Programmierung 1

Logik & Bedingungen

Monika Schak

Woche 2

01. November 2023

Hochschule Fulda
University of Applied Sciences

Wiederholung

- **Variablen** dienen dazu, Daten zu verwalten
  - Variablen haben einen **Bezeichner**, eine **Speicheradresse** und einen **Datentyp**
  - Variablenwerte können durch **Zuweisungen** verändert werden
- Aufbau von **Zuweisungen**: `<variable> = <ausdruck>;`
  - Wert des Ausdrucks wird mittels **aktuellem Variablenzustand** ermittelt
  - Variablen auf der linken Seite der Zuweisung wird der ermittelte Wert zugeordnet
  - Die Zuweisung (mit Zuweisungsoperator =) ist selbst ein **Ausdruck**
  - Ein mit einem Semikolon abgeschlossener Ausdruck ist eine **Anweisung**
- Ein **Programm** ist eine geordnete Folge von Anweisungen
  - Es besteht damit aus einer **Anweisungssequenz**
  - Anweisungsblöcke sind durch **geschweifte Klammern** zusammengefasst

Hochschule Fulda
University of Applied Sciences

Boolesche Variablen

- Datentyp: `bool`
  Mit C99-Standard als `_Bool` eingeführt
  Bindet man `stdbool.h` ein, kann man den Typ `bool` verwenden
- Variablen vom Typ `bool` können einen von zwei Werten annehmen:
  - `false` (in C wird 0 als false ausgewertet)
  - `true` (in C wird alles ungleich 0 als true ausgewertet)
- Beispiel: `bool human = true;`

Hochschule Fulda
University of Applied Sciences

Boolesche Ausdrücke

Boolesche Operatoren berechnen aus einem oder zwei booleschen Ausgangswerten (vom Typ `bool`) einen neuen booleschen Wert

- Logisches UND `&&` (nur *true*, wenn beide Argumente **true**)
- Logisches ODER `||` (nur **false**, wenn beide Argumente *false*)
- Logisches Nicht: `!` (ergibt den jeweils anderen Wahrheitswert)

a	b	a && b
false	*false*	*false*
false	*true*	*false*
true	*false*	*false*
true	*true*	*true*

a	b	a || b
false	*false*	*false*
false	*true*	*true*
true	*false*	*true*
true	*true*	*true*

a	!a
false	*true*
true	*false*

Hochschule Fulda
University of Applied Sciences

Übung

- Ordnen Sie die Begriffe dem folgenden Code zu: Datentyp, Variable, Wert, Deklaration, Zuweisung

```
bool a = true, b = false;
int x = 18, y = (x/4) % 3, z;
x = x * 2;
a = a && true;
b = (a && b) || false;
//printf("%d\n", x);
printf("%d\n", y);
```

- Was sind die Werte der Variablen a, b, x, y und z nach Ausführung des obigen Codes?
- Was wird am Ende ausgegeben?

Hochschule Fulda
University of Applied Sciences

Vergleiche

- Das Resultat von Vergleichen (z.B. zwischen Zahlen) ist ein boolescher Wert
  - == (Gleichheit), != (Ungleichheit): Vergleich von Zahlen, Zeichen und booleschen Werten auf Gleichheit bzw. Ungleichheit
  - > (Größer), >= (Größer gleich): Vergleich von Zahlen, welche größer ist
  - < (Kleiner), <= (Kleiner gleich): Vergleich von Zahlen, welche kleiner ist
- In der Bedingung von Fallunterscheidungen (sog. `if`-Anweisungen) steht ein boolescher Ausdruck, der einen Wahrheitswert ergibt
  - Fallunterscheidung dienen der Modellierung alternativer Programmabläufe

