Ingenieurinformatik (FK 03)
Übung 4

VORBEREITUNG

Erstellen Sie das Struktogramm der Funktion **`trapez`** für die Übung 4b mithilfe des Programms **`Structorizer`**.

ÜBUNG 4A (FUNKTIONEN)

Einführung

Eine **Funktion** ist ein Teil eines Programms der aus (ggf. mehreren) anderen Programmteilen heraus aufgerufen werden kann.
Es ist möglich, Werte (**Parameter**) an Funktionen zu übergeben und entweder

einen Rückgabewert (eines bestimmten Typs) oder
keinen Rückgabewert (**`void`**-Funktion oder auch Prozedur genannt)

an den Aufrufer zurückzugeben.

Eine Funktion bearbeitet in der Regel eine einzelne, konkrete (Teil-)Aufgabe.
Beispielsweise dient die Funktion **`printf`** zur Ausgabe auf dem Bildschirm, die Funktion **`scanf`** zur Eingabe von der Tastatur oder **`sqrt`** berechnet die Quadratwurzel eines übergebenen **`double`**-Wertes.
Wird ein Wert zurückgegeben kann der Aufrufer (abhängig von der Aufgabenstellung) diesen Wert weiterverarbeiten

```
anzahl=printf("Hallo %d\n", wert);
```

oder auch nicht

```
printf("Hallo %d\n", wert);
```

Bei der Programmierung von bzw. mit Funktionen unterscheidet man:

- **Funktionsdeklaration**

Dies geschieht durch Funktionsprototypen, z. B.

```
double myPow(float, unsigned);
```

Bekanntmachen der Existenz einer Funktion durch Festlegen des Typs der Rückgabe (**`double`**), des Funktionsnamens (**`myPow`**) und der Anzahl bzw. Typ der Parameter (2 Parameter - 1. Parameter vom Typ **`float`** und 2. vom Typ **`unsigned`**).
Dies dient nur als Hinweis für den Übersetzer, wenn nachfolgend im Quellcode der Funktionsname auftaucht.

- **Funktionsdefinition**

Dies geschieht durch Implementierung der Funktion, z. B.

```
double myPow(float wert, unsigned exp)
{
   double produkt=1;
   unsigned i;
   for (i=0; i<exp; i++)
        produkt=produkt*wert;
   return produkt;
}
```

Beschreibung des Aussehens einer Funktion ("Funktionsdeklaration") und Realisierung des Funktionsablaufes (z. B. Algorithmus der Problemstellung).
Merke: Ein Funktionsdefinition ist gleichzeitig eine Funktionsdeklaration, nicht aber umgekehrt!

- **Funktionsaufruf**

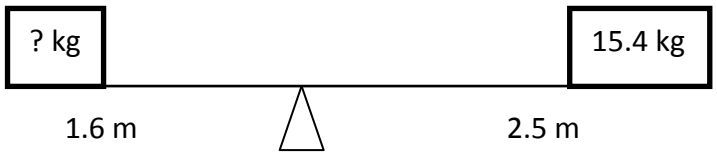
  Eine Funktion kann von beliebig vielen Stellen im Programm (mit unterschiedlichen Parameterwerten) aufgerufen werden (vgl. **printf**) , z. B.

```
...
double wert;
...
wert=myPow(4.5, 3);
printf("4.5^3   = %f\n", wert);
printf("4.5^3^5 = %f\n", myPow(wert, 5));
...
```

  Ein Funktionsaufruf ist als "Unterprogramm" zu verstehen. Es wird die Realisierung der Funktion ausgeführt und nach Beendigung wird der Programmablauf des Aufrufers an der Aufrufstelle fortgesetzt.

Ziel

Es soll das vorgegebene Beispiel **ueb4a.c** analysiert und modifiziert werden. Danach soll der Quellcode um die Funktion **multDiv** ergänzt werden, um ein typisches Dreisatzproblem zu lösen:



Der Funktionsaufruf **multDiv(15.4, 2.5, 1.6)** soll als Rückgabewert die Masse des Körpers links liefern, bei dem die Waage im Gleichgeicht ist.

