Aufgabe 4: Nandu

Teilnahme-ID: 68772

Team-ID: 00083

Bearbeiter/-in dieser Aufgabe:
Linus Schumann

19. November 2023

Inhaltsverzeichnis

1 Lösungsidee

1.1 Analyse der verschiedenen Blöcken

Grundsätzlich besteht die Aufgabe darin, dass eine Vernetzung von verschiedenen Blöcken gegeben ist. Diese Blöcke sind jeweils 1x2 Felder groß und können in verschiedenen Ausführungen vorkommen. Folgende Blöcke sind dabei möglich:

1.1.1 Weißer Block

Dieser Block besitzt 2 Inputs bzw. Lichtsensoren und 2 Outputs bzw. Lampen. Dabei arbeitet der Block nach dem Prinzip eines NAND-Gatters. Das heißt, dass die Lampen nur dann nicht leuchten, wenn beide Lichtsensoren aktiviert sind. Ansonsten sind die Lampen ausgeschaltet. Für die Wahrheitstabelle gilt also folgendes:

Input 1	**Input 2**	**Output 1 bzw. 2**
0	0	1
0	1	1
1	0	1
1	1	0

Tabelle 1: Wahrheitstabelle des weißen Blocks

1.1.2 Roter Block

Dieser Block kann in zwei verschiedenen Ausführungen existieren. Die erste Ausführung besitzt nur einen Input links und die zweite Ausführung besitzt nur einen Input rechts. Beide Ausführungen besitzen jedoch jeweils zwei Outputs. Dabei arbeitet der Block nach dem Prinzip eines NOT-Gatters. Das heißt, dass die Lampen nur dann leuchten, wenn der Lichtsensor nicht aktiviert ist. Ansonsten sind die Lampen ausgeschaltet. Für die Wahrheitstabelle gilt also folgendes:

Input	**Output 1 bzw. 2**
0	1
1	0

Tabelle 2: Wahrheitstabelle des roten Blocks

1.1.3 Blauer Block

Dieser Block besitzt 2 Inputs bzw. Lichtsensoren und 2 Outputs bzw. Lampen. Dabei arbeitet der Block nach dem Prinzip eines OR-Gatters. Das heißt, dass die Lampen nur dann ausgeschaltet sind, wenn beide Lichtsensoren nicht aktiviert sind. Ansonsten sind die Lampen eingeschaltet. Für die Wahrheitstabelle gilt also folgendes:

Input 1	**Input 2**	**Output 1 bzw. 2**
0	0	0
0	1	1
1	0	1
1	1	1

Tabelle 3: Wahrheitstabelle des blauen Blocks

1.1.4 Weitere Blöcke

Weiter gibt es noch Lichtquellen und Lichtsensoren. Die nur zu Beginn bzw. zum Ende der Konstruktion zu finden sind.

1.2 Fragestellung

Generell wird nun für jede binäre Kombination der Lichtquellen eine Ausgabe der Lichtsensoren gesucht. Auch ist dabei noch wichtig zu beachten, dass das Licht nur von einer Lichtquelle zu einem Lichtsensor gelangen kann, wenn kein freier Platz zwischen den beiden ist.

1.3 Lösungsansatz

Der Ansatz zum Lösen dieser Aufgabe ist im Prinzip simpel. Zuerst müssen alle möglichen binären Kombinationen der Lichtquellen ermittelt werden. Dann muss für jede dieser Kombinationen die Ausgabe der Lichtsensoren ermittelt werden. Dazu kann einfach das Verhalten der Blöcke Stück für Stück simuliert werden. Dies kann aufgrund der Aufgabe dabei Zeile für Zeile erfolgen. Am Schluss ergeben sich somit die gesuchten Ausgaben an der Lichtsensoren. Diese können dann wieder als binäre Kombinationen interpretiert werden und somit die Lösung der Aufgabe darstellen. Es kann dann eine Tabelle erstellt werden, die für jede binäre Kombination der Lichtquellen die entsprechende binäre Kombination der Lichtsensoren enthält.

2 Umsetzung

2.1 Allgemeines

Im Folgenden wird die Umsetzung der in Abschnitt 1 beschriebene Lösungsidee, näher erläutert. Grundsätzlich wurde diese Idee dabei in C++, genauer gesagt in der Datei "Aufgabe_4.cpp" implementiert. Diese Datei befindet sich im Verzeichnis "./source/".

