**Programmier-Grundkonzepte in C# und .NET (2)**

**Switch-Anweisung**

**Klassische switch-Anweisung**

Die klassische switch-Anweisung prüft verschiedene Werte einer Variable und führt den entsprechenden Code aus.

**Beispiel: Wochentag bestimmen**

using System;

class Program

{

static void Main()

{

int tag = 3;

switch (tag)

{

case 1:

Console.WriteLine("Montag");

break;

case 2:

Console.WriteLine("Dienstag");

break;

case 3:

Console.WriteLine("Mittwoch");

break;

case 4:

Console.WriteLine("Donnerstag");

break;

case 5:

Console.WriteLine("Freitag");

break;

case 6:

case 7:

Console.WriteLine("Wochenende!");

break;

default:

Console.WriteLine("Ungültige Eingabe.");

break;

}

}

}

**switch mit enum-Werten**

Ein enum ist oft praktischer als numerische Werte.

**Beispiel: Status eines Auftrags**

enum Status { Offen, InBearbeitung, Abgeschlossen, Storniert }

class Program

{

static void Main()

{

Status auftragsStatus = Status.InBearbeitung;

switch (auftragsStatus)

{

case Status.Offen:

Console.WriteLine("Auftrag wurde erfasst.");

break;

case Status.InBearbeitung:

Console.WriteLine("Auftrag ist in Bearbeitung.");

break;

case Status.Abgeschlossen:

Console.WriteLine("Auftrag ist abgeschlossen.");

break;

case Status.Storniert:

Console.WriteLine("Auftrag wurde storniert.");

break;

}

}

}

**switch mit when**

Man kann when für zusätzliche Bedingungen innerhalb case-Blöcken nutzen.

**Beispiel: Rabatte nach Alter**

int alter = 17;

switch (alter)

{

case int n when n < 18:

Console.WriteLine("Du bekommst einen Jugendrabatt.");

break;

case int n when n >= 65:

Console.WriteLine("Seniorenrabatt verfügbar.");

break;

default:

Console.WriteLine("Normaler Preis.");

break;

}

**switch expressions**

Statt einer langen switch-Anweisung kann man kompaktere **Switch-Ausdrücke** verwenden.

**Beispiel: Notenbewertung mit switch expression**

int punkte = 85;

string note = punkte switch

{

>= 90 => "Sehr gut",

>= 80 => "Gut",

>= 70 => "Befriedigend",

>= 60 => "Ausreichend",

\_ => "Nicht bestanden"

};

Console.WriteLine(note);

**switch mit Tupeln (C# 8.0+)**

Mit Tupeln lassen sich mehrere Variablen gleichzeitig in switch überprüfen.

**Beispiel: Schachfigur-Bewegung bestimmen**

(string figur, bool istErsterZug) = ("Bauer", true);

int maxFelder = (figur, istErsterZug) switch

{

("Bauer", true) => 2,

("Bauer", false) => 1,

("Springer", \_) => 3, // "\_" bedeutet, dass der Wert egal ist

("Läufer", \_) => 8,

\_ => 0

};

Console.WriteLine($"Maximal bewegliche Felder: {maxFelder}");

|  |  |  |
| --- | --- | --- |
| **Variante** | **Vorteile** | **Beste Anwendung** |
| **Klassischer switch** | Einfach, gut für feste Werte | Menüauswahl, Steuerlogik |
| **switch mit enum** | Lesbarer Code durch enum-Namen | Statusabfragen (z. B. Auftragsstatus) |
| **switch mit when** | Bedingungen innerhalb case | Komplexe Fallunterscheidungen |
| **switch expressions** | Kompakt, ideal für Zuweisungen | Notenvergabe, Statusübersetzungen |
| **switch mit Tupeln** | Mehrere Werte gleichzeitig prüfen | Kombinationen aus Parametern, z. B. Schachregeln |

**Praktische Übung:**

**Übung 1:**  
Erstelle ein Programm, das den Benutzer nach einer Zahl fragt und dann den entsprechenden Wochentag (1 = Montag, 2 = Dienstag, etc.) ausgibt.

**Übung 2:**  
Schreibe ein Programm, das die Eingabe eines Monats (1-12) vom Benutzer entgegennimmt und den Namen des Monats ausgibt.

**Schleifen**

**Inhalte:**

**for-Schleife:** Iteration mit einer festen Anzahl von Wiederholungen

**while-Schleife:** Schleifen mit einer Bedingung

**do-while-Schleife:** Schleifen, die mindestens einmal ausgeführt werden

**for-Schleife**

Die for-Schleife wird genutzt, wenn die Anzahl der Durchläufe bekannt ist.

**Beispiel: Zahlen von 1 bis 5 ausgeben**

for (int i = 1; i <= 5; i++)

{

Console.WriteLine($"Durchlauf {i}");

}

**while-Schleife**

Die while-Schleife wird verwendet, wenn die Anzahl der Durchläufe **nicht bekannt** ist, aber eine Bedingung erfüllt sein muss.

