**03 Notenstatistik**

**Zusammenfassung der Problemstellung**

Das Programm soll eine Anzahl von Punkten von Studierenden und die benötigte Punktzahl für die Note 6 einlesen. Im Anschluss sollen die Noten der einzelnen Studierenden berechnet werden. Die Ausgabe am Schluss ist eine Statistik über das gesamte und enthält Anzahl vorkommen jeder Note, Angaben wie beste- und schlechteste vorkommende Note, Notendurchschnitt, Anzahl Studierende und Anzahl Studierende, die bestanden haben (Note > =4).

**Lösungsvarianten**

Um die Berechnung aller benötigten Informationen beieinander zu haben gibt es ein struct, das alle Variablen enthält. Es gibt eine Funktion, welche die Noten aus den Punkten berechnet, eine Funktion, welche die Werte für die Statistik berechnet und eine Funktion für die Ausgabe der Statistik. Die main-Funktion liest die Daten ein und übergibt sie den Funktionen. Wir haben uns für zusätzliche Funktionen entschieden, um verschiedene Aufgaben innerhalb des Programms klar aufzuteilen und damit Übersicht zu schaffen.

**Modulübersicht**

Da es sich um ein sehr kleines Programm handelt, befindet sich der ganze Code in main.c, aufgeteilt auf die 4 Funktionen main, get\_statistics, get\_mark, print\_statistics.

**Globale Variablen:**

* struct Statistics

**Lokale Variablen:**

* int pointlist[100] (main)
* int len (main)
* int all\_points\_inserted (main)
* int points\_6 (main)
* Statistic statistics (main)
* Char rerun (main)
* int grades[len]
* int mark6 (get\_statistics)
* int mark5 (get\_statistics)
* int mark4 (get\_statistics)
* int mark3 (get\_statistics)
* int mark2 (get\_statistics)
* int mark1 (get\_statistics)
* int best\_mark (get\_statistics)
* int worst\_mark (get\_statistics)
* double average\_mark (get\_statistics)
* int i (get\_statistics)
* double mark (get\_mark)

**Programmablauf**

Main > ruft get\_statistics auf, zur Berechnung aller für die Ausgabe benötigten Infos. Get\_statistics ruft innerhalb noch get\_mark auf, wo aus den Punkte die Note berechnet und zurückgegeben wird. Get\_statistics gibt das struct Statistics zurück an die main-Funktion, die zum Schluss print\_statistics aufruft alle Angaben auf der Konsole ausgibt.

**Tests**

*test\_berechnung\_schaltjahr:* prüft die Berechnung des Folgetages im spezialfall Schaltjahr

*test\_berechnung\_monatwechsel:* prüft die Berechnung des Folgetages für einen normalen Monatswechsel

*test\_berechnung\_jahreswechsel:* prüft die Berechnung des Folgetages für einen normalen Jahreswechsel

*test\_berechnung\_erstes\_datum:* prüft die Berechnung des Folgetages für das erste gültige Datum

*test\_teste\_gueltigkeit\_vor\_erstem\_datum:* prüft, ob der Gültigkeitstest ein Datum vor dem ersten gültigen Datum ablehnt

*test\_teste\_gueltigkeit\_von\_datum\_tag:* prüft, ob der Gültigkeitstest ein ungültiges Datum ablehnt

*test\_teste\_gueltigkeit\_von\_datum\_monat:* prüft, ob der Gültigkeitstest ein ungültiges Datum ablehnt

*test\_teste\_gueltigkeit\_von\_datum\_in\_zukunft:* prüft, ob der Gültigkeitstest ein Datum in der Zukunft akzeptiert

Alle Tests liefen erfolgreich durch.

**Erkenntnisse**

Das Programm läuft fehlerfrei.

Die Funktion get\_statistics könnte noch weiter auf mehrere Funktionen aufgeteilt werden. Z.B das zählen, wie oft jede Note vorkommt, könnte ausgelagert werden. Eine weitere Möglichkeit einer Umstrukturierung wäre stattdessen get\_mark ebenfalls von der main-funktion aus aufzurufen. Jedoch macht es entsprechend der Aufteilung nach Zuständigkeiten, mehr Sinn, wenn get\_statistics sich darum kümmert.

Mögliche Erweiterungen

**Anhang A: Quelltext**

/\* ----------------------------------------------------------------------------

\* -- \_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_ -

\* -- |\_ \_| | \_\_\_\_|/ \_\_\_\_| -

\* -- | | \_ \_\_ | |\_\_ | (\_\_\_ Institute of Embedded Systems -

\* -- | | | '\_ \| \_\_| \\_\_\_ \ Zuercher Hochschule Winterthur -

\* -- \_| |\_| | | | |\_\_\_\_ \_\_\_\_) | (University of Applied Sciences) -

\* -- |\_\_\_\_\_|\_| |\_|\_\_\_\_\_\_|\_\_\_\_\_/ 8401 Winterthur, Switzerland -

\* ----------------------------------------------------------------------------

\*/

/\*\*

\* @file

\* @brief Lab implementation

\*/

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

typedef struct {

int students;

int points\_6;

int best\_mark;

int worst\_mark;

double average\_mark;

int mark\_6;

int mark\_5;

int mark\_4;

int mark\_3;

int mark\_2;

int mark\_1;

int passed;

} Statistics;

Statistics get\_statistics(int pointlist[], int len, int points\_6);

int get\_mark(int points, int points\_6);

void print\_statistics(Statistics);

/\*\*

\* @brief Main entry point. Reads Input: Points from all students and

\* min. Points that ar needed for mark 6.

