LANDESBERUFSSCHULE 4 SALZBURG

Informatik

Struktur, Aufzählungstyp

LBS 4

Dieses Skript dient als zusätzliche Lernunterlage für Informatik

Inhalt

Struktur

In C gibt es die Möglichkeit eigene Datentypen zu erstellen. Das ist besonders wichtig für logisch, zusammengehörige Daten. Eine Adresse ist ein logisch zusammengehörender Datensatz der in einem Element abgespeichert werden soll. Die einzelnen Daten werden mit einer Struktur als einzelner Datentyp behandelt. Strukturen können aus beliebigen Datentypen zusammengestellt werden.

Syntax einer Struktur:

struct Name { Daten, Daten, Daten, ….}

struct definiert einen neuen Datentyp: NAME des Datentyp

Daten: einzelne Variablen, Arrays → int age; char *name[50];

Beispiel in C:

```
struct address
{
  char  name[50];
  char street[20];
  char  tel [15];
  int age;
};
```

Am Ende der Definition muss ein Semikolon stehen!

Damit die Struktur verwendet werden kann, muss eine Variable deklariert werden.

```
struct address huber;
```

Damit Speicher reserviert wird muss die Strukturvariable initialisiert werden. Zum Zugriff auf die einzelnen Elemente in der Struktur wird der Operator „.“ (Punkt) verwendet.

Beispiel: initialisieren

```
huber.age = 34;
strcpy(huber.name, "Helmut Huber“);
…….
```

Die Struktur kann aber auch bei der Deklaration initialisiert werden.

Beispiel: initialisieren II

```
struct address huber ={
"Helmut Huber“,
"Schießstattstraße 5“,
"+435671298",
34
};
```

Weiteres haben Sie die Möglichkeit aus Strukturvariablen Arrays zu erstellen. Der Vorgang ist analog zu Arrays mit primitiven Datentypen (int,char,float,…)

```
struct address customer[4];
……..
customer[2].name = "Helmut Huber“;
```

Für Zeiger von Strukturen kann anstelle des Dereferenzierungsoperators ***** und den Punktoperator **.** der Pfeiloperator **->** verwendet warden.

```
(*a).age = 12;
a->age = 12;
```

Beispielcode C:

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

struct address{ //Struktur definieren
  char  name[50];
  char street[20];
  char  tel [15];
  int age;
  int set;
};

int setData(struct address *a){ //Funktion zum Befüllen der Struktur

if(a->set){
printf("Daten vorhanden!\n");
return -1;
}
printf ("Name: ");
fgets (a->name, (strlen(a->name))-1, stdin); //Kurschreibweise a->name für (*a).name
scanf ("%d", &a->age); //
a->set = 1;

}
int main() {
        struct address customer[10]; //Array deklarieren
int check = 3;
for(i = 0; i < 10;i++){
   customer[i].set = 0; //set auf 0 setzen
}
if(&customer[1].set == 1){          //Abfrage ob Daten schon gesetzt wurden
printf("Daten vorhanden!\n");
}
else{
printf("Daten wurden geschrieben\n");
}

return 0;
}
```

Aufzählungstyp: enum

Enum dient zur Aufzählung von Konstanten. Diese werden in einem Datentyp zusammengefasst. Enum entspricht einer Struktur für Aufzählugen.

Syntax von enum

Enum NAME { Aufzählungen};

Enum: Schlüsselwort
NAME: Name der Struktur
Werte der Aufzählungen

Beispiel:

```
enum num { one, two, three};
```

C nummeriert die Konstanten von 0 aufwärts. Wenn man bestimmte Zahlenwerte für die Konstanten einsetzen will, kann man diese in der Deklaration bestimmen.

```
enum num { one = 1, two = 2, three = 3};
```

Beispiel C:

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

enum num { one = 12, two, three}; //definieren

#include <stdio.h>
#include <stdlib.h>

enum { kreuz, pik, karo, herz };


int main(void) {

printf("Wert der Karte: %d\n", pik);
printf("Wert von two: %d\n", two);

   return 0;
}
```