Hochschule Fulda
University of Applied Sciences

Bedingte Anweisungen

- Die Schlüsselwörter für bedingte Anweisungen sind `if` und `else`
- Erst wird der Bedingungsausdruck ausgewertet: `n > 0`
  $\rightarrow$ muss einen Wahrheitswert ergeben
- Falls das Ergebnis `true` ist, weiter mit erstem Block, sonst mit zweitem (else)
- Bei einer einseitigen Fallunterscheidung gibt es nur den ersten Block
- Geschweifte Klammern definieren Blöcke (sollten immer ordentlich eingerückt werden)
- In Blöcken können wieder Anweisungen oder Fallunterscheidungen stehen, es gibt aber auch leere Blöcke `{ }`

```
int n = 3;

if (n < 0) {
  printf("Die Zahl ist negativ.\n");
}
else {
  printf("Die Zahl ist positiv.\n");
}
```

Hochschule Fulda
University of Applied Sciences

Eingabe

```
int age;
printf("Bitte Alter eingeben: ");
scanf("%d", &age);
printf("Du wurdest %d geboren.\n", 2022-age);
```

- Die Funktion `scanf()` liest von der Standardeingabe (Tastatur)
  - `scanf()` ist ähnlich aufgebaut wie `printf()`
  - Formatzeichen `%d` spezifiziert, dass eine ganze Zahl eingelesen werden soll
  - Zeichen `&` ist ein Adressoperator
- In C muss beim Einlesen von Werten durch die Speicheradresse angegeben werden (mittels Adressoperator), wo der Wert im Speicher abgelegt werden soll

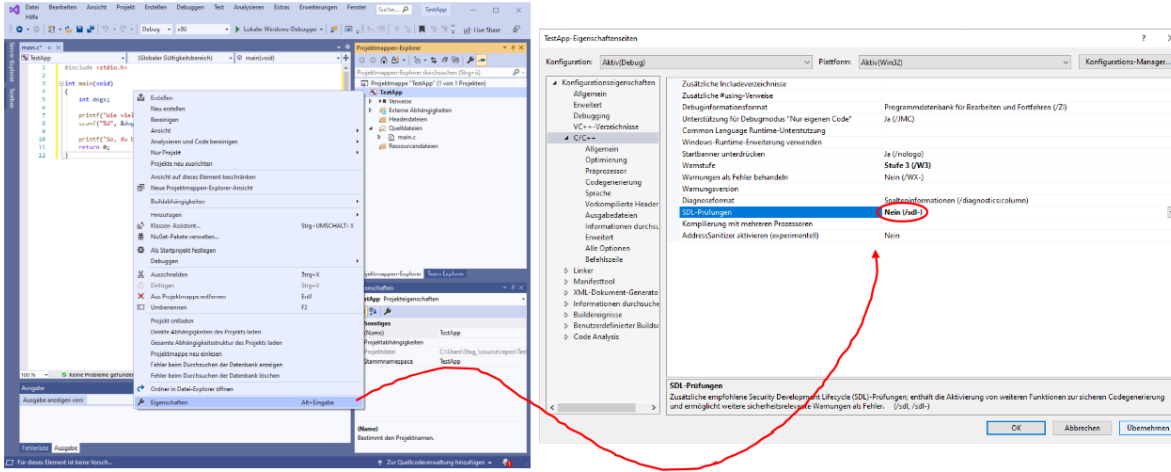
Hochschule Fulda
University of Applied Sciences

Probleme bei Eingabefunktionen

- Eingabefunktionen wie `scanf()` haben generell Sicherheitsproblem
  - Leider auf jeder Plattform, ist aber erst für Produktivcode relevant
  - `scanf()` prüft nicht auf Pufferüberlauf, was unbehandelt Hackern Tür und Tor öffnet, ist aber leichter zu nutzen als direkt die komplette Problembehandlung umzusetzen
- MS Visual Studio sieht solche Funktionen als veraltet (deprecated)
  - man muss diese Funktionen explizit erlaubt (s. nächste Folie)
  - Alternativ gibt es seit C11-Standard (wird leider noch nicht überall unterstützt) eine sichere Variante: `scanf_s()`

Hochschule Fulda
University of Applied Sciences

Probleme bei Eingabefunktionen



Ternärer Operator

- Beispiel: Bestimmung des Maximus (= größter Wert)

```
if (a > b) {
  x = a;
} else {
  x = b;
}
```

- Geht auch übersichtlicher als Einzeiler:

```
x = (a > b) ? a : b;
```

- Hier wird der größte Wert mit Hilfe des sog. **Ternären Operators** bestimmt (über die Bedingung a > b) und das Ergebnis wird x zugewiesen

Hochschule Fulda
University of Applied Sciences

Ternärer Operator

- Bedingungsoperator `?:` ist einziger ternärer Operator, d.h. Operator mit 3 Operanden, (z.B. + ist binärer Operator, ! ist unärer Operator)
- Syntax: `<Bedingungsausdruck> ? <Anweisung1> : <Anweisung2>`
  - Falls Bedingung wahr: Anweisung1 wird ausgewertet, sonst Anweisung 2
  - Es wird immer nur genau ein Asudruck ausgewertet
  - Kurzform der if/else-Anweisung (v.a. bei Zuweisungen)
- Gilt für jeden Datentyp, kann zugewiesen werden wie andere Ausdrücke auch, oder alleine stehen
- Beispiel:

```
printf("Zahl %d ist %sgerade\n", zahl, (zahl % 2 == 0) ? "" : "un");
```

Hochschule Fulda
University of Applied Sciences

Datentyp “char”

- F|ur einzelne Zeichen (Characters) gibt es den Datentyp `char`
- Dient zur Speicherung einzelner Buchstaben, Ziffern u. Sonderzeichen (inkl. Steuerzeichen = Escape-Sequenzen wie `\n`) aus ASCII-Zeichensatz
  ASCII-Tabelle:
  `https://www.c-howto.de/tutorial/anhang/ascii-tabelle/`
- Die Größe eines `char`s beträgt genau 1 Byte: Wertebereich von -128 bis +127 (bzw. von 0 bis 255 = Entspricht dann Eintrag in der ASCII-Tabelle)
- Angabe eines Zeichenwertes in einfachen Anführungszeichen
- Für Ein- und Ausgaben gibt es das Formatzeichen %c
- Beispiel:

```
char c0 = 'p', c1 = 'r', c2 = 'o', c3 = 'g', nl = '\n';
printf("%c%c%c%c%c", c0, c1, c2, c3, nl);
```

Hochschule Fulda
University of Applied Sciences

Komplexere if-Anweisungen

Beispiel: Ein einfacher Taschenrechner

```
char operator;
int wert1, wert2, erg;

printf("Term angeben (z.B. 1 + 1)!\n");
scanf("%d %c %d", &wert1, &operator, &wert2);

// TODO: Implementierung der Berechnung

if (operator == ' ') {
  printf("Fehler, falsche Eingabe\n");
} else {
  printf("%d %c %d = %d\n", wert1, operator, wert2, erg);
}
```

Hochschule Fulda
University of Applied Sciences

Mehrfachauswahl

```
switch (operator) {
  case '+': erg = wert1 + wert2; break;
  case '-': erg = wert1 - wert2; break;
  case '*': erg = wert1 * wert2; break;
  case '/': erg = wert1 / wert2; break;
  case '%': erg = wert1 % wert2; break;
  default: operator = ' ';
}
```

- Bedingungsauswahl im *Schalter* (`switch`) liefert Wert (vom Typ `int` oder `char`)
- Für jeden möglichen Wert ist ein Fall (`case`) vorgesehen
- Nicht berücksichtigte Fälle werden unter `default` behandelt
- Die Anweisung `break` bedeutet: wird wird abgebrochen und Ausführung nach dem `switch` fortgeführt
- Ohne `break` wird je nachfolgendes `case` weiter abgearbeitet

Hochschule Fulda
University of Applied Sciences

Zusammenfassung

- Zwei Typen von Fallunterscheidungen
  - ```
    if (Bedingung_erfuellt) { ... }
    else if (andere_Bedingung) { ... }
    else { ... ]
    ```
  - ```
    switch (Ausdruck) { // Typ char oder int
      case Const_Term_1: ... break;
      case Const_Term_2: ... break;
      case Const_Term_3: ... break;
      ...
      default: ...
    ```
- Bedingungsoperator (Ternärer Operator):
  - `<Bedingung> ? <Anweisung1> : <Anweisung2>`

Hochschule Fulda
University of Applied Sciences

Logische Aussagen

- Boolesche Operatoren: UND, ODER, NICHT
  - Je nach Fachrichtung gibt es verschiedene Schreibweisen:

		C	Mathe	DigiTech
Konjunktion	UND	`a && b`	$a \wedge b$	a * b
Disjunktion	ODER	`a || b`	$a \vee b$	a + b
Negation	NICHT	`!a`	$\neg a$	$\overline{a}$

- In C und in der Digitaltechnik wird **0** als *false* und **1** als *true* interpretiert
- Auswertungsreihenfolge:
  - UND bindet stärker als ODER
  - NICHT bindet stärker als UND sowie ODER
  - Vergleichsoperatoren binden schwächer als NICHT, aber stärker als UND/ODER
  - Im Zweifel, oder bei anderer Reihenfolge: Klammern setzen!

Hochschule Fulda
University of Applied Sciences

Regeln

Kommutativgesetze	$a \wedge b = b \wedge a$	$a \vee b = b \vee a$
Assoziativgesetze	$(a \wedge b) \wedge c = a \wedge (b \wedge c)$	$(a \vee b) \vee c = a \vee (b \vee c)$
Idempotenzgesetze	$a \wedge a = a$	$a \vee a = a$
Distributivgesetze	$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$	$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$
Neutralitätsgesetze	$a \wedge 1 = a$	$a \vee 0 = a$
Extremalgesetze	$a \wedge 0 = 0$	$a \vee 1 = 1$
Doppelnegationsgesetz	$\neg(\neg a) = a$	
De Morganschesetze	$\neg(a \wedge b) = \neg a \vee \neg b$	$\neg(a \vee b) = \neg a \wedge \neg b$
Komplementärgesetze	$a \wedge \neg a = 0$	$a \vee \neg a = 1$
Dualitätsgesetze	$\neg 0 = 1$	$\neg 1 = 0$
Absorptionsgesetze	$a \vee (a \wedge b) = a$	$a \wedge (a \vee b) = a$

Hochschule Fulda
University of Applied Sciences

Übung

- Gegeben ist eine main-Funktion, die drei Integer-Variablen `a`, `b`, `c` einliest, sowie eine Character-Variable `ok` (gültige Werte für `ok` sind `'y'` für yes und `'n'` für no)
- Das Programm soll *Condition true* ausgeben, wenn `a < b < c`, außer `ok` hat den Wert `'y'`, wobei dann die Bedingung `a < b` nicht unbedingt gelten muss. Andernfalls soll *Condition false* ausgegeben werden.

```
if ((________________) || (____________________) {
  printf("Condition true\n");
} else {
  printf("Condition false\n");
}
```

Hochschule Fulda
University of Applied Sciences

Übung

- Gegeben ist eine main-Funktion, die ein `int` namens `age` und ein `char` namens `stillGoingStrong` einliest. Gültige Werte für `stillGoingStrong` sind `'y'` bzw. `'n'` (wie vorhin). Bei anderen Werten soll nur eine Fehlermeldung ausgegeben werden.
- Das Programm wertet ansonsten aus, ob eine Party mit Gästen vom Alter `age` erfolgreich ist. In diesem Fall soll *Erfolg* ausgegeben werden, sonst *Pleite*.
- Eine Party ist erfolgreich, wenn das Alter zwischen 18 und 30 (einschließlich) liegt. Die Party ist außerdem erfolgreich, falls `stillGoingStrong` den Wert `'y'` hat (unabhängig vom Alter), denn in dem Fall haben auch noch Ältere Spaß daran.
- **Wie muss die Fallunterscheidung aussehen?**

Hochschule Fulda
University of Applied Sciences