Strukturierte Programmierung Übung 7

Auszuarbeiten bis 17.12.20

1. Summenbildung (2 Punkte)

Entwickeln Sie eine C-Funktion, die die Werte zwischen 1 und einer Obergrenze n zusammenzählt und als Ergebnis zurückliefert.

2. Summenbildung – Erweiterung 1 (2 Punkte)

Entwickeln Sie auf der Basis von Aufgabe 1 einen C-Funktion, die nur die geraden Zahlen, also jene, die ganzzahlig durch 2 ganzzahlig geteilt werden können, zusammengezählt und das Ergebnis zurückliefert.

3. Summenbildung – Erweiterung 2 (2 Punkte)

Entwickeln Sie auf der Basis von Aufgabe 1 einen C-Funktion, die nur die Zahlen zwischen einer Untergrenze m und einer Obergrenze n zusammenzählt und das Ergebnis zurückliefert.

Beispiel: m = 5, n = 7

summe = 5 + 6 + 7

4. Bestimmen des Maximums (2 Punkte)

Gegeben ist ein Feld von Ganzzahlenwerten. Die Werte werden mittels einer Initialisierungsliste festgelegt d.h. int values[] = { 3, 5, 10, 2, 1 };

Entwickeln Sie eine C-Funktion, die den größten Wert bestimmt und als Ergebnis zurückliefert

5. Reihenentwicklung der Exponentialfunktion (6 Punkte)

Die Exponentialfunktion e^z soll mit Hilfe einer Reihenentwicklung berechnet werden.
Schreiben Sie ein Programm in C, welches die ersten N Glieder der Reihenentwicklung

$$e^z = \underbrace{1}_{1.\text{Glied}} + \underbrace{\frac{z}{1!}}_{2.\text{Glied}} + \underbrace{\frac{z^2}{2!}}_{3.\text{Glied}} + \ldots + \underbrace{\frac{z^{N-1}}{(N-1)!}}_{\text{N.Glied}} + \ldots$$

aufsummiert. z soll *reell* und N soll *ganzzahlig* sein.

Hinweis: Die Terme der Reihe sind in einer Schleife aufzusummieren. Dabei soll die Fakultät in der Schleife selbst berechnet werden (keine Bibliotheksfunktion!).

6. Näherungsweise Berechnung der Quadratwurzel (6 Punkte)

Die Quadratwurzel $y = \sqrt{x}$ kann mit dem Newtonschen Näherungsverfahren für $x \geq 0$ wie folgt berechnet werden:

$y_0 = x/2$ und $y_n = 1/2 (y_{n-1} + x / y_{n-1})$, für $n > 0$

Gesucht ist ein C-Programm, das mit Hilfe des Näherungsverfahrens eine Lösung für die Quadratwurzel von x liefert, sodass gilt $| y_n - y_{n-1} | < eps$

Achten Sie auf den sinnvollen Einsatz von Funktionen.

1 Summenbildung

Lösungsidee:
Mit einer Schleife alle Werte zwischen 1 und der angegebener oberen Grenze zusammenzählen.

Testfälle: 4, 0, 1, 2

C-Program code & Test Ausgaben --> Siehe beigelegtes .c file.

2 Summenbilden – Erweiterung 1

Lösungsidee:
Mit einer Schleife alle geraden Werte, die durch 2 ganzzahlig geteilt werden können, zwischen 1 und der angegebenen oberen Grenze zusammenzählen.

Testfälle: 8, 0, 1, 2

C-Program code & Test Ausgaben --> Siehe beigelegtes .c file.

3 Summenbildung – Erweiterung 2

Lösungsidee:
Mit einer Schleife alle Werte zwischen angegebener unteren Grenze und oberen Grenze zusammenzählen.

Testfälle: (5,7), (-5, 7), (-3, -1), (5, 1), (0, 0)

C-Program code & Test Ausgaben --> Siehe beigelegtes .c file.

4 Bestimmen des Maximums

Lösungsidee:
Zuerts werden die Übergabeparameter überprüft. Die länge der Arrays wird mit ‚#define' definiert. Die ‚max' Variable wird mit dem ersten Wert vom array intialisiert damit die ‚max' Variable einen richtigen Anfangswert hat. Dannach wird mit der vordefinierten länge vom Array durch das Array iteriert und in jeder Iteration wird überprüft ob es ein neuen max Wert gibt.

Testfälle: { 3, 5, 10, 2, 1 }, { -10, -1, -4, -9, -57 }, { 5 }, {}

C-Program code & Test Ausgaben --> Siehe beigelegtes .c file.

5 Reihenentwicklung der Exponentialfunktion

Lösungsidee:
Zuerts werden die Übergabeparameter überprüft. Die Reihe wird dann anhand der gegebenen Formel berechnet. Die Fakultät wird innerhalb der Funktion bestimmt und weitergeführt.

Testfälle: (5.00, 7), (1.00, 0), (22.45, 4), (0.00, 10), (-1.00, 10)

C-Program code & Test Ausgaben --> Siehe beigelegtes .c file.

6 Näherungsweise Berechnung der Quadratwurzel

Lösungsidee:
Mit dem Heron-Verfahren erhält man eine schrittweise Annäherung an die gesuchte Wurzel Zahl. Dabei wird zuerst ein Schätzwert hergenommen (wurzel/2) und dannach immer weiter verfeinert. Das Verfahren wird abgebrochen, wenn die gewünschte Genauigkeit erreicht ist.

Testfälle: (0.00002, 5), (-0.00002, 2), (0.000002, 0), (0.0001, 126), (1, 126)

C-Program code & Test Ausgaben --> Siehe beigelegtes .c file.