



## Berufsbildende Schulen des Landkeis Quedlinburg

Betriebs- und Volkswirtschaftslehre/Wirtschaftsinformatik

### BESONDERE LERNLEISTUNG

Veranschaulichung volkswirtschaftlicher Probleme

**Vorgelegt von:**

Michael Rennecke, FG 00/M Michael Renn-  
ecke

**Gutachter:**

Herr Dr. Haufe

Frau Blath

Klausstraße 132  
06493 Königerode

Telefon: 039484/8105

KönigerodeKönigerode, den 03.01.2003

# I Inhaltsverzeichnis

<b>II</b>	<b>Einleitung .....</b>	<b>1</b>
<b>III</b>	<b>Das BWL-Programm.....</b>	<b>3</b>
<b>1</b>	<b>Elastizitätsrechnung .....</b>	<b>4</b>
1.1	Überblick.....	4
1.2	Möglichkeiten der Rationalisierung .....	4
1.3	Berechnung der Elastizität.....	4
1.3.1	Allgemein.....	4
1.3.2	Der Ablauf der Berechnung in Visual Basic .....	5
1.3.3	Die Berechnung in C++.....	5
1.4	Bedienung und Aufbau der Elastizitätsrechner .....	5
1.4.1	Der Visual Basic-Elastizitätsrechner .....	5
1.4.2	Der C++-Elastizitätsrechner .....	6
1.5	Elastizitätsrechnung mit EXCEL.....	6
1.6	Beurteilung der Lösungen.....	7
<b>2</b>	<b>Der Cournotsche Punkt .....</b>	<b>8</b>
2.1	Überblick.....	8
2.2	Möglichkeiten der Rationalisierung .....	8
2.3	Ermittlung des Cournotschen Punktes.....	8
2.3.1	Die Berechnung in C++.....	8
2.3.2	Die Berechnung in Visual Basic.....	9
2.4	Die Bedienung der Lösungen.....	9
2.4.1	Der C++-Monopolrechner.....	9
2.4.2	Der Visual Basic Monopolrechner.....	10
2.5	Das vollkommene Monopol mit EXCEL.....	10
2.5.1	Die Wertetabelle.....	10
2.5.2	Die graphische Darstellung.....	11
2.6	Beurteilung der Lösungen.....	11
<b>3</b>	<b>Geldschöpfung.....</b>	<b>13</b>
3.1	Überblick.....	13
3.2	Möglichkeiten der Rationalisierung.....	13
3.3	Berechnung der Entwicklung der Geldschöpfung.....	13
3.4	Die C++- und Visual Basic-Lösung.....	14
3.5	Funktionsweise der Visual Basic Lösung.....	14
3.6	Die EXCEL-Lösung.....	15
3.7	Beurteilung der Lösungen.....	15
<b>4</b>	<b>Vergleich der Lösungen.....</b>	<b>16</b>
4.1	Portierbarkeit.....	16
4.2	Ressourcenverbrauch.....	16
4.3	Korrektheit.....	17

4.4	Robustheit.....	17
4.5	Erweiterbarkeit.....	18
4.6	Wiederverwendbarkeit.....	18
4.7	Integrität.....	18
4.8	Arbeitsaufwand.....	18
<b>5</b>	<b>Demonstration der Elastizität.....</b>	<b>20</b>
5.1	Überblick.....	20
5.2	Funktionsweise.....	20
5.3	Die Oberfläche.....	20
5.4	Bewertung.....	20
<b>6</b>	<b>Einschätzung.....</b>	<b>21</b>
<b>IV</b>	<b>Quellenverzeichnis.....</b>	<b>22</b>
1.	Bücher .....	22
2.	Digitale Medien .....	22
<b>V</b>	<b>Anhang.....</b>	<b>23</b>
1.	Die fkt::double chardouble-Funktion.....	23
2.	Die CBUVWLView::Determinaten-Funktion.....	24
3.	Geldschöpfung.....	26
<b>VI</b>	<b>Kurzfassung.....</b>	<b>28</b>
<b>VII</b>	<b>Eidesstattliche Erklärung.....</b>	<b>29</b>

## II Einleitung

Meine besonderen Lernleistung hat folgendes Thema:

Veranschaulichung ausgewählter volkswirtschaftlicher Themen der betriebs- und volkswirtschafts Leistungskurse 2 und 3

- a) unter Verwendung von Standardsoftware
- b) unter Verwendung von Programmiersprachen (Visual Basic, C++)
- c) Vergleich der Lösungen von den Aufgaben a und b unter folgenden Aspekten: Benutzerfreundlichkeit, Robustheit, Zuverlässigkeit, Arbeitsaufwand, Portierbarkeit<sup>1</sup> usw.

Mit dieser besonderen Lernleistung möchte ich zeigen, wie man volkswirtschaftliche Probleme zeitsparend lösen kann. Jedes Problem wird in dieser besonderen Lernleistung dreimal gelöst. Die Lösungen vergleiche ich dann, um die Aufgabe c zu lösen. Mit der Aufgabe c möchte ich zeigen, welche der drei Lösungen, die Beste unter bestimmten Kriterien ist. Der Punkt a soll verdeutlichen, wie jeder Computerbenutzer ein Problem lösen kann. Die Lösungen mit der Programmiersprache Visual Basic haben ein höheres Niveau. Diese sollten für Schüler, die Programmiergrundlagen aus den Wirtschaftsinformatikunterricht haben, nachvollziehen können. Aus diesem Grund sind alle Visual Basic-Lösungen eigenständige Programme. Dadurch ist das einzelne Programm klein und überschaubar. Die C++-Lösung soll zeigen wie man derartige Probleme in der Praxis lösen könnte, da die meiste kommerzielle Software auch in C++ geschrieben ist.

### Zur Lösung der Aufgaben

Mit der Programmiersprache C++ werde ich ein komplexes Programmsystem erstellen, mit dem man alle Probleme, die ich behandle, lösen kann. Lösungen, welche ich mit Standardsoftware und Visual Basic erstelle, sind als eigenständige Tools<sup>2</sup> zu sehen. Ich habe mich so entschieden, da ein weiteres Programmsystem in Visual Basic ein zu großer Programmieraufwand wäre und die Lösung wäre sehr schwer zu verstehen, wenn man nur über das Schulwissen in der Programmierung verfügt. Bei den Tools werde ich aber versuchen einen anderen Lösungsweg einzuschlagen, um auch Aufgaben eines anderen Typs zu rechnen.

Die Lösungen, welche ich mittels der Standardsoftware bringe, werden im entsprechenden Kapitel dieser besonderen Lernleistung dokumentiert und kommentiert. Die Kommentare kann ich nicht in die Kalkulationstabellen schreiben, da diese das Layout schädigen.

Den Source-Code<sup>3</sup> werde ich mit den Mitteln der jeweiligen Programmiersprache kommentieren. Dieser wird auszugsweise im Anhang und komplett auf der CD-ROM zu finden sein. Die Funktionsweise eines Programms wird kurz verbal beschrieben. Eine ausführliche Beschreibung aller Funktionen würde den Rahmen der Lernleistung sprengen. Jedes Programm wird auch noch im dazugehörigen Kapitel dokumentiert.

### Zur Einteilung der Kapitel

Jedes Kapitel ist nach dem gleichem System aufgebaut. Im ersten Unterpunkt stelle ich das volkswirtschaftliche Problem kurz vor, damit Sie sich einen groben Überblick verschaffen können. Im darauf folgenden Punkt erläutere ich die Rationalisierungsmöglichkeiten. So

können Sie sich schon ein grobes Bild der Lösungen machen. In den nächsten Abschnitten löse ich die Aufgabenstellung mit den entsprechenden Werkzeugen. Im letzten Kapitel wird die Aufgabe c gelöst.

### III Das BVWL-Programm

Wie in der Einleitung schon erwähnt, werde ich mit C++ ein Programmsystem programmieren. An dieser Stelle erkläre ich die Funktionsweise des Programms. Dadurch können Sie den Zusammenhang der einzelnen Funktionen besser verstehen.

In der Datei *funktionen.cpp* befinden sich alle Funktionen, die ich zur Verarbeitung und zum Testen der Daten benötige. Man findet z.B. Testfunktionen, welche sagen, ob der eingegebene Wert eine Zahl ist oder nicht. Es gibt zwei derartige Testfunktionen: Für Dezimalzahlen und natürliche Zahlen.

Die Funktion *double fkt::chardouble(char zahl\_text[12])* ist eine sehr wichtige Funktion, denn sie wandelt eine Dezimalzahl so um, dass der Computer sie lesen kann. Der grundlegende Gedanke in dieser Funktion ist der, dass das Komma in einen Dezimalpunkt umgewandelt wird. Dazu werden alle Vorkommastellen in eine Variable<sup>4</sup> gespeichert, ein Dezimalpunkt gesetzt und dann die Nachkommastellen hinzugefügt. Diese Variable wird anschließend in eine Variable umgewandelt, welche vom Typ Zahl ist. Den Source-Code dieses Algorithmus finden Sie im Anhang.

Weitere Funktionen sind die Rundenfunktionen und Funktionen zur Konvertierung meiner selbst programmierten Datentypen.

Das Aussehen des Fensters wird in der Datei *MainFrm.cpp* bestimmt. Die dort befindlichen Routinen sorgen für ein reibungslosen Ablauf zwischen den verschiedenen Elementen. Das sind die Symbolleiste, die Menüleiste und die Dialogboxen.