Durchführung

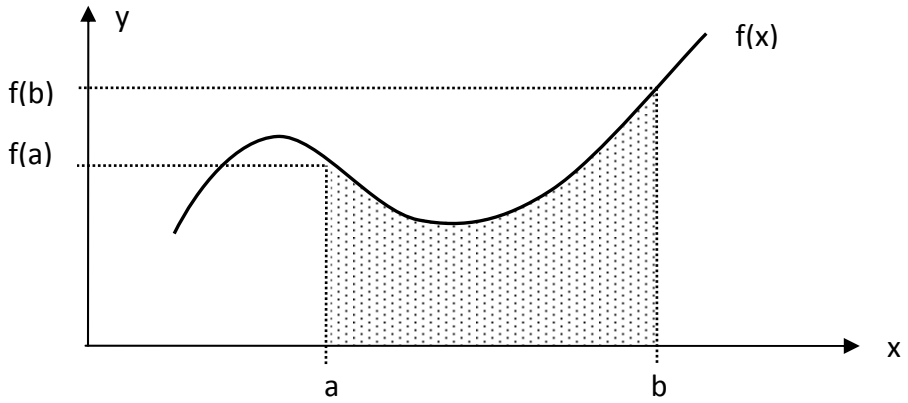
- Starten Sie das Programm **QT Creator**.
- Erstellen Sie in diesem Arbeitsbereich ein **C-Projekt** mit dem Namen **uebung4** unter dem Verzeichnis **U:\workspace**
- Kopieren Sie nun die bereitgestellte Datei **ueb4a.c** in das Verzeichnis **U:\workspace\uebung4** mithilfe des Windows-Explorer.
- Fügen Sie die Datei dem Projekt hinzu und entfernen die bereitgestellte Datei **main.c**.
- **Frage A1:** Wo befindet sich im Quellcode der Funktionsaufruf, die Funktionsdeklaration und Funktionsdefinition von **addiere2fach**?
  Ergänzen Sie den Quellcode um geeignete Kommentare, welche die Fragen beantworten.
- Erstellen Sie das Programm, führen es mit dem Debugger im Einzelschritt (*Step Over – Einzelschritt über*) aus und beobachten Sie die Änderungen der Variablen.
  Sollten Sie mit dem Zeilenmarker in der Zeile
  erg = addiere2fach(x, y);
  sein, dann springen Sie in die Funktion mit *Step-Into* (*Einzelschritt herein* – Abkürzungstaste: F11).
- Benennen Sie die Variable **param1** in **x** und **param2** in **y** um.
- Erstellen Sie den aktualisierten Code und wiederholen Sie den Debug-Vorgang.
- **Frage A2:** Ändert sich etwas an der Ausgabe des Programms? Was passiert mit dem Variablenwert von **x** in der Funktion **main**?
- Machen Sie aus **addiere2fach** eine reine Verarbeitungsfunktion gemäß dem EVA-Prinzip.
- Ergänzen Sie Ihren Quellcode um die Anweisung
  printf("Zweite Berechnung: %d\n", addiere2fach(15, 25));
  und führen Sie das erneut Programm aus.

- Erweitern Sie das Programm um die Funktion **`multDiv`**.
  - Überlegen Sie zuerst wieviele Parameter nötig sind, welchen Datentyp diese besitzen und welchen Datentyp der Rückgabewert der Funktion besitzt.
  - Ergänzen Sie den Quellcode um die Funktionsdeklaration.
  - Überlegen Sie wie die Parameter miteinander mathematisch verknüpft werden müssen, um die Lösung des o. g. Problems zu erhalten.
  - Implementieren Sie die Funktion **`multDiv`**.
  - Rufen Sie die Funktion in **`main`** auf und geben den Rückgabewert der Funktion **`multDiv`** geeignet auf dem Bildschirm aus.

ÜBUNG 4B (NUMERISCHE INTEGRATION MITHILFE DES TRAPEZVERFAHRENS)

Einführung

Es ist ein Programm zu entwickeln, dass eine beliebige Funktion numerisch integrieren kann.