Um das implementierte Programm zu starten, kann das Batch Skript "Aufgabe_4.bat" im Verzeichnis "./executables/" genutzt werden. Dieses startet das von mir für Windows kompilierte Programm ("Aufgabe_4.exe"). Für andere Betriebssysteme müsste die Source-Datei erneut auf dem entsprechenden Rechner kompiliert werden.

Im Verzeichnis "./beispieleingaben/" befinden sich alle in dieser Dokumentation aufgeführten Beispiele und im Verzeichnis "./beispielausgaben/" befinden sich dementsprechend die gesicherten Ausgaben. Damit Letztere besser von den Beispieldaten unterschieden werden können, werden diese mit der Dateiendung "_out.txt" gespeichert.

2.2 Implementierung der Lösungsidee

Nun werden die einzelnen Bestandteile der Implementierung näher erläutert.

2.2.1 Main-Funktion

Die Main-Funktion ist der Einstiegspunkt des Programms. Sie liest zuerst den Namen der Input-Datei ein und öffnet diese. Dann wird diese Input-Datei mit der Funktion "readData()" eingelesen und der eigentliche Algorithmus kann starten. Dazu wird zuerst die Funktion "generateInputs()", dann die Funktion "calculateResults()" und schließlich die Funktion "printResults()" aufgerufen.

2.2.2 generateInputs-Funktion

Die Funktion "generateInputs()" nutzt die Funktion "generateCombinationsRecursiv()", um alle möglichen binären Kombinationen der Lichtquellen zu ermitteln. Dabei ist die Anzahl an Lichtquellen die Länge der binären Kombinationen. Alle möglichen binären Kombinationen werden dann in einem Vektor gespeichert.

2.2.3 calculateResults-Funktion

Die Funktion "calculateResults()" nutzt die Funktion "calculateResult()", um für jede binäre Kombination der Lichtquellen die entsprechende binäre Kombination der Lichtsensoren zu ermitteln. Dabei wird für jede binäre Kombination diese Funktion aufgerufen und das Ergebnis in einem Vektor gespeichert. Dieser Vektor wird dann zurückgegeben.
Um das Ergebnis zu berechnen wird die "states"-Matrix genutzt, in der für jedes Feld der aktuelle Zustand (0 oder 1 bzw. an oder aus) gespeichert wird. Nun wird zuerst die erste Zeile der "states"-Matrix

an den passenden Stellen auf die Werte der binären Kombination der Lichtquellen gesetzt. Dann wird die eingelesene "field"-Matrix Zeile für Zeile durchgegangen und dabei die "states"-Matrix immer weiter aktualisiert. Je nachdem welcher Block an der aktuellen Position ist, wird die "states"-Matrix entsprechend aktualisiert. Am Schluss werden dann die Werte der Lichtsensoren in einem Vektor gespeichert und wie oben geschrieben zurückgegeben.

2.2.4 printResults-Funktion

Die Funktion "printResults()" gibt die Ergebnisse in Form einer Tabelle wieder. Dabei wird einfach jeweils die binäre Kombination der Lichtquellen und die entsprechende binäre Kombination der Lichtsensoren nebeneinander ausgegeben.

2.3 Laufzeitanalyse

Da die Funktion "calculateResult()" jedes Feld einer 2-dimensionalen Matrix durchgeht, liegt die Laufzeit dieser Funktion in der Laufzeitklasse $O(n * m)$. Dabei ist n die Anzahl der Zeilen und m die Anzahl der Spalten der Eingabe-Matrix.
Auch muss noch die Laufzeit der Funktion "generateCombinations" betrachtet werden. Dabei stellt man fest, dass ein Binärbaum sich aufbaut und die Anzahl der Knoten in diesem Binärbaum 2^p entspricht. Dabei ist p die Anzahl der Lichtquellen. Da die Funktion "generateCombinationsRecursiv()" für jeden Knoten einmal aufgerufen wird, liegt die Laufzeit dieser Funktion in der Laufzeitklasse $O(2^p)$.
Da allerdings in den vorgegebenen Beispielen die Anzahl der Lichtquellen nie mehr als $p = 6$ beträgt, wird die Funktion "generateCombinationsRecursiv()" nie mehr als $sum_{n=0}^{6} 2^n = 126$ mal aufgerufen. Somit ist für diese Beispiele die Laufzeit dieser Funktion eigentlich irrelevant.
Somit liegt die Laufzeit des Programms (bei Betrachtung der gegebenen Beispiele) in der Laufzeitklasse $O(n * m)$.

