Euklidscher Algorithmus zur Bestimmung des ggT. Ich habe gewusst, wie ich den ggT mit Euklid berechnen kann, habe aber den Algorithmus zuerst herleiten müssen.

Ich habe mich für die klassische, iterative Variante entschieden. Grundsätzlich, weil ich mich an der while – Schleife üben wollte. Die Rekursiven Varianten bevorzuge ich eigentlich nie den Iterativen, falls möglich (App A), das bezieht natürlich vor allem auf Java Programme.

**int** euklid(**int** dividend, **int** divisor){  
  
 **if** (dividend == 0){  
 **return** divisor;  
 }  
  
 **while**(divisor != 0){  
 **if** (dividend > divisor){  
 dividend -= divisor;  
 } **else** {  
 divisor -= dividend;  
 }  
 }  
 **return** dividend;  
}

Die „modernen“ Varianten wären eine andere Möglichkeit gewesen. Da ich, aber die klassische Variante selber, ohne Probleme programmieren konnte, bin ich bei ihr geblieben. Die modernen Varianten habe ich bei Recherchen gefunden. (App B)

Das kgV lässt sich im Anschluss leicht mit dem ggT berechnen.

**int** smallestSameMultiplier(**int** dividend, **int** divisor){  
  
 **int** sameDivisor = euklid(dividend, divisor);  
 **int** smallestMultiplier = (dividend \* divisor) / sameDivisor;  
  
 **return** smallestMultiplier;  
}

Die Hauptprobleme hatte ich damit, dass ich mit der Schleife kämpfen musste um eine neue Rechnung durchführen zu können.

**int** choiceInput(**void**) {  
  
 **unsigned char** choice;  
  
 (**void**) printf("\nWould you like to continue [Y/N]? \n");  
 (**void**) scanf("%c",&choice);  
  
 **if**(choice == 'y' || choice == 'Y'){  
 **return** 1;  
 } **else if**(choice == 'n' || choice == 'N'){  
 **return** 0;  
 } **else** {  
 choiceInput();  
 }  
}

Nachdem ich die choice variable auf “unsigned” gesetzt habe, hat das Programm mit Wiederholungen funktioniert.

Appendix A:

Rekursiv:

**public** **int** euklid(**int** a, **int** b)

{

**if** (b == 0) {

**return** a;

} **else** **if** (a == 0) {

**return** b;

} **else** **if** (a > b) {

**return** euklid(a - b, b);

} **else** {

**return** euklid(a, b - a);

}

}

Appendix B:

Moderne Varianten:

Rekursiv:

**public** **int** euklid(**int** a, **int** b)

{

**if** (b == 0)

**return** a;

**else**

**return** euklid(b, a % b);

}

Interativ:

**public** **int** euklid(**int** a, **int** b)

{

**int** tmp = 0;

**while** (b != 0) {

tmp = a % b;

a = b;

b = tmp;

}

**return** a;

}

Appendix C:

Flow Chart der Main Methode:

