LU01.L04 - Funktionaler ggT Algorithmus

Der GGT (größter gemeinsamer Teiler) von zwei Zahlen kann funktional durch den Euklidischen Algorithmus berechnet werden. Hier ist ein Beispiel in Python, das den Euklidischen Algorithmus in einer funktionalen Programmierweise darstellt:

```python
def ggt(a, b):
    if b == 0:
        return a
    else:
        return ggt(b, a % b)

x = 60
y = 48
result = ggt(x, y)
print('Der größte gemeinsame Teiler von', x, 'und', y, 'ist', result)  # Ausgabe: Der größte gemeinsame Teiler von 60 und 48 ist 12
```

Genauer Ablauf des rekursiven ggT-Algorithmus

Euklidischer Algorithmus:

Der Euklidische Algorithmus basiert auf der Beobachtung, dass der GGT von zwei Zahlen a und b auch der GGT von b und a `%` b ist.

Rekursion:

Rekursion ist ein Ansatz, bei dem eine Funktion sich selbst aufruft. Im Euklidischen Algorithmus ist die rekursive Funktion so definiert:

```python
def ggt(a, b):
    if b == 0:
        return a
    else:
        return ggt(b, a % b)
```

Die Rekursion geht weiter, indem b als erstes Argument und der Rest von a geteilt durch b als zweites Argument übergeben wird, bis b `0` wird.

Basisfall:

Der Basisfall ist der Fall, der die Rekursion beendet. Im Euklidischen Algorithmus ist der Basisfall, wenn b `0` ist:

```python
if b == 0:
    return a
```

Hier endet die Rekursion, und der Wert von a wird als GGT zurückgegeben.

Beispiel für Rekursion:

Nehmen wir an, `a = 60` und `b = 48`. Die Rekursion funktioniert dann wie folgt:

1. `ggt(60, 48)`: Da `b != 0`, ruft es `ggt(48, 60 % 48)` auf, d.h. `ggt(48, 12)`.
2. `ggt(48, 12)`: Da `b != 0`, ruft es `ggt(12, 48 % 12)` auf, d.h. `ggt(12, 0)`.
3. `ggt(12, 0)`: Da `b == 0`, ist der Basisfall erreicht, und 12 wird als GGT zurückgegeben.

Dieser rekursive Ansatz ist eine natürliche und elegante Methode, den Euklidischen Algorithmus auszudrücken und ist ein starkes Beispiel für die Macht der funktionalen Programmierung.

Erklärung:

Der Euklidische Algorithmus verwendet Rekursion, um den GGT von zwei Zahlen zu finden. Hier ist eine Schritt-für-Schritt-Erklärung des obigen Codes:

1. **Basisfall:** Wenn der Wert von b 0 ist, ist a der GGT der beiden Zahlen.
2. **Rekursiver Fall:** Wenn b nicht 0 ist, wird die Funktion ggt erneut mit den Werten b und a % b aufgerufen.
3. **Endrekursion:** Die Rekursion wird fortgesetzt, bis der Wert von b 0 wird. Da dies ein Tail-Recursion-Szenario ist (der rekursive Aufruf ist die letzte Operation, die durchgeführt wird), kann die Tail-Recursion-Optimierung in einigen Programmiersprachen die Leistung verbessern.

Fazit

Die rekursive Natur dieses Algorithmus ist ein perfektes Beispiel dafür, wie die funktionale Programmierung für mathematische Berechnungen verwendet werden kann. Sie vermeidet veränderbare Variablen und Schleifen, wodurch der Code klar und elegant bleibt.