Damit die Daten angezeigt werden, sind die Funktionen in der Datei *BUVWLView.cpp* von Nöten. Für die Darstellung von Graphen wird die Funktion *void CBUVWLView::OnDraw(CDC\* pDC)* benutzt. Zu jeder graphischen Ausgabe gehört eine Variable, worin steht, ob diese Funktion im Augenblick benutzt wird. Wenn die Auswertung einer solchen Variablen positiv ist, dann werden die Zeichenfunktionen abgearbeitet. Wenn mehrere Funktionen zeitgleich in Benutzung sind, werden auch diese Zeichenfunktionen ausgeführt. Die Darstellung und Einteilung des Koordinatensystems wird auch in dieser Funktion gemacht.

Alle Funktionen, die für das Speichern von Daten, im Hauptspeicher oder auf Datenträgern nötig sind, befinden sich in der Datei *BUVWLDoc.cpp*. Außerdem findet man dort auch noch Info-Boxen für die verschiedenen volkswirtschaftlichen Probleme. Diese sind in dieser Datei, da der Zugriff auf die nötigen Daten schneller vonstatten geht. Denn wie schon erwähnt werden hier die Daten im Hauptspeicher gespeichert.

Die Datei *KoorSys.cpp* enthält alle Funktionen, die das Koordinatensystem einteilen und beschriften. Der primäre Inhalt dieser Datei sind aber die Routinen, welche den Eingabedialog für die Eingabe der Parameter des Koordinatensystems erzeugen. Diese Daten werden durch die Datei *BUVWLDoc.cpp* gespeichert und durch die Datei *BUVWLView.cpp* ausgegeben.

# 1 Elastizitätsrechnung

## 1.1 Überblick

Die Elastizitätsrechnung gibt an, wie die Nachfrage auf Änderungen des Preises bzw. des Einkommens reagiert. Es gibt drei verschiedene Elastizitäten:

- a) Die direkte Preiselastizität gibt Auskunft darüber, wie sich die Nachfragemenge eines Gutes ändert, wenn man den Preis des Gutes ändert.
- b) Die indirekte Preiselastizität benutzt man um auszurechnen, wie sich die Preisänderung eines Gutes auf die Nachfragemenge eines anderen Gutes auswirkt.
- c) Mit der Einkommenselastizität berechnet man wie sich die Nachfragemenge ändert, wenn sich das Volkseinkommen ändert.

$$\text{Allgemeine Formel: } EL = \frac{\Delta x \%}{\Delta P \%}$$

$$\text{umgestellte Formel: } EL = \frac{\Delta x * P}{x * \Delta P} \quad (x = \text{Nachfragemenge, } P = \text{Preis})$$

## 1.2 Möglichkeiten der Rationalisierung

Man kann die Prozentzahlen vom Computer automatisch berechnen lassen. Dadurch erspart man sich diese Berechnungen im Vorfeld und somit ist eine Fehlerquelle ausgeschaltet. Um den Komfort der Softwarelösungen zu erhöhen, kann man die Änderung in Prozent einer Größe oder zwei Werte einer Größe eingeben. Die fehlenden Werte werden dann von der Software, neben dem Ergebnis ausgegeben, sofern es die eingegebenen Werte zulassen. So erhält man einen kompletten Überblick über sein Problem. Eine Möglichkeit, die ich mit C++ realisiere ist, dass die Elastizität graphisch dargestellt wird, was die Anschaulichkeit erhöht.

## 1.3 Berechnung der Elastizität

### 1.3.1 Allgemein

Die Elastizität wird nach der allgemeinen Formel berechnet. Es ist egal um welche Elastizität es sich handelt, denn pauschal gilt: Die Elastizität ist die Mengenänderung in Prozent geteilt durch die Preisänderung in Prozent. Die Prozentwerte für Mengen- bzw. Preisänderung findet die Prozedur in den Eingabefeldern. Falls eine oder beide Prozentzahlen gegen Unendlich laufen, wird die Elastizität mit dem Grenzwert berechnet. Danach folgt ein einfacher Dreisatz, der den Wert der Elastizität berechnet, wenn die Prozentwerte vorliegen.

### **1.3.2 Der Ablauf der Berechnung in Visual Basic**

Das Programm liest die Daten der Eingabefelder aus und speichert sie in Variablen. Es wird dann abgefragt, ob in den Variablen ein Wert ist. Bei einer gültigen Eingabe enthalten zwei bis vier Variablen Werte. Es gibt zwei Kontrollstrukturen (if- bzw. Wenn-Anweisungen). So werden alle logischen Möglichkeiten der Eingabe abgefangen. Die erste Kontrollstruktur behandelt die Eingabe der Mengenänderung und die zweite der Preisänderung. In diesen Kontrollstrukturen wird jeweils der fehlende dritte Wert berechnet. Danach wird ein Unterprogramm aufgerufen, das die Prozentzahl für die Menge bzw. den Preis berechnet, die Prozentrechnung wird mit dem Dreisatz gelöst. Nachdem das Programm die Prozentzahlen für die Menge und den Preis berechnet hat, wird ein weiteres Unterprogramm gestartet. Dieses berechnet die Elastizität.

### **1.3.3 Die Berechnung in C++**

Es werden nach dem Betätigen der OK-Taste die Daten nacheinander in Variablen geschrieben. Man muss aber alle für die Elastizität relevanten Eingabefelder ausfüllen. Danach wird der Inhalt der Eingabefelder auf Fehler überprüft. Sowie das Programm einen Fehler bemerkt, zeigt es diesen an und löscht die fehlerhafte Eingabe. Wenn alle Eingabewerte richtig sind, werden sie in eine Struktur<sup>5</sup> geschrieben. Die Elastizität wird dann graphisch im Koordinatensystem dargestellt und alle Daten über den Graphen werden in einer Infobox angezeigt, sowie der Zahlenwert der Elastizität. Dieser wird von der Infobox, nach der umgestellten Formel berechnet. Wenn die Elastizität unendlich ist, wird das über eine if-Funktion abgefangen und als Wert der Elastizität erscheint dann das Wort „unendlich“.

## **1.4 Bedienung und Aufbau der Elastizitätsrechner**

### **1.4.1 Der Visual Basic-Elastizitätsrechner**

Die Oberfläche dieser Lösung habe ich mit Tool Tipp<sup>6</sup> Texten ausgestattet, damit man sich schnell an sie gewöhnt.

Auf der Oberfläche stehen Ihnen acht Eingabefelder zur Verfügung. Wenn sie mit der Eingabe fertig sind, können Sie den Berechnen-Button betätigen. Daraufhin erscheint auf der Oberfläche die Elastizität.

Mit dem Button Neu werden alle Eingabefelder geleert. Über den Info-Button erreichen Sie ein Infodialog mit einigen Informationen. Das Programm können Sie mit dem Beenden-Button schließen.

Die Art der Elastizität kann man mit den drei Knöpfen festlegen, welche von einem Rahmen umgeben sind.

### **1.4.2 Der C++-Elastizitätsrechner**

Wie schon in der Einleitung erwähnt, ist dieser Elastizitätsrechner in einem Programmsystem untergebracht. Sie haben drei Möglichkeiten auf diese Funktion zuzugreifen. Man kann in der Symbolleiste den Taschenrechner anklicken oder im Menü Funktionen den Punkt Graphen auswählen, alternativ kann man auch die Tastenkombination Strg+G benutzen. In allen drei Fällen erscheint ein Dialog. In diesem müssen sie alle Felder bis auf das Feld Bezeichnung ausfüllen. Wenn sie eine ungültige Eingabe gemacht haben, erhalten sie eine Fehlermeldung. Das Feld Bezeichnung enthält optional die Bezeichnung des Graphen. Im Pull-Down-Menü Farbe ist die Farbe des Graphen auszuwählen. Wenn alle Eingaben gemacht sind, kann man den Button OK seine Eingabe bestätigen.

Nun wird der Graph in das Koordinatensystem gezeichnet. In einem Dialog sehen Sie nun alle Parameter des Graphen und die Elastizität. Man kann maximal zehn Graphen eingeben. Den Infodialog mit den Parametern aller Graphen kann man auch schließen und mit der Tastenkombination Alt+I, mit einem Klick auf „EL“ in der Symbolleiste oder der Punkt Infobox im Menü Ansicht wieder anzeigen lassen.

Wenn man einen Graphen löschen möchte, kann man dies auch mit den bewährten drei Möglichkeiten tun; die Tastenkombination ist Strg+L, in der Symbolleiste ist es der Müll-eimer und der Punkt Elastizität löschen im Menü Funktionen. Es erscheint ein Dialog. Sie können ein Eintrag anklicken und dann mit den Button Löschen löschen. Nun ist der gelöschte Eintrag nicht mehr in der Liste, aber erst nach dem Betätigen der OK-Taste ist der Eintrag wirklich gelöscht. Wenn sie Abbrechen wählen, wird nichts gelöscht, auch wenn sie schon einen Eintrag im Löschen gelöscht haben. Sie können natürlich auch mehrere Einträge nacheinander löschen.

## **1.5 Elastizitätsrechnung mit EXCEL**

Diese Aufgabe habe ich mit Microsoft EXECL gelöst. Damit die Bedienung so einfach wie möglich bleibt, sind Makros<sup>7</sup> eingebaut. Diese helfen nur bei der Bedienung und nicht bei der Berechnung der Elastizität. Für die Berechnung der Elastizität benutzte ich nur die Mittel der Tabellenkalkulation, damit es keine programmierte Visual Basic-Lösung wird. Denn EXECL benutzt als Makrosprache Visual Basic for Applikations (VBA) und das ist mit Visual Basic zum Teil identisch.

