Bedingte Anweisungen

>> **Bedingte Anweisungen** ermöglichen es Programmen, Entscheidungen zu treffen und sich je nach Wert von Variablen oder Benutzereingaben unterschiedlich zu verhalten. Man nennt diese auch **Verzweigungen** (Englisch: ***forks*** oder ***branches***).

>> Wir sehen uns die verschiedenen Varianten an, mit denen man in C solche bedingten Anweisungen implementieren kann.

Einige davon kommen Ihnen vielleicht aus Scratch bekannt vor.

if-Anweisung

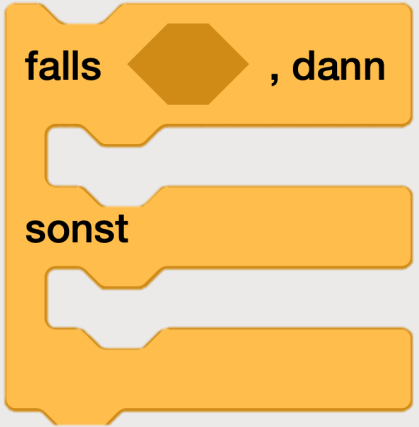
```
if (boolescher Ausdruck)
{
    // einige Anweisungen
}
```

Wenn der boolesche Ausdruck wahr ist, werden alle Codezeilen zwischen den geschweiften Klammern { ... } in der Reihenfolge von oben nach unten ausgeführt.

Wird der boolesche Ausdruck als falsch ausgewertet, werden diese Zeilen nicht ausgeführt.

if-else-Anweisung

```
if (boolescher Ausdruck)
{
    // einige Anweisungen
}
else
{
    // weitere Anweisungen
}
```



Ergibt der boolesche Ausdruck „wahr“, werden alle Codezeilen nach dem „if“ ausgeführt. Ist der boolesche Ausdruck „falsch“, werden alle Codezeilen nach dem „else“ ausgeführt.

else if

```
if (boolean-expr1)
{
    // erster Block
}
else if (boolean-expr2)
{
    // zweiter Block
}
else if (boolean-expr3)
{
    // dritter Block
}
else
{
    // vierter Block
}
```

Ermöglicht es, mehrere Bedingungen von oben nach unten zu überprüfen. Wird dabei eine Bedingung erreicht, die wahr ist, wird der dazugehörige Codeblock ausgeführt, alle anderen werden übersprungen.

Das „else“ am Ende ist optional, ist aber sehr empfehlenswert, um keine Fälle zu vergessen.

Mehrere if-Anweisungen

```
if (boolean-expr1)
{
    // erster Block
}
if (boolean-expr2)
{
    // zweiter Block
}
if (boolean-expr3)
{
    // dritter Block
}
else
{
    // vierter Block
}
```

Es ist auch möglich, eine Kette von Zweigen zu erstellen, die sich nicht gegenseitig ausschließen.

In diesem Beispiel schließen sich nur der dritte und vierte Zweig gegenseitig aus. Das „else“ gehört immer zum unmittelbar vorherigen „if“-Block.

switch-Anweisung

```
int x = get_int("");
switch(x)
{
case 1:
    printf("One!\n");
    break;
case 2:
    printf("Two!\n");
    break;
case 3:
    printf("Three!\n");
    break;
default:
    printf("Sorry!\n");
}
```

Die switch-Anweisung ermöglicht eine Aufzählung einzelner Fälle, anstatt boolesche Ausdrücke zu verwenden.

Es ist wichtig, zwischen den einzelnen Fällen ein „break“ zu setzen, sonst „fällt man durch“ jeden Fall (es sei denn, das ist das gewünschte Verhalten).

switch-Anweisung

```
int x = get_int("");
switch(x)
{
case 4:
    printf("Vier!\n");
case 3:
    printf("Drei!\n");
case 2:
    printf("Zwei!\n");
case 1:
    printf("Eins!\n");
default:
    printf("Fertig!\n");
}
```

Die switch-Anweisung ermöglicht eine Aufzählung einzelner Fälle, anstatt boolesche Ausdrücke zu verwenden.

Es ist wichtig, zwischen den einzelnen Fällen ein „break“ zu setzen, sonst „fällt man durch“ jeden Fall (es sei denn, das ist das gewünschte Verhalten).