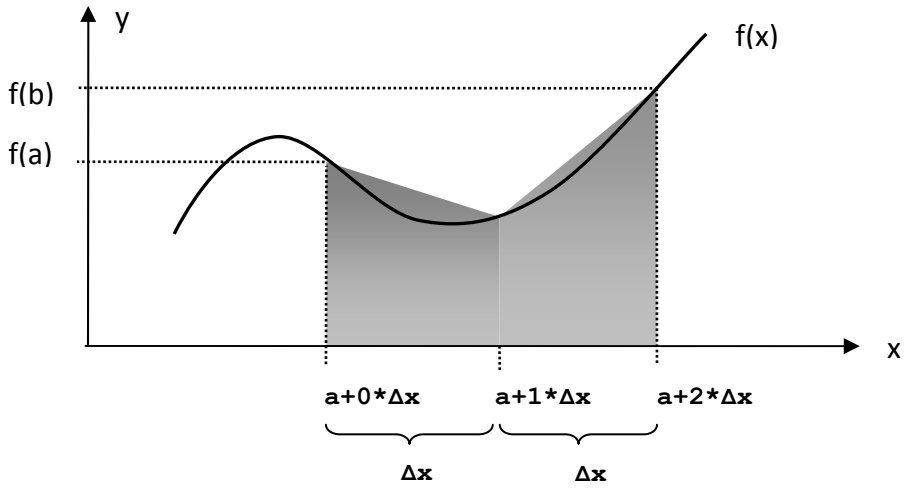
Gegeben:

Integrand	**`y = f(y)`**
Integrationsgrenzen	**`a, b`**

Anforderung:

- Ändert sich der Integrand, dann soll eine Änderung nur an einer Stelle im Quellcode nötig sein.
- Die Genauigkeit der Berechnung soll einstellbar sein.
- Es soll abschließend überprüft werden, ob der Algorithmus richtig arbeitet.

Um die Fläche unter dem Graphen näherungsweise zu bestimmen wird der Bereich der Integrationsgrenzen zu gleichen Teilen segmentiert und die Fläche jedes Segments berechnet und aufaddiert. Nachfolgend eine grafische Darstellung für 2 Segmente (**`n=2`**):



Je höher die Anzahl der Segmente (**n**), desto kleiner **Δx** und damit auch der Fehler der Flächenberechnung.
Der Wert **Δx** und die Fläche errechnet sich damit wie folgt:

$$\Delta x = \frac{b-a}{n}$$

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n-1}\left(\frac{f(a+i\cdot\Delta x)+\ f(a+(i+1)\cdot\Delta x)}{2}\cdot\Delta x\right) =$$

$$\left(\frac{f(a+\mathbf{0}\cdot\Delta\boldsymbol{x})+\ f(a+\mathbf{1}\cdot\Delta\boldsymbol{x})}{2}\cdot\Delta x\right)+\left(\frac{f(a+\mathbf{1}\cdot\Delta\boldsymbol{x})+\ f(a+\mathbf{2}\cdot\Delta\boldsymbol{x})}{2}\cdot\Delta x\right)+\ \ldots+$$
$$\left(\frac{f(a+(\boldsymbol{n}-\mathbf{1})\cdot\Delta\boldsymbol{x})+\ f(a+\boldsymbol{n}\cdot\Delta\boldsymbol{x})}{2}\cdot\Delta x\right)$$

Ziel

Implementieren Sie die fehlende Funktion:

```
double trapez(double a, double b, int n);
```

trapez integriert eine Funktion **f(x)** nach dem Trapezverfahren im Bereich von **a** bis **b**. Es werden insgesamt **n** Teilflächen berechnet und aufsummiert.
Tipp: Während der Berechnung der Gesamtfläche muss der Funktionswert **f(x)** an verschiedenen Stellen **x** berechnet werden.
Dazu ist im vorgegebenen Quellcode folgende Funktion definiert:

```
double f(double x);
```

Der **x**-Wert wird als Parameter übergeben, der Wert **y = f(x)** wird als Rückgabewert zurückgegeben.

