

Wintersemester 2006/07

**Einführung in die Informatik für  
Naturwissenschaftler und Ingenieure  
(alias Einführung in die Programmierung)  
(Vorlesung)**

Prof. Dr. Günter Rudolph

Fachbereich Informatik

Lehrstuhl für Algorithm Engineering





## Inhalt

- Funktionen
  - mit / ohne Parameter
  - mit / ohne Rückgabewerte
- Übergabemechanismen
  - Übergabe eines Wertes
  - Übergabe einer Referenz
  - Übergabe eines Zeigers
- Programmieren mit Funktionen



### Wir kennen bisher:

- Datentypen zur Modellierung von Daten (inkl. Zeiger)
- Kontrollstrukturen zur Gestaltung des internen Informationsflusses

⇒ Damit lassen sich – im Prinzip – alle Programmieraufgaben lösen!

Wenn man aber

**mehrfach das gleiche** nur mit verschiedenen Daten tun muss,

dann müsste man

den **gleichen Quellcode mehrfach** im Programm stehen haben!

⇒ unwirtschaftlich, schlecht wartbar und deshalb fehleranfällig!



### Funktion in der Mathematik:

$$f: \mathbb{R} \rightarrow \mathbb{R}$$

$$f(x) = \sin(x)$$

$y = f(0.5)$  führt zur

- Berechnung von  $\sin(0.5)$ ,
- Rückgabe des Ergebnisses,
- Zuweisung des Ergebnisses an Variable  $y$ .

$z = f(0.2)$  an anderer Stelle führt zur

- Berechnung von  $\sin(0.2)$ ,
- Rückgabe des Ergebnisses,
- Zuweisung des Ergebnisses an Variable  $z$ .



## Funktionen in C++

```
int main() {  
    double x = 0.5, y, z;  
  
    y = sin(x);  
    z = sin(0.2);  
  
    std::cout << y << " " << z << std::endl;  
    return 0;  
}
```

Achtung!  
`main()` ist Funktion!  
Nur 1x verwendbar!

Die Funktion `sin(·)` ist eine Standardfunktion.

Standardfunktionen werden vom Hersteller bereitgestellt und sind in Bibliotheken abgelegt. Bereitstellung durch `#include` – Anweisung: `#include <math.h>`

Programmierer kann eigene, benutzerdefinierte Funktionen schreiben.



### Welche Arten von Funktionen gibt es?

- a) Funktionen ohne Parameter und ohne Rückgabewert: `clearscreen()` ;
- b) Funktionen mit Parameter aber ohne Rückgabewert: `background(blue)` ;
- c) Funktionen ohne Parameter aber mit Rückgabewert: `uhrzeit = time()` ;
- d) Funktionen mit Parameter und mit Rückgabewert: `y = sin(x)` ;

### Konstruktionsregeln für

- Standardfunktionen und
- benutzerdefinierte Funktionen

sind gleich



## (a) Funktionen ohne Parameter und ohne Rückgabewert

- Funktionsdeklaration:

`void Bezeichner ();`

Prototyp der Funktion



Nichts zwischen Klammern  $\Rightarrow$  keine Parameter

Name der Funktion

`void` (= leer) zeigt an, dass kein Wert zurückgegeben wird



### (a) Funktionen ohne Parameter und ohne Rückgabewert

- Funktionsdefinition:

```
void Bezeichner () {  
  
    // Anweisungen  
  
}
```

```
// Beispiel:  
void zeichne_sterne() {  
    int k = 10;  
    while (k--) std::cout << '*';  
    std::cout << std::endl;  
}
```

#### **Achtung:**

Variable, die in einer Funktion definiert werden, sind nur innerhalb der Funktion gültig.

Nach Verlassen der Funktion sind diese Variablen ungültig!



### (a) Funktionen ohne Parameter und ohne Rückgabewert

- Funktionsaufruf:

Bezeichner ();

```
// Beispiel:  
  
#include <iostream>  
  
int main() {  
  
    zeichne_sterne();  
    zeichne_sterne();  
    zeichne_sterne();  
  
    return 0;  
  
}
```

#### **Achtung:**

Die Funktionsdefinition muss vor dem 1. Funktionsaufruf stehen!

#### **Alternativ:**

Die Funktionsdeklaration muss vor dem 1. Funktionsaufruf stehen. Dann kann die Funktionsdefinition später, also auch nach dem ersten Funktionsaufruf, erfolgen.



