Grundlagen der Programmierung in C# – Selektion

In C# gibt es zwei verschiedene Möglichkeiten, die Selektion bzw. bedingte Anweisungen realisieren:

Ein- und zweiseitige Alternative

```
if (...)
{
  // Anweisungsblock für die erfüllte Bedignung
}
else
{
  // Anweisungsblock für die nicht erfüllte Bedingung
}
```

- Nach IF folgt in runden Klammern die zu überprüfende Bedingung.
- Nach der schließenden runden Klammer der Bedingung folgt der Anweisungsblock, der ausgeführt wird, wenn die Bedingung wahr ist.
- Nach dem Schlüsselwort ELSE kann ein Anweisungsblock folgen, der ausgeführt wird, wenn die Bedingung nicht erfüllt wurde. Dieser darf enfallen.

Bsp.:

```
Console.Write("Gib eine ganze Zahl ein: ");
string eingabe = Console.ReadLine();
int zahl = Convert.ToInt32(eingabe);

if (zahl > 4)
{
  Console.WriteLine("Die Zahl ist größer als 4.");
}
else
{
  Console.WriteLine("Die Zahl ist nicht größer als 4.");
}
```

Für die Bedingung können die üblichen Vergleichsoperatoren <, >, <=, >= und == (für Gleichheit) verwendet werden. Weiterhin gibt es auch ein *ungleich*: !=.
Es können auch mehrere Bedingungen mit && (und) bzw. || (oder) verknüpft werden:

```
Console.Write("Gib eine ganze Zahl ein: ");
string eingabe = Console.ReadLine();
int zahl = Convert.ToInt32(eingabe);

if (zahl > 4 && zahl < 10) // die Zahl soll größer als 4 und kleiner als 10 sein
{
  Console.WriteLine("Die Zahl liegt zwischen 4 und 10.");
}
else
{
  Console.WriteLine("Die Zahl liegt nicht zwischen 4 und 10.");
}
```

```
Console.Write("Gib eine Zahl ein, die kleiner als 4 oder größer als 10 ist: ");
string eingabe = Console.ReadLine();
double zahl = Convert.ToInt32(eingabe);

```

```
5  if (zahl < 4 || zahl > 10)
6  {
7  Console.WriteLine("Deine eingegebene Zahl ist ok.");
8  }
9  else
10 {
11 Console.WriteLine("Deine Zahl entspricht nicht den Anforderungen.");
12 }
```

Achtung Besteht der Anweisungsblock nur aus einer einzigen Anweisung, dann können die geschweiften Klammern dieses Blockes weggelassen werden.

```
1  Console.Write("Gib eine Zahl ein, die kleiner als 4 oder größer als 10 ist: ");
2  string eingabe = Console.ReadLine();
3  double zahl = Convert.ToInt32(eingabe);
4
5  if (zahl < 4 || zahl > 10)
6    Console.WriteLine("Deine eingegebene Zahl ist ok.");
7  else Console.WriteLine("Deine Zahl entspricht nicht den Anforderungen.");
```

Natürlich können Alternativen auch verschachtelt werden:

```
1  if (Bedingung1)
2  {
3    if (Bedigung2)
4    {
5      // Bedingung1 und Bedingung2 sind erfüllt
6    }
7    else
8    {
9      // Bedingung1 ist erfüllt, Bedingung2 aber nicht
10   }
11 }
12 else
13 {
14   // Bedingung 1 ist nicht erfüllt
15 }
```

Achtung Nutze die Einrückungsmöglichkeiten des Visual Studios, um deinen Code übersichtlich zu gestalten. Auch das Zusammenfassen mittels geschweifter Klammern zu Codeblöcken erhöht die Lesbarkeit deutlich.

Aufgabe 1. Schreibe ein Programm, dass eine reelle Zahl einliest und ihre entgegensetzte Zahl ausgibt. D. h. bei Eingabe von z. B. $-3{,}45$ wird 3,45 ausgegeben.

Aufgabe 2. Schreibe ein Programm, dass eine Zahl einliest und ihr Reziprokes (Kehrwert) ausgibt.

Aufgabe 3. Schreibe ein Programm, das bei Eingabe von p und q für eine quadratische Gleichung $x^2 + px + q = 0$ entscheidet, ob sie zwei, eine oder keine Lösung besitzt.

Mehrseitige Auswahl – die Fallunterscheidung

Gibt es mehr Möglichkeiten als nur ja oder nein, so muss eine Fallunterscheidung herhalten:

```
Console.Write("Gib eine ganze Zahl ein: ");
string eingabe = Console.ReadLine();
int zahl = Convert.ToInt32(eingabe);

switch (zahl)
{
  case 1:
    Console.WriteLine("Cool!");
    break;
  case 2:
    Console.WriteLine("Gerade noch cool!");
    break;
  default:
    Console.WriteLine("Uncool!");
    break;
}
```

(a) Die Fallunterscheidung beginnt immer mit dem Schlüsselwort SWITCH gefolgt von runden Klammern, in denen die Variable steht, anhand derer die Fälle unterschieden werden sollen.

(b) Im Anweisungsblock der Fallunterscheidung werden die einzelnen Fälle mit dem Schlüsselwort CASE eingeleitet; danach folgt der entsprechende Wert der Variable. Abgeschlossen wird jeder CASE-Block mit der Anweisung BREAK; (CASE-Blöcke müssen nicht mit geschweiften Klammern zusammengefasst werden.)

(c) Am Ende kann ein DEFAULT-Zweig folgen, der für alle nicht angeführten Fälle ausgeführt wird. Auch dieser wird mit einem BREAK; beendet.

(d) Es können auch mehrere Fälle zusammengefasst werden:

```
switch (zahl)
{
case 1:
  Console.WriteLine("Cool!");
  break;
case 2:
case 3:
  Console.WriteLine("Gerade noch cool!");
  break;
default:
  Console.WriteLine("Uncool!");
break;
}
```

In diesem Beispiel wird sowohl bei Eingabe einer 2 als auch bei einer 3 die gleiche Ausgabe („Gerade noch cool!“) getätigt.

Achtung Nicht jeder Datentyp ist in einer SWITCH-Bedingung erlaubt. Zugelassene Datentypen sind: CHAR, INT, BOOLEAN oder STRING. Werden jedoch CHAR oder STRING verwendet, muss der Fall in Anführungsstrichen stehen.

```
Console.Write("Gib den Buchstaben der richtigen Antwort ein: ");
string antwort = Console.ReadLine();

switch (antwort)
{
```

```
6     case "a":
7       Console.WriteLine("Richtige Antwort!");
8       break;
9     case "b":
10    case "c":
11    case "d":
12      Console.WriteLine("Falsche Antwort!");
13      break;
14  }
```

Aufgabe 4. Schreibe ein Programm, das dem Benutzer eine Frage mit fünf verschiedenen Antworten zur Auswahl stellt. Der Benutzer soll die Frage beantworten. Je nach Antwort erhält der Nutzer ein Feedback über die Richtigkeit seiner Auswahl.