CARL VON OSSIETZKY universität OLDENBURG

Teil

Imperative Programmierung

Unterrichtseinheit 8

Boolesche Funktionen (Hamster-Modell)

Dr. Dietrich Boles

Gliederung

- Motivation
- return-Anweisung
- Boolesche Funktionen
  - Definition
  - Aufruf
  - Beispiele
- Seiteneffekte
- Codekonventionen
- Zusammenfassung

Motivation

Prozeduren: Definition neuer Befehle

Boolesche Funktionen: Definition neuer Testbefehle;

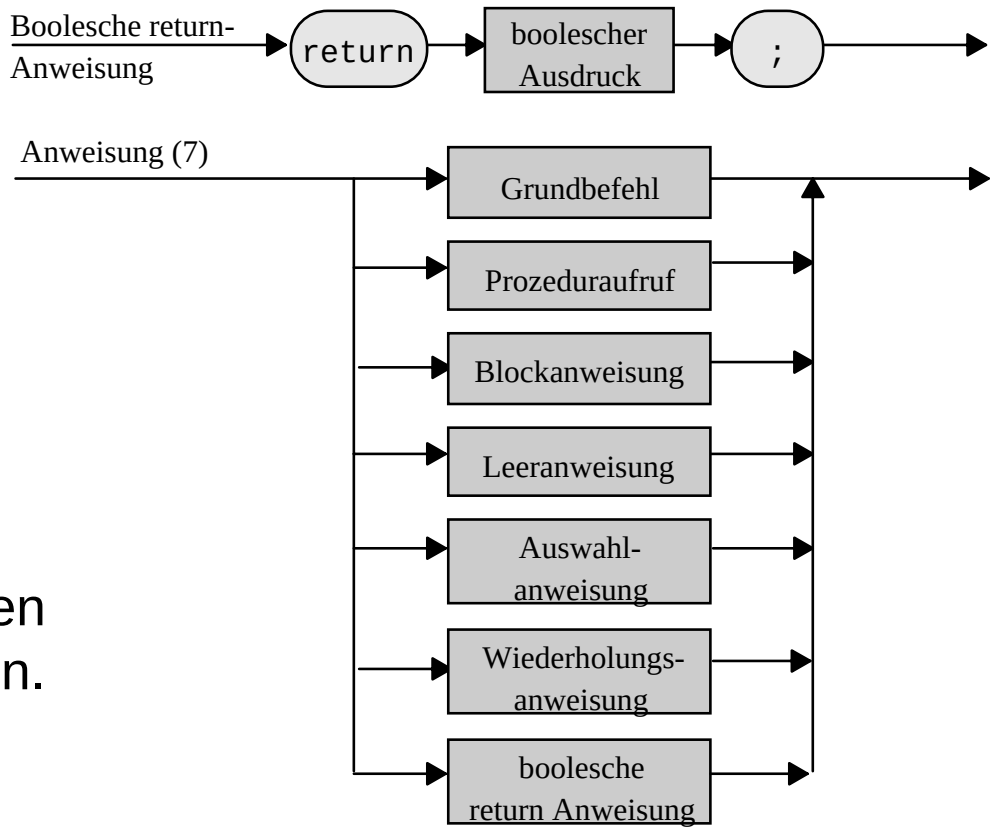
liefern einen Wahrheitswert

```
mauerDa()
rechtsFrei()
fuenfKoernerDa()
linksFrei()
...
```

```
if (linksFrei()) {
    linksUm();
    vor();
}
```

return-Anweisung

Syntax:

