Prozedurale Programmierung – Übung 11

WS 2018/19 *Prof. Dr. F.J. Schmitt*

Hochschule **Rosenheim**
University of Applied Sciences

In der Community wird ein (fast leeres) Projekt in der Datei „polygon.zip“ bereitgestellt.

Aufgabe 1

Entwickeln Sie ein Programm zur Berechnung der Fläche eines regelmäßigen Polygons und der Abweichung der Fläche von der eines Kreises mit dem Radius des Umkreises des Polygons. Es soll eine Tabelle auf dem Bildschirm ausgeben, die wie folgt aussieht:

```
------------------------------------------------
Flaeche regelmaessiger Polygone
------------------------------------------------
Anzahl Ecken    Seitenlaenge    Flaeche         Abweichung zu Kreisflaeche
  3                 5.54          13.30                18.87
  4                 4.53          20.48                11.69
  5                 3.76          24.35                 7.82
  6                 3.20          26.60                 5.57
  7                 2.78          28.02                 4.15
  8                 2.45          28.96                 3.21
  9                 2.19          29.62                 2.55
 10                 1.98          30.09                 2.08
 11                 1.80          30.45                 1.72
 12                 1.66          30.72                 1.45
 13                 1.53          30.93                 1.24
 14                 1.42          31.10                 1.07
 15                 1.33          31.24                 0.93
```

Die minimale und maximale Anzahl Ecken sowie der Umkreisradius sollen über scanf() vom Benutzer eingegeben werden. Die Formeln zur Berechnung lauten:

- Seitenlänge: $l = 2r \sin(\pi/n)$
- Fläche des Polygons: $A = nl^2 / (4 \tan(\pi/n))$
- Kreisfläche A_K: $A_K = r^2\pi$

Anmerkungen:

- Definieren Sie einen neuen Datentyp mainParameter_t (eine struct, mit typedef) in dem die eingegebenen Werte gesammelt gespeichert werden.
- Definieren Sie einen neuen Datentyp polygon_t (eine struct, mit typedef) in der die berechneten Daten eines einzelnen Polygons gespeichert werden (also: Anzahl Ecken, Fläche, Seitenlänge, Abweichung der Fläche von der Kreisfläche).
- Definieren Sie in main() ein Feld vom Typ polygon_t (mit einer vordefinierten Maximalgröße), in dem die Daten der Tabelle komplett gespeichert werden können.
- Schreiben Sie eine Funktion berechnePolygone(), die das Feld und die eingegeben Parameter übergeben bekommt und das Feld mit Daten befüllt. Der Prototyp soll so aussehen: `void berechnePolygone(polygon_t daten[], mainParameter_t params);`
- Schreiben Sie eine Funktion AusgabePolygone(), die das Feld und die Feldgröße übergeben bekommt und die Tabelle auf dem Bildschirm ausgibt. Der Prototyp soll so aussehen: `void AusgabePolygone(polygon_t daten[], int anzZeilen);`

Aufgabe 2

Ändern Sie das Programm nun so ab, dass das Feld vom Typ polygon_t nicht mehr lokal mit vordefinierter Maximalgröße angelegt wird, sondern mit dynamischer Speicherverwaltung genau in der erforderlichen Größe.

Aufgabe 3

Nun soll das Programm so geändert werden, dass die Parameter nicht mehr mit scanf() eingegeben werden, sondern über die Kommandozeile und argv an main(). Das Programm soll aufgerufen werden mit:

```
polygon -emin <MINECKEN> -emax <MAXECKEN> -r <UMKREISRADIUS>
```

also für die obige Tabelle z.B. mit:

```
polygon -emin 3 -emax 15 -r 3.2
```

Es soll möglich sein die Kommandozeilenparameter in beliebiger Reihenfolge einzugeben.

Anmerkungen:

- Schreiben Sie eine Funktion scanCommandLine(), die als Parameter argc und argv übergeben bekommt. Zurückgegeben werden sollen die in der Kommandozeile übergebenen Werte für emin, emax und r sowie ein Fehlercode. Der Fehlercode gibt an, ob die Parameter im zulässigen Bereich liegen und ist bereits in funktionen.h definiert (Typ: cl_errors_t). Alle dort enthaltenen Fehler sollen auch überprüft werden. Im Fall mehrerer Fehler soll nur der als letzter detektierte zurückgeliefert werden. Der Prototyp soll so aussehen:
  ```
  cl_errors_t scanCommandLine(int argc, char **argv, mainParameter_t *params);
  ```
- Schreiben Sie eine Funktion printCLError(), die als Parameter den Rückgabewert von scanCommandLine() übergeben bekommt und einen evtl. vorhandenen Fehler an den Benutzer meldet. Im Fehlerfall soll hier auch das Programm beendet werden. Der Prototyp soll so aussehen:
  ```
  void printCLError(cl_errors_t error);
  ```