Programmiertechnik IKA 9/13 – Übungsblatt 26.03.2014 / Teil 1

Funktionstest

Erstellen Sie eine neue leere Projektmappe mit der Bezeichnung „IhrNachname_Funktionen“, also z:B: „Wemheuer_Funktionen“.

Legen Sie in dieser Projektmappe EIN neues Projekt an mit dem Namen „Funktionen“. ALLE Aufgaben sind in diesem einen Projekt in EINEM Programm nacheinander zu lösen. Gehen Sie die jeweils nächste Aufgabe aber erst an, wenn die bisherigen Aufgaben fehlerfrei kompiliert und ausgeführt werden. Schreiben Sie im Programmcode an den jeweiligen Stellen zwischen den Aufgaben einen entsprechenden Kommentar hin.

Tipp: Damit beim erneuten Kompilieren und Ausführen nicht alle bisherigen Programmteile / Aufgaben noch einmal durchlaufen werden müssen, können Sie die bis dahin erstellten Programmteile des Hauptprogramms zeitweise auskommentieren durch Verwendung von „/*“ und „*/“. Vergessen Sie aber nicht, diese Kommentare im fertigen Endprogramm wieder aufzuheben...

Aufgabe 1:

Schreiben Sie eine Funktion, die:

- nichts entgegennimmt,
- nichts zurückliefert
- den Text „Hallo Welt!“ mit anschließendem Zeilenumbruch auf dem Bildschirm ausgibt.

Den Namen der Funktion wählen Sie selbst.
Rufen Sie die Funktion im Hauptprogramm auf.

Aufgabe 2:

Schreiben Sie eine Funktion, die:

- einen Float-Wert entgegennimmt,
- den Wert verdoppelt und zurückgibt

Rufen Sie die Funktion im Hauptprogramm mit dem literalen („hartcodierten“, von Ihnen im Programm fest vorgegebenen) Wert „123.45“ auf und geben das Ergebnis am Bildschirm aus.

Aufgabe 3:

Schreiben Sie eine Funktion, die:

- einen Buchstaben entgegennimmt,
- nichts zurückgibt,
- auf dem Bildschirm den Text „Der Buchstabe lautet: “ und den übergebenen Buchstaben mit anschließendem Zeilenumbruch ausgibt.

Deklarieren Sie im Hauptprogramm eine Zeichen-Variable und initialisieren sie mit einem Buchstaben Ihrer Wahl, rufen Sie anschließend die Funktion mit dieser Variablen auf.

Aufgabe 4:

(a) Schreiben Sie eine Prozedur („Subroutine“, also eine Funktion ohne Rückgabewerte), die:

- auf dem Bildschirm alle Sonderzeichen mit den ASCII-Werten 210 bis 220 in einer Zeile mit anschließendem Zeilenumbruch anzeigt.

Den Namen der Prozedur wählen Sie selbst, rufen Sie die Prozedur im Hauptprogramm einmal auf.

Hinweis: Um einen ASCII-Wert **a** als Buchstabe anzuzeigen, verwenden Sie die Schreibweise:

```
cout << char(a);
```

(b) Schreiben Sie eine weitere ähnliche Prozedur, die ebenfalls ASCII-Zeichen anzeigt, der Sie nun aber den ersten und den letzten anzuzeigenden ASCII-Wert als Parameter (VON, BIS) übergeben.

Aufgabe 5:

Schreiben Sie eine Funktion namens „produkt“, die:

- 3 Ganzzahlen entgegen nimmt,
- das Produkt der drei Zahlen zurückgibt.

Rufen Sie die Funktion im Hauptprogramm mit drei beliebigen literalen („hartcodierten“, von Ihnen im Programm fest vorgegebenen) Werten auf und geben das Ergebnis am Bildschirm aus.

Aufgabe 6:

Im Hauptprogramm möge der Benutzer drei Zahlen eingeben, mit vorheriger Eingabeaufforderung „Bitte geben Sie drei Ganzzahlen ein:“. Rufen Sie die Funktion aus Aufgabe 5 nun mit diesen Werten auf und geben das Ergebnis am Bildschirm aus.

Aufgabe 7:

Schreiben Sie eine Funktion namens „groesser10“, die:

- einen Ganzzahlwert entgegennimmt,
- einen bool-Wert zurückliefert:
    - true, wenn die Ganzzahl größer als 10 ist,
    - false, wenn nicht

Rufen Sie die Funktion 5 mal im Hauptprogramm auf und geben jeweils das Ergebnis mit einem geeigneten Text Ihrer Wahl am Bildschirm aus:

1. mit einem literalen (im Funktionsaufruf stehenden) Wert, der garantiert true zurückliefert
2. mit einem literalen Wert, der garantiert false zurückliefert
3. mit einer Variablen, die Sie an dieser Stelle im Programm erst deklarieren und mit dem Wert „123“ initialisieren
4. weisen Sie dieser Variablen nun einen Wert kleiner als 10 zu und rufen die Funktion erneut auf.
5. Lassen Sie den Benutzer nun dieser Variablen einen neuen Wert zuweisen und rufen die Funktion erneut auf.

Die Anweisungen, um den Text, z.B. „Die Zahl ist größer als 10“, auszugeben, können Sie bei den fünf Funktionsaufrufen beispielsweise mit „Copy&Paste“ weitestgehend übernehmen.