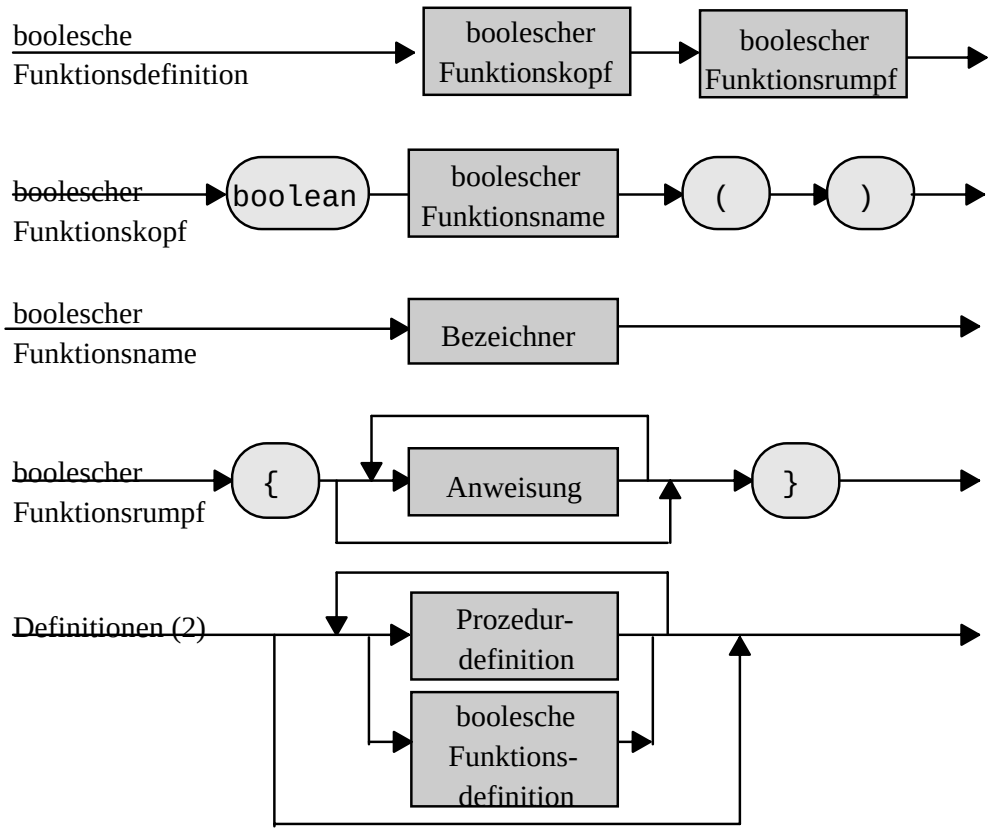


Zusatzbedingung:
Darf nur in einer booleschen Funktion verwendet werden.

Semantik: Der Ausdruck wird ausgewertet und die Funktion unmittelbar verlassen. Geliefert wird der Wert des Ausdrucks.

Boolesche Funktionen / Definition

Syntax:

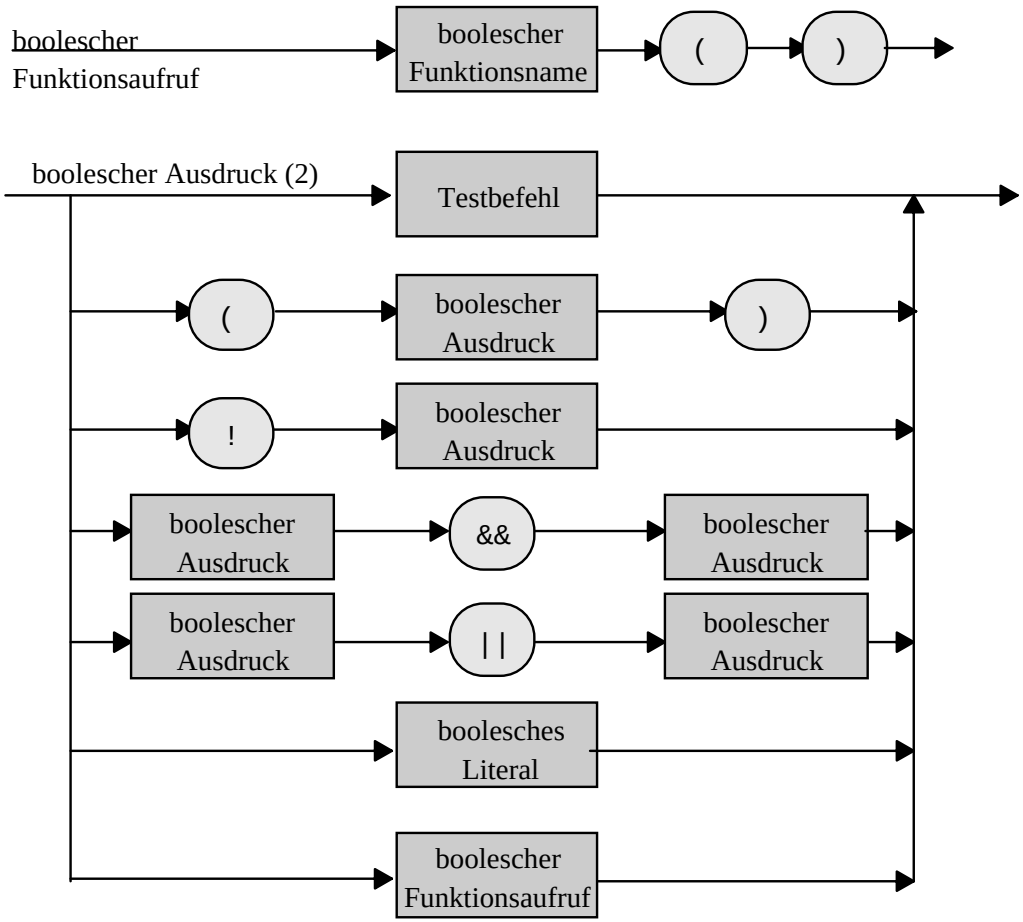


Zusatzbedingung: In jedem möglichen Weg durch den Rumpf muss eine boolesche return-Anweisung stehen!

