Inhaltsverzeichnis

1. Kompilieren und Ausführen der `.cpp`-Quellcode-Dateien 1
   1.1 Windows 10: Anleitung zum Kompilieren und Ausführen mit dem MSVC-Compiler (Intel/x86 und ARM) 2
   1.2 macOS Monterey (12.2.1): Anleitung zum Kompilieren und Ausführen mit g++/Apple clang 4
   1.3 Ubuntu 18.04: Anleitung zum Kompilieren und Ausführen mit g++ 5
2. Code-Editoren 5
3. Versionsverwaltungs-Werkzeuge 5

> Bei Rückfragen stellen Sie diese gerne im Exercise Panel des Moodles.

Wir stellen Ihnen für alle Aufgaben entsprechende C++-Quellcode-Dateien (`.cpp`) bereit, in denen Sie an den relevanten Stellen (in der Regel mit `ToDo`-Kommentaren markiert) die fehlenden Implementierungen gemäß der Aufgabenstellungen hinzufügen sollen.

Beispiel:

```cpp
int fibonacci(int number)
{
    // ToDo: Exercise 1.c - count number of calculation steps

    // ToDo: Exercise 1.b - return 0 on bad arguments

    // ToDo: Exercise 1.b - retrieve nth fibonacci number iteratively

    return 0;
}
```

Die bereitgestellten C++-Quellcode-Dateien sind dabei so konzipiert, dass Sie auch ohne Hinzufügen Ihrer Lösungen bereits erfolgreich kompiliert und anschließend ausgeführt werden können – wobei das Ergebnis des Ausführens der Programme dann natürlich noch nicht den Anforderungen aus den Aufgabenstellungen genügt.

1. Kompilieren und Ausführen der `.cpp`-Quellcode-Dateien

Da es sich bei C++ um eine kompilierte Sprache handelt, benötigen Sie einen C++-Compiler, mit dem Sie die C++-Quellcode-Dateien (`.cpp`) in auf Ihrem jeweiligen System ausführbare Programme überführen können.

Für die verschiedenen Betriebssysteme (bspw. Windows, macOS, Linux/Ubuntu) und Systemarchitekturen (bspw. Intel/x86, ARM) gibt es jeweils mehrere C++-Compiler, die Sie verwenden können. Im Folgen-

den sind unsere jeweiligen Empfehlungen für die unterschiedlichen Betriebssystem-Systemarchitektur-Kombinationen aufgelistet:

Systemarchitektur	Betriebssystem	Empfohlener C++-Compiler
Intel/x86 oder ARM	Windows 10	MSVC
Intel/x86	macOS	g++ (AppleClang)
ARM (M1-Prozessor)	macOS	g++ (AppleClang)
Intel/x86 oder ARM	Ubuntu 18.04	g++

Im Folgenden wird die Installation und Verwendung der jeweils empfohlenen Compiler auf den genannten Betriebssystemen beschrieben.

1.1 Windows 10: Anleitung zum Kompilieren und Ausführen mit dem MSVC-Compiler (Intel/x86 und ARM)

> **Hinweis zur Wahl des Compilers:** Alternativ zum MSVC-Compiler können Sie natürlich auch bspw. über das Windows Subsystem for Linux (WSL) oder über MinGW den g++- oder clang-Compiler installieren und nutzen. Diese sollten Sie analog zur Beschreibung für Linux/Ubuntu bzw. macOS verwenden können.