## (a) Funktionen ohne Parameter und ohne Rückgabewert

```
// Komplettes Beispiel: bsp1.exe
#include <iostream>

void zeichne_sterne() {
    int k = 10;
    while (k--) std::cout << '*\n';
    std::cout << std::endl;
}

int main() {
    zeichne_sterne();
    zeichne_sterne();
    zeichne_sterne();
    return 0;
}
```

Zuerst Funktionsdefinition.

Dann Funktionsaufrufe.

Ausgabe:

```
C:\WINDOWS\system32\cmd.exe
E:\EINI>bsp1
*****
*****
*****
E:\EINI>_
```



## (a) Funktionen ohne Parameter und ohne Rückgabewert

```
// Komplettes Beispiel: bsp1a.exe
#include <iostream>
void zeichne_sterne();
int main() {
    zeichne_sterne();
    zeichne_sterne();
    zeichne_sterne();
    return 0;
}
void zeichne_sterne() {
    int k = 10;
    while (k--) std::cout << '* ';
    std::cout << std::endl;
}
```

Zuerst Funktionsdeklaration.

Dann Funktionsaufrufe.

Später Funktionsdefinition.

Ausgabe:

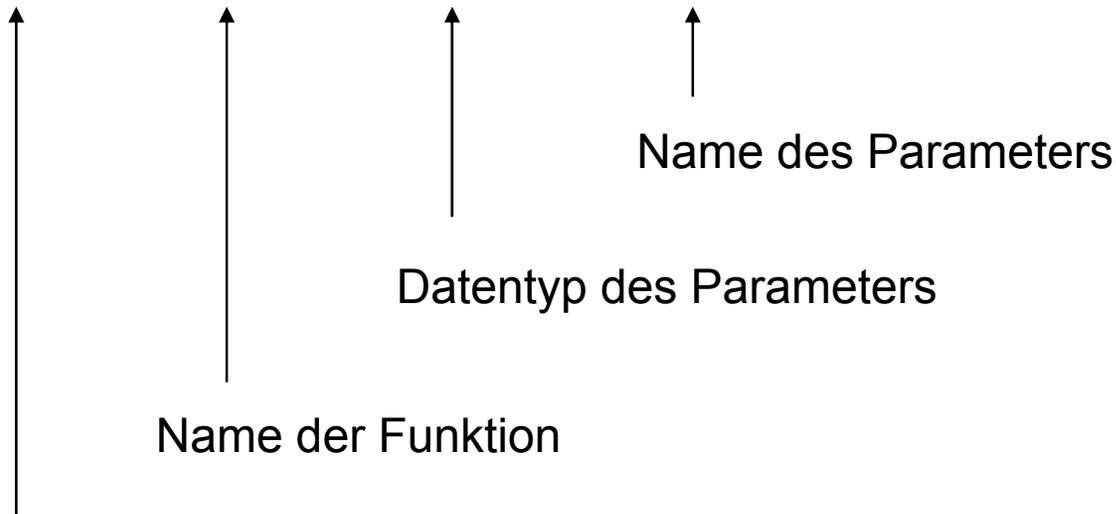
```
C:\WINDOWS\system32\cmd.exe
E:\EINI>bsp1a
*****
*****
*****
E:\EINI>
```



## (b) Funktionen mit Parameter aber ohne Rückgabewert

- Funktionsdeklaration:

`void` Bezeichner (Datentyp Bezeichner) ;



`void` (= leer) zeigt an, dass kein Wert zurückgegeben wird



### (b) Funktionen mit Parameter aber ohne Rückgabewert

- Funktionsdefinition:

```
void Bezeichner (Datentyp Bezeichner) {  
  
    // Anweisungen  
  
}
```

```
// Beispiel:  
void zeichne_sterne(int k) {  
    while (k-->0) std::cout << '*';  
    std::cout << std::endl;  
}
```



### (b) Funktionen mit Parameter aber ohne Rückgabewert

- Funktionsaufruf:

Bezeichner (Parameter) ;

```
// Beispiel:  
#include <iostream>  
  
int main() {  
    zeichne_sterne(10);  
    zeichne_sterne( 2);  
    zeichne_sterne( 5);  
    return 0;  
}
```

#### **Achtung:**

Parameter muss dem Datentyp entsprechen, der in Funktionsdeklaration bzw. Funktionsdefinition angegeben ist.

