

Schriftliche Prüfung Programmieren 1

M.Gerstner, H.Kristl,
J.Plate, K-G.Rauh

Teil 2

- Bearbeitungszeit: 90 Minuten
- zugelassene Hilfsmittel:
beliebige schriftliche,
keine Rechner

Gutes Gelingen!



fachhochschule

munich university of applied sciences

münchen

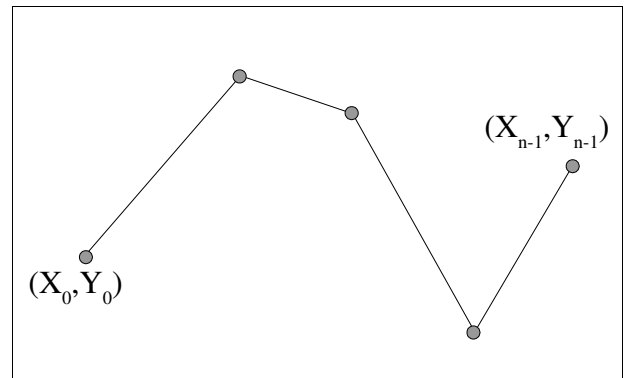
Name, Vorname
Semester
Raumnummer, Platznummer

1. Polygonzug [33 Punkte]

Ein **Polygonzug** in der Ebene ist durch n Punkte (X_i, Y_i) definiert, mit $i = 0, 1, \dots, n-1$.

Die **Länge** eines Polygonzugs berechnet man durch Aufsummieren der $n-1$ Längen seiner Teilstücke. Die Länge des Teilstücks von (X_i, Y_i) nach (X_{i+1}, Y_{i+1}) ergibt sich dank Pythagoras als

$$L_i = \sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2}$$



Die **Fläche unter** dem Polygonzug ist die Summe der Teilflächen. Die Teilfläche unter dem Geradenstück zwischen den Punkten (X_i, Y_i) und (X_{i+1}, Y_{i+1}) erhält man dank Trapezformel als

$$F_i = 0.5 * (Y_{i+1} + Y_i) * (X_{i+1} - X_i)$$

Aufgabe: Schreiben Sie ein Programm in der Sprache C. Es soll zunächst vom Benutzer die gewünschte Anzahl n der Punkte erfragen, dann die Koordinatenpaare (X_i, Y_i) einlesen, anschließend die Länge und die Fläche berechnen und ausgeben.

Hinweis: Um "statische" C-Arrays verwenden zu können, darf vorausgesetzt werden, dass ein Polygonzug höchstens 256 Punkte enthält. Eine Überprüfung, ob der Eingabewert das Maximum überschreitet, ist (selbstverständlich) Aufgabe des Programmierers!

2. Würfelsimulation [23 Punkte]

Wie lange muss man würfeln, bis zwei Sechser hintereinander kommen? Diese Frage soll mittels Computersimulation geklärt werden. Hierzu sollen Sie die Definition der folgenden beiden Unterrouتين in der Programmiersprache C schreiben.

- Die Funktion *eine_runde()*, die solange würfelt (d.h. ganze Zufallszahlen von 1 bis 6 erzeugt), bis zwei Sechser hintereinander kommen. Die Anzahl der Würfe (eine ganze Zahl) wird zurückgeliefert.
- *simuliere()* erhält als Parameter die vom Benutzer gewünschte Anzahl an Runden - genau so oft wird dann die Funktion *eine_runde()* aufgerufen. Am Ende wird der Durchschnitt (eine Fließkommazahl) der Ergebnisse von *eine_runde()* ausgegeben.

Hinweis: Die C-Funktion "*int rand(void)*" liefert eine Zufallszahl größer oder gleich 0. Diese Zufallszahl muss in Ihrem Programm dann (per Integer-Arithmetik) auf den Bereich von 1 - 6 abgebildet werden.

3. Beschränkte Liste [37 Punkte]

In einer Binärdatei sind Daten abgelegt, deren Datensätze eine einheitliche Struktur aufweisen. Die Deklaration dieser Struktur lautet

```
struct satz_t;
```

Die genaue Definition ist hierbei nicht von Interesse und soll für diese Aufgabe absichtlich verborgen bleiben.

Die Datensätze sollen nacheinander eingelesen und in dynamisch allokiertem Speicher abgelegt werden. Die Funktion *readDyn()* liest den nächsten Datensatz der geöffneten Datei, sorgt für dynamischen Speicher und deponiert die Daten dort. Zurückgeliefert wird ein Zeiger auf diesen Speicher – oder NULL, falls der Vorgang nicht erfolgreich war (z.B. weil kein weiterer Datensatz mehr vorhanden war oder kein Speicher beschafft werden konnte) Der Prototyp lautet:

```
struct satz_t * readDyn(FILE*);
```

Teilaufgabe 1: Schreiben Sie eine Definition der Funktion *readDyn()*.

Die **fünf** zuletzt eingelesenen Werte sollen in einer linearen Liste abgelegt werden. Ein neuer Datensatz ersetzt dabei den jeweils ältesten in der Liste. Als Verwaltungs-Struktur bietet sich an:

```
struct list_t {  
    struct satz_t *pNutz;  
    struct list_t *pNext;  
};
```

Zur Verwaltung der Liste selbst wird für die Wurzel folgende Struktur verwendet:

```
struct rootInfo_t {  
    struct list_t * pRoot;  
    int cntRoot;  
};
```

Dabei ist *pRoot* ein Zeiger auf das *zuletzt eingetragene* Element der Liste. *cntRoot* enthält die aktuelle Anzahl von Elementen in der Liste.

Die Funktion *putIn()* hat als Aufgabe, den Datensatz von *readDyn()* in die Liste einzuhängen. Dabei hat sie zu berücksichtigen, wie viele Elemente die Liste enthält. Gegebenenfalls ist der älteste Eintrag (komplett!) zu entfernen.

Teilaufgabe 2:

- Welche Übergabeparameter und welchen Rückgabewert sollte *putIn()* besitzen? schreiben Sie den Prototyp auf.
- Zeigen Sie exemplarisch, wie man *putIn()* in Verbindung mit einem Aufruf von *readDyn()* beispielsweise in *main()* verwenden könnte.
- Realisieren Sie die Funktion *putIn()*

