LANDESBERUFSSCHULE 4 SALZBURG

Informatik

Felder, Zeichenketten, Array

LBS 4

Dieses Skript dient als zusätzliche Lernunterlage für Informatik

Inhalt

Felder (Arrays)

Bis jetzt haben Sie einen Wert in einer Variablen abgespeichert. Stellen Sie sich vor, Sie müssen ein Programm schreiben das die Tagestemperaturen von einer Woche speichern soll.
Es besteht natürlich die Möglichkeit für jeden Wochentag eine eigene Variable zu deklarieren. Besser ist es aber wenn die Werte in einer einzigen Variable gespeichert werden können.

Für diesen Zweck gibt es unter C die Felder oder Arrays. Mit diesen können mehrere Werte in einer einzigen Variablen gespeichert werden.

Deklarieren von Feldern:

Das Array wird genauso wie eine normale Variable deklariert. Der Unterschied ist die Dimension am Ende der Deklaration. Diese wird in eckigen Klammern mit einer Zahl ausgeführt. Die Zahl bestimmt die Anzahl der einzelnen Felder.

Datentyp Name Dimension

int temp [7];

Die einzelnen Felder können über den Index angesprochen werden

Vorsicht:!! Der Index beginnt bei 0 an zu zählen

Nachstehend ist ein Array mit Index zu sehen.

Index 0 Feld1	Index 1 Feld2	Index 2 Feld 3	Index 3 Feld 4	Index4 Feld 5	Index5 Feld 6	Index6 Feld 7

Bsp.:

```
temp[0] = 1; // erstes Feld im Array
temp[3] = 12; //viertes Feld im Array
temp[1] = 3; // zweites Feld im Array
```

Arrays können auch mehrdimensionale Felder enthalten. So kann man eine Matrix folgenderweise deklarieren:

Beispiel:

```
int temp [5] [5];
```

Dieses Beispiel würde eine zweidimensionale Matrix bereitstellen.
Nachstehend wird der schematische Zugriff auf die einzelnen Felder gezeigt.

[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]
[2][0]	[2][1]	[2][2]	[2][3]	[2][4]
[3][0]	[3][1]	[3][2]	[3][3]	[3][4]
[4][0]	[4][1]	[4][2]	[4][3]	[4][4]

Beispiel: Eindimensionales Array C-Code

```
#include <stdio.h>

int main(void) {
	int i;
	double temperatur[31];
	double gesamt=0,durchschnitt=0;

	printf("\t\tTagestemperaturen des Monats Juli\n\n");

	for(i=0; i<=30; i++) {	/* Eingabe aller Werte */
		printf("%2i. Tag : ",i+1);
		scanf("%lf",&temperatur[i]);
	}
	printf("\nAlle Werte des August.\n");
	//Werte aller Tage ausgeben
	for(i=0; i<=30; i++)	/* Ausgabe aller Werte */
		printf("\n%2i. Tag : %2.1f Grad",i+1,temperatur[i]);

	//Durchschnittswert berechnen
	for(i=0; i<=30; i++)	/* Durchschnittstemperatur */
		gesamt += temperatur[i];
	durchschnitt = gesamt / 31;
	printf("\n\nDurchschn. Temperatur = %2.1f Grad.",durchschnitt);
}
```

Zeichenketten

Zeichenketten sind ein wichtiges Element in Programmiersprachen. Sie dienen für Meldungen wie, Fehler, Mitteilungen, usw. an den Benutzer. Die Programmiersprache stellt keinen Datentyp für Zeichenketten (engl. strings) zur Verfügung.

Zeichenketten werden in C mittels char-Arrays dargestellt. Obwohl es keinen Datentyp für Strings gibt, stellt die Programmiersprache C ca. 20 Funktionen zum Bearbeiten von Zeichenketten zur Verfügung.

Deklarieren von Zeichenketten:

char text [6] = {‘H’, ‘a’, ‘l’, ‘l’, ‘o’};

char string [5];

Im Speicher sieht das Feld folgendermaßen aus.