Hier: `int`

Kann Konstante oder Variable sein.



### (b) Funktionen mit Parameter aber ohne Rückgabewert

```
// Komplettes Beispiel: bsp2.exe
#include <iostream>

void zeichne_sterne(int k) {

    while (k--) std::cout << '*\n';
    std::cout << std::endl;
}

int main() {

    zeichne_sterne(10);
    zeichne_sterne(2);
    zeichne_sterne(7);

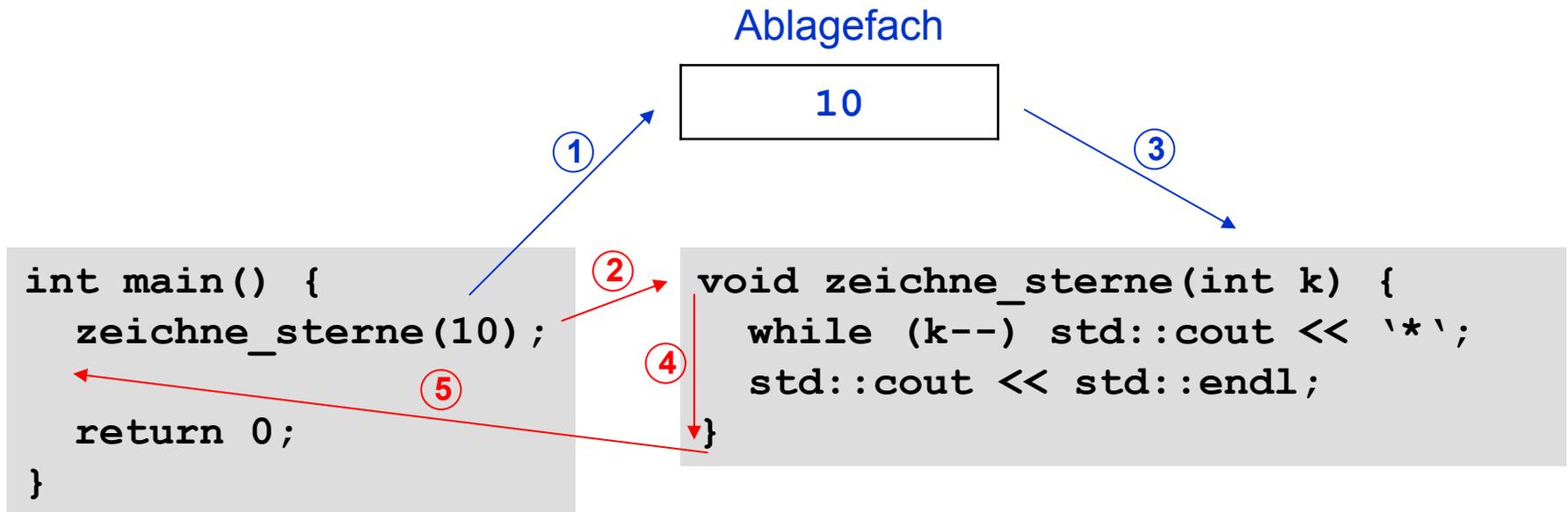
    return 0;
}
```

Ausgabe:

```
C:\WINDOWS\system32\cmd.exe
E:\EINI>bsp2
*****
**
*****
E:\EINI>
```



## Wie wird die Parameterübergabe technisch realisiert?



1. bei Aufruf `zeichne_sterne(10)` wird Parameter 10 ins Ablagefach gelegt
2. der Rechner springt an die Stelle, wo Funktionsanweisungen anfangen
3. der Wert 10 wird aus dem Ablagefach geholt und `k` zugewiesen
4. die Funktionsanweisungen werden ausgeführt
5. nach Beendigung der Funktionsanweisungen Rücksprung hinter Aufruf



### (b) Funktionen mit Parameter aber ohne Rückgabewert

```
// Komplettes Beispiel: bsp2a.exe
#include <iostream>

void zeichne_sterne(int k) {
    while (k-->0) std::cout << '* ';
    std::cout << std::endl;
}

int main() {
    int i;

    for (i = 10; i > 0; i--)
        zeichne_sterne(i);

    return 0;
}
```

Ausgabe:

