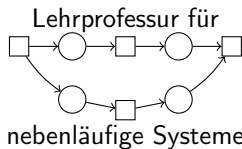


Vorkurs Informatik

Array und Strings

Robert Lorenz

Tag 6



Was ist ein Array?

Definition

- Ein **Array** vom Typ T ist eine Variable mit endlich vielen **Komponenten** vom Typ T.
- Jede **Komponente** ist eine Variable vom Typ T.

In einem Array können wir also mehrere Werte eines einheitlichen Datentyps speichern. Das benötigt man für verschiedene Anwendungen, zum Beispiel in der Mathematik zum Speichern von Vektoren.

- Als Datentyp kommen zum Beispiel die bisher besprochenen Datentypen `int`, `char` und `double` in Frage.
- Man kann aber auch Arrays benutzen, deren Komponenten wieder Arrays sind - das sind sogenannte 2-dimensionale Arrays und zum Beispiel geeignet für Matrizenrechnung in der Mathematik.

Was ist ein Array?

Deklaration

Deklaration

`T <Arrayname>[n];`

- `<Arrayname>` ist der Name für ein Array mit `n` Komponenten vom Typ `T`.
- `T` kann ein beliebiger Datentyp sein.
- `n` ist die **Arraylänge** und muss eine nichtnegative, ganze Zahl sein.
- Der Compiler reserviert Speicherplatz für genau diese `n` Komponenten vom Typ `T`

Beispiel

Deklaration eines Arrays mit 10 Komponenten vom Typ `int`:

```
int numbers[10];
```

Was ist ein Array?

Komponenten

Komponenten

`T <Arrayname>[n];`

- `<Arrayname>[i]` ist eine Variable vom Typ `T` und bezeichnet die $(i+1)$ -te Komponente des Arrays.
- `i` ist der ganzzahlige **Arrayindex** - er beginnt bei 0.
- Die Komponenten können mit dem Arrayindex **in einer Schleife** durchlaufen werden

Beispiel

```
int numbers[10];
```

- Wert der ersten Komponente auf 1 setzen:
`numbers[0] = 1;`
- Wert der zweiten Komponente soll um 1 höher sein:
`numbers[1] = numbers[0] + 1;`

Wie benutzt man ein Array?

Werte speichern

Werte müssen komponentenweise gespeichert werden

- Wertzuweisung an Komponente:
`<Arrayname>[i] = <Ausdruck>;`
- Wertzuweisung in der Deklaration :
`T <Arrayname>[n]={<Konstante>, ..., <Konstante>;}`
(Wertebelegung der Komponenten von links nach rechts, ggf.
Auffüllen mit Nullwerten):

Ein Array mit Zufallszahlen anlegen

```
1 int numbers[10];  
2 for(int i = 0; i < 10; ++i) {  
3     numbers[i] = rand();  
4 }
```

Wie benutzt man ein Array?

Typische Fehler

Der **Arrayname** repräsentiert die **Speicheradresse**, an der die Komponenten abgespeichert werden. Diese Adresse wird vom Compiler einmalig zugewiesen und ist deshalb **konstant**.

Man kann Arraynamen keine Werte zuweisen

```
<Arrayname> = <Ausdruck>;
```

- Erzeugt Compilerfehler ...
- ... da die zugewiesene Adresse konstant ist.

Man kann mit Arraynamen keine Komponentenwerte vergleichen

```
<Arrayname1> == <Arrayname2>
```

- vergleicht Speicheradressen ...
- ... und nicht den Inhalt der Arraykomponenten.

Wie benutzt man ein Array?

Typische Fehler

Compiler verhindert **nicht** den Zugriff über den reservierten Bereich hinaus

```
int numbers[10];  
numbers[20] = 1;
```

- erzeugt **keinen** Compilerfehler
- erzeugt aber zur Laufzeit **Fehler bis hin zum Programmabsturz**
(da in andere reservierte Speicherbereiche hineingeschrieben wird)
- allerdings gibt der Compiler eine Warnmeldung aus

Andererseits muss der reservierte Speicherbereich nicht komplett benutzt werden.

Wie benutzt man ein Array?

Arrays als Funktionsparameter

Arrays als Eingabeparameter

Für einen Eingabeparameter einer Funktion der Form

`T <Parameter>[]`

wird bei Funktionsaufruf ein Arrayname (vom Typ T) übergeben

Hinweise

- Die Anzahl der Komponenten gehört **nicht zur Typangabe**:
Es können Arrays unterschiedlicher Länge übergeben werden
- Die Anzahl der Komponenten wird **nicht mit dem Arraynamen übergeben**:
Die Arraylänge muss als **zusätzlicher Eingabeparameter** übergeben werden (außer bei Arrays vom Typ `char` - siehe später)

Wie benutzt man ein Array?

Arrays als Funktionsparameter

Arrays als Eingabeparameter

Für einen Eingabeparameter einer Funktion der Form

T <Parameter> []

wird bei Funktionsaufruf ein Arrayname (vom Typ T) übergeben

Ein Array mit Zufallszahlen befüllen

```
1 void array_random(int numbers[], int length) {  
2     for(int i = 0; i < length; ++i) {  
3         numbers[i] = rand();  
4     }  
5 }
```

Wie benutzt man ein Array?

