Einführung in die Programmiersprache C

Alexander Batoulis

3. Mai 2014

Beispielprogramm in C

```c
#include <stdio.h>
int main()
{
   printf("Völlig isoliert: E-techniker sucht Kontakte \n");
   return 0;
}
```

- *include* nicht vergessen
- Rückgabewert der *main*-Funktion muss *0* sein
- Escape-Sequenzen bei dem *printf*-Befehl verwenden

C ist nicht objektorientiert

```c
public class Car{
   public String Name;
   public void drive() {...}
}
public static void main(String[] args) {
   Car lancia = new Car();
   lancia.drive();
        }
```

C ist *nicht* objektorientiert. Es ist nicht möglich, aus eigens geschriebenen Klassen Objekte zu erzeugen

Zeiger in C

```c
printf("Adresse Zahl: %p \n", &zahl);
printf("Wert Zahl: %d \n", zahl);
printf("Adresse pointer: %p \n", &pointer);
printf("Wert pointer: %p \n", pointer);
printf("Wert von Adresse %p %d \n", pointer, *pointer);
```

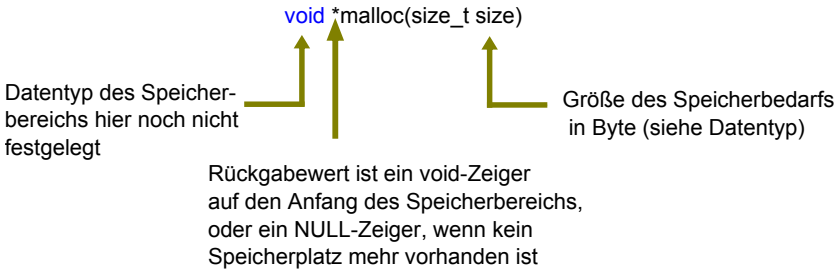
- speichert Adresse einer anderen Variable

→ leitet Anfrage auf einen Wert weiter

- sollte selben Variablentyp haben wie die Variable auf die er zeigt

Ausgabe:
Adresse pointer: 0028FEE8
Wert pointer: 0028FEEC
Wert von Adresse 0028FEEC 7
Adresse Zahl: 0028FEEC
Wert Zahl: 7

Memory Allocation



```c
int size=0;
int *array;

printf("Array-Groesse eingeben: ");
scanf("%d", &size);
// Speicher reservieren
array = (int *) malloc(size * sizeof(int));
if(array != NULL)
   printf("\nSpeicher ist reserviert\n");
else
   printf("\nKein freier Speicher vorhanden.\n");
```

- Datentyp *size_t* vom Typ *long int*

→ wird für Angabe einer Datengröße in Byte verwendet

- Funktion *sizeof(Datentyp)* gibt bei Übergabe eines Datentyps (*int, double, char,...*) benötigte Speichergröße in Byte an

→ Rückgabewert ist vom Datentyp *size_t*

- *calloc* (Cleared Memory Allocation) macht das selbe wie *malloc*, es werden jedoch alle Speicherstellen mit *0* initialisiert

Aufzählungen und Strukturen

```c
const int mo = 0,di = 1,mi = 2,do = 3,fr = 4,sa = 5,so = 6;
enum {mo, di, mi, do, fr, sa, so};
        int projektlabor = do;
```

Mit *enum* können Variablen direkt zusammengefasst werden.

```c
struct datum {
   int tag;
   char monat[10]
   int jahr;
   };
struct datum ferien;
ferien.jahr = 2014;
struct datum geburtstag = (17, "Dezember", 1999);
```

Zunächst zusammenhanglose Variablen können mittels der Funktion struct strukturiert werden. Unbedingt darauf achten, das Semikolon hinter dem *struct* zu setzen.

Funktionen und Funktionsprototypen

```c
#include <stdio.h>
float eingabeZahl();
float multi(float zahl1, float zahl2);
void ausgabeErgebnis(float ergebnis);
int main(){...}
// Funktionen
float eingabeZahl() {...}
float multi(float zahl1, float zahl2) {...}
void ausgabeErgebnis(float ergebnis) {...}
```

- werden wie in Java geschrieben
- Rückgabetyp Funktionsname (Parameterliste) { Anweisungen }
- jede Funktion muss über der main-Funktion stehen oder mittels Funktions-Prototyp deklariert werden

Bei den Funktionsprototypen müssen lediglich Rückgabetyp, Funktionsname und die Parameterliste angegeben werden. Die Anweisungen der jeweiligen Funktion werden im Verlauf des Programms wie gewohnt geschrieben.

Quellen

http://de.wikipedia.org/
http://www.c-howto.de/
http://de.wikibooks.org/wiki/C-Programmierung