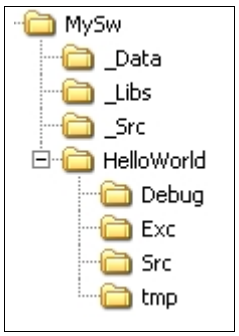
Einstellungen bei der Arbeit mit MS Visual Studio Professional 2017 (VS2017):



Die u.a. Vorgehensweise erzeugt ein VS-Projekt (intern geführt unter dem Namen Debug), das unterhalb eines bereits bestehenden Verzeichnisses [~\MySw\ ]HelloWorld eine Verzeichnisstruktur nutzt und

- die Projekt-Dateien unter HelloWorld\Debug,
- das lauffähige Test.exe unter HelloWorld\Exc,
- alle C/C++ - Quellen unter HelloWorld\Src,
- temporäre Dateien unter HelloWorld\tmp ablegt.

Das so implementierte Projekt nutzt (evtl. gemeinsam mit anderen Projekten)

- Daten unter [~\MySw\]_Data,
- Bibliotheken unter [~\MySw\]_Libs und
- Quelldateien unter [~\MySw\]_Src.

Test.exe kann sowohl von der Entwicklungsumgebung als auch von HelloWorld\Exc aus (Doppelklick) gestartet werden und in beiden Fällen zur Laufzeit auf Daten im _Data-Verzeichnis zugreifen. Aktuelles Arbeitsverzeichnis ist HelloWorld\Debug.

Bereits compilierte Quellen (*.obj) können z.B. ins Verzeichnis [~\MySw\]_Libs kopiert und per Drag&Drop im Projektmappen-Explorer von VS2017 als Quellcodedateien eingebunden werden. Der Linker berücksichtigt sie auch ohne Vorhandensein des Quellcodes.

Die Unterverzeichnisse von HelloWorld können auch als Grundlage für weitere Projekte verwendet werden – etwa für ein Projekt Hello2. Dazu sollte man

- das Verzeichnis HelloWorld2 erzeugen,
- die 4 Unterverzeichnisse von HelloWorld kopieren,
- jene unter den kopierten Dateien löschen, die man für Hello2 nicht verwenden möchte.

Download-Seite von Microsoft für Visual Studio:
https://www.visualstudio.com/de/?rr=https%3A%2F%2Fblogs.msdn.microsoft.com%2Fvcblog%2F2017%2F12%2F08%2Fc17-feature-removals-and-deprecations%2F

Download-Seite der THM für Visual Studio Community 2017:
https://e5.onthehub.com/WebStore/OfferingDetails.aspx?o=18e65c13-7a03-e711-9427-b8ca3a5db7a1&ws=d26d7a8e-769b-e011-969d-0030487d8897&vsro=8

Erste Schritte mit C++ in Visual Studio:
https://docs.microsoft.com/de-de/visualstudio/ide/getting-started-with-cpp-in-visual-studio

Für die hier geschilderte Anwendung zu installierender Workload:

- Desktopentwicklung mit C++

  (Für Windows 7 oder 8 unter „Zusammenfassung“ nicht „Windows 10 SDK (10.0.16299.0) für Desktop C++“, sondern „Windows 8.1 SDK und UCRT SDK“, ggf. auch „Windows XP-Unterstützung für C++“ wählen!)

- Für Linux-Anwendungen auch „Linux Entwicklung mit C++“ wählen

Anmerkung:
Die unten aufgeführte Vorgehensweise mit VS ist so aufgezeichnet, daß die Eingaben aus der elektronischen Form dieses Dokumentes direkt in die Entwicklungsumgebung kopiert werden können.

Leeres Projekt erstellen:	Datei ⇒ Neu ⇒ Projekt ⇒ Installiert: Visual C++ ⇒ Leeres Projekt ⇒ Name: `Debug` ⇒ Speicherort: [~\MySw\] ⇒ Projektmappenname: `HelloWorld` (Häkchen bei „Projektmappenverzeichnis erstellen“): ⇒ OK
Verzeichnisstruktur erzeugen:	Neben dem erzeugten Ordner Debug hinzufügen: Exc, Src, tmp; in den Ordner Src Test.c (nicht: Test.cpp) und Test.h einer „Hello World“-Anwendung o.a. kopieren (ggf. Editor benutzen)
`main()` u.a. einfügen:	Im Projektmappen-Explorer (Drag&Drop o. re. Maustaste) Headerdateien und Quelldateien einfügen bzw. Projekt ⇒ [Neues / Vorhandenes] Element hinzufügen ⇒ ... Zu erstellende Dateien: Datei ⇒ Neu ⇒ Datei ⇒ C++-Datei ...
Konfiguration Ein-/ Ausgabe:	(Nach Klick auf das Editor-Fenster) Projekt ⇒ Debug-Eigenschaften… ⇒ Konfigurationseigenschaften ⇒ Allgemein: ⇒ Windows SDK-Version: 10.0.17134.0 (für Win10) bzw. 8.1 (für frühere Windows-Versionen) ⇒ Ausgabeverzeichnis: `..\Exc\` ⇒ Zwischenverzeichnis: `..\tmp\` ⇒ Zielname: `Test`

