Hardwarenahe Software-Entwicklung

Hardwarenahe Software-Entwicklung

1 Einführung in C

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: while

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: while, for

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: while, for, do−while

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe
  - „Haupteffekt“: Rückgabe der Anzahl ausgegebener Zeichen

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe
  - „Haupteffekt“: Rückgabe der Anzahl ausgegebener Zeichen

  Operatoren mit Seiteneffekten: −− ++

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe
  - „Haupteffekt“: Rückgabe der Anzahl ausgegebener Zeichen

  Operatoren mit Seiteneffekten: −− ++ = += −= ∗= /= %=

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe
  - „Haupteffekt“: Rückgabe der Anzahl ausgegebener Zeichen

  Operatoren mit Seiteneffekten: −− ++ = += −= ∗= /= %=
- **Funktionen** schreiben

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe
  - „Haupteffekt“: Rückgabe der Anzahl ausgegebener Zeichen

  Operatoren mit Seiteneffekten: −− ++ = += −= ∗= /= %=
- **Funktionen** schreiben
  Gültigkeitsbereiche von Variablen

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe
  - „Haupteffekt“: Rückgabe der Anzahl ausgegebener Zeichen

  Operatoren mit Seiteneffekten: −− ++ = += −= ∗= /= %=
- **Funktionen** schreiben
  Gültigkeitsbereiche und Initialisierung von Variablen

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe
  - „Haupteffekt“: Rückgabe der Anzahl ausgegebener Zeichen

  Operatoren mit Seiteneffekten: −− ++ = += −= ∗= /= %=
- **Funktionen** schreiben
  Gültigkeitsbereiche und Initialisierung von Variablen
- **Zeiger**

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe
  - „Haupteffekt“: Rückgabe der Anzahl ausgegebener Zeichen

  Operatoren mit Seiteneffekten: −− ++ = += −= ∗= /= %=
- **Funktionen** schreiben
  Gültigkeitsbereiche und Initialisierung von Variablen
- **Zeiger**, Zeiger-Arithmetik

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe
  - „Haupteffekt“: Rückgabe der Anzahl ausgegebener Zeichen

  Operatoren mit Seiteneffekten: −− ++ = += −= ∗= /= %=
- **Funktionen** schreiben
  Gültigkeitsbereiche und Initialisierung von Variablen
- **Zeiger**, Zeiger-Arithmetik
- **Arrays**

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe
  - „Haupteffekt“: Rückgabe der Anzahl ausgegebener Zeichen

  Operatoren mit Seiteneffekten: −− ++ = += −= ∗= /= %=
- **Funktionen** schreiben
  Gültigkeitsbereiche und Initialisierung von Variablen
- **Zeiger**, Zeiger-Arithmetik
- **Arrays** sind i. w. dasselbe wie Zeiger

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe
  - „Haupteffekt“: Rückgabe der Anzahl ausgegebener Zeichen

  Operatoren mit Seiteneffekten: −− ++ = += −= ∗= /= %=
- **Funktionen** schreiben
  Gültigkeitsbereiche und Initialisierung von Variablen
- **Zeiger**, Zeiger-Arithmetik
- **Arrays** sind i. w. dasselbe wie Zeiger:
  foo[i] ist Abkürzung für ∗(foo + i)

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe
  - „Haupteffekt“: Rückgabe der Anzahl ausgegebener Zeichen

  Operatoren mit Seiteneffekten: −− ++ = += −= ∗= /= %=
- **Funktionen** schreiben
  Gültigkeitsbereiche und Initialisierung von Variablen
- **Zeiger**, Zeiger-Arithmetik
- **Arrays** sind i. w. dasselbe wie Zeiger:
  foo[i] ist Abkürzung für ∗(foo + i)
- **Strings**

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe
  - „Haupteffekt“: Rückgabe der Anzahl ausgegebener Zeichen

  Operatoren mit Seiteneffekten: −− ++ = += −= ∗= /= %=
- **Funktionen** schreiben
  Gültigkeitsbereiche und Initialisierung von Variablen
- **Zeiger**, Zeiger-Arithmetik
- **Arrays** sind i. w. dasselbe wie Zeiger:
  foo[i] ist Abkürzung für ∗(foo + i)
- **Strings** sind $\left\{ \begin{array}{c} \text{Arrays von} \\ \text{Zeiger auf} \end{array} \right\}$ **char**s

Hardwarenahe Software-Entwicklung

1 Einführung in C

- **Schleifen**: **while**, **for**, **do−while**
- Ausdruck als Anweisung: Wert wird ignoriert
  nur sinnvoll, wenn Ausdruck **Seiteneffekt** hat

  Beispiel: printf()
  - Seiteneffekt: Ausgabe
  - „Haupteffekt“: Rückgabe der Anzahl ausgegebener Zeichen

  Operatoren mit Seiteneffekten: −− ++ = += −= ∗= /= %=
- **Funktionen** schreiben
  Gültigkeitsbereiche und Initialisierung von Variablen
- **Zeiger**, Zeiger-Arithmetik
- **Arrays** sind i. w. dasselbe wie Zeiger:
  foo[i] ist Abkürzung für ∗(foo + i)
- **Strings** sind $\left\{ \begin{array}{c} \text{Arrays von} \\ \text{Zeiger auf} \end{array} \right\}$ **char**s, d. h. ganze/n Zahlen

1.8 Arrays und Strings

Parameter des Hauptprogramms

```
#include <stdio.h>

int main (int argc, char **argv)
{
   printf ("argc = %d\n", argc);
   for (int i = 0; i < argc; i++)
      printf ("argv[%d] = \"%s\"\n", i, argv[i]);
   return 0;
}
```

1.8 Arrays und Strings

Parameter des Hauptprogramms

```
#include <stdio.h>

int main (int argc, char **argv)
{
   printf ("argc = %d\n", argc);
   for (int i = 0; argv[i]; i++)
      printf ("argv[%d] = \"%s\"\n", i, argv[i]);
   return 0;
}
```

1.9 Strukturen

```
#include <stdio.h>

typedef struct
{
   char day, month;
   int year;
}
date;

int main (void)
{
   date today = { 11, 4, 2012 };
   printf ("%d.%d.%d\n", today.day, today.month, today.year);
   return 0;
}
```

1.9 Strukturen

```
#include <stdio.h>

typedef struct
{
   char day, month;
   int year;
}
date;

void set_date (date *d)
{
   (*d).day = 11;
   (*d).month = 4;
   (*d).year = 2012;
}
```

```
int main (void)
{
   date today;
   set_date (&today);
   printf ("%d.%d.%d\n", today.day,
            today.month, today.year);
   return 0;
}
```

1.9 Strukturen

```
#include <stdio.h>

typedef struct
{
   char day, month;
   int year;
}
date;

void set_date (date *d)
{
   d->day = 11;
   d->month = 4;
   d->year = 2012;
}
```

foo->bar ist Abkürzung für (*foo).bar

```
int main (void)
{
   date today;
   set_date (&today);
   printf ("%d.%d.%d\n", today.day,
            today.month, today.year);
   return 0;
}
```

1.9 Strukturen

```
#include <stdio.h>

typedef struct
{
   char day, month;
   int year;
}
date;

void set_date (date *d)
{
   d->day = 11;
   d->month = 4;
   d->year = 2012;
}
```

Aufgabe

Schreiben Sie eine Funktion
inc_date (date *d),
die ein gegebenes Datum d
auf den nächsten Tag setzt.

Schaltjahre beachten!

```
int main (void)
{
   date today;
   set_date (&today);
   printf ("%d.%d.%d\n", today.day,
            today.month, today.year);
   return 0;
}
```