Übungen Ing.-Inf. – KE 4

Der C-Coding Styleguide ist einzuhalten!

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

Einstellung	**Wert**
Solution Platform	x86
Properties->Conf. Properties->C/C++->General->Warning Level	Level4 (/W4)
Properties->Conf. Properties->C/C++->General->Treat Warnings As Errors	Yes (/WX)
Properties->Conf. Properties->C/C++->General->SDL checks	Yes (/sdl)
Properties->Conf. Properties->C/C++->Code Generation->Basic Runtime Checks	Default
Properties->Conf. Properties->C/C++->Code Generation->Security Checks	Enable Security Checks (/GS)

Solution muss in Debug und Release fehlerfrei kompilierbar sein.

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, welches die folgenden Kontrollstrukturen enthält:

1.1 Einfache Selektion: Lesen Sie von der Tastatur einen Wert für die Variable iVarA ein. Ist dieser Wert größer als 7, dann geben Sie „Wert von iVarA ist groesser als 7“ auf der Konsole aus.

1.2 Zweifache Selektion: Lesen Sie von der Tastatur einen Wert für die Variable iVarB ein. Ist dieser Wert größer als 7, dann geben Sie „Wert von iVarB ist groesser als 7“ auf der Konsole aus. Ist der Wert kleiner oder gleich 7, dann geben Sie „Wert von iVarB ist kleiner oder gleich 7“ auf der Konsole aus.

1.3 Mehrfache Selektion: Lesen Sie von der Tastatur einen Wert für die Variable iVarC ein. In Abhängigkeit von iVarC soll iVarD die folgenden Werte annehmen:

iVarC	iVarD
0	-5
1	4711
2	5
3	3
4	73
Ansonsten	42

Geben Sie anschließend den Wert von iVarD auf der Konsole aus.

1.4 Mehrfache Selektion mit Fall-through: Lesen Sie von der Tastatur einen Wert für die Variable iVarC ein. In Abhängigkeit von iVarC soll iVarD die folgenden Werte annehmen:

iVarC	iVarD
0	-99
1	-99
2	3
3	4
4	5
Ansonsten	42

Verwenden Sie dazu **drei** Fall-through. Geben Sie anschließend der Wert von iVarD auf der Konsole aus.

1.5 Kopfgetestete Schleife: Implementieren Sie nacheinander zwei kopfgetestete Schleifen, die einen Schleifenzähler iCounter von 0 bis 9 hochzählen (for- und while-Schleife). Geben Sie den Schleifenzähler im Schleifenrumpf und nach der Schleife aus. Die Schleifenrümpfe sollen zehnmal durchlaufen werden.

1.6 Endegetestete Schleife: Implementieren Sie eine endegetestete Schleife die einen Schleifenzähler iCounter von 0 bis 9 hochzählt. Geben Sie den Schleifenzähler im Schleifenrumpf und nach der Schleife aus. Der Schleifenrumpf soll zehnmal durchlaufen werden.

Aufgabe 2:
Implementieren Sie ein Programm, welches drei unsigned int Werte (Tag, Monat, Jahr) von der Tastatur einliest. Ihr Programm soll überprüfen, ob es sich um ein korrektes Datum handelt. Eine selbst zu erstellende Funktion *int IsLeapYear(unsigned int uiYear)* ist hier zu empfehlen. Ebenso sollten Sie anhand des Monats erst die maximale Anzahl an Tagen im Monat ermitteln (switch-Kontrollstruktur, dabei Fall-through verwenden).

Für Jahr soll nur ein Wert größer oder gleich 1900 akzeptiert werden.
Für Monat soll nur ein Wert im Bereich 1-12 akzeptiert werden.
Für Tag soll nur ein Wert im Bereich 1-uiMaxDaysInMonth akzeptiert werden. uiMaxDaysInMonth soll anhand des Monats ermittelt werden. Für Februar ist dabei zu bestimmen, ob es ein Schaltjahr ist oder nicht.

Testen Sie Ihr Programm mit den folgenden Werten:

Testfall #	Tag	Monat	Jahr	Ergebnis
1	1	1	1899	Jahr nicht korrekt
2	1	0	1900	Monat nicht korrekt
3	1	13	1900	Monat nicht korrekt
4	0	1	1900	Tag nicht korrekt
5	32	1	1900	Tag nicht korrekt
6	31	4	1900	Tag nicht korrekt
7	29	2	1900	Tag nicht korrekt
8	29	2	2100	Tag nicht korrekt
9	29	2	2000	Datum korrekt
10	31	12	2400	Datum korrekt
11	27	1	2019	Datum korrekt

Bestimmen Sie mit Testwell CMT v(G), LOCpro und mittels eigener Rechnung c% Ihrer Funktionen und schreiben Sie diese Werte als Kommentar in main.c.

Übungen Ing.-Inf. – KE 4

Aufgabe 3:
Implementieren Sie ein Programm, welches die folgenden beide quadratischen Figuren jeweils mittels zweier Schleifen und einer zweifachen Selektion auf dem Bildschirm ausgibt. Die Breite (und damit auch die Höhe der Figuren) ist vorher von Tastatur einzulesen und soll einen ungeraden Wert haben.

Figur 1
```
ABBBB
AABBB
AAABB
AAAAB
AAAAA
```

Figur 2
```
ABBBB
AABBB
AAABB
AABBB
ABBBB
```

Aufgabe 4:
Implementieren Sie ein Programm, welches die folgenden mathematischen Ausdrücke iterativ (mittels Schleife) und rekursiv berechnet. Implementieren Sie hierzu vier zusätzliche Funktionen.

unsigned long long int CalcFacultyIterativ(unsigned int uiN);
unsigned long long int CalcFacultyRecursiv(unsigned int uiN); // siehe Script
unsigned long long int CalcSumIterativ(unsigned int uiN);
unsigned long long int CalcSumRecursiv(unsigned int uiN);

Dabei ist n vor dem Aufruf der Funktionen einmal über die Tastatur einzugeben. Die vier berechneten Ergebnisse sollen auf der Konsole ausgegeben werden. Der Formatierer für unsigned long long int lautet: %llu

$$n! = \begin{cases} 1, & n = 0 \\ \prod_{i=1}^{n} i = 1 \cdot 2 \cdot .. \cdot n, & n > 0 \end{cases} \qquad n, i \in N_o \quad n \geq i$$

$$sum(n) = \sum_{i=0}^{n} i = 0 + 1 + .. + n \qquad n, i \in N_0 \quad n \geq i$$