### **Aufbau und Funktionsweise der Tabelle**

Die Arbeitsmappe beinhaltet zwei Tabellen. Nur die erste Tabelle ist für den Benutzer interessant, da sich dort der eigentliche Rechner befindet. In der zweiten Tabelle befinden sich die Routinen zur Berechnung der Elastizität.

Sie können nur Werte in die weißen Zellen eingeben. Mit den drei Buttons können sie die Art der Elastizität festlegen. Der Button Neu löscht eine alte Eingabe. Die Tabelle fängt schon während der Eingabe an zu rechnen, deshalb brauchen sie kein Button für den Start der Berechnung zu betätigen. Die Lösung erscheint in einer Tabelle unter der Eingabetabelle.

Im Rechenteil der Tabelle wird die Eingabe übertragen. EXCEL berechnet nun aus der Eingabe die fehlenden Werte. Da es zu unübersichtlich ist, wenn man die Berechnung mit logischen Strukturen erledigt, gibt es für jede Möglichkeit eine extra Berechnung.

Dadurch erhält man mehrere gleiche Ergebnisse für einen Wert. Damit ich mit diesem weiter rechnen kann, wird das arithmetische Mittel aller Werte gebildet. Dieses benutze ich für die weitere Berechnung. Denn diese Position in der Tabelle ist fest.

## **1.6 Beurteilung der Lösungen**

Die Visual Basic-Lösung ist auf den ersten Blick mit der EXCEL-Lösung identisch aber sie rechnen auf verschiedenen Wegen. Die C++-Lösung fällt durch ihre graphische Ausgabe auf und der etwas anderen Darstellung der Ergebnisse.

Die EXCEL-Lösung ist meiner Meinung nach die schlechteste Lösung. Denn die Rechenroutinen wirken, wegen des begrenzten Funktionsumfang von EXCEL sehr unübersichtlich. Man könnte natürlich auch hier, wie in der Visual Basic-Lösung mit logischen Funktionen arbeiten. Diese kann man aber nicht strukturieren, was sich sehr nachteilig auf die Wartbarkeit auswirkt. Auf das Abfangen von Fehlern habe ich absichtlich verzichtet, das geht natürlich auf Kosten der Benutzerfreundlichkeit. Die Behandlung von Eingabefeldern ist in EXCEL relativ schwierig aber man erhält die von EXCEL typischen Fehlermeldungen. Ein Benutzer, der EXCEL oft benutzt, müsste so seinen Eingabefehler schnell finden. Der große Vorteil der EXCEL-Lösung ist der, dass sie auch ein Laie gut nachvollziehen kann.

Die Visual Basic-Lösung hat die Nachteile der geringen Benutzerfreundlichkeit nicht, da alle Eingabefehler abgefangen werden und danach erhält man eine klar formulierte Fehlermeldung. Der größte Nachteil dieser Lösung ist, dass Laien den Source-Code schwer nachvollziehen können. Ein Programmierer findet sich hingegen recht schnell im Source-Code zurecht, da das Programm relativ klein ist. Aus diesem Fakt resultiert die gute Wartbarkeit. Wenn man nur die Elastizität berechnen möchte, ist diese Lösung mit die erste Wahl.

Die C++-Lösung des Elastizitätsrechners weist eine hohe Benutzerfreundlichkeit auf. Da er in meinem Programmsystem integriert ist, gibt alle drei Möglichkeiten, wie in vielen Windows-Programmen, wie man zu dieser Funktion gelangt: Die Symbol-, Menüleiste und der Hot-Key. So kann sich jeder eine Art der Bedienung auswählen. Den Source-Code können nur erfahrene Programmierer lesen, da ich viele Sprachmittel von C++ verwende. Einige Funktionen sind trotzdem noch recht leicht zu verstehen, weil man den Quellcode in C++ sehr gut strukturieren kann. Die Wartbarkeit ist gut, aber nicht so gut wie in der Visual Basic-Lösung. Der C++-Elastizitätsrechner hat eine graphische Ausgabe der Elastizität, was die Anschaulichkeit sehr erhöht.

Was die Zuverlässigkeit angeht, so kann ich behaupten, alle drei Lösungen arbeiten auf dem gleichen hohen Niveau. Wenn man keine graphische Ausgabe der Elastizität benötigt ist die Visual Basic-Lösung, meiner Meinung nach, zu bevorzugen. Die C++-Lösung hat ihre Stärken in der graphischen Ausgabe, was ein Vorteil gegenüber der Visual Basic Lösung ist.

## **2 Der Cournotsche Punkt**

### **2.1 Überblick**

Der Cournotsche Punkt zeigt die für den Monopolisten die gewinnmaximierende Preis-Mengen-Kombination. Wenn er den errechneten Preis annimmt dann wird er auch ca. die errechnete Menge absetzen und somit den höchsten Gewinn erzielen.

### **2.2 Möglichkeiten der Rationalisierung**

Bei dem graphischen Weg zur Lösung dieses Problems lässt sich sehr viel rationalisieren. Die für das Diagramm benötigten Werte kann man rechnerisch ermitteln und danach diese Werte in ein Koordinatensystem eintragen. Den Cournotschen Punkt kann man dann von der Software suchen und „ablesen“ lassen, was die Genauigkeit erhöht.

Für den mathematischen Lösungsweg benötigt man seine Preis-Absatz-Funktion und die Kosten. Daraus resultieren dann zwei Gleichungen, die abgeleitet werden müssen und dann muss der Wert für  $x$  (entspricht der Menge) berechnet werden. Diese ganze Berechnung kann der Computer machen, dadurch braucht man keine Kenntnisse über Ableitungen. Wenn man beide Wege ohne Hilfsmittel ausführt, benötigt man viel Zeit, insbesondere für die graphische Lösung. Durch das Lösen mit dem Computer hat man also einen sehr großen Zeitvorteil.

### **2.3 Ermittlung des Cournotschen Punktes**

#### **2.3.1 Die Berechnung in C++**

Die C++-Lösung ermittelt den Cournotschen Punkt auf dem graphischen Weg. Dieser wird aber berechnet.

In einem Dialog erstellt man die Wertetabelle. Die Werte werden sofort auf nicht numerische Eingaben hin überprüft, falls dieser Test negativ ausfällt, erscheint eine Fehlermeldung. Während der Eingabe werden schon die ersten Unbekannten, wie Gewinn und Gesamterlös ermittelt, außerdem werden die eingegebenen Daten nach der Menge sortiert. Wenn man alle Daten eingegeben hat, werden die Grenzerlöse und die Grenzkosten ermittelt. Anschließend wird die Differenz von Grenzerlös und Grenzkosten berechnet, ist diese kleiner als die Aktuelle, werden die Differenz der Grenzerlös und der Grenzkosten gespeichert. Dieser Schritt ist sehr wichtig, um den Cournotschen Punkt zu finden, denn an der Stelle, wo die Differenz zwischen Grenzerlöse und Grenzkosten am kleinsten ist, befindet sich der Cournotsche Punkt.

Nachdem alle Daten diese Prozedur durchlaufen haben, wird der Cournotsche Punkt nach einer von mir entwickelten Methode berechnet. Dafür wird mit der Funktion

*CBUVWLView::Determinaten(double \*wert, double P[4][2])* der x-Wert und danach der y-Wert des Cournotschen Punkts berechnet. Wenn es keinen Cournotschen Punkt geben sollte, schreibt das Programm -1 in den Cournotschen Punkt. Wenn dann alle Daten von der Infobox ausgegeben werden, wird „--“ an die Stelle des Cournotschen Punktes geschrieben, falls es keinen geben sollte.

### **2.3.2 Berechnung in Visual Basic**

Diese Lösung arbeitet nach einem ganz anderen Verfahren zur Berechnung des Cournotschen Punkts, als die C++-Lösung. Der Cournotsche Punkt wird hier auf dem mathematischen Weg ermittelt. Dazu benötigt das Programm folgende Eingabewerte: Preis-Absatz-Funktion, fixe Gesamtkosten und variable Stückkosten.

Zuerst bildet das Programm aus der Preis-Absatz-Funktion die Erlösfunktion und leitet diese einmal ab. Danach wird der Cournotsche Punkt ermittelt. Dafür werden die erste Ableitung der Erlösfunktion mit den variablen Stückkosten gleichgesetzt. Das Ergebnis ist die Cournotsche Menge. Damit man den Cournotschen Preis erhält, wird die Cournotsche Menge in die Preis-Absatz-Funktion eingesetzt. Die Lösung ist dann der Cournotsche Preis.

Nun folgt die Ausgabe des Cournotschen Punktes. Anschließend wird die Kostenfunktion und deren erste Ableitung gebildet und ausgegeben, damit man einen Gesamtüberblick erhält. Die erste Ableitung der Kostenfunktion sind die variablen Stückkosten. Aus diesem Grund kann ich bei der Berechnung der Cournotschen Menge die variablen Stückkosten benutzen und muss nicht vorher die erste Ableitung der Kostenfunktion bilden. Als letzte Ausgabe folgt die Erlösfunktion und ihre erste Ableitung.