Arrays als Funktionsparameter

Ein Array mit Zufallszahlen befüllen - main-Funktion

```
1 #define MAX_LENGTH 100
2 void array_random(int numbers[], int length);
3 int main(void) {
4     int numbers[MAX_LENGTH];
5     srand(time(NULL));
6     int length = rand() % (MAX_LENGTH + 1);
7     array_random(numbers, length); /* Zufallswerte eintragen */
8     return 0;
9 }
```

- Array wird in `main` deklariert (Zeile 4) und dann an die Funktion übergeben (Zeile 7).
- Zeile 1: Eigene symbolische Konstante definieren (verhindert sog. *Magic numbers* im Code).
- Zeile 6: Array nur teilweise befüllen.

Problemstellung

- Finde für eine **unsortierte** Folge ganzer Zahlen $a_1, \dots, a_n$ und eine ganze Suchzahl s die **erste** Position von s in $a_1, \dots, a_n$ (also das kleinste i mit $a_i = s$).
- Zeige durch einen Fehlerwert an, falls s nicht in $a_1, \dots, a_n$ vorhanden ist

Lösungsidee:

Elemente der Zahlenfolge werden der Reihe nach (sequentiell) durchlaufen und jeweils mit der Suchzahl verglichen, solange die Suchzahl noch nicht gefunden wurde.

- Implementiere die Zahlenfolge durch ein Array
- **Achtung:** Die Arraylänge n muss gesondert übergeben werden
- **Achtung:** Der Arrayindex beginnt bei 0 und nicht bei 1

```
1 int search_unsorted(int a[], int n, int s) {  
2     for(int i = 0; i < n; ++i) {  
3         if (a[i] == s)  
4             return i;  
5     }  
6     return -1;  
7 }
```

Algorithmen für Arrays

Das Maximum einer Folge finden

Problemstellung

Finde in einer **unsortierten** Folge ganzer Zahlen $a_1, \dots, a_n$ das Maximum (den größten Wert)

Lösungsidee:

Durchlaufe die Elemente der Zahlenfolge der Reihe nach (sequentiell), und merke dir nach jedem Element das bisher gefundene Maximum

Algorithmen für Arrays

Das Maximum einer Folge finden

Beispiel

Finde Maximum in der Folge 7, 12, 4, 9:

- Schritt 1 (1. Element 7): Setze das aktuelle Maximum auf 7
- Schritt 2 (2. Element 12): Aktualisiere das aktuelle Maximum auf 12
- Schritt 3 (3. Element 4): Maximum bleibt unverändert
- Schritt 4 (4. Element 9): Maximum bleibt unverändert

Algorithmen für Arrays

Das Maximum einer Folge finden

Lösungsidee:

Durchlaufe die Elemente der Zahlenfolge der Reihe nach (sequentiell), und merke dir nach jedem Element das bisher gefundene Maximum

```
1 int array_max(int a[], int n) {  
2     int max = a[0];  
3     for (int i = 1; i < n; ++i) {  
4         if (a[i] > max) {  
5             max = a[i];  
6         }  
7     }  
8     return max;  
9 }
```

Definition und Deklaration

Strings sind Arrays vom Typ `char` und werden wie folgt deklariert:
`char w[n];`

- `w` ist ein **String** aus maximal `n-1` Zeichen.
- `w` repräsentiert die im Speicher abgelegte Zeichenfolge ab der ersten Speicherzelle des Arrays bis zur ersten Speicherzelle mit dem Inhalt `'\0'`.
- Das Zeichen `'\0'` heißt **binäre Null** und ist das erste Zeichen in der ASCII-Tabelle.

Beispiel

Deklaration eines Strings für maximal 9 Zeichen:
`char w[10];`

Weitere Varianten der Deklaration

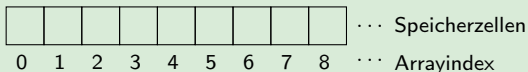
Ein String der Länge $n-1$ kann alternativ wie folgt deklariert werden:

- `char w[] = {c1, ..., cn-1};`
für Zeichen $c_1, \dots, c_{n-1}$. Hierbei bekommt `w[i-1]` den Wert c_i .
- `char w[] = W;`
für eine Stringkonstante W der Länge $n-1$. Hierbei bekommt `w[i-1]` als Wert das i -te Zeichen in W .