**Beispiel: Zahleneingabe bis 0 eingegeben wird**

int zahl;

do

{

Console.Write("Gib eine Zahl ein (0 zum Beenden): ");

zahl = Convert.ToInt32(Console.ReadLine());

} while (zahl != 0);

**do-while-Schleife**

Die do-while-Schleife funktioniert wie while, aber führt den Code **mindestens einmal** aus, bevor die Bedingung geprüft wird.

**Beispiel: Benutzer muss mindestens einmal raten**

int geheimzahl = 7;

int eingabe;

do

{

Console.Write("Rate die Zahl: ");

eingabe = Convert.ToInt32(Console.ReadLine());

} while (eingabe != geheimzahl);

Console.WriteLine("Richtig geraten!");

**Unendliche Schleife mit for oder while**

Manchmal wird eine Endlosschleife benötigt, z. B. für einen Server-Loop oder ein Spiel.

**Beispiel: Endlosschleife mit while**

while (true)

{

Console.WriteLine("Diese Schleife läuft unendlich!");

}

**Beispiel: Endlosschleife mit for**

for (;;)

{

Console.WriteLine("Unendliche for-Schleife!");

}

**break – Schleife beenden**

break beendet eine Schleife sofort.

**Beispiel: Zahlensuche mit while**

int n = 1;

while (true)

{

if (n == 5)

{

Console.WriteLine("5 gefunden! Beende die Schleife.");

break;

}

Console.WriteLine(n);

n++;

}

**continue – Aktuellen Durchlauf überspringen**

continue überspringt den aktuellen Schleifendurchlauf und setzt mit dem nächsten fort.

**Beispiel: Alle Zahlen außer 3 ausgeben**

for (int i = 1; i <= 5; i++)

{

if (i == 3)

continue; // Überspringt die 3

Console.WriteLine(i);

}

**goto – Schleifensteuerung (selten empfohlen)**

goto kann genutzt werden, um direkt zu einer bestimmten Code-Stelle zu springen.

**Beispiel: Schleife mit goto (nicht empfohlen)**

int x = 0;

start:

Console.WriteLine(x);

x++;

if (x < 5)

goto start; // Springt zurück zu "start"

**Fazit: Wann welche Schleife nutzen?**

|  |  |
| --- | --- |
| **Schleife** | **Beste Anwendung** |
| **for** | Wenn die Anzahl der Durchläufe bekannt ist |
| **while** | Wenn die Bedingung vor der Ausführung geprüft wird |
| **do-while** | Wenn die Schleife **mindestens einmal** laufen muss |
| **break** | Um eine Schleife gezielt zu beenden |
| **continue** | Um einen Schleifendurchlauf zu überspringen |
| **while (true)** | Für Endlosschleifen (z. B. Spiele oder Server) |
| **goto** | Sollte vermieden werden |

**Praktische Übung:**

**Übung 1:**  
Erstelle ein Programm, das die Summe der ersten n Zahlen berechnet, wobei n vom Benutzer eingegeben wird.

**Übung 2:**  
Schreibe ein Programm, das mit einer while-Schleife alle geraden Zahlen von 1 bis 20 ausgibt.

**Übung 3:**  
Erstelle ein Programm, das mit einer do-while-Schleife den Benutzer nach einer Zahl fragt und diese so lange verdoppelt, bis sie mehr als 100 ist.

**Enums**

Ein enum (Enumeration) in C# ist ein Wertetyp, der eine Sammlung von benannten Konstanten definiert. enum-Werte sind intern Ganzzahlen (int standardmäßig), können aber auch auf andere Ganzzahltypen basieren.

enum Wochentag

{

Montag, // 0

Dienstag, // 1

Mittwoch, // 2

Donnerstag, // 3

Freitag, // 4

Samstag, // 5

Sonntag // 6

}

Standardmäßig beginnt der erste Wert bei 0 und die folgenden Werte werden hochgezählt.

**Eigene Werte zuweisen**

enum Status

{

Aktiv = 1,

Inaktiv = 0,

Gelöscht = -1

}

**Enum verwenden**

Status userStatus = Status.Aktiv;

Console.WriteLine(userStatus

Console.WriteLine((int)userStatus

foreach (var wert in Enum.GetValues(typeof(Wochentag)))

{

Console.WriteLine(wert);

}

**Enum als Parameter**

void SetStatus(Status status)

{

if (status == Status.Aktiv)

Console.WriteLine("Benutzer ist aktiv.");

}

SetStatus(Status.Inaktiv);

**Enum mit anderem Basistyp**

Enums können einen anderen Ganzzahltyp als Basis haben (z. B. byte, short, long).

enum Fehlercode : byte

{

KeinFehler = 0,

Warnung = 1,

Kritisch = 255

}

**Enum in String umwandeln**

string statusText = Status.Aktiv.ToString(); // "Aktiv"

String in Enum umwandeln

Status parsedStatus = (Status)Enum.Parse(typeof(Status), "Aktiv");

Mit Enum.TryParse sicherer:

if (Enum.TryParse("Aktiv", out Status parsedStatus))

{

Console.WriteLine(parsedStatus); // Aktiv

}