Allgemeines

Siehe Testat 0

Aufgabe 6: Strukturen

Gegeben ist eine Datenstruktur für eine Parabel (bzw. Polynom zweiten Grades, $ax^2 + bx + c$), die in der Datei `testat-6.h` deklariert ist:

Listing 1: Parabel-Header-Datei (testat-6.h)

```
1  #ifndef testat_6
2  #define testat_6 1
3
4  struct parabel {
5      double a;
6      double b;
7      double c;
8  };
9
10 int scheitelhoehe(struct parabel * p, double *y);
11 // Wenn p->a != 0 (Parabel) dann wird in *y die Scheitelhoehe geschrieben und 0 zurueckgege
12 // Wenn p->a == 0 (Gerade) dann wird *y nicht geschrieben und 1 zurueckgegeben
13
14
15 void sort_parabeln(struct parabel * p, int n);
16 // Sortiert das Feld nach Scheitelhoehe aufsteigend
17 // Geraden werden hinter die Parabeln eingefuegt (ohne weitere Sortierung)
18
19 #endif
```

Entwickeln Sie zwei Funktionen:

a) Scheitelhöhe

```
int scheitelhoehe(struct parabel * p, double *y);
```

Der erste Parameter stellt eine Referenz auf **genau eine Datenstruktur** dar. Berechnet wird in `* y` die Scheitelhöhe der Parabel `*p`, falls `*p` eine Parabel ist, also `p->a` $\neq 0$. In diesem Fall (also bei einer Parabel) wird 0 zurückgegeben, andernfalls (also bei einer Geraden) 1. Bei einer Geraden darf in y von Ihrer Funktion nichts reingeschrieben werden, da es keine Scheitelhöhe gibt. (vgl. testat-6.h in den Kommentaren).

b) Sortierung

```
void sort_parabeln(struct parabel * p, int n);
```

In diesem Fall wird als Parameter eine Referenz auf **ein Feld von Datenstrukturen** übergeben. Die Methode sortiert das übergebene Feld nach folgenden Kriterien:

- Parabeln werden nach Scheitelhöhe aufsteigend (von kleiner Scheitelhöhe zu großer Scheitelhöhe) in das Feld einsortiert
- Geraden werden im Feld hinter die Parabeln eingefügt

Die Prototypen der Funktionen sind in der Datei `testat-6.h` deklariert, welche wie in der vorgegebenen Beispieldatei `testat-6.c` (siehe später) eingebunden werden kann mittels include.

Spielen Sie eine C-Datei mit dem Namen `<matrikel-nr>-testat-6.c` (also bspw. 12345-testat-6.c) in moodle hoch. Diese enthält die zwei Funktionen. Zum Sortieren dürfen Sie entweder einen beliebigen Sortieralgorithmus implementieren oder den Quicksort (qsort) der Standard-Bibliothek aufrufen.

- Die gesamte Kommunikation der Funktionen muss über die Schnittstelle erfolgen.
- Innerhalb der Funktion sind keine Ein- und Ausgaben erlaubt.
- Es sind keine globalen Variablen erlaubt.
- Die Methode scheitelhoehe darf zum Sortieren verwendet werden.
- Listing 1 und Listing 2 sind in Moodle verfügbar.
- Abgabe ist 24.05.2019 8:00 Uhr
- Gruppenarbeit in einer Gruppenstärke von max. 3 Personen ist für dieses Testat erlaubt. Die Namen aller beteiligten Personen muss in der abzugebenden Datei ganz oben in einem Kommentar vermerkt werden. Jeder Student lädt eine eigene Datei mit der Lösung der Gruppe hoch. Die Angabe der Gruppenmitglieder verhindert einen Plagiatsverdacht.
- Sie dürfen Listing 2 von moodle in den üblichen Dateinamen `<matrikel-nr>-testat-6.c` umbenennen, die zwei Funktionen implementieren und abgeben. Die Abgaben werden nur in den beiden zu implementierenden Funktionen auf Plagiatsverdacht geprüft.
- Sie geben nur die C-Datei ab, nicht die Header-Datei.
- Es wird aber empfohlen, noch **weitere Tests** in das Hauptprogramm einzubauen. Das Hauptprogramm wird zwar nicht mitgetestet, darf aber nicht zu Compiler-Warnings oder -Fehlern führen.

Ein mögliches Hauptprogramm könnte sein:

Listing 2: Parabel-Hauptprogramm (testat-6.c)

```c
#include <stdio.h>
#include "testat-6.h"

int scheitelhoehe(struct parabel * p, double *y) {
    int rc = 0;
// ...
    return rc;
}

void sort_parabeln(struct parabel * p, int n) {
    // ...
}

int main() {
    struct parabel p[] = {
        {1,2,3},
        {2,5,-19},
        {0,0,0},
        {-1,0,0}
    };
    double y;
    printf("Index 0 ist eine echte Parabel: %d\n", scheitelhoehe(p, &y) == 0);
    sort_parabeln(p, sizeof(p) / sizeof(struct parabel));
    return 0;
}
```