Semantik: Es wird eine neue Funktion eingeführt.

Boolesche Funktionen / Aufruf

Syntax:



Semantik:

Der Funktionsrumpf wird ausgeführt. Der Wert ergibt sich aus dem von der Funktion gelieferten Wert.

Boolesche Funktionen / Beispiel 1

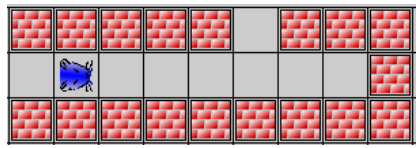
Aufgabe:

Links Nische suchen und hineinbegeben.

```
boolean mauerDa() {
    return !vornFrei();
}

boolean linksFrei() {
    linksUm();
    if (vornFrei()) {
        rechtsUm();
        return true;
    } else {
        rechtsUm();
        return false;
    }
}
```

Typische Landschaft:



```
void main() {
    while (!mauerDa() &&
           !linksFrei()) {
        vor();
    }
    if (linksFrei()) {
        linksUm();
        vor();
    }
}

void rechtsUm() {
    linksUm();
    linksUm();
    linksUm();
}
```

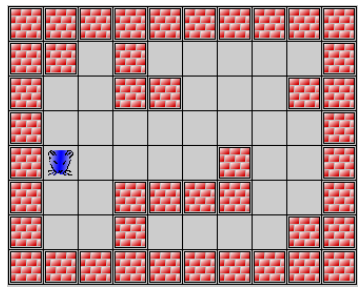
Demo

Boolesche Funktionen / Beispiel 2

Aufgabe:

Der Hamster (mit mindestens einem Korn im Maul) soll entlang der Wand laufen, bis er zur Ausgangsposition zurückgekehrt ist.

Typische Landschaft:



Demo

Programm:

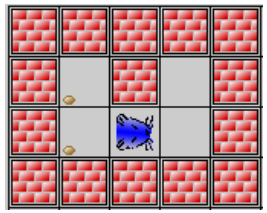
```
void main() {
  gib();  // Markierung
  vor();
  while (!kornDa()) {
    if (rechtsFrei()) {
      rechtsUm(); vor();
    } else if (vornFrei()) {
      vor();
    } else if (linksFrei()) {
      linksUm(); vor();
    } else {
      kehrt(); vor();
    }
  }
}
```

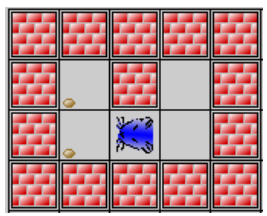
```
boolean linksFrei() {
  linksUm();
  if (vornFrei()) {
    rechtsUm(); return true;
  } else {
    rechtsUm(); return false;
} }
boolean rechtsFrei() {
  rechtsUm();
  if (vornFrei()) {
    linksUm(); return true;
  } else {
    linksUm(); return false;
} }
```

Seiteneffekte

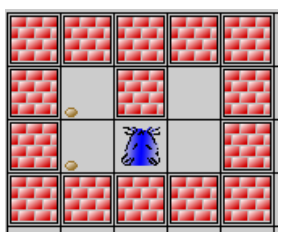
```
boolean linksFrei() {
    linksUm();
    return vornFrei();
}

void main() {
    if (linksFrei())
        ;
    if (!linksFrei())
        ;
}
```





```
void main() {
    if (linksFrei())
        ;
    else
        ;
}
```



Codekonventionen

- Funktionsname: Anfangsbuchstabe klein; Anfangsbuchstaben neuer Wortbestandteile groß
- Funktionsname: aussagekräftig !!!
- Funktionskopf und `{` in eine Zeile
- `}` unterhalb des b von `boolean`
- innere Anweisungen um 4 Spalten einrücken
- Hinter Funktionsnamen kein Leerzeichen
- Funktionsdefinition und -aufruf: Zwischen `( )` kein Leerzeichen
- Funktionsdefinition: Hinter `( )` ein Leerzeichen
- mehrere Prozedur- und Funktionsdefinitionen jeweils durch Leerzeile trennen

Zusammenfassung

- Mit Hilfe von booleschen Funktionen können neue Testbefehle definiert werden
- Bei der Funktionsdefinition wird der neue Testbefehl vereinbart
- Beim Funktionsaufruf wird der neue Testbefehl ausgeführt und liefert einen Wahrheitswert