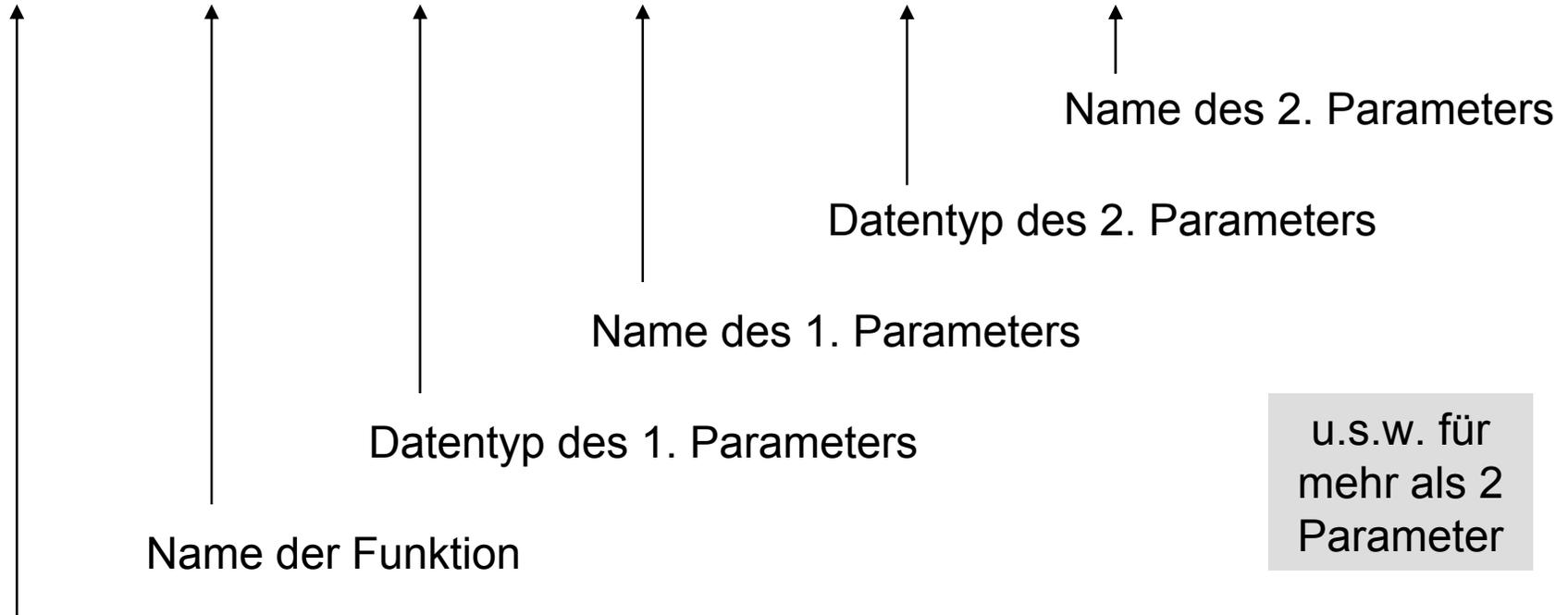
```
C:\WINDOWS\system32\cmd.exe
E:\EINI>bsp2a
*****
*****
*****
*****
*****
*****
*****
****
***
**
*
E:\EINI>_
```



## (b) Funktionen mit Parametern aber ohne Rückgabewert

- Funktionsdeklaration:

```
void Bezeichner (Datentyp1 Bezeichner1, Datentyp2 Bezeichner2) ;
```



u.s.w. für  
mehr als 2  
Parameter

`void` (= leer) zeigt an, dass kein Wert zurückgegeben wird



### (b) Funktionen mit Parametern aber ohne Rückgabewert

- Funktionsdefinition:

```
void Bezeichner (Datentyp1 Bezeichner1, Datentyp2 Bezeichner2) {  
  
    // Anweisungen  
  
}
```

```
// Beispiel:  
void zeichne_zeichen(int k, char c) {  
    // zeichne k Zeichen der Sorte c  
    while (k-->0) std::cout << c;  
    std::cout << std::endl;  
}
```



### (b) Funktionen mit Parametern aber ohne Rückgabewert

- Funktionsaufruf:

Bezeichner (Parameter1, Parameter2) ;

```
// Beispiel:  
#include <iostream>  
  
int main() {  
  
    zeichne_zeichen(10, '*' );  
    zeichne_zeichen( 2, 'A' );  
    zeichne_zeichen( 5, '0' );  
  
    return 0;  
  
}
```

**Natürlich:**

Bei mehr als 2 Parametern wird die Parameterliste länger!