## **2.4 Die Bedienung der Lösungen**

### **2.4.1 Der C++-Monopolrechner**

Sie können die Monopolfunktion im Programmsystem auf folgenden Wegen erreichen: Die Tastenkombination Strg+G, den Punkt Graphen im Menü Funktion und über den Taschenrechner in der Symbolleiste. Nun sehen sie einen Dialog, anschließend müssen Sie den Reiter Monopol anklicken, um zur Monopolfunktion zu gelangen.

Im Pull-Down-Menü Darstellung können Sie die graphische Form der Darstellung wählen. Es gibt die Gesamt- und die Stückbetrachtung, alternativ können Sie sich auch beide Darstellungen in einem Diagramm einzeichnen lassen. In den folgenden drei Eingabefeldern müssen Sie die Zahlenwerte für Menge, Gesamtkosten und Stückpreis eintragen. Für die Menge erhalten Sie einen Vorschlag, den Sie aber nicht annehmen müssen. Dann müssen Sie den Button Hinzufügen betätigen. Nachdem Sie das getan haben, erscheinen Ihre Werte in der rechts stehenden Ausgabebox. Wenn Sie einmal einen falschen Datensatz eingegeben haben, können Sie diesen in der Ausgabebox markieren und dann auf Löschen klicken. Sie können auch den falschen Datensatz doppelt anklicken, dieser ist dann wieder in den Eingabefeldern zu finden und kann bearbeitet werden. Der Datensatz wird nun gelöscht. Bei jeder Eingabe werden die Zahlen auf numerische Werte hin überprüft, wenn Sie ein Feld leer lassen, wird angenommen Sie wollten eine Null eingeben.

Wenn Sie alle Datensätze eingegeben haben, müssen Sie die OK-Taste anklicken. Wenn Sie Werte eingegeben haben, bei denen das Koordinatensystem nicht ausreicht, erhalten Sie eine Fehlermeldung, mit der Ursache und der Zeile, in der sich der Fehler befindet. Die Graphen werden, wenn keine Fehler auftreten, entsprechend Ihrer Auswahl gezeichnet. Außerdem erscheint eine Infobox mit allen Informationen über Ihre Eingabe, den berechneten Werten und dem Cournotschen Punkt. Diese Infobox können sie auch schließen und mit Alt+M, den Punkt Monopol-Box im Menü Ansicht oder das „M“ in der Symbolleiste öffnen. Wenn Sie noch weitere Werte eingeben wollen, einige löschen oder eine andere Darstellungsform wollen, verfahren Sie so wie am Anfang des Punktes beschrieben. Ihre eingegebenen Werte sind dann schon in der Ausgabebox und können wie gewohnt bearbeitet werden.

### **2.4.2 Der Visual Basic Monopolrechner**

Wenn Sie das Programm gestartet haben, müssen Sie eine Preis-Absatz-Funktion eingeben. Sie müssen nur Anstieg, das Operationszeichen des zweiten Koeffizienten und den zweiten Koeffizienten in die entsprechenden Eingabefelder schreiben. In das Feld fixe Gesamtkosten müssen die fixen Gesamtkosten eingegeben werden, das gleiche gilt für das Feld variable Stückkosten. Die Berechnung startet nachdem Sie die Berechnen-Taste angeklickt haben. Mit dem Button Neu werden alle Eingabefelder geleert, der Knopf Leeren löscht den Inhalt der Ausgabebox und die Ende-Taste schließt das Programm. Alle Eingabefelder und Tasten sind mit Tool Tipp Texten ausgestattet, die Ihnen die Funktion des jeweiligen Element zeigt.

## **2.5 Das vollkommene Monopol mit EXCEL**

Mit der EXCEL-Lösung wird der Cournotsche Punkt nicht ausgerechnet. Dieses Verfahren gleicht dem, wie man den Cournotschen Punkt von Hand auf den graphischen Weg berechnet. EXCEL wirkt aber bei der Erstellung der Diagramme und der Wertetabelle unterstützend. Den Cournotschen Punkt muss man, wenn alle Daten eingegeben sind, ablesen.

### **2.5.1 Die Wertetabelle**

Die Wertetabelle befindet sich auf dem ersten Blatt in der Arbeitsmappe. Mit dem Button oben links werden die Eingaben in der Wertetabelle gelöscht. Das Löschen der Eingaben wird mit einem Makro gesteuert. Man muss die Spalten Menge, Gesamtkosten und Preis ausfüllen. Wenn das geschehen ist, werden der Gesamterlös und der Gewinn berechnet. Die nächste Eingabe erfolgt in der übernächsten Zeile, wenn man versucht in die nächste zu schreiben, erhält man eine Meldung, dass das nicht geht. Nachdem die Zeile fertig eingegeben ist, werden die Grenzkosten und der Grenzerlös berechnet. Die eben beschriebenen Routinen wiederholen sich bis man mit der Eingabe aufhört oder man das Tabellenende erreicht hat.

## **2.5.2 Die graphische Darstellung**

Die graphische Darstellung der Stück- bzw. Gesamtbetrachtung befindet sich in den gleichnamigen Tabellen. Die Funktionsweise der Darstellung ist in beiden Fällen gleich. Es werden sich für das Diagramm alle benötigten Werte aus der Wertetabelle geholt und in der Tabelle, wo sich das Diagramm befindet, hinterlegt. Das eigentliche Diagramm ist die vorgefertigte Diagrammfunktion von EXCEL. Die Kurven werden rund dargestellt, was aber dann manchmal zur Folge hat, dass es in den Graphen zwischen zwei Punkten einen Extrempunkt gibt. Wenn man nur die Punkte verbinden lässt, wird der Graph in der Darstellung zu eckig. Den Cournotschen Punkt muss man dann ablesen.

## **2.6 Beurteilung der Lösungen**

Alle drei Lösungen sind verschieden, wobei sich die C++- und die EXCEL-Lösung vom Grundkonzept ähneln.

Die EXCEL-Lösung ist meiner Meinung nach wieder die schlechteste. Denn mit ihr kann man den Cournotschen Punkt nur ablesen und es werden keine Eingabefehler abgefangen. Die EXCEL-Lösung hat aber eine überdurchschnittlich gute Wartbarkeit, weil die Funktionen sehr einfach sind. So kann jeder den Rechenweg nachvollziehen und die Funktionsweise der Tabelle verstehen. Da die Diagramme mit den Diagrammassisten erstellt wurden, kann man diese auch schnell und effektiv verändern. Das dürfte allerdings nicht nötig sein, weil man den Cournotschen Punkt nur ablesen kann erreicht man keine höhere Genauigkeit, als wenn man das ganze von Hand macht. Durch die Computerunterstützung benötigt man nur ein Minimum an Zeit. Die Benutzerfreundlichkeit ist in dieser Lösung am höchsten, weil man das Problem auch auf die gleiche Art von Hand lösen würde.

Die Visual Basic-Lösung besticht durch ihren geringen Eingabeaufwand. Das kommt aber daher, dass der Cournotsche Punkt auf den mathematischen Weg ermittelt wird. Der größte Vorteil dieser Lösung gegenüber den anderen ist der, dass man mehrere Berechnungen machen kann und ihre Lösungen miteinander vergleichen kann. Es gibt aber einen entscheidenden Nachteil, als Unternehmer weiß man sich nicht immer seine Preis-Absatz-Funktion. Deshalb ist diese Lösung weniger realitätsnah. Die Wartbarkeit ist bei dieser Lösung gut, man benötigt neben wenigen Programmierkenntnissen noch Wissen über Ableitungen um den Source-Code nachzuvollziehen. Positiv fällt bei dieser Lösung auch auf, dass sämtliche Eingabefehler abgefangen werden. Wenn man die Preis-Absatz-Funktion zur Verfügung hat, ist diese Lösung die beste, man verzichtet aber auf eine graphische Auswertung.

Die C++-Lösung ist in das Programmsystem eingebaut. Die lässt die Oberfläche kaum Wünsche offen. Sie bietet alle Werkzeuge, die man für eine effektive Bearbeitung der Daten benötigt. Das ist der Lösch-Button, die Möglichkeit eine Eingabe zu editieren, die automatische Sortierung der Datensätze nach der Menge und Fehlermeldungen, die den Fehler genau lokalisieren. Durch diese Fülle an Werkzeugen kann ich behaupten, dass die Benutzerfreundlichkeit gut ist und sie sind auch noch leicht zu benutzen. Den Source-Code dieser Lösung können nur Programmierer verstehen. Ein Grund ist, dass die Daten im Eingabedialog auf ihre weitere Verarbeitung vorbereitet werden. Außerdem sind die Algo-

rithmen auf mehrere Funktionen verteilt. Das habe ich gemacht, damit die Funktionen übersichtlich bleiben, was die Wartbarkeit erhöht. Durch die Vorbereitung der Daten erziele ich einen Geschwindigkeitsvorteil, weil alle Daten schon richtig sortiert sind und schon auf Zahlenwerte hin überprüft wurden. Die Wartbarkeit ist aber im Vergleich zu den anderen Lösungen deutlich schlechter, da die Funktionen immer noch sehr komplex sind. Ich habe auch noch viele Sprachmittel von C++ verwendet, dadurch werden die Funktionen effektiver, aber deshalb für Anfänger kaum nachvollziehbar.