1. Laden Sie sich auf https://visualstudio.microsoft.com/downloads/#build-tools-for-visual-studio-2022 die „Build Tools for Visual Studio 2022“ herunter und starten Sie den Installationsprozess. Wählen Sie unter *Desktop- und Mobilgeräte* „Desktopentwicklung mit C++“ aus. Stellen Sie sicher, dass unter *Installationsdetails* › *Optional* die erstgenannte Komponente („MSVC …“) ausgewählt ist. Für die Kompilierung auf ARM-Geräten ist außerdem die Komponente „Windows 10 SDK“ erforderlich. Die Installation sollte ca. 4,2 GB freien Speicherplatz erfordern und ca. 5 Minuten dauern. Wenn Sie nach der Installation dazu aufgefordert werden, Ihr System neu zu starten, kommen Sie dieser Aufforderung entsprechend nach.
    - Alternativ können Sie sich auch, wie auf https://docs.microsoft.com/de-de/cpp/build/walkthrough-compiling-a-native-cpp-program-on-the-command-line?view=msvc-170 beschrieben, die kostenlose Visual Studio Community-Edition als IDE installieren und den mitgelieferten MSVC-Compiler benutzen.
2. Öffnen Sie die nun installierte „Developer PowerShell for VS 2022“ über das Windows-Startmenü (Eingabe von „Developer PowerShell“ die Suchfunktion des geöffneten Startmenüs sollte genügen).

- Alternativ können Sie auch die installierte „Developer Command Prompt for VS 2022“ (also die klassische `cmd`-Eingabeaufforderung anstelle der PowerShell) verwenden.

3. Führen Sie in der geöffneten PowerShell/Eingabeaufforderung den `cl`-Befehl aus. Er sollte erfolgreich ausgeführt werden und Ihnen eine Eingabe ähnlich der folgenden anzeigen:

```
Microsoft (R) C/C++-Optimierungscompiler Version 19.31.31104 für
    x86
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.

Syntax: cl [ Option... ] Dateiname... [ /link Linkeroption... ]
```

4. Zum Kompilieren einer der `.cpp`-Dateien aus den Übungsaufgaben führen Sie den folgenden Befehl in der „Developer PowerShell/Command Prompt for VS 2022“ im Ordner der jeweiligen Datei aus:

```
cl /EHsc /std:c++17 .\<Name der .cpp-Datei>
```

   Beispiel:

```
cl /EHsc /std:c++17 .\fibonacci_iterative.cpp
```

   Der Befehl sollte zu einer Ausgabe wie der folgenden führen:

```
Microsoft (R) C/C++-Optimierungscompiler Version 19.31.31104 für
    x86
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.

fibonacci_iterative.cpp
Microsoft (R) Incremental Linker Version 14.31.31104.0
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:fibonacci_iterative.exe
fibonacci_iterative.obj
```

   Wie in der Ausgabe zu erkennen, wurde eine ausführbare `.exe`-Datei mit dem Namen der entsprechenden `.cpp`-Datei erzeugt.

   Diese ausführbare `.exe`-Datei kann dann in der PowerShell/Eingabeaufforderung (mit eventuellen Parametern) ausgeführt werden:

```
.\fibonacci_iterative.exe 4
4 : 3 : #2
```

> **Hinweis zu ARM-basierten Geräten:** Über das beschriebene Vorgehen werden sowohl für die Kompilierung als auch fürs Ausführen der Programme der x86-basierte MSVC-Compiler und das erstellte, ausführbare Programm emuliert (und nicht für die Ziel-Systemarchitektur ARM kompiliert).

> Dies kann negative Einflüsse auf die Laufzeit-Leistung der Programme haben.

> **Hinweis zu C++-Standards:** Für C++ existieren unterschiedliche standardisierte Versionen, womit seit 1998 mehrmals Modernisierungen vorgenommen wurden. Wir nutzen für die Übungsaufgaben C++17. Falls Sie im Rahmen der Bearbeitung einer Übungsaufgabe Features eines neueren C++-Standards verwenden sollten, können Sie das entsprechende Compiler-Flag anpassen (z. B. `/std:c++20` statt `/std:c++17`). Fügen Sie in diesem Fall für den Korrektor einen Hinweis in Form einer Readme-Datei hinzu, mit welchem C++-Standard in der Übung gearbeitet wurde.