### (b) Funktionen mit Parametern aber ohne Rückgabewert

```
// Komplettes Beispiel: Bsp2b.exe
#include <iostream>
void zeichne_zeichen(int k, char c)
{
    // zeichne k Zeichen der Sorte c
    while (k--) std::cout << c;
    std::cout << std::endl;
}

int main() {
    zeichne_zeichen(10, '*' );
    zeichne_zeichen( 2, 'A' );
    zeichne_zeichen( 5, '0' );

    return 0;
}
```

Ausgabe:

```
C:\WINDOWS\system32\cmd.exe
E:\EINI>bsp2b
*****
AA
00000
E:\EINI>
```



## (b) Funktionen mit Parametern aber ohne Rückgabewert

```
// Komplettes Beispiel: Bsp2c.exe
#include <iostream>
void zeichne_zeichen(int k, char c)
{
    // zeichne k Zeichen der Sorte c
    while (k--) std::cout << c;
    std::cout << std::endl;
}

int main() {
    int i;
    for (i = 0; i < 26; i++)
        zeichne_zeichen(i + 1, 'A' + i);
    return 0;
}
```

Ausgabe:

```
C:\WINDOWS\system32\cmd.exe
E:\EINI>bsp2c
A
BB
CCC
DDDD
EEEE
FFFFFF
GGGGGG
HHHHHHH
IIIIIIII
JJJJJJJJ
KKKKKKKKKK
LLLLLLLLLLLL
MMMMMMMMMMMM
NNNNNNNNNNNN
OOOOOOOOOOOO
PPPPPPPPPPPP
QQQQQQQQQQQQ
RRRRRRRRRRRR
SSSSSSSSSSSS
TTTTTTTTTTTT
UUUUUUUUUUUU
VVVVVVVVVVVV
WWWWWWWWWWWW
XXXXXXXXXXXX
YYYYYYYYYYYY
ZZZZZZZZZZZZ
E:\EINI>
```



## (c) Funktionen ohne Parameter aber mit Rückgabewert

- Funktionsdeklaration:

Datentyp Bezeichner ( ) ;



Nichts zwischen Klammern  $\Rightarrow$  keine Parameter

Name der Funktion

Datentyp des Wertes, der zurückgegeben wird



### (c) Funktionen ohne Parameter aber mit Rückgabewert

- Funktionsdefinition:

```
Datentyp Bezeichner () {  
    // Anweisungen  
    return Rückgabewert;  
}
```

#### **Achtung!**

Datentyp des Rückgabewertes muss mit dem in der Funktionsdefinition angegebenen Datentyp übereinstimmen.

```
// Beispiel:  
bool Fortsetzen() {  
    char c;  
    do {  
        std::cout << "Fortsetzen (j/n)? ";  
        std::cin >> c;  
    } while (c != 'j' && c != 'n');  
    if (c == 'j') return true;  
    return false;  
}
```



### (c) Funktionen ohne Parameter aber mit Rückgabewert

- Funktionsaufruf:

Variable = Bezeichner ();

**oder:**

Rückgabewert ohne  
Speicherung verwenden

```
// Beispiel:  
  
#include <iostream>  
  
int main() {  
    int i = 0;  
    do {  
        zeichne_zeichen(i + 1, 'A' + i);  
        i = (i + 1) % 5;  
    } while (fortsetzen());  
    return 0;  
}
```