Ich favorisiere die C++ Lösung, weil man eine graphische Ausgabe hat und die Achsen des Koordinatensystems kann man selbst einteilen. Das ist bei der EXCEL-Lösung nicht möglich. Die Visual Basic Lösung hat aber auch ihre Berechtigung, wenn man eine Preis-Absatz-Funktion gegeben hat kann man den Cournotschen Punkt nur über Umwege mit den beiden anderen Lösungen berechnen bzw. ablesen. In den meisten Fällen sind aber keine Funktionen gegeben, deshalb ist die C++-Lösung die beste. Nach meinen Tests arbeiten alle Programme zuverlässig und geben keine falschen Daten aus. Bei der EXCEL-Lösung muss man nur die Grafiken richtig interpretieren.

## 3 Geldschöpfung

### 3.1 Überblick

Unter Geldschöpfung versteht man den Vorgang, bei dem neues Geld in den Umlauf gebracht wird. Hier wird nur die Geldschöpfung der Kreditinstitute behandelt. Diese schöpfen Geld, indem sie Kredite vergeben. Es stellt sich dann die Frage, wie viele Kredite können in einem Bankensystem aus einer bestimmten Einlage vergeben werden. Die Summe der Kredite hängt von zwei Faktoren ab: Den Reservesatz und der Bargeldabflussquote im Bankensystem.

### 3.2 Möglichkeiten der Rationalisierung

Um die maximale Geldschöpfungsmöglichkeit im Bankensystem zu berechnen, gibt es eine Formel. Diese Berechnung kann der Computer machen, da man sich relativ leicht verrechnen kann. Außerdem ist es interessant die Entwicklung der Geldschöpfung zu verfolgen. Dazu fertigt man eine Tabelle an und rechnet alle Werte für jede Bank neu aus. Das ist sehr zeitaufwendig, mit einer Schleife<sup>8</sup> kann der Computer die Entwicklung der Geldschöpfung für beliebig viele Banken schnell und effektiv berechnen.

Der Benutzer muss nur die Startparameter dem Programm mitteilen und der Rest wird dann vom Computer berechnet. So kann man sich schnell die Auswirkungen verschiedener Reservesätze und Bargeldabflussquoten verdeutlichen.

### 3.3 Berechnung der Entwicklung der Geldschöpfung

Alle drei Lösungen arbeiten nach dem folgenden Prinzip. Deshalb werden die Algorithmen nicht noch einmal beschrieben. Es wird vorausgesetzt, dass alle Prozentzahlen als Dezimalzahl vorliegen.

Für die Berechnung benötigt man die Überschussreserve (*res*) oder die erste Einlage (*E*) sowie den Reservesatz (*r*) und die Bargeldabflussquote (*c*). Die Höhe der ersten Einlage wird nach folgender Formel berechnet:  $E = res * (1 - r)$ .

Wenn dann Einlage vorhanden ist, wird eine Schleife abgearbeitet. Diese berechnet nun die Barreserve (*bar*). Diese muss die Bank einbehalten und darf sie nicht weiter als Kredit vergeben. Die Formel lautet:  $bar = E * r$ . Die Überschussreserve ist gleich der maximalen Höhe der Kredite, die die Bank vergeben kann. Sie berechnet sich so:  $res = E - bar$ . Der Bargeldabfluss (*ab*) zeigt an wie viel Geld den Kreditnehmern bar ausgezahlt werden. Man berechnet den Bargeldabfluss:  $ab = res * c$ . Die Einlage der nächsten Bank im Bankensystem berechnet sich nach folgender Formel:  $E = res - bar$ . Die eben beschriebenen Schritte werden solange wiederholt wie es Banken im Bankensystem gibt.

Es gibt auch die Möglichkeit, die maximale Geldschöpfungsmöglichkeit ( $G_{max}$ ) im Bankensystem zu berechnen. Dieser Wert ist von großer Bedeutung. Er zeigt an wie viel Geld

aus einer bestimmten Einlage geschöpft werden kann. Die Formel lautet:  $G_{max} = \frac{res}{r + c * (1 - r)}$

### **3.4 Die C++- und Visual Basic-Lösung**

Die beiden Lösungen arbeiten nach dem gleichen Prinzip, deshalb wird es nur einmal erklärt.

Als erstes wird der Inhalt der Eingabefelder abgefragt und auf dezimale Zahlenwerte geprüft. Nun folgt die im Punkt 3.3 beschriebene Schleife. In ihr befinden sich außerdem die Ausgaberroutinen für die verschiedenen Werte. Diese werden sofort nach der Berechnung ausgegeben. Nachdem die Schleife abgearbeitet ist, werden noch die restlichen Werte, wie die maximale Geldschöpfung im Bankensystem und der Geldschöpfungsmultiplikator ausgegeben.

### **3.5 Funktionsweise der Visual Basic Lösung**

Der Programmablauf ist sehr einfach und ich möchte an dieser Stelle den Ablauf eines Programms erläutern. Um die Erklärungen besser verfolgen zu können, ist der Quellcode (Geldschöpfung) im Anhang mit Zeilennummern versehen.

Wenn man auf den Button „Berechnen“ klickt, wird der Programmteil von Zeile 1 bis Zeile 114 abgearbeitet. Er kann auch vorher verlassen werden, durch den Befehl *Exit Sub* wie man ihn z.B. in Zeile 21 findet.

In den Zeilen 1 – 13 werden dem Programm alle Speicherplätze mitgeteilt, welche es einzurichten hat.

Damit das Programm im Falle eines Programmfehlers eine Fehlermeldung ausgibt, springt es durch den Befehl in Zeile 16 zur Zeile 111 und gibt eine Fehlermeldung aus. Diese ist durch die Zeilen 112 und 113 definiert.

In den Zeilen 18 – 55 werden die Eingabefelder ausgelesen und sie werden auf ihren Inhalt hin geprüft. So wird sichergestellt, dass alle nötigen Werte für den Start der Berechnung vorliegen und nur Zahlen eingegeben sind. Aus programmiertechnischen Gründen werden in den Zeilen 35 – 37 die Einlage berechnet.

Der Tabellenkopf wird in Zeile 55 und 56 erstellt und ausgegeben.

Nun folgt der eigentliche Rechenteil des Programms. In den Zeilen 63 und 64 werden die Prozentwerte in Dezimalzahlen umgerechnet. Dadurch wird der spätere Rechenaufwand geringer. In Zeile 67 wird der Geldschöpfungsmultiplikator und in Zeile 70 die maximale Geldschöpfung berechnet. Wie schon erwähnt wird die Entwicklung der Geldschöpfung mit einer Schleife berechnet. Diese umspannt die Zeilen 73 – 93. Die Werte für Reserve, Überschussreserve, Kredit und Bargeldabfluss für die jeweilige Bank werden in den Zeilen 74 – 76 berechnet (siehe 3.3). Die Ergebnisse werden in den Zeilen 80 – 85 formatiert, sodass die Ausgabe gut aussieht. Diese ist in den Zeilen 87 – 90 programmiert. Die Einlage für die nächste Bank wird in Zeile 92 berechnet. Die Schleife wird solange wiederholt, wie es Banken im Bankensystem geben soll. Diese Anzahl hat der Benutzer bei der Eingabe dem Programm mitgeteilt.

Es folgt nun in den Zeilen 95 – 108 die weitere Ausgabe von den Geldschöpfungsmultiplikator und der maximalen Geldschöpfung. In der Zeile 110 wird der Programmteil verlassen. Das Programm kann nun weitere Eingaben vom Benutzer verarbeiten.

Was ich hier beschrieben habe, ist nur der „Rechenteil“ des Programms. Die anderen Funktionen können Sie im Source-Code, welcher sich auf der CD-ROM befindet, nachvollziehen.

### **3.6 Die EXCEL-Lösung**

Diese Lösung arbeitet genau so, als wenn man das Problem von Hand löst. Man gibt alle gegebenen Werte in die entsprechenden Zellen ein. Nun werden die Werte nach den schon beschriebenen Formeln berechnet. Zuerst wird die erste Einlage berechnet, wenn sie nicht gegeben wurde, danach werden die Reserve, die Überschussreserve, der Kredit und der Bargeldabfluss berechnet. Das ganze wird in den nächsten 7 Zeilen wiederholt, das ist keine Schleife, sondern alles einzelne Berechnungen.

In der Zeile Bank x werden die Summen im Bankensystem angezeigt, wobei x eine sehr große Zahl darstellt.

Wie schon in den anderen EXCEL-Lösungen habe ich ein Makro programmiert, welches die Eingabe löscht. Es wird abgearbeitet, wenn man den Button „Neu“ betätigt.

### **3.7 Beurteilung der Lösungen**

Diese drei Lösungen kann man am besten miteinander vergleichen, denn sie arbeiten nach dem gleichen Prinzip.

Mein Favorit ist die Visual Basic-Lösung. Sie vereint bei dieser Betrachtung alles, was eine gute Lösung ausmacht. Durch die geringe Komplexität der Aufgabenstellung an das Programm ist der Quellcode sehr leicht zu verstehen. Man muss lediglich die Syntax kennen, um das Programm zu warten und nachvollziehen zu können. Die Bedienbarkeit ist durch das von Windows übernommene Bedienkonzept sehr einfach. Deshalb wurde eine hohe Benutzerfreundlichkeit erzielt. Einen Kritikpunkt an der Bedienbarkeit gibt es aber, wenn man die erste Einlage bzw. die Überschussreserve eingibt, wird das andere Eingabefeld nicht gesperrt. Darauf habe ich verzichtet, damit auch alle Funktionen des Programms nachvollziehbar sind. Es werden auch alle Eingabefehler abgefangen. Dadurch kann das Programm nicht durch Fehleingaben abstürzen.