2.4 Zusatzaufgabe

Auch die Zusatzaufgabe wurde von mir bearbeitet. Dazu habe ich mithilfe von "HTML", "CSS" und "JavaScript" eine WebApp erstellt. Diese WebApp kann im Browser geöffnet werden und ist zum aktuellen Zeitpunkt (19. November 2023) unter folgender Adresse zu finden: `https://meineseite.ddns.net/nandu/`.
Nach dem Einsendeschluss wird diese Adresse auch öffentlich zugänglich sein.
Da ich nicht garantieren kann, dass die WebApp auch dauerhaft unter dieser Adresse zu finden ist, ist die WebApp auch noch im Verzeichnis "./web-app/" zu finden.

2.4.1 Umsetzung der Zusatzaufgabe

Um die WebApp zu erstellen, habe ich zuerst eine HTML-Datei erstellt (index.html). Diese wird mithilfe der CSS-Datei (style.css) gestaltet. Die Funktionalitäten wurden mit JavaScript bzw. mithilfe der JavaScript-Bibliothek "konva.js" implementiert. Diese Bibliothek stellt Funktionen zur Verfügung, um mithilfe von JavaScript ein Canvas-Element mit verschiedenen Formen zu füllen. Auch kann hiermit ein Drag-and-drop Prinzip implementiert werden.

2.4.2 Funktionsweise der WebApp

Die WebApp besteht aus einer Navigationsleiste im oberen Bereich und einem Canvas-Element im unteren Bereich. In der Navigationsleiste kann eine Kombination direkt im Canvas ausgeführt oder auch downgeloadet werden. Auch kann eine vorkonfigurierte Kombination geladen werden.
Im Canvas Element kann man sich frei bewegen und mithilfe von Drag-and-drop verschiedene Kombinationen erstellen. Dabei können verschiedene Blöcke mit den Tasten 1-6 der Tastatur an der aktuellen Mausposition platziert werden. Auch können Blöcke mit der Maus verschoben werden. Genaueres zu dieser Steuerung lässt sich auch in der WebApp unter dem Punkt "help" nachlesen.
Auch ist noch wichtig zu erwähnen, dass in der WebApp die Kombination wie auf dem Aufgabenbogen von links nach rechts eingegeben und ausgeführt werden. Der Export bzw. Import von Dateien funktioniert allerdings, wie bei den Beispielen, von oben nach unten.

2.4.3 Lauffähigkeit auf mobilen Geräten

In normalen Browsern auf Desktop Computern funktioniert die WebApp einwandfrei. Auch auf mobilen Geräten funktioniert die WebApp, allerdings ist die Steuerung aufgrund der fehlenden Tastatur etwas eingeschränkt. Man könnte die WebApp also auch auf Tablet Computern nutzen, allerdings ist die Nutzung auf Smartphones oder Tablets ohne physische Tastatur nicht zu empfehlen.

3 Beispiele

Im Folgenden werden alle Ausgaben der Beispiele aufgeführt. Die Ausgaben sind dabei der Inhalt der Dateien im Verzeichnis "./beispielausgaben/". Bei Beispiel 1 und 2 wird außerdem einmal das Eingabebild, dass in der WebApp zu sehen ist, aufgeführt.
Die Beispiele 6 und 7 sind meine eigenen Beispiele und zeigen, was passiert, wenn nur ein Lichtsensor bzw. kein Lichtsensor vorhanden ist.

3.1 Beispiel 1

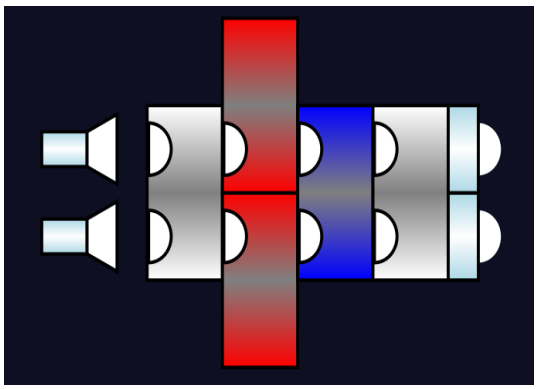

Abbildung 1: Eingabebild aus der WebApp zu Beispiel 1

```
Q1    Q2     | L1    L2
off   off    | on    on
off   on     | on    on
on    off    | on    on
on    on     | off   off
```