## (c) Funktionen ohne Parameter aber mit Rückgabewert

```
// Komplettes Beispiel: bsp3.exe
#include <iostream>

void zeichne_zeichen(int k, char c) {
    while (k--) std::cout << c;
    std::cout << std::endl;
}

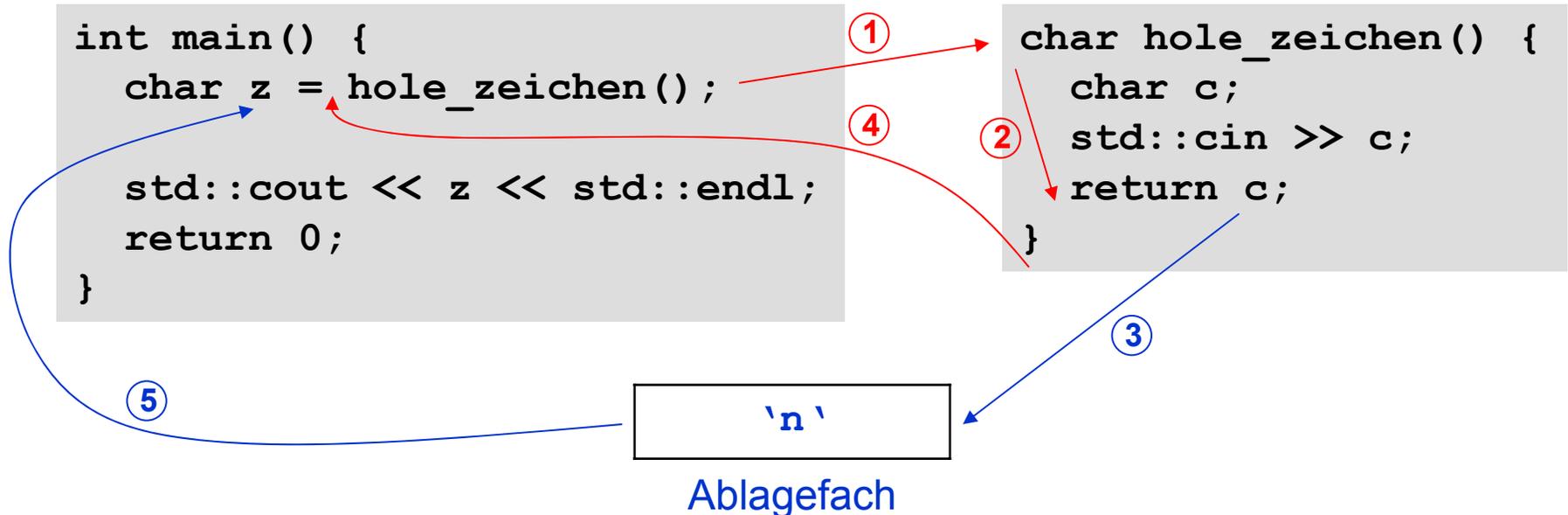
bool fortsetzen() {
    char c;
    do {
        std::cout << "Fortsetzen (j/n)? ";
        std::cin >> c;
    } while (c != 'j' && c != 'n');
    if (c == 'j') return true;
    return false;
}

int main() {
    int i = 0;
    do {
        zeichne_zeichen(i + 1, 'A' + i);
        i = (i + 1) % 5;
    } while (fortsetzen());
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
E:\EINI>bsp3
A
Fortsetzen (j/n)? j
BB
Fortsetzen (j/n)? j
CCC
Fortsetzen (j/n)? j
DDDD
Fortsetzen (j/n)? j
EEEEE
Fortsetzen (j/n)? j
A
Fortsetzen (j/n)? j
BB
Fortsetzen (j/n)? J
Fortsetzen (j/n)? x
Fortsetzen (j/n)? p
Fortsetzen (j/n)? N
Fortsetzen (j/n)? j
CCC
Fortsetzen (j/n)? j
DDDD
Fortsetzen (j/n)? n
E:\EINI>
```