Beispiele

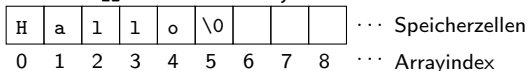
- `char w[] = {'H', 'a', 'l', 'l', 'o'};`
- `char w[] = "Hallo";`

Strings im Speicher



String "Hallo" im Speicher

```
char w[] = "Hallo";
```



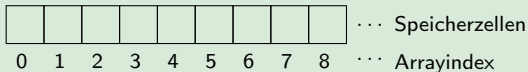
Leerer String (= Länge 0)

```
char w[8];
```

```
w[0] = '\0';
```

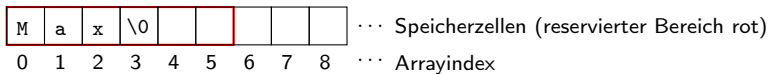


Strings im Speicher

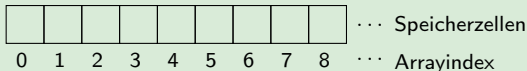


Der reservierte Speicherbereich muss nicht komplett
ausgenutzt werden

```
char w[6];  
strcpy(w, "Max"); /*Kopierfunktion aus string.h */
```

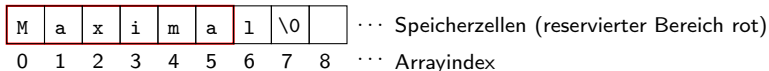


Strings im Speicher



ACHTUNG: C erlaubt, dass man über den reservierten Speicherbereich hinaus schreibt

```
char w[6];  
strcpy(w, "Maximal");
```



Unbedingt vermeiden: Da nachfolgende Variablen im Speicher überschrieben werden, kommt es zu Rechen- und Programmfehlern!

Wie benutzt man Strings?

Strings als Funktionsparameter

Strings als Eingabeparameter

Für einen Eingabeparameter einer Funktion der Form

```
char <Parameter>[]
```

wird bei Funktionsaufruf ein String übergeben

Hinweise

- Die Anzahl der Zeichen gehört **nicht zur Typangabe**:
Es können Strings unterschiedlicher Länge übergeben werden
- Die Länge des Strings muss **nicht** zusätzlich übergeben werden: Das Ende des Strings wird im Speicher durch eine binäre Null markiert.
- Man kann lesend und schreibend auf den String zugreifen.

Wie benutzt man Strings?

Strings als Funktionsparameter

Alternative Deklaration von Strings als Eingabeparameter

- `char * <Parameter>`

Man kann lesend und schreibend auf den String zugreifen.

- `const char * <Parameter>`

Man kann nur lesend auf den String zugreifen.

Beispiel (Bibliotheksfunktion)

```
char *strcpy(char *s, const char *ct):
```

Kopiert ct nach s und schließt s mit der binären Null ab; gibt s zurück.

Wie benutzt man ein Strings?

Strings als Funktionsparameter

Strings als Eingabeparameter

Für einen Eingabeparameter einer Funktion der Form

`char <Parameter>[]`

wird bei Funktionsaufruf ein String übergeben

Die Länge eines Strings berechnen (vereinfachte Version)

```
1 int my_strlen(char w[]) {  
2     int i = 0;  
3     while (w[i] != '\0')  
4         ++i;  
5     return i;  
6 }
```

Wir programmieren die Bibliotheksfunktion `strlen` aus `string.h` nach (vereinfachte Version):

```
1 int my_strlen(char w[]) {  
2     int i = 0;  
3     while (w[i] != '\0')  
4         ++i;  
5     return i;  
6 }
```

- Der Aufruf `my_strlen("Hallo")` gibt 5 zurück
- Strings sind im Speicher immer mit `'\0'` abgeschlossen
- Die Länge des Strings entspricht dem Index `n` mit `w[n] == '\0'`
- String wird in `while`-Schleife so weit durchlaufen, bis Index von `'\0'` gefunden wurde. Dieser Index wird zurückgegeben.

Wir können einen String `w` entweder zeichenweise mit einer Schleife ausgeben (Übungsaufgabe).

Oder mit `printf` und der Umwandlungsangabe `%s`:

```
printf("%s", w);
```

Wie für Arrays gilt:

Strings und Wertzuweisungen

- Eine Wertzuweisung an einen String w (außer direkt in der Deklaration) ist nicht möglich (Compilerfehler, da w konstant)!
- Nur den Buchstaben $w[i]$ des Strings können Werte zugewiesen werden

Strings und Vergleiche

- Ein Vergleich zweier Strings v und w der Form $v == w$ vergleicht Adressen (Ergebnis ist immer 0)!
- Den Inhalt zweier Strings muss man über deren Buchstaben mit $v[i] == w[i]$ vergleichen.
- Dafür gibt es Bibliotheksfunktionen

Wie für Arrays gilt:

Änderung der Stringlänge zur Laufzeit?

- Die Anzahl der Buchstaben wird (in C) in der Deklaration festgelegt und kann danach nicht verändert werden!
- Wird auf nicht reservierten Speicherbereich zugegriffen, erzeugt das keinen Compiler-, sondern einen Laufzeitfehler!
- **Folgerung 1:** Ist die Anzahl der Buchstaben eine Konstante, so wähle diese groß genug für die Anwendung und lege sie einmalig mit `#define` fest (Vermeidung von *Magic Numbers* im Code)
- **Folgerung 2:** Benutze zur Laufzeit nur Zeichenketten, die maximal so lang sind wie der in der Deklaration festgelegte Speicherbereich