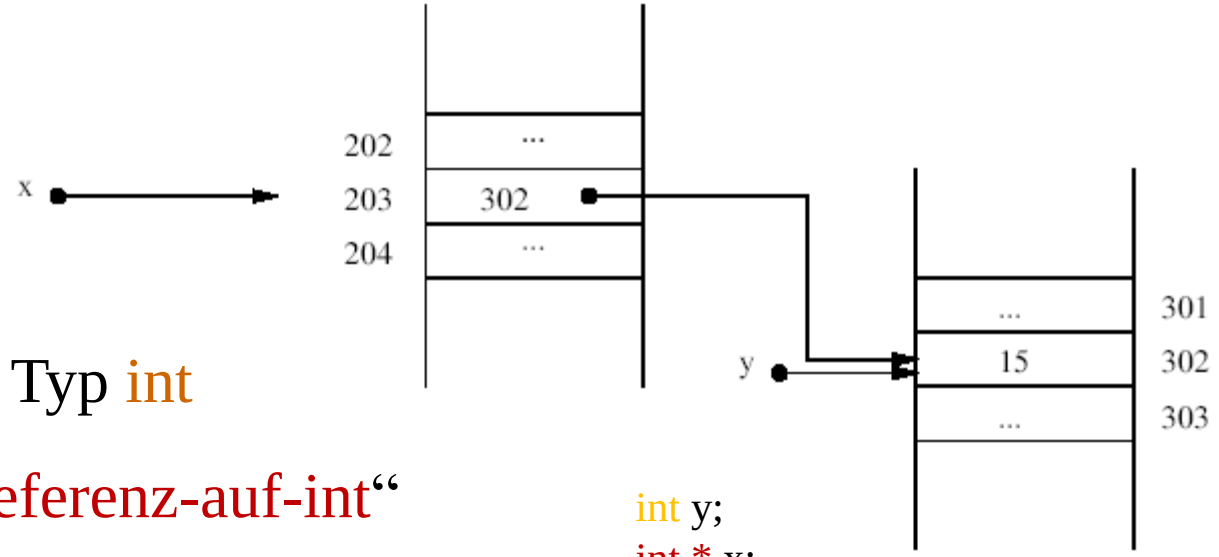


Zeigervariablen

- Variablen, deren Wert wiederum eine Referenz ist, heißen **Zeigervariablen** (pointer variable, pointer) oder **Referenzvariablen** (reference variable, reference)
- Der spezielle Wert null, das einzige Element des **Nulltyps** (null type), symbolisiert die leere Referenz, die auf keine gültige Speicherstelle verweist
  - Auf Maschinenebene ist null i.A. durch den Zahlwert Null repräsentiert

Zeigervariablen

- **Beispiel**
  - Der Wert der Zeigervariable x ist die Referenz der Variable y, die ihrerseits den Wert 15 hat



y hat Typ int

x hat Typ „Referenz-auf-int“

```
int y;
int * x;
y=15;
x= &y;
*x=15;
```

universitätbonn

Arrays

Reihungen (arrays)

- Eine eindimensionale Reihung besteht aus einer bestimmten Anzahl von Daten gleicher Art
  - Kann als einzelne Zeile oder Spalte einer Tabelle gedacht werden
  - Auf jedes Element der Reihung kann mit demselben Zeitaufwand zugegriffen werden
    - ⇨ Z. B. in der Form a[i]
  - Auf diese Art werden etwa Werte einer Funktion an den Stellen i gespeichert, wie z. B. die Werte eines Eingabesignals zu den Zeitpunkten i=1,...,k

Zeitpunkt	1	2	3	4	...	30	31
Signalstärke	10.5	10.5	12.2	9.8	...	13.1	13.3

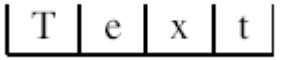
Reihungen (arrays)

- Sind die Reihungselemente von einem Typ T (z. B. Ganzzahl), so ist die Reihung selbst vom Typ *Reihung von T*
- *Zweidimensionale Reihungen* speichern die Werte mehrerer eindimensionaler Zeilen (sofern alle vom gleichen Typ sind) in Tabellen-(Matrix-)Form
  - a[i, j] ist das Element in der j-ten Spalte der i-ten Zeile
  - Alternative Syntax (in PASCAL) zu a[i][j]
- Entsprechend dreidimensionale (bzw. *n*-dimensionale) Reihungen