1.2 macOS Monterey (12.2.1): Anleitung zum Kompilieren und Ausführen mit g++/Apple clang

1. Öffnen Sie die Terminal-App oder einen anderen Terminal-Emulator wie bspw. iTerm2 und wechseln Sie in den Ordner, in dem die `.cpp`-Quellcodedatei(en) liegen, die Sie kompilieren und ausführen möchten.
2. Führen Sie den folgenden Befehl zum Kompilieren der jeweiligen Datei aus:

   ```
   g++ <Name der .cpp-Datei> -o <Name des zu erstellenden, ausführbaren Programms>
   ```

   Beispiel:

   ```
   g++ fibonacci_iterative.cpp -o fibonacci_iterative
   ```

   Wenn das Kompilieren erfolgreich ausgeführt wurde, erfolgt eine leere/keine Ausgabe – andernfalls werden entsprechende Fehlermeldungen ausgegeben. Sollten Fehler auftreten, stellen Sie zunächst sicher, den notwendigen C++-Standard zu aktivieren, bspw. für C++17:

   ```
   g++ --std=c++17 <Name der .cpp-Datei>
   ```

   Die resultierende, ausführbare Datei können Sie dann im Terminal entsprechend ausführen:

   ```
   ./fibonacci_iterative 4
   4 : 3 : #2
   ```

> **Hinweis zu C++-Standards:** Für C++ existieren unterschiedliche standardisierte Versionen, womit seit 1998 mehrmals Modernisierungen vorgenommen wurden. Wir nutzen für die Übungsaufgaben C++17. Beim Kompilieren der Programme mit `g++` müssen Sie darauf achten, den zu verwendenden C++-Standard mittels der `-std=<C++-Standard>`-Option (für C++17 also `-std=c++17`) entsprechend zu setzen. Falls Sie im Rahmen der Bearbeitung einer Übungsaufgabe Features eines neueren C++-Standards verwenden sollten, können Sie das entsprechende Compiler-Flag anpassen (z. B. `/std:c++20` statt `/std:c++17`). Fügen Sie in diesem Fall für den Korrektor einen

> Hinweis in Form einer Readme-Datei hinzu, mit welchem C++-Standard in der Übung gearbeitet wurde.

1.3 Ubuntu 18.04: Anleitung zum Kompilieren und Ausführen mit g++

Das Kompilieren und Ausführen der C++-Quellcode-Dateien unter Ubuntu oder anderen Unixoiden Systemen sollte bei Verwendung des g++-Compilers analog zu den o. g. Schritten unter macOS funktionieren.

Sollte auf Ihrem System kein g++ installiert sein, können Sie den Compiler bspw. unter Ubuntu über Ausführung des folgenden Befehls im Terminal installieren:

```
1 sudo apt install build-essential
```

2. Code-Editoren

Wir empfehlen die Verwendung von Visual Studio Code, wobei jeder andere Text-Editor (wie bspw. Sublime Text oder Atom) und einige *Integrated Development Environments* (IDEs) (wie bspw. IntelliJ CLion oder Visual Studio) ebenfalls geeignet sind.

Für Visual Studio Code empfiehlt sich die Verwendung des C/C++ Extension Pack für entsprechendes Syntax-Highlighting und IntelliSense-Auto-Vervollständigungen.

3. Versionsverwaltungs-Werkzeuge

Um einen besseren Überblick über gemachte Änderungen an den Übungsaufgaben behalten zu können und unterschiedliche Lösungsvarianten einfacher ausprobieren zu können, empfehlen wir die Verwendung der Git-Versionskontrolle.

Falls Sie mit der Verwendung noch nicht aus anderen Lehrveranstaltungen vertraut sind, können Sie auf eine der zahlreichen Anleitungen im Internet zurückkommen (bspw. unter https://git-scm.com/docs/gittutorial) oder Mitglieder des Teaching-Teams um Hilfe bitten.