Prof. Dr. Andreas Claßen, Prof. Dr. Thomas Dey

Pflicht-Offline-Aufgabe 07-01, INF & WI & MCD:

Text rechtsbündig setzen

(geübte C++ Konstrukte: Strings, Array, `for` Schleife)

Schreiben Sie ein C++ Programm, bei dem der Benutzer maximal vier einzeilige Texte eingeben kann (ggfs. inklusive Leerzeichen und/oder Satzzeichen). Die eingegebenen Textzeilen sollen dann rechtsbündig wieder ausgegeben werden, wobei links jeweils mit dem Zeichen `#` (anstelle der normalerweise üblichen Leerzeichen) aufgefüllt werde.

Dieses Auffüllen soll von Ihnen selbst mittels String-Funktionen (`length()`, `at()`, `+`, `+=` etc.) programmiert werden. Benutzen Sie also bitte nicht die `setw()` und `setfill()` Manipulatoren der Standardausgabe `cout`.

Der Benutzer soll die Eingabe der Textzeilen auch vorzeitig durch leere Eingabe (Eingabe einer komplett leeren Zeile) beenden können. Anderenfalls sollen (maximal) vier Eingabezeilen entgegengenommen werden, d.h. nach der vierten nicht-leeren Eingabezeile soll das Programm nicht nach weiteren Zeilen fragen.

Das rechtsbündige Setzen der eingegebenen Textzeilen soll wie folgt durchgeführt werden: Das Programm bestimme die Länge der längsten eingegebenen nicht-leeren Eingabezeile. Alle anderen nicht-leeren Eingabezeilen werden dann von links mit `#` Zeichen aufgefüllt, bis sie identische Länge zur längsten Eingabezeile haben.

Testläufe (Benutzereingaben zur Verdeutlichung unterstrichen):

```
Textzeile = ? Die erste Eingabezeile.
Textzeile = ? Und noch eine Zeile.
Textzeile = ? Und noch eine.
Textzeile = ? Eine sehr sehr lange Eingabezeile.
###########Die erste Eingabezeile.
#############Und noch eine Zeile.
###################Und noch eine.
Eine sehr sehr lange Eingabezeile.
Drücken Sie eine beliebige Taste . . .
```

```
Textzeile = ?
Drücken Sie eine beliebige Taste . . .
```

```
Textzeile = ? 12 34
Textzeile = ? abcd efghij
Textzeile = ? A!B?C!
Textzeile = ?
######12 34
abcd efghij
#####A!B?C!
Drücken Sie eine beliebige Taste . . .
```

```
Textzeile = ? 12 34
Textzeile = ? abcd efghij
Textzeile = ?
######12 34
abcd efghij
Drücken Sie eine beliebige Taste . . .
```

Prof. Dr. Andreas Claßen, Prof. Dr. Thomas Dey

Freiwillige Offline-Aufgabe 07-02 (INF & WI & MCD): Wortliste eines Texts

(geübte C++ Konstrukte: string, Array, Schleifen)

Schreiben Sie ein C++ Programm, welches einen Text in Form von mehreren Eingabezeilen einliest. Das Programm soll die in dem Text auftretenden Worte in der Reihenfolge ihres Auftretens ausgeben. Worte sind dabei durch Leerzeichen bzw. Anfang/Ende jeder Eingabezeile begrenzt.

Es kommen maximal `5` Eingabezeilen vor. Eine leere Eingabezeile beende die Eingabe aber vorzeitig.
Eingabezeilen dürfen nur Leerzeichen und Buchstaben (Großbuchstaben, Kleinbuchstaben) enthalten. D.h. Satzzeichen sind also z.B. nicht erlaubt. Der Benutzer machen nur korrekte Eingaben.

Da die Wortliste erst nach allen Eingaben ausgegeben wird, müssen alle Eingaben zwischengespeichert werden. Benutzen Sie dazu ein Array aus `string`, mit 5 Array-Einträgen.

Testläufe (Benutzereingaben sind unterstrichen):

```
Eingabezeile = ? Dies ist die erste Zeile des Texts
Eingabezeile = ? Und dies ist noch          eine Zeile
Eingabezeile = ?       Und noch eine weitere Zeile im Text
Eingabezeile = ?
Dies
ist
die
erste
Zeile
des
Texts
Und
dies
ist
noch
eine
Zeile
Und
noch
eine
weitere
Zeile
```

```
im
Text
Drücken Sie eine beliebige Taste . . .
```

Freiwillige Offline-Aufgabe 07-03 (INF & WI & MCD): Wortliste eines Texts, mit Häufigkeiten

(geübte C++ Konstrukte: string, Array, Schleifen, struct)

Schreiben Sie ein C++ Programm, welches einen Text in Form von mehreren Eingabezeilen einliest. Das Programm soll die in dem Text auftretenden Worte in der Reihenfolge ihres Auftretens mitsamt der Häufigkeit ihres Vorkommens im Text ausgeben.

Einzelne Eingabezeilen und Worte sollen als Strings gespeichert werden. Die „Sammlung aller nicht-leeren Eingabezeilen" sowie die „Sammlung der Worthäufigkeiten" sollen als statische Arrays gespeichert werden. Es kommen maximal `5` Eingabezeilen und maximal `1000` Worte vor.

Eine leere Eingabezeile beende die Eingabe. Eingabezeilen können auch Leerzeichen enthalten. Leerzeichen, Punkt, Komma sowie Anfang und Ende jeder Eingabezeile fungieren als Wortbegrenzungen. Im Eingabetext sollen keine anderen Satzzeichen als Punkt und Komma (und Leerzeichen) vorkommen. Ihr Programm kann davon ausgehen, dass der Benutzer nur korrekte Eingaben macht.

Die Worthäufigkeit jedes Worts soll in einer Datenstruktur …

```
struct w_haeufigkeit {
    string wort;
    int haeufigkeit;
};
```

… gespeichert werden (und davon ein Array mit `1000` Einträgen).

Es sollen für das Ermitteln der Worthäufigkeiten folgende Funktionen definiert werden (diese Funktionen werden von Jenkins ggfs. auch einzeln getestet, sie müssen also vorhanden sein und wie vorgeschrieben definiert werden):

```
string naechstes_wort(string zeile, int& pos);
```

… suche ausgehend von der Position `pos` in der Zeile das nächste Wort und gebe es zurück. `pos` wird dann auf die erste Position (in der Zeile) hinter dem Wort geändert. Sollte es ausgehend von `pos` kein nächstes Wort in der Zeile geben, so werde ein leerer String zurückgegeben.

```
void zaehle_wort(string wort,
                 w_haeufigkeit haeufigkeiten[],
                 int& w_count);
```

… suche das Wort `wort` in der Wortsammlung `haeufigkeiten[]`, die aktuell bis zur Position `w_count-1` belegt sei. Kommt das Wort schon vor, so werde seine Häufigkeit um `1` erhöht. Kommt das Wort bisher noch nicht vor, so werde es hinzugefügt und seine Häufigkeit auf `1` gesetzt.

Die Arrays für den Eingabetext und die gesammelten Worthäufigkeiten seien lokale Variablen des Hauptprogramms. Benutzen Sie ggfs. folgendes Programmgerüst:

```
#include <iostream>
#include <string>
using namespace std;

struct w_haeufigkeit {
	string wort;
	int haeufigkeit;
};

string naechstes_wort(string zeile, int& pos)
{

}

void zaehle_wort(string wort, w_haeufigkeit haeufigkeiten[], int& w_count)
{

}

int main()
{
	int z_count = 0, w_count = 0;
	string wort = "";
	string eingabe[5];

	/* Eingabe hier, setzt auch z_count auf die Anzahl nicht-leerer Zeilen */ {

	w_haeufigkeit haeufigkeiten[1000];

	for (int k = 0; k < z_count; k++)
	{
		int pos = 0;
		while (true)
		{
			wort = naechstes_wort(eingabe[k], pos);
			if (wort == "")
				break;
			zaehle_wort(wort, haeufigkeiten, w_count);
		}
	}
```

```
	for (int k = 0; k < w_count; k++)
		cout << haeufigkeiten[k].wort << ": "
			<< haeufigkeiten[k].haeufigkeit << endl;

	system("PAUSE");
	return 0;
}
```

Testläufe (Benutzereingaben sind unterstrichen):

```
Eingabezeile = ? Dies ist die erste Zeile des Texts.
Eingabezeile = ? Und dies, ist, noch          eine Zeile.
Eingabezeile = ? .Und noch eine weitere Zeile, im Text.
Eingabezeile = ?
Dies: 1
ist: 2
die: 1
erste: 1
Zeile: 3
des: 1
Texts: 1
Und: 2
dies: 1
noch: 2
eine: 2
weitere: 1
im: 1
Text: 1
Drücken Sie eine beliebige Taste . . .
```

Freiwillige Offline-Aufgabe 07-04 (INF & MCD & WI): Funktion `filter()`

(geübte C++ Konstrukte: struct, Arrays, Funktionen mit Array Parametern)

Gegeben sei die folgende `struct` zur Erfassung von Personendaten:

```
struct Person
{
    string nachname, vorname;
    int alter;
    char geschlecht;
};
```

Gegeben sei ferner das folgende Array zur Speicherung solcher Daten:

```
Person personen[100];
int anzahl_personen = 0;
```

Programmieren Sie zuerst ein Hauptprogramm, mit dem man Personendaten gemäß des folgenden Testlaufs einlesen kann:

Testlauf (Benutzereingaben sind unterstrichen):

```
Eine weitere Person eingeben (j/n)? j
Bitte den Nachnamen der 1. Person eingeben: ? Musterfrau
Bitte den Vornamen der 1. Person eingeben: ? Petra
Bitte das Alter der 1. Person eingeben: ? 33
Bitte das Geschlecht der 1. Person eingeben: ? w
Eine weitere Person eingeben (j/n)? j
Bitte den Nachnamen der 2. Person eingeben: ? Mustermann
Bitte den Vornamen der 2. Person eingeben: ? Peter
Bitte das Alter der 2. Person eingeben: ? 44
Bitte das Geschlecht der 2. Person eingeben: ? m
Eine weitere Person eingeben (j/n)? n
Drücken Sie eine beliebige Taste . . .
```

Das Hauptprogramm kann davon ausgehen, dass der Benutzer nur korrekte Eingaben macht, bis auf die `j/n` Eingabe: Diese soll explizit auf

Korrektheit geprüft werden, und bei Falscheingabe soll die Eingabeaufforderung wiederholt werden.

Testlauf (Benutzereingaben sind unterstrichen):

```
Eine weitere Person eingeben (j/n)? x
Eine weitere Person eingeben (j/n)? j
Bitte den Nachnamen der 1. Person eingeben: ? Musterfrau
Bitte den Vornamen der 1. Person eingeben: ? Petra
Bitte das Alter der 1. Person eingeben: ? 33
Bitte das Geschlecht der 1. Person eingeben: ? w
Eine weitere Person eingeben (j/n)? n
Drücken Sie eine beliebige Taste . . .
```

Sollten Sie in ihrem Hauptprogramm zwischen der Eingabe mittels `cin >>` und der Eingabe mittels `getline(…)` wechseln, so kann es notwendig werden, nach jeder Eingabe mittels `cin >>` ein …

```
cin.clear();
cin.ignore();
```

… einzufügen, damit die Eingabe korrekt funktioniert.

Programmieren Sie nun eine Funktion

`void filter(` /* Parameter siehe Aufgabenstellung */ `);`

… welche das Personen-Array als ersten Parameter und `anzahl_personen` als zweiten Parameter entgegennimmt. Die Funktion soll vom Benutzer erfragen, welche Personendaten ausgegeben und gezählt werden sollen, und dies dann entsprechend umsetzen. Siehe Testläufe (der Teil ab „`Ihre Auswahl`“ werde durch die Funktion realisiert). Die Benutzereingabe braucht nicht auf Korrektheit geprüft zu werden.

Ihre Funktion soll aus dem Hauptprogramm aufgerufen werden. Ergänzen Sie dieses also entsprechend.

Um nicht beim Testen des Programms jedes Mal alle Personendaten neu eingeben zu müssen, ersetzen Sie bitte im Hauptprogramm die beiden Zeilen zur Definition des Arrays durch

```
Person personen[100] = {
{ "Musterfrau1", "Petra1", 18, 'w' },
{ "Mustermann1", "Klaus1", 18, 'm' },
{ "Mustermaedchen1", "Lisa1", 1, 'w' },
{ "Musterjunge1", "Jan1", 1, 'm' },
{ "Musterfrau2", "Petra2", 19, 'w' },
{ "Mustermann2", "Klaus2", 19, 'm' },
{ "Mustermaedchen2", "Lisa2", 2, 'w' },
{ "Musterjunge2", "Jan2", 2, 'm' },
{ "Musterfrau3", "Petra3", 20, 'w' },
{ "Mustermann3", "Klaus3", 20, 'm' },
{ "Mustermaedchen3", "Lisa3", 3, 'w' },
{ "Musterjunge3", "Jan3", 3, 'm' },
{ "Musterfrau4", "Petra4", 21, 'w' },
{ "Mustermann4", "Klaus4", 21, 'm' },
{ "Mustermaedchen4", "Lisa4", 4, 'w' },
{ "Musterjunge4", "Jan4", 4, 'm' },
{ "Musterfrau5", "Petra5", 22, 'w' },
{ "Mustermann5", "Klaus5", 22, 'm' },
{ "Mustermaedchen5", "Lisa5", 5, 'w' },
{ "Musterjunge5", "Jan5", 5, 'm' },
{ "Musterfrau6", "Petra6", 23, 'w' },
{ "Mustermann6", "Klaus6", 23, 'm' },
{ "Mustermaedchen6", "Lisa6", 6, 'w' },
{ "Musterjunge6", "Jan6", 6, 'm' },
{ "Musterfrau7", "Petra7", 24, 'w' },
{ "Mustermann7", "Klaus7", 24, 'm' },
{ "Mustermaedchen7", "Lisa7", 7, 'w' },
{ "Musterjunge7", "Jan7", 7, 'm' },
{ "Musterfrau8", "Petra8", 25, 'w' },
{ "Mustermann8", "Klaus8", 25, 'm' },
{ "Mustermaedchen8", "Lisa8", 8, 'w' },
{ "Musterjunge8", "Jan8", 8, 'm' }
};
int anzahl_personen = 32;
```

Testläufe des Gesamtprogramms, d.h. Hauptprogramm plus Funktion (Benutzereingaben sind unterstrichen):

```
Eine weitere Person eingeben (j/n)? n
Ihre Auswahl:
1 - Weibliche Erwachsene
2 - Maennliche Erwachsene
3 - Weibliche Kinder
4 - Maennliche Kinder
? 1
Musterfrau1, Petra1, w, 18
Musterfrau2, Petra2, w, 19
Musterfrau3, Petra3, w, 20
Musterfrau4, Petra4, w, 21
Musterfrau5, Petra5, w, 22
Musterfrau6, Petra6, w, 23
Musterfrau7, Petra7, w, 24
Musterfrau8, Petra8, w, 25
Summe: 8
Drücken Sie eine beliebige Taste . . .
```

```
Eine weitere Person eingeben (j/n)? n
Ihre Auswahl:
1 - Weibliche Erwachsene
2 - Maennliche Erwachsene
3 - Weibliche Kinder
4 - Maennliche Kinder
? 4
Musterjunge1, Jan1, m, 1
Musterjunge2, Jan2, m, 2
Musterjunge3, Jan3, m, 3
Musterjunge4, Jan4, m, 4
Musterjunge5, Jan5, m, 5
Musterjunge6, Jan6, m, 6
Musterjunge7, Jan7, m, 7
Musterjunge8, Jan8, m, 8
Summe: 8
Drücken Sie eine beliebige Taste . . .
```

```
Eine weitere Person eingeben (j/n)? j
Bitte den Nachnamen der 33. Person eingeben: ? Musterfrau33
Bitte den Vornamen der 33. Person eingeben: ? Petra
Bitte das Alter der 33. Person eingeben: ? 33
Bitte das Geschlecht der 33. Person eingeben: ? w
Eine weitere Person eingeben (j/n)? j
Bitte den Nachnamen der 34. Person eingeben: ? Mustermann44
Bitte den Vornamen der 34. Person eingeben: ? Peter
Bitte das Alter der 34. Person eingeben: ? 44
Bitte das Geschlecht der 34. Person eingeben: ? m
Eine weitere Person eingeben (j/n)? n
Ihre Auswahl:
1 - Weibliche Erwachsene
2 - Maennliche Erwachsene
3 - Weibliche Kinder
4 - Maennliche Kinder
? 1
Musterfrau1, Petra1, w, 18
Musterfrau2, Petra2, w, 19
Musterfrau3, Petra3, w, 20
Musterfrau4, Petra4, w, 21
Musterfrau5, Petra5, w, 22
Musterfrau6, Petra6, w, 23
Musterfrau7, Petra7, w, 24
Musterfrau8, Petra8, w, 25
Musterfrau33, Petra, w, 33
Summe: 9
Drücken Sie eine beliebige Taste . . .
```

Pflicht-Offline-Aufgabe O 07-05 (INF & WI & MCD): Morse Alphabet

(`struct`, Funktionen)

Schreiben Sie ein Programm, welches einen einzeiligen Text einliest (ggfs. mit Leerzeichen) und diesen Text im Morse-Alphabet codiert wieder ausgibt.

Die Zeichen des Morse-Alphabets finden Sie im vorgegebenen Code-Rahmen.

Nach den Morsezeichen für jeden einzelnen Buchstaben des Originaltexts soll das Zeichen `#` ausgegeben werden, um die Morsezeichen für die einzelnen Buchstaben visuell voneinander zu trennen. Leerzeichen des Originaltexts (die ja im Originaltext die Worte voneinander trennen) werden im Morse-Alphabet als Pausen dargestellt, in unserer Ausgabe jedoch durch drei aufeinanderfolgende Vorwärts-Schrägstriche `///`.

Codegerüst zur Definition der Morsezeichen:

```
struct T_Morse_Data
{
    char letter;
    std::string morse_code;
};

const T_Morse_Data morse_data[] = {
        { 'a', ".-" },
        { 'b', "-..." },
        { 'c', "-.-." },
        { 'd', "-.." },
        { 'e', "." },
        { 'f', "..-." },
        { 'g', "--." },
        { 'h', "...." },
        { 'i', ".." },
        { 'j', ".---" },
        { 'k', "-.-" },
        { 'l', ".-.." },
        { 'm', "--" },
        { 'n', "-." },
        { 'o', "---" },
        { 'p', ".--." },
        { 'q', "--.-" },
```

```
        { 'r', ".-." },
        { 's', "..." },
        { 't', "-" },
        { 'u', "..-" },
        { 'v', "...-" },
        { 'w', ".--" },
        { 'x', "-..-" },
        { 'y', "-.--" },
        { 'z', "--.." },
        { '0', "-----" },
        { '1', ".----" },
        { '2', "..---" },
        { '3', "...--" },
        { '4', "....-" },
        { '5', "....." },
        { '6', "-...." },
        { '7', "--..." },
        { '8', "---.." },
        { '9', "----." },
        // Leerzeichen zur Worttrennung werden im Morsecode
        // durch einen Slash '/' umgeben von Leerzeichen
        // dargestellt ...
        // (Alternative: Trennung durch 3 Leerzeichen)
        // Hier bei uns, um die Testläufe eindeutiger zu
        // machen: Trennung mittels drei Slashes
        { ' ', "///" },

        { '.', ".-.-.-" },
        { ',', "--..--" },
        { ':', "---..." },
        { '?', "..--.." },
        { '-', "-....-" },
        { '/', "-..-." },
        { '@', ".--.-." },
        { '=', "-...-" }
};

const int morse_data_len = 45;
```

Testläufe (Benutzereingaben zur Verdeutlichung unterstrichen):

```
Bitte Text eingeben (ggfs. mit Leerzeichen): ? Dies ist ein Text.
-..#..#.#...#///#..#...#-#///#.#..#-.#///#-#.#-..-#-#.-.-.-#
Drücken Sie eine beliebige Taste . . .
```

```
Bitte Text eingeben (ggfs. mit Leerzeichen): ? 0123456789
-----#.----#..---#...--#....-#.....#-....#--...#---..#----.#
Drücken Sie eine beliebige Taste . . .
```

```
Bitte Text eingeben (ggfs. mit Leerzeichen): ? ,:?@=-
--..--#---...#..--..#.--.-.#-...-#-....-#
Drücken Sie eine beliebige Taste . . .
```

```
Bitte Text eingeben (ggfs. mit Leerzeichen): ?

Drücken Sie eine beliebige Taste . . .
```