Die C++-Lösung ist auch gut, aber sie hat wie alle C++-Lösungen die Schwäche der mangelnden Wartbarkeit, durch die Integration in das BUVWL-Programm. Die Wartbarkeit hat sich aber im Gegensatz zu den anderen Lösungen deutlich verbessert. Denn dieser Programmteil benutzt das Programmsystem nur für die Kommunikation mit Windows. Die schwierige Wartbarkeit resultiert hier auf den komplizierten Methoden, um die Ein- bzw. Ausgabefelder anzusprechen. Ein Programmierer, welcher die Sprache kennt, müsste sich sehr schnell zurecht finden, da der Quellcode sehr überschaubar ist. Die Oberfläche ist sehr gut. Denn man kann keine Einlage eingeben, wenn man schon eine Überschussreserve eingeben hat und umgekehrt. Das sorgt für eine hohe Benutzerfreundlichkeit. Durch dieses Extra ist der Source-Code an dieser Stelle schwerer zu verstehen.

Die EXCEL-Lösung hat einige Nachteile. Ein eventueller Fehler ist schwer zu finden, da es jede Formel 8-mal gibt. Wie bei den anderen EXCEL-Lösungen ist hier auch das Manko der mangelnden Analyse der Eingabe und der daraus resultierenden Fehler. Ein Schüler wird sich schnell in der Lösung zurecht finden, weil sie exakt den Rechenweg benutzt, wie man ihn in der Schule vermittelt bekommt. Die Benutzerfreundlichkeit wurde mit den schon bekannten Neu-Button erhöht und die farbliche Gestaltung der Tabelle trägt auch dazu bei. Diese Lösung arbeitet auch wie die anderen sehr zuverlässig.

## 4 Vergleich der Lösungen

### 4.1 Portierbarkeit

Die beste Portierbarkeit haben die EXCEL-Lösungen. Die Tabellenkalkulation EXCEL gibt es für alle Microsoftbetriebsysteme ab Windows 95 und für den Apple Macintosh PC. Die Lösungen sind auch unter allen LINUX-Distributionen und Solaris brauchbar. Denn das Officepaket StarOffice kann auch die EXCEL-Dateien lesen und ist für diese Betriebssysteme verfügbar.

Mit den Visual Basic-Lösungen sieht es da schon schlechter aus. Wenn man sie auf den Apple Macintosh PC benutzen möchte, muss man sie teilweise anpassen. Eine Portierung auf die Betriebssysteme LINUX, UNIX, Solaris, BSD usw. ist nicht möglich. Da es keine Visual Basic Compiler für diese Systeme gibt.

Die C++-Lösung hat die beste Portierbarkeit, da es für jedes mir bekannte Betriebssystem einen Compiler gibt. Da C++ von sich auch keinerlei Befehle für graphische Ausgabe oder die Erstellung von Fenstern hat, muss man sämtliche Ausgaberroutinen ändern und neu programmieren. Wenn alle Routinen, welche auf Windows abgestimmt sind entfernt werden, dann läuft das Programm auch auf einem Betriebssystem ohne graphische Oberfläche z.B MS-DOS.

### 4.2 Ressourcenverbrauch

Hier betrachte ich zwei Dinge: Hauptspeicherverbrauch und Dateigröße. Diese Größen sind in der heutigen Zeit zu vernachlässigen. Bei ausgelasteten und älteren Systemen sind diese Werte aber noch interessant.

#### Dateigröße

Die alle Visual Basic-Lösungen zusammen verbrauchen ca. 102kB Speicherplatz und haben so auf dem ersten Blick den geringsten Speicherverbrauch. Jedes Visual Basic-Programm benötigt zusätzlich noch Bibliotheken<sup>9</sup>. Diese verbrauchen noch einmal ca. 300kB Speicher. Diese sollten aber auf den meisten Rechnern schon vorhanden sein. Deshalb werte ich ihre Größe nicht.

Die EXCEL-Lösungen verbrauchen ca. 213kB Speicherplatz. Man benötigt aber noch eine Tabellenkalkulation. Diese fällt mit ca. 80MB noch ins Gewicht. Man kann die Tabellenkalkulation auch für andere Zwecke benutzen als die hier gelösten Probleme. Eine Tabellenkalkulation sollte auf keinen Rechner fehlen, deshalb betrachte ich ihre Größe auch nicht.

Die C++-Lösung verschlingt mit ca. 381kB den meisten Speicherplatz. Dieser hohe Verbrauch lässt dadurch erklären, dass alle benötigten Bibliotheken in dem Programm enthalten sind. Den Vorteil dieses Konzeptes erläutere ich an anderer Stelle. Wenn man das BVWL-Programm benutzt, benötigt man nur das Programm und keine weiteren Dateien oder Bibliotheken.

### Hauptspeicherverbrauch

Ich bin zu dem Ergebnis gekommen, dass die C++-Lösung den geringsten Verbrauch mit 840kB hat. Das resultiert aus der hardwarenahen Programmierung mit C++ und auf den Verzicht von Bibliotheken

Die Visual Basic-Lösungen benötigen ca. 1088kB pro Lösung, wenn man alle Lösungen parallel benutzt, werden ca. 3264kB Speicher verbraucht. Dieser relativ hohe Wert resultiert daher, dass man mit Visual Basic nicht systemnah programmieren kann. Ein weiterer Grund sind die Bibliotheken, die jedes Visual Basic-Programm benutzen muss. Dort sind auch Informationen, die meine Programme nicht benötigen, enthalten. Diese verbrauchen aber auch Speicher.

Bei den EXCEL-Lösungen steigt der Speicherverbrauch nicht im gleichen Verhältnis zur Anzahl der geöffneten Tabellen. Das kann man damit erklären, dass die Tabellenkalkulation nur einmal Platz im Hauptspeicher verbraucht und dann kommt noch der Speicher-verbrauch der Tabellen hinzu. Wenn man alle drei Tabellen zur gleichen Zeit geöffnet hat, werden ca. 6522kB Speicher verbraucht, eine Tabelle hingegen benötigt ca. 5756kB Speicher.

## **4.3 Korrektheit**

Die Korrektheit ist die Fähigkeit von Softwareprodukten, die definierten Anforderungen exakt zu erfüllen.

Nach meinen Beispielaufgaben rechnen alle Lösungen richtig, d.h. alle Lösungen sind auf dem Gebiet der Korrektheit gleich gut.

## **4.4 Robustheit**

Die Robustheit ist die Fähigkeit von Softwaresystemen, auch unter außergewöhnlichen Bedingungen zu funktionieren.

Hier schneidet das BVWL-Programm am besten ab. Ich habe es nicht geschafft, es zum Absturz zu bringen. Es wurden alle Fehler abgefangen. Auf fremden Systemen gib es auch keine Probleme. Das erklärt sich dadurch, dass keine Bibliotheken benutzt werden. Denn diese sind sehr fehleranfällig, verschiedene Versionen von Bibliotheken führen auch zu Fehlern. Aus den Grund der Robustheit habe ich deshalb auf Bibliotheken verzichtet.

Diese sind aber der Grund, weshalb die Visual Basic-Lösungen hier schlechter abschneiden. Die nötigen Bibliotheken sind nicht immer vorhanden und es kann Probleme mit verschiedenen Versionen von ihnen geben. Es werden alle Eingabefehler abgefangen, sodass das Programm nicht durch Fehleingaben abstürzt.

Die EXCEL-Lösungen haben die geringste Robustheit. Es kommt aus Gründen, die mir fremd sind, schon mal vor, dass EXCEL abstürzt oder sich aufhängt. Dieser Fakt ist nicht gerade förderlich für eine gute Bewertung. Durch Fehleingaben in den Zellen der Tabellen stürzt EXCEL eigentlich nicht ab, mir ist es jedenfalls nicht bekannt.

## **4.5 Erweiterbarkeit**

Die Visual Basic- und EXCEL-Lösungen erfüllen dieses Kriterium in keiner Weise. Denn sie wurden als Tools konzipiert und sind nicht erweiterbar.

Das BVWL-Programm ist durch seinen modularen Aufbau sehr gut erweiterbar. An dem in Kapitel III beschriebenen Teil sind alle Funktionen angehängt. So kann man beliebig viele Module an das Grundgerüst anhängen. Wenn man zu viele Funktionen in das Programm einbaut, wird es unübersichtlich.

## **4.6 Wiederverwendbarkeit**

Aus den EXCEL-Lösungen kann man nichts für andere Programme wieder verwenden, denn alle Funktionen sind auf die Tabellen abgestimmt.

Bei den Visual Basic Lösungen ist es etwas besser. Den großen Teil der Funktionen kann man aber nicht für die Lösung anderer Probleme verwenden.

Durch den modularen Aufbau des BVWL-Programms kann man sehr viele Funktionen wieder verwenden. Man kann alle Funktionen der Datei *funktionen.cpp* für völlig andere Sachverhalte benutzen mit der *CBUVWLView::Determinaten-Funktion* kann man auch mathematische Probleme lösen. Die gute Wiederverwendbarkeit von Funktionen resultiert mit daraus, dass man in C++ sehr viel selbst programmieren muss. In anderen Sprachen gibt es dafür schon fertige Funktionen.

## **4.7 Integrität**

Integrität ist die Fähigkeit von Software, wie gut ihre verschiedenen Komponenten gegen unberechtigte Zugriffe und Veränderungen geschützt sind.

