

Name

Matr.-Nr.

Klausur Programmieren III – TI3

SS 12

Prof. Dr. K. Baer

Hilfsmittel: 1 Blatt DIN A4, beidseitig beschr.

Bearbeitungszeit: 90 Min.

1. Bitte tragen Sie zuerst Name und Matrikelnummer ein!
2. Kontrollieren Sie die Vollständigkeit der Aufgabenblätter.
3. Die Klausur besteht aus 6 Aufgaben. Verschaffen sie sich einen kurzen Überblick über die Aufgaben und beginnen Sie am besten mit der Aufgabe, die Ihnen am ehesten ein Erfolgserlebnis bringt.
4. Lesen Sie die Aufgabenstellung aufmerksam durch, bevor Sie eine Aufgabe lösen!
5. Verwenden Sie zur Beantwortung der Fragen den vorgesehenen Platz auf den Aufgabenblättern.
6. Schreiben Sie leserlich. Nicht lesbare Teile werden mit 0 Punkten bewertet!

Viel Erfolg!

Aufgabe	1	2	3	4	5	6	Summe
Punkte	20	6	14	20	6	24	90
erreicht							

Aufgabe 1 (20 Punkte = 2+11+2+2+1+ 2)

Gegeben Sie folgende Implementierung:

```
class Bauteil {
protected:
    string name;
public:
    Bauteil(){ name = "???" ; cout << "Bauteil erzeugt" << endl; }
    ~Bauteil(){ cout << "Bauteil zerstört" << endl; }
    void zeigeName(){ cout << "Name: " << name << endl; }
};

class A : public Bauteil {
public:
    A() { name = "A"; cout << name << " erstellt" << endl; }
    ~A(){ cout << name << " zerstört" << endl; }
};

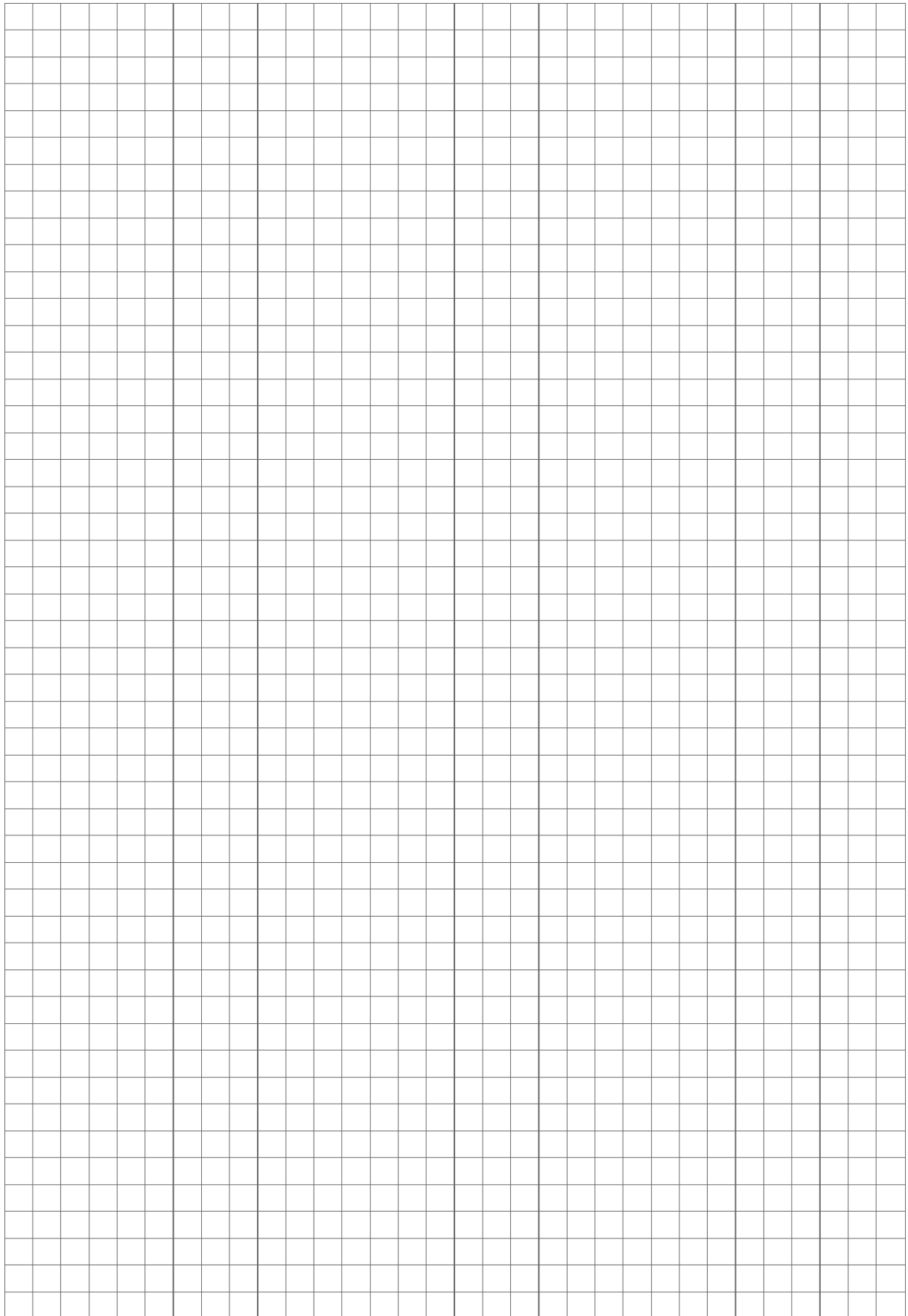
class B : public Bauteil {
public:
    B() { name = "B"; cout << name << " erstellt" << endl; }
    ~B(){ cout << name << " zerstört" << endl; }
    void zeigeName(){ cout << "B-Teile" << endl; Bauteil::zeigeName(); }
};

class Baugruppe {
private:
    set<Bauteil*> gruppe;
    typedef set<Bauteil*>::iterator Iter;
public:
    Baugruppe(){ cout << "Baugruppe erstellt" << endl; }
    Baugruppe(const Baugruppe& other){ cout << "Baugruppe kopiert" << endl;
        for(Iter i=other.gruppe.begin(); i != other.gruppe.end(); ++i){
            gruppe.insert(*i);
        }
    }
    virtual ~Baugruppe(){ cout << "Baugruppe zerstört" << endl; }
    void hinzufuegen(Bauteil* teil){ gruppe.insert(teil); }
    void entfernen( Bauteil* teil){ gruppe.erase(teil); }
    void zeigeElemente(){
        cout << endl << "-----";
        cout << "\nBaugruppe bestueckt mit: " << endl;
        for(Iter i = gruppe.begin(); i != gruppe.end(); ++i){
            (*i)->zeigeName();
        }
        cout << "-----" << endl;
    }
};
```

A blank sheet of graph paper with a grid pattern. The grid consists of small squares formed by thin gray lines. There are four vertical lines that divide the page into five equal-width columns. There are also horizontal lines that divide the page into rows. The entire page is covered by this grid pattern.

Geben Sie bitte zur Verdeutlichung für jede Ausgabezeile die **Zeilennummer** in der Methode test() an, die diese Ausgabe erzeugt.

This image shows a full page of blank graph paper. The grid consists of thin, light gray horizontal and vertical lines that intersect to form small squares across the entire surface. There are no margins, text, or other markings on the paper.



A full-page sheet of white graph paper featuring a uniform grid of thin gray lines. The grid consists of small squares covering the entire area. There are no margins, text, or other markings on the page.

Baugruppe(const Baugruppe& other)

anzumerken?

A full-page sheet of white graph paper featuring a uniform grid of thin black horizontal and vertical lines. The grid covers the entire area of the page, providing a template for drawing or writing.

Aufgabe 2 (6 Punkte)

Wie geht man sinnigerweise bei der Operatorüberladung vor? Erläutern Sie – etwa am Beispiel einer Klasse Bruch – welche Probleme typischerweise auftauchen und wie man am besten damit umgeht. Wie erreicht man bspw. im Falle der Klasse Bruch, dass die Implementierung das Kommutativgesetz erfüllt?

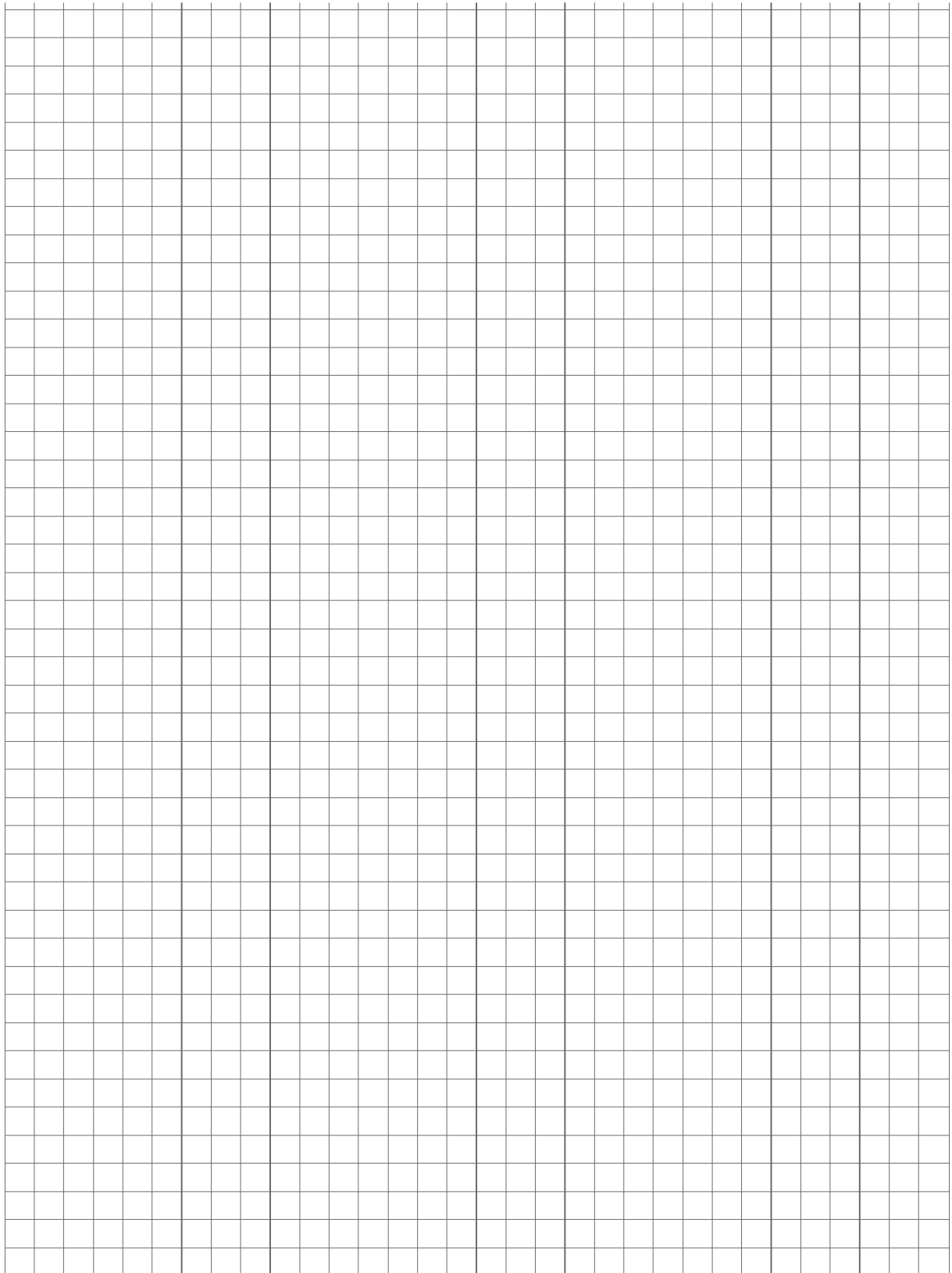
A large grid of graph paper, consisting of 20 columns and 30 rows of small squares, intended for the student to write their answer to the task.

Aufgabe 3 (14 Punkte=7+7)

- a) Erläutern Sie das friend-Konzept in C++!
(Zweck, Verwendung, Beispiel)

A large grid of graph paper, consisting of 20 columns and 30 rows of small squares, intended for the student to write their answer to the question.

- b) Was ist ein Binder in der STL?
(Zweck, Verwendung, Beispiel)



Aufgabe 4 (20 Punkte)

Eine Matrix kann aufgefasst werden als ein Feld von Feldern.

Bei Verwendung der STL kann man eine Template-Klasse für eine Matrix auf einfache Weise entwickeln, indem man eine Template-Klasse `Matrix` implementiert, die öffentlich von `vector< vector<T> >` abgeleitet wird.

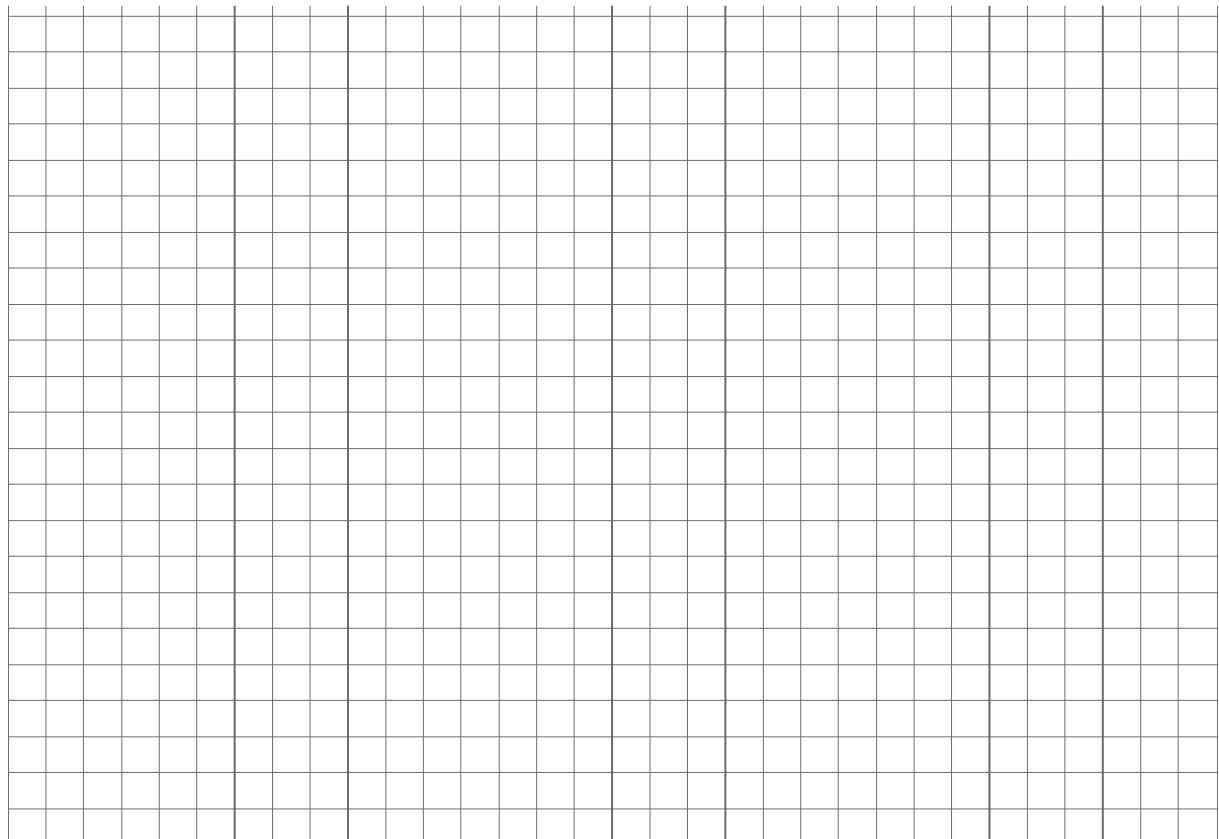
Entwickeln Sie eine solche Template-Klasse mit folgenden öffentlichen Methoden:

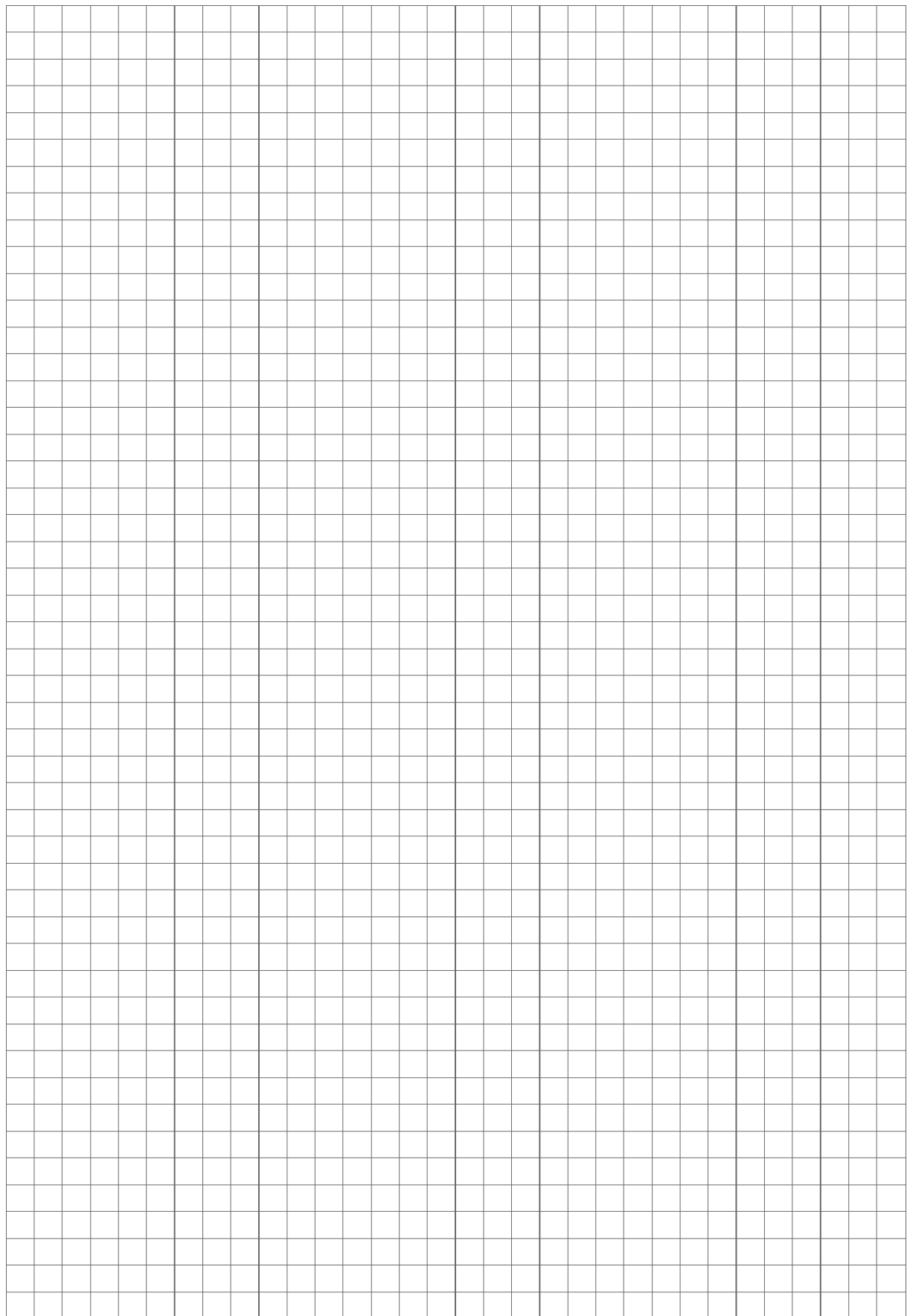
- ein 2-stelliger Konstruktor zur Angabe der Matrix-Dimension
- den Funktionen `Rows()` bzw. `Columns()` zur Ermittlung der Dimensionen
- eine Funktion `init()`, die erlaubt, die Matrix-Elemente mit einem angegebenen Wert zu initialisieren
- eine Funktion `I()`, die die Einheitsmatrix zurückgibt (überall 0, nur Diagonale mit 1 besetzt)
- den Ausgabeoperator `operator<<`, mit dem die Matrix in der Form wie folgende Einheitsmatrix ausgegeben wird (D.h., Zeilennr: Werte durch Leerzeichen getrennt)

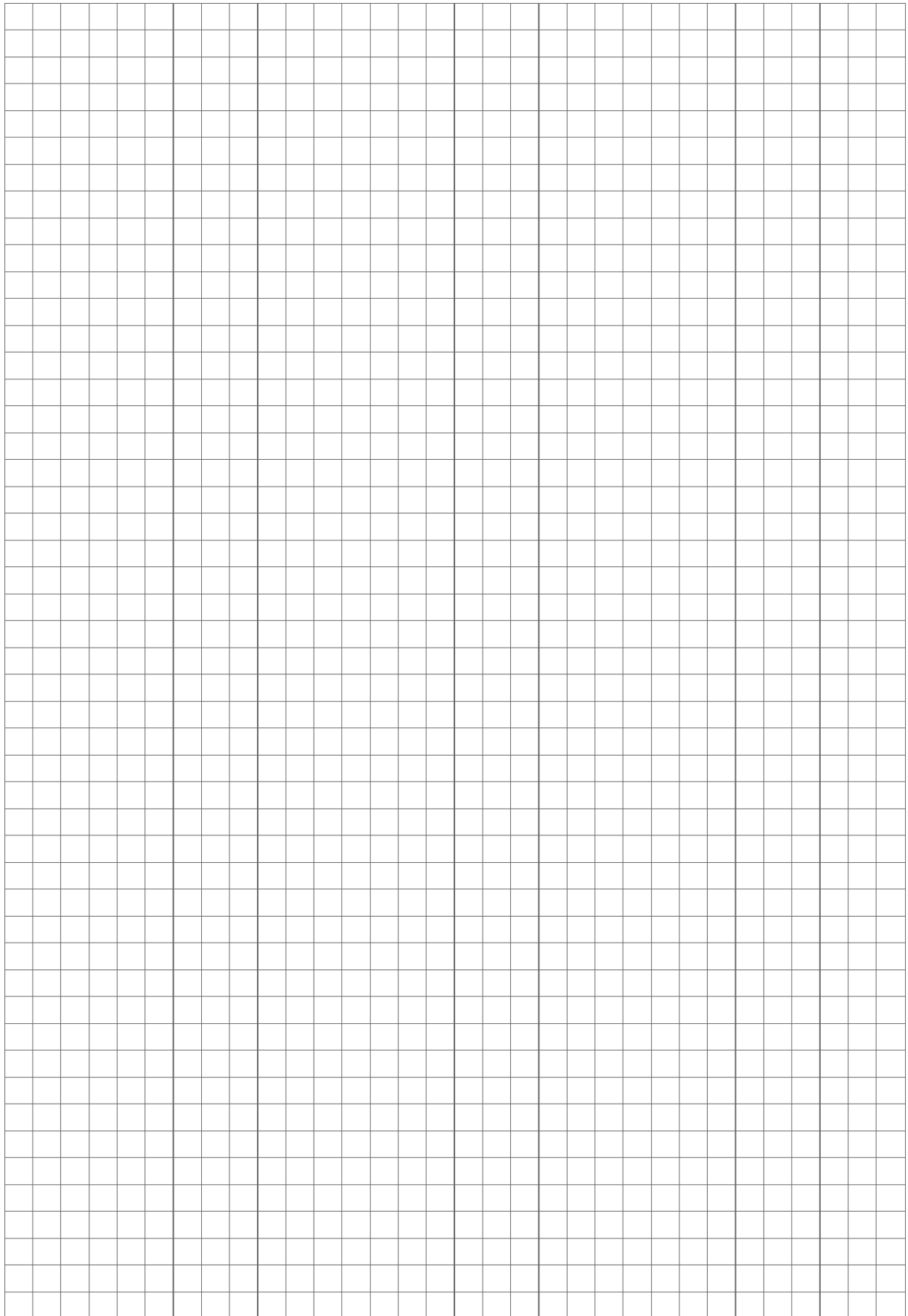
0: 1 0 0 0

1: 0 1 0 0

2: 0 0 1 0







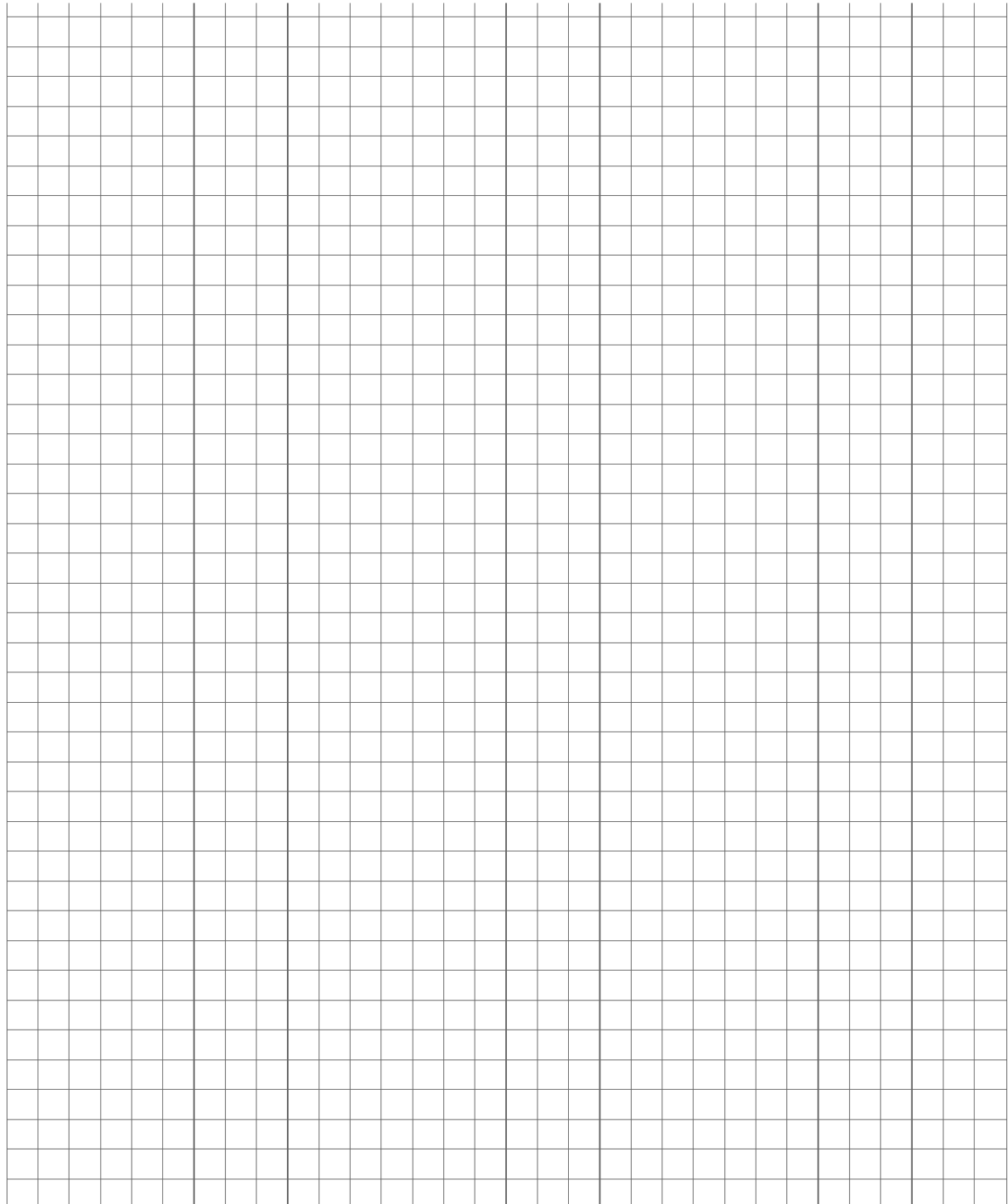
Aufgabe 5 (6 Punkte)

Erläutern Sie, was man unter einem downcast versteht.

Stellen Sie dar, in welchen Situationen Downcasts problematisch sind.

Stellen Sie dar, wie man korrekt damit umgeht.

Geben Sie ein Beispiel.

A large grid of graph paper, consisting of 20 columns and 30 rows of small squares, intended for the student to write their answer to the task.

Aufgabe 6 (24 Punkte = 2+1+6+4+11)

Entwickeln Sie eine Klasse „PlzOrtMapper“, die das Mapping von Postleitzahlen und Orten unterstützt.

Gegeben sei eine Textdatei nach untenstehendem Muster, in der pro Zeile eine Zuordnung PLZ zu Ort angegeben ist:

89075 Ulm
89076 Ulm
89077 Ulm
89077 Ulm-Soeflingen
89078 Ulm
89079 Goegglingen
89079 Ulm
89079 Ulm-Donautal
89079 Ulm-Eggingen
89079 Ulm-Einsingen
89079 Ulm-Goegglingen
89079 Ulm-Unterweiler
89079 Ulm-Wiblingen
89080 Ulm
89081 Ulm
89081 Ulm-Jungingen
89081 Ulm-Lehr
89081 Ulm-Maehringen
89081 Ulm-Seligweiler
89081 Ulm-Soeflingen
89081 Ulm-Ermingen
89081 Ulm-Jungingen

Daraus ist zu erkennen, dass ein Ort mehrere PLZ haben kann (z.B. Ulm). Genauso kann allerdings eine PLZ auch für mehrere Orte gültig sein (z.B. 89079)

- a) Erläutern Sie die Vorteile einer MultiMap.
Aus welchen Elementen ist eine Map / Multimap aufgebaut?

[illegible]

- b) Entwickeln Sie die Klasse „PlzOrtMapper“. Sie soll folgende Funktionen unterstützen:

```
bool einlesen(string dateiname) throw (FileError);  
int zaehleVorkommen(string ort);  
string listPlzOrt( ... );
```

- Überlegen Sie sich zunächst eine geeignete Datenstruktur, mit der Sie PLZ/Ort-Paarungen am einfachsten innerhalb Ihrer Klasse verwalten können.
- Die Methode „einlesen“ soll – falls die Datei nicht geöffnet werden kann – ein entsprechendes Fehlerobjekt „FileError“ zur Ausnahmebehandlung erzeugen. Der Dateiname soll an der aufrufenden Stelle dem Fehlerobjekt entnommen werden können.
- Die Methode zaehleVorkommen() liefert die Anzahl PLZ/Ort-Paarungen zu einem gegebenen Ort.
- Die Methode listPlzOrt() soll PLZ/Ort-Paarungen als string liefern. Es soll möglich sein, die Funktion so aufzurufen, dass
 - alle Paarungen mit dem genauen Ortsnamen,
 - alle Paarungen, in denen der Ortsname den gegebenen Suchstring enthält,
 - alle Paarungen, die zu einer PLZ gehören,aufgelistet werden.

Hinweise zur STL:

- Multimap bietet eine Elementfunktion „equal_range“, mit der die Grenzen eines Bereichs in der Map ermittelt werden können, in dem der Schlüssel einen bestimmten Wert hat:

```
pair<iterator,iterator> equal_range ( const key_type& x );
```

- Der Abstand zweier Iteratoren kann mit Hilfe der globalen Funktion distance ermittelt werden:

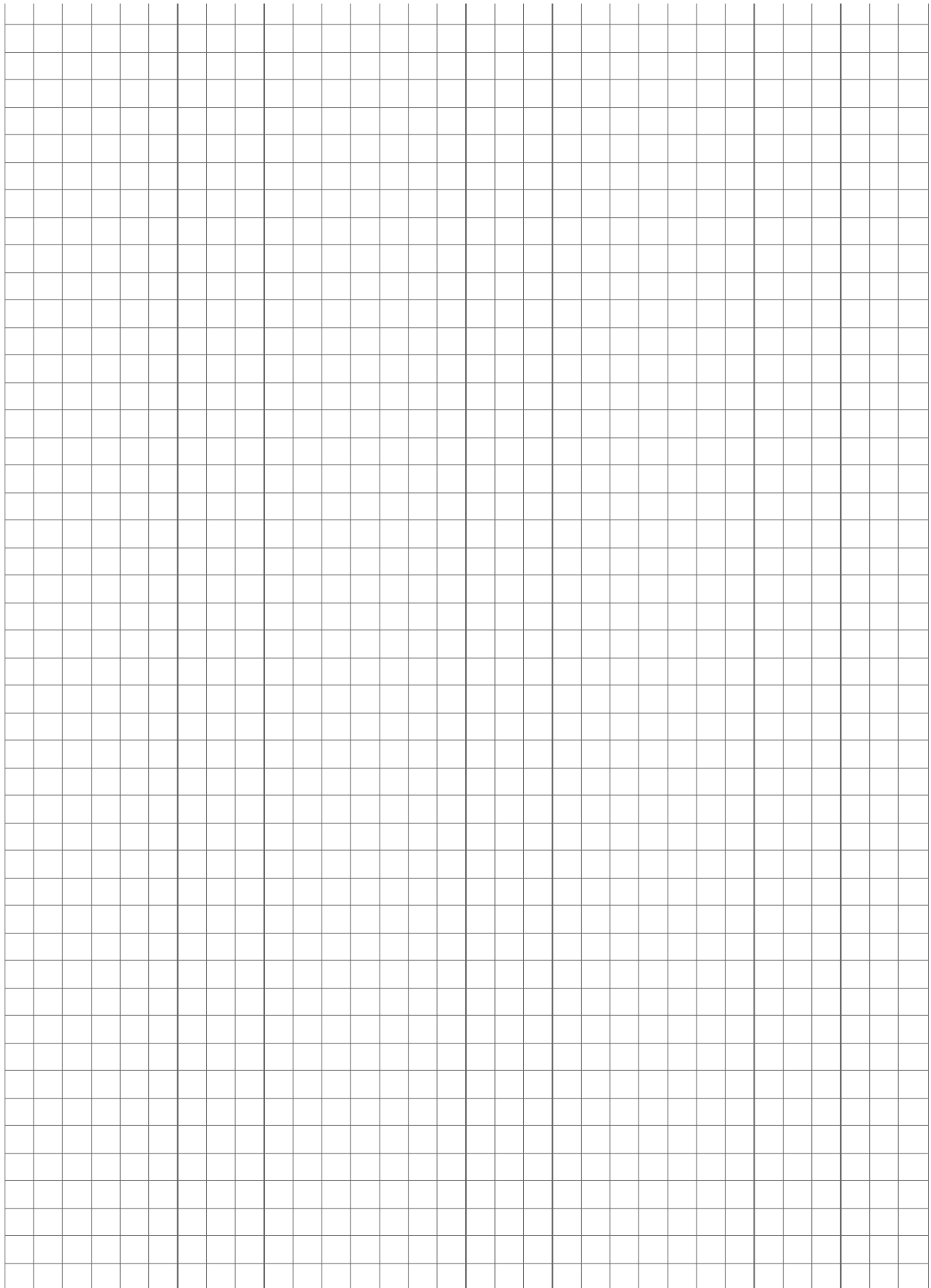
```
size_t = distance( iterator1, iterator2 );
```

- Mit Hilfe der globalen Function find_if kann in einem Container nach Werten gesucht werden, die eine bestimmte Bedingung (ein Prädikat) erfüllen. Dazu benötigt die Funktion den Bereich im Container, in dem gesucht werden soll sowie die Vergleichsfunktion. Sie liefert die Position des ersten gefundenen Elements (oder last, falls nichts gefunden wurde).

```
template <class InputIterator, class Predicate>  
InputIterator find_if ( InputIterator first,  
                      InputIterator last,  
                      Predicate pred );
```

[illegible][illegible]

b3) Implementieren Sie die Methode **zaehleVorkommen**



b4) Implementieren die Methode listPlzOrt.

Hilfestellung:

- Die unterschiedlichen Suchen können Sie über unterschiedliche Funktoren abbilden
- Einmal wird nach Plz (\rightarrow int), die anderen Male nach Ort (\rightarrow string) gesucht. Wenn man listPlzOrt als template implementiert, kann man entsprechend Implementierungsaufwand sparen.
- In stringstream-Objekte kann mit dem üblichen Ausgabe-Operator << wie auf die Konsole gesschrieben werden. stringstream-Objekte besitzen eine Methode str() zur Konvertierung in einen string.
- Strings bieten die Funktion

```
size_t find ( const string& str, size_t pos = 0 ) const;
```

die die Position des ersten Auftretens des Suchstrings liefert.

Ein Testprogramm sollte Output nach folgendem Muster liefern:

Liste Postleitzahl(en) zu Ort: Ulm

```
89070  Ulm
89073  Ulm
89074  Ulm
89075  Ulm
... (usw.)
```

Liste PLZ zu Orten, die den Namensbestandteil enthalten: Soef

```
89077  Ulm-Soeflingen
89081  Ulm-Soeflingen
```

Liste alle Orte, die zu einer Postleitzahl gehören: 89079

```
89079  Goegglingen
89079  Ulm
89079  Ulm-Donautal
89079  Ulm-Eggingen
89079  Ulm-Einsingen
89079  Ulm-Goegglingen
89079  Ulm-Unterweiler
89079  Ulm-Wiblingen
```