Reihungen (arrays)

- Arrays repräsentieren also Funktionen vom Indexbereich in einen Wertebereich, der der Typ der Array-Elemente ist
- Sei etwa *t*: IN × IN → IR eine Funktion, die einem Koordinatenpaar einen Temperaturwert zuordnet
  - Der Wert der Funktion an der Stelle (1,1) , also *t*(1,1) findet sich dann im Array t an der Stelle t[1][1]
- Arrays eignen sich in der Praxis grundsätzlich nur dann zur Speicherung einer Funktion, wenn diese dicht ist, d. h. wenn die Abbildung für die allermeisten Indexwerte definiert ist
  - Sonst würde eine Arraydarstellung viel zuviel Platz beanspruchen
  - Außerdem geht dies nur für endliche Funktionen

Reihungen (arrays): Zeichenreihen (strings)

- Wichtiger Spezialfall
  - Zeichenreihen (strings)
    - ⇨ Reihung von Zeichen (array of char)
  - Viele Programmiersprachen haben dafür eigene Syntax
    - ⇨ In Java z.B. Buchstabenfolge in Anführungszeichen
      - "Text"
        - Ergib folgendes Speicherbild 
    - ⇨ Für andere Reihungen Notation mit geschweiften Klammern
      - {10.5, 10.5, 12.2, 9.8,..., 13.1, 13.3}
      - {'T', 'e', 'x', 't'}

Reihungsvariablen

- In C sind Reihungen (arrays) Spezialfälle von Zeigervariablen
  - Die auf einen Speicherbereich zeigen, in dem mehrere Variablen des gleichen Typs liegen
  - Und auf die indiziert zugegriffen werden kann
    - ⇨ Mithilfe des []-Operators
- Ist T ein Typ, so ist T[] der Typ **Reihung** (array) von Elementen des Typs T
  - Kurz **Reihung von T** oder **T array**
  - **Allerdings muss in C die folgende Syntax zur Deklaration verwendet werden**
    - ⇨ **T varname []**
      - **Diese Syntax ist auch in Java erlaubt**
      - **Neben der systematischeren**
      - **T[] varname**

Reihungsvariablen

- Eine **Reihungsvariable** T a [n]; ist eine Referenzvariable, deren Wert auf ein konkretes Reihungsobjekt verweist
- Der Compiler stellt uns eine passende Anzahl von Variablennamen a[0], a[1], ... a[n-1] zur Verfügung
- Die Variable a selbst ist eine Zeigervariable auf den Anfang des Arrays

Reihungen (Arrays)

- Beispiele

```
int count[5]; /* 5 element integer array */
float miles[10]; /* 10 element floating point array */

count[4] = 20; /* code 4 to update the 5th element */
```

So kann man ‚count' bei dessen Deklaration initialisieren:

```
main() {
  int i;
  int count[5]={10, 20, 30};
  for (i=0; i<5; i++) {
    printf("%d ", count[i]);
   }
}
```

Das Ergebnis ist:

10 20 30 0 0

Reihungen (Arrays)

- Bemerkung
  - In dieser Art der Deklaration von Arrays wird der Speicher auf dem Stapelspeicher allokiert
    - ⇨ Und dort auch automatisch wieder deallokiert
    - ⇨ Gut für “kleinere Beispiele”
      - – Bis zu einigen tausend Speicherstellen
  - Bei größeren Beispielen
    - ⇨ Allokation des Arrays auf dem Haldenspeicher
    - ⇨ Siehe später

Verwendung von Arrays in Unterprogrammen

- In Unterprogrammen sollten Referenzvariablen als Parameter für ein Array verwendet werden
  - Übergabe von diesen sogenannten statischen Arrays direkt problematisch

Verwendung von Arrays in Unterprogrammen

Hier kann auch: unsigned statt double verwendet werden (ist sogar besser); bei Mittelwert eines integer-arrays ist aber ein double zurückzugeben

```
1  #include <stdio.h>
2
3  /* bestimme das maximum eines arrays */
4  double array_max(unsigned *array, unsigned length) {
5    unsigned i, max = 0;
6    for (i=0; i<length; i++)
7      if (array[i] > max) max = array[i];
8    return max;
9  }
10
11 int main() {
12   unsigned a[5] = { 12, 9, 1, 3, 7 };
13   printf("%f\n", array_max(a,5));
14 }
```