Algorithmen und Datenstrukturen
Versuch 3

VORBEREITUNGSAUFGABEN

- Erstellen Sie das Struktogramm (Nassi-Shneiderman) für die Funktionen `main` und `heron` wie in Aufgabe 3a verlangt wird.

AUFGABENSTELLUNG

- Es ist ein Programm zu erstellen, das die Quadratwurzel einer positiven Ganzzahl errechnet.
  Es wird dazu das Iterationsverfahren von Heron verwendet.

$$Result_{i+1} = \frac{Result_i + \frac{n}{Result_i}}{2}$$

  Die Iteration ist zu beenden, wenn der Unterschied von $Result_{i+1}$ zu $Result_i$ eine zuvor festgelegte Genauigkeit unterschreitet.
- Zur Ermittlung des absoluten Betrags eines `double`-Wertes ist eine geeignete Funktions namens `myAbs` in der Datei `heron.c` zu erstellen.
  Hinweis: Diese wird für die Ermittlung der aktuellen Genauigkeit benötigt!
- Zur Ermittlung der Quadratwurzel ist die Funktion
  `double heron(unsigned n);`
  in der Datei `heron.c` zu erstellen.
  Der übergebene Wert stellt das Funktionsargument dar, für das die Quadratwurzel errechnet werden soll.
  Der ermittelte Wert soll vom Typ `double` sein.
- Implementierungshinweise für `heron`:
  - Die Funktion verwendet zur Berechnung zwei Variablen *`resultOld`*. und *`result`*.
  - *`resultOld`* wird mit dem Wert von `n` initialisiert, *`result`* mit dem Startwert 1.
  - In einer geeigneten Schleife wird
    - zuerst das Resultat der letzten Berechnung (Wert von *`result`*) in *`resultOld`* gesichert und
    - dann der nächste Näherungswert gemäß der oben dargestellten Formel errechnet.

    Die Schleife wird beendet, wenn der Unterschied der beiden Werte die Genauigkeit von `1e-5` (Makroname: `EPSILON`) unterschreitet.
  - Für jeden Iterationsdurchlauf der Schleife soll ein Iterationszähler mit den beiden Werten von `resultOld` und `result` ausgegeben werden.
    Beispiel für `n==0`:

```
Iteration 1 : 1.000000 -> 0.500000
Iteration 2 : 0.500000 -> 0.250000
Iteration 3 : 0.250000 -> 0.125000
...
```

- Die Einsprungfunktion `main` soll den Text `"Eingabe von n: "` ausgeben, danach mithilfe von `scanf` den Wert erfragen; sollte `scanf` einen gültigen Wert eingelesen haben, dann ist die Funktion `heron` mit dem eingegebenen Zahlenwert aufzurufen und das Resultat geeignet auszugeben.
  Dies ist so lange zu wiederholen bis kein gültiger Zahlenwert eingegeben wird.
  Hinweis: `scanf` liefert die Anzahl der korrekt eingelesenen Werte als Funktionswert.
- Die Funktion `main` ist in der Datei `versuch3.c` zu implementieren.

Algorithmen und Datenstrukturen
Versuch 3

Aufgabe 3a

VERSUCHSDURCHFÜHRUNG

- Melden Sie sich mit Ihrem Account an und öffnen Sie ein Terminal-Fenster.
- Erstellen Sie zuerst das Verzeichnis `versuch3` in Ihrem Heimatverzeichnis (`/home/`*`Benutzername`*).
- Verwenden Sie den Editor `gedit` zum Erstellen der Datei `heron.c` im eben erstellten Verzeichnis.
- Implementieren Sie eine Funktion `myAbs` in ANSI-C 89/90, die den absoluten Betrag des Funktionsarguments liefert.
- Setzen Sie den oben dargestellten Algorithmus in ANSI-C 89/90 innerhalb der Funktion
  `double heron(unsigned);`
  um. Verwenden Sie darin Ihre Funktion `myAbs`.
- Verwenden Sie für die Ausgabe von `double`-Variablenwerte in der `printf`-Funktion den Formatspezifizierer `"%f"`.
- Erstellen Sie die Headerdatei `heron.h` in der die Funktionsprototypen von `heron.c` und `EPSILON` zu finden ist.
  Es soll sichergestellt werden, dass der Inhalt der Headerdatei nur einmal innerhalb einer C-Datei inkludiert wird.
- Implementieren Sie die Funktion `main` in der Datei `versuch3.c` gemäß den obengenannten Vorgaben.
- Erstellen Sie ein ausführbares Programm.
- Falls Ihre Umsetzung (mit den Compiler-Schaltern "`-Wall -pedantic -ansi`") Warnungen oder Fehler erzeugt, korrigieren Sie Ihren Quellcode.
- Lassen Sie sich Ihr Struktogramm und Ihr lauffähiges Programm von Ihrem Betreuer abnehmen.

Aufgabe 3b

Die Ausgaben während der Iteration sollen nun nur noch für den Entwicklungsprozess ausgegeben werden. Dazu sollen Präprozessoranweisungen zur **bedingten Compilierung** verwendet werden.
Vergleichen Sie außerdem Ihr Ergebnis mit der Ausgabe des Ergebnisses der Standard-C Funktion `sqrt`.

VERSUCHSDURCHFÜHRUNG

- Ergänzen Sie Ihre Funktion `heron`, dass Quellcode, der für die Iterationsausgabe verantwortlich ist, **nur dann übersetzt wird**, wenn die symbolische Konstante **`DEBUG`** definiert wurde.
- Übersetzen Sie Ihren Quellcode mit der Definition von `DEBUG` im Aufruf von `gcc`.
- Testen Sie Ihr Programm sowohl **mit** als auch **ohne** der Definition von `DEBUG`.
- Ergänzen Sie die Funktion `main` dahingehend, dass zusätzlich der Funktionswert von `sqrt` ausgegeben wird.
- Für die Berechnung der Wurzel steht Ihnen die Funktion
  `double sqrt(double);`
  zur Verfügung, die in der Includedatei **`math.h`** vereinbart und in der Bibliothek **`m.lib`** implementiert ist.
- Falls Ihre Umsetzung (mit den Compiler-Schaltern "`-Wall -pedantic -ansi`") Warnungen oder Fehler erzeugt, korrigieren Sie Ihren Quellcode.
- Binden Sie Ihre ausführbare Datei mit dem Schalter "`-lm`", um die Funktion `sqrt` aus der mathematischen Bibliothek hinzuzufügen.
- Lassen Sie sich Ihr lauffähiges Programm von Ihrem Betreuer abnehmen.