Fachhochschule Bingen

Programmieren

Datentypen / Felder / Strings

Prof. Dr. Maximilian Mengel,
Programmiermethodik,
Grundlagen der Informatik und Multimedia
Gebäude 1, Raum 212
Tel.: 06721-409 152
E-Mail: mengel@fh-bingen.de

Datentypen (Wiederholung)

- Ganze Zahlen
  - int
  - short / long int
  - unsigned / signed int
- Gleitpunktzahlen
  - float
  - double
  - long float / long double
- Zeichen
  - char

Weitere Datentypen (Benutzerdefiniert)

- enum
  - Aufzählungstyp
  - Typ-Deklaration*

<enum-typ> ::= enum <enum-Name> { <Wert >{,<Wert>}* };

  - Variablen-Deklaration

<enum-Variable> ::= <enum-type> <Variablen-Name> {,<Variablen-Name>}*;

  - Bespiel:

```
enum Farbe {rot, gruen, blau};
enum Farbe meineLieblingsfarbe;
meineLieblingsfarbe = rot;
```

*** vereinfacht**

Weitere Datentypen (Benutzerdefiniert)

- typedef
  - Neuer Bezeichner (Name) für einen einfachen Datentyp
  - Typ-Deklaration*

<typedef-typ> ::= typedef <einfacher-Datentyp> <typedef-Name>;

  - Variablen-Deklaration

< typedef -Variable> ::= < typedef-Name > <V-Name> {,<V-Name>}*;

  - Bespiel:

```
typedef unsigned int PosZahlen;
PosZahlen eineZahl;
eineZahl = 25;
```

*** vereinfacht**

Felder / Vektoren

- Felder und Vektoren (engl. Arrays) fassen mehrere Variablen gleichen Typs zusammen
- Die Anzahl der Elemente eines Feldes, bzw. Vektors muß eindeutig festgelegt sein
- Die Numerierung der Elemente beginnt immer bei 0 und geht bis Anzahl-1
- C überprüft nicht ob Sie gültige Indizes benutzen
  - Wenn Sie das Element Nr. Anzahl bei obigem Array schreiben, dann schreiben Sie irgendwo im Speicher ihren Wert

Vektoren

- Vektoren sind eindimensionale Felder
- Beispiel:

```
int messreihe[10];
… …
if (messreihe[0] < 0)
… …
for (i=0;i<10;++i)
{
… messreihe[i] …
}
```


Felder

- Felder (Arrays) sind mehrdimensional zusammengehörende Elemente
- Beispiel:

```
char tic-tac-toe-feld[3][3];
… …
if (tic-tac-toe-feld[0][0]==“ “)
… …
```


Felder / Arrays

- Deklaration mit Initialisierung*:

<Feld-var> ::= <Datentyp> <Variable>[= <Var-Init>] {,<Variable>[= <Var-Init>]}*;
<Variable> ::= <V-Name>{[<Zahl>]}+
<Var-Init> ::= {<Konstante> {, <Konstante>}*} | {<Var-Init>{,<Var-Init>}*}

- Beispiel

```
int Vektor[3] = {1,2,3};
int matrix[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
char abc[4] ={‘a‘,‘b‘,‘c‘,‘\0‘};
```

*** vereinfacht und nicht 100% korrekt**

Felder: Besonderheiten

- Bei einer durchgeführten Initialisierung muß man die Zahl der Komponenten nicht festlegen:

```
char abc[] = {'a','b','c','\0'};
int messreihe[] = {3,5,2,7,9,1,15};
```

- Bei scanf(…) kann man auf den Adress-Operator & verzichten, da der Feldname gleichzeitig die Adresse des ersten Elementes darstellt:

```
int a[3]; char name[20];
scanf("%d",a);    /* = scanf("%d",&a[0]); */
scanf("%s",name); /* einlesen eines Strings (char-Array)*/
```