Listing 1: Ausgabe Beispiel 1

3.2 Beispiel 2

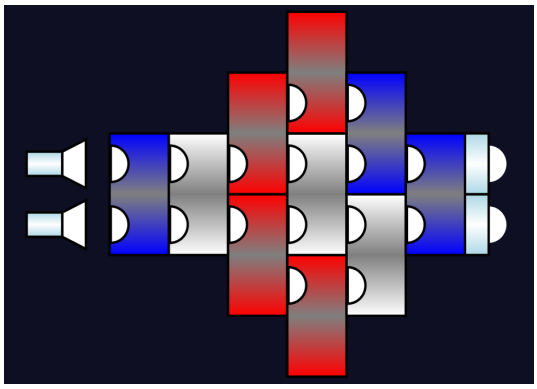

Abbildung 2: Eingabebild aus der WebApp zu Beispiel 2

```
Q1   Q2    | L1   L2
off  off   | on   on
off  on    | on   on
on   off   | on   on
on   on    | on   on
```

Listing 2: Ausgabe Beispiel 2

3.3 Beispiel 3

```
Q1   Q2   Q3    | L1   L2   L3   L4
off  off  off   | on   on   off  on
off  off  on    | on   on   off  on
off  on   off   | on   on   off  on
off  on   on    | on   on   on   on
on   off  off   | on   on   off  on
on   off  on    | on   on   on   on
on   on   off   | on   on   off  on
on   on   on    | on   on   on   on
```

Listing 3: Ausgabe Beispiel 3

3.4 Beispiel 4

```
Q1   Q2   Q3   Q4    | L1   L2
off  off  off  off   | off  off
off  off  off  on    | off  off
off  off  on   off   | on   on
off  off  on   on    | on   on
off  on   off  off   | on   on
off  on   off  on    | on   on
off  on   on   off   | on   on
off  on   on   on    | on   on
on   off  off  off   | off  off
on   off  off  on    | off  off
on   off  on   off   | off  off
on   off  on   on    | off  off
on   on   off  off   | off  off
on   on   off  on    | off  off
on   on   on   off   | off  off
on   on   on   on    | off  off
```

Listing 4: Ausgabe Beispiel 4

3.5 Beispiel 5

```
Q1   Q2   Q3   Q4   Q5   Q6    | L1   L2   L3   L4   L5
off  off  off  off  off  off   | off  on   on   on   off
off  off  off  off  off  on    | off  on   on   on   off
off  off  off  off  on   off   | off  on   on   on   on
off  off  off  off  on   on    | off  on   on   on   on
off  off  off  on   off  off   | off  on   on   on   off
off  off  off  on   off  on    | off  on   on   on   off
off  off  off  on   on   off   | off  on   on   on   on
off  off  off  on   on   on    | off  on   on   on   on
off  off  on   off  off  off   | off  on   on   on   off
off  off  on   off  off  on    | off  on   on   on   off
off  off  on   off  on   off   | off  on   on   on   on
off  off  on   off  on   on    | off  on   on   on   on
off  off  on   on   off  off   | off  on   on   on   off
off  off  on   on   off  on    | off  on   on   on   off
```

```
off off on  on  on  off | off on  on  on  on
off off on  on  on  on  | off on  on  on  on
off on  off off off off | off on  on  on  off
off on  off off off on  | off on  on  on  off
off on  off off on  off | off on  on  on  on
off on  off off on  on  | off on  on  on  on
off on  off on  off off | off on  on  on  off
off on  off on  off on  | off on  on  on  off
off on  off on  on  off | off on  on  on  on
off on  off on  on  on  | off on  on  on  on
off on  on  off off off | off on  on  on  off
off on  on  off off on  | off on  on  on  off
off on  on  off on  off | off on  on  on  on
off on  on  off on  on  | off on  on  on  on
off on  on  on  off off | off on  on  on  off
off on  on  on  off on  | off on  on  on  off
off on  on  on  on  off | off on  on  on  on
off on  on  on  on  on  | off on  on  on  on
on  off off off off off | off on  on  on  off
on  off off off off on  | off on  on  on  off
on  off off off on  off | off on  on  on  on
on  off off off on  on  | off on  on  on  on
on  off off on  off off | off on  on  on  off
on  off off on  off on  | off on  on  on  off
on  off off on  on  off | off on  on  on  on
on  off off on  on  on  | off on  on  on  on
on  off on  off off off | off on  on  on  off
on  off on  off off on  | off on  on  on  off
on  off on  off on  off | off on  on  on  on
on  off on  off on  on  | off on  on  on  on
on  off on  on  off off | off on  on  on  off
on  off on  on  off on  | off on  on  on  off
on  off on  on  on  off | off on  on  on  on
on  off on  on  on  on  | off on  on  on  on
on  on  off off off off | off on  on  on  off
on  on  off off off on  | off on  on  on  off
on  on  off off on  off | off on  on  on  on
on  on  off off on  on  | off on  on  on  on
on  on  off on  off off | off on  on  on  off
on  on  off on  off on  | off on  on  on  off
on  on  off on  on  off | off on  on  on  on
on  on  off on  on  on  | off on  on  on  on
on  on  on  off off off | off on  on  on  off
on  on  on  off off on  | off on  on  on  off
on  on  on  off on  off | off on  on  on  on
on  on  on  off on  on  | off on  on  on  on
on  on  on  on  off off | off on  on  on  off
on  on  on  on  off on  | off on  on  on  off
on  on  on  on  on  off | off on  on  on  on
on  on  on  on  on  on  | off on  on  on  on
```

Listing 5: Ausgabe Beispiel 5

3.6 Eigene Beispiele

3.6.1 Beispiel 6

```
4 6
X  Q1 Q2 X
X  W  W  X
r  R  R  r
X  B  B  X
X  W  W  X
X  L1 X  X
```

Listing 6: Eingabe Beispiel 6

```
Q1    Q2     | L1
off   off    | on
off   on     | on
on    off    | on
on    on     | off
```

Listing 7: Ausgabe Beispiel 6

3.6.2 Beispiel 7

```
4 6
X  Q1 Q2 X
X  W  W  X
r  R  R  r
X  B  B  X
X  W  W  X
X  X  X  X
```

Listing 8: Eingabe Beispiel 7

```
Q1    Q2     |
off   off    |
off   on     |
on    off    |
on    on     |
```

Listing 9: Ausgabe Beispiel 7

4 Quellcode

Im Folgenden wird der Quellcode der Datei ”Aufgabe_4.cpp” aufgeführt. Zur näheren Erläuterung des Quellcodes wurden Kommentare hinzugefügt, die mit ”//” und in grüner Schrift gekennzeichnet sind.

```
#include <bits/stdc++.h>

#define ve vector
#define vb ve<bool>
#define everyN(var) for(int var = 0; var < n; var++)
#define everyM(var) for(int var = 0; var < m; var++)

using namespace std;
ifstream inputFile; // input file stream
ofstream outputFile; // output file stream
int n, m, nOfInputs = 0;
ve<vb> states, results, combinations;
ve<ve<string>> field;

/**
 * @brief Get the Filename object
 *
 * @return string (filename)
 */
string getFilename(){
    string filename;
    cout << "Please enter filename without file extension" << endl; // print message to
    ↪ user
    cout << "Files must be located in '/beispieleingaben/'" << endl;
    cout << "-> ";
    cin >> filename; // get input from user
    return filename;
}

/**
```

```
 * @brief Read data from input file
 *
 */
void readData(){
    inputFile >> m >> n;
    field.resize(n);
    everyN(i){
        field[i].resize(m);
        everyM(j){
            inputFile >> field[i][j];
            if(field[i][j].at(0) == 'Q') nOfInputs++;
        }
    }
    inputFile.close();
}

/**
 * @brief Reset all entries in states matrix to false
 *
 */
void resetStates(){
    states.resize(n);
    everyN(i){
        states[i].resize(m);
        everyM(j)
            states[i][j] = false;
    }
}

/**
 * @brief Generate all possible binary combinations of length n
 *
 * @param n Length of combinations
 * @param arr Combination array
 * @param i Current index
 */
void generateCombinationsRecursiv(int n, bool arr[], int i){
    if(i == n){
        combinations.push_back(vb(arr, arr+n));
        return;
    }
    arr[i] = false;
    generateCombinationsRecursiv(n, arr, i+1);
    arr[i] = true;
    generateCombinationsRecursiv(n, arr, i+1);
}

/**
 * @brief Start recursive combination generation
 *
 */
void generateInputs(){
    bool arr[n];
    generateCombinationsRecursiv(nOfInputs, arr, 0);
}

/**
 * @brief Raise error and exit program
 *
 */
void raiseError(){
    cout << "unexcepted character" << endl;
    cout << "invalid input" << endl;
    exit(1);
}

/**
 * @brief Set the value of state array at (i,j) and (i,j+1) to val
 *
 * @param i
 * @param j
 * @param val
 */
```

```
103 void setStates(int i, int j, bool val){
        states[i][j] = val;
105     states[i][j+1] = val;
    }
107
    /**
109  * @brief Calculate result for given combination
     *
111  * @param combinationIndex
     * @return vector<bool> (result)
113  */
    vb calculateResult(int combinationIndex){
115     vb result;
        int cur = 0;
117     everyM(j)
            if(field[0][j].at(0) == 'Q')
119             states[0][j] = combinations[combinationIndex][cur++];
        for(int i = 1; i < n-1; i++){
121         everyM(j){
                if(field[i][j] == "W"){
123                 if(field[i][j+1] != "W") raiseError();
                    setStates(i, j, !(states[i-1][j] && states[i-1][j+1]));
125             } else if(field[i][j] == "B"){
                    if(field[i][j+1] != "B") raiseError();
127                 setStates(i, j, states[i-1][j] || states[i-1][j+1]);
                } else if(field[i][j] == "r"){
129                 if(field[i][j+1] != "R") raiseError();
                    setStates(i, j, !states[i-1][j+1]);
131             } else if(field[i][j] == "R"){
                    if(field[i][j+1] != "r") raiseError();
133                 setStates(i, j, !states[i-1][j]);
                } else {
135                 if(field[i][j] != "X") raiseError();
                    j--;
137             }
                j++;
139         }
        }
141     everyM(j)
            if(field[n-1][j].at(0) == 'L')
143             result.push_back(states[n-2][j]);
        return result;
145 }

147 /**
     * @brief Start calculation of all results
149  *        and measure time
     */
151 void calculateResults(){
        long long starttime = clock();
153     for(int i = 0; i < combinations.size(); i++){
            resetStates();
155         results.push_back(calculateResult(i));
        }
157     cout << "Calculated␣results␣in:␣" << ((long long)clock() - starttime) << "␣ms" <<
        ↪ endl;
    }
159
    /**
161  * @brief Print out results to console and output file
     *
163  * @param filename
     */
165 void printResults(string filename){
        outputFile.open("../beispielausgaben/"+filename+"_out.txt");
167     for(int i = 0; i < nOfInputs; i++){
            cout << "Q" << i+1 << "␣␣␣";
169         outputFile << "Q" << i+1 << "␣␣␣";
        }
171     cout << "␣|␣";
        outputFile << "␣|␣";
173     for(int i = 0; i < results[0].size(); i++){
            cout << "L" << i+1 << "␣␣␣";
```

```
175         outputFile << "L" << i+1 << "␣␣␣";
        }
177     cout << endl;
        outputFile << endl;
179     for(int i = 0; i < results.size(); i++){
            for(int j = 0; j < combinations[i].size(); j++){
181             cout << (combinations[i][j] ? "on␣␣␣" : "off␣␣");
                outputFile << (combinations[i][j] ? "on␣␣␣" : "off␣␣");
183         }
            cout << "␣|␣";
185         outputFile << "␣|␣";
            for(int j = 0; j < results[i].size();j++){
187             cout << (results[i][j] ? "on␣␣␣" : "off␣␣");
                outputFile << (results[i][j] ? "on␣␣␣" : "off␣␣");
189         }
            cout << endl;
191         outputFile << endl;
        }
193     outputFile.close();
}
195
/**
197  * @brief Main function of program
     *
199  * @return int (exit code)
     */
201 int main(){
        string filename = getFilename(); // get filename for in-/output
203     inputFile.open("../beispieleingaben/"+filename+".txt"); // open input file
        if(inputFile.is_open()) readData();
205     else {
            cout << "File␣not␣found" << endl;
207         exit(1);
        }
209     generateInputs();
        calculateResults();
211     printResults(filename);
        return 0;
213 }
```