Übung10: Funktionsaufrufe, Rekursion

Zweck dieser Übung ist, die Erkenntnis in Funktionen zu vertiefen. Teil davon ist die Programmierung einer Rekursive Funktion.

Verzeichnis von Begriffen

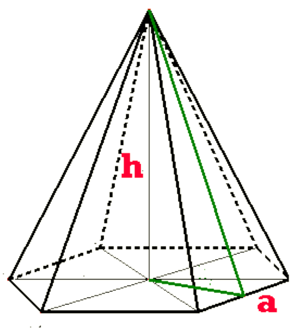
– Funktionsaufruf
– Entwicklungsteam
– Verknüpfung von Modulen
– Stapel
– Rekursion
– Pyramid

AUFGABE 1:

In diesem Beispiel wollen wir das Volumen einer Pyramide von Höhe ***h*** mit einem regelmäßigen Vieleck mit ***n*** Seiten von Seitenlänge ***a*** als Grundfläche.

$$Volumen = \frac{1}{3} \cdot Grundflaeche \cdot h$$

$$Grundflaeche = \frac{n \cdot a^2}{4 \cdot \tan\left(\frac{\pi}{n}\right)}$$



Dafür brauchen wir:

- **Funktion `PyramideVol`**
  die Funktion übernimmt als Parameter: **1)** Höhe *(float),* **2)** Anzahl von Seiten der Grundfläche **3)** Seitenlänge der Grundfläche *(float)*
  Diese Funktion berechnet das Volumen durch die obigen Formel. Sie berechnet nicht selber die Grundfläche jedoch ruft die Funktion `Grundflaeche` auf

- **Funktion `Grundflaeche`**
  Die Funktion übernimmt als Parameter 1) Anzahl von Seiten der Grundfläche 2) Seitenlänge der Grundfläche (float)
  Berechnet die Flaeche eines Vieleckes nach dem obigen Formel

Das wird durch ein Team von 3 Programmierer durchgeführt:

- **Programmierer A:** Der muss das Hauptprogramm *`main.cpp`* implementieren da werden Höhe, Seitenanzahl und Seitenlänge eingelesen und sie werden als Parameter für die Funktion **`PyramideVol`** verwendet. Durch diese Funktion wird das Volumen berechnet und ausgegeben. Diese Funktion hat der Programmierer B in Modul *`PyramideVol.cpp/.hpp`* programmiert.

- **Programmierer B:** Implementiert die Funktion **`PyramideVol`** im Modul *`PyramideVol.cpp/.hpp.`* Dafür muss die Funktion **`Grundflaeche`** nüin Anspruch nehmen.

- **Programmierer C:** Implementiert die Funktion **`Grundflaeche`** im Modul *`grundflaeche.cpp/.hpp.`* Dafür ist nützlich die Bibliothek *`math.h`*

AUFGABE 2: Wir möchten die Berechnung des *ggT* (größten gemeinsamen Teilers) programmieren.
Zielprogramm:

1. Eingabe von 2 ganzen Zahlen
2. Berechnung des *ggTs*
3. Ausgabe des Ergebnis

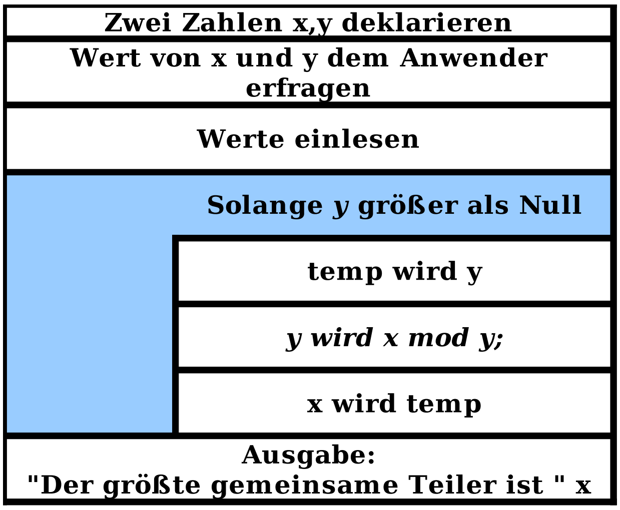
Lösungswege A und B:
A)Rekursiv Der ggT kann lässt sich rekursiv definieren:

$$ggT(x, y) = \begin{cases} x & wenn\ y = 0 \\ ggT(y, x \bmod y) & sonst \end{cases}$$

wo *mod* ist das Modulo, d.h. der Rest der Division y durch x. Dies entspricht einem arithmetischen Operator in C/C++, nämlich %. Das heißt x mod y wird `x % y` in C++ implementiert.

Implementieren diese Rekursive Funktion um die Aufgabe zu lösen.

B)Iterativ Die Iterative Version des ggT ist aus dem Struktogramm zu programmieren.



Beispiel:
Wenn Sie die Funktion ggT mit 120 (2^3*3*5) und 36 (2^2*3^2) aufrufen. Wird 12 (2^2*3) die Rückgabe, nämlich der größte gemeinsame Teiler.
ggT(120, 36) → 120 modulo 36 beträgt 12; ggT(36, 12) → 36 modulo 12 beträgt 0; ggT (12, 0) → also der ggT ist 12

Geschichtliche Anmerkung:
Die Erste Beschreibung dieses Algorithmus befindet sich im Euklids Meisterwerk „Die Elementen" um 300 vor Christus.

AUFGABE 3:
Betrachten Sie diese Summe ganzer Zahlen: $summe(n) = n + (n-1) + \cdots + 3 + 2 + 1$
I. Sie können gewiss durch eine Schleife dies programmieren. Das ist sehr einfach.
II. Und jetzt die Rekursive Funktion Programmieren. Denken Sie mal, wie wird summe(n) rekursiv definiert.