Notenstatistik

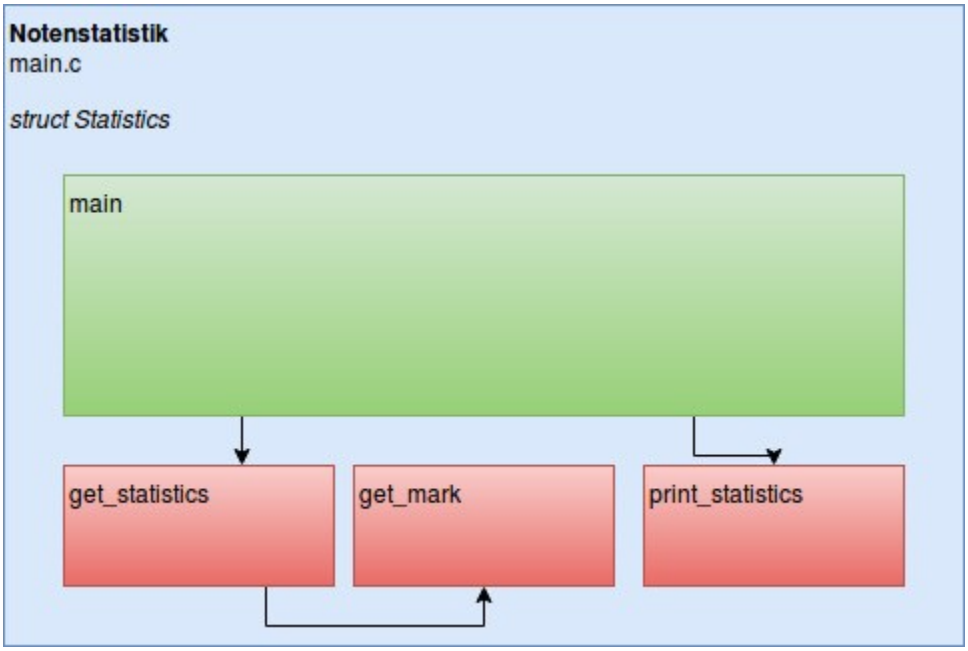
Zusammenfassung der Problemstellung

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 relativ 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_mainausgabe: prüft, ob die Berechnung wie erwartet auf der Konsole dargestellt wird.
test_get_mark: prüft die Funktion get_mark auf verschiedene Werte und stellt sicher, dass die Noten anhand der Punktzahl richtig berechnet werden.
test_get_statistics: prüft die Funktion get_statistics und stellt sicher, dass die Statistiken korrekt gespeichert werden.

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.

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");
}
```