Die EXCEL-Lösungen schneiden hier am schlechtesten ab. Denn in EXCEL gibt es nur die Möglichkeit die Arbeitsmappe und die Tabellen mit einem Passwort zu schützen. Dieses kann man aber mit ein bisschen Arbeit und den passenden Tools herausfinden.

Bei den Visual Basic Lösungen ist es besser. Man kann nur alle Bezeichnungen im Programm relativ schnell ändern. Dazu benötigt man nur einen normalen Editor. Für Änderungen an den Formeln muss man wie im BVWL-Programm den Maschinencode<sup>10</sup> des Programms ändern, was nur wenige können und sehr arbeitsintensiv ist.

Das BVWL-Programm ist gegen schnelle Änderungen geschützt. Für Änderungen jeglicher Art benötigt man Programme, die sehr schwer zu bedienen sind, da man den Maschinencode verändern muss.

## **4.8 Arbeitsaufwand**

Am schnellsten habe ich die EXCEL-Lösungen erstellt. Das kommt daher, weil man sich nur um das Problem kümmern muss und die, in der Schule vermittelten Arbeitsschritte fast immer übernehmen kann.

Die Visual Basic-Lösungen haben auch einen relativ geringen Arbeitsaufwand. Für die

Programmierung des reinen Problems habe ich ca. eine Viertelstunde jeweils benötigt. Das ganze „Drumherum“, wie Abfangen von Eingabefehlern und Gestaltung der Oberfläche, hat dann noch einmal sehr viel Zeit in Anspruch genommen.

Am längsten habe ich für das BWL-Programm benötigt. Das kommt daher, dass man sich in C++ um Dinge kümmern muss, die in Visual Basic automatisch funktionieren. Der ganze Aufwand lohnt sich allerdings, denn ich konnte dadurch die Oberfläche nach meinen Ideen anpassen.

## **5 Demonstration der Elastizität**

### **5.1 Überblick**

Diese Funktion ist in das BVWL-Programm integriert. Mit ihr kann man den Schülern den Änderung Graphen bei sich änderter Elastizität zeigen. Dieses Werkzeug kann nichts berechnen, alles was man zu tun hat, ist die Bedienung eines Schiebereglers.

### **5.2 Funktionsweise**

Das Programm hat unveränderliche Startwerte, da es bei einer Demonstration nicht darauf ankommt.

In der Startposition steht der Graph  $45^\circ$  zur Geraden  $g: y = 50$ . Diese ist der Bezug für die weiteren Berechnungen. Mit den Schieberegler wählt man den Winkel zwischen  $g:$  und den Graphen. Das kann man machen, weil die Elastizität vom Winkel zwischen Graph und  $g:$  abhängt.

Weil die Startwerte vorgegeben sind, wird der Graph immer im Punkt  $(50;50)$  gedreht.

Wenn man den Graphen in diesen Punkt mit den Startwerten rotieren lässt, beschreibt er einen Kreis. Dieser hat einen Radius von  $\sqrt{1800}$ , mit diesem und dem Winkel kann man die Schnittpunkte des Graphen und mit dem Kreis berechnen. Aus diesen Werten berechnet sich dann die Elastizität nach der Formel aus Kapitel 1.

Anschließend wird der Graph im Koordinatensystem dargestellt.

### **5.3 Die Oberfläche**

Dieses Tool ist in einen separaten Dialog untergebracht. Sie können es getrennt von dem Hauptprogramm bewegen. Auf dem Dialog findet man fünf Ausgabefelder mit Zahlen. Alle Ausgabefelder sind beschriftet. Der Regler unten im Dialog regelt die Elastizität. In den Ausgabefeldern kann man dann die veränderten Werte sehen. Die graphische Ausgabe erfolgt zeitgleich im Hauptprogramm.

Um das Tool zu schließen, muss man nur die Schaltfläche „Schließen“ betätigen.

### **5.4 Bewertung**

Dieses Problem habe ich nur mit C++ gelöst, mit den anderen Mitteln wäre es zu aufwendig. Deshalb ist dieses Tool nicht die Bewertung in Kapitel 4 und 6 eingeflossen. Die Bedienbarkeit ist sehr gut, denn es gibt nur ein Bedienelement, den Schieberegler. Bei der Bedienung dieses Tools kann man also keine Fehler machen. Die Wartbarkeit ist auch gut, denn alle Routinen sind in einer Datei untergebracht. Die Ausgabe erfolgt aber in der Datei *BUVWLView.cpp*. Der Teil, welcher für diese Ausgabe verantwortlich ist, ist aber auch sehr klein und übersichtlich.

## **6 Einschätzung**

Ich finde das BVWL-Programm am besten, denn man hat alle Werkzeuge in einem Programm und kann schnell zwischen ihnen wechseln. Gut finde ich auch die graphische Ausgabe bei dieser Lösung. Dadurch kann man sie sehr anschaulich im Unterricht einsetzen. Bei den anderen Lösungen muss man für eine andere Aufgabe auch ein anderes Tool bzw. Arbeitsmappe benutzen. Das finde ich umständlich.

Wenn man die Lösungen im Rahmen des Informatikunterrichts benutzen möchte, sind die Visual Basic- und EXCEL-Lösungen vorzuziehen. Aus zwei Gründen: An unserer Schule wird mit C++ nicht gearbeitet und der Arbeitsaufwand zum Nachprogrammieren bzw. Verstehen des kompletten Programms steht in keinem Verhältnis zum Nutzen.

Die Berechtigung haben alle Lösungen, denn sie haben teilweise verschiedene Lösungsansätze. Man kann dadurch ein breites Spektrum an Aufgaben rechnen. Vorteilhaft der Programme ist die Rechengeschwindigkeit, so kann der Lehrer ein falsches Ergebnis, das durch falsche Startwerte herauskommt, sehr schnell überprüfen. Es ist auch möglich, dass jeder Schüler eine andere Aufgabe bekommt. Die verschiedenen Aufgaben kann man abspeichern und dann mit den Schülern vergleichen.

## IV Quellenverzeichnis

### 1. Bücher

Stroustrup, Bjarne: Die C++ Programmiersprache, 4. aktualisierte Aufl., München; Boston; San Francisco; Harlow; Don Mills; Sydney; Mexico City; Madrid; Amsterdam (Addison-Wesley) 2000

Louis, Dirk: Jetzt lerne ich C++ 6, 1. Aufl., München (Markt und Technik) 1999

Louis, Dirk / Toth Viktor: Visual C++ 6, Profihandbuch, 1. Aufl., (Markt+Technik Verlag München) 1999

Spona, Helma: Visual Basic 6; Das bhv Taschenbuch, 1. Aufl., Kaarst (verlag moderne industrie Buch) 2001

Böhm, Oliver: C++ echt einfach, Poing (Franzis Verlag GmbH) 2000

Huttary, Rudolf: visual basic6 referenz; new technology, 1. Aufl., München (Markt+Technik Verlag) 2000

### 2. Digitale Medien

Visual C++-Dokumentation: Version 3.01.0.8043, Microsoft Corporation

QBASIC Hilfssystem, Version 1.1, Microsoft Corporation

Willms, André: Die C++-Programmierung, Programmiersprache, Programmieretechnik, Datenstrukturen

# V Anhang

## 1. Die fkt::double chardouble-Funktion

Diese Funktion wandelt ein Komma in einen Dezimalpunkt um.

```
double fkt::chardouble(char zahl_text[12])
{
    char ausg[12], ziffer;
    double zahl_dou;
    int laenge_vor, laenge, zahl_int, x;

    if (strlen(zahl_text)==0) // Leere Eingabefelder abfangen
    {
        zahl_dou=0;
    }
    else
    {
        // Ziffern vor dem Komma isolieren und in ausg schreiben
        zahl_dou=atof(zahl_text);
        zahl_int=(int) (zahl_dou);
        sprintf(ausg,"%d", zahl_int);

        // Dezimalpunkt hinzufügen
        strcat(ausg, ".");

        // Längen der Strings bestimmen
        laenge_vor=strlen(ausg);
        laenge=strlen(zahl_text);

        for(x=laenge_vor;x<laenge;x++)
        {
            // Isolieren der Ziffern nach dem Komma
            ziffer=zahl_text[x];
            // Bau der neuen Zahl
            ausg[x]=ziffer;
        }
        // Konvertierung des String in Double
        zahl_dou=atof(ausg);
    }
    return zahl_dou;
}
```

## 2. Die CBUVWLView::Determinaten-Funktion

Diese Funktion berechnet den Schnittpunkt von zwei Vektoren. Damit die Funktion auch einen logisch wahren Wert zurück gibt, muss sich der Schnittpunkt der Vektoren auch noch zwischen den gegebenen Punkten befinden.

```
bool CBUVWLView::Determinate(double *wert, double P[4][2])
{
    double D, D_x, a1, a2, b1, b2, c1, c2, vektor[2], feld[4];

    // Koeffizienten bilden ->Vektorgleichungssystem
    a1=P[1][0]-P[0][0];
    a2=P[1][1]-P[0][1];
    b1=(P[3][0]-P[2][0])*-1;
    b2=(P[3][1]-P[2][1])*-1;

    // Absolutglieder bilden ->Vektorgleichungssystem
    c1=P[2][0]-P[0][0];
    c2=P[2][1]-P[0][1];

    // Hauptdeterminante
    D=(a1*b2)-(a2*b1);
    // Neberdeterminante
    D_x=(c1*b2)-(c2*b1);

    // Vektor berechnen
    vektor[0]=P[0][0]+((D_x/D)*a1);
    vektor[1]=P[0][1]+((D_x/D)*a2);

    // Größten und kleinsten x-Wert finden
    feld[0]=P[0][0];
    feld[1]=P[1][0];
    feld[2]=P[2][0];
    feld[3]=P[3][0];
    fkt::sortieren(4, feld);
    // Überprüfen ob der Vektor zwischen den Punkten ist
    if((feld[3]>=vektor[0])&&(feld[2]<=vektor[0]))
    {
        // Größten und kleinsten y-Wert finden
        feld[0]=P[0][1];
        feld[1]=P[1][1];
        feld[2]=P[2][1];
        feld[3]=P[3][1];
        fkt::sortieren(4, feld);
        // Überprüfen ob der Vektor zwischen den Punkten ist
        if((feld[3]>=vektor[1])&&(feld[2]<=vektor[1]))
        {
            wert[0]=vektor[0];
            wert[1]=vektor[1];
            return 1;
        }
    }
}
return 0;}
```