Bibliotheken und/oder Obj-Dateien einbinden:	Projekt ⇒ Debug-Eigenschaften…
	⇒ Konfigurationseigenschaften ⇒ C/C++:
	⇒ Allgemein ⇒ Zusätzliche Includeverzeichnisse:
	`..\Src;..\..\_Data;..\..\_Libs;..\..\_Src`
	⇒ Linker
	⇒ Allgemein ⇒ Ausgabedatei: `..\Exc\Test.exe`
	⇒ Zusätzliche Bibliotheksverzeichnisse:
	`..\..\_Data;..\..\_Libs;..\..\_Src; ..\..\_Dat`
	⇒ Eingabe ⇒ Zusätzliche Abhängigkeiten (nur bei Bedarf!):
	`OpenGL32.Lib; GLU32.Lib; GLUT32.lib`
	Falls benötigt, auch:
	`mui.lib` ⇒ OK

Compilieren:	Erstellen ⇒ Debug neu erstellen

Achtung: Für die Nutzung von OpenGL mit GLUT sollte
- auf `C:\WINDOWS\system32` die Datei `opengl32.dll` und
- auf `C:\WINDOWS\system` die Datei `GLUT32.DLL` kopiert werden.

Die o.a. Dateien `OpenGL32.Lib, GLU32.Lib, GLUT32.lib` (und evtl. `mui.lib`) sollten in `_Libs` kopiert werden.

Zur **Weitergabe eines Projekts** können gelöscht werden: (1) der Inhalt von `tmp` ganz; (2) in `Exc` alles bis auf die exe-Datei; (3) in `Debug` alles bis auf `Debug.vcxproj`

Hinweise auf mögliche Irrtümer oder Mißverständnisse werden dankbar angenommen.

Die folgenden Einstellungen sind nur bei besonderem Bedarf interessant:

Verbleibende Fragen:	• Beim ersten Compilieren Frage, wo Debug.sln zu speichern ist (⇒ OK) • Bei jedem Start des entwickelten Programms (F5 o.ä.) Meldung, das Projekt sei veraltet (⇒ z.B. in der Header-Datei: `#pragma warning(disable : 4996)`)
Evtl. zu ändern:	Projekt ⇒ Debug-Eigenschaften… ⇒ Konfigurationseigenschaften ⇒ Linker ⇒ Allgemein ⇒ Inkrementelles Verknüpfen aktivieren: Nein

Optionale Zusätze:	
Multithreading erlauben:	Projekt ⇒ Debug-Eigenschaften … ⇒ Konfigurationseigenschaften ⇒ C/C++ ⇒ Codegenerierung ⇒ Laufzeitbibliothek: `Multithreaded-Debug (/MTd)` (Voreinstellung: `Multithreaded-Debug-DLL (/MDd)`)

Bei Bedarf Unterdrückung des „Console window“:
 	
– alternativ dazu im Code:
Problem unter Win 7:
	Projekt ⇒ Debug-Eigenschaften… ⇒
	Konfigurationseigenschaften ⇒ Linker
	⇒ Befehlszeile ⇒ Weitere Optionen:
	`/entry:"mainCRTStartup" /subsystem:windows`
	`#pragma comment( linker, "/subsystem:\"windows\" /entry:\"mainCRTStartup\"" )`
	Konsole-Fenster wird immer kurz eingeblendet

Änderung des Arbeitsverzeichnisses (meist unnötig):	Projekt ⇒ Debug-Eigenschaften… ⇒ Konfigurationseigenschaften ⇒ Debuggen ⇒ Arbeitsverzeichnis
Vermeidung des Verzeichnisses Debug\ipch:	Extras ⇒ Optionen ⇒Text-Editor ⇒ C/C++ ⇒ Erweitert ⇒ Immer Ausweichpfad verwenden: False
Seltenes Kompatibilitätsproblem:	Projekt ⇒ Debug-Eigenschaften … ⇒ Konfigurationseigenschaften ⇒ C/C++ ⇒ Codegenerierung: Überprüfen von kleinen Typen: Ja (/RTCc)

Für Windows-Versionen, die mit Unicode arbeiten (typisch für Win10) sollte beachtet werden, daß die „wide“ (2-Byte-)Version der „klassischen“ `_getch()`-Anweisung aufzurufen ist; dies kann mit folgender Direktive allgemein gelöst werden:

```
#define _getch() _getwch()
```

Weitere Hinweise von Microsoft:

https://blogs.msdn.microsoft.com/vcblog/2017/12/08/c17-feature-removals-and-deprecations/