## Wie wird die Funktionswertrückgabe realisiert?



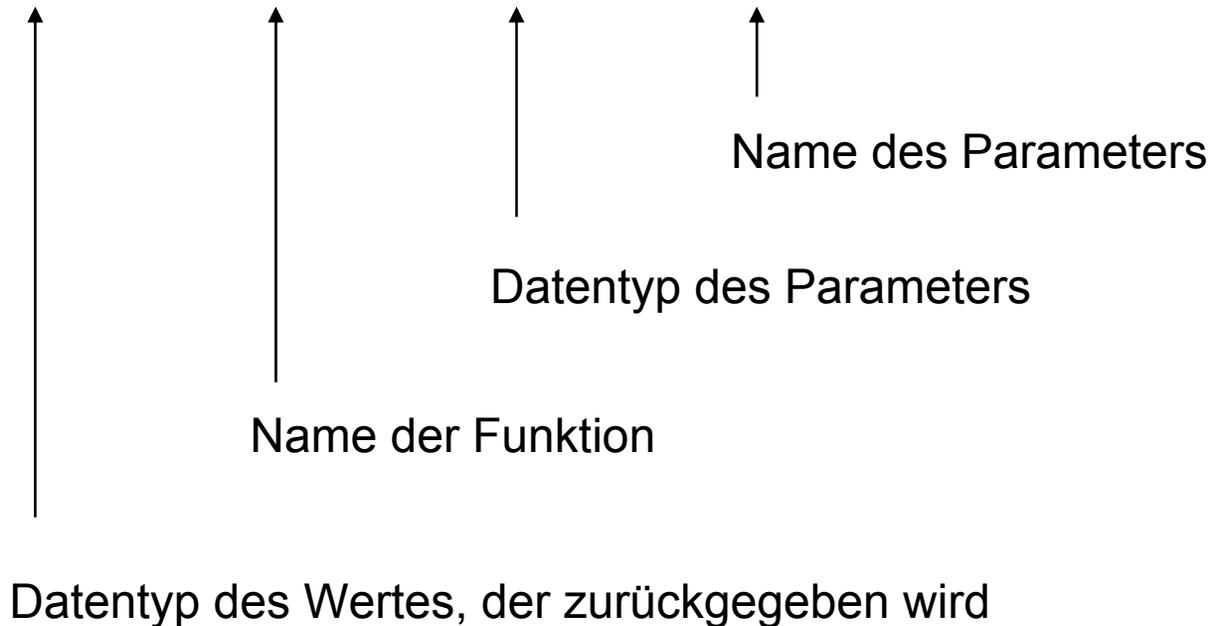
1. Rechner springt bei Aufruf `hole_zeichen()` zu den Funktionsanweisungen
2. Die Funktionsanweisungen werden ausgeführt
3. Bei `return c` wird der aktuelle Wert von `c` ins Ablagefach gelegt
4. Rücksprung zur aufrufenden Stelle
5. Der zuzuweisende Wert wird aus dem Ablagefach geholt und zugewiesen



## (d) Funktionen mit Parameter und mit Rückgabewert

- Funktionsdeklaration:

Datentyp Bezeichner (Datentyp Bezeichner) ;





### (d) Funktionen mit Parameter und mit Rückgabewert

- Funktionsdefinition:

```
Datentyp Bezeichner (Datentyp Bezeichner) {  
    // Anweisungen  
    return Rückgabewert;  
}
```

```
// Beispiel:  
double polynom(double x) {  
    return 3 * x * x * x - 2 * x * x + x - 1;  
}
```

Offensichtlich wird hier für einen Eingabewert  $x$  das Polynom

$$p(x) = 3x^3 - 2x^2 + x - 1$$

berechnet und dessen Wert per **return** zurückgeliefert.



### (d) Funktionen mit Parameter und mit Rückgabewert

- Funktionsaufruf:

Variable = Bezeichner (Parameter) ;

*oder:* Rückgabewert ohne  
Speicherung verwenden

```
// Beispiel:  
  
#include <iostream>  
#using namespace std;  
  
int main() {  
    double x;  
    for (x = -1.0; x <= 1.0; x += 0.1)  
        cout << "p(" << x << ") = "  
            << polynom(x) << endl;  
    return 0;  
}
```



## (d) Funktionen mit Parameter und mit Rückgabewert

```
// Komplettes Beispiel: Bsp4.exe
#include <iostream>
using namespace std;

double polynom(double x) {
    return 3 * x * x * x -
           2 * x * x + x - 1;
}

int main() {
    double x;
    for (x = -1.0; x <= 1.0; x += 0.1)
        cout << "p(" << x << ") = "
              << polynom(x) << endl;
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
E:\EINI>bsp4
p(-1)= -7
p(-0.9)= -5.707
p(-0.8)= -4.616
p(-0.7)= -3.709
p(-0.6)= -2.968
p(-0.5)= -2.375
p(-0.4)= -1.912
p(-0.3)= -1.561
p(-0.2)= -1.304
p(-0.1)= -1.123
p(-1.38778e-016)= -1
p(0.1)= -0.917
p(0.2)= -0.856
p(0.3)= -0.799
p(0.4)= -0.728
p(0.5)= -0.625
p(0.6)= -0.472
p(0.7)= -0.251
p(0.8)= 0.056
p(0.9)= 0.467
p(1)= 1
E:\EINI>
```