### 3. Geldschöpfung

Dieser Programmteil liest die Eingabewerte ein und berechnet die Tabelle für die Entwicklung der Geldschöpfung. Er ist aus den Visual Basic Programm Geldschöpfung.

```
1 Private Sub calc_Click()
2     Dim einlage As Double, eT As String * 9 'Einlage im Bankensystem
3     Dim r As Double      'Reservesatz im Bankensystem
4     Dim reserve As Double, rT As String * 9 'Überschussreserve
5     Dim anz_b As Single  'Anzahl der Banken im Bankensystem
6     Dim c As Double     'Bargeldabflussquote
7     Dim ausg As String  'Ausgabezeile
8     Dim n As Single, nT As String * 3      'Zählvariable
9     Dim abfluss As Double, aT As String * 9 'Bargeldabfluss
10    Dim bar As Double, bT As String * 9     'Barreserve
11
12    Dim f As String      'Format für Ausgabe
13    Dim m, G As Double
14
15    'Einlesen der Daten
16    On Error GoTo Fehler
17
18    If (Me.r_satz.Text = "") Then
19        MsgBox "Sie müssen einen Reservesatz angeben!", vbExclamation + vbOKOnly, "Fehler - kein Wert"
20        Exit Sub
21    Else
22        r = Me.r_satz.Text
23    End If
24
25
26    If (Me.banken.Text = "") Then
27        MsgBox "Sie müssen die Anzahl der Banken angeben!", vbExclamation + vbOKOnly, "Fehler - kein Wert"
28        Exit Sub
29    Else
30        anz_b = Me.banken.Text
31    End If
32
33
34    If (Me.einlage.Text = "") Then
35        If Not Me.reserve.Text = "" Then
36            r = Me.reserve.Text
37            einlage = reserve * 100 / (100 - r)
38        Else
39            MsgBox "Sie müssen einen Wert für Überschussreserve oder Einlage angeben!", vbExclamation + vbOKOnly, "Fehler - kein Wert"
40            Exit Sub
41        End If
42    Else
43        einlage = Me.einlage.Text
44        reserve = einlage * (100 - r) / 100
45    End If
46
47
48
```

```
49   If (Me.bargeld.Text = "") Then
50       c = 0
51   Else
52       c = Me.bargeld.Text
53   End If
54   ausg = "Bank   Einlage   Reseve   Überschuss   Kredit   Abfluss"
55   Me.ausg.AddItem ausg
56
57
58
59
60 'Rechenteil
61
62   ' r und c in Zahlen umwandeln
63   r = r / 100
64   c = c / 100
65
66   'Multiplikator
67   m = 1 / (r + c * (1 - r))
68
69   'max. Geldschöpfung
70   G = reserve * m
71
72
73   For n = 1 To anz_b Step 1
74       bar = einlage * r
75       reserve = einlage - bar
76       abfluss = reserve * c
77
78
79       'Ausgabe
80       f = "#####0.00"
81       eT = Format(einlage, f)
82       rT = Format(reserve, f)
83       aT = Format(abfluss, f)
84       bT = Format(bar, f)
85       nT = Format(n, "##0")
86
87       ausg = nT & "   " & eT & "   " & bT & "   " & rT & "   " &
88           & rT & "   " & aT
89       Me.ausg.AddItem ausg
90
91       einlage = reserve - abfluss
92   Next n
93
94
95   'Leerzeilen einfügen
96   ausg = ""
97   Me.ausg.AddItem ausg
98
99   ausg = "Geldschöpfungsmultiplikator: " & m
100  Me.ausg.AddItem ausg
101
102  ausg = "Gesamte Geldschöpfung: " & Format(G, "#.00")
103  Me.ausg.AddItem ausg
104
105  'Leerzeilen einfügen
```

```
106     ausg = ""
107     Me.ausg.AddItem ausg
108     Me.ausg.AddItem ausg
109
110     Exit Sub
111 Fehler:
112     MsgBox "Sie dürfen nur Zahlen eingeben", vbExclamation + vbQuestion
113         vbOKOnly, "Fehler - keine Zahl"
114 End Sub
```

## VI Kurzfassung

Diese besondere Lernleistung behandelt folgende volkswirtschaftliche Probleme: Elastizitätsrechnung, Ermittlung des Cournotschen Punkt und Geldschöpfung. Mit dieser Lernleistung zeige ich, wie man diese Probleme rechentechnisch lösen kann.

Es wird beschrieben, wie die Probleme mit C++ und Visual Basic programmiert werden sowie die Lösung mit der Tabellenkalkulation ECXEL.

Die Elastizität zeigt an, wie die Nachfrage auf Änderungen des Preises bzw. der Menge reagiert. Die Darstellung umfasst die allgemeinen Möglichkeiten der Rationalisierung und die Beschreibungen der jeweiligen Lösungen. Im Anschluss werden die Lösungen verglichen und ich bin zu den Entschluss gekommen, dass die Visual Basic-Lösung die Beste ist. Das nächste Kapitel beschäftigt sich mit der Ermittlung des Cournotsche Punkt. Er zeigt die für den Monopolisten die gewinnmaximierende Preis-Mengenkombination. Die Lösung dieses Problems ist recht vielschichtig, denn es gibt zwei grundlegend verschiedene Verfahren zur Berechnung. Das ist das graphische und das rechnerische Verfahren. Das Rechnerische habe ich mit Visual Basic gelöst. Bei der graphischen Lösung mit C++ habe ich ein eigenes Verfahren zur Berechnung des Cournotschen Punkt entwickelt. Diese Lösung ist nach meiner Bewertung die beste.

Als nächstes folgt die Geldschöpfung. Bei diesen Problem wird die Entwicklung der Geldschöpfung im Bankensystem dargestellt. Alle drei Lösungen arbeiten nach dem gleichen Prinzip, deshalb sind sie gut zu vergleichen. Ich habe außerdem in diesen Kapitel den genauen Ablauf einer Programmfunktion aus Visual Basic erläutert. Dadurch erhält man einen Einblick in die Funktionsweise eines Programms. Die Visual Basic-Lösung hat bei der Bewertung am Besten abgeschnitten.

Im Kapitel 4 werden die Lösungen unter verschiedenen Gesichtspunkten bewertet. Dieses Kapitel ist mit dem 6. in Zusammenhang zu sehen. Ich komme zu Schluss zu der Behauptung, dass das BVWL-Programm am Besten ist. Denn mit ihm kann man alle drei Probleme lösen.

Damit man die Elastizitätsrechnung mehr veranschaulichen kann, beschreibe ich im 5. Kapitel eine Funktion, die die Elastizitätsrechnung demonstriert. Diese Möglichkeit hat man nur in der C++-Lösung. Dieses Tool hat eine sehr einfache Bedienung.

## **VII Eidesstattliche Erklärung**

Ich, Michel Rennecke, erkläre hiermit an Eides statt, dass:

1. ich meine besondere Lernleistung eigenständig und ohne fremde Hilfe angefertigt habe;
2. ich die Übernahme wörtlicher Zitate aus der Literatur/Internet sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe;
3. ich meine besondere Lernleistung bei keiner anderen Prüfung vorgelegt habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

(Ort, Datum)

(Unterschrift)

- 1 a) Fähigkeit, Dateien in unterschiedlichen Programmen laden zu können (importieren/exportieren)  
b) Fähigkeit, Programme auf unterschiedlich Hardware und Betriebssysteme anzupassen
- 2 englische Bezeichnung für Werkzeug; also ein kleines Programm mit einem sehr begrenzten Funktionsumfang
- 3 auch Quell-Code; Originärer ASCII-Text, der ein Programm in einer höheren Programmiersprache darstellt
- 4 Platzhalter für veränderliche Werte; stellvertretend für eine Adresse im Hauptspeicher
- 5 Es sind mehrere Variablen zusammengefasst
- 6 Kurze Hilfstexte, die erscheinen, wenn man über ein Steuerelement mit der Maus überquert
- 7 Folge von Befehlen bzw. eine Kombination von Tasten- und Mausklicks (ein kleines „Programm“)
- 8 Arbeitet die gleiche(n) Funktion(en) hintereinander mehrmals ab
- 9 Beinhalten spezielle Programmroutinen, die erst aus einer sogenannten Programmbibliothek geladen werden, wenn diese auch benötigt werden. Dieses Verfahren spart Speicherplatz im Arbeitsspeicher.
- 10 Für den Prozessor erforderliche Darstellung von Befehlen im binären Zahlenformat.
- 11 Zeigt an, dass man den Inhalt der nächsten Zeile hier anhängen muss