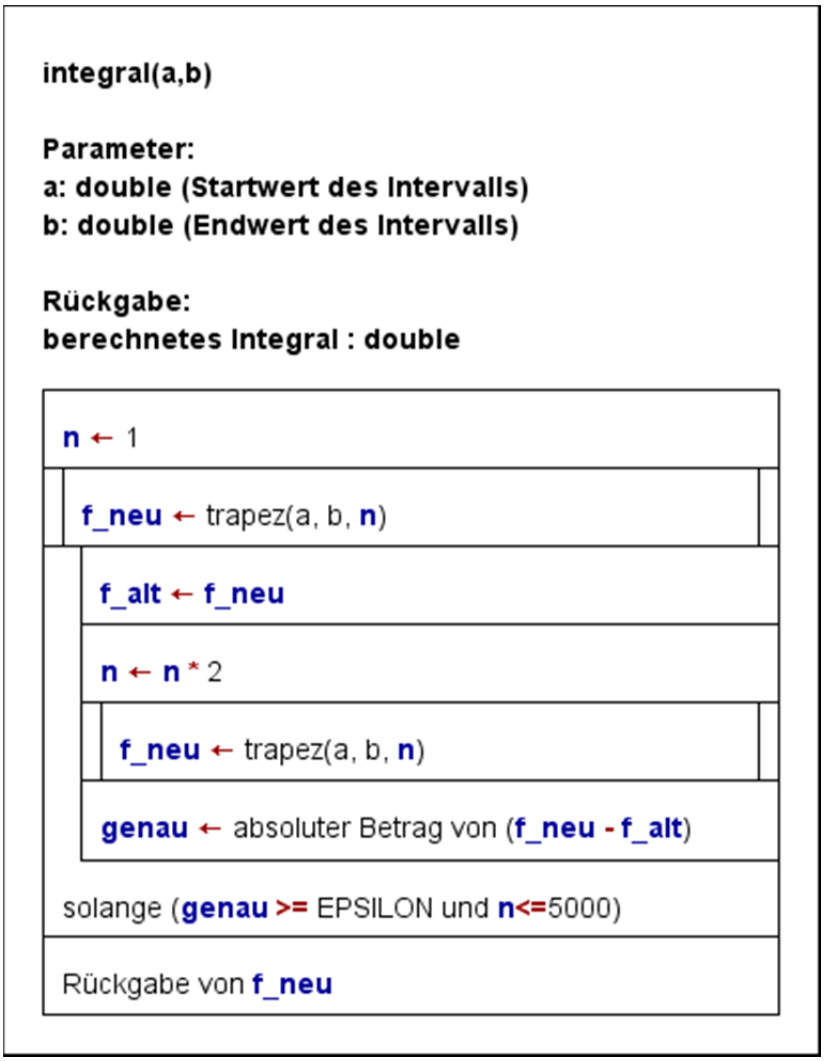
Durchführung

- Schließen Sie ggf. bereits geöffnete Projekte und erstellen Sie ein neues **C-Projekt** mit dem Namen **integration**.
- Kopieren Sie nun die bereitgestellte Datei **integration.c** in das Verzeichnis **U:\workspace\integration** mithilfe des Windows-Explorer.
- Fügen Sie die Datei dem Projekt hinzu und entfernen die bereitgestellte Datei **main.c**.
- Realisieren Sie die Funktion **trapez** gemäß den Vorgaben und Ihres vorbereiteten Struktogramms.
- Erstellen Sie das Programm **integration.exe**.
- Starten Sie das Programm über die Eingabeaufforderung (*Command Prompt*) indem Sie die ausführbare Datei im Verezichnisbaum unterhalb **U:\workspace** suchen und ausführen.
- Überprüfen Sie, ob Sie für $F = \int_{\mathbf{0}}^{\mathbf{2\pi}} \mathbf{sin^2}\,\boldsymbol{x}\,\boldsymbol{dx}$ als Ergebnis den Wert von **π** erhalten?
- **Frage B1:** Ändern Sie Ihr Programm geeignet ab, um $F = \int_{\mathbf{0}}^{\mathbf{3}} \boldsymbol{x^2 dx}$ zu ermitteln. Wie lautet das Ergebnis?
- Testen Sie Ihr Programm noch für unterschiedliche Grenzwerte.

Zusatzaufgabe (optional)

Erweitern Sie das numerische Integrationsprogramm, so dass die Anzahl der Teilflächen (n) nicht mehr fest eingestellt ist. Stattdessen soll die Anzahl solange immer weiter verdoppelt werden, bis die Ergebnisse der Funktion **`trapez`** „genau genug“ sind.

Erstellen Sie dazu eine zusätzliche Funktion **`integral`** gemäß dem nachfolgenden Struktogramm, welche die bereits bestehende Funktion **`trapez`** aufruft.
Vergleichen Sie die Ergebnisse für **`n`** und (**`2*n`**) Teilflächen und beenden Sie das Verfahren, wenn der relative Unterschied beider Ergebnisse eine vorgegebene Grenze (**`EPSILON`**) unterschreitet oder die Anzahl der Teilflächen (**`n`**) größer als 5000 ist.



- **Hinweis:** Zur Ermittlung des absoluten Betrags einer **`double`**-Variablen steht Ihnen die Funktion

  ```
  double fabs(double);
  ```

  die in **`math.h`** deklariert ist zur Verfügung.
- Implementieren Sie die Funktion **`integral`** und rufen diese in **`main`** (anstelle von **`trapez`**) auf.
- Testen Sie Ihr Programm noch für unterschiedliche Integranden und Grenzwerte.