Programmiertechnik IKA 9/13 – Übungsblatt 26.03.2014 / Teil 2

Aufgabe 8:

Sie können ohne Bedenken davon ausgehen, dass alle Programmteile, die Sie mit „Copy&Paste“ an mehreren Stellen verwenden, besser in einer Funktion oder Prozedur aufgehoben sind. Bei Programmierprofis klingeln bei Verwendung der Tastenkombination „Strg-C / Strg-V“ sämtliche Alarmglocken…

Schreiben Sie deshalb eine Funktion namens „ZeigeGroesser10an“ – mit Prototyp – die:

- einen bool-Wert entgegennimmt
- den Text „Zahl ist größer als 10“ am Bildschirm ausgibt, wenn der bool-Wert true ist,
- den Text „Zahl ist kleiner als 11“ am Bildschirm ausgibt, wenn der bool-Wert false ist

Rufen Sie nun die Funktion „groesser10“ aus Aufgabe 7 mit einem literalen Wert auf, der garantiert **true** zurückgibt und speichern das Ergebnis des Funktionsaufrufs in einer neuen Variable namens „MeinBoolWert“. Rufen Sie anschließend die hier erstellte Funktion „ZeigeGroesser10an“ mit diesem Variablenwert auf.

Rufen Sie die Funktion „groesser10“ aus Aufgabe 7 mit einem literalen Wert auf, der garantiert **false** zurückgibt. Speichern Sie das Ergebnis aber nicht, sondern verwenden den Rückgabewert direkt als Übergabeparameter für die nun aufzurufende Funktion „ZeigeGroesser10an“. Das klingt komplizierter als es ist und müsste so aussehen:

```
ZeigeGroesser10an(groesser10(8));
```

Funktioniert? Prima!

Aufgabe 9:

Kopieren Sie die Definition der in Aufgabe 8 erstellten Funktion „ZeigeGroesser10an“ und nennen die Funktion „PrintGroesser“.

Erweitern Sie die Parameterliste um einen zu übergebenden String und ersetzen in der Funktions-definition überall, wo das Wort „Zahl“ vorkommt, diesen Text durch den übergebenen String.
Damit die String-Übergabe funktioniert, sollten Sie die Bibliothek <string> in Ihr Projekt aufnehmen.

a) Rufen Sie die Funktion nun auf mit den Übergabeparametern true und der Zeichenkette „Produkt“. Als Ergebnis sollten Sie auf dem Bildschirm sehen: **`Produkt ist größer als 10`**
b) Definieren Sie nun eine Bool-Variable mit dem Wert false und einen String mit dem Wert „Ergebnis“. Rufen Sie die Funktion „PrintGroesser“ nun mit diesen Werten auf. Auf dem Bildschirm müsste nun stehen: **`Ergebnis ist kleiner als 11`**
c) In Aufgabe 7-5 hat der Benutzer eine Zahl eingegeben. Verwenden Sie diese Zahl, um eine Anzeige „Die verwendete Zahl ist kleiner als 11“ zu erzeugen
(oder “…größer als 10“, je nachdem…).

Aufgabe 10 (die finale Bewährungsprobe):

- Lassen Sie den Benutzer drei Ganzzahlen eingeben und zeigen das Produkt der drei Zahlen am Bildschirm an (siehe Aufgaben 5 und 6)
- Geben Sie in der Zeile darunter an, ob das soeben berechnete Produkt größer als 10 ist oder nicht (siehe Aufgaben 7 bis 9)

Jetzt sind wir (fast) fertig!

Letzte Aufgabe:

Wir stellen unsere Lösungen anderen Anwendern zur Verfügung, indem wir ALLE erzeugten Funktionen sowie DAS GESAMTE Hauptprogramm in einer eigenen Header-Datei veröffentlichen:

- Erstellen Sie in der Projektmappe ein neues Projekt namens „Funktionstest“.
- Legen Sie im neuen Projekt eine neue leere Header-Datei an und nennen sie „Funktionstest.h“.
- Kopieren Sie ALLES aus Ihrer bisherigen Lösungsdatei in diese Header-Datei.
- Benennen Sie in der Header-Datei die main-Funktion um in „Funktionstest“.
- Inkludieren Sie nun Ihre eben erstellte Header-Datei „Funktionstest.h“.
- Rufen Sie in der main-Funktion nun Ihre in der Headerdatei vorhandene Funktion „Funktionstest“ auf.

Legen Sie Ihr neues Projekt als Startprojekt fest und kompilieren es.

Klappt‘s? Prima!

Ein allerletztes Update:

Modifizieren Sie den Code in Ihrer Header-Datei so, dass zwischen den Bildschirmausgaben zu den einzelnen Aufgaben jeweils eine zusätzliche Leerzeile und eine Überschrift angezeigt wird, etwa so:

```
------------ Aufgabe 5: ------------
```

Verwenden Sie dazu eine Funktion. Ob und – wenn ja – welchen Übergabeparameter Sie dafür benötigen oder ob ein Rückgabewert nötig ist, legen Sie nun selbst fest.

Im Hauptprogramm ändern Sie gar nichts, sondern kompilieren einfach das Projekt „Funktionstest“ neu. Erstellen Sie sowohl eine Debug- als auch eine Release-Version und vergleichen Sie die Dateigrößen der beiden erzeugten ausführbaren Dateien:

Dateigröße Debug-Version: _____

Dateigröße Release-Version: _____