Übungen Ing.-Inf. – KE 9

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	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 den Namen einer Textdatei mit scanf_s einliest. Danach wird diese Textdatei im Textmodus zum Lesen geöffnet. Lesen Sie zeilenweise den Inhalt der Datei ein und geben Sie die Zeilen auf dem Bildschirm aus.
Verwenden Sie dabei die Funktion fgets. Besonderheit dieser Funktion ist, dass das CRLF aus der Datei ebenso eingelesen wird und nachher in der Ergebniszeichenkette enthalten ist.

Aufgabe 2:
Implementieren Sie ein Programm, welches den Namen einer csv-Textdatei mit scanf_s einliest. Danach wird diese Textdatei im Textmodus zum Lesen geöffnet. Lesen Sie zeilenweise den Inhalt der Datei ein und extrahieren Sie jeweils die drei Tokens einer Zeile. Geben Sie die drei Tokens in jeweils einer Zeile aus.

```
Sheldon;Cooper;PhD
Hofstadter;Leonard;PhD
Wolowitz;Howard;MSc
Koothrappali;Rajesh;PhD
```

Schreiben Sie anschließend in die Datei “NerdsRead.txt” die Anzahl der eingelesenen Datensätze mit einem „PhD“ als drittem Token.

Aufgabe 3:
Erstellen Sie ein neues Programm mit dem folgenden Programmcode.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

#define DOSHOWOUTPUT 1
// Multiple lines in a function macro: / is must be used at the end of the line
#define SMALLER10(Value1,Value2) (int)((((int)(Value1) + 10) < (int)(Value2)) ? \
                                  (1) : (0))

int main(void)
{
   int iX = 0;
   int iY = 10;
```

```
   printf("Hello World!\n");

#if DOSHOWOUTPUT
   printf("Hello user!\n");
#endif

   if (SMALLER10(iX, iY))
   {
      printf("First Value + 10 is smaller than second Value!\n");
   }

   // Wait for user interaction
   _getch();

      return EXIT_SUCCESS;
}
```

Erweitern Sie den Code mit einem Funktionsmakro aus dem Skript.

Danach gehen Sie wie folgt vor:

- Rechter Mouseklick auf das Projekt im Solution Explorer
- Properties anwählen
- Configuration Properties -> C/C++ -> Preprocessor anwählen
- Preprocess to a File auf "Yes (/P)" setzen
- Drücken Sie danach auf OK, damit die Einstellungen gespeichert sind.

Compilieren Sie jetzt erneut!

- Welcher Dateityp entsteht jetzt beim Übersetzen des Programms? (Hinweis: Schauen Sie in dem Verzeichnis nach, wo vormals die exe-Datei hinterlegt wurde: Debug oder Release).
- Erklären Sie, warum beim Übersetzen ein Linker-Fehler entsteht?
- Öffnen Sie die erstellte Datei. Erklären Sie, was aus der ursprünglichen Zeile `#include <conio.h>` umgewandelt wurde.
- Was ist mit den Kommentaren passiert und warum?
- Ändern Sie den Wert des defines `DOSHOWOUTPUT` auf 0. Was hat sich jetzt bei einem einem erneuten Präprozessordurchlauf geändert?
- Wie wird die Codegenerierung abhängig von #defines genannt?

Schreiben Sie die Antworten als Kommentar über die Funktion main.c.

Aufgabe 4:
Implementieren Sie ein Programm, welches die folgenden drei Funktionsmakros enthält:

- SWAP(iX,iY): Tauscht die Inhalte von iX und iY unter Verwendung einer weiteren Variablen, die innerhalb des Makros definiert ist (mehrzeiliges Makro)
- ISODD(iX): Gibt eine 1 zurück, wenn die Zahl ungerade ist, ansonsten eine 0 (Bitweise Operatoren)
- GETSECONDBYTE(uVal): Gibt das zweite Byte einer unsigned int Variablen zurück (Bitweise Operatoren).

Testen Sie jedes Funktionsmakro mittels dreier Aufrufe in main.