Strings

- Zeichen-Vektoren (char[ ]) werden i.Allg. als Strings bezeichnet
- Strings müssen immer ein ´\0´ als letztes Zeichen besitzen
  - In einem String können nur Anzahl-1 Zeichen (ohne das ´\0´-Zeichen) gespeichert werden
- Für Strings existieren keine Operatoren
- Die Bibliothek string.h enthält Funktionen für den Umgang mit Strings

Einige Funktionen für Strings

- strcpy(String1, String2)
  - kopiert String2 in String1
- strlen(String)
  - liefert die Anzahl der Zeichen in String (ohne das Zeichen `\0`
- strcmp(String1, String2)
  - Vergleicht String1 und String 2 lexikographisch das Resultat ist:
    - < 0 falls String1 < String2 ist
    - = 0 falls String1 == String2 ist
    - > 0 falls String1 > String2 ist

… einige Funktionen für Strings

- strcat(String1, String2)
  - Fügt hinter String1 String2 an. Ergebnis wird zurückgegeben
- strncpy(String1, String2, Anzahl)
  - Kopiert maximal *Anzahl* Zeichen aus String2 in String1
- strncat(String1, String2, Anzahl)
  - Fügt hinter String1 maximal *Anzahl* Zeichen aus String2 an. Das Resultat wird zurückgegeben.
- strchr(String1, Zeichen)
  - Sucht das erste Auftreten von *Zeichen* in String1. Der Teilstring ab dem Zeichen wird zurückgegeben.

Beispiel

```
#include <stdio.h>
#include <string.h>
main()
{
char string1[] = "Hallo !";
char string2[] = {‘A’,‘B’,'\0'};
char string3[20];
strcpy(string3,string1);
printf(“Nach strcpy() ist String 3 = %s\n",string3);
strcpy(string3,strcat(string1,string2));
printf((" Nach strcat() und strcpy() ist String 3 = %s\n", string3);
printf("\n strcmp(%s,%s) = %d\n",string1,string2,strcmp(string1,string2));
printf("String3 eingeben\n");
scanf("%s",string3);
printf("Der eingegebene String (%s) hat %d Zeichen\n",string3, strlen(string3));
}
```


Typische String-Fehler

```
#include <stdio.h>
#include <string.h>
main()
{
char string1[3] = "Hallo!";          /* Fehler String ist zu klein für Wert*/
char string2[3] = {'A','B','C'};    /* Fehler \0 fehlt */
char string3[5] = “HALLO!“;         /* Fehler \0 fehlt, bzw. kein Platz */
string3 = “AB“;                     /* Fehler keine Zuweisung möglich */
string3 = string2;                  /* Fehler keine Zuweisung möglich */
string3 = strcat(string1,string2);  /* Fehler keine Zuweisung möglich */
strcpy(string1,“HALLO!“);           /* Fehler String ist zu klein für Wert*/
scanf("%s",&string3);               /* Fehler Adress-Operator & ist zuviel */
printf("String3:%s\n",string3[0]);  /* Fehler String erwartet, Char uebergeben */
}
```


Beispiel Programm

```
#include <stdio.h>
#include <string.h>
main()
{
char string1[] = “?Hallo!“, ch;
int i;
ch = string1[0];
for (i=0;i<strlen(string1)-1;++i)
   string1[i] = string1[i+1];
string1[strlen(string1)-1] = ch;
printf("String:%s\n",string1);
}
```

- Was gibt das Programm aus?
- Was passiert wenn strlen(string1) -1 durch strlen(string1) ersetzt wird?
- Wie sieht ein entsprechendes Programm aus, indem möglichst wenig mit einzelnen Zeichen gearbeitet wird?

Beispiel Programm

```
#include <stdio.h>
#include <string.h>
main()
{
char string1[] = “?Hallo!", ch;
ch = string1[1];
strcpy(string1,strchr(string1,ch));
printf("String:%s\n",string1);
}
```

- Was passiert wenn string1[ ] = “HHallo!“?