Übungen Ing.-Inf. – KE 3

Der C-Coding Styleguide ist einzuhalten!

Folgende Einstellungen sind für Debug und Release (All Configurations) vorzunehmen:

Einstellung	**Wert**
Solution Platform	x86
Properties->Conf. Properties->C/C++->General->Warning Level	Level3 (/W3)
Properties->Conf. Properties->C/C++->General->Treat Warnings As Errors	No (/WX-)
Properties->Conf. Properties->C/C++->General->SDL checks	No (/sdl-)
Properties->Conf. Properties->C/C++->Code Generation->Basic Runtime Checks	Default
Properties->Conf. Properties->C/C++->Code Generation->Security Checks	Disable Security Checks (/GS-)

Packen Sie das gesamte Verzeichnis der Solution in eine Zip-Datei und laden Sie diese in Moodle pünktlich hoch.

Aufgabe 1:
Implementieren Sie ein Programm, welche die Funktionen *CalcVolumeSphere* und *CalcAreaCircle* implementiert. Deklarieren Sie die Funktionen oberhalb von *main* und definieren Sie die Funktionen unterhalb von *main*. Überlegen Sie sich die Datentypen für den Übergabeparameter und den Rückgabewert. Verwenden Sie für π eine eigene globale konstante Variable. Definieren und initialisieren Sie in *main* eine lokale Variable für den Radius. Übergeben Sie diese Variable an die Funktionen. Die beiden Rückgabewerte speichern Sie in weiteren zwei Variablen ab. Geben Sie anschließend den Wert dieser Variablen auf dem Bildschirm aus.

Die Gleichungen zur Berechnung des Kugelvolumens und der Kreisfläche lautet:

$$V = \frac{4}{3} \cdot \pi \cdot r^3$$

$$A = \pi \cdot r^2$$

Testen Sie Ihr Programm mit den folgenden Werten für den Radius: -1, 0, 1, 100. Rufen Sie viermal nacheinander Ihre beiden Funktionen auf und übergeben jeweils die einzelnen obigen Werte als literale Konstanten. Der Datentyp der literalen Konstante muss mit dem Datentyp des Übergabeparameters übereinstimmen. Stimmen Ihre implementierten Gleichungen?
Ein Radius von -1 soll nach Definition nicht möglich sein. Warum fällt dies bei der Berechnung von A nicht sofort auf?

Erstellen Sie ein Strukturdiagramm Ihrer Anwendung und speichern Sie dieses als pdf im Sourceverzeichnis des Projektes ab.

Übungen Ing.-Inf. – KE 3

Aufgabe 2:
Implementieren Sie ein Programm, welches neben *main* noch eine weitere Funktion *int TestGlobal(void)* enthält. Deklarieren Sie *TestGlobal* über *main* und definieren Sie diese nach *main*. Definieren Sie eine globale Variable iVar und initialisieren Sie diese mit dem Wert 42. Definieren Sie in *TestGlobal* eine lokale Variable auto int iVar. Initialisieren Sie diese Variable mit dem Wert 72. Führen Sie anschließend in *TestGlobal* ein Postinkrement von iVar durch und geben Sie gleich danach in *TestGlobal* den Wert von iVar auf der Konsole aus. Die Funktion *main* soll einmal *TestGlobal* aufrufen und anschließend den Wert iVar sowie den Rückgabewert von *TestGlobal* auf der Konsole ausgeben.

Welche Variable verwendet Ihr Programm in *main* und in *TestGlobal*?

Entfernen Sie nun das Schlüssel auto und verifizieren Sie, dass sich im Programmverhalten nichts ändert. Ab jetzt kann auf das optionale Schlüsselwort **auto** bei lokalen Variablen verzichtet werden.

Formulieren Sie als Kommentar über *main* die folgende Gesetzmäßigkeit:
„Bei namensgleichen Variablen (Namenskonflikt) verwendet der Compiler die Variable mit dem ……. Gültigkeitsbereich!“

Hinweis: Auf globale Variablen soll in den folgenden Übungen – falls diese nicht explizit gefordert werden – verzichtet werden.

Aufgabe 3:
Implementieren Sie ein Programm, welches nur *main* als Funktion enthält. Lesen Sie in *main* ein Zeichen von der Tastatur mittels *int _getch()* ein. Diese Funktion ist in der Bibliothek conio.h enthalten, welche nicht zum ANSI-Standard gehört. Die Funktion *_getch()* gibt im Gegensatz zu getchar das eingelesene Zeichen nicht auf der Konsole aus. Geben Sie anschließend das eingelesene Zeichen und dessen ASCII-Code in einer neuen Zeile aus (zwei unterschiedliche Formatierer verwenden und die Variable zweimal nach der Zeichenkette von *printf* angeben).
Lesen Sie anschließend erneut ein zweites Zeichen mit *_getch()* ein und löschen Sie anschließend mit *_clrscr()* den Bildschirm. Dazu ist die statische Bibliothek Utilities zu verwenden, welche Sie in Moodle finden. Geben Sie anschließend das Zeichen auf der Konsole mit *_gotoxy()* aus. *_gotoxy()* setzt den Cursor an die entsprechende Stelle. Alle weiteren Ausgaben mit *printf* setzen an dieser Stelle fort. Die x- und y-Koordinate soll dabei die letzte Ziffer des ASCII-Codes des zweiten Zeichens sein.
Beispiel: ASCII-Code ist 63 -> x- und y-Koordinate ist 3.
Dazu benötigen Sie einen wichtigen Operator, der in einer der vorherigen KE besonders hervorgehoben wurde.

Verwendung der Utililities.lib:

- Zipdatei im Sourcecodeverzeichnis entpacken
- Utilities.h in Main.c inkludieren
- Utilities.lib hinzufügen: Properties -> Configuration Properties -> Linker -> Input-> AdditionalDependencis: Hier für All Configurations vor kernel32.lib noch *Utilities.lib;*

Hinweis: Die Utilities.lib wurde Ihnen als Debug-Version zur Verfügung gestellt. Beim Übersetzen mit Release werden zwei Warnungen generiert. Diese können Sie ignorieren.

Übungen Ing.-Inf. – KE 3

Aufgabe 4:
Implementieren Sie ein Programm, welches die Variablen dValA, dValB und dValC von der Tastatur mit scanf_s einliest.
Danach soll das Programm dValD mit folgender Gleichung berechnen:

$$dValD = dValA * \sqrt{dValB - dValC}$$

Recherchieren Sie im Internet, welche Funktion Sie in C für die Wurzel verwenden müssen. Welche Bibliothek müssen Sie inkludieren?

Geben Sie anschließen dValD auf der Konsole aus.

Erkennen Sie unter welcher Bedingung die Berechnung der Wurzel nicht funktioniert? In diesem Fall wird auf der Konsole eine Zeichenkette mit den Buchstaben „nan“ ausgegeben. Was könnte dies bedeuten?