Arbeitsblatt 5. Funktionen

Aufgabe 5.1. Potenzfunktion

a) Die Standardfunktion Math.Pow(x, n) berechnet die Potenz x^n. Diese sollen Sie nachimplementieren, natürlich ohne Math.Pow() zu verwenden. Multiplizieren Sie dafür x n-mal mit sich selbst.

Schreiben Sie also eine Funktion Potenz(x, n), welche zu einer vorgegebenen Double-Zahl x die n-te Potenz x^n berechnet. n soll dabei eine positive Integer-Zahl sein. Der Funktionskopf sieht so aus:

```
static double Potenz(double x, int n)
```

Schreiben Sie ein Hauptprogramm (Main), welches zur Eingabe zweier Werte "basis" und "exponent" auffordert, die Potenz $basis^{exponent}$ mittels der Potenz-Funktion berechnet und ausgibt. Benennen Sie Ihre Variablen und Parameter sinnvoll!

Beispiel:

```
Basis eingeben: 2
Exponent eingeben: 8
2^8 = 256

Basis eingeben: 3
Exponent eingeben: 3
3^3 = 27
```

b) Ändern Sie Ihr Hauptprogramm so ab, dass alle Potenzen von 1…exponent ausgegeben werden.

```
Basis eingeben: 2
Exponent eingeben: 5
2^0: 1
2^1: 2
2^2: 4
2^3: 8
2^4: 16
2^5: 32
```

Aufgabe 5.2. "Quer-Zahlen": Zahlen aus Quersumme und Querprodukt

Die Quersumme einer positiven ganzen Zahl berechnet sich durch die Summe ihrer Ziffern, d.h.

Quersumme(3521) = 3+5+2+1. Analog kann man auch ein Querprodukt bilden.

a) Schreiben Sie eine Funktion für die Berechnung der Quersumme und testen Sie sie mit Hilfe eines geeigneten Aufrufs. Konvertieren Sie die Zahl nicht in einen String, sondern nutzen Sie Modulo und ganzzahlige Division, um die Quersumme zu berechnen wie in der Vorlesung.

b) Schreiben Sie eine weitere Funktion für die Berechnung des Querprodukts.

c) Es gibt Zahlen, bei denen die Summe aus Quersumme und Querprodukt gleich der Zahl ist. Zum Beispiel 59 = 5+9 + 5*9. Schreiben Sie eine Funktion IstQuerzahl(zahl), die diese Eigenschaft für eine Zahl überprüft. Nutzen Sie dabei die beiden anderen Funktionen Quersumme und Querprodukt. (Man kann den Funktionsrumpf dann sogar in einer Zeile hinschreiben!) Testen Sie die neue Funktion durch einen geeigneten Aufruf.

d) Erstellen Sie unter Verwendung von IstQuerZahl ein Hauptprogramm, das alle zweistelligen Zahlen ausgibt, für die diese Eigenschaft zutrifft. Das Ergebnis ist überraschend!

Aufgabe 5.3. Berechnung eines Bezugspreises

Entwickeln Sie eine Funktion für die Berechnung eines Bezugspreises nach folgendem Muster:

```
Listenpreis                                                  125,00 €
- Rabatt 10%                                                 -12,50 €
+ Versandkosten                                               6,50 €
= Bezugspreis                                                119,00 €
```

a) Schreiben Sie die Funktion zur Berechnung des Bezugspreises einer Ware.
Parameter für die Funktion sind:

- Listenpreis
- Versandart (1 = Päckchen, 2 = Paket bis 10kg, 3 = Paket über 10kg).
- Rabatt (in Prozent)

Es soll gelten:

- Versandkosten für ein Päckchen sind 4,90 €, für ein Paket 6,50 € und für ein Paket über 10 kg bei 12,00 €
- Bei ungültigen Werten soll ein Preis von -1 zurückgegeben werden.

b) Testen Sie Ihre Funktion durch entsprechende Aufrufe in der Main-Funktion. Nutzen Sie bei Problemen den Debugger (F11 – Step-Into). Testen Sie mit folgenden Parametern:

- Testfall 1: wie oben angegeben (Versand = Paket)
- Testfall 2: Listenpreis 200 €, 3% Rabatt, Lieferung als Päckchen
- Testfall 3: Listenpreis 375 €, kein Rabatt, Lieferung als Paket
- Testfall 4: ungültige Versandart
- Testfall 5: ungültiger Rabatt-Wert

```
Console.WriteLine($"Fall 1: Preis = {Bezugspreis(125, 2, 10):0.00} Euro");
Console.WriteLine($"Fall 2: Preis = {Bezugspreis(200, 1, 3):0.00} Euro");
Console.WriteLine($"Fall 3: Preis = {Bezugspreis(375, 3, 0):0.00} Euro");
Console.WriteLine($"Fehler: Preis = {Bezugspreis(50, 4, 0):0.00} Euro");
Console.WriteLine($"Fehler: Preis = {Bezugspreis(150, 1, -1):0.00} Euro");

Fall 1: Preis = 119,00 Euro
Fall 2: Preis = 198,90 Euro
Fall 3: Preis = 387,00 Euro
Fehler: Preis = -1,00 Euro
Fehler: Preis = -1,00 Euro
```

Aufgabe 5.4. Zahl umdrehen

Schreiben Sie eine Funktion "Twist", welche eine beliebige ganze Zahl (positiv oder negativ) als Eingabe bekommt und die Zahl zurückgibt, welche die Ziffern in umgekehrter Reihenfolge enthält. Führende Nullen gehen dabei verloren.

Dabei dürfen Sie die Zahl natürlich nicht in einen String konvertieren oder ähnliches, sondern nur Integer-Logik (Division und Modulo) verwenden.

Die letzte Ziffer zu ermitteln ist ja einfach. Nutzen Sie das für Ihren Algorithmus.

```
Console.WriteLine("Umgekehrte Zahl: "+Twist(4567));
Console.WriteLine("Umgekehrte Zahl: "+Twist(1234567890));

7654
987654321
```