H	a	l	l	o	\0

An jedem Ende einer Zeichenkette muss diese terminiert (abgeschlossen) werden. Das geschieht mit **der binären NULL** '\0'

Man muss bei der Deklaration die Länge der Zeichenkette + 1 angeben!

Die binäre NULL muss nur in Ausnahmefällen gesetzt werden.

gets()
Zum Einlesen von Zeichenketten muss nicht unbedingt scanf() verwendet werden. Einfacher geht es mit gets().

Die Funktion gets() erwartet als Parameter eine Zeichenkette.

gets(char *string)

Damit printf() eine Zeichenkette ausgibt wird der Platzhalter **%s** benötigt.

Im nächsten Abschnitt lernen Sie einige Zeichenkettenfunktionen kennen.

strcat(), strlen(), strcpy() und strcmp()

Zeichenkettenfunktionen:

Funktionen für Zeichenketten fügen automatisch die binäre NULL am Ende der Zeichenkette ein.

Die Funktionen für Zeichenketten sind in der **<string.h>** definiert.

strlen() zählt die Anzahl der Zeichen im Array. Damit diese Funktionen verwendet werden können benötigen Sie die Header-Datei **<string.h>.**

Das Beispiel nimmt eine Zeichenkette entgegen und zählt die Anzahl der einzelnen Zeichen. Anschließend wird das Ergebnis ausgegeben.

Beispiel C-Code:

```
#include <stdio.h>
#include <string.h>

main() {
        char satz[80];
        printf("\nBitte einen Satz eingeben\n>");
        gets(satz);
        printf("\n%s",satz);
        printf("\nDer Satz besteht aus %i Zeichen.",strlen(satz));
return 0;
}
```

Im nachstehenden Beispiel wird an der 6. Stelle der Zeichenkette eine **binäre Null** eingetragen.
Welcher Wert wird von strlen() ermittelt?

```
#include <stdio.h>
#include <string.h>

main() {
        char satz[80];
        printf("\nBitte einen Satz eingeben\n>");
        gets(satz);
        printf("\n%s",satz);
        printf("\nDer Satz besteht aus %i Zeichen.",strlen(satz));
        printf("\n\nIch schreibe in satz[5] eine binaere Null.\n");
        satz[5]='\0';
        printf("\n%s",satz);
        printf("\nDer Satz besteht aus %i Zeichen.",strlen(satz));
}
```

Aufgabe: Schreiben Sie ein Programm, dass eine Zeichenkette entgegen nimmt und in einer while-Schleife Zeichen für Zeichen in der Konsole ausgibt.

Für die nachstehenden Beispiele wird das Array

char text [10];

0	1	2	3	4	5	6	7	8	9

strcpy()
Diese Funktion kopiert eine Zeichenkette in eine andere. Als Parameter erwartet die Funktion ein Feld (Parameter 1) und ein Feld oder eine Konstante für den Parameter 2. Aufruf der Funktion ist nachstehend dargestellt.

strcpy(zeichenkette1, zeichenkette2);

Bsp.:

```
strcpy(text,"soft")
```

kopiert den Text “soft“ in das Feld text.

0	1	2	3	4	5	6	7	8	9
s	o	f	t	\0					

strcat()
Diese Funktion hängt an eine bestehende Zeichenkette eine neu an. strcat() erwartet ebenfalls wie strcmp() zwei Zeichenketten. Die Anweisung lautet:

```
strcat(text,"ware");
```

fügt den Text "ware" an die bestehende Zeichenkette an. Die **binäre NULL** wird an das Ende der Zeichenkette angefügt.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
s	o	f	t	w	a	r	e	\0	

strcmp()

Die Funktion vergleicht zwei Zeichenketten und gibt als Rückgabewert das Ergebnis an

- wenn die erste Zeichenkette kleiner als die zweite ist wird als Ergebnis <0 (negativ) zurückgegeben
- wenn die Zeichenketten gleich sind wird als Ergebnis 0 zurückgegeben
- wenn die erste Zeichenkette größer ist wird >0 (positiv) zurückgegeben.

Der Aufruf der Funktion erfolgt analog zu den anderen Funktionen:

```
strcmp(<zeichenkette1>, <zeichenkette2>)
```