Bedingter Operator

```
int x;
if (expr)
{
    x = 5;
}
else
{
    x = 6;
}
```

Syntax allgemein:

`(boolean-expr) ? value_if_true : value_if_false`

lässt sich mit dem *ternären bedingten Operator* (**? :**) abkürzen:

```
int x = (expr) ? 5 : 6;
```

Guter Stil

Ein guter Stil verbessert die Lesbarkeit und Wartbarkeit. Guter Stil bei bedingten Anweisungen:

- geschweifte Klammern auf eigenen Zeilen, richtig ausgerichtet;
- je ein Leerzeichen nach „if“, „else if“ und „else“;
- einheitliche Einrückung (4 Leerzeichen);
- ein Leerzeichen um Operatoren (>, ==, …);
- keine Leerzeichen unmittelbar nach öffnenden Klammern oder vor schließenden Klammern und
- öffnende geschweifte Klammer nicht in derselben Zeile wie Bedingung.

```
if (x < 0)
{
    printf("negativ\n");
}
else
{
    printf("mindestens 0\n");
}
```

EXTRAS IN 3 MINUTEN

FRAGEN – ANTWORTEN – RÄTSEL
UND KURZE ZUSAMMENFASSUNG

F1: Was sind die drei Arten von bedingten Anweisungen, die besprochen wurden?

F2: Was ist der Zweck des „default“-Falls in einer switch-Anweisung?

F3: Schreiben Sie die folgende Anweisung so um, dass der ternäre Vergleichsoperator verwendet wird:

```
if (x > 10) { y = 20; } else { y = 5; }
```

```c
int benutzerAlter = 17;
int postAlter = 48;  // in Stunden
int istPrivat = 0;   // 0 für nein, 1 für ja
int enthaeltHashtag = 1;  // 0 für nein, 1 für ja
int anzahlInteraktionen = 75;


if (!istPrivat && (benutzerAlter >= 18 || postAlter < 24))
{
    if (enthaeltHashtag && anzahlInteraktionen > 100)
    {
        printf("Trending");
    }
    else if (anzahlInteraktionen > 50 || (enthaeltHashtag && postAlter < 12))
    {
        printf("Erhöhte Sichtbarkeit");
    }
    else
    {
        printf("Normale Sichtbarkeit");
    }
}
else
{
    printf("Eingeschränkte Sichtbarkeit");
}
```

```c
int currentPlayer = 1;     // Spielernummer: 1 oder 2
int consecutiveSixes = 0;  // vorheriger Wurf war keine 6
int roll = 4;              // aktueller Würfelwurf

printf("Player %d rolled a %d.\n", currentPlayer, roll);

if (roll == 6)
{
    consecutiveSixes++;

    if (consecutiveSixes >= 3)
    {
        printf("Player %d wins!\n", currentPlayer);
    }
}
else
{
    consecutiveSixes = 0; // Zähler zurücksetzen
    currentPlayer = (currentPlayer == 1) ? 2 : 1;
    printf("It's now Player %d's turn.\n", currentPlayer);
}
```

```c
int currentPlayer = 1, consecutiveSixes = 0;
int roll = 3;
int sumPlayer1 = 15, sumPlayer2 = 12;

printf("Player %d rolled a %d.\n", currentPlayer, roll);
if (roll == 6) {
    if (++consecutiveSixes >= 3) {
        printf("Player %d rolled three sixes in a row!\n", currentPlayer);
        if (sumPlayer1 > sumPlayer2)
            printf("Player 1 wins!\n");
        else
            printf("Player 2 wins!\n");
    }
} else {
    consecutiveSixes = 0;
    if (currentPlayer == 1)
        sumPlayer1 += roll;
    else
        sumPlayer2 += roll;
    currentPlayer = (1 - (currentPlayer - 1)) + 1;
    printf("It's now Player %d's turn.\n", currentPlayer);
}
```

Bedingte Anweisungen ermöglichen Entscheidungen in Programmen basierend auf Variablenwerten oder Benutzereingaben.

if-Anweisungen führen Code aus, wenn eine Bedingung wahr ist; *else* fügt einen alternativen Codeblock hinzu.

else if ermöglicht mehrere Bedingungen in einer Kette; die Reihenfolge ist wichtig.

switch-Anweisungen erlauben die Aufzählung diskreter Fälle statt boolescher Ausdrücke.

Der ternäre Operator (?:) ist eine Kurzform für einfache *if-else* Zuweisungen.

Guter Programmierstil (Einrückung, Leerzeichen, Klammernsetzung) verbessert Lesbarkeit und Wartbarkeit des Codes.