\* @returns Returns EXIT\_SUCCESS (=0) on success,

\* EXIT\_FAILURE (=1) on failure.

\*/

int main(void)

{

int pointlist[100];

int len = 0;

int all\_points\_inserted = 0;

int points\_6 = 0;

Statistics statistics;

char rerun = 'y';

//read input Data: points from all Students

(void)printf("Insert points from each student, finish with '-1':\n");

while (!all\_points\_inserted) {

(void)scanf("%d", &pointlist[len]);

if (pointlist[len] >= 0) {

len++;

} else if (pointlist[len] == -1) {

all\_points\_inserted = 1;

} else {

(void)printf("Points must be > 0!\n");

}

}

//read points needed for Mark 6

(void)printf("Insert points needed for a 6:\n");

(void)scanf("%d", &points\_6);

//calculate Statistic, print Statistic and decide, if rerun the

//calculation with new number needed for Mark 6.

do {

statistics = get\_statistics(pointlist, len, points\_6);

(void)print\_statistics(statistics);

(void)printf("Enter new minimal points for grade 6 (y/n?)");

(void)scanf("%s", &rerun);

if (rerun == 'y') {

(void)scanf("%d", &points\_6);

}

} while (rerun == 'y');

return EXIT\_SUCCESS;

}

/\*

\*\* function to set all values (students, points\_6, best\_mark,

\*\* worst\_mark, average\_mark, mark\_6, mark\_5, mark\_4, mark\_3, mark\_2,

\*\* mark\_1, passed) in struct Statistic.

\*\*

\*/

Statistics get\_statistics(int pointlist[], int len, int points\_6) {

Statistics statistics = {len, points\_6};

int grades[len];

int mark6 = 0;

int mark5 = 0;

int mark4 = 0;

int mark3 = 0;

int mark2 = 0;

int mark1 = 0;

int best\_mark =1;

int worst\_mark=6;

double average\_mark;

//safe Mark for each Students in a List

for(int i = 0; i < len; i++) {

grades[i] = get\_mark(pointlist[i], points\_6);

}

//count, how often each Mark occurs and safe value of best and

//worst occurring Mark.

for(int i = 0; i < len; i++) {

best\_mark = best\_mark > grades[i] ? best\_mark : grades[i];

worst\_mark = worst\_mark < grades[i] ? worst\_mark : grades[i];

switch(grades[i]) {

case 1: mark1++;

break;

case 2: mark2++;

break;

case 3: mark3++;

break;

case 4: mark4++;

break;

case 5: mark5++;

break;

case 6: mark6++;

}

}

//calculate the average Mark

average\_mark = (1\*mark1 + 2\*mark2 + 3\*mark3 + 4\*mark4 +

5\*mark5 + 6\*mark6)/(double)len;

//set all values in struct Statistics

statistics.best\_mark = best\_mark;

statistics.worst\_mark = worst\_mark;

statistics.average\_mark = average\_mark;

statistics.mark\_1 = mark1;

statistics.mark\_2 = mark2;

statistics.mark\_3 = mark3;

statistics.mark\_4 = mark4;

statistics.mark\_5 = mark5;

statistics.mark\_6 = mark6;

statistics.passed = mark6 + mark5 + mark4;

return statistics;

}

/\*

\*\* function to calculate the Mark of a Student depending on th number of   
\*\* Points.

\*/

int get\_mark(int points, int points\_6) {

double mark = 1 + ((5.0\*points)/points\_6);

if (mark > 6.0) {

mark = 6.0;

}

if ((mark - (int)mark) > 0.5) {

mark = ceil(mark);

} else {

mark = floor(mark);

}

return (int)mark;

}

/\*

\*\* function to print all values from struct Statistic

\*\*

\*/

void print\_statistics(Statistics statistics) {

(void)printf("--------------------------------------------------------\n");

(void)printf("Statistics (%d students, %d points needed for mark 6):\n", statistics.students, statistics.points\_6);

(void)printf("Mark 6: %d\n", statistics.mark\_6);

(void)printf("Mark 5: %d\n", statistics.mark\_5);

(void)printf("Mark 4: %d\n", statistics.mark\_4);

(void)printf("Mark 3: %d\n", statistics.mark\_3);

(void)printf("Mark 2: %d\n", statistics.mark\_2);

(void)printf("Mark 1: %d\n\n", statistics.mark\_1);

(void)printf("Best mark: %d\n", statistics.best\_mark);

(void)printf("Worst mark: %d\n", statistics.worst\_mark);

(void)printf("Average mark: %.2f\n", statistics.average\_mark);

(void)printf("Mark >= 4: %d students %d Percent\n", statistics.passed, (statistics.passed\*100)/statistics.students);

(void)printf("--------------------------------------------------------\n");

}