Technische Informatik 3 – Embedded Systems

Kapitel 2: Zeiger

Prof. Dr. Benjamin Kormann
Fakultät für Elektro- und Informationstechnik
31.03.2023

Variablen – Verfügbarkeit und Lebensdauer

Lebensdauer von Variablen

- automatisch: Variable wird bei Eintritt in den Anweisungsblock erzeugt und beim Verlassen wieder gelöscht
- statisch: Variable wird zu Programmbeginn einmal deklariert und initialisiert und existiert während der gesamten Programmlaufzeit

Verfügbarkeit von Variablen

- lokal: Innerhalb des Anweisungsblocks
- global (Datei): globale Variablen
- global (Programm): Variable befindet sich in einer anderen Datei. Schlüsselwort `extern` wird benötigt

→ Es wird die “lokalste” Variable verwendet.

```c
#include <stdio.h>

// Globale Variable
int myGlobalVariable = 73;
// Externe Variable
extern int myExternalVariable;

void myFunction()
{
    // Variable wird nach Funktionsende gelöscht
    int i = 5;
    printf("%d\n", i);
    i++;
}

void myFunction2()
{
    // Variable bleibt nach Funktionsende erhalten
    static int i = 0;
    printf("%d\n", i);
    i++;
}

int main()
{
    // lokale Variable
    int myLocalVariable = 12;

    // Funktionsaufrufe
    myFunction();
    myFunction();
    myFunction2();
    myFunction2();

    return 0;
}
```

Ausgabe

```
5
5
0
1
```

Variablen – Typumwandlung

Bei arithmetischen Operationen wird auf den höheren Typ umgewandelt

```
float f1 = 2.2;
int   i1 = 8;

printf("%f\n", f1+i1);
```

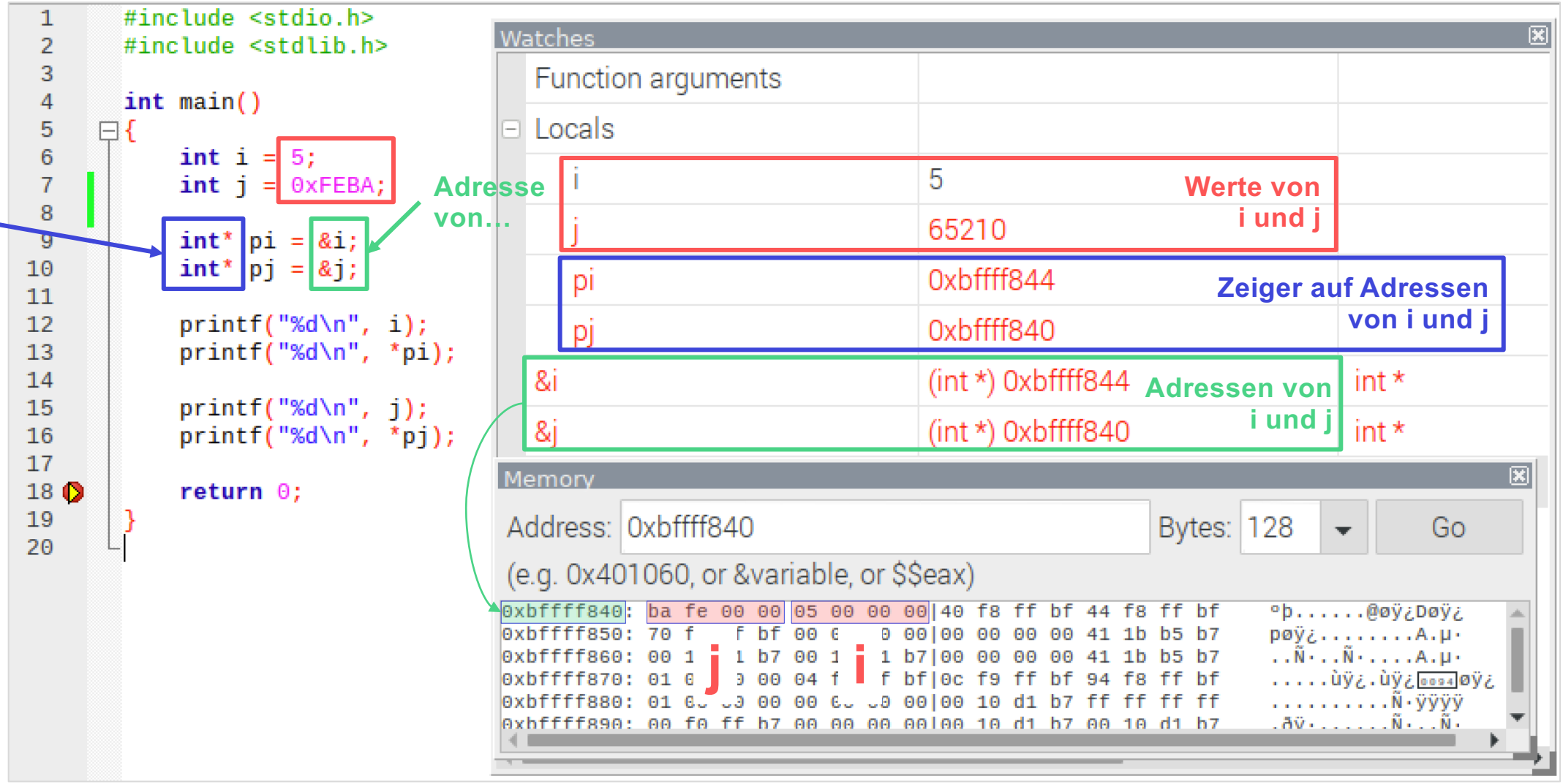
Bei Operanden gleichen Typs bleibt das Ergebnis von diesem Typ (Vorsicht: `int`)

```
int v1 = 100;

double v2 = v1 * 1000 / 3600;
double v3 = 1000 / 3600 * v1;

printf("%d\n", v1);
printf("%lf\n", v2);
printf("%lf\n", v3);
```

Explizite Typumwandlung (type cast)

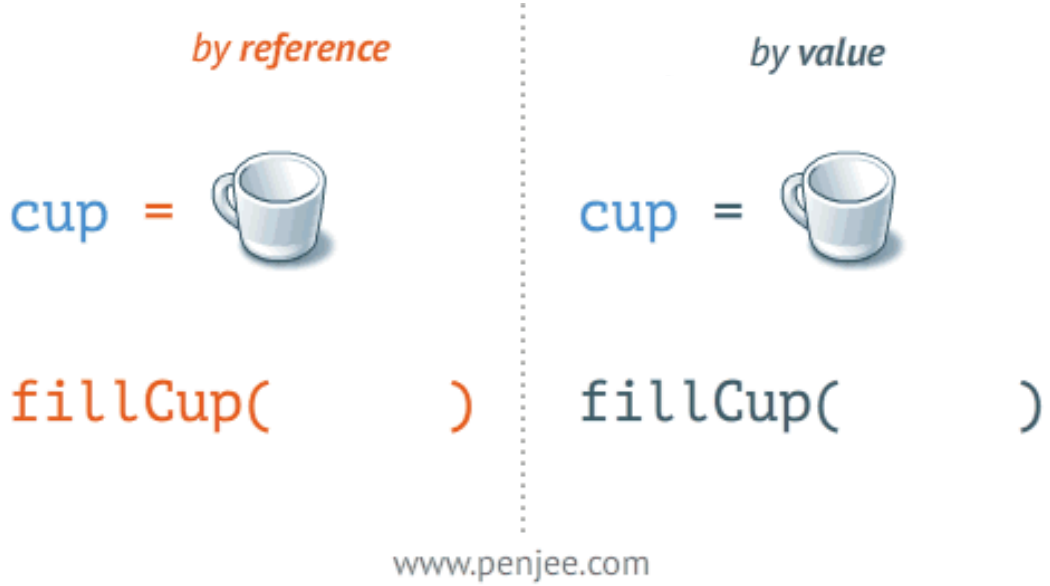
```
int v1 = 100;

double v2 = (double)v1 * 1000 / 3600;
double v3 = 1000.0 / 3600 * v1;

printf("%d\n", v1);
printf("%lf\n", v2);
printf("%lf\n", v3);
```

Zeiger – Datenablage von Variablen im Speicher




```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      int i = 5;
7      int j = 0xFEBA;
8
9      int* pi = &i;
10     int* pj = &j;
11
12     printf("%d\n", i);
13     printf("%d\n", *pi);
14
15     printf("%d\n", j);
16     printf("%d\n", *pj);
17
18     return 0;
19 }
20
```

Zeiger – Parameter von Funktionen
call-by-value und call-by-reference



Zeiger – Parameter von Funktionen
call-by-value und call-by-reference

Übergabe als call-by-value

- Parameter als lokale Kopie
- Die lokale Kopie wird beim Verlassen wieder gelöscht
- Der ursprüngliche Variablenwert wird beibehalten

Übergabe als call-by-reference

- Es wird ein lokaler Zeiger angelegt
- Der Wert der ursprünglichen Variable, auf die der Zeiger zeigt kann verändert werden
- Der lokale Zeiger wird beim Verlassen wieder gelöscht
- Diese Form der Parameterübergabe ermöglicht Übergabeparameter als sog. `out`-Parameter zu verwenden

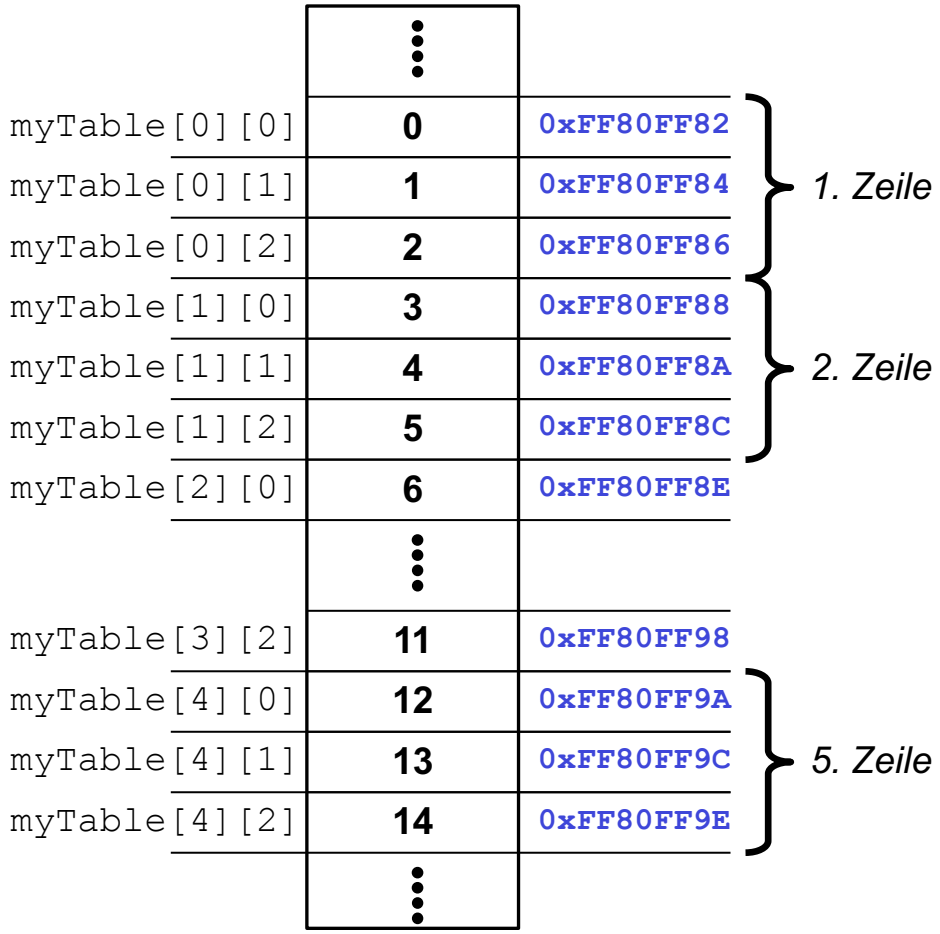
```c
#include <stdio.h>

// call-by-value
void swap1(int a, int b)
{
    int tmp = a;
    a       = b;
    b       = tmp;
}

// call-by-reference
void swap2(int* pa, int* pb)
{
    int tmp = *pa;
    *pa     = *pb;
    *pb     = tmp;
}


int main()
{
    int a = 5;
    int b = 7;

    // Wertausgabe
    printf("%d und %d\n", a, b);

    swap1(a, b);
    printf("%d und %d\n", a, b);
    swap2(&a, &b)
    printf("%d und %d\n", a, b);

    return 0;
}
```

Ausgabe

```
5 und 7
5 und 7
7 und 5
```

Zeiger – Grundlage für (mehrdimensionale) Arrays

```c
#include <stdio.h>

#define ROWS 5
#define COLS 3

int main()
{
    short value = 0;
    // Deklaration short-Array
    short myTable[ROWS][COLS];

    // Array mit Werten füllen
    for(int i = 0; i < ROWS; i++)
    {
        for(int j = 0; j < COLS; j++)
        {
            // Werte: 0, 1, 2, ..., 14
            myTable[i][j] = value++;
        }
    }

    return 0;
}
```



Zusammenfassung

Zeiger Cheat Sheet

Variablen

- Lebensdauer: automatisch und statisch
- Verfügbarkeit: lokal, global

Typumwandlungen

- Explizite Konvertierung durch casting mit ()-Klammerung
- Implizite Konvertierung hin zum höheren Typ (Vorsicht bei `int`-Berechnungen)

Zeiger

- Datenablage in Speicher (Syntax: * zur Zeigerdeklaration und Dereferenzierung, & als Adressoperator)
- Parameter: call-by-value und call-by-reference
- Arrays werden in C durch Zeiger abgebildet; Mehrdimensionale Arrays